"Game Manual 0"

a guide for FTC teams

enjoy!

Published on : September, 2019

Last updated : March, 2024

Game Manual 0 Table of Contents

Start Here

Are you a rookie team? Are you new to FTC®? Are you just looking for a refresher? Check out this section for a curated list of the most pertinent pages!

1

Mission Statement

Hello! Welcome and thank you for taking the time to read Game Manual 0, the premier resource for new and upcoming *FIRST*® Tech Challenge teams! If you are a rookie or a new team, we would especially like to welcome you to the FTC® community. We want to make your beginning steps and transition into FTC as seamless as possible, and offer some of our own experience and advice compiled over the years.

The goal of Game Manual 0 is to create a comprehensive guide for FTC teams. Physical resources for FTC are few and far between, and online resources are scattered across the interwebs. Typically, the largest inhibitor for newer teams is the lack of knowledge base, as robotics is a relatively new STEM field. New teams also have not made connections to experienced teams who might mentor or offer advice throughout the season. Thus, they are left in the dark to figure things out, seemingly with no way out. While the journey of learning cannot be shortcut, Game Manual 0 seeks to address these shortcomings by providing a starter's guide to the hardware and software in *FIRST* Tech Challenge.

When perusing this guide, it is important to keep in mind the authors' perspective. Many of the teams who contributed in the writing of this guide are veteran championship level teams in the upper echelon of FTC. **This means that most of our recommendations are almost solely from the competitive advantage standpoint.** For example, teams can learn just as much from using a Tetrix kit compared to a goBILDA kit. Just because there is a competitive advantage to one does not translate into a learning disadvantage for the other. The guide has a plethora of useful information and knowledge (such as FTC and basic engineering principles) for all teams, including teams which are primarily trying to learn rather than compete. However, do take our recommendations with a grain of salt. We want to help all teams in FTC, but as our experience has been on the competitive side of the fence, certain parts of the guide may not be very applicable to some teams.

Furthermore, the guide was originally written with a specific audience in mind - young teams who have recently begun their journey in FTC. Our recommendations are generally geared towards more inexperienced teams. However, in the past couple years we have expanded the guide to provide more in-depth resources around more complex topics. In doing so, we hope that Game Manual 0 will be a useful resource to rookie and veteran teams alike.

Before diving in, a short disclaimer: **this guide is not "How to Build a Championship-Winning Robot."** The purpose of Game Manual 0 is simply to provide knowledge, advice, and tips on how to get started.

Note: It is of the utmost importance that all teams learn the proper way (by following the engineering design process), not by reading a step-by-step guide or instruction manual. Thus, while Game Manual 0 has plenty of advice, we do not have specific steps included. Good luck, and have fun in FTC!

— Kleptomaniac

2 Start Here

Useful Resources

A list of useful resources and accompanying descriptions, with especially pertinent ones marked.

(page 369) Being a Team

Covers how to start, organize, and run an effective team.

(page 9)

2.1 Hardware

Design Strategy

A list of general tips for your team's design process.

(page 25) Design Tradeoffs

Goes over some common robot design decisions and typically superior alternatives to them.

(page 87) Drivetrains

Because it doesn't matter how good the rest of your mechanisms are if the robot cannot get to where it needs to be (except maybe if you're 8813 in Relic Recovery).

(page 90) Active Intake

Because you want to be able to pick stuff up. Touch it, own it.

(page 170) Linear Motion Guide

Moving things further than the constraints of the robot is an extremely common task, and linear motion is often one of the best ways to do this.

(page 129) Power Transmission

Different ways to get rotation from point A to point B.

(page 110)

2.2 Programming/Control System

Options for Programming

Because without programming, the robot is just an expensive brick.

(page 253) Control Systems

Because without a control system, the robot is just a (significantly less) expensive brick.

(page 219) Wiring Guide

Because disconnects are no fun.

(page 211) Programming Tutorial - Mecanum Drivetrain

Because mecanum wheels without the right code are some very expensive questionable quality traction wheels.

(page 280) Computer Vision

Explains the different common computer vision systems and how to use them (hopefully this doesn't end with a robot uprising).

(page 321)

3

Know Your Lingo

Many teams may be unfamiliar with the various terms and slang that are often thrown around at competitions and meets. Here we have compiled a guide of some commonly used terms to get you up to speed.

3.1 Competitions

Program Delivery Partner (PDP) Director or directors for events and activities in a region. Used to be referred to as the Affiliate Partner (AP).

Qualifier Tournament with 5-6 qualifying matches and best-of-three alliance elimination matches. Judging included. Will advance to States/Regional level.

League Meet Low-stress competition with matches counting towards league rank at a League Tournament. No judging.

League Tournament Tournament with ranked matches and alliance elimination matches for teams participating in that League. Judging included. Will advance to States/Regional level. Sometimes called a "League Championship".

Alliance Group of two or, in eliminations, two or three teams that compete. In elimination matches each team must play at least once.

Ranking Points (RP) Primary basis in rankings at traditional events. For the Power Play season, each team receives 2 RP for winning a qualification match, 1 for tieing, and 0 for losing.

TieBreaker Points (TBP) Secondary determinant in team rankings for traditional events, primary determinant for remote events. For the Power Play season, this is split up into two parts; TBP1 and TBP2. TBP1 is used before TBP2, and is the alliances/teams autonomous period score. TBP2 is the alliances/teams end game score. (Whether it is alliances/teams depends on if the team is at traditional events, respectively.)

Qualification Match Qualifying matches (generally 5) with random alliance partners and opponents will be played to determine team rankings.

Elimination Match 2 alliances of 2 or 3 teams each compete in best of 3 matches to determine winning alliance. Also known as elim matches, or just elims.

3.2 Robot Components

DC Motor Open-loop *RS-555 series* 12 volt motors used to power mechanisms. Max of 8.

Servo Closed-loop *servo-motors* used for precise movement. Controlled by PWM signals. Max of 12.

Drivetrain (DT) Mechanism responsible for the movement of the robot.

Intake Mechanism responsible for picking up game elements from the field by using rotational motion.

Claw Mechanism responsible for picking up game elements by grasping.

Linear Extension/Slide Mechanism responsible for extending parts beyond the starting configuration of the robot in a linear fashion.

Arm Mechanism responsible for extending the reach of the robot through rotational motion or multi-axis movement.

REV Expansion Hub Hardware controller for the robot, connected to *Robot Controller* phone. Controls 4 motor and encoders, 6 servos, and various sensors via digital, analog, and I2C ports. Maximum of 2 hubs.

REV Control Hub Integrated Expansion Hub + *Robot Controller* phone. It serves many of the same purposes as the Expansion Hub but eliminates the need for a *Robot Controller* phone. 1 Expansion Hub may be used along side 1 Control Hub.

Driver Station (DS) Android phone connected to the gamepad controller. Drivers interact with the DS phone to start/stop robot.

Robot Controller Android phone connected to the Expansion Hub. Connects to DS phone via WiFi-Direct.

Gamepad Controller An Xbox or PS4 style controller which the driver uses to control the robot during driver-control period. Maximum of 2.

4 Being a Team

Some tips on how to start, run, and organize a team, along with some outreach/marketing tips.

4.1 Starting a Team

So you want to start an FTC® team - here's how! Here's a few tips on how to get up and running. This article covers recruiting members and mentors, structuring your team, and pitching your budget.

4.1.1 Reach out locally to recruit members.

Chances are, if you're interested in robotics, you'll find quite a few people in your area interested in the same thing! Recruiting them is all about spreading your message and sparking interest in your team, and for that, advertising is everything. Make a catchy flyer and hold a couple interest meetings! A handy tip to get people to attend is to bring food - people love free food…

At these interest events, discuss how FTC is interesting and beneficial to those who participate. Make sure to have information for both parents and students! While you're recruiting, make sure to be inclusive - people may not have figured out that their passions lie within the scope of *FIRST*®, and they may do so in creative ways when given the opportunity!

Tip: To ensure a sustainable team, you should recruit from grades 7 upwards if possible to allow for a constant flow of members. The key to sustainability is new faces!

Make sure to get everyone's contact information so you can reach out to keep them interested.

4.1.2 Find mentors.

It's understandable if you have difficulty with this step, but it is critically important: good mentors can make the difference between a great season and a horrible one! Note the phrase mentors, rather than mentor. A strong structure of facilitators can assist you with the wide variety of the engineering and business problems you will be tackling throughout the season, while a single mentor can be a valuable asset but only has so much time and expertise to give.

Now, to find mentors: reaching out to STEM teachers in your area is a great way to find interest, but you can also reach out to local businesses to see if any of their employees would be interested in lending a hand! (See the email guide for more tips.) New recruits can ease this process a little, and if a new mentor knows a team member it'll make it easier for them to mesh with the rest of the team. Receiving mentorship over text and video is generally less recommended, as in-person mentors can communicate and demonstrate mechanical and software concepts more efficiently.

Important: To compete, teams are required to have at least two mentors over the age of 18. Additional mentors can be younger than 18. All mentors must pass the *FIRST* Youth Protection Screening, a once-per-year background check for mentors and volunteers. This is all handled by *FIRST*, and is free.

While parents and other family members can be helpful mentors in a pinch, it's important to have outside mentorship so that your team can survive even after you graduate.

Generally mentors fall into a few primary roles. Some mentors may be purely technical, offering advice on your robot and engineering documentation; some mentors focus on "soft skills" such as leadership, fundraising, marketing, and team logistics. Many mentors blur the lines between these roles, but some prefer to stick to one area or another.

Tip: Don't be afraid to involve mentors who aren't self proclaimed as highly "technical". They're sometimes the best ones.

4.1.3 Pick a comfortable structure for your team.

There are 3 common team structures, each with advantages and disadvantages:

School Team

Run as a school-affiliated team/club with school funding. Watch for red tape regarding funds and parts. Also watch for it on the field :)

Important: If starting a school team, make sure you figure out whether you can accept external sponsorships. This can affect your financial planning and fundraising.

Advantages:

- Easier to sustain
- Potential for consistent funding

Disadvantages:

- Less practice schedule freedom

- Arbitrary part sourcing restrictions
- Competing for funding yearly
- Limited/no summer practice

Home Team

Run out of a house or garage. Budget is whatever fundraising you can acquire.

Advantages:

- Freer practice schedule
- No part sourcing constraints

Disadvantages:

- Harder to obtain sponsorships
- Space/residential constraints

Community Team

Run out of a 501(c)3. Generally the hardest to set up, but offers the most flexibility.

Tip: If going the community team route, consider finding a local nonprofit to work with. This can greatly reduce the work and time required compared to starting your own, and many are happy to expand into STEM if you pitch it convincingly! Some common nonprofits include 4-H groups, Scout posts, homeschool groups, and libraries.

Advantages:

- Potential dedicated practice space
- Freedom of practice schedule
- No part sourcing constraints
- Lucrative for donors (through tax write-offs!)

Disadvantages:

- Harder to sustain
- Many administrative tasks

4.1.4 Find funding and create your team!

Ultimately, the way you fund your team is dependent on which team structure you use.

School Team

If you decide to run a school team, you should reach out to your superintendent, school board, and principal. Make sure to loop in a STEM teacher or two, and maybe even a curriculum administrator.

Home Team

For home teams, pitch to your parents or guardians! They'll probably be dealing with a majority of the headache anyways so be nice :)

Community Team

If you're pitching to a community nonprofit, make sure you highlight how *FIRST* will help them grow. Offer to volunteer your and your teammates' time to help the organization with their programs, especially STEM ones, to create a mutually beneficial relationship.

All 3 pitches should follow a similar structure; the easiest and most efficient involves a slide deck presentation. Make sure you highlight the value proposition (benefits that will be delivered) by starting an FTC team: educational experience, building future STEM professionals, increasing the potential of the area's technology economy, etc. You may not think it's important, but many decision-makers take these factors into account when deciding to approve your FTC team!

4.1.5 Don't underpitch your budget needs!

Here's a breakdown of a typical FTC season budget:

- *FIRST* Registration is $295, and allows you to obtain your team number and compete.
- Region and competition costs vary depending on your region, ranging from $250 to $800 or more. Check with your region's Program Delivery Partner for more information.
- A full competition game set will run you $450 / season.
- A competition field will also run you $659, and - unless you take care with the tiles that make it up - it will cost you additional money to replace every 2-3 years.
- The *FIRST* Control and Communication set will cost you $265, and their Electronics set will cost $282. Buying an additional Expansion Hub to maximize your robot's actuation will cost another $250. If you're careful, these are one-time purchases.
- The goBILDA FTC Starter Kit, which is recommended to start you with basic hardware, will run you $600 (with the FTC discount). If you CAD design your robot in advance, you can go cheaper.
- Extra money for sensors and servos - each can run you up to $100 each! Assuming 2 servos and 3 sensors, that's $500.

The above budget adds up to $4,000 in startup costs, and roughly $2,500 per year. You can definitely start a team on a budget much cheaper than this, but if you have the opportunity to pitch a larger budget, seize it!

Tip: If you are unable to achieve this amount of funding, you can make this budget more manageable through several methods:

- **Fundraising and sponsorship** are essential ways to ensure a team's survival. See the fundraising guide (coming soon) for more.

- **Several general cost-saving measures can be employed throughout the season. These include:**
  - Keeping stock of your inventory and not over-ordering parts.
  - Treating your parts well.
  - Using #black-market on the FTC Discord to obtain used parts.
  - Applying for seasonal FTC grants and the goBILDA FTC Discount.
  - Investing in a 3D printer to fabricate your own parts.
  - If using custom metal parts, utilizing SendCutSend or Fabworks, which are generally more financially viable than in-house CNC machining.
- As a last resort, **team fees** can be collected from members as a requirement to participate. These can bar less fortunate individuals from participating, so they are not recommended. **Minimum fundraising requirements** for your members can boost your budget similarly and without hardship for these members.

4.1.6 Now what?

You have your team, you have mentors, and you have a budget. Now it's time to be a team!

4.2 Team Organization

A group of individuals is nothing without structure and organization. Learning the individual strengths of each of your teammates to create robots together will turn you into a force fit to conquer even the toughest game challenges. This article covers team organization and collaboration, knowledge sharing, gracious professionalism, and fostering a safe team environment.

4.2.1 Team Structure

The first step to organizing a team is determining who does what. Generally, FTC® teams are organized into design, assembly / mechanical, software, and outreach squads, with different sub-squads for different tasks - for example, an outreach squad could have a sub-squad working on industry connections. In addition, your team will probably form cross-squad teams, such as your drive team and scouting team.

To effectively manage these squads, you will need to set a carefully chosen leadership structure - leads should possess good delegation skills and be willing to facilitate learning experiences for those assigned to their squad. Optionally, a captain can be chosen as well to oversee squad leads.

Tip: Each individual member should decide which squad they would like to participate in, as people work better on tasks they care about! In addition, each member should be involved with outreach in some form, to get them involved in the community and teach them the importance of service. In addition, this will improve your judged presentations.

Finally, your team will need to decide when it meets. Week to week, this will largely depend on the amount of work that needs to get done, but it is recommended to schedule 1-2 mandatory meetings a week to facilitate collaboration.

4.2.2 Obtaining Knowledge

Teams should strive to increase their knowledge of FTC design, software, and outreach both before and during the season. The more you know and experience, the easier it will be to tackle engineering challenges you encounter.

Practice over the summer.

Summer projects are a great way to explore robot concepts and start outreach. For instance, your team could research different lift designs and create a decision matrix of strengths and weaknesses. More often than not, the designs you research will make it onto your competition robot in the upcoming season - and regardless, the mechanical techniques you learn will be helpful. Reaching out to STEM businesses before the season will help you gain insights that could be helpful for the upcoming year, and planning and implementing outreach initiatives in your community will give you a headstart for the season.

Create and utilize resources.

To start the learning process, Game Manual 0 may be useful! Have your team peruse the resources here, and use that as a starting point for your own knowledge base. Make sure to check out the *Useful Resources* (page 369) section, where other resources are cataloged. Additionally, compiling your own resources will allow your team to explore different techniques and materials, and provide useful documentation for future team members. You could create a library of FTC legal sensors and goBILDA kits. Make sure to include all your members in these processes.

Cross-train your members.

This is the most important knowledge-related pointer, in my opinion. Having all your members learn at least one skill outside of their expertise will go a long way towards increasing collaboration and efficiency. Programmers who CAD will be able to design and integrate their own sensor cases, reducing design team workload. Designers who know outreach will be able to write their parts and speak to their designs more effectively in judging. Outreach people who know the mechanical and software processes on their robot are more effective.

Create a training program for your recruits.

When new members join your team, some of them will probably not know how to build and program a robot, or may have experiences with different techniques and softwares. Document a specific onboarding program for your new members - what logins will you have to give them? What softwares will they need to learn? What are some design techniques they will have to learn? Assign dedicated members to teach younger members about these techniques and to share their knowledge.

4.2.3 Gracious Professionalism

Teams should be gracious and professional both at practice and events - stress to your members that each of them is a representative of your team. Encourage them to actively assist other teams both at events and online. Finally, keep an eye on your team and ensure everyone is getting along well and being respectful.

4.2.4 Fostering a Safe Environment

Your team should strive to be as inclusive as possible. Ensure other members are not making homophobic, sexist, racist and/or transphobic jokes, and educate them on why they should refrain from doing so. Be actively mindful to not create a "boys club" culture where women are objectified and sidelined, and make sure every member feels comfortable in their position. Everyone should feel comfortable in your team environment.

4.2.5 Team Identity

Create a logo design for your team, and a corresponding style guide on fonts and colors to keep consistency when creating marketing materials. This will make it easier for you to design posters, flyers, t-shirts, stickers, and even branded plates for your robot!

4.3 Collaboration and Efficiency

FTC® is built on collaboration, and there are several ways to make collaborating as efficient as possible. Here's how!

4.3.1 Why Collaborate?

The benefit of collaboration is expanded creativity. Your team members aren't identical - each one has a different method of approaching problems. Combining these unique mindsets often leads to creative, efficient, and effective solutions to software and mechanical issues.

4.3.2 General Collaboration

Your team's leadership should communicate daily about the status of their squads - it's imperative to ensure smooth team operations. You can use platforms like Discord and Slack to facilitate organized conversations. In addition, collaborative platforms like Canva and Google Drive are ideal for hosting engineering documentation, judging presentations, and marketing materials. The more material your members can access, the more they can contribute to.

Ideally, every member will also be openly documenting the projects they work on. This way, if another member needs to take over or work to improve a project, they can utilize publicly available documentation to work effectively.

Tip: Host mandatory team meetings at least once a week, and share each squad's progress. This will give your team up-to-date knowledge about your robot and allow everyone to offer ideas at every stage of the design, programming, and outreach process.

4.3.3 Design Collaboration

Your designers probably have different styles of design and mechanical preferences, and their unique preferences can improve each of the different mechanisms on your robot.

Competitive Design

One way to maximize your designers' creativity is to competitively prototype. 2-3 designers pick a mechanism and develop their own take on the design, then physically prototype them to test against certain criteria. The most successful design integrates the most successful components of each and is iteratively improved by a singular designer.

The biggest benefit to this approach is that you can integrate aspects of each designers' personal techniques.

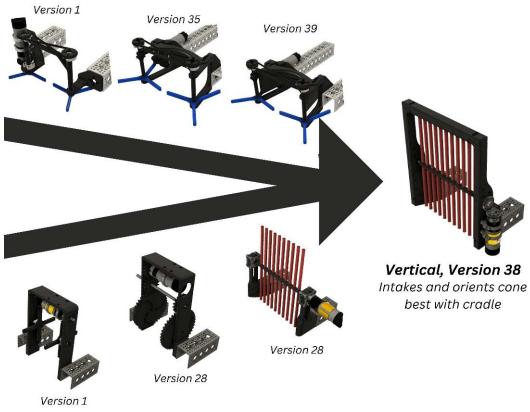


Fig. 1: 7149 Enforcers, PowerPlay, Example of Competitive Prototyping. Two of their designers built an intake, with the final optimized design utilizing the best aspects from each prototype.

Cloud CAD

Your design team (and the rest of the team, if possible) should have access to your full suite of designs. Sharing your designs allows for teammates to help each other improve by sharing techniques and tricks, and allows your outreach team to easily create renders for presentations and marketing.

This can be accomplished by using CAD software built around shared folders like Onshape or Fusion 360. It can also be accomplished with other CAD software, like Solidworks or Inventor, but this is generally significantly more complicated to set up.

4.3.4 Software Collaboration

Software collaboration is easier to facilitate. Robot code should be version controlled regardless of the size of your software squad, so teaching your members how to share a team Github is an easy way to collaborate. Additionally, you can utilize pair programming (multiple members writing software together on one computer) to help catch bugs.

4.3.5 Efficiency

Collaboration only works if exercised efficiently. Here are some tips!

- **Work from home.** Team meetings should be used as time to catch up between squads, make decisions, and physically test parts of the robot. CAD design iterations can be created and reviewed at home, and software can be written outside of practice hours. This will allow you more time to test and reflect.
- **Delegate tasks effectively.** Squad leads should serve as facilitators, assigning tasks to squad members depending on their ability. If work is hoarded by one or two people, the overall workflow is slowed down immensely. Trust your teammates.
- **Follow deadlines and don't procrastinate.** Setting deadlines allows you to plan your season, and the earlier you start before a deadline, the more time you have to ask for help, tackle unexpected problems, and iterate more.
- **Share.** If a team member has to search for documentation you were supposed to complete or an email you were supposed to send out, it creates a massive amount of administrative headache. Make sure you CC the correct people on emails and share your work with your entire team.
- **Don't hide problems.** If you broke something or calculated something incorrectly, don't shove it under the rug. Your teammates should (and will) be incredibly understanding and will help to fix the problem. Being correct all the time is impossible, and everyone makes mistakes!
- **Communicate often** with your squad and your team. Communicating is incredibly important on a team, as you can get fast feedback on problems, spark new ideas, and leverage the creativity of everyone on your team.

4.4 Outreach Basics

A guide to starting your team's portfolio of outreach events.

4.4.1 What Do You Care About?

A concerning amount of teams see outreach as a means to obtain awards. While it is true that the Motivate and Connect Awards rely on outreach performance as award criteria, outreach is so much more than that, and as FTC® teams, we are poised to make a unique impact. In order to make the most of your potential, you should target outreach that you care about.

Tip: From a judging perspective, it is also easier to talk about topics you care deeply about and are invested in.

Many teams often hope to create large-scale programs that impact tens of thousands. However, many smaller outreach programs can often be more impactful than one big, unwieldy program that may never get off the ground. Consider what your local area needs - maybe your library needs volunteers to teach STEM, or your local FLL team requires mentorship. Maybe there are no other FLL or FTC teams in your area. Regardless, your goal should be to advocate for STEM and *FIRST*® in your area.

Important: Create a plan! Any set of outreaches should be outlined in a business plan, documenting how they serve as a way to expand your program's reach. This plan should also contain your team's vision, budget goals, and growth objectives.

4.4.2 Spreading The Word

Once you've zeroed in on the programs you'd like to create or contribute to, figure out who you need to contact and what you need to pitch. See *Pitching Your Team* (page 20) for more tips on how to write your pitch out.

Motivate

Let's say you've decided to go with hosting weekly STEM nights at your local library.

The Pitch: STEM night hosted by x members of local robotics team on x weeknight, for x weeks.

- Write an email to your library's administration and CC (carbon copy) a librarian you know, introducing your team and describing your idea. Stress that you will dedicate team resources and members to this idea and see it through for the timeframe you specify.
- Assign specific members to this program and loop them into the conversation as well.
- Work with the library to determine a time each week to host your STEM night.
- Finally, assigned members work together to write lesson plans in advance and create promotional materials to share on social media and with friends/family.

Fundraising

Your team has a fundraising idea to bag at a local supermarket.

The Pitch: Support local STEM students by allowing local robotics team to bag at your store!

- Visit your local supermarket and talk to a manager or customer service rep to politely introduce your team.
- Respectfully ask if they would be willing to let you bag for customers and set up tip jars.
- Obtain contact information of the person you spoke to, and write a follow-up email to them thanking them and confirming a date and time.
- Assign members to create promotional materials to share on social media and with friends/family.

Connect / Sponsorship

Outreach to create connections and obtain sponsorships are relatively similar. Send lots of emails to local companies asking for either help or sponsorship. You should try to tie the engineering problems you are encountering in the season to the specific company you are requesting aid from; this can make it easier for the company to justify donating their money or time.

A great way to find businesses to reach out to is through your local Chamber of Commerce (use Google to find their website), who generally will provide a large list of member businesses in the area. Try to send one email to every business on that list. In addition, target companies in the nearest metro area.

Tip: Promotional materials should be striking and feature your team's branding, with an emphasis on succinct design. Your goal is to maximize information delivery using as few words as possible.

4.4.3 General Outreach Tips

- **Reach out for donations early.** Companies are most willing to donate money from September-November, when they are closing out the fiscal year. Reach out for sponsorships and donations around this time, as you will encounter more success.
- **Raise more money than you need.** You never know when you'll need to buy parts in an emergency, or qualify for the World Championship unexpectedly. Having extra money is never a bad thing.
- **Be consistent.** It is easy to let outreach fall by the wayside when building a robot. However, programs that survive long periods of time are defined by their consistent outreach, as it creates inroads into their community for recruitment and funding.
- **Be persistent.** If your idea doesn't work the first time or doesn't find the success you hoped, don't give up! Evaluate what went wrong and try again.
- **Diversify.** Try to do as many unique outreach events as possible, as it'll be fun and enriching!
- **Maintain your relationships.** Send thank you notes to organizations who host you and work with you, and keep them updated about your team's progress. It is easier to ask for a favor from an organization that likes your team.
- **(Respectful) email spam always works.** Send lots of emails. Your fundraising, Connect, and sponsorship email target should be 100+ a season - per category. The more emails you send, the higher the chance you will get a response.
- **Visit in person.** It is much harder to ignore an in-person visit you pay to a company than it is to ignore an email. Prepare a one-page flyer and elevator pitch, and remember you have a limited amount of time to make an impression on the person you're talking to.

4.4.4 Housekeeping

Keep a list of the programs your team has created or contributed to. Ensure you take photos at every event with team members. Keep track of your **Reach** (tangible interaction or observation of your team), **Engagement** (how many people interacted with your team/you interacted with, e.g. by attending events), **Impact** (how many people are directly impacted by your outreach events), and **Conversion** (how many people get involved with your program.)

If looking to include these statistics in your judged presentation or portfolio, ensure you adhere to the "Awards Definitions" located in the appendix of Game Manual Part 1 that outline terms relating to outreach.

These terms are: *Started*, *Mentored*, *Assisted*, *Provided Published Resources*, *Ran*, *Hosted*, *Reached*, and *Advocated*.

4.5 Pitching Your Team

Reaching out and getting attention is hard. Here's some tips on how to sell your team!

4.5.1 Types of Pitches

As part of outreach, you'll generally be pitching some form of idea to some person or organization. Here's a couple examples of different pitch structures:

Cold Call

This is probably the most common type of pitch you'll make. Cold calling involves reaching out to someone unprompted, and can take the form of an email or phone call.

"Startup"

Startup pitches are delivered when meeting with a potential connection or sponsor. Normally given as a slideshow, they introduce your team identity and unique goals, differentiate you from other teams, and tell a story of "why us?".

Elevator

Elevator pitches are "designed to be given on an elevator ride", as your "ask" is condensed into 30 seconds. You should use this when visiting local businesses or networking events to grab attention quickly.

Social Media

This is a social media post designed to promote something - be it your recruitment campaign, an event you're holding, or a fundraiser. Your pitch should be short and pop on whatever design you build around it.

Follow Up

Perhaps the most important type of pitch. A follow up can serve to highlight an email that was never responded to or confirm an event, sponsorship, or connection. You "secure the bag" with a follow up that is concise and echoes the main points of the original email.

4.5.2 The Art of Persuasion

Your pitch should utilize rhetorical and psychological techniques to convince your audience more effectively.

Rhetorical Techniques

There are 3 main rhetorical appeals: pathos, ethos, and logos. Pathos "tugs on the heartstrings" of someone's emotions. Ethos is the idea of "social credibility" and leverages a speaker's authority. Logos relates to ideas of logic, reasoning, and analysis. When creating a pitch, you will use all 3 of these appeals to persuade your listener effectively.

Here's a brief example of a fundraising email that utilizes these 3 appeals:

> Dear Company Name,
>
> We are *FIRST*® Tech Challenge Robotics Team #XXXX, representing Our School or Organization. Our team competes against 6,500 teams worldwide, and after a hard-fought season from September to now, was selected to represent our region at the *FIRST* World Championship and compete alongside 191 other teams (x% of teams worldwide)!
>
> However, we now have to raise $y for registration, transportation, and housing to successfully compete. As a community organization, we rely on the businesses and organizations around us to help us develop future STEM professionals and spread science and technology throughout our region. Even a $z donation would help us greatly in attending the championship, and we would love to feature your company's logo on our robot with a donation above $n.
>
> Thank you for your time! We hope you will consider contributing to this monumental opportunity.
>
> Our Team.

Here are examples of the various appeals in this letter.

Pathos

- "…we rely on the businesses and organizations around us" conveys the desperate need for funding.
- "…(x% of teams worldwide)" creates a heightened sense of importance for qualifying for worlds.
- "Hard-fought season from September to now" highlights the length of time your team dedicates to building robots and shows to recipients that your team means business!

Logos

- "Develop future STEM professionals and spread science and technology" appeals to the company's logical need for more employees in the future, or an individual's desire for the advancement of society.
- "Feature your company's logo on our robot" conveys a tangible marketing benefit for the company you are reaching out to.

Ethos

- "Develop future STEM professionals and spread science and technology" also appeals to ethos, as your team (direct recipients of these benefits) are speaking to the continued benefits that stem from this donation.

Psychological Techniques

Two major psychological techniques you can use to improve your pitch are the self-reference effect and the primacy-recency effect.

The **self-reference effect** is defined as "a tendency for people to encode information differently depending on whether they are implicated in the information" - which basically means that people remember information better if they believe they are connected in some way to it. If you find connections between your pitch and your contact, they will be more likely to remember you and your ideas!

The **primacy-recency effect** is "the observation that new information is most easily absorbed and retained at the beginning and end of a learning session". This means that information presented at the beginning and end of a pitch is the information that your audience is most likely to remember. To adapt for the primacy-recency effect, your presentations and pitches should contain its most powerful content at the beginning and end.

4.5.3 Be Concise!

All of the pitches listed at the beginning of this article rely on the principle of being concise. People that deal with tens, hundreds, or thousands of people asking for their money and aid generally have short attention spans. To stand out, you need to distill a lot of information and rhetoric into a few short sentences or slides.

Figure out how your team relates to the contact you are approaching, and develop your pitch around this relation; finding common ground is a great way to relate yourself to your contact and leverage the self-reference effect! You can do this by researching the company's core values and initiatives in advance. For instance, if a company is very invested in community STEM education programs, highlight that your team develops STEM professionals in a unique way. Once you've figured out what connects your team with your contact, create the content of your pitch around these ideas.

After developing your pitch, ensure that all information is relevant. Cutting content that doesn't connect with the company or organization you are presenting to is a surefire way to save time. If you choose to present a slideshow, keep your slides bare. An overloaded slide will draw attention away from your team, which is bad - people connect better with people than walls of text. Also, while reviewing your script or pitch, ensure each idea is stated only once. Overall, your goal is to create a clear picture of your team, impact, and connection in as few words as possible.

4.5.4 Putting It All Together

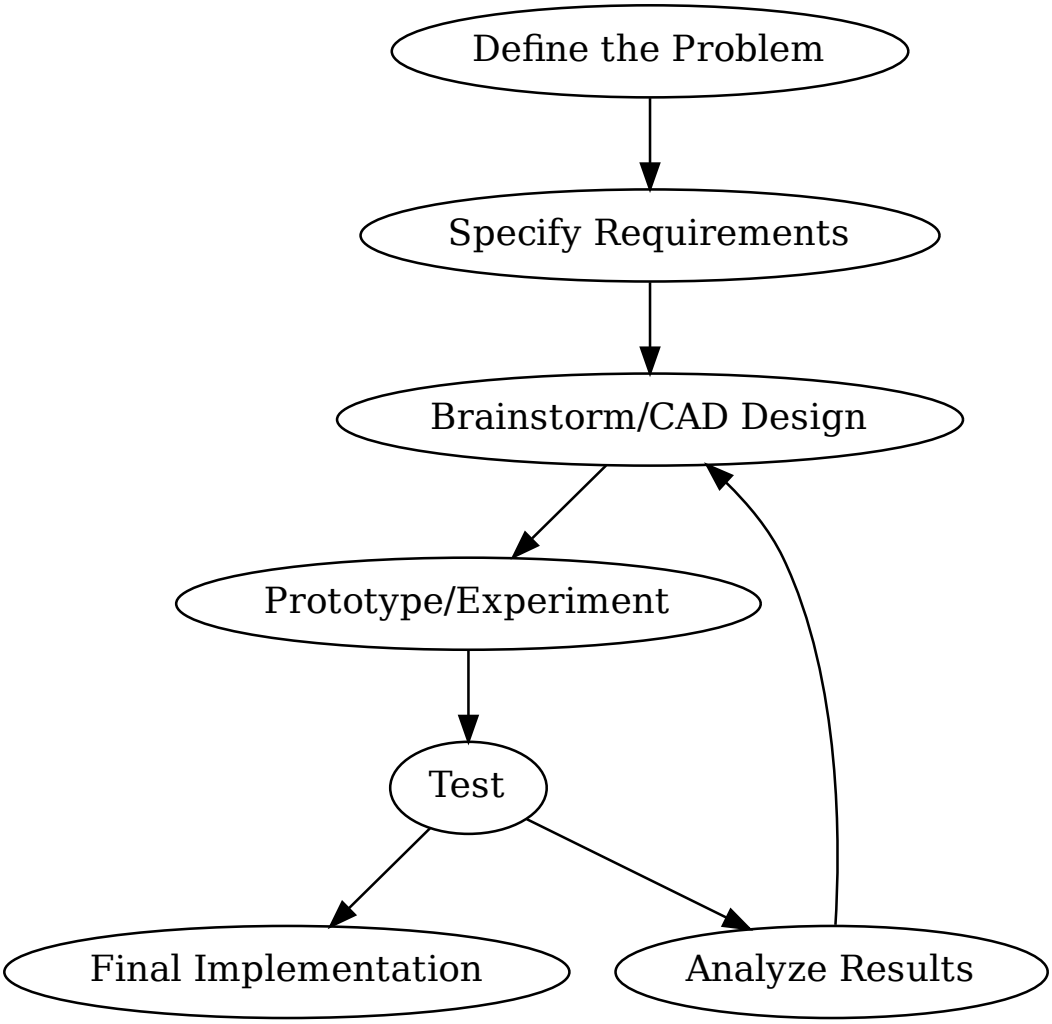
Now that you have these tips and techniques at your disposal, integrate them into your presentations, flyers, and networking events! With passion and hard work, you'll hopefully find your rate of success skyrocketing :)

5 Design Skills

This chapter covers basic principles of robot design and design skills that can assist you in building your robot.

5.1 Engineering Design Process

Note: The exact process will vary from team to team, so the graphic is not meant to be a hard and fast rule.



5.1.1 Components of the Engineering Design Process

In general, the design process contains these steps; however, every team will modify the process to their needs and limitations.

- Define the problem: What are you trying to solve? What is the timeline?
- Specify requirements: What are the necessities for your proposed solution?
- Brainstorm/CAD Design: Draw or sketch a prototype on paper or in CAD.
- Prototype/Experiment: Put a first design together using physical materials.
- Test: Ensure you thoroughly test every part of your prototype to detect flaws.
- Analyze results: Examine what you learned from your tests to iterate successively.

- Final implementation: Polish and solidify a final design that won't change.

As an example, let's look at an imaginary team building a drivetrain for Rover Ruckus (RR2), which was 2018-2019's game. In RR2, there was a crater, which was about 3" tall, and could be traversed. In that year's game, there were two main options: traverse said crater, or not traverse it and instead reach over with an arm.

The team must first specify requirements for the drivetrain. One of the most important aspects of a drivetrain is maneuverability. Another requirement could be speed, traction, reliability, etc. From the team's specified requirements, they would now look at drivetrains that fit their requirements. If the team wanted to traverse the crater, then a 4WD or 6WD would be an optimal design. If they did not need to traverse the crater, many options are still on the table such as the holonomic drivetrains.

The next step would be to brainstorm actual designs. It would be prudent to have more than one student to design one drivetrain prototype, so that the team could test more than one drivetrain.

From there, a team could begin the prototyping and testing process. This could involve tests of the time taken to cross the crater, time from crater to lander, top speed, maneuverability, etc. Simple tests may be conducted instead of complicated ones. For example, if the robot is projected to weigh 30 pounds, a dumbbell placed on top of the drivetrain simulates the extra weight well to see if the drivetrain can get over the crater easily and reliably, so the robot will not get stuck on the crater edge.

Next, analyze results and iterate. If, for example, there was too little clearance, and the drivetrain got stuck on the crater part of the time, then it would be necessary to raise the drivebase in order to remediate this problem. However, don't change too many things at once - **you want to change ONLY ONE variable at a time**, or else you won't know what may cause a further problem should it arise. Always try to change one variable at a time.

Note: It often takes multiple iterations to get things right, so don't be discouraged if your second or even third attempt doesn't perform as well as you think it will. Many teams have 10+ iterations of intake designs in order to refine and hone down their design to be optimally efficient. While that many tries isn't recommended for new teams, don't be afraid to modify *one factor* at a time to isolate and solve problems.

Your final implementation could be just an upgraded prototype, or, if you had used subpar/scrap materials for a prototype, you could change them out for durable ones.

5.2 Design Strategy

A useful resource is the championship conference presentation from Karthik Kanagasabapathy, former lead mentor (and current advisor) of Hall of Fame team FRC® 1114, Simbotics, on effective design and competition strategies: Effective FIRST Strategies.[1] See also the slideshow from the presentation[2].

[1] https://www.youtube.com/watch?v=5fifL47TvzE
[2] https://www.simbotics.org/_files/ugd/81d293_2417ace601d84fb5afaf62f424ad5bd3.pdf

5.2.1 General Design Tips and Mistakes

Here are some good tips for teams to make decisions in the high-level design and strategic planning stages.

Problem	Solution
Do everything at once • Robot becomes half-baked • Cannot excel in one area	**Perfect one objective first** • Robot is highly optimized • Consistently excels in one area
Overcomplicate • More time needed to iterate • Less reliable	**Simplify** • Best designs are usually simplest • Less moving parts
Score-first design • Neglect proper principles • Often wildly inconsistent	**Design for consistency** • Usually reliability > scoring ability • Great plus for alliance selection
Build haphazardly • Build with subpar materials • Inadequate support structure	**Build for reliability** • Remove unneeded moving parts • Eliminate single points of failure
Fret about design • Wastes testing time • Design alone is not enough	**Focus on execution** • Make a decision, then stick to it • Execution often beats design

Doing Everything At Once Perfecting One Objective First

Consistency is king.

Important: A common pitfall for first year teams is trying to accomplish all the game objectives at once, especially in tele-op and endgame.

This is highly discouraged because oftentimes new teams do not have the experience to do so. It is no small achievement to have a consistent robot that completes all objectives in competition, even at the higher levels.

Too often, we see teams bring half-baked robots that will attempt to do everything in a match, but excel at nothing. Even if they succeed, it is often by thin margins and cannot be repeated. This robot could be much more successful if the team spent their time to perfect one mechanism first.

Teams should always remember the principle that a robot that can complete one thing consistently will likely be more competitive than the robot that does everything inconsistently. We recommend teams focus on one objective during tele-op/endgame and perfect it.

Tip: Typically, teams which have a solid autonomous and consistent endgame can be competitive at the Qualifier level. This is a recommended goal for new teams.

Overcomplex Simple

Important: Another common trap that teams fall into is to overcomplicate needlessly. Simplifying your robot simplifies possible headaches later.

While some robots are very complicated, keep in mind that those teams are generally experienced, have some sort of machining capability, and fully design their robot in CAD. However, many world-class teams often build designs that are ingenious yet ridiculously simple.

Some advantages to simplicity are that the robot has less points of failure, given that the robot has less moving parts. Additionally, it takes much less time to iterate through and perfect a simple mechanism as opposed to a complicated one. The reasoning is that a complicated system has many more variables that need to be adjusted/could cause problems.

Keeping things simple can be practically achieved through a couple of ways.

1. Limit the degrees of motion that the mechanism operates in. For example, a linear slide goes in and out in a straight line, as opposed to an arm, which rotates along an axis. Doing so will serve to eliminate forces that otherwise could adversely affect the mechanism.
2. Another way to simplify is to build for the shortest travel distance. Obviously, the shortest distance from A to B is in a straight line, so teams should strive to keep the game elements approximately within a reasonably straight line. This can help in solving possible problems if the game elements need to change direction too many times.

Score-first Designing Designing for Consistency

Important: Teams should prioritize consistency over scoring ability.

The tortoise beats the rabbit. An overused parable, but it still holds a kernel of truth. Why? Because the tortoise, which plodded along consistently, beat the rabbit, which had hot and cold streaks.

A hallmark of any successful team is consistency and reliability throughout the competition season and even across seasons. Sports dynasties are dynasties for the reason that they compete at a high level not for a couple games, but for multiple seasons. Without the power of consistency, it will be nearly impossible to win games, let alone a tournament.

Too many teams fall into the pit of prioritizing scoring ability more than anything else, which is a grave error. In keeping with the first tip, to perfect one objective first, this practice will serve to increase consistency.

Important: While scoring ability should be a priority and objective when designing mechanisms, it is not everything in this game. We advise being consistent at low and medium scoring levels than inconsistently scoring at a high level.

Focus on being able to do that one thing every single time throughout your matches, and you will begin to see how important consistency is. **This tip is equally as important during alliance selections. Top teams will prioritize teams that are consistent far more than scoring ability.** They are not afraid to look at teams who can't score much, but can contribute every time to the alliance score, rather than selecting a boom-or-bust pick.

Building haphazardly Building for reliability

Important: Build for the worst case scenario, not the best case scenario. When building, teams often overlook a key principle: build for reliability. All too often, teams skimp on the quality of construction as well as materials, which leads to one of the most common reasons for unsuccessful tournaments: part failure.

Teams also do not take into account the rigors of competition and build as if the robot will not encounter opposing robots. Sufficient driver practice will be able to better simulate in-game conditions and test the reliability of the robot. To remedy this problem, refer to the *Materials Guide* (page 69) to gain a better understanding of what materials are recommended for use.

If possible, teams should build with redundancy in mind. For example, if one set of linear slides fails due to a wire snapping, having a second set will still allow the robot to operate instead of sitting dead in the water. Practically, doubling mechanisms, motors, and servos is a common method to build for redundancy.

In addition, teams often forget to account for twisting or compression forces that may occur upon the mechanism.

While we cannot give any specific recommendations, do keep in mind what forces the support structure of your mechanism must bear along the full range of motion, and account for what occurs when it might hit another robot/field wall/field. Building more robustly is always worth the time spent. However, it is good to think about the extra weight that results.

Furthermore, a common cause of robot disconnect is wiring issues. Refer to the *Wiring section* (page 211) for more information; in short, make sure to plan ahead and leave space for wires, and use strain relief whenever possible.

All these tips combined will help your robot become more reliable, a key characteristic of all world-level robots.

Fretting about Design Focusing on Execution

Tip: A good execution of a bad design will beat a bad execution of good design.

Important: FTC® is all about how well you execute in both the mechanical aspect and the driver aspect. If your goal is winning, then how mechanically beautiful your robot is doesn't matter. Your goal is less of impressing the judges but performing the best you possibly can on the field.

It is very possible to take a bad design, execute it well, and still be competitive at a high level. Even though not many teams are able to do so, it still goes to show that the method of implementation is very important. When brainstorming designs, try not to get hung up on small details if possible.

It is important to discuss different designs and debate the pros & cons, but after a design has been picked, stay with it unless there are major flaws that were originally overlooked. Changing designs will throw away the time spent on the original design, when teams could have kept improving it or practiced more. It is possible to rebuild your robot mid-season, and many top teams have done so to great success.

However, this is not recommended for rookie and new teams due to the general lack of experience. Realistically, expect to spend 50-100+ hours to rebuild a robot from the ground up. Focus on how you can iterate your current design to be as effective, efficient, and refined as possible.

5.2.2 Competition Tips and Mistakes

Here are some good tips for teams to make decisions in the execution stage.

Problem	Solution
Neglect drive practice • Drivers unfamiliar with robot • Robot reliability untested	**Constant driver training** • Drivers comfortable with controls • Proven and tested robot
No game strategy • Lack of contingency plans • Weakens alliance strength	**Strategical driving** • Only purposeful actions taken • Efficient and effective driving
Fully driver-controlled • Less efficient • Requires more practice	**Partially automated tasks** • Relieves stress on driver • Removes human error

Neglecting driver practice Constant driver training

Important: A persistent problem with new teams is neglecting driver practice. Drive practice is to be done throughout the season, **not the week before competition.**

No matter how good your robot is, the robot is only 50% of the equation. The driver(s) are the other 50% which determine the success of the team as a whole. Even if your robot is the best in the world, an inferior robot with a competent drive team more than likely will beat your robot with a poor drive team. By April's world championships, most top teams have run hundreds of practice matches.

This gives some obvious advantages compared to a team with lesser practice.

1. The driver(s) are totally familiar with handling the robot in every situation.
2. The robot is proven to be reliable enough to survive hours of operation.
3. The immense data that teams gather from test runs are used to optimize every element of the robot.

Driver practice not only familiarizes the driver(s) with the robot and serves as a test of robot reliability, it also simulates in-game conditions. **Learning to push the limits of your robot should be done during practice, not at a competition.** This way, drivers will become more comfortable driving under stress and pressure. Teams with intensive driver practice will purposely make things more difficult (such as placing a disabled robot in the middle of the field or unplugging a drivetrain motor).

While this may seem extreme, it is really just a form of preparedness. If your driver doesn't know how to react, then you need more drive practice.

No game strategy Strategic driving

Similar to drive practice, this is something that many inexperienced teams ignore. A sports example is handy - even with the most talented players, a team won't go far without good game strategy.

Important: A less capable team with better strategy execution can often pull off an upset. Planning a strategy ensures that every second in the 2:30 game time is used to maximum efficiency, which yields maximum points.

For example, drivers should know exactly where the robot needs to be positioned after the autonomous to tele-op switch. Practicing this switch will save a couple of seconds when drivers have to think "what do I do now?" In very competitive matches, these few seconds may be able to gain your team an extra cycle. Knowing when to transition from a tele-op to endgame objective is equally important (hint: perfect one first) and will save valuable time. Strategy should **always be used to maximize points** - whether this is a positioning strategy to access the game elements, or a defensive strategy to hinder the other alliance from scoring.

Tip: In most seasons, denying the other alliance 10 points is the same value as scoring 10 points in every match.

However, it is not advisable for rookie teams to play defense due to the specific rules surrounding this strategy. If a team wishes to execute a defensive strategy, be sure to read all the rules as defense can easily incur penalties/cards if done improperly.

Term

Defense Defense is a strategy employed with the goal of preventing the opposing alliance from scoring points, or at least significantly slowing the opposition's scoring.

This strategy can backfire if drivers illegally play defense and incur penalties and/or cards for their alliance. Defense is usually played by obstructing the opposing alliance, either by strategically positioning the robot to obstruct access or pushing another team's robot into a disadvantageous position.

Fully driver-controlled driving Partially automated tasks

Important: Autonomous should not be limited to only the autonomous mode. Automating simple tasks can be a real time-saver and efficiency boost to teams.

---

1. Automating tasks can save time and reduce the need for driver multi-tasking. Drivers should always be controlling the robot with as few button presses as possible. For example, automatically stopping the intake mechanism when game elements have been collected saves a button press.
2. Autonomously operating some mechanisms has the advantage of eliminating driver error and relieves stress. For example, if a lift has to extend to exactly 30 inches, a motor with an *encoder* can complete that with 100% accuracy at full speed, compared to a human driver's minor error.

Note: Autonomous functions should be able to be overrided by manual input in case something goes wrong (e.g. encoder is unplugged, a part breaks, etc.) to prevent damage to the robot and to be compliant with

game rules.

5.3 Computer-aided design (CAD)

5.3.1 Introduction

Term

Computer-aided design (CAD) CAD is software most commonly used to aid the design and drafting of parts and assemblies in engineering. In FTC®, CAD is used to make 3D models of robots as well as design custom parts.

CAD is not necessary in FTC to build a successful robot. Many successful teams didn't CAD their robot at all. Other successful teams only made parts of their robot in CAD. Still other teams fully designed their robot in CAD.

So, what's the point of CAD, then? It is encouraged that new teams try their hand at CAD, especially if you have a mentor or parent who is experienced in such an area. CAD is beneficial for multiple reasons.

1. CAD solves a lot of preventable headaches, such as spacing issues. Thus, it will save time when you discover problems in CAD that you can remediate before you build your robot.
2. CAD is a professional tool that is used in many STEM fields. Having CAD knowledge and skills will be beneficial in your future career, should you study and work in those fields.
3. If you desire to create 3D-printed or machined parts, CAD will be necessary to do so.
4. CAD can reduce the cost of building a robot by helping to determine which parts you need for a robot before actually spending the money to buy those parts.

However, CAD is not the magic genie that will guarantee you success in FTC. When used properly, it is a great tool to aid teams in building their robot. Keep in mind, though, that many teams have had success without CAD.

Starting off, choose a CAD program and learn it as well as possible. It may be a good idea to spend a few weeks just finding objects lying around, sit down with a ruler or calipers, and make a quick CAD model. A good way to test accuracy is to choose a solid object made up of one known material. Weigh the object and make the model in CAD, apply that material to it, and see what the weight difference is. This is a good way to test how accurate the CAD model is to the real part. It really doesn't matter what object it is - just find something and make it to the best of your abilities. There are also tons of videos on YouTube; a good one is TFI who makes detailed tutorials for Autodesk Inventor.

5.3.2 Overview of CAD Programs

There are many possible CAD programs that teams can learn, all of which can be downloaded for free under a student or FTC team license. You'll have to do a bit of research here, as the requirements for free copies vary based on the program. Here are a couple suggestions to consider:

Onshape[3]

A fully-featured CAD package, but it runs entirely in the cloud. It can run on any computer (even Chromebooks!) and has iOS, iPadOS, and Android apps as well.

It has all of the same core features as SolidWorks and Inventor, as well as the best collaboration workflow in the industry - many people call it "the Google Docs of CAD." Onshape allows multiple people to work on the same document at the same time, and allows users to "follow" each other and see what's on another user's screen.

OnShape also has FeatureScript, a programming language where you can write custom features. The community has created a lot of very useful FeatureScripts already which you can use completely for free.

Onshape also has a comprehensive tutorial system (https://learn.onshape.com) that will not only teach you how to use their software, but how to approach design problems.

If you're just starting out with CAD, or you don't have access to powerful computers, Onshape is the software for you.

SolidWorks[4]

An industry standard CAD package made by Dassault Systèmes. It's as fully featured as CAD software gets, including great simulation features and a very robust assembly environment. It's used widely in industry and is also the program of choice for most college-level engineering classes.

However, **it isn't available for Mac users**, and you'll need a pretty beefy computer to run it (16GB RAM is standard). SolidWorks also comes with a solid simulation program if you wish to test the structural properties of your robot or a custom-designed part.

If you have mentors or team members with previous experience in SolidWorks or an engineering class at your school that teaches SolidWorks, it will be your best choice.

Inventor[5]

Autodesk's industrial CAD offering. It offers many similar features to SolidWorks, but has a different UI and three distinct assembly modes.

While it's used by many companies in the industry, it doesn't appear in very many college curriculums. Inventor is generally the second choice for companies who don't use SolidWorks but instead are based around the Autodesk universe.

Inventor is also not available for Mac, but it may run better on lower-spec PCs.

If you have mentors or team members with previous experience in Inventor or an engineering class at your school that teaches Inventor, it will be your best choice.

[3] https://www.onshape.com/en/education/
[4] https://app.smartsheet.com/b/form/6762f6652a04487ca9786fcb06b84cb5
[5] https://www.autodesk.com/education/edu-software/overview?sorting=featured&page=1

Fusion 360[6]

A cloud-based all-in-one CAD/CAM package, also made by Autodesk. Fusion 360 is cross platform compatible, although it doesn't run very well on low-spec computers.

It has a powerful CAM environment for machining your parts, and it has intuitive and easy cloud rendering that gives the heavy lifting to Autodesk's servers. To maintain a simpler UI, Fusion skips out on a lot of the more advanced features found in SolidWorks and Inventor, although this isn't really much of a problem.

However, a more noteworthy difference is that Fusion ignores every single industry standard, creating its own structure and organizational system. Beyond simple sketches and extrude features, Fusion's modeling and assembly system is unique and not compatible with any other CAD software, making it hard to switch away from Fusion.

Because of this, if you aren't careful, Fusion's file hierarchy can actively encourage bad design habits and discourages reusability by allowing users to create new parts without designing them individually first.

If one is careful to follow good design practices, Fusion is a solid option.

Creo (formerly known as Pro/E)[7]

A family of CAD/CAM applications developed by PTC (Parametric Technology Corporation). Creo parametric is the main CAD package that includes robust assembly and part modeling similar to Solidworks.

The main advantage of Creo compared to other CAD software is the complex part relations and constraints, however, most new users find this aspect difficult to grasp completely. The Creo package includes an integrated local rendering engine and thorough simulation system. The rendering engine can be used to create photo-realistic renders of anything between single-part simple to multi-component complex designs. Design enhancements can be directly integrated into parts from materials/geometry simulations in Creo Simulate.

Creo has many features, but the general 80-20 rule applies - 20% of the features will create 80% of the designs. The Creo package also includes a version based file sharing system called Windchill which most professional companies use, but for FTC purposes Grabcad will suffice.

Learning Creo can be more of a challenge than other CAD software, because of the limited available tutorials online in addition to the complex relations and constraints structure. Our recommendation is to learn Creo from someone who already has experience with the software.

College-level engineering classes as well as numerous companies in the automotive, aerospace and consumer industry use Creo. Since Creo makes use of Windows OS file system it will not run on Mac. However, a Windows emulator can be installed to run Creo on Mac.

5.3.3 Getting Parts

All vendors (REV, goBILDA, Actobotics, AndyMark, Tetrix) provide 3d models of the parts they sell in STEP format, which can be imported by any of the CAD programs above. Some vendors also offer repositories or zip files containing *STEP file* of all the models they sell.

- Vendor CAD Libraries
  - Actobotics (ServoCity)[8]

[6] https://www.autodesk.com/education/edu-software/overview?sorting=featured&page=1
[7] https://www.ptc.com/en/products/education/free-software/standalone-educator
[8] https://www.servocity.com/step-files/

  - REV Robotics[9]
- 10650 Hazmat Robotics CAD Library[10]
- FTC Onshape Parts Library (contains a majority of parts from most major vendors)[11]

5.3.4 File sharing

Teams often have multiple members working on CAD models, and thus require a file sharing system for ensuring that each person has up-to-date files. Some recommendations include GrabCAD, Box, Google Drive, or Dropbox.

It is also a good idea that one team member work on the model at a time to prevent confusion.

5.3.5 Useful Resources

More CAD resources can be found in the *CAD section of the Useful Resources page* (page 370).

5.4 CAD Tutorials

5.4.1 CAD Tutorial Part 1 - Drivetrain in an Hour

Choosing the Drivetrain

After learning your *CAD* program of choice, determine the necessary requirements for the drivetrain based on the current game. Teams should shoot for the wheelbase that works the best in that specific field's layout.

For instance, in Relic Recovery (2017-2018) a drivetrain required precision to not only grab glyphs from the center pit, but also to line up against the cryptobox. Thus, mecanum wheels and a wide center section of the robot proved an advantage over a 6 wheel tank drive. (However, it should be noted that with sufficient practice and competent drivers, any drive base can be competitive up to a certain extent).

After selecting a drivebase, determine the number of motors. Keep in mind the eight motor limit is a pain that shouldn't be ignored. A good rule of thumb is four motors for driving and four motors for the other mechanisms (e.g. intakes, linear slides, arm, etc.) For most modern FTC® games, you need minimum 7 motors to be highly competitive, although 8 is a good rule of thumb.

[9] https://workbench.grabcad.com/workbench/projects/gcEvgrMnw6kRPx7OR6r45Gvb2t-iOdLiNG3m_ALpdGYzK_#/space/gcFd6nwp5Brrc3ks-92gagLZCV2FkceNTX3qGzaMvy2wQD/folder/2906404

[10] https://workbench.grabcad.com/workbench/projects/gcpgZgLBwhIdL0FfUKJJfM75cqa9RW1ncXaL-lQ4KOl1wa#/space/gcSzacmSeI-l19BYQNPm422pSHLenRxOxVtmaD-Pzynwsq/folder/6578524

[11] https://ftconshape.com/introduction-to-the-ftc-parts-library/

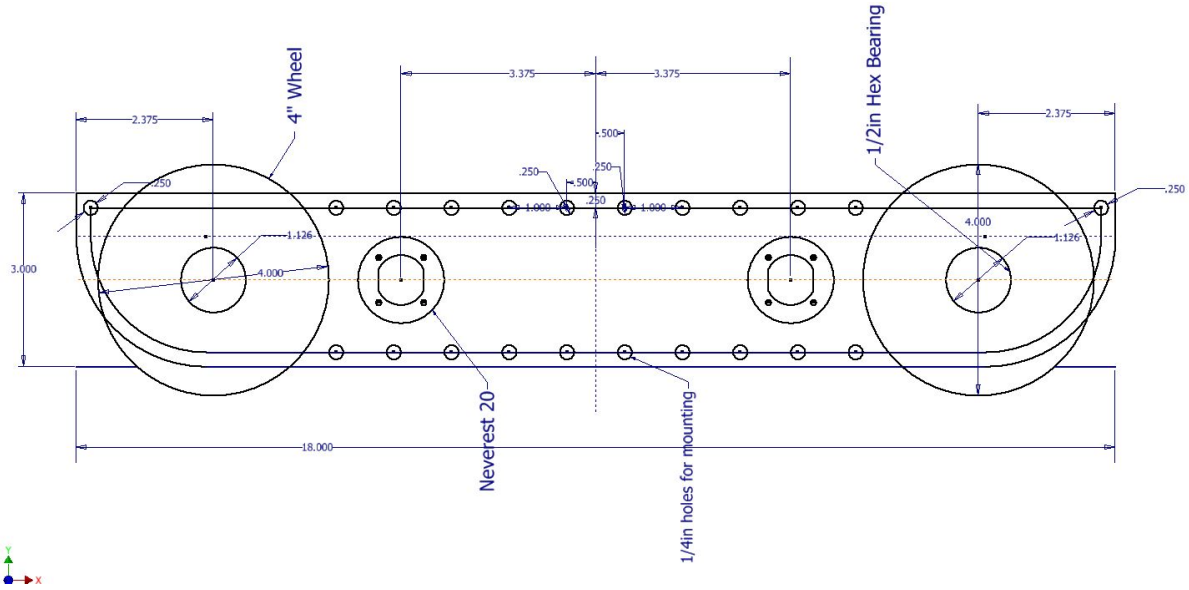
Designing the Drivetrain Plates

After learning the CAD software, it's time to start the actual design. Here are some things to figure out before starting:

- Drive Type (mecanum, 6wd, 8wd, etc.)
- Number of Motors (four motors recommended in most cases)
- Type of wheels (*Traction*, *omni*, etc.)
- Drive power (*belt*, *chain*, *gear*)

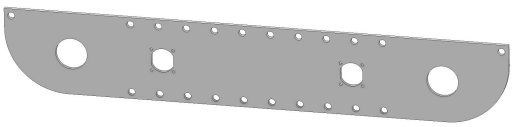
To keep it simple, this example uses a 4 wheel tank drive using four motors. The wheels selected are 2 Colson wheels for traction, and 2 omni wheels to aid in turning.

First, make the left side of the drivebase. After completing it, all you have to do is mirror the left side to the right, so you don't have to do each side individually. Start with a 2D sketch of everything before trying to extrude and make actual 3D objects.

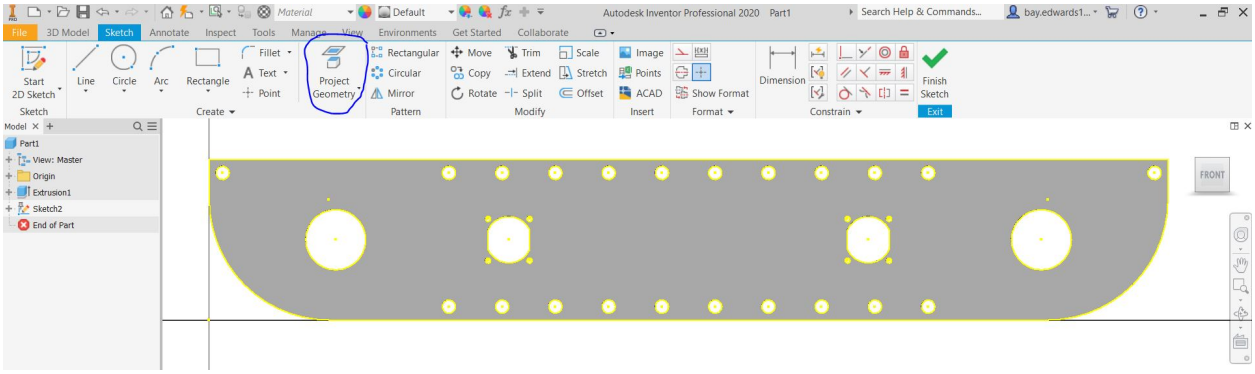


This is a sketch of the inner plate of the drive base. Everything should be laid out in a 2D sketch to determine the mounting holes, *bore*, *center-to-center distance*, etc. 2D sketches are extremely helpful and are highly recommended in any project. After the sketch is completed, everything else falls into place and becomes pretty simple.

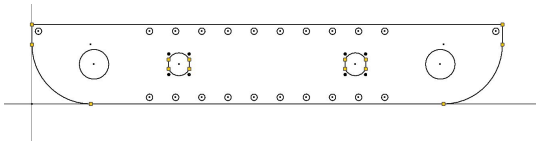
After this, extrude that sketch into the first plate of the drivetrain. Typically, a standard thickness of aluminum plate is 1/8". Thinner plate (3/32") can be used as well, but generally most teams stick to 1/8". Extrude the plate to that thickness. Below is the sketch after extruding.



The next step will be making the outer plate for the drivebase. It is even faster to do than the inner. To do this, simply create a new part. Go back to your inner plate and start a 2D sketch.



After starting the new sketch on the inner plate, hit “Project Geometry” and just click anywhere on the part. It should highlight every outline of the part. (Shown here is a yellow line; yours might be red, blue or some other color.) Now click and drag across the part selecting every line on the screen. Now go hit CTRL + C, then go to the new part and hit create 2D Sketch. Next hit CTRL + V.



It should look like an exact copy of the inner plate but now as a sketch. Delete your motor mounts out of the middle, then extrude the outer plate.

This is what the outer plate looks like, an almost exact copy of the inner one without the holes for the motors. Now with those two plates made, it’s really just time to assemble the rest of the drivetrain, which is by far the most time consuming. Now, for some info on what to use to attach the two plates together, generally standoffs or churro is highly recommended. To attach the two halves of the drivetrain, use either channel, extrusion, or a custom u-brace. Some teams prefer a custom brace as it is a good way to stiffen up the drivetrain while requiring very little maintenance over the season. It is possible to use peanut extrusion or kit channel, which alternatively works just as well.

Note that when using a custom drivetrain, you can cut out material from your drivetrain plates. This process is called **pocketing**. While not a vital step, pocketing helps you save weight. However, be careful not to remove too much material; if done, the plates become less sturdy. More about pocketing is in the next section.

Additional Considerations

Powering wheels can be done is a couple different ways through either belts and pulleys, chains and sprockets, gears, or even powered directly from the motor. Direct drive and chains are the simpler of the options, with direct drive not needing a calculated distance at all just have to set the motor exactly where the center of the wheel is. Chains allow for a little bit of slack not needing an exact center to center distance in the wrap like belts and pulleys do. Finally gears which need to be a certain distance apart from each other to mesh properly and not skip or bind.

Mounting motors is done in a plate style by face mounting the motor into the innermost drivetrain plate. It can also be done by mounting the motors to a 3rd plate, located in between the outside and inside. This

allows for the motor to take up less space in the middle of the robot, but adds complexity. Motors should always be as low as possible and depending where you want the center of mass, either the middle or towards the back of the robot. It is also worth keeping in mind the type of power transmission and the expediency of doing so in light of the motor placement.

Ground clearance is all dependent on if there are any obstacles on the field, as well as what your team wants to do in that game in regards to said obstacles.

For example, in Rover Ruckus some teams with tank drivetrains decided to enter the crater. Therefore, they left enough space to not beach themselves on top of the crater, a common mistake that inexperienced teams often make.

Other teams decided to ignore driving over the crater and decided to reach over with an arm or slide system, which meant they didn't need a lot of ground clearance for their drivebase.

Typically, anywhere from .25 inches of clearance to .5 inches (if you want to be safe) on a completely flat field will allow for the weight of the robot to push into the foam tiles. Nothing else from the robot should touch the ground.

Something you can do is set the robot in CAD onto a field. Set up obstacles such as the crater and simulate driving over the crater by moving it across like you think it would in the real world.

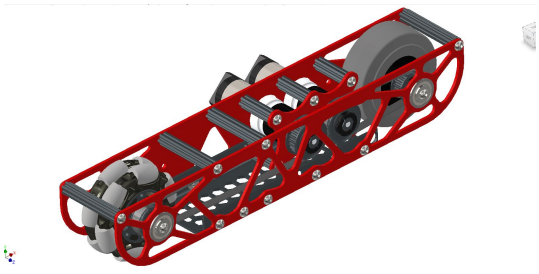
If either of the plates intersect with the obstacle, add some more clearance so you don't get beached like a sad whale.

A general rule of thumb for most teams is the wider the intake, the better the chance of picking up the game piece. However, this is super game dependent. If you need to pick up a 6" cube like in Relic Recovery then you would not need 14" of space for your intake.

However, if you need to pick up a ball like in Velocity Vortex, the bigger the intake gives you better chances of grabbing the balls. Keep this in mind when designing drive pods - try to keep them as thin as possible without sacrificing rigidity and strength to maximize space for other mechanisms and wiring.

Connecting your two plates together is really simple. Some standoffs or churro extrusion from AndyMark is a relatively easy way to connect them together with a few bolts. Just make a few 1/4 in. holes in your sketch where you want the churro tube to be. Decide how long the churro needs to be. Remember to leave enough space between the plates for your wheels, pulleys, sprockets, and spacers. You don't need to go overkill on how many standoffs you need in between your plates; however, put them in strategic places where support is needed.

Shown below is a drive pod, which is one half of the drivetrain, including the shafts, bearings, wheels, motors, belts, etc. In short, the drive pod has everything that will be built in real life. This particular one is the left side, but to make the right side create an offset plane, select the mirror tool, then hit mirror.



After mirroring the drive pod to make your opposite side, connect those two halves together and you're done with the drivetrain. Below is a rendering of the complete drivetrain in CAD.

5.4.2 CAD Tutorial Part 2 - Pocketing Guide

Term

Pocketing "Pocketing" is a common term in FTC and FRC® lingo that refers to cutting out excess material from a CAD designed part. Pocketing helps to reduce weight and can increase strength of a part. This may seem counterintuitive (how can removing material strengthen a part?) but pocketing can reduce stress buildup, especially at corners.

Pocketing is often seen on drivetrain sheet metal plates which will be CNC machined. In FRC, pocketing is often used to reduce weight of the rectangular aluminum tubes.

There are several ways to machine pockets into material including milling, routing, water jet cutting, laser cutting and even hand drilling. Depending on your access to tooling, pocketing can be more or less difficult for you.

CNC milling and routing excel at pocketing aluminum box tubing, whereas water jet and laser cutting excel at pocketing plates. Whether pocketing on box tubing or plates, the design is fairly similar.

When designing pockets, **it's important to consider the type of material, thickness, and how much stress will be on the part**. Materials that are weaker, thinner or under significant stress should have less "aggressive" pocketing and materials that are stronger, thicker or under less stress can have more "aggressive" pocketing. Aggressive pocketing refers to the amount of material removal from the blank part (more aggressive = more material removal).

Although a bit complex to understand, FEA (finite element analysis) can be used to determine appropriate strut thickness when pocketing. FEA can be used to generate pocketing geometry, but that is an entirely different rabbit hole.

Fig. 1: 731 Wannabee Strange, Rover Ruckus, FEA of inner drivetrain plate

Designing concise and advantageous pocketing is as simple as drawing circles and tangent lines. Parametric pockets can be defined by one or two offset values. The offset values determine the thickness of the remaining material.

Parametric means that the entire sketch is defined by a parameter, in this case is the offset value which when adjusted will automatically adjust the entire sketch (in terms of material thickness).

There are several references that can be drawn on every plate/tube which are screw holes, bearing holes, and corners. Each reference will get its own construction/sketch circle or two. Ideally all of the construction circles are one of less than 4 sizes to keep the pocketing consistent and simple.

First are the screw hole construction circles with radius of the screw hole radius plus the offset value. Next are bearing holes with radius of bearing hole radius plus offset value. Then are edges with construction circles with the radius of an offset value. Then the most important circles are at each of the screw and bearing holes, which will define the strut thickness.

The circles at the center of each screw and bearing hole will have the diameter of an offset value. After all of the construction circles are drawn, tangent lines can be drawn to create the pocketing geometry. Using the parametric offset value will make it easy to adjust strut thickness by just changing one or two values.

Tangent lines are drawn between the circles on the edges with other circles on edges and between the circles at the center of each bearing and screw hole. The circles with radius of bearing hole and screw hole plus offset value make sure that there is enough material around the bearing and screw holes. An example is below.

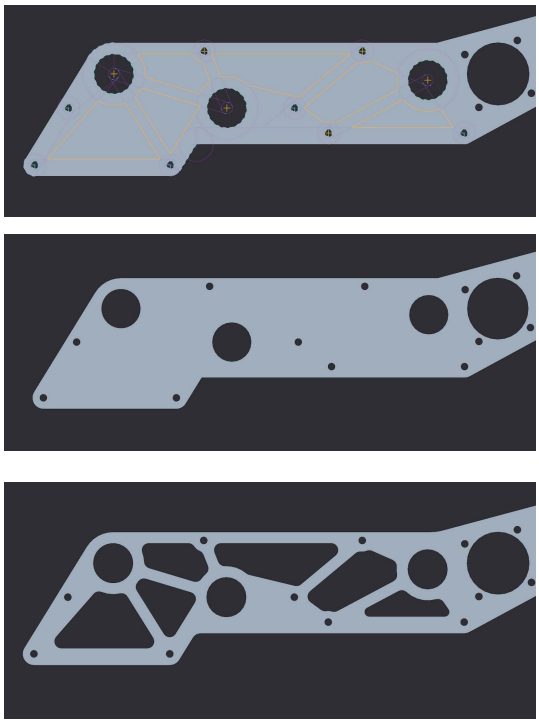


Fig. 2: 731 Wannabee Strange, 2019 Summer VCC Cadathon, Outer Mechanism Plate

The last step in pocketing is adding rounds to each and every corner, especially inner corners. Rounds relieve stress buildup at corners and make it easier to machine. Some machines, such as mills and routers, are also unable to machine tight internal corners. For those parts that need minimum rigidity loss and a lot more machine time on their hands, pockets don't need to be cut all the way.

Waterjet cutters and *laser cutters* are only able to cut material all the way through, but routers and mills are able to make surface pockets. These pockets don't go all the way through the material and are multitudes more rigid than thru pocketing.

The downside is increased machining time. The increased time is from the "lawn mowing" tool cutting path verses simply cutting the edges of the geometry. It is also more difficult to machine, because more material is milled out and chip ejection becomes more important.

If you don't have access to any precision tools, a hand drill/drill press and large drill bit/flat bottom boring bits

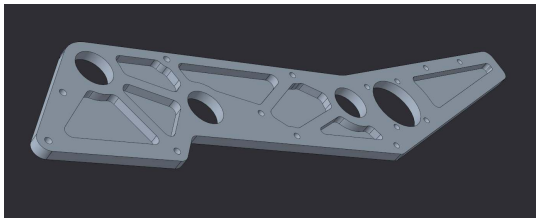


Fig. 3: Surface Pocketing Example

can create pockets in material. Although this is the simplest form of pocketing, there is a straightforward way to optimize the circular drill method.

Since the main goal of pocketing is to remove as much material as possible without significantly sacrificing the structural stability, the holes need to be drilled in specific positions with the right size bit.

The most effective way to find the specific positions and drill bit sizes, is to first create a pocketing design as you would do with circles and tangent lines. Then draw holes tangent to the struts created by the circles and tangent lines. An example is below with the orange as the holes to drill positioned tangentially to the regular pocketed edges.

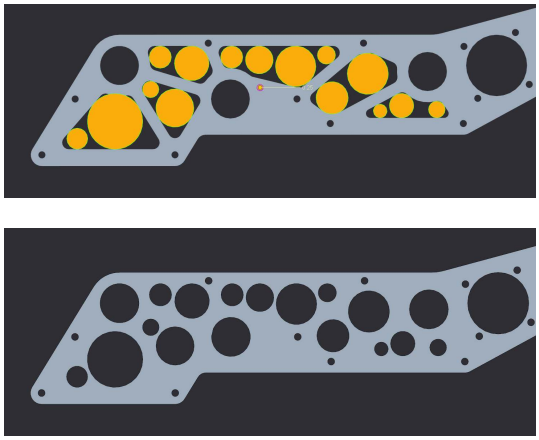


Fig. 4: Optimal Drill Pocketing Method Example

Although it may look like a random mess and it may take a while longer than just randomly "cheese holing", this method will yield the greatest weight reduction to structural rigidity loss ratio using the drill pocketing method.

A very important tip to pocketing is to do it last when designing a part. Parts should not be designed around the pocketing pattern, rather the pocketing should be designed around the part. If there are too many holes in a part, or the part is too small to be pocketed with an offset value, then it's probably not worth it to pocket.

Pocketing can reduce part weight, but when using traditional machining methods can take a significant amount of extra time. Although, when adding pockets to parts that are going to be 3D printed, it can in some cases decrease print time as well as material used.

The pocketing method above is the simplest parametric method to pocketing, but more complex methods exist. For instance, the image below is an example of a complex double iso-grid pocketing pattern optimized for metal 3D printing.

When the pockets are designed around a 3D printed part, many new possibilities open up in terms of minimum inner corner radius, resolution and dimensions. Now of course, 3D printed parts can be pocketed in the same way as traditional parts with similar results.

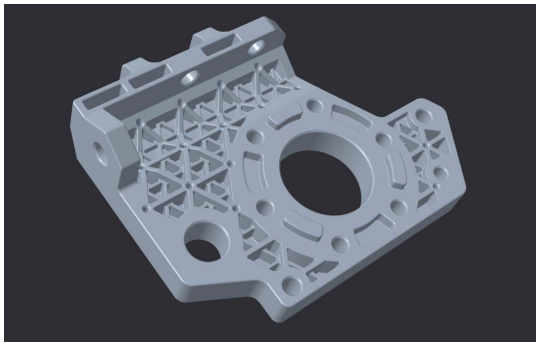
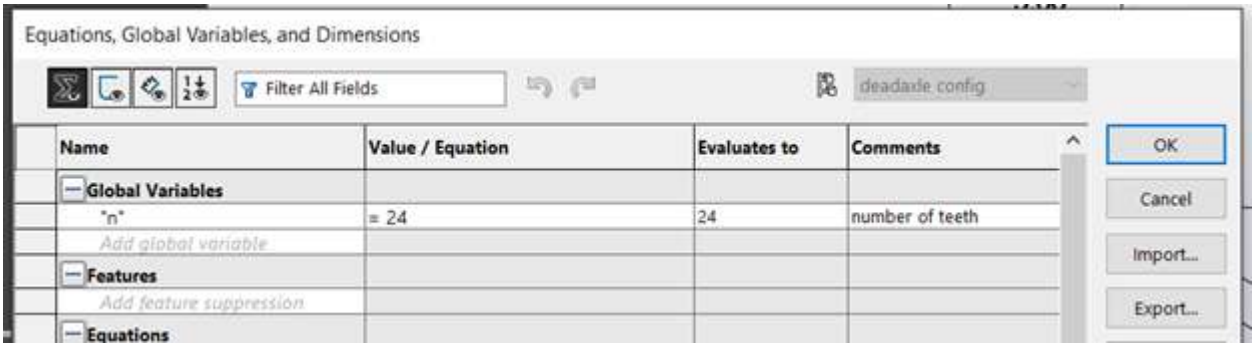


Fig. 5: 731 Wannabee Strange, Rover Ruckus, Arm Pivot Mount

5.4.3 CAD Tutorial Part 3 - Custom Pulley Template

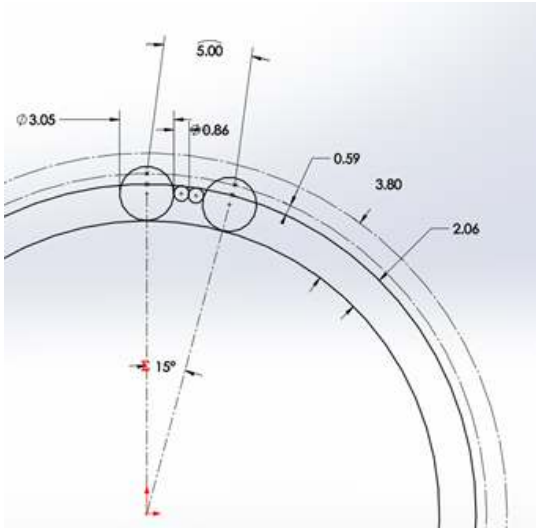
When designing methods of power transmission, it's useful to have an adjustable pulley generator to rapidly rearrange C-C (center to center) distance for design changes. Typically, FTC teams use the HTD5 belt profile due to its deep tooth profile, which adds resistance to slipping and increases load capacity. This tutorial will focus on the HTD5 profile, but it is relatively easy to adapt for different profiles.

To make the pulley fully parametric (adjustable without redoing the base sketch), we will use Equations (in Solidworks and Creo), Parameters (Fusion 360 and Inventor) or Variables (Onshape). Equations allow a user to quickly adjust values and change multiple dimensions in a sketch or feature.



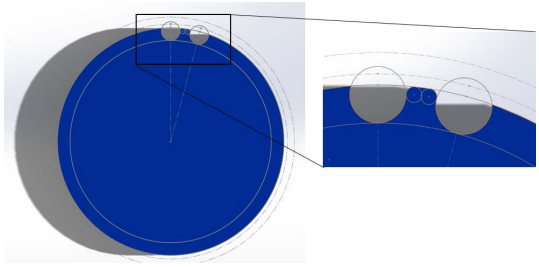
First, define a new variable "n" and set a default value of 24. This is crucial since "n" will affect the number of teeth, which will define the angle between teeth and the circular pattern.

Copy the sketch below.

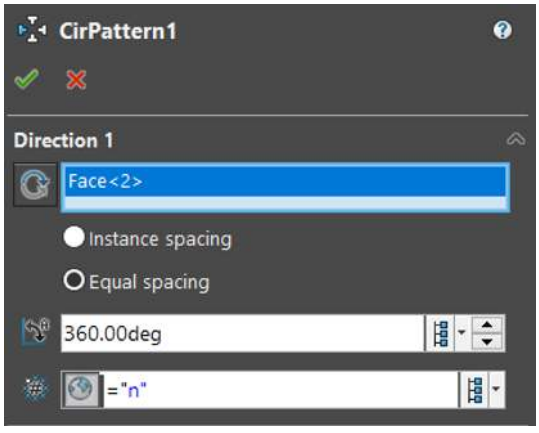


- The 15° equation is done by typing =360/”n” into the text box.
- Note that 5mm dimension at the top describes arc length, which is done in Solidworks by first selecting the two points and the connecting arc.
- The two big circles are tangent to the two smaller circles, but the two smaller circles are not tangent to each other.

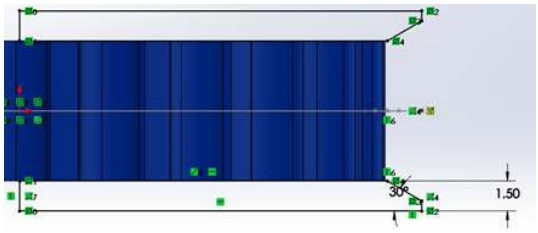
Leave this sketch as a reference and use “Convert Entities” to create sketches for additional features.



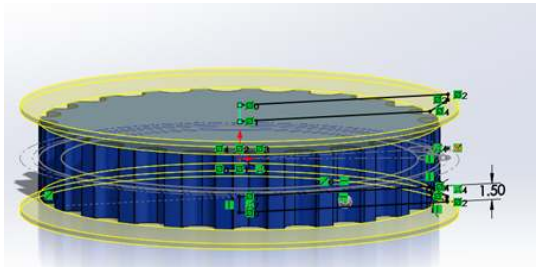
Next, extrude the outer bold circle. Cut-extrude the profile in the reference sketch. Do these features separately.



Now, just create a Circular Pattern. Define Direction 1 as the top face and create “n” instances of the cut-extrude feature.



Now just sketch on the side plane and sketch the flanges. This is up to you, but I prefer to keep the outer point vertical to a point pierced through the outer circle. That way, the flange changes with respect to “n”.



5.5 Design Glossary

COTS COTS (Commercial Off the Shelf) parts refer to parts that teams can purchase physically or through an online retailer.

> **Warning:** FTC® teams are limited to one degree of freedom (with some exceptions) to COTS parts. Therefore, buying a drawer slide is an allowable part, as there is only one degree of freedom, but purchasing a multi-axis arm isn’t.

However, teams can buy individual parts and assemble them together into a mechanism that has more than one degree of freedom. This doesn’t apply to drivetrain kits.

Packaging Packaging refers to the relative size and location of components on the robot. Generally, you want to design and locate (or package) components in the most space-efficient way you can.

STEP file A STEP file is a filetype used to store 3D data about a part. It is recognized by different CAD softwares including SolidWorks, Inventor, Creo, etc.

6 Hardware Components

This chapter covers the basics of building the robot: tools, materials, kits of parts.

6.1 Kit and Hardware Guide

Before beginning your season, your team is presented with a crucial decision - one that will undoubtedly heavily influence the team's direction of hardware design in at least the first year. The *FIRST*® storefront currently offers two kits for new teams: the TETRIX kit from Pitsco, and the REV Starter kit from REV Robotics. However, teams should carefully consider other options, such as goBILDA, before selecting a kit.

Every build system has advantages and disadvantages, which we have highlighted in our detailed build system guides below. While the guide may not be fully objective, recommendations are based on our own experience with the different kits, so the slant is there with reason. As with pretty much anything in FTC®, there is no one right answer - but there are better answers than others. Hopefully, this guide gives some solid advice on which kit might be the best for your team.

6.1.1 Why Use a Kit?

After all, plenty of successful robots have not followed this framework. However, we still recommend new teams purchase a starter kit for one big reason. Established teams are all but guaranteed to have spare parts lying around to use to build their next bot. However, rookie teams, as is obvious, don't have the plethora of parts from previous seasons to use. Thus, new teams should purchase these parts themselves in order to have something to build from once the season starts, and starter kits offer these parts for less money than they would be if bought individually. As will be discussed, options that don't involve kits exist and are certainly very useful, but sticking to parts designed for FTC is recommended for new teams as a starting point.

6.1.2 Which Kit Should One Choose?

Choice of a kit is a matter of many a debate in FTC forums, and each team has their favorites. If you are a rookie team and do not have any experience with any of the kits above, we would recommend starting with a kit from either REV Robotics or goBILDA. These kits provide a good selection of parts, reliability, and value for money. They are easily expandable and great part flexibility allows for customization. Keep in mind that compatibility between kits varies and will be impacted by the unit system used.

- goBILDA (metric) is slightly more expensive, but is easier to get started with. It has an extensive parts catalog and can be adapted to other build systems like REV. However, its price point may deter some teams with lesser budgets.
- REV Robotics (metric) is slightly cheaper but generally has a bit higher learning curve than most other kits, as it is *extrusion* rather than *channel* based. *Extrusion* takes more effort to work with (you need to cut aluminum extrusion to length); however, it allows for maximum design freedom and adjustability. REV is also available from the *FIRST* storefront, which might be convenient for teams doing their purchasing through official school procurement systems.
- Tetrix (metric and imperial) is probably the simplest system to work with, but its part selection is limited, and the use of *4.7mm shaft* with *set screws* is inferior to *clamping hubs* used in other systems.

Attention: While VEX also sells some parts aimed towards FTC, **Game Manual 0 cannot endorse or support any VEX Robotics/IFI (VEX's parent company) products in any manner.** We cannot in good conscience drive people to support a business with a history of extremely concerning accusations of workplace harassment, toxicity, and general behavior that does not align with the ideals of *FIRST*.

goBILDA

goBILDA (https://www.gobilda.com/) is a fixed-pitch, *channel* based building system using metric units. It was recently released in the 2018-2019 season and is made by the same company as Actobotics.

Special Considerations

- goBILDA is based on metric measurements using M4 hardware. The hole pattern has 4mm holes on 8mm grid.
- The kit is primarily structured around *U-channel*, which is strong and durable.
- goBILDA's robust motion system allows teams to easily build drivetrains and mechanisms without worrying about placement and tensioning.
- It is arguable, but goBILDA has likely a low learning curve due to its simple yet extremely functional catalog of parts.
- goBILDA parts can generally be easily adapted to other kit parts such as REV.
- Additionally, goBILDA is still being developed at a quick pace and is responsive to the needs of the FTC® community.
- **goBILDA offers a 25% discount for *FIRST*® teams**.

Notable components

- goBILDA uses *8mm pitch chain* and HTD 3 or HTD 5 *belts* for motion.
- goBILDA mostly uses *clamping hubs* (called Sonic hubs), eliminating the unreliable *set screws*.
- goBILDA *shafts* are metric (*6mm D* and *8mm rounded hex*).
- The primary structural component is the 48mm aluminum *U-channel*, shown below.

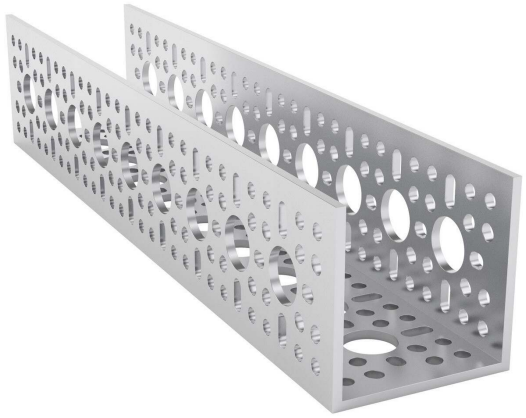


Fig. 1: goBILDA *U-channel* (48x48 mm)

- Low-Side channel (smaller profile *channel*) complements full size channel. Low-Side channel is just 12mm in height and allows for more compact builds. Additionally, Low-Side channel opens up possibilities such as a kit-based *parallel plate drivetrain*. Low-Side channel also forms the basis of the cascading kit with a 2 or 4 stage option.

Fig. 2: goBILDA low-side channel (12x48 mm)

- Mini-Low-Side channel (even smaller profile *channel*) that compliments Low-Side channel. Mini-Low-Side channel is 12mm tall and just 32mm wide, making it great for small structures.
- Viper-Slides are goBILDA's version of the commonly used drawer slides for linear extension in 336mm and 240mm lengths. Notably, goBILDA sells a Viper-Slide kit, which contains all of the parts to assemble either a 2 stack or 4 stack of slides. In addition, they are one of the few vendors who sells a belted version of these slides. The 240mm slide is also uniquely bidirectional, sliding both ways.
- goRAIL is *extrusion* that complements the *channel* offerings well. goRAIL is used in the linear actuator kit for extending mechanisms.

Fig. 3: goBILDA mini-low-side channel (12x48 mm)

Verdict

goBILDA's kit is a solid, albeit slightly more expensive, option for new teams. It offers a great deal of flexibility and part options from their catalog. goBILDA is a solid choice due to reasonable pricing, low learning curve, and thought-out design that reduces headaches.

Kit of Parts

- 2023-2024 Kit Parts[12]

Advantages

- goBILDA's Low-Side U-channel opens up many new possibilities due to its flexibility and compactness. For example, one can now make a *parallel plate drivetrain* without custom machining, or make custom width *U-channel*.
- goBILDA also has *Servoblocks* (identical to Actobotics other than hole pattern) which drastically increase servo life.
- goBILDA has native large bore *hex shaft* support (12 mm REX is comparable to 3/8" rounded hex) which is one of the main advantages of custom fabrication brought to a kit based system.
- goBILDA also is able to interface with TETRIX *channel* because they share some holes. For more information about TETRIX-goBILDA compatibility, check out this website[13]. It is also quite compatible with REV's ecosystem.
- goBILDA has a well thought-out *ball bearing* based motion system with smart motion transfer. It is easy to do *chain* or *belt* in channel.

  Note: 8mm *chain*, HTD 3, and HTD 5 mm *belt* can do perfect *C2C (center to center)* on the goBILDA pattern. This saves a lot of headache when needing to calculate *C2C* distances.

- goBILDA has plenty of *shaft*, *ball bearing*, and pillow block options.

[12] https://www.gobilda.com/ftc-starter-kit-2023-2024-season/
[13] https://gobildatetrix.blogspot.com/

- goBILDA motors can *face mount* natively into *channel*, eliminating the need for motor mounts and providing a robust, reliable way to mount motors.
- goBILDA has some special parts are unavailable in other systems, like *square beam shafts*.
- goBILDA provides a large selection of 12V DC motors. Their *Yellow Jacket motors* use orbital gearboxes and are available in 10 different gear ratios ranging from 30 RPM to 1620 RPM. No other manufacturer offers such varied gear ratios out of the box.

Disadvantages

- goBILDA is not the cheapest build system; it is relatively equivalent in pricing to Actobotics, but REV is cheaper.
- goBILDA *U-channels* are larger than the *channels* from Tetrix, resulting in larger builds. However, this is more than compensated by the fact that one can put a *goBILDA motor* ***inside*** a *channel*. In addition, the introduction of mini-low-side channel allows for more compact builds where needed
- Because of metric pattern spacing, goBILDA utilizes 8mm pitch *chain*, as opposed to the FTC standard #25 Imperial *chain*. This means that other kits' *chain* and *sprockets* won't work with goBILDA *chain* and *sprockets*.

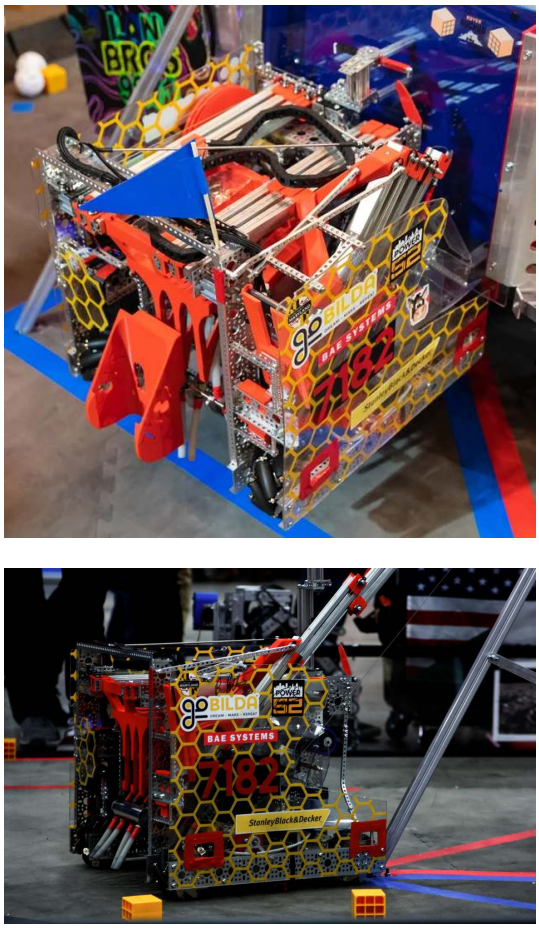


Fig. 4: An example of a successful goBILDA based robot, 7182 Mechanical Paradox Cubed: Finalist Alliance 1st Pick (Detroit), Rover Ruckus

REV Robotics

The REV kit is an infinite-pitch, *extrusion* based building system which uses the metric system and M3 hardware.

Special Considerations

- The REV ecosystem is based upon 15x15mm aluminum *extrusion* and is complemented by 45x45mm *channels*.
- With *extrusion*, there are no fixed mounting holes, increasing adjustability and flexibility. For example, tensioning *chain* is simple when sliding the mount or bracket increases tension.
- The REV system uses metric dimensions (15 mm extrusion, M3 hardware), with the exception of #25 roller *chain* (FTC® standard, imperial).
- REV has purposefully designed with compatibility in mind, as pattern adapters ease compatibility issues.
- Many REV parts are made of Delrin, a high-wear resistance plastic, which reduces overall cost. However, REV offers aluminum options for high-load parts as well.
- REV does have a steeper learning curve than *channel* based build systems, owing to the fact that constructing structurally sound mechanisms requires just a little more thought.

Notable components

- REV's bread and butter is the 15x15mm aluminum *extrusion*. It accepts M3 hex screws which slide along the grooves.
- In addition, REV also offers both 45x45mm *U-channel* and 45x45mm *C-channel*, which is stronger than *extrusion* and provides stiffness and support when needed.
- REV also offers punch tubing for teams wanting a more permanent mounting system while having the flexibility of extrusion.
- REV uses 5 mm hex steel *shafts* and a *ball-bearing* motion support system. The shaft may be cut to length, which can be very useful for certain use cases. Many vendors have added compatibility to 5 mm hex *shaft*.
- REV offers 3 types of *HD HEX motors*: spur gear, planetary, and UltraPlanetary motors. The UltraPlanetary has customizable planetary ring gear ratios for a very affordable price.
- REV also has pattern adapters for other systems built into many parts such as the aluminum brackets.

Verdict

The REV kit is good for teams willing to invest the time into an extrusion building system. It is the most flexible kit as it is extrusion based, yet it has the ability to integrate *channel* along with *extrusion*. REV offers the option to upgrade parts for those wanting a further investment into the REV building system.

Note: One advantage to the REV kit is the compatibility of 15x15mm MiSUMI *extrusion*.

15x15mm REV *extrusion* isn't as structurally strong as MiSUMI for two reasons.

1. REV extrusion is 6063 aluminum, while MiSUMI *extrusion* is made out of A6N01SS-T5 aluminum, a stronger and stiffer alloy.
2. The MiSUMI *extrusion* has a larger surface area, so there is more area of contact.

MiSUMI offers greater strength at a lower bulk cost. Additionally, MiSUMI will cut to the half millimeter for free, making it a great option for teams needing an exact cut. The drawback to MiSUMI is that it is quite a bit heavier than the REV *extrusion*. It is encouraged that teams use MiSUMI for drivetrain and structural support, and REV for mechanisms that will be under low to medium load.

REV also sells punch tubing, which is 15 mm aluminum tubing that allows teams to use the 15 mm REV building system without having the disadvantages of *extrusion*, such as that parts come loose over time.

With punch tubing, teams must pre-drill holes and attach, unlike extrusion, where teams can slide and adjust mechanisms.

Thus, it is recommended that teams use extrusion in prototyping/iterative design, and use punch tubing on the final iteration of their robot to save money. Punch tubing is compatible with the Metric Step Drill and 1/8” or 3.2mm pop-rivets.

Fig. 5: REV 15x15mm Extrusion

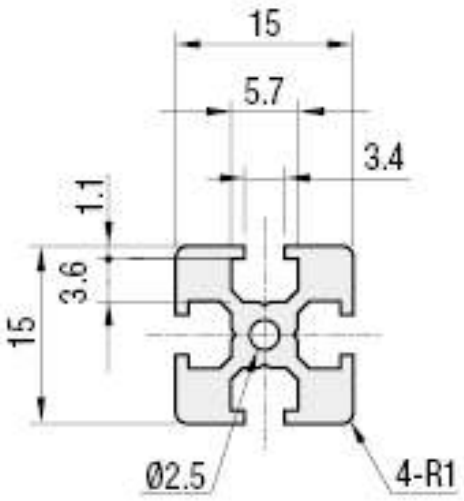


Fig. 6: MiSUMI 15x15mm Extrusion Profile

Kit of Parts

- FTC Starter Kit V3[14]

Advantages

- *Extrusion* systems don't need to worry about tensioning as mounts are adjustable to your needs.
- *Extrusion* allows teams to save space as opposed to *channel*, and is lighter than aluminum *channel*.
- *Extrusion* allows infinite positioning options instead of being locked in to a specific distance - useful for fine-tuning a mechanism.
- Delrin products are inexpensive yet durable for most use cases.
- REV has the option to upgrade to aluminum parts if need be - something that no other build system offers. (not for all parts)
- 5 mm hex is a robust *shaft* and motion system and is easily adaptable to *UltraHex* 1/2" hex *shaft*. Other companies have adapting options with 5 mm hex.
- Punch Tubing is a great final iteration option if you are sure about placement.
- *Channel* complements extrusion extremely well - having the adjustability of extrusion and the rigidity of channel makes it a very solid build system.
- Generally, REV products are designed with affordability in mind, and so the components are of good value for money.

[14] https://www.revrobotics.com/rev-45-1883/

Disadvantages

- 15x15mm *extrusion* is not as sturdy as *channel* options under high stress loads.
- Steeper learning curve, more time consuming to measure and cut *extrusion* to length.
- Requires tools such as a saw and bandsaw.
- Requires forethought and planning of *extrusion* length and placement.
- Parts loosen over time (to remediate: use punch tubing)
- M3 bolts, especially those sold directly by REV, are prone to bending under higher load (such as when used as an *axle* for a pulley)
- 5 mm hex *shaft* is also prone to bending, especially if the *shaft* is long.

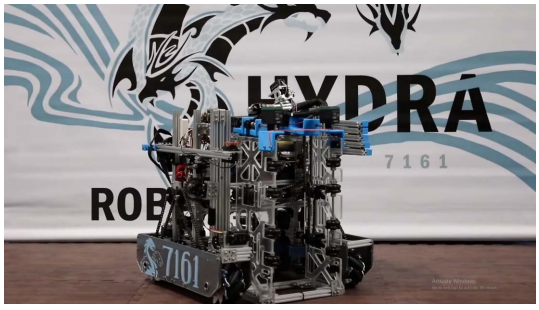


Fig. 7: 7161 ViperBots Hydra, Finalist Alliance 1st Pick (Houston), Relic Recovery

Fig. 8: 9889 Cruise Control, Rover Ruckus

Fig. 9: 6299 ViperBots QuadX, Velocity Vortex

Fig. 10: 11115 Gluten Free used both REV and Actobotics for the 2017-2018 season. Finalist Alliance Captain (Detroit), Relic Recovery

TETRIX

Attention: TETRIX is generally not recommended due to its restricted flexibility and low long-term reliability. This page serves as historical reference for teams that have and use TETRIX parts, but there is little reason to purchase them.

The TETRIX build system is a fixed pitch, channel-based building system that uses both imperial and metric units.

Special Considerations

- The TETRIX build system revolves around 32 mm aluminum *c-channels* and a 16mm bolt circle, on 16mm spacing (centers of the circles are 16mm apart). TETRIX is patterned along these *channels* to create many mounting options for building structures.
- These channels have a tendency to flex and bend under load, often requiring reinforcement using multiple *channels* to maintain their shape.
- While TETRIX *channel* is measured in metric units, the system employs Imperial (SAE) bolts and Imperial *chain*. This mix of units means that things often don't quite line up how they're supposed to, causing unforeseen problems.
- Additional tensioners are often required when using *chain* on TETRIX.
- TETRIX *gears* and *sprockets* offer very limited options for creating ratios. The aluminum *gears* tend to grind away very quickly, especially if they are not supported correctly on both sides.

- Tetrix parts generally do not have the best track record for reliability and longevity due to subpar material choices.
- Tetrix parts are slightly overpriced and lack bang-for-your-buck value found in kits such as REV and goBILDA.

Verdict

While you may be considering TETRIX as a starter kit, we encourage you to explore your options before selecting a kit. While Tetrix does pick up quite a lot of flak from the community, it isn't a bad choice in terms of physical qualities such as strength, but the mounting options and hole patterns leave more to be desired and restrict teams in terms of flexibility.

Kit of Parts

- Kit of Parts[15]
- 2020-2021 Kit of Parts BoM[16]

Advantages

- The TETRIX kit, being the most basic of all kit options, is easy to learn and provides a variety of options in building. The kit itself comes with *c-channel*, which is aluminum shaped in a C. It has pre-cut holes so motors, *gears*, or *drive shafts* can be seamlessly integrated into the channel. For a beginner team with little to no experience, a TETRIX kit allows you to assemble a working drivetrain in a couple of hours (Note: Most other kits allow you to do the same thing, but with more customization options).
- Furthermore, TETRIX is decent in terms of structural integrity, as long as the channels are loaded in the correct orientation. Typically, the orientation should be like an "n", with the top face upward. Connecting the channels with *locknut* instead of regular nuts aids in longevity. However, TETRIX is the weakest of the build systems and is very prone to bending, especially with long pieces of channel that have multiple axes of load.
- It is simple to build basic mechanisms such as an arm using the gears and d-shaft. However, there are potential drawbacks to doing so.

Disadvantages

- The TorqueNADO motors are comparable with *NeveRest* 60:1. They are usable, however have more limited uses due to their slow *gear ratio*, which means high torque but relatively slow output speed. The TorqueNADO uses spur gearboxes, and will not handle as much shock load as other planetary options, namely *AndyMark 20 Orbitals*, *REV HD Planetary*, or the *goBILDA Yellow Jacket Planetaries*.
- The aluminum gears seem to grind against each other often, even with correct spacing, and do not last very long, especially under high torque situations.
- TETRIX hubs are *6mm or 4.7mm round* and are based on *set screws* which are torque transferring. These *set screws* are notorious for coming loose under load, so special care must be taken to continuously tighten these screws. Additionally, it is highly recommended that teams use some sort of

[15] https://www.pitsco.com/Competitions-Clubs-and-Programs/FIRST-Tech-Challenge/TETRIX-FTC-Competition-Set
[16] https://asset.pitsco.com/sharedimages/resources/ftcset_productlist-0719.pdf

threadlocker on set screws, whether it be *Loctite Blue (removable) or Loctite Red (nonremovable)*. A possible workaround is purchasing 6mm D *clamping hubs* from goBILDA to use on TETRIX *shafts*, however, this also necessitates the use of a pattern adapter from the goBILDA pattern to TETRIX.

- Due to using a metric based pattern but using imperial based holes and *chain* standards, *chain* will not have perfect tension when running from one hole in *channel* to another. An external *tensioner* will be required to implement chain systems.
- TETRIX is also the most expensive kit on average while providing the most limited build options. Finally, the tight spacing and mix of units limits teams if they would want to integrate custom parts with TETRIX.
- Set screws on TETRIX hubs can mar motor *shafts*, and the *bore* of the hub itself can enlarge, leading to a wobbly or misaligned hub.

Tips for use

In any build system, it is important to properly support the structure of your robot. This includes supporting axles at two or more points of contact, or having multiple points of support for a piece of *channel*. Due to TETRIX's relatively fragile nature, this advice goes double with TETRIX. Instead of just supporting a *shaft* at two points, it is recommended to support the *shaft* at 4 or more (if you are using a 4.7mm standard). When mounting motors, it is recommended that *two clamping motor mounts* are used to mount one motor. Using *standoffs* inside *channel* to prevent the *channel* from bending inward or outward is also highly recommended.

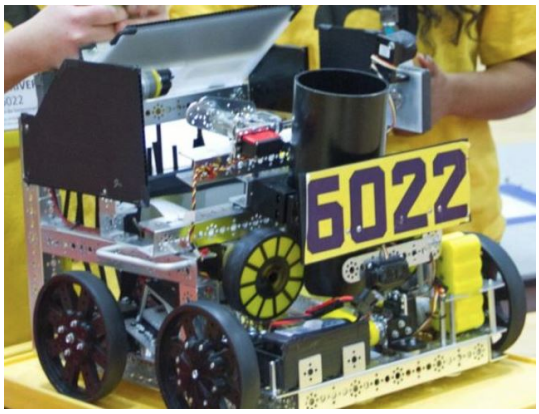


Fig. 11: 6022 To Be Determined: Worlds Semifinalist (St. Louis), Velocity Vortex
TETRIX based robots have succeeded in competition, albeit it has become rarer in recent years.

Actobotics (Discontinued)

Attention: As of June 2023, ServoCity is discontinuing the Actobotics line[17] to focus on goBILDA. We are retaining these pages as historical reference for teams that continue to have and use Actobotics parts.

The Actobotics kit from ServoCity (https://www.servocity.com/actobotics) is a fixed-pitch, *channel-based* building system using imperial units. Its 1.5" *c-channel* and *ball-bearing* based motion system allows teams to iterate and build mechanisms entirely from kit parts.

[17] https://www.servocity.com/blog/farewell-to-actobotics/

Special Considerations

- Actobotics' *channel* hole pattern has many more mounting holes than a Tetrix channel, so it offers more flexibility in terms of mounting.
- Actobotics uses imperial units across the board, allowing for clean spacing and fitting. However, most other kits use metric.
- Actobotics can interface with other kits such as REV through a variety of Pattern Adapters.
- ServoCity offers a 25% off discount for all *FIRST*® teams making, pricing competitive: https://www.servocity.com/first_team_discounts

Notable Components

- Mini-channel is offered in addition to the standard size *channel* for non-structural components.
- Actobotics' X-rail *extrusion* system offers immense adjustability and flexibility as it adds a *extrusion* component to a *channel-based* kit.
- To complement the structural offerings, Actobotics uses a series of clamping hubs and a robust 1/4" steel *D-shaft*.
- The Linear Motion Kit uses X-rail *extrusion* to build extending mechanisms.
- The *Servoblocks*, which prolongs the life of a servo, are recommended for all teams, regardless of kit. However, *Servoblocks* seamlessly interface with the Actobotics ecosystem.
- Actobotics furthermore offers motion options such as *bevel gears* and Linear Actuator kits (using *lead screws*) for specific use cases.

Verdict

Prior to being discontinued, Actobotics was a solid choice for new teams, offering a reliable base kit with many options to expand upon. Because it is no longer being produced we recommend teams consider other kits instead.

Kit of Parts

- 2020-2021 Kit Parts[18]

Advantages

- Actobotics provides a great value fixed pitch build system that is generally easier to assemble than *extrusion-based* systems which require cutting *extrusion* to length.
- Actobotics has the patented *Servoblocks*, which help drastically increase servo life by protecting it from shock loads. More information may be found in the glossary.
- Actobotics is easily compatible with other build systems such as REV's kit using adapters which can be found on the Servocity website.

[18] https://www.servocity.com/ftc-competition-kit-20-21-season/

- Actobotics is more sturdy than TETRIX in terms of drivetrain flex and has more support options to prevent structural bending.
- The *clamping hubs* offered are more reliable than *set screws*, as clamping engages the shaft in more places than a *set screw* (one face). As discussed in the *TETRIX section* (page 54), set screws in particular are vulnerable to loosening, especially without application of Loctite.
- Actobotics' motion system is **very** robust and relies on *ball bearings*, which have a lower coefficient of friction than bushings. Actobotics is compatible with the 5mm hex used by REV with their adaptable hubs.
- Actobotics also allows for *face mounting* of motors as opposed to *clamp mounting*.

Disadvantages

- Actobotics is no longer being produced by ServoCity, so it may be difficult to locate some parts.
- Actobotics is not very cheap, so its cost may be prohibitive for some teams with a low budget. Note that with the 25% off FTC® team discount, Actobotics can be cheaper than TETRIX. The only cheaper build system is REV.
- The XL *belts* from ServoCity are not great. It is recommended that *belts* are purchased from ServoCity instead, which may necessitate the use of 3d printed pulleys. There also isn't a large amount of space inside the channel for a belt or chain run, vastly limiting the size of the pulleys.
- *Channel* takes up more space than *extrusion*, so mechanisms can be a bit larger with the Actobotics kit. To remediate this issue, ServoCity sells *mini-channel* which is a similar size as *extrusion*.

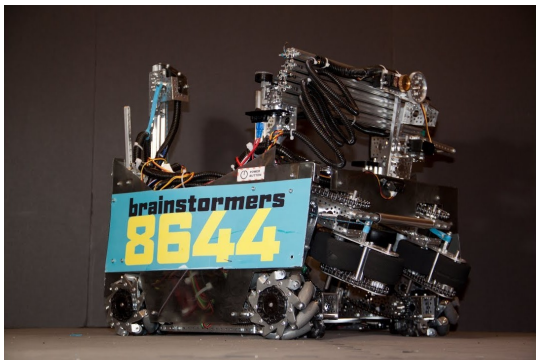


Fig. 12: An example of a successful Actobotics based robot, 8644 Brainstormers: 2018 Winning Alliance Captain (Detroit), Relic Recovery

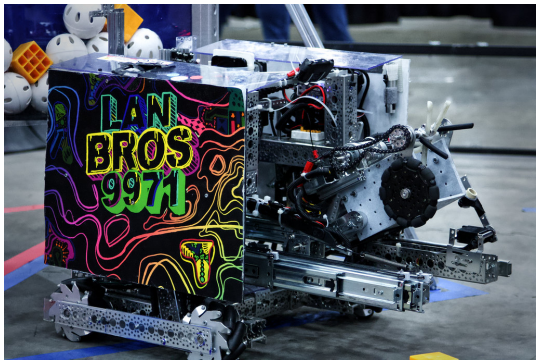


Fig. 13: Another successful Actobotics based robot, 9971 LanBros: 2019 Winning Alliance Captain (Detroit), Rover Ruckus

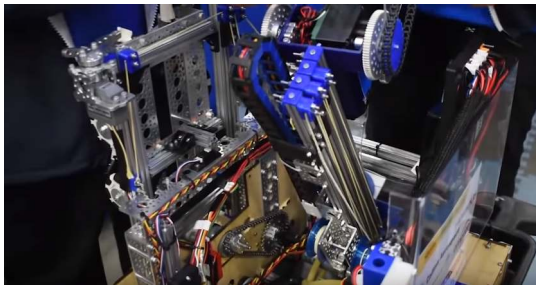


Fig. 14: 9794 Wizards.exe used both Actobotics and REV *extrusion* to build their Rover Ruckus robot

Custom

Many teams elect to ignore kits entirely and instead make their own mechanisms completely from scratch. While this allows for nearly unlimited design freedom, keep in mind *full* Custom has many caveats and drawbacks. A full CAD model is required for a custom robot, as well as machining capability. Custom robots are not suitable for a new team.

However, forgoing a full custom robot opens up many opportunities. Many teams choose to create a robot that is a mix of kit parts and custom parts. Our recommendation is to use kit parts as the backbone of the robot, and custom fabricate any additional parts necessary, instead of the other way around. That way, you can rely on the strength and durability of the kit while retaining the ability to customize small pieces and mechanisms to your advantage.

The methods that teams use to make custom parts typically involve either additive or subtractive manufacturing: either building a part from raw material (3d printing) or removing material from stock to create a part (Milling, Routing, and Laser Cutting).

You can now find details on 3d printing and milling in our *Custom Manufacturing* (page 69) section, with tutorials and general tips to help bring your parts from CAD to real life.

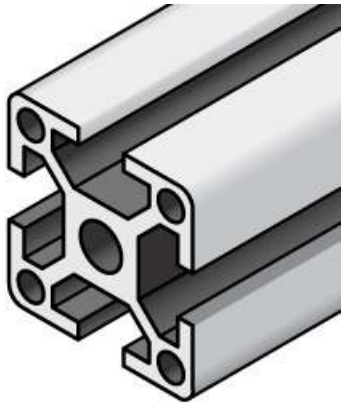
Kit Glossary

Channel Channel (more precisely called C-Channel) is aluminum that is in the profile of a C. (It is also sometimes called U-Channel.) Channel, along with *extrusion*, is the most common structural build element in FTC®, and is found in Tetrix, REV, Actobotics, and goBILDA kits.

Channel is fixed pitch, which means that there are pre-drilled holes that limit mounting to finite locations. It can be used to easily construct drivetrains; however, be aware that *gear* and *chain* mesh may not be with channel.

Extrusion Extrusion is aluminum shaped into slotted profiles able to accept certain types of hardware. For FTC, the most common is the 15mm extrusion, used in the REV and Misumi products. 15mm extrusion accepts M3 bolts and nuts (note that only regular M3 nuts can fit inside the slot, not *locknuts*).

Extrusion is not a fixed pitch system, allowing teams to adjust components as they wish. This makes it simple to achieve correct tension and put mechanisms where *channel* would limit mounting. The adjustability of extrusion is especially useful in precise situations, such as intake geometry. However, extra care must be taken to ensure components do not shift under load.



Ball Bearing Ball bearings refer to bearings with steel balls arranged in a circular fashion. This allows rotation of an element with less friction than a bushing, primarily because the surface area (or contact area) is much less than in a *bushing*.

Bearings are definitely recommended for drivetrain and high speed usage. Bearings are used in the Actobotics, goBILDA, and REV kits, and are commonly sold by most robotics vendors.

Servoblocks Servoblocks, sold by Servocity/Actobotics/goBILDA, are a way to mount *servos* to the goBILDA and Actobotics systems. It is by far the best way to mount servos because it decreases the load on the servo spline, which is the weakest part of the servo. This is because under load, the servo spline teeth can easily become stripped, rendering the servo unusable. While Servoblocks are not cheap, they are one of the best investments for teams to pursue.

Fig. 15: Actobotics dual ball bearing hub

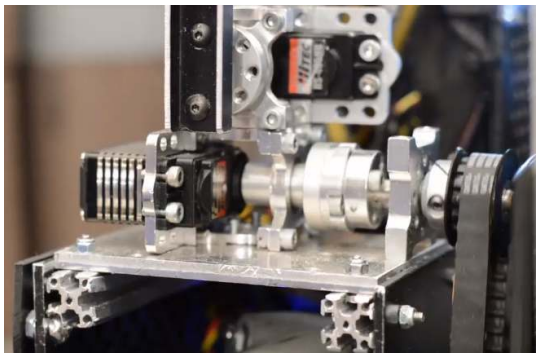


Fig. 16: 10030 7 Sigma, Relic Recovery

Clamping Hub A clamping hub is used to fixate part such as *sprockets* or *gears* on shafts. It is also used to prevent shafts from moving laterally. Unlike shaft collars, clamping hubs use screws to apply clamping force around the entire shaft, giving a better hold. As a result, clamping hubs are recommended over shaft collars.

Bevel gear Bevel gears are *gears* that transfer power along different axes, which are perpendicular to each other. Bevel gears are generally considered more inefficient than regular gears.

However, bevel gears can be very useful, especially in areas of limited space where the motor can be

placed perpendicular to the element it is driving, and not in the same plane.

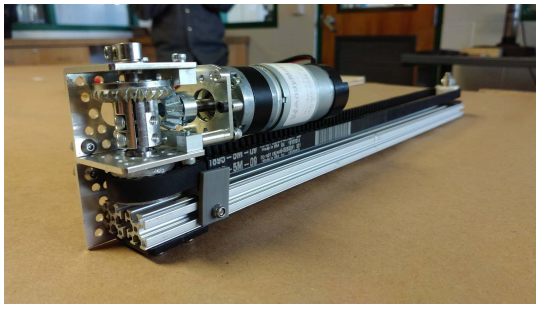


Fig. 17: 3736 Serious Business, Rover Ruckus

Lead Screw A lead screw is very similar to a threaded rod. It is used for high load and high torque application such as hanging. However, due to the nature of the threaded rod, lead screws are generally quite slow compared to linear slides. The speed of a lead screw is determined by two factors. The first is how fast the motor outputs, and the second is the number of threads per inch (TPI).

Shaft A shaft is a piece of shaped metal used in power transmission. Shafts are the primary method to transfer power from motor to wheel. Generally, shafts are made out of steel, so do not use a bandsaw to cut a shaft. Rather, use a hacksaw, as hacksaw blades can cut through steel. There are different kinds of bores in FTC, which are listed below.

- Round shaft
- D-shaft: has a flat part for set screws, otherwise round
- Hex shaft: six sided shaft
- Rounded Hex shaft: hex shaft that's been rounded so that it can run in round bearings
- Keyed shaft: round shaft which has a keyway (a slot) through the shaft

Locknut A locknut is a nut that resists vibration by the nyloc inside. Nyloc is a type of plastic that holds the bolt securely on to the nut when it is screwed in. It is advised that teams purchase locknuts instead of regular nuts as FTC mechanisms often become loose over time.

Bushing A bushing is primarily mounted on the outside of a *shaft*. It rotates in a pillow block, which holds the bushing. Generally, both are made out of a low-friction material such as Delrin or bronze.

Bushings are less efficient than *ball bearings* because they have a larger surface of contact, but are acceptable for low-load situations or low-budget teams.

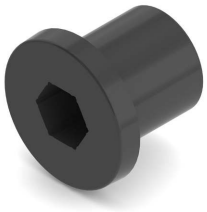


Fig. 18: REV Bushing

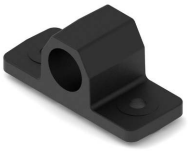


Fig. 19: REV Pillow Block

Churro Churro is a 1/2” or 3/8” hex product sold by AndyMark. It has a bore that is easily tapped to accommodate 1/4-20 and 1/4-28 bolts, and is commonly used as a large *standoff*. It is light and cheap compared to other hex products.

> **Warning:** Using churro as *shaft* is highly discouraged, as it is slightly undersized as well as prone to twisting.

Set Screw A set screw is generally a hex socket screw that is used to fasten parts such as *sprockets* or *gears* to a *shaft*, or to fix a shaft in place so that it doesn’t move around. Due to the hex socket, allen keys must be used to tighten and loosen set screws.

> **Warning:** **Set screws are not recommended for drivetrain and high-load applications** since there is very little surface area in contact with the shaft (only the tip of the screw). This makes the set screw likely to damage the shaft. Therefore, set screws can become loose very easily.

If set screws must be used, then it is imperative to use *Loctite* to reduce the chance of them shaking loose.

Note: *Clamping hubs* are much preferred to set screws, as clamping hubs apply pressure to the whole diameter of the shaft, as opposed to just one point.

Shaft Collar A shaft collar, which has a *set screw*, is fitted on to a shaft in order to secure parts.

Bore The bore refers to the shape of the opening that the shaft is inserted into. For example, the bore for a 5 mm hex *shaft* is the hexagonal shape.

"Stripping the bore" means that over time, the bore will lose its hexagonal shape, and become close to a circular shape, rendering the bore (and subsequently, the part it is on) useless.

Clamp Mounting Clamp mounting refers to securing a motor primarily by using friction instead of screws attached to the motor itself. This is generally discouraged as the motor can become loosened over time.

Tip: Use friction tape around the surface of the motor that is clamped down so that it will have less chance of moving around.

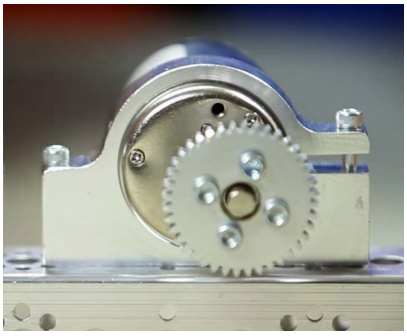


Fig. 20: TETRIX clamp mount and v1 motor

Face Mounting Face mounting refers to mounting the motor by affixing the motor directly to the mount using bolts. This is the preferable way of mounting the motor (compared to *clamp mounting*) because it is less likely to loosen over time, especially with the use of *Loctite* on the bolts.

Note: It is advisable that 4-6 bolts be used to face mount for redundancy.

Additionally, there is no way that the motor might rotate and cause a loss of tension in *belts* or *chain*.

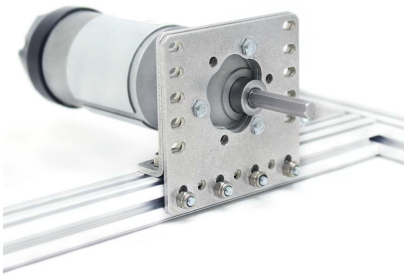


Fig. 21: REV v2 Motor Facemounted

6.2 Tools List

Here are a selection of necessary and useful tools for your team. You'll obviously need to purchase the basics such as the right screwdriver, hex keys, and drill bits. For a bit more precision and the ability to dabble in custom, opt for a drill press and bandsaw. A router might be useful for teams interested in working with wood, a great material for prototyping (and also for final iterations!)

6.2.1 Necessary

- **Safety glasses. Wear them when you're using power tools. Seriously.**
- Philips screwdrivers, assorted sizes
- **Hex drivers and hex L-keys**
  - 7/64" hex (Actobotics and TETRIX)
  - 3/32", ball head for set screws is discouraged (TETRIX)
  - 2.5/3mm hex (goBILDA)
- **Nut drivers and wrenches**
  - 5.5 mm hex nut driver/wrench (REV)
  - 7 mm hex nut driver/wrench (goBILDA)
- Drill and drill bits
- Pliers, needle-nose and locking
- Metal file (sandpaper not recommended, you're not working with wood)
- Quick lock clamps (2+) or vise

- Hammer and mallet
- Centerpunch
- Hacksaw (cuts through steel shafts)
- Wire stripper/wire cutter
- Zip ties/Velcro ties
- Electrical tape
- Stainless steel ruler and rafter square
- Sharp pencil or very fine permanent marker

6.2.2 Helpful

- Bandsaw

> **Caution:** A bandsaw **cannot** cut through steel *shafts*!

- Impact driver
- Drill press
- Miter saw with non-ferrous metal cutting blade
- Dremel (use sparingly; Dremel =/= bandsaw)
- Grip tape
- Caliper
- Soldering iron
- Heat gun
- Router or table saw
- Jigsaw
- Metal brake
- 3d printer

A 3d printer, CNC machine, or laser cutter greatly increases your ability to create custom robot designs. For most teams, buying a CNC or laser cutter is way beyond their budget, but you might be able to get access to one through school, local college, or nearby makerspace.

6.3 Tips and Tricks

Below is a collection of tips and tricks on basic building. Some of them may seem obvious, but almost every FTC® newbie made these mistakes at least once.

- **Color code your tools**. If you are using several different sizes of hex drivers (e.g. 2.5 mm and 3 mm used in goBILDA), color code them, using different color electric tape for different sizes.
- **Avoid set screws**. *Set screws* easily come loose, causing the hub to slip. In addition, *set screws* damage the *axle*, sometimes making it very hard to remove the hub later. For these reasons, whenever possible, use *clamping hubs* and collars instead of *set screw* ones. If you must use set screw hubs, use a non-permanent threadlocker such as Loctite blue[19] to prevent them from loosening.
- **Only use locknuts**. Never use regular nuts in your builds - they easily come loose under vibration. Kep nuts used in TETRIX are better, but they are still prone to loosening. For best results, always use *nylon locknuts*.
- **Do not use socket head screws on plastic**. Using socket head screws for attaching plastic parts (in particular, for attaching servos) damages the plastic. Use button head screws instead, or use socket head screws with washers.
- **Make sure screws are not hitting anything**. When you are attaching a part, make sure the screw you are using is not longer than the depth of the threaded hole - otherwise, the screw will hit the bottom of the hole and you will be unable to tighten it properly.

  This situation commonly happens when attaching parts to a *t-slot extrusion* (Misumi, REV, goBILDA's goRAIL) - if the screw is too long, it will hit the bottom of the slot, and no matter how hard you try, you won't get a tight connection. Another case when this situation arises is when using attachment blocks where two screw holes intersect - if you are not careful, you could have one screw hitting another.
- **Do not connect two components which both have threaded holes**. To get tight connection, the screw should go through an **unthreaded** hole in one component into a **threaded** hole in another, or through two unthreaded holes into a nut.
- **Removing stripped screws**. A common problem is removing a screw in which the hex socket head is damaged or worn out and thus the regular hex driver doesn't provide enough holding power to loosen the screw. Here are some ways to deal with the problem.
  - Get a better hex driver. Hex drivers can also be worn out, especially ball head hex drivers. Get a new hex driver (not ball head), which is not yet worn out, and it might give you more traction with stubborn screws.
  - Put a rubber band between the tip of the hex driver and the socket. This increases the traction.
  - Use a hacksaw or a Dremel to cut a slot in the screw head, turning it into a slotted screw that can be removed using regular flat screwdriver.
  - Use a screw extractor[20]
  - If nothing works, drill it out.
  - If that doesn't work, remember that a screw is not a screw if it is liquid metal. This is obviously not a very good idea. :)

  Needless to say, once you removed the damaged screw, discard it immediately - do not put it back in the box with other screws.

[19] https://www.loctiteproducts.com/en/products/specialty-products/specialty/loctite_threadlockerblue242.html
[20] https://www.amazon.com/dp/B07GZ17QD9/

7 Custom Manufacturing

This chapter covers the basics of custom materials and manufacturing for all of your creative needs!

7.1 Materials Guide

In FTC®, teams have design freedom in terms of what raw materials to use. However, there are definitely some important recommendations regarding material usage.

Common raw materials may be found at your local hardware store. McMaster-Carr and OnlineMetals are two frequently used online vendors. For more vendors, check the Appendix.

Here are recommended materials listed in order of importance.

7.1.1 Recommended Materials

- Aluminum channel & angle
- Aluminum extrusion (15 and 20 mm)
- Polycarbonate (Lexan™)
- ABS
- Delrin
- HDPE

7.1.2 Conditionally Recommended

- Aluminum sheet
- Aluminum *extrusion* (1 inch)
- Plywood
- Polyvinyl chloride (PVC)

7.1.3 Not Recommended

- Medium-density fiberboard (MDF)
- Steel
- Acrylic

7.1.4 Metals

Aluminum

- A high strength, medium-high density material. Suitable for use in nearly every application; recommended in load-bearing applications.
- Aluminum comes in *channel*, *extrusion*, flat, angle, sheet/plate, and is used in some drawer slides.

Aluminum *channel* is used in many build system kits and is very popular among teams of various skill levels. *channel* is strong yet relatively lightweight, and offers many mounting options for teams.

15 mm *extrusion* is compatible with M3 hardware, allowing teams to slide in bolts to their desired location. REV and MiSUMI offer 15 mm *extrusion*. REV *extrusion* is not as great structurally, but is lighter than MiSUMI *extrusion*. MiSUMI is more resistant to *flexing and/or twisting under load*. Both REV and MiSUMI *extrusion* are sold in bulk, and MiSUMI has the option to cut to the half millimeter. Keep in mind that a lot of *extrusion* can add up quickly in terms of weight.

Term

Torsional Rigidity Torsional rigidity refers to how difficult it is to twist an object due to an applied torque. This mainly refers to *extrusion*, as it is easier to twist extrusion than *channel* or an angle piece, for example.

Torsional rigidity has consequences particularly in building drivetrains, as the drivetrain is the last mechanism on your robot that should flex or bend when weight or force is applied to it.

MiSUMI and 8020 also offer aluminum *extrusions* in other sizes, such as 20mm and 1". 20 mm *extrusion* can be a good choice if you need a sturdier frame than provided by 15mm *extrusion*. Note that then you would need to buy special nuts, as 20mm *extrusion* is not compatible with M3 nuts. 1" extrusion is regularly used as the primary building system in FRC®, but is definitely overkill for FTC.

Aluminum flat and aluminum angle is widely available at hardware stores. In certain applications, such as adapting from different build systems, it is possible to drill custom adapter plates to mount mechanisms to the drivetrain. Aluminum angle is also a very sturdy structural support piece that takes up relatively little space, and can adapt to any build system. We suggest using 1/8" alumimum with drivetrain or mounting applications, and 1/16" aluminum for low-load situations.

Aluminum drawer slides, often with *ball bearings*, are recommended over steel drawer slides due to weight savings. Refer to the *Linear Motion section* (page 129) for more information.

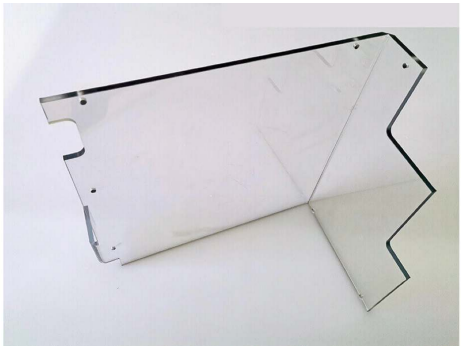
Sheet aluminum is generally used for drivetrain plates on custom drivetrains. The recommended thickness is 1/8" or 3/16". Because it is a plate, sheet aluminum will bend if not supported correctly with *standoffs* or *channel*. Only load the sheet in the plane that it is in (if the sheet is vertical, then only put vertical load on it; do not load it horizontally). However, there may be some applications that would benefit from a slight bit of flex for adjustability - in those cases, use your judgment and test it out for yourself!

Steel

- Steel is unnecessarily heavy for FTC structure. Aluminum provides plenty of strength at a fraction of the weight, and doesn't require welding.
- The proper uses of steel in FTC are in shafts (most are made out of precision-ground stainless steel) and gearboxes.
- Steel drawer slides can be used, but aluminum slides are highly recommended.

7.1.5 Plastics

Lexan



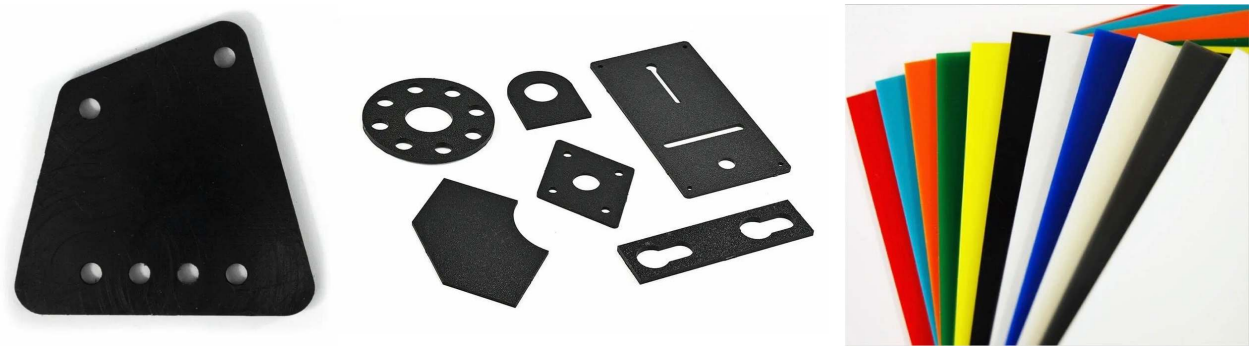
- Polycarbonate, commonly known by its brand name Lexan, is a material that is great for applications such as drivetrain plates or intake collector boxes.
- Lexan can bear load and is very impact-resistant.
- Lexan is commonly used in intake and deposit mechanisms as it is clear, allowing the drive team to see into the intake itself, an advantage over wood.

- Thick Lexan can be used for drivetrain plates, though this is not recommended for inexperienced teams.

Note: Lexan is one of the most expensive materials per square foot, so make sure you have carefully planned out what you are cutting before doing so.

Thin Lexan can be bent with a metal brake or sheet bender. It is recommended for teams to use bends instead of connecting with bolts - bending tends to be much stronger than bolting as it means the part stays in one continuous piece. If a sheet bender is out of the question, it is possible to use a heat gun or camping burners to heat up the Lexan in order to bend it. This is not recommended as it can cause injury and bubbling if the Lexan is overheated. Alternatively, “cold bending”, bending along a straight edge without applying heat can work for thinner sheets.

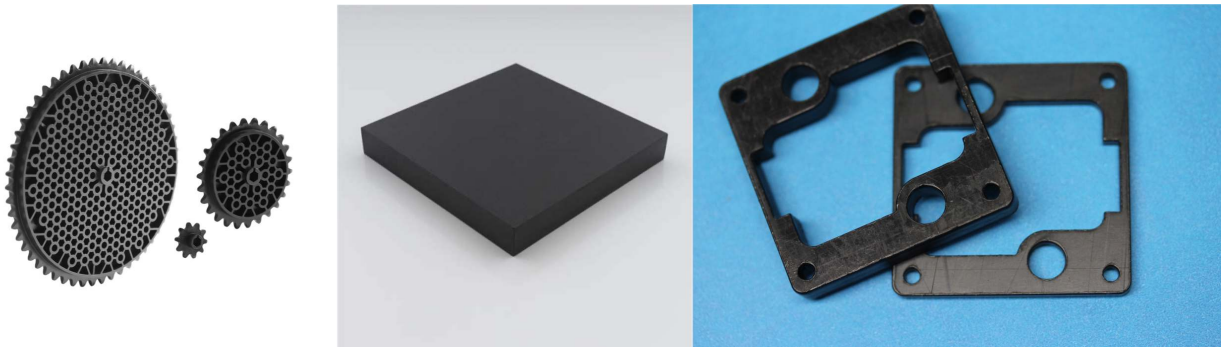
ABS



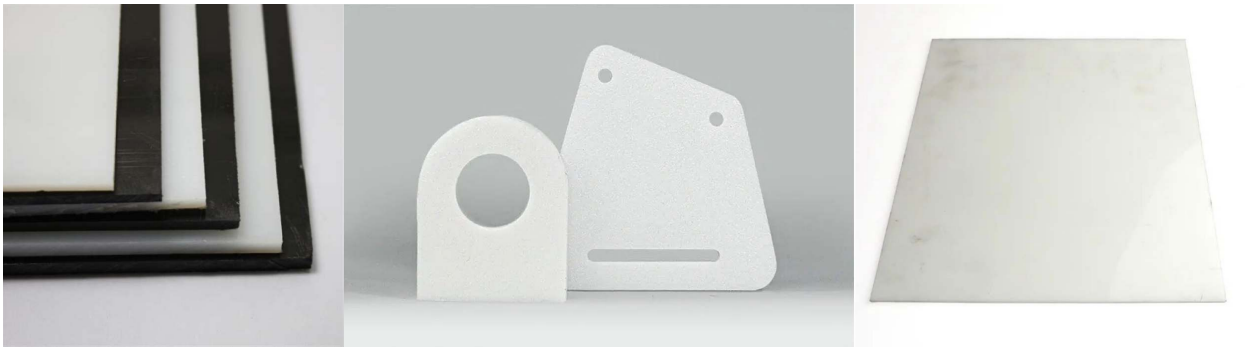
- ABS is a hard low-friction plastic that is easily machinable.
- ABS sheet can be used for side panels, ramps, and even drivetrain plates.
- It is not as strong as Delrin, but strong enough for most purposes.
- ABS is cheaper than Lexan and Delrin.
- ABS can be machined using hand tools or CNC. ABS sheets can be also bent using sheet bender (recommended) or a heat gun.
- Thin (1/16 inch) ABS sheets can also be cut using shears.

Delrin

- Acetal, also known under brand name Delrin, is a low-friction strong plastic. It is a common plastic used for wheels, plastic gears, and sprockets (especially in the REV system).
- It is a durable and strong plastic that is not easily cracked. Delrin can be used in drivetrain plates (use 1/4” or 3/16” thickness), but Delrin sheet is also quite expensive.
- Delrin can be machined in a variety of ways including laser cutting, CNC routing and hand sculpting. However, bending Delrin is much harder than bending Lexan or ABS.



HDPE



- HDPE stands for High Density Polyethylene and usually comes in opaque white or black sheets and plates
- HDPE is a cheaper alternative with low-friction characteristics similar to Delrin and flexibility/structure near polycarbonate.
- HDPE is easily machined by both hand tools and machine tools. However, it will burn in a laser cutter, so **do not laser cut HDPE.**

Polyvinyl chloride (PVC)

- PVC is a common lightweight plastic ubiquitous at any hardware store.
- PVC is most commonly sold as pipe, though PVC sheet does exist.
- FTC teams generally use PVC in order to customize intake rollers, especially for surgical tubing intakes. Due to the soft plastic, it is very easy to drill through, so teams often attach the surgical tubing to a PVC pipe. The PVC roller will then be attached to the intake motor.
- Since PVC is cheap, teams can easily experiment with different mechanism configurations.
- PVC pipe is sometimes used in cable management to run wires through, as it comes in different diameters and can easily be mounted.
- PVC sheet can be bought or made by cutting vertically along the pipe so that the profile looks like a C. Bake in oven. PVC sheet is pliable yet sturdy, making it a good option for backstops or customizing connectors that require some sort of flex.

Acrylic

> **Warning:** Acrylic is not a load-bearing material. It will crack and possibly shatter under impact.
>
> Do not use it on drivetrains by any means!

- Acrylic is a transparent thermoplastic commonly known as Plexiglass.
- As acrylic is not a structural material, it should only be used in mechanisms where there is no chance of shock impact.
- It may be used for aesthetic purposes or as shielding (to protect game elements from falling into the robot or from other robots tangling with wires, etc.)
- Treat acrylic as fancy clear cardboard - it cannot be loaded and should only be used in very specific circumstances.

7.1.6 Other

Plywood and MDF

- A medium to high strength material suitable for use in a wide variety of applications. Keep in mind that wood cannot be exposed to water or excessive humidity, as lumber for use in FTC isn't treated and may warp or expand. Do not try to use laser cut wood for a drivetrain. This is especially important if you live in a humid region such as Florida - wood expanding can completely ruin a custom drivetrain.
- Baltic Birch is the highest grade plywood, used in commercial applications and furniture. It contains 8+ layers (usually), is extremely dense, and is recommended for high-load applications or structure. It is quite an expensive material, so prototype and plan carefully before cutting.
- Plywood is not recommended for final iterations, but can be a cheap prototyping material. It doesn't bear load especially well and can flex quite a bit.
- MDF is generally discouraged as there are better options and absorbs water easily.

Carbon Fiber

> **Danger: Machining carbon fiber, like any fibrous substance, is a significant SAFETY HAZARD! Carbon fiber dust especially can cause cancer and is incredibly dangerous. DO NOT MACHINE/CUT carbon fiber unless you know what you are doing. When you do, make sure to either use machinery that is designed to cut carbon fiber, or cut in a well ventilated area with sufficient respiratory protection and running water over the carbon fiber.**

- Carbon fiber is one of the strongest materials for FTC use. For most teams, it is totally overkill, but it can be used in some specific applications.
  - Carbon fiber rods are used in custom linear slide extensions or multi-axis arms.
- It is probably the most expensive material to purchase.

Cardboard

Please do not use cardboard as a load-bearing material. We have seen too many teams use cardboard in ways that it shouldn't be used. Treat cardboard as a sheet of paper: it has no structural rigidity and only should be used as guides to channel pieces from A to B.

7.2 3D Printing

7.2.1 Kinds of 3D Printing

There are a few different kinds of 3D printing. FDM (Fused Deposition Modeling) (also known as Fused Filament Fabrication) extrudes a melted filament to create a part and is the most common type and the one we'll focus on in this guide. SLA (stereolithography) and SLS (Selective Laser Sintering) are both options for 3D printing plastics, but they are generally more complex, expensive, or hold other disadvantages in FTC® applications. For those reasons, they are not recommended.

Metal 3D printing (SLS and others) is also becoming more and more available, but is not in the scope of this guide.

Tip: Consider checking out the FTC docs 3D printing section[21], a guide for FDM 3D printing within the scope of FTC. It covers topics such as: bed adhesion, tolerances, designing for 3D printing, tuning, and hardware choices.

7.2.2 Advantages of 3D Printing

- 3D printing allows for customizable sizing and perfect optimization; for example, teams can print a spool of the exact diameter needed for optimal speed, or a belt pulley with a certain number of teeth.
- 3D printing allows teams to adapt between kits and individual parts easily, as not all kits have adaptable mounts or brackets. A good example of this are the Nexus *mecanum* bore adapters that teams 3D print.
- 3D printing allows teams to fabricate parts that would otherwise be impossible with materials such as aluminum due to machining restrictions.
- 3D printing allows teams to have customizable strain relief on wires and connections. This is a great project and well worth your time.

7.2.3 Disadvantages of 3D Printing

- **If you are out of 3D printed spares at a competition, you're probably out of luck. Teams are advised to print at least one set of every single 3D printed part as spares for competition**.
- 3D printed components are generally weaker than other materials such as aluminum. However, printing in the proper orientation can be very strong - teams have 3D printed hooks and other parts to support their FRC® robot (120 pounds) and FTC robots (40 pounds).

[21] https://ftc-docs.firstinspires.org/en/latest/manufacturing/3d_printing/index.html

- 3D printed parts should only be loaded in one orientation. That is, if the robot is hanging from a hook, the only load should be on the bottom face of the curved part of the hook. Try to eliminate side loads as much as possible to avoid part failure.
- The size of 3D printed parts is limited by the size of your print bed.
- Large and thick prints can take a long time (overnight) to print and can run the risk of failure.
- 3D printing can end up quite expensive, though filament can be found for a reasonable price on online vendors such as Amazon.

7.2.4 Common Filaments

For almost every part that needs to be 3D printed for FTC, **PLA (or PLA+, Pro, etc) and/or PETG will meet all the needs for strength, durability, and aesthetics.** These two filament types are by far the easiest to print, and are sold by many manufacturers for reasonable prices. Most of the other filaments here offer very specific advantages (like TPU) that come at the cost of effort, time, and money.

Danger: If your printer's hotend (the part that melts the filament) has a PTFE (Teflon) lining where the PTFE tube goes all the way down to the heat block (common in lower price printers like the Ender 3 and its variants), then you **should not be printing at or above 250 degrees Celsius**. Doing so will cause the PTFE tube to degrade and melt, releasing toxic fumes. If you need to print at these temperatures and you have a PTFE lined hotend, you can look at upgrading to an all-metal hotend.

PLA (Polylactic Acid)

The most common 3D Printing filament is polylactic acid, or PLA. It is a plastic made from biological sources like corn starch and sugar cane. PLA is stiff but more brittle than other filament options and tends to have little to no warp when printing. PLA is well suited to the majority of robot parts, but it may not hold up well to shock loads (impacts to parts), and as such parts should be designed accordingly.

- PLA hotend temperatures range from 190°-230° C
- PLA bed temperatures range from 20°-60° C, but a heated bed isn't strictly required for PLA

Tip: Due to the relatively low melting point of PLA, it is not advisable to leave PLA parts in locations such as a hot car, as this can produce severe warping in those parts.

There are many variations of PLA sold by different manufacturers, like PLA+ or PLA Pro. These filaments have various additives in them to improve strength, printability, and other properties. While more expensive, these products can greatly improve the performance of PLA and cover its weaknesses.

PETG (Polyethylene Terephthalate Glycol)

PETG can be described as a strength upgrade to PLA. It is not difficult to print, but often has noticeably more stringing and other minor artifacts. While it technically has a lower tensile strength than PLA, it is far less brittle and withstands impacts better, through slight flexing. It is a great option for FTC parts which need to be impact resistant where PLA will not suffice. Its higher temperature resistance also means it won't warp in a high ambient temperature such as a hot car.

Warning: PETG is known for bonding very well to print beds, **especially glass and PEI**, and is known to rip chunks out of the bed. It is a good idea to add some glue stick or hairspray before printing it.

- PETG hotend temperatures range from 230°-260° C
- PETG bed temperatures range from 60°-80° C

7.2.5 Less Common Filaments

These filaments are less used than those listed above, but can still find plenty of use cases on an FTC robot. These usually are used due to specific material properties such as flexibility or durability. These often come however, with substantial obstacles for printing that prevent some printers from printing them out of the box, along with sometimes being significantly more expensive.

ABS (Acrylonitrile Butadiene Styrene)

ABS used to be the standard filament for printing before PLA became commercially available. You've probably used ABS before in LEGO® pieces. It can withstand high loads and is quite ductile. This comes at the cost of printing difficulty, where an enclosure is often necessary to raise the ambient temperature and prevent severe part warping. The strength improvements over PLA can be more easily found in PETG, so ABS parts are not as common in FTC. ABS is quite affordable though, sold at the same prices as PLA.

- ABS hotend temperatures range from 230°-250° C
- ABS bed temperatures range from 100°-120° C
- Enclosure highly recommended to prevent warping

Due to the difficulty of printing ABS and its limitations, one might look at alternatives such as ASA which offer similar properties with much better printability.

TPU/TPE (Themoplastic polyurethane/Thermoplastic elastomer)

TPU and TPE are both common printing filaments that are widely used for their flexible material properties. This allows one to create printed parts that can easily flex and bend. Sold under many different durometers (a measure on the Shore Hardness Scale of the hardness/flexibility of a material), TPU/TPE's high impact resistance and layer adhesion make it not only a versatile filament, but an extremely durable one. In FTC, teams use TPU/TPE in roles such as printed intake flaps in place of a tube, as well as custom belts for low-load applications.

Tip: Due to its flexible nature, printers that use a Bowden tube extrusion system, where the extruder motor is not placed on the hotend, will find it extremely difficult to print TPU/TPE.

- TPU/TPE hotend temperatures range from 210°-250° C
- TPU/TPE does not usually need a heated bed, but if one is used it should not go over 60 °C as this will fuse TPU with print bed.
- TPU/TPE has the tendency to absorb a lot of moisture from the air, and thus will likely need to be dried before and perhaps during a print.
- Direct drive extruder is highly recommended

7.2.6 Exotic Filaments

There is rarely ever any need for these filaments in FTC. They offer extremely good material properties for parts that need to be subjected to high forces and adverse environments. They are all much more expensive than any of the filaments listed above, and offer a multitude of challenges for printing.

Nylon

Nylon filaments can be glass-filled, carbon fiber-filled, or pure. Nylon is the king of impact resistance in many situations where the part can flex out of the way, instead of completely breaking. Occasionally nylon is used for parts like wheel covers on drivetrains and in places where it will be repeatedly hit and battered. Nylon requires very high temperatures, generally requires an enclosure, and absolutely must be dried before (and while) printing.

- Nylon hotend temperatures range from 240°-260° C
- Nylon bed temperatures range from 55°-80° C
- Nylon is infamous for absorbing moisture from the air and should be thoroughly dried before and during printing. Failure to do this will probably result in a nearly unusable part.
- Enclosure recommended

Carbon Fiber-filled

Many filaments are also sold with the addition of small chunks of carbon fiber mixed into the filament itself. While often though of as an extreme strength improvement, these filaments are instead meant to be stiffer and help to improve the printability of filaments like nylon. Carbon fiber-filled filaments generally require higher temperatures, and a hardened steel nozzle, but if you can print the pure variants of those filaments, you should be able to print their carbon fiber-filled counterparts.

Polycarbonate (PC)

Polycarbonate and its variants are very very strong, technical materials. PC shines in its ability to be very rigid, and handle shock loads exceedingly well. PC also requires being dry, having a printer capable of **very** high temperatures, and an enclosure. It is a very challenging material to print, and is often very expensive. There is little reason to ever need printed polycarbonate parts in FTC, with no use cases requiring its strength.

There are several PC blends that can be much easier to print, a standout example is PolyMaker PolyMax PC. It is an easier-to-print, lower temp PC that retains many of the advantages of pure PC. PolyLite is not quite as impact-resistant, but a lot cheaper. Both are much easier to print than pure PC.

- PC hotend temperatures range from 250°-320° C

- PC bed temperatures range from 80°-140° C
- Enclosure required
- Filament must be kept dry

High-end exotic filaments

There are a few other materials that can have very high-end benefits, and push the envelope on what 3D printing can accomplish, but should not be printed if you are not *very* confident in your printing skills, and have basically no use in FTC. These materials include, but are not limited to, Delrin (Polyoxymethylene Homopolymer Acetal), PEI (Polyether Imide, brand name ULTEM), PEEK (Polyether Ether Ketone), and PEKK (Polyetherketoneketone). These materials are extremely hard to print, require ludicrous temperatures (some to the point where a hardened steel nozzle begins to melt), and are extremely expensive.

7.2.7 3D Printing Design Guide

Here is a quick guide on designing 3D printed parts that we hope is helpful for teams who may be unfamiliar with 3D parts.

The first consideration when designing 3D printed parts is print orientation. This refers to the side that contacts the print bed. Preferably, the part should have a flat bottom to maximize contact with the print bed.

Tip: Maximizing contact with the print bed will make sure the part doesn't delaminate or warp from the bed and increase print quality.

If it is impossible for the part to have a flat side to print on, a simple solution is to split the part into multiple parts along a plane. For instance, the gearbox plate below didn't have a flat side to print on, so it was split in half. The part was later sandwiched with numerous screws and plastic glue. If this plate was printed as one part instead of split in half, support would have to be used to create all of the necessary holes. Parts that use no support material make sure that the least amount of plastic is wasted.

Tip: Don't chamfer or round anything on the perimeter of the first layer on the bottom face of the part. Chamfering or rounding will increase the chances of the part warping, especially on unheated print plates.

Draft Angle

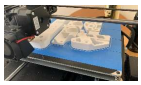
Draft angle refers to the overhang angle between the part side and normal vector from the print plate. The maximum draft angle refers to the maximum angle the printer can print without support material and is based on the printer, print settings (notably speed, temperature and cooling) and filament type. When trying to decrease support material, consider every overhang angle and make sure it is within the maximum draft angle. Staying within the maximum draft angle will also decrease the chances of part warping.

Stress Vectors

Perhaps the most important consideration is stress vectors. 3D printed parts are inherently stronger on two axes and weaker on one axis. The weaker axis comes from the layering action that defines FDM 3D printing. A common fix to this is to simply increase the print temperature up to a certain limit until it starts decreasing print quality, but the weaker axis can be resolved by again splitting up into multiple parts. The point to get across is to try to increase strength by optimizing sections of the part on the plane they are being printed on. For instance, this assembly below was responsible for hanging the entire robot, so it had to be the maximum strength possible for a 3D printed part.

Tip: It might seem counterintuitive to split up a part into multiple parts to increase strength, but there is a logic behind it.

The part could have been easily printed as a single part, but it would be fairly weak when stresses are exerted in the upward direction. Splitting the part and creating new flat surfaces to print on will strengthen each axis. *If one small part failed, the robot might still be able to somewhat function.* This would be preferable to the entire piece failing at once. In this example, the side pieces were printed as separate parts on a complementary axis to strengthen the assembly.



This assembly is a good example of considering part orientation, draft angles, and stress vectors in each part of the design. Complex parts can be made strong and without any support by simply splitting it up in the right way.

7.3 Machining

7.3.1 Advantages of Machining

- Machining allows teams to create practically any part without limitations.
- Machining allows teams to create custom drivetrain plates, arms, linear slides, etc to fit their design needs more effectively than kits.
- Machining is cool. Yes, really.

7.3.2 Disadvantages of Machining

- Perhaps the biggest barrier when making a fully custom robot is that it requires very expensive manufacturing equipment. While 3D printers have become more and more common in FTC® and allow teams to create or customize small parts, many teams do not have access to equipment such as lathes, routers, waterjet cutters, or CNC machines.
- Fully custom robots need to be fully designed in CAD. Sketching will not cut it when building custom robots.
- Another barrier is that prototyping with fully custom systems is almost always much slower than using kit parts. The full CAD process takes time, and fabrication/getting parts fabricated by a shop or sponsor is another lengthy process. Notwithstanding those two factors, if something goes drastically wrong then it will require a repeat process of CAD and fabrication.

It must be noted that teams do not need to pick between a fully kit-based robot or a fully custom robot. In fact, many successful teams are actually a hybrid of both kit and custom parts, as can be seen in many of example robots in this section.

However, our suggestion is that inexperienced teams should prioritize 3D printing over machining for fabricating custom parts, at least for the near term.

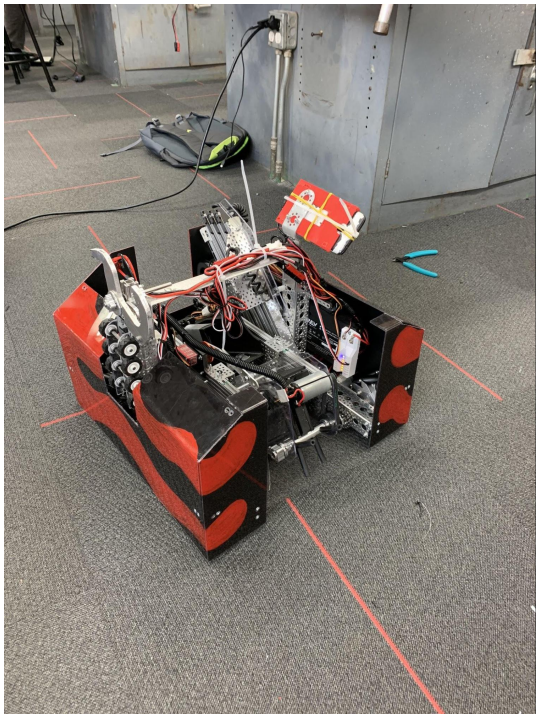


Fig. 1: 8680 Kraken Pinion, Division Semifinalist (Detroit Worlds), Rover Ruckus

Utilizes a kit-based drivetrain (Actobotics) and horizontal linear slides (REV) with a custom intake, housing, and mineral container. Kit linear actuator (Actobotics) with custom hook.

Utilizes a 3D printed (PETG and TPU) and machined (milled) arm/intake with a custom HDPE drive train (CNC routed)

Fig. 2: 8393 Giant Diencephalic BrainSTEM Robotics Team, Winning Alliance First Pick (Detroit), Relic Recovery, fully custom

Fig. 3: 731 Wannabee Strange, Rover Ruckus, Design Award Finalist (Houston)

7.4 Laser Cutting

Laser cutting can be a useful tool that allows FTC® teams to quickly produce complex custom parts. This guide will cover the uses of laser cutting in FTC, common types of laser cutters, and how to optimize your design for laser cutting.

If your team does not have access to a laser cutter, consider checking with sponsors or local makerspaces, or using a service like Fabworks[22] or SendCutSend[23]. These may have access to laser cutters that can cut more exotic materials. For example, Fabworks and SendCutSend both cut metals such as aluminum and steel, and SendCutSend specifically offers a wide variety of woods, plastics, metals, and composites. These services may also offer post-processing such as bending, tapping, and powder coating.

7.4.1 Common Types of Laser Cutters

In general, there are three types of laser cutters FTC teams may have access to: **diode lasers**, **CO2 lasers**, and **fiber lasers**.

Diode

These tend to be the smallest, cheapest, and lightest duty. They move around a small (<20 watts) laser diode on an exposed gantry. Usually they can only cut paper, cardboard or thin wood. In FTC, these can be useful for quick cardboard prototypes and for engraving decorative items such as team numbers and sponsor logos.

CO2

These are typically enclosed machines, using a moving mirror to aim the light from a 30+ watt laser tube. These can cut wood and plastics, but can only mark metal. They can be very useful in FTC for custom parts.

Fiber

These combine beams from multiple laser diodes into a fiber optic cable and can cut metal. Desktop versions usually have a very limited working area, often just a couple inches. These are very expensive and few FTC teams have access to them.

> **Warning:** These are generalizations, and are not always true. For example, some diode cutters can cut thin metals. For more information, always research your specific laser cutter.

[22] https://www.fabworks.com/
[23] https://sendcutsend.com/

7.4.2 What Can You Cut?

As mentioned above, CO2 lasers—like those commonly available to FTC teams—can cut wood and some plastics. Some of the most useful materials you can cut with a CO2 laser in the 40 watt range are:

Wood

Most thin wood can be easily lasercut and is great for prototypes as well as light-duty parts. For more info see *the material guide* (page 74). Please note that some woods - such as MDF - contain glues that release fumes when cut. Oily or resinous woods may also have a heightened risk of catching on fire.

Acrylic

Acrylic is a very commonly used material for laser cutting. Due to its tendency to fracture under load, it mainly finds use on FTC bots as decorative plates or as guides to funnel game elements. For more information on using acrylic in FTC, see *the material guide* (page 74).

Delrin

Delrin, also known as acetal, can be safely cut on most CO2 lasers with proper ventilation. Laser cut Delrin can be used to make everything from motor mounts to linear slide inserts to whole drivetrain plates. More information on delrin can be found in *the material guide* (page 72).

7.4.3 What Can't You Cut?

Danger: There are some materials, mostly plastics, you should **NEVER** try cutting on a desktop laser.

PVC

This emits toxic fumes which can damage your laser cutter and your lungs.

Polycarbonate

If you try to cut polycarbonate it will discolor and burn instead of cutting, releasing toxic fumes and potentially starting a fire.

ABS and HDPE

These will both melt instead of cutting cleanly.

Any unknown plastic

If you don't know what a plastic is, don't take the risk of cutting it. It could potentially burn or release toxic gasses. Be careful, as some different plastics, such as acrylic and polycarbonate, can resemble each other.

> **Warning:** This is not an exhaustive list and you should always research materials to ensure they will be safe before cutting them.

7.4.4 Design Guide

Note: When using a sponsor, machine shop, or service like Fabworks or SendCutSend, parts may be adjusted by the vendor for you. Check with the service you are using before you make these adjustments!

There are a few quirks of laser cut parts that you should keep in mind when designing them

Kerf

Most laser cutting software does not account for the width of the laser beam when cutting. This means if you require precise outer dimensions or hole diameters, you need to manually offset your paths in CAD.

Taper

As the lens in a laser cutter focuses light, it naturally creates a cone shaped beam. This leads to a beveled edge on parts. While this is usually not a problem for plates, it means pressfit non-flanged bearings may work their way out over time as they are squeezed harder at one side than another. One way to get around this is to lase holes undersized and then drill or ream them out to their final dimension to achieve vertical walls.

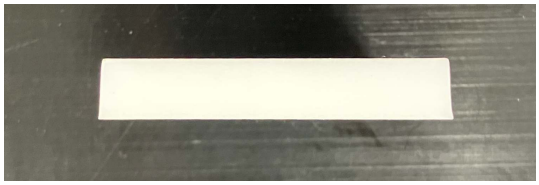


Fig. 4: Side view of a laser-cut acrylic part, showing the taper

7.4.5 Other Uses of Laser Cutting

A laser cutter can be used to precisely mark the positions of holes in a metal sheet, which can then be drilled out with a drill press to make metal plates.

7.5 Manufacturing Glossary

Laser Cutter A laser cutter is a tool that uses a high-power laser to cut through sheet metal or similar material. The laser is guided by CNC to cut preprogrammed patterns into the sheet.

Waterjet Cutter A waterjet cutter is a tool which cuts sheet metal and other materials via extremely high pressure water focused into a small stream. Waterjet cutters are commonly used in industrial fabrication and can follow preprogrammed instructions to cut patterns, similar to a CNC system.

Standoff A standoff is a fastener with two threaded ends and usually has a hex profile to be used with a wrench. These ends are usually female threaded, meaning that they can have a screw threaded into them.

This is usually a more compact alternative to a long screw and spacers, and can be used to space things out as well as fasten them. Custom standoffs can be made out of hex stock, such as *AndyMark Churro*.

Standoffs are usually used in drivetrain purposes, such as in parallel plate drivetrains, where the plates must be separated and supported by standoffs at equal distances.

8 Common Mechanisms

This chapter covers suggestions for design of common components of FTC® robots: drivetrain, linear motion (slides and lifts), claws, intakes, and more.

8.1 Design Tradeoffs

Problem	Solution
2 motor drivetrain • Less power • Less acceleration	**4 motor drivetrain** • Ability to traverse obstacles • Improved power and acceleration
Pushbot • Poor agility/manuverability • Poor top speed	**Mecanum, 6WD, etc.** • More agile, higher top speed • Customizable gear ratio
Claw • Controls 1 element at a time • Easy to break	**Intake** • Controls multiple elements at a time • Much more efficient
Spur gear gearboxes • Not for high load use cases • Will break under shock load	**Planetary gearboxes** • Better for drivetrains and high load • Resistant to shock loads
Single/multi-axis arm • Often needs high gear ratio • More complex than linear extension	**Linear extension** • Generally faster than arms • Much more precise

8.1.1 2 Motor Drivetrain 4 Motor Drivetrain

Important: In general, it is not recommended for teams to use 2 motors on the drivetrain, but instead use 4. This is mainly due to the added power and increased acceleration 4 motors provide.

Typically, top speed is determined by the gear ratio and the motor specifications, not the number of motors. **However**, acceleration is affected by the number of motors, and as FTC® robots need to change direction and accelerate numerous times per match, slow acceleration has a significant adverse effect on the competitiveness of the robot. In addition, 2 motor robots may struggle to get over obstacles or climb up ramps, due to less power. One question that often comes up is "don't I need more motors for other things on the robot?" This is a valid question, but the answer is generally no. It is possible to build a competitive robot with 4 motors allocated to drivetrain, and 4 motors for other mechanisms, so there should be no reason to skimp. Do remember that the drivetrain is the foundation of the robot. Your mechanisms will not be anywhere near optimized if the drivetrain can't get your robot from A to B quickly and efficiently. There has not been a game in FTC history where the top robots needed >4 motors for the drivetrain, so this is a pretty safe rule to follow.

8.1.2 Pushbot Mecanum, 6WD, Other Recommended Drivetrains

Important: The pushbot drivetrain, commonly built by first-year teams using the *FIRST*®-provided guides, is not recommended as a competitive drivetrain.

We do recommend teams who have purchased the Tetrix kit to build it for educational purposes only - that is, to get familiarized with the parts and basic building principles using a *channel-based* kit. However, we do not advise that teams use that pushbot at a competition due to its many flaws.

1. The pushbot is powered by 2 motors, and as stated above, there isn't a reason to stay with 2 motors on drivetrain.
2. The pushbot has poor top speed and turning ability, given that the gear ratio (40:1 on 4 inch wheels) is half the speed that many teams use.
3. It is not advisable to use direct drive.

However, most if not all of these problems will be solved by using a four-motor drivetrain such as the ones recommended in the drivetrain guide (mecanum, 6WD, etc.). Therefore, it is recommended for teams to refer to the *drivetrain* (page 90) section and see which drivetrain would fit best for their overall game strategy.

8.1.3 Passive Intake/Claw Active Intake

Important: Active intakes (ones with continuous rotational motion) should always be prioritized over passive intakes and grippers.

Active intakes are much more efficient and effective in picking up common game elements such as balls, cubes, and rectangular prisms than claws. This has been a widely accepted rule; many Worlds-level robots over the years use intakes. **The exception** is that a claw should be used for irregularly shaped objects that would be impossible to control via intake; for example, the relic in Relic Recovery.

Intakes have two major advantages over claws.

1. Intakes can control multiple game elements at a time.
2. Intakes are indiscriminate at picking up objects, making them much more efficient.

Claws can only pick up one object at a time, and the driver needs to aim the claw at that specific object to grab it. With an intake, the driver does not need to focus on one game element - instead; intakes will just pick up anything in its path, if designed properly. Claws are also prone to breakage, and thus suffer to defensive robots. They are also generally more fragile than intakes. Therefore, active intakes are as a result much more efficient than claws. Nearly every competitive robot from past years has used active intakes to great effect, so there is plenty of precedent to follow.

8.1.4 Spur Gear Gearboxes Planetary Gearboxes

Note: Spur gear gearboxes are fine for most applications for a rookie team. We are not advocating necessarily having to upgrade to planetary motors, but there are some advantages which may become useful in more advanced use cases such as high-load systems.

Important: Spur gear gearboxes have inherent disadvantages to planetary gearboxes. Spur gear gearboxes should not be used in high-load situations, primarily because the gears can strip and destroy the gearbox.

An example would be a drivetrain that has to change directions repeatedly and quickly. Planetary gearboxes are much better suited for drivetrain and arms, due to the configuration of the sun and planet gears. In addition, spur gear gearboxes are prone to shock loads; therefore, direct drive is not advisable on drivetrains. Refer to the *gearbox anatomy* (page 227) section for more complete information on gearboxes. **This refers to using spur gear gearboxes which are attached directly to the pinion gear of the motor. It does not mean external ratios outside of the motor gearbox, which will always be in a spur gear configuration.**

8.1.5 Single/multi axis arm Linear extension

Important: Teams are generally advised to stay away from arms and move in the direction of linear slides, primarily due to the issue of complexity. This is because arms typically are less effective than linear extensions and are harder to implement properly.

Arms require a high gear ratio, so arms must be supported extremely well to bear the torque that the motor provides. A poorly supported and/or constructed arm will cause the driver needless pain as it is exceedingly difficult to line up an arm that constantly shakes. In contrast, linear extensions do not need to worry about gear ratios and gearboxes. They can be optimized to be more efficient than arms, and typically are more precise, as linear motion is easier to control than angular motion. Another positive is that linear slides can have more extension than arms, with some reaching 5+ feet in length.

8.2 Drivetrains

This section will cover the heart of any robot, the drivetrain. The purpose of the drivetrain is to facilitate the movement of the robot, and thus is crucial to the overall function of the robot. If the drivetrain doesn't work, the rest of the robot won't work either. There are many possible types of drivetrains in FTC®, which we have covered in the guide. Drivetrains are split into two main types: tank (skid-steer) and holonomic.

8.2.1 Drivetrain types

Tank Drivetrain A tank drivetrain primarily utilizes *traction wheels* and cannot strafe (move sideways). To change directions, a tank drivetrain relies on either turning the wheels on the left and right side in the opposite direction (thus spinning the robot) or running one side faster than the other side (thus making the robot take the path of an arc). Tank drivetrains prioritize traction and acceleration over pure maneuverability, giving these drivetrains the potential to traverse obstacles and play defense. Tank drivetrains are relatively simple to build, yet are still competitive at the highest levels.

Holonomic Drivetrain A holonomic drivetrain, in contrast to a tank drivetrain, can move sideways, due to using either *mecanum* or *omni* wheels. These kinds of wheels have special rollers that allow strafing movements. Thus, holonomic drivetrains prioritize movement over traction. Holonomic drivetrains eliminate the time it takes to turn for a tank drivetrain. However, holonomic is susceptible to defense and can suffer with a heavy robot. Holonomic has been proven to be competitive at the highest level for many years, and is common among world-class robots.

8.2.2 Drivetrain selection

When building any mechanism, teams must list out some necessities or desired features. Here are some priorities for that we think are important for each drivetrain:

1. **Reliability**: Reliability, the key to success in *FIRST*® Tech Challenge, starts with the drivetrain, the foundation to any robot. One aspect of reliability to consider is the type of motor and gearbox that is used in the drivetrain. For example, spur gearboxes are more likely to break under load than a planetary gearboxes. (See the *Motor Guide* (page 224) for details). Thus, spur gear motors are not the optimal choice for drivetrain, especially if the robot is projected to be on the heavy side (30+ pounds).

Note: Generally, more complex drivetrains pose more reliability challenges for inexperienced teams. Our advice is to stick to the simpler drivetrains such as 4 or 6 wheel drive and mecanum drive.

---

2. **Agility**: There are many factors to agility: top speed, acceleration, turning radius, turn speed, and ability to strafe. Note that turning radius is an often overlooked feature that is critical to the overall agility of the drivetrain. Generally, a solid drivetrain should have a free speed (speed under no load) in the range of 4.5-6 feet/second.

3. **Number of motors and gear ratio**: Generally, new teams may try to use only two motors on the drivetrain. While this is possible, it is not recommended, as all competitive teams use 4+ motors on the drivetrain. Another issue stemming from experience is that most teams' drivetrains are too slow. More advanced teams may focus on the ability to play *defense*, but in general, maneuverability and speed are the main factors to a successful drivetrain.

   60:1 and 40:1 motors are **almost always too slow for FTC drivetrain use cases**. Any gear ratio between 16:1 and 20:1 is perfectly reasonable on 4 inch wheels. 19.2:1 on 4 inch wheels is a popular choice

because it enables one to go 1 to 1 off of a 19.2:1 planetary motor. This ratio gives a great balance, having near instant acceleration and a high top speed.

On 3 inch wheels, the equivalent ratios are 12:1 to 15:1, which makes 13.7:1 on 3 inch wheels convenient, as it can be taken 1 to 1 off of goBILDA's 13.7:1 planetary gearbox motor. Teams can slow the drivetrain down in code by providing less power to the motors if needed.

Warning: It is not recommended for teams to use spur gearboxes on their drivetrain. Instead, use planetary gearboxes, as they are less prone to shock loads and breakage.

4. **Traction/Pushing Power**: While this feature is often overemphasized, it is still very important. Pushing power describes a drivetrain's ability to endure defense/engage in defense. In addition, traction will be important if the drivetrain must traverse obstacles or some sort of terrain. Many factors affect the pushing power of a drivetrain, including wheel type, motor gearing, and overall weight of the robot.

   Note: If you already have a very agile drivetrain with experienced drivers, a team can avoid defense instead of having to fend it off or engage in it.

5. **Powering the drivetrain**: Generally, there are four options for power transmission: direct drive, *chain*, *gear*, and belt. More detail about each option can be found in the power transmission section.

   Teams should stay away from direct drive, as gearboxes are prone to breaking, especially under shock loads (for example, if the wheel is hit by another robot, or the wheel slams into the wall).

   We recommend belted drivetrains, but realize that belt is a difficult option for new teams. Chain and gears are also great options - chain requires a bit more forethought, as 1+ tensioner per side is required to maintain correct tension in the chain.

   CAD or a detailed sketch is generally recommended with chain in order to visualize the chain run (where the chain will be placed). Gears are slightly simpler, and can be a fantastic and easy option, especially when using extrusion. We would advise to stay away from TETRIX gears, and use the gears from other kits such as the REV delrin gears (with hex hub strengthener) or REV aluminum gears.

   See the *Power Transmission* (page 110) section for more details.

An important step is determining what you want out of your drivetrain.

- Do you want speed?
- Pushing power?
- Ability to get over terrain?
- Do you need to strafe?

All of these questions should be answered before choosing a drivetrain.

8.2.3 Drivetrain Options

Tank (Skid-Steer) Drivetrains

2 Wheel Drive (Pushbot Drive)

Recommended only for first drivetrain, not for competitions

This rookie drivetrain is considered one of the inferior drivetrains, though it is usable at low competition levels. This is the introductory drivetrain type for many rookies, as it is promoted in official guides published by *FIRST*® (giving it the name pushbot). It often has *direct driven traction wheels* with unpowered *omni wheels*.

This type of drivetrain has poor turning as the center of turning is at the back of the robot between the two powered wheels. In comparison to other drivetrains, it has poor acceleration due to only using two motors.

Even though it may not be an optimal drivetrain, it is still possible to be competitive as long as the drivetrain is reliable. As a consensus, we would advise every new team to build the pushbot primarily to learn how to build with a kit. The pushbot is a good starting point and helps the team get familiarized with using kit parts, attaching wheels, mounting motors, etc. However, it is subpar to every other drivetrain in a competitive context.

Attention: While the pushbot is a good first drivetrain for new teams to get acquainted with the kit, it is recommended that teams move away from this drivetrain when building their competition robot.

Advantages

- Most simple drivetrain to build
- No need to worry about powering all four wheels

Disadvantages

- Slower than other options
- Underpowered (all other drivetrain options typically use 4 motors)
- Uses the Tetrix MAX motor, which is prone to burning out easily, is underpowered, and has a fragile gearbox.
- Lacks agility and maneuverability due to 2 motor turning
- Poor acceleration due to 2 motors
- Often *direct driven*, which is highly discouraged for drivetrains

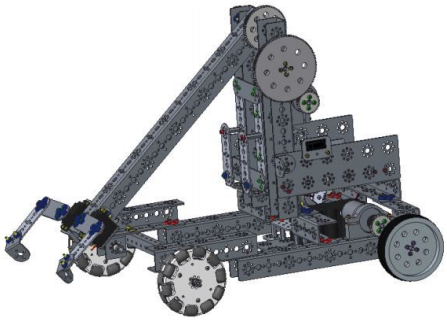


Fig. 1: Example pushbot drivetrain included in the starter FTC® guide

4 Wheel Drive

In its most common form, the drivetrain uses the same wheel layout as the two wheel drivetrain (2 traction wheels in the back, 2 omnis in the front), but with one notable difference: **all four wheels are powered**.

Some teams will put all four traction wheels or use all four omni wheels. It is not suggested to use all 4 traction wheels due to poor turning that results in this setup. This is caused by wheel scrub. Using omni wheels for all four wheels will result in incredible turning, with the robot rotating around its center.

Term

Wheel scrub Wheel scrub refers to friction between the side of the wheel and the floor tile. It inhibits turning as the drivetrain must overcome this frictional force in order to turn the robot. Wheel scrub is most common on 4 or 6 wheel tank drivetrains that do not have a *center drop*.

However, this advantage comes with a major loss of traction. For these reasons, many teams choose to use two traction wheels and two omni wheels for a balance between quick turning and traction. The primary advantage of this drivetrain over other tank drivetrains is its ability to easily move across raised terrain when the bot's center is raised above the terrain.

It is suggested that a four wheel drivetrain be close to, or exactly, a square. Otherwise, one may encounter problems turning.

Note: *Weight distribution* is furthermore a large factor that should be considered: the more weight in the back, the better.

Term

Weight distribution Weight distribution generally refers to how the weight of the robot is proportioned. It is desirable to have a relatively 50-50 (50% of weight in the front half, 50% in back half) so that the drivetrain has optimal manuverability and turning.

Off-center turning, which may or may not be a drawback, is nearly ubiquitous among 4 wheel drives. This may not be a problem for teams, but it is good to be aware of it. Off-center turning can be an advantage, but be warned that turning will be slightly slower on 4 wheel drives than their six or eight wheel counterparts.

Advantages

- More maneuverable than 2WD
- Solid acceleration and traction
- Can traverse terrain if the chassis is raised high enough
- Good pushing power for defense, yet maneuverable enough to avoid it

Disadvantages

- Can tip more easily than 6WD/8WD with a high center of mass
- All traction wheel 4WD can have decreased maneuverability
- Weight distribution factors into the turning point and turning radius of the robot

CAD Examples of Four Wheel Drive (Click to expand)

goBILDA COTS

Click here to explore this model in OnShape online CAD[24]

REV COTS

Click here to explore this model in OnShape online CAD[25]

goBILDA Custom

Click here to explore this model in OnShape online CAD[26]

REV Custom

Click here to explore this model in OnShape online CAD[27]

[24] https://cad.onshape.com/documents/45549489f570f3694569a2df/w/85ff26b9fca4988ebc4df3b4/e/873a6e756fd385a1b743bdc1

[25] https://cad.onshape.com/documents/45549489f570f3694569a2df/w/85ff26b9fca4988ebc4df3b4/e/e02b1ee98816af5505b528e2

[26] https://cad.onshape.com/documents/45549489f570f3694569a2df/w/85ff26b9fca4988ebc4df3b4/e/16bae3d8b801874d9b1daaff

[27] https://cad.onshape.com/documents/45549489f570f3694569a2df/w/85ff26b9fca4988ebc4df3b4/e/fab2c2e33242281f0d46e524

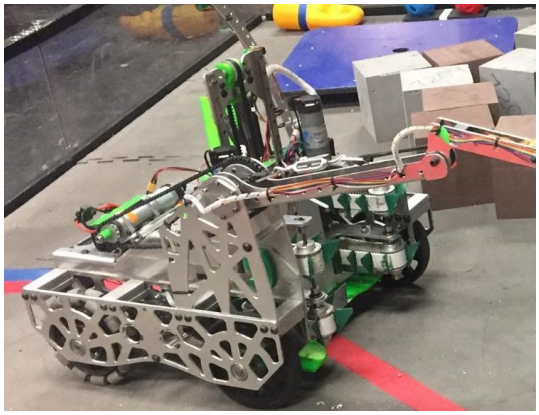


Fig. 2: 7209 Tech Hogs, Relic Recovery

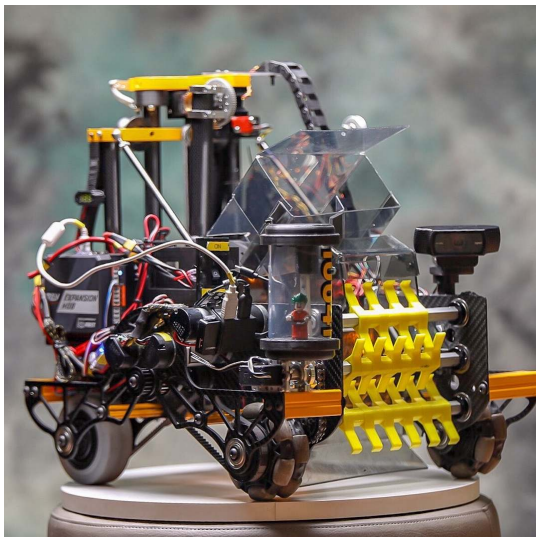


Fig. 3: 10641 Atomic Gears, Rover Ruckus

6 Wheel Drive (6WD)

A 6 wheel drivetrain is a common competitive drivetrain in FTC for multiple reasons: it has fantastic traction, great turning, and by having 6 wheels, the drivetrain has more contact with the ground, helping with stability and traction. There are two main types of 6 wheel drivetrains: ones with **corner omnis** and ones with a **drop center**.

Term

Drop Center A drop center 6WD is a 6 wheel drive with the center wheel mounted slightly below the other two wheels. The drop should be anywhere from more than 1/16" up to 1/4". However, the recommended drop is around 1/8". Typically, lighter robots (<25 lbs.) can have less drop, while heavier robots (>25 lbs.) perform slightly better with more than 1/8" drop.

The purpose of dropping the middle wheel is to ensure that only 4 wheels are in contact with the ground at all times. This is because turning with 6 wheels on the ground introduces lots of friction, making it very difficult to turn quickly. Note that the **required drop may vary depending on both field condition and weight of the overall robot**.

Turning can drastically degrade due to a difference in material underneath the field, leading to the robot sinking down further than usual.

Most drop center 6 wheel drives are made using custom drivetrains because it is difficult to get the center wheel drop using a kit based build system (a notable exception being REV-based kit drivetrains). It is possible to execute a drop center using goBILDA and Actobotics using pillow blocks, but it is a little bit more awkward. However, with the new goBILDA drop-center bearing plate[28], it is straightforward to make a drop center drivetrain on goBILDA channel.

6 wheel drives with corner *omni wheels* do not need to have a center drop. It attempts to solve the issue of turning by replacing the corner traction wheels with omni wheels, allowing the drivetrain to achieve better turning, albeit with slightly less traction than a center-drop. This is very easily buildable in kits, and is a great all-around drivetrain. Drop center and corner omnis can be combined for maximum turning reliability, although this comes with side effects like rocking and reduced traction.

Advantages

- Great traction and maneuverability
- Good acceleration, can have a high top speed
- Great stability under all robot weights
- Able to play defense

[28] https://www.gobilda.com/1616-series-drop-center-bearing-plate-2mm-drop-4-pack/

Disadvantages

- Drop-center 6WD is tricky to build with *channel* based systems, though this has been partially mitigated with the goBILDA drop-center bearing plate[29]
- Drop-center 6WD is slightly worse at turning, but has more traction
- Drop-center 6WD is dependent on field conditions
- Corner omni 6WD has less traction
- Slow *gear ratios* will make a 6WD feel sluggish

CAD Examples of Six Wheel Drive (Click to expand)

goBILDA COTS

Click here to explore this model in OnShape online CAD[30]

REV COTS

Click here to explore this model in OnShape online CAD[31]

goBILDA Custom

Click here to explore this model in OnShape online CAD[32]

REV Custom

Click here to explore this model in OnShape online CAD[33]

8 Wheel Drive

An 8 wheel drivetrain is less common than its 6WD counterpart, combining elements found in both 4 wheel and 6 wheel drivetrains. For example, the 6 wheel drivetrain generally will have a dropped center wheel so that the robot turns on four wheels instead of six, reducing friction and increasing turning mobility.

On an 8 wheel drive, the center four wheels are dropped. This means that when turning, only these middle four wheels are touching the ground. Thus, the 8 wheel drivetrain has slightly more stability while turning than a 6 wheel drive, whereas 6 wheel drives can turn more quickly. Furthermore, since the 8 wheel drive has wheels in the same place as a 4 wheel drive, it has the stability of a 4 wheel drive.

[29] https://www.gobilda.com/1616-series-drop-center-bearing-plate-2mm-drop-4-pack/
[30] https://cad.onshape.com/documents/45549489f570f3694569a2df/w/85ff26b9fca4988ebc4df3b4/e/97e67997606a54fcabd367ac
[31] https://cad.onshape.com/documents/45549489f570f3694569a2df/w/85ff26b9fca4988ebc4df3b4/e/a787322363f7646a8b7cb69e
[32] https://cad.onshape.com/documents/45549489f570f3694569a2df/w/85ff26b9fca4988ebc4df3b4/e/05fbe7b65c6c4be0b0cddabb
[33] https://cad.onshape.com/documents/45549489f570f3694569a2df/w/85ff26b9fca4988ebc4df3b4/e/258524b3c7582178ba684ac5

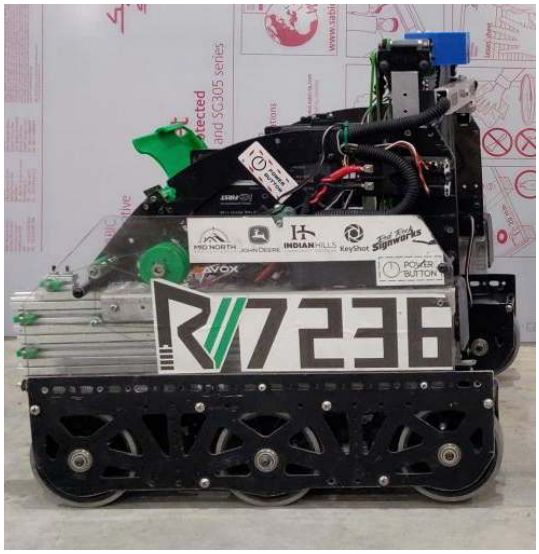


Fig. 4: 7236 Recharged Green, Rover Ruckus; drop center 6WD

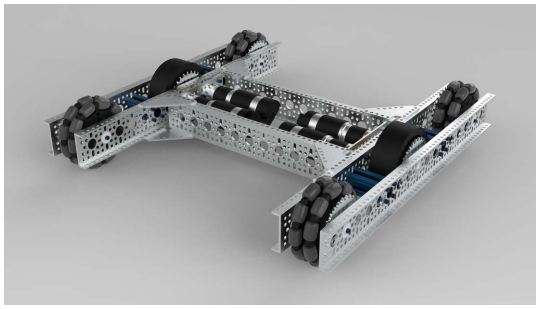


Fig. 5: Ethan Doak, goBILDA 6 wheel drive with corner omni wheels

It is suggested that all eight wheels should be powered, and *planetary* motors should be used over a *spur gear* motor.

Teams also have the option of using doubled omni wheels on the outer four wheels. Doing so will reduce traction/pushing power and increase mobility.

Advantages

- Combines the stability of 6WD with the agility of 4WD
- Even more stable than 6WD
- Fantastic traction and acceleration
- Great for defense

Disadvantages

- Takes up more space than 6WD
- Powering all 8 wheels can be tricky
- Without adequate center drop, turning can be drastically reduced

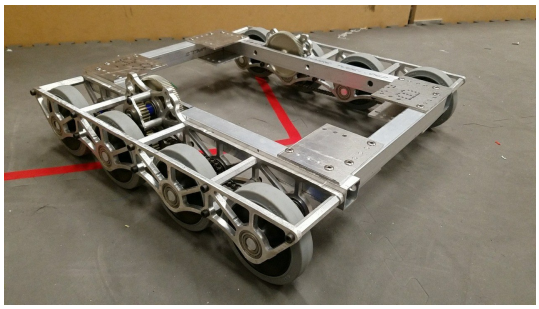


Fig. 6: 3846 Maelstrom, Rover Ruckus

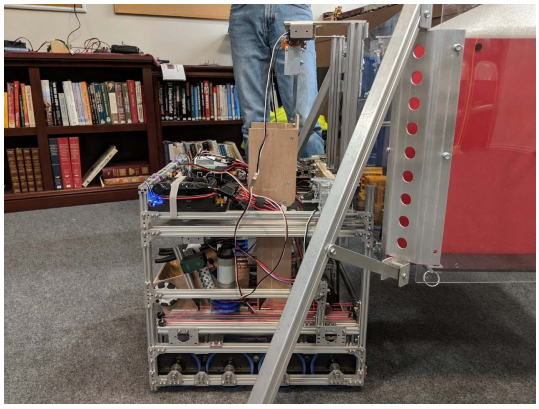


Fig. 7: 13075 Coram Deo Academy Robotics, Rover Ruckus

Tread Drive

Tread drive is the use of tank treads or wide belts to power movement, much like a real life tank. Unfortunately, in FTC, it is not a competitive drivetrain for a number of reasons.

Tread is complex, and has many points of failure. Treads are also very prone to defense, and a slight hit from another robot is enough to misalign the treads. *Commercial Off-The-Shelf (COTS)* tread options aren't great either - TETRIX tracks have a tendency to snap and derail when used on robot drivetrains, making them not suitable for competition use.

While it is possible to implement tread drive successfully, such as in the example below, most inexperienced teams do not have the capability and know-how to do so. Tread drive has negligible traction improvements at the cost of maneuverability. There are better options to traverse terrain, such as a 4WD.

Advantages

- Very good at traversing terrain
- Fantastic traction and pushing power

Disadvantages

- Suffers in maneuverability and top speed
- Very complex to implement
- Treads are prone to breakage and can fall off easily

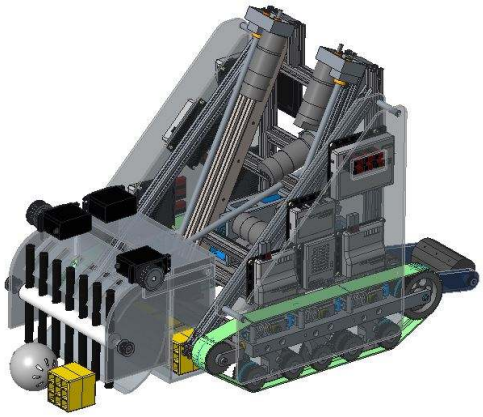


Fig. 8: 5975 Cybots, Res-Q

Holonomic Drivetrains

Mecanum Drive

Mecanum drivetrains consist of four mecanum wheels which are powered independently by one motor. This configuration angles the velocity of each wheel, allowing the robot to strafe.

Term

Mecanum Wheel Mecanum wheels are a special type of wheel that enable maneuverability and holonomic strafing as opposed to traditional wheels. They consist of a series of rubber rollers rotated 45 degrees to either the left or right.

In a conventional mecanum drivetrain, running the wheels on one diagonal in the opposite direction to those on the other diagonal causes sideways movement. Combinations of these wheel motions allow for vehicle motion in any direction with any vehicle rotation (including no rotation at all).

The primary advantage to mecanum drive is the maneuverability it affords, especially because the robot can strafe instead of turn and drive. The rollers on mecanum wheels form a 45 degree angle with the wheel's axis of rotation, which means that mecanum drivetrains can't strafe as fast as they can drive forward.

This can be explained by discussing the forces involved. When each wheel rotates, it applies a friction force to the ground, which moves the robot. When moving forward, both sets of left wheels rotate in the same direction at the same speed, and both sets of right wheels rotate in the same direction at the same speed, meaning that the forces do not oppose each other. However, when strafing, neither the two left wheels nor the two right wheels are rotating at the same speed. In many cases, they even rotate in opposite directions.

These two opposing forces cause the rollers to slip more and more, which angles the robot's velocity at the expense of traction (more slipping results in a loss of speed). However, the wheels do still slip when moving forward but not as drastically as they do when strafing.

This is the primary disadvantage to mecanum drivetrains: they tend not to have much pushing power and thus, are vulnerable to defense by a sturdy tank drive.

Due to the fact that mecanum wheels are more likely to slip because of the diagonal rollers, an optional addition to mecanum drives is a separate odometry mechanism in order to track the robot's location during autonomous.

Attention: It is important to note that in order to maximize the efficiency and stability of mecanum drives, when viewed from above, the rollers of each wheel should point towards the center of the robot, forming an X shape, rather than a rhombus.

The primary reason for this is that it allows the drivetrain to turn significantly faster than it would otherwise be able to. When using the suggested setup, when viewed from the robot's underside, the rollers form a rhombus. This allows the force applied by the wheels on the ground to act tangent to the turn radius, leading to faster turning.

See this video[34] and this other video[35] for a more in depth explanation.

Advantages

- Fantastic maneuverability and agility due to strafing, can avoid defense very well
- Good acceleration, can have high top speed
- Very versatile drivetrain for nearly any game

Disadvantages

- Suffers in traction, as mecanum rollers have a lower coefficient of friction than traction wheels; cannot traverse terrain
- Able to be pushed around on defense
- Wheels must be powered independently, so there is no redundancy

CAD Examples of Mecanum Drive (Click to expand)

goBILDA COTS

Click here to explore this model in OnShape online CAD[36]

REV COTS

Click here to explore this model in OnShape online CAD[37]

[34] https://www.youtube.com/watch?v=xgWf_t8owI0&t=3152s
[35] https://youtu.be/YJaX3vQ6kHw?t=123
[36] https://cad.onshape.com/documents/45549489f570f3694569a2df/w/85ff26b9fca4988ebc4df3b4/e/df9bcea72fcdd7e4aee4134b
[37] https://cad.onshape.com/documents/45549489f570f3694569a2df/w/85ff26b9fca4988ebc4df3b4/e/b8d4a6cfe4f26c5170f40a6c

goBILDA Custom

Click here to explore this model in OnShape online CAD[38]

REV Custom

Click here to explore this model in OnShape online CAD[39]

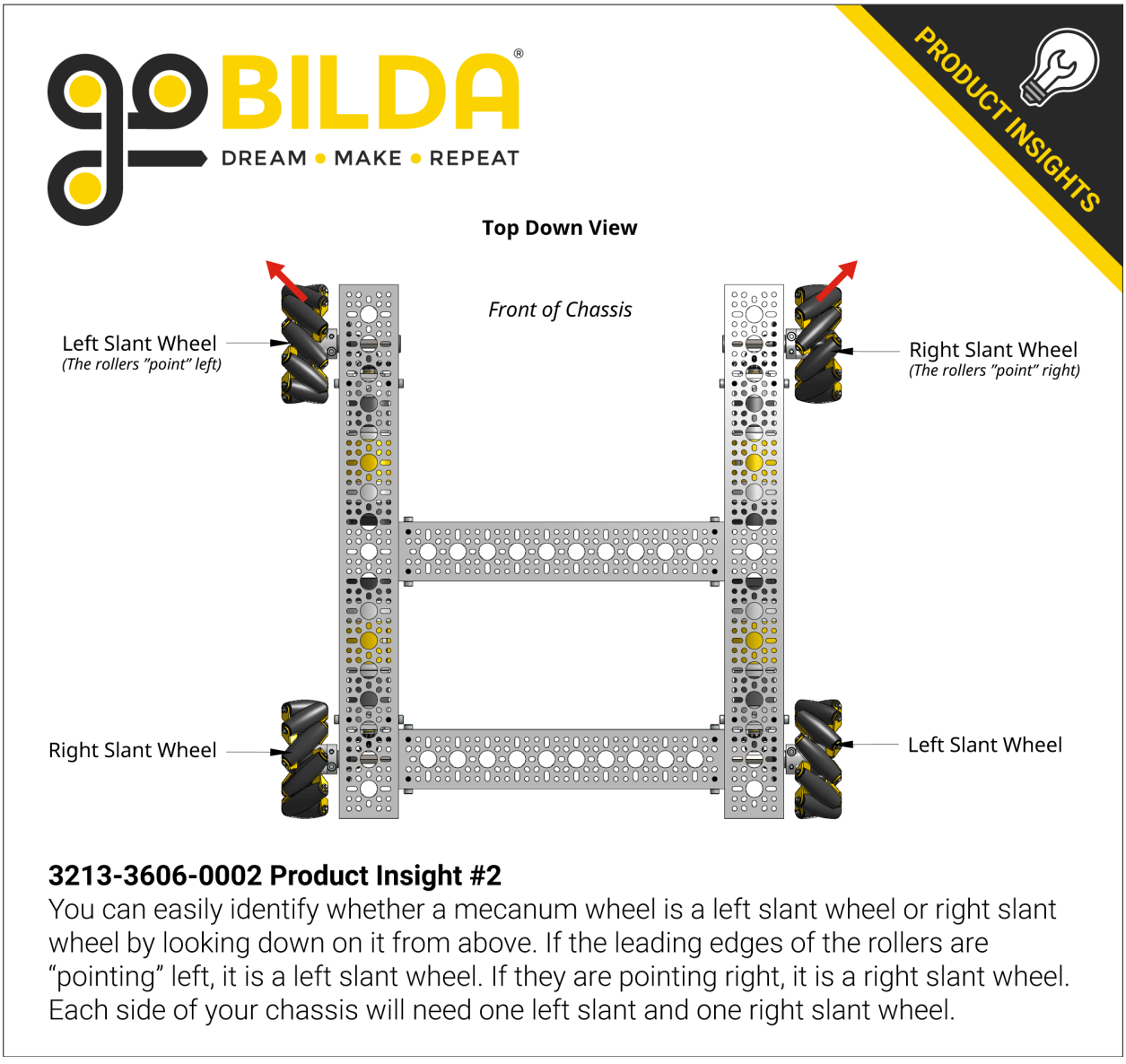


Fig. 9: Configuration for mecanum wheels, courtesy goBILDA

[38] https://cad.onshape.com/documents/45549489f570f3694569a2df/w/85ff26b9fca4988ebc4df3b4/e/cc2df2fbce0e9ca393696b54

[39] https://cad.onshape.com/documents/45549489f570f3694569a2df/w/85ff26b9fca4988ebc4df3b4/e/40c3266dab2e444011cce79c

Mecanum Wheels Miniguide

One of the most important features of a mecanum wheel is the mechanism that allows the roller to spin—either *bushings* or *ball bearings*. Mecanums which use ball bearings strafe better than those which use bushings, as the rollers can spin with less friction.

Note: In the past, some teams have invested in 6 inch diameter mecanum wheels. These are generally much more expensive and generally do not provide significant performance benefits. We highly recommend sticking with wheels that are between 3 and 4 inches in diameter.

Recommended

These wheels are the gold standard of mecanum wheels, if you are buying new mecanum wheels there is very little reason to buy any mecanum wheels not from this section.

- goBILDA Mecanum Wheels v2[40] ($127.49 with team discount): The goBILDA v2 Mecanum Wheels are the gold standard for teams wanting a quality design in all aspects. They feature a 96mm diameter and a thickness of 38mm, with ball bearing supported, 70A durometer rollers that provide excellent strafing performance. With the use of recessed 16x16mm and 32x32mm holes, these wheels have the largest amount of mounting options and can fit all shaft styles commonly used in FTC®.
- REV Robotics Mecanum Wheels[41] ($127.50 with team discount): The REV Robotics Mecanum Wheels come in at only 75mm in diameter, providing a smaller mecanum wheel option to FTC teams; however, they are not the thinnest option at 40.8mm thick. They feature ball bearing supported rollers that provide exceptional strafing capabilities and traction. The hole pattern featured on these wheels is compatible with Andymark Nubs as well as the REV Robotics Universal Hex Adapter v2 (which is included); however, in order to use other shaft options the use of an adapter or physical modification will likely be required.

Viable

These mecanums have acceptable performance but there is very little reason to buy them at this point, as they have been superseded by better ones. If you already own them, they are a viable option, but consider looking at some of the mecanums from the *Recommended* (page 104) section.

- Nexus Ball Bearing Mecanum Wheels[42] ($134.00): These wheels are 100 mm in diameter and 59 mm wide. They strafe excellently because of the use of ball bearings. However, they are more expensive, take up more space, have less traction, and are harder to mount to than the recommended wheels.
- goBILDA Mecanum Wheels v1[43] (discontinued): These wheels are very similar to the Nexus Ball Bearing Mecanum Wheels, but with a different color scheme and better mounting options.
- Andymark Heavy Duty Mecanum Wheels[44] ($225): These are easily the most expensive mecanums on the list. These wheels are 4” in diameter and 1.65” wide. These are bushing based mecanums, so while they strafe decently they still perform worse than bearing based ones. They have a good amount of traction, more than the Nexus bearing or bushing wheels.

[40] https://www.gobilda.com/96mm-mecanum-wheel-set-70a-durometer-bearing-supported-rollers/
[41] https://www.revrobotics.com/rev-45-1655/
[42] https://www.robotshop.com/en/100mm-mecanum-wheel-set.html
[43] https://www.gobilda.com/96mm-mecanum-wheel-set-70a-durometer-bearing-supported-rollers/
[44] https://www.andymark.com/products/4-in-hd-mecanum-wheel-set-options

- Nexus Bushing Mecanum Wheels[45] ($80.00): This is the Nexus Ball Bearing Mecanum Wheel with bushings instead of ball bearings. There is not much to say about them except that they strafe decently but worse than the ball bearing based equivalent.

Not Recommended

There is almost no reason to use these mecanum wheels—they perform very poorly and are not much cheaper than those in the *Recommended* (page 104) section. We cannot recommend the use of these on an FTC robot, if at all possible replace them with mecanums from the *Recommended* (page 104) section.

- TETRIX Mecanum Wheels[46] ($149.00): These mecanums are designed with a bushing based, hard plastic roller which in turn creates poor strafing performance. The integrated hub is a set screw based design with a round bore, causing unreliability as well as taking up extra space.
- Andymark Standard Duty Mecanum Wheels[47] ($77.00): These wheels barely strafe and are super fragile.

Fig. 10: 8103 Null Robotics, Rover Ruckus, **using Nexus mecanum bearing**

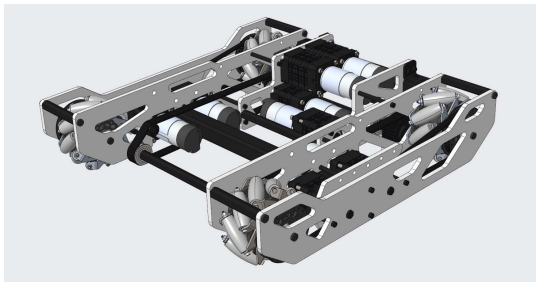


Fig. 11: 731 Wannabee Strange, Rover Ruckus, using **AndyMark HD mecanum wheels**

[45] https://www.robotshop.com/en/100mm-aluminum-mecanum-wheel-set.html
[46] https://www.pitsco.com/TETRIX-MAX-Mecanum-Wheels
[47] https://www.andymark.com/products/4-in-standard-mecanum-wheel-set

X-Drive

X-Drive is a holonomic omni-wheel based drivetrain. This type of drive involves mounting 4 omni wheels at the corner of the robot at a 45 degree angle.

One notable difference between X-Drive and mecanum is strafe speed. While, as mentioned in the mecanum section, the ratio of strafe speed to forward speed is noticeably less than 1, the ratio on an X-Drive is exactly 1 due to the rotational symmetry of the wheel placement. This means that an X-Drive bot's strafe speed and forward speed are equivalent. The drivetrains are slower, however, when strafing at 45° (approximately $\frac{\sqrt{2}}{2}$ of its forward speed).

Even though X-drive has good turning and acceleration, the main downside to the drive is packaging/form factor. Packaging refers to how easy/convenient the drivetrain fits into the overall design of the robot.

Ideally, the drivetrain should take up as little space as possible to make it easier to design mechanisms around. Because the omni wheels are offset, packaging a X-Drive is more difficult than other types of holonomic drive like mecanum or H-Drive. Also because of the strange packaging, it is relatively difficult to cleanly transfer power from the motors to wheels, meaning that most X-Drives end up being direct-driven, which is bad for the lifespan of the motor gearbox.

Note: When using X-Drive, the robot moves forwards/backwards/straight side-to-side $\sqrt{2}$ times faster than a drivetrain with wheels in the normal orientation (with the same gear ratio and wheel size).

For an explanation of why exactly this is, see this analysis[48].

Advantages

- Good maneuverability and agility
- Good acceleration

Disadvantages

- Prone to defense, pushed around easily
- Often uses direct drive due to awkward form factor

H-Drive

H-Drive (also known as U-drive, depending on the configuration) is a holonomic type drive that uses all omni wheels. H-Drive relies on a set of "strafer wheels" that are perpendicular to the forward/backward wheels to achieve strafing. H-Drive is similar to a fusion of a tank drivetrain while retaining the maneuverability and strafing of holonomic drivetrains.

H-Drive is theoretically very easy to code, but most teams employ some sort of gyro correction to strafe straight, although it is not necessary with proper weight distribution.

H-Drive has a number of possible motor configurations - 1 or 2 motors can be put on each forward drive pod, and one or two motors can be put on the strafe wheels. In the configuration with one motor on each forward drive pod, H-Drive has slightly reduced acceleration compared to mecanum drive.

[48] https://www.chiefdelphi.com/t/paper-mecanum-and-omni-kinematic-and-force-analysis/106153

Fig. 12: 731 Wannabee Strange, Velocity Vortex

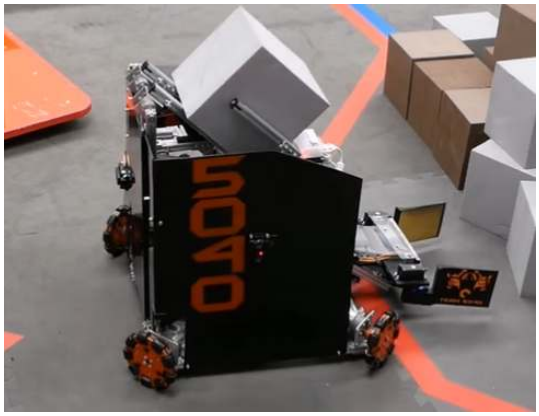


Fig. 13: 5040 Nuts and Bolts, Relic Recovery

For the highest possible reliability, many FRC® teams will suspend their strafe wheels on a rocker system to ensure that all wheels are in contact with the ground while the robot is not strafing.

By far the biggest advantage of H-drive is its ability to accommodate multiple motor distributions. For instance, if you want to dedicate only 3 motors to your mechanisms and you have a motor left over, using a 1 strafe motor, 4 drive motor configuration is absolutely viable. Or if you dedicate 5 motors for your mechanisms, H-drive with 2 drive motors and 1 strafe motor is definitely optimal.

Advantages

- Combines tank and holonomic drivetrain advantages
- Can be used with 3 or 5 motors
- Good traction and top speed
- Great maneuverability and agility

Disadvantages

- Strafing is slightly less effective than mecanum
- Complex suspension occasionally needed, depending on design

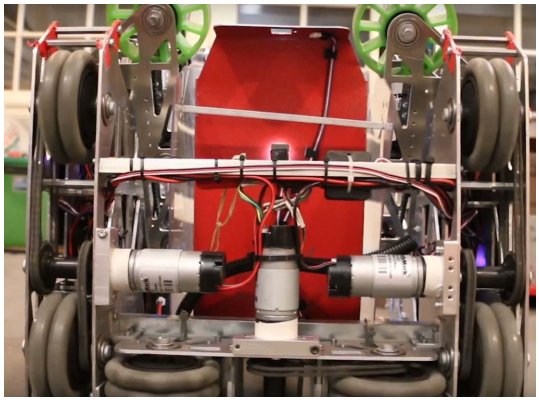


Fig. 14: 9804 Bomb Squad, Relic Recovery

Sample Drivetrains

In this section, we compile several sample drivetrains which you can use as a starting point for your own designs. You should also check the *gallery of robot designs* (page 375) in the Appendix.

goBILDA strafer chassis[49] Basic mecanum chassis by goBILDA. Simple and easy to build chassis, providing an excellent starting point for robots built from goBILDA parts.

goBILDA BeeLine Chassis[50] Basic 6 wheel drive chassis by goBILDA. Simple and easy to build chassis, providing an excellent starting point for robots built from goBILDA parts.

[49] https://www.gobilda.com/strafer-chassis-kit-v4/
[50] https://www.gobilda.com/beeline-chassis-kit/

Strafer chassis with Dead Wheels[Page 109, 51] Modification of goBILDA chassis by FTC® 9794 Wizards.exe, which adds *dead wheels*. Also see their video tutorials: https://www.youtube.com/watch?v=OjNvAD350M4&list=PLICNg-rquurYgWAQGhu6iC0At75vgqFJp

REV Robotics 6 wheel drivetrain[52] Sample 6 wheel drivetrain built using REV kit.

AndyMark Tile Runner chassis[53] A universal chassis kit sold by AndyMark. It is available in several modifications: 6 wheel, mecanum, tank tread. It is expensive, but using provided 3d models as inspiration for your own drivetrain is free.

Wizards.exe 6 wheel drive chassis[54] Another design by FTC 9794 Wizards.exe. Combines custom side panels with Actobotics parts (gears, chain sprockets).

Custom drivetrain by FTC 14875 LightSpeed[55] This design uses goBILDA mecanum wheels, motors, and some channels combined with custom side plates.

NX VANTAGE-H[56] Another custom drivetrain using goBILDA mecanum wheels, by redditor /u/nateless. Note that this design can be viewed but not downloaded—this is intentional; see his original reddit post[57].

8.2.4 Glossary

Omni Wheel Omni(directional) wheels, sold by many different vendors, are a special type of wheel that prioritizes mobility and strafing (moving laterally) over traction or front-back movement. They are similar to *mecanum wheels* in that omni wheels have rubber rollers that rotate perpendicular to the plane of the wheel.

Thus, the robot can move sideways (although the robot is not powered in the sideways direction). It is also utilized as a low-friction wheel in 4 wheel, 6 wheel, and 8 wheel drivetrains instead of having corner traction wheels.

Furthermore, X-drive utilizes four omni wheels, though traction is at a minimum.

A mecanum wheel is *technically* an omnidirectional wheel, but when generally referred to, an "omni wheel" has rollers rotated 90 degrees to the rotation of the wheel, where a mecanum wheel is generally 45 degrees.

[51] https://drive.google.com/file/d/1R85u8nGGmBu5_6jIztOH3-5_W4XK08Mb/view?usp=drive_open
[52] https://docs.revrobotics.com/duo-build/channel-drivetrain-build-guide
[53] https://www.andymark.com/products/tilerunner-options
[54] https://drive.google.com/file/d/1iu2UUNlqoQ6bS1vnoRPtUI0Uv3lLjNec/view?usp=drive_open
[55] https://drive.google.com/file/d/1iu2UUNlqoQ6bS1vnoRPtUI0Uv3lLjNec/view?usp=drive_open
[56] https://cad.onshape.com/documents/3d22333d5ba0abcc62edb57e/w/fa027f644666441544a378c6/e/693039a92658a00632996b28
[57] https://www.reddit.com/r/FTC/comments/c8vlsj/cad_for_nx_vantageh_is_going_public/

Traction wheel A traction or grip wheel is a wheel designed for maximum grip. It has an outer ring made of rubber, and its wide track ensures a larger contact patch with the ground. Traction wheels are commonly found in tank drivetrains. They are sold in different sizes and thicknesses by different manufacturers.

Strafing Strafing is the act of moving sideways or laterally (somewhat similar to drifting). It is possible with omni or mecanum wheels, and not possible with traction wheels.

Parallel Plate Drivetrain A parallel plate drivetrain is a drivetrain that has drive pods that consist of 2 plates spread apart with wheels and drive transmission in between them.

These plates can be anywhere from 1” to 5” apart, depending on the space requirements of the wheels and drive system. Generally, a pod width of 3” or less is desired to maximize the space between the drive pods for mechanisms such as an intake.

8.3 Power Transmission

When building any mechanism, it is important to consider how the mechanism will be powered, as gear ratio and form factor are two factors that affect both spacing and efficiency. There are four main forms of power transmission: direct drive, gears, chain, and timing belt. In addition, there are different ways of powering a mechanism that is mounted on an axle.

8.3.1 Motion Mounting

When mounting a mechanism, it is important to know how to transfer power to it. There are many ways of accomplishing this using an axle.

Live Axle

Term

Live Axle The simplest method of power transmission is called “Live Axle”. In this case, the mechanism is physically mounted on a powered axle: **when the axle turns, the mechanism turns with it**. The mechanism on the axle generally is mounted to a hub or pattern adaptor which transmits power from the axle. For this reason, live axle is generally recommended when powering multiple things together on one axle.

This method of power transmission is most commonly used on mechanisms that are mounted directly on a motor, as well as some mechanisms like intakes.

Advantages

- Simplest power transmission to build
- Buildable with any COTS system
- Can power several mechanisms on the same axle together easily
- Can be used directly off of a motor shaft

Disadvantages

- Can physically take up more space (bearings needed on either side of a shaft)
- Usually requires some form of *Shaft Retention* (page 112) to prevent the shaft from moving

Dead Axle

Term

Dead Axle Another form of power transmission is called "Dead Axle". In a dead axle setup, the mechanism is mounted on a fixed axle via bearings: **it is free to spin around the axle**. This method requires the power transmission to be physically bolted to the mechanism, because the axle itself will not rotate. For this reason, dead axle is generally recommended when powering one thing that is between two plates, as the axle itself can serve as a standoff to provide support between the plates.

Advantages

- Simple power transmission with custom building
- Axle can be used as a standoff support between different plates
- Can require less parts then a live axle setup

Disadvantages

- Harder to transmit power over a long shaft
- Usually limited to powering mechanisms between two parallel plates

Zombie Axle (Coaxial)

Term

Zombie Axle Zombie axle is when one shaft serves as a dead axle for one mechanism and a live axle for another mechanism. This means **one mechanism is mounted on the shaft via bearings, but the shaft is free to rotate independently to power a second mechanism**. This setup can be used to transmit power to two things using the same revolution point, making it easy to power mechanisms like arms

with intakes, or suspension drivetrain pods. Zombie Axle is only recommended when two mechanisms must be powered coaxially to each other.

Advantages

- Only power transmission method that can power two mechanisms on the same revolution point

Disadvantages

- Has the disadvantages of both live and dead axle
- More mechanically complex

8.3.2 Shaft Retention

If you are powering any mechanism with a *live axle* system, you will need a way to retain the shaft so that it does not move axially (along its length). It is necessary to constrain movement in both directions, and two methods can be mixed and matched on one shaft. Generally, this is done by having something contact the bearing's inner race that is constrained axially on the shaft. It is important to make sure the bearings themselves are also constrained.

Note: Make sure to use an appropriately sized shim/washer between the thing retaining the shaft and the bearing, such that it only touches the bearings inner race. Otherwise, the rubbing will lead to inefficiency.

Outlined below are some common methods FTC teams use for shaft retention.

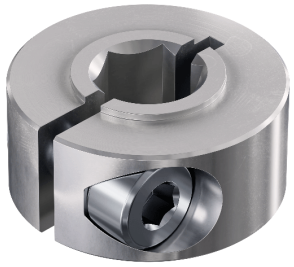
Shaft Collars

Shaft collars are fitted onto a shaft in order to secure it. There are two primary types: set screw collars and clamping collars.

Set Screw Collars

Set screw collars use a set screw (typically a grub screw) to tighten down onto the shaft. While set screw collars are cheap and readily available, there are some disadvantages. The screw digs into the shaft, creating a raised burr, making it difficult to remove/adjust the collar or to use the shaft for something else in the future.

Clamping Collars



A clamping shaft collar uses typical bolts to clamp the shaft, applying force all around the shaft instead of just in one place. They may or may not have a built-in shim to rest against a bearing race. They are generally recommended over set screw collars, especially for high-load applications. However, they do tend to be more expensive and larger.

Shaft Retaining Rings

Shaft retaining rings clip into a groove in a shaft and provide a simple and compact way to retain it. Unlike collars, they cannot be moved along the shaft and are reliant on having a machined groove in the correct location. Many teams use E-clips, a type of retaining ring, with goBILDA's 8mm REX shafting, as it can be bought with an E-clip preinstalled.

Bolts and Washers

When using a shaft with a threaded bore, you can use a bolt and washer on each end to retain it. It is important to use a thread locking compound to prevent the bolts from loosening over time.

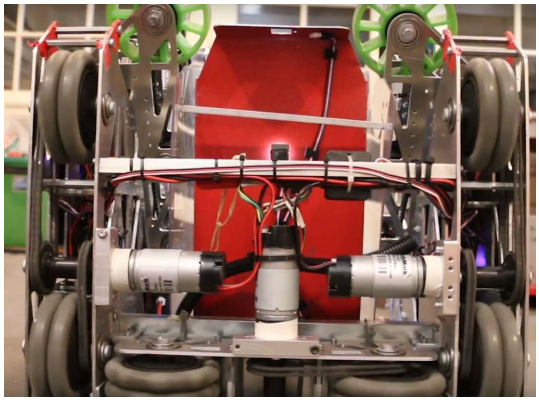


Fig. 15: 9804 Bomb Squad's Relic Recovery bot used bolts to retain its wheel shafts.

8.3.3 Direct Drive

Term

Direct Drive Powering a mechanical part (most commonly a drive wheel) directly from the motor axis. Many new and inexperienced teams will use this method to power their drivetrain as it is the simplest way to do so. However, there are significant drawbacks to this method.

Direct drive puts unnecessary load on the drive motor. This is because shock loads can destroy gearboxes, even *planetary gearboxes*. Gearboxes are able to withstand load along the axis of rotation, such as what occurs when the wheel changes direction. This is a normal situation of load. However, in direct drive, the gearbox shaft can be exposed to other shock loads outside of the normal axis. This happens when the wheel comes into contact with another robot or the field wall, which honestly happens more than you'd think. This can bend the motor shaft or permanently damage the gearbox.

It also limits the *gear ratio* to whatever ratio the motor gearbox is at. One advantage of all the other three transmission systems is the ability to gear up or down, based on a team's needs. Direct drive cannot do so, and if your drivetrain uses 40:1 gearboxes, there is no way to reduce this ratio to a faster 20:1 gearbox, for example.

Motor shafts are not built to carry large amounts of downward load. This can lead to bending of the motor shaft, since in direct drive the shaft is only supported by one side, the gearbox. Generally, a principle among all engineers is to support the shaft on both sides, which isn't possible in this case. This leads to *cantilevering* the shaft, something that should be avoided in general. Thus, with a heavy robot the motor shafts can easily be bent due to a lack of support. (While typically it is impossible to support a motor shaft on both ends, it is a common rule of thumb to have dual support on wheels or other non-motor shafts.)

Term

Cantilever A cantilever refers to when an object (usually a *shaft*) is only supported on one side. While this provides theoretically less support, as long as the shaft is still supported at two points by *bearings*

or *bushings*, cantilever is still a sound building technique. Many drivetrains are cantilevered, which provides for easy access to wheels.

Supporting a shaft on both sides is theoretically more structurally sound, although in most cases you won't notice a difference.

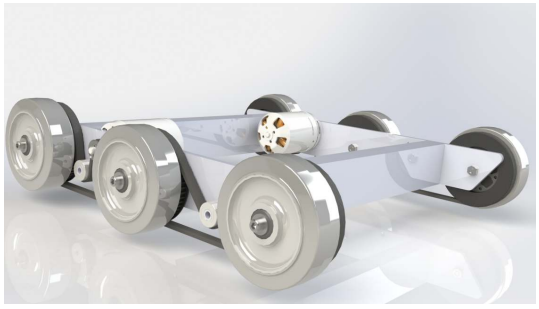


Fig. 16: Sanford's Prototype

Advantages

- Saves space
- Easy to build; most simple form of transmission

Disadvantages

- Prone to shock loads which destroy the gearbox
- Limited gear ratio to the motor itself
- Wears the gearbox of the motor faster
- Can bend the motor shaft

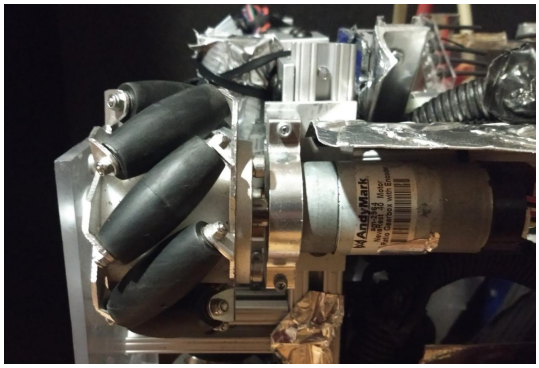


Fig. 17: 11316, Weapons of Mass Construction
Outside Supported Direct Drive (**Preferred Version of Direct Drive**)

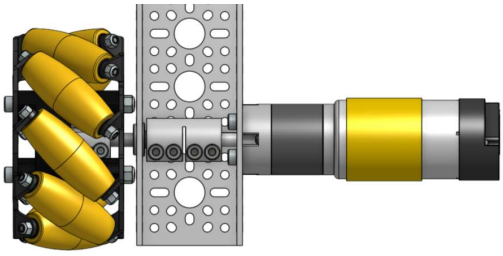


Fig. 18: Ethan Doak
Inside Supported Direct Drive (**Preferred Version of Direct Drive**)

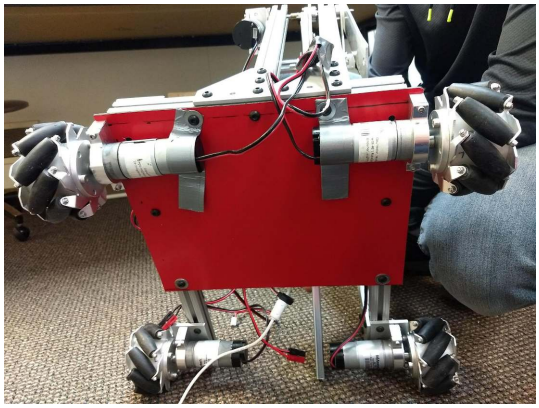


Fig. 19: 11316, Weapons of Mass Construction
Unsupported Direct Drive (**not recommended**)

8.3.4 Gears

Gears, like sprockets and pulleys, are used in power transmission for three common applications: changing the direction of power, changing the amount of torque, and changing RPM. Gears are a less common transmission option than chain, but are still very viable for most use cases. Gears are equally as reliable as chain, but can't be used for transferring power over long distances. Many teams dislike gears when using kit-based channels because the hole pattern limits which combinations of gears can be used. Consequently, it may be difficult to achieve a desired specific gear ratio. However, with *extrusion* systems, it's very easy to use different ratios, since the gears can be moved around in extrusion.

Term

Gear A gear is a machine part that has cut teeth, usually written in the form "numberT" (e.g. 32T, 86T). It is a form of power transmission that reverses the direction of rotation when used. The most common material for gears to be made of is aluminum or delrin plastic.

Fig. 20: 56T REV aluminum gear

Gears are made in different materials, with the most common being 7075 aluminum. **Never mesh plastic and metal gears together**. It is acceptable to mesh different types of metal gears together, as long as they have the same diametral pitch. It is advised to stay away from TETRIX aluminum gears as they wear down very easily. Some REV gears are made out of Delrin, a self-lubricating plastic. It is a durable material, but keep in mind that it is very possible to strip the bore using a plastic gear. Thus, we advise using the REV Hex Hub Strengthener to avoid stripping the bore on Delrin gears.

Bevel gears are a special type of gear that allows power transmission in two different planes. It is especially useful in tight spaces where a regular motor mounting position would not work.

Pitch Diameter Calculation

$$PD = Module * Number\ of\ Teeth$$
$$DP = \frac{Number\ of\ Teeth}{PD}$$
$$PD = OD - (2 * Module)$$

Pitch circle refers to an imaginary circle that contacts the pitch circle of any other gear with which it is in mesh. Basically, each gear has a pitch circle. When gear 1 is meshed with gear 2, the pitch circles of both gears should touch exactly in the middle of where the teeth interlock with each other.

Meshing Gears

Term

Mesh Meshing refers to the overlapping contact between a gear tooth and another gear tooth, chain and sprocket, or belt and pulley.

A proper mesh is essential to ensure maximum torque transmission. Too little mesh can result in no power transfer, derailment or gears grinding/wearing down faster. Too much mesh can produce unwanted friction and introduce inefficiencies within the drive system.

When meshing gears, it is important that the gears are not too loose nor too tight. If the gears are too loose, the teeth will easily wear out, decreasing its longevity. If the gears are too tight, however, they will have too much friction and possibly grind or bind up. The ideal way to mesh gears are to make sure the teeth interlock and just touch the base of the gear.

Attention: If possible, it's best to avoid meshing gears with a clamping motor mount – due to the sensitivity of the mesh, even the slightest movement of the motor inside the clamping mount can cause the gears to slip or damage each other.

Calculating center-to-center distances using gears is quite simple. In order to calculate the desired center distance between two given gears, you must know the number of teeth for each gear and the **diametral pitch** of your gears (the number of teeth per inch of the gear's diameter). With these two pieces of information, you can use the equation $D = \frac{T_1+T_2}{P}$. In this equation, D is the distance between the center of both gears, T_1 and T_2 are the number of teeth of each gear in question, and P is the diametral pitch of the gears.

The **module** (abbreviated **MOD**) of a gear is used similarly to diametral pitch. It is the number of millimeters of the gear's diameter per tooth of the gear. The equation to find center distance D is $D = \frac{(T_1+T_2)*M}{2}$, where T_1 and T_2 are the number of teeth of each gear in question, and M is the module of the gear.

Note: Be sure to never *mesh* gears that are not of the same diametral pitch. (A notable exception is 32 diametral pitch and 0.8 MOD gears. These are close enough to be perfectly fine.)

Additionally, it is possible to average the pitch diameters of the two gears to find the correct center-to-center distance.

As with sprockets, it is important to line up the gears so that they do not accidentally slip. Especially when using extrusion, it is possible that the gear may not be parallel to the extrusion, as the two supporting ends

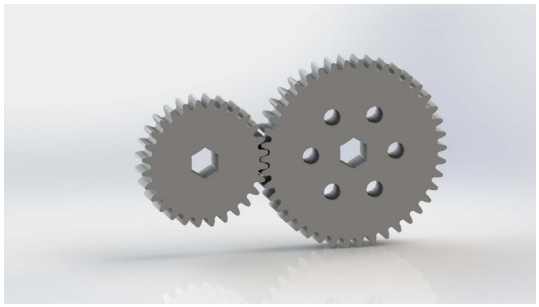


Fig. 21: Correct gear mesh

may not be perfectly in line with one another. It is imperative that the gear be lined up as straight as possible to prevent damage or gear binding.

It is highly recommended to use white lithium grease or a similar lubricant between the gears to reduce friction and possible binding.

Advantages

- **Gears are a solid and proven power transmission method**. Early examples of gears date back to the 4th century BC, so you're using technology with millennia of development behind it. When it comes to gears, there's not much that we haven't figured out.
- **Gears are simple to use with both channel and extrusion**. On channel, your gears are already spaced correctly - you just need to choose the right pair of gears. Extrusion gives you even more flexibility - just slide your gears into mesh, and you can have whatever ratio you want.
- **Gears can give you big reductions in small areas**. Depending on the gear combination, one can achieve big ratios in reduction in very small spaces (for example, a 10 tooth gear and a 100 tooth gear will take much less space than a 10 tooth sprocket and a 100 tooth sprocket).
- **Gears require no tensioning: once the spacing is correct, the gears will operate quickly**. Unlike chain or belt, there is nothing further transferring the power, which cuts out the need to properly tension chain or belt. This of course has the drawback of not being able to transfer power far distances.

Disadvantages

- **Sometimes, the ratio you want might not be easy to build**. Channel spacing limits gear ratios, but this can be circumvented with compound ratios and a bit of creativity.
- **Long distance power transfer is impractical with gears**. If you need to transfer power long distances, gear combinations can become complicated very quickly, so belt/chain is preferrable.
- **Meshing gears can be tricky**. It's only made worse by the sensitivity of a gear mesh. However, channels do solve this problem, providing pre-spaced holes to easily mesh your gears. Do keep in mind that gear mesh may not be perfect, even with channel.
- **Gears usually wear faster than sprockets** if there is too much friction between the gears. Teams can use white lithium grease or similar lubricant to help remedy this problem.

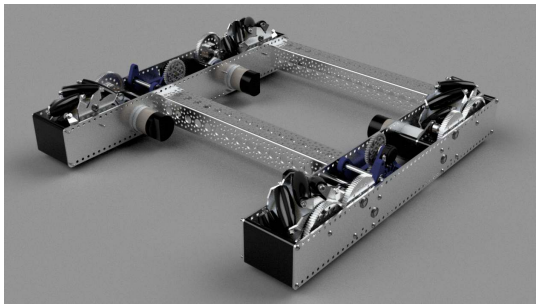


Fig. 22: 11115 Gluten Free gear-based drivetrain

Fig. 23: 13075 Coram Deo Robotics, Rover Ruckus gear-based drivetrain

8.3.5 Roller Chain

When your shafts aren't right next to each other, roller chain and sprockets will allow you to transmit power securely between your shafts.

Term	
Chain	Roller chain is made up of a series of links joined by pins. Each link can rotate around its pins, creating a dynamic loop that can conform to any shape. The pins in the chain engage the gaps between teeth on each sprocket. Chain number refers to the type and size of chain that is compatible with the sprocket. #25 chain and 8mm chain are both commonly used in FTC®.

If you've ridden a bike, chances are that you've already seen roller chain - the chain on your FTC robot is similar, but it's probably a different *pitch* (different size). Chains most commonly used in FTC are #25 (1/4 inch pitch) or 8mm pitch.

When using chain, often there is a master link. This is a special type of link that has a removable end capsule in order to shorten the chain. However, as it is removable, it is not a very reliable chain link and can loosen and fall off under prolonged usage. There have been teams who have had master chain links fail during competition, costing them a match in the elimination rounds.

A chain breaker eliminates the need for master links because it can break and join chain at any point.

Attention: It is highly recommended that teams purchase a chain breaker (we recommend the DarkSoul chain breaker for #25 chain, and the goBILDA Chain Breaker for 8mm pitch chain) instead of using master links, which are prone to failure.

Fig. 24: The removable master link is shown on the right.

Center-to-Center calculations

The equation to calculate *center-to-center* for chain is quite complicated. Many online calculators[58] can calculate C-C distances without going through the tedious calculations. However, the complete formula is below.

$$C = \frac{P}{8} * (2L - (N+n) + \sqrt{(2L - (N+n))^2 - \frac{8}{\pi^2} * (N-n)^2})$$

$$L = \frac{2C}{P} + \frac{N+n}{2} + \frac{P(\frac{N-n}{2\pi})^2}{C}$$

- C = center-to-center distance, inches
- L = chain length in pitches
- P = pitch of chain
- N = number of teeth in large sprocket
- n = number of teeth in small sprocket

Chain Wrap

Chain should, at the very least, have 90 degrees of contact with the sprocket. The best practice is to have 180 degrees or more of contact, as it is very unlikely to fall off with proper tensioning. Chain skipping, especially on drivetrains or arms, is very possible without proper chain wrap or tensioning.

When tensioning chain, be sure to not undertension or overtension chain. Undertensioning chain can result in the chain falling off the sprocket or chain skipping, where the chain can skip along the sprocket. Overtensioning the chain often results in the motor burning out, or less seriously, a loss of efficiency. Push along the chain run, and if the chain moves slightly without significant resistance, chances are you've done it correctly. If it's too tight, then the chain will barely move under a gentle press.

[58] https://reca.lc/chains

Best practices for wrap

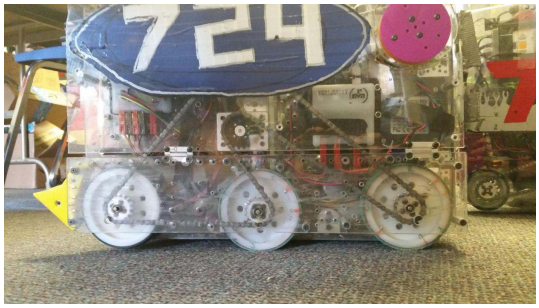


Fig. 25: 724 Rednek Robotics Wun, Relic Recovery

Fig. 26: 8103 Null Robotics, Rover Ruckus

Advantages

- **Chain can take a beating**. No matter what your application is, metal chain is usually up for the challenge. #25 chain can hold up to 930lbs before breaking, and there's nothing you'll do in FTC that will exert that force. (If your chain does break, it's most likely due to a faulty Master Link or sprockets that are not correctly aligned.)
- **Chain can be however long or short as you wish**. If your ratio changes or your shafts move, it's easy to adapt your chain run - just break the chain and put it back together at its new length. You can often do this without even removing the chain from your robot.
- **Chain can be pretty precise**. When properly tensioned, roller chain doesn't have very much slop. However, you really need to get your chain tension right to reduce slop, and you'll probably want an adjustable tensioner for when the chain stretches. This can be done easily if using extrusion systems, as the sprocket can be adjusted for tension.

Disadvantages

- **Chain stretches over time**. As it's used, the connections between the links and rollers can stretch a bit. While it doesn't look like much, this stretching can introduce lots of slop into your chain run and even derail it in some cases. You'll most likely need an adjustable tensioner to keep your tension over time - some teams have used spring-loaded dynamic tensioners to automatically compensate for any changes.
- **The smaller the sprocket, the faster the chain stretch**. This is because when the chain is run on a smaller sprocket, more force is applied due to a smaller radius.
- **Chain wrap, especially in one chain run, can be problematic**. Typically, teams use either one or two chain runs (pieces of chain) per drivetrain side. However, one chain run can require more than one idler sprocket and get very complex in order to maintain proper chain wrap.
- **Sprockets are really big**. If you want a really high reduction using chain, you'll pay for it in the space that it takes up. Sprocket teeth are much larger than gear or pulley teeth, so your reductions are going to be much larger.

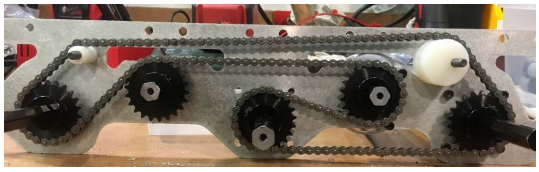


Fig. 27: 7244 OUT of the BOX Robotics, Relic Recovery

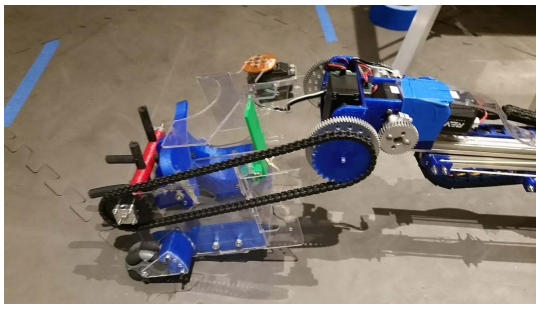


Fig. 28: 9794 Wizards.exe, Rover Ruckus, plastic chain on intake

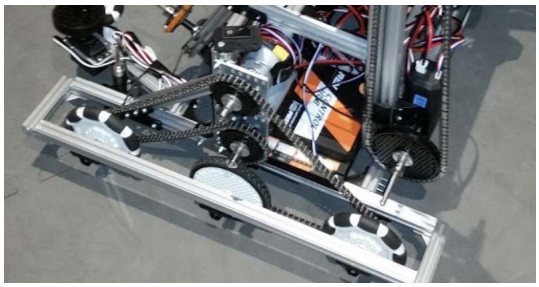


Fig. 29: Properly done chain wrap with the REV system

8.3.6 Timing Belt

When you think of a belt, you're probably thinking of a very important men's fashion accessory. However, there's another type of belt, and it's way more relevant to robotics - the timing belt. If you've ever tinkered with the insides of a car before, you probably recognize timing belts as an important component designed to keep everything under the hood in sync.

Term

Timing Belt Timing belts use a series of small, wide teeth to engage a pulley with a number of matching grooves. They earn their name because they can be very precise, transmitting power with virtually no slop and ensuring a snug connection between shafts.

Fig. 30: Timing belts and a pulley

While a timing belt may complete a similar objective to *chain*, its characteristics and strengths are very different. Timing belts are lighter and more compact than chains, but they lack the customizability of their bulkier brother - belts come in a closed loop of predetermined length, and there's no changing that length on the fly.

Like chain, belt is identified by its *pitch* - common pitches found on FTC® robots include HTD 5mm, HTD 3mm, and GT2 3mm.

When using timing belts, correct tension is very important. There are two main ways to get your tension right. The first is easy - goBILDA and Actobotics already have belts integrated into their hole patterns. You can buy correctly sized belt directly from each vendor, and your tension will be perfect as soon as the belt is installed. As your designs gain complexity, so will your belt runs - maybe there are more than 2 pulleys, and maybe your pulleys are all different sizes. To compensate for this, the second way to ensure tension is to use a dynamic tensioner, similar to those found in complex chain runs. To design for these tensioners, we recommend planning more complex belt runs in CAD before building them in real life.

Belt Calculator

The actual calculations to determine which belt to use to get close to a given center-to-center (*C2C*) distance are complicated. Below is a calculator to help out:

Belt Calculator

The web version of gm0 has a belt calculator available here.

> **Warning:** This calculator will suggest belt lengths even if they are difficult to get from vendors. Make sure the C2C distance you design around is for a belt length that you can purchase.

SDP-SI has a more advanced calculator[59], as does ReCalc[60]. The equations for calculating these values by hand can be found in SDP-SI's Designing a Miniature Belt and Pulley Drive System Design Guide[61].

Belt Wrap

Belt should, at the very least, have 90 degrees of contact with the pulley. The best practice is to have 180 degrees or more of contact, as it is very unlikely to fall off with proper tensioning. Belt skipping, especially on drivetrains or arms, is very possible without proper belt wrap or tensioning. When tensioning belt, be sure to not undertension or overtension it. Undertensioning belt can result in the belt falling off the pulley or belt skipping, where the belt can skip along the pulley. Overtensioning belt often results in the motor burning out, or less seriously, a loss of efficiency. Push along the belt, and if it moves slightly without significant resistance, chances are you've done it correctly. If it's too tight, then the belt will barely move under a gentle press.

Best practices for wrap

Fig. 31: Ethan Doak, goBILDA

[59] https://sdp-si.com/tools/center-distance-designer.php
[60] https://www.reca.lc/belts
[61] https://www.sdp-si.com/Belt-Drive/Designing-a-miniature-belt-drive.pdf

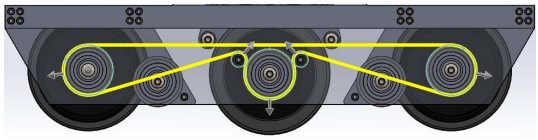


Fig. 32: 8103 Null Robotics, offseason prototype, properly done belt wrap with tensioners

Advantages:

- **Pulleys can be made at home**. Pulleys can be 3D printed for most situations, allowing you to cut costs and create unique tooth counts easily.
- **Belts are very strong**. They're reinforced with fiberglass cords that are incredibly hard to break, giving belts immense strength. (*If you break a belt, it's most likely because it was out of alignment or tensioned far too tightly*.)
- **When tensioned correctly, there is absolutely no slop**. Engines use timing belt for a reason - because it's the best possible solution for them to perfectly synchronize their shafts. There's nothing that matches the rotational accuracy of a properly tensioned belt.
- **Belts are efficient and quiet**. Compared to the loud shredding sound of a chain run, belt runs are dead silent, and they're more efficient than chains (although this makes zero practical impact in the robotics use case).

Disadvantages:

- **Belts aren't customizable**. You buy a belt of a specific length and you're stuck with that length until you buy another one. This isn't too bad if you're planning out your robot properly, but chain will work better for prototypes where the chain length will be changing often.
- **Belts can be wider than alternatives (especially chain)**. This probably won't have much of an impact, but belt can often be wider than other power transmission methods, so it may not always fit.
- **Belts can be expensive (but you'll save money with pulleys)**. While you can buy chain 10 feet at a time, you'll most likely be buying each belt brand new. While this can get expensive, you'll be saving money on pulleys.

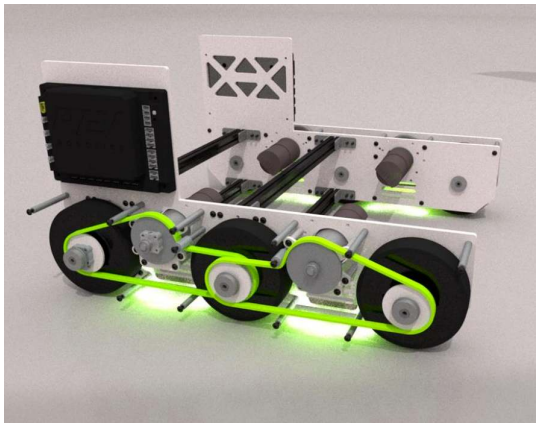


Fig. 33: 7236 Recharged Green, Rover Ruckus

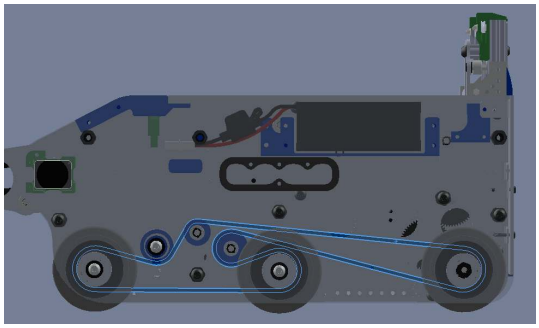


Fig. 34: 8417 Lectric Legends, Rover Ruckus

Purchasing Belts

Once you know what pitch and length belt you want, purchasing a belt requires navigating the vendor's website. Almost every COTS belt vendor will identify belts by 3 values: pitch, pitch length, and width. Pitch is the pitch of the belt, pitch length is the length of the belt (belt teeth times pitch), and width is the width of the belt. From there, its a matter of searching the vendor's page for the right belt.

SDP-SI

SDP-SI is an established mechanical parts vendor that sells a large assortment of COTS belts. To navigate the site, go to the main belts page[62], navigate to the correct belt type, then select the option for the correct pitch.

Important: Make sure to select the Single Sided belt type, and not the Timing Belt Stock type. Timing Belt Stock is a long stretch of belt that is not in a loop, whereas Single Sided belts are a loop.

For example, to purchase an HTD-5 belt, first navigate to GATES HTD Timing Belts, navigate to 5mm pitch, and select buy now under the single sided belt option.

V-Belt Guys

V-Belt Guys stocks a large number of options for belts. They are known for stocking almost every width of belt, since they cut belts to size for the order. Searching the site uses belt codes with the format "Pitch Length-Pitch Code-Width". For example, a 200 pitch length HTD5 belt with a 5mm side to side width would be the code 200-5m-5. A list of common pitch codes is below.

Name	Pitch Code
HTD-5	5m
HTD-3	3m
GT3-2mm	2mgt
GT3-3mm	3mgt
GT3-5mm	5mgt

[62] https://www.sdp-si.com/products/details/timing-belt-detail.php

8.3.7 Power Transmission Glossary

C2C Center to center (C2C) refers to the distance between the centers of a pair of *sprockets*, pulleys or *gears*. This will affect *chain*/*belt* tension and gear meshing, so calculating this correctly is essential.

Clearance Diameter The clearance diameter is the diameter of the imaginary circle that contains the entire pulley, sprocket, or gear. For pulleys and sprockets, this includes the belt or chain on the component. Clearance diameter is used to check for interferences with other mechanisms. It is usually larger then *Pitch Diameter (PD)* and *Outside Diameter (OD)*

GT2 Belt GT2 belt is a type of synchronous timing belt commonly used on drivetrains and other mechanisms. Its available in different widths and pitches, although the most common is GT2 3mm (3mm *Pitch*), compatible with REV Robotics *COTS* Pulleys. See *Timing Belt*

HTD Belt HTD belt is a type of synchronous timing belt commonly used on drivetrains. It is available in different widths to accommodate different sized pulleys. The most common is HTD3 (3mm *Pitch*) and HTD5 (5mm *Pitch*) belts, as these are compatible with goBILDA *COTS* pulleys. See *Timing Belt*

Idler An idler *gear*, *sprocket*, or pulley is one that is purposely not used for driving anything else on the *shaft*. The purpose of this idler is, in the case of gears, to transfer power to another direction.

For *chain* and *belt*, idlers are more common, and are usually adjustable to maintain tension.

Loctite Loctite is thread locking fluid used so that bolts do not come loose under use and vibration. Loctite should be applied to the threads of the bolts. There are two types of Loctite: blue, which is removable, and red, which is permanent (and we mean it).

Note: It is highly recommended that teams use Loctite on all motor and *servo* mounts, as well as any mechanism prone to vibration.

Danger: **THE BOTTLE COLOR AND THE FLUID COLOR ARE REVERSED.** When we refer to the "color", we mean the fluid color. Blue loctite usually comes in a red bottle.

Fig. 35: Blue Loctite (removable, in red tube), Red Loctite (permament, in blue tube)

Outside Diameter (OD) The diameter of the imaginary circle traced by the outermost face of the teeth on a gear, sprocket, or pulley. This will be larger then the *Pitch Diameter (PD)* for a gear, but smaller then the *Pitch Diameter (PD)* for a pulley or sprocket.

Pitch Pitch refers to the distance between the center of one tooth of a gear, sprocket, or pulley to another. In chains it refers to the distance from one pin to another, and in belts it refers to the distance betewen one groove to another.

Pitch Diameter (PD) An imaginary circle around a component used for various calculations. For gears, it is the imaginary circle that mates with any other gear's pitch diameter when the gears are properly spaced. For chain and belt, it is the circle which is traced by the middle of the belt or chain as the pulley or sprocket rotates. For gears, the pitch diameter will be smaller then the *Outside Diameter (OD)*, but with chain and belts, the pitch diameter will be larger. $PD = (tooth * pitch)/\pi$

Sprocket A sprocket is a mechanical part that transfers power through its cogs, which fit into chain. It is similar to a *gear*, except that instead of meshing with another gear, the sprocket meshes with chain. See *Chain*

Fig. 36: Delrin 20 Tooth #25 sprocket

8.4 Linear Motion Guide

Linear motion is one of the most important components of a successful robot. In most games, teams are required to reach into an area that the drivetrain cannot access in order to pick up or deposit game elements. For example, in 2017-2018, Relic Recovery, teams needed to extend an arm of some sort to grab the Relic that was placed in the corner of the playing field. This area of the game field was nearly impossible for a robot to drive to, so a linear extension was necessary.

Access of > 18" is necessary for nearly all games; >24" is preferred. In some games, an extension of 36" or more may be needed.

Note: It is possible to achieve extension with an arm, but since this guide is geared toward newer teams, linear extensions should be prioritized over arms. Refer to the *Arm Guide* (page 160) for more information.

8.4.1 Drawer Slides

You've *definitely* used a drawer slide before - at least two of them are mounted to almost any drawer that you've opened. Teams use these drawer slides for linear motion, often stacking them using 3D printed spacers to achieve plenty of extension.

These slides are available from a number of different vendors, and come in many varieties, so choosing the right slide can seem overwhelming. Steel drawer slides are common, but can be hard to mount, as they aren't made to be stacked. Aluminum drawer slides, such as the MiSUMI slides, are generally the best option for teams.

We recommend the Viper Slides or MiSUMI slides for newer teams. Viper Slides are cheaper and interface better with goBILDA products, while MiSUMI slides interface better with REV products. Viper slides are also available in a complete kit[63]

Note: When running two sets of drawer slides together, they should be mounted to face each other in order to reduce sag. This is especially important for horizontal extensions.

Fig. 37: 7236 Recharged Green, Rover Ruckus

On 7236's robot, the slides are not facing each other. The sag caused by this is largely mitigated by supporting the extension at the end with omni wheels.

Fig. 38: 12791 Iterative Intentions, Power Play

On 12791's robot, the slides face each other, which decreases sag in comparison to other orientations.

Listed below are the recommended drawer slides.

[63] https://www.gobilda.com/2-stage-viper-slide-kit/

Steel-rolled cabinet drawer slides

Available from your local hardware store, steel slides aren't a bad option for FTC® teams, as they are heavy-duty enough for most use cases. However, these kinds of slides are much heavier than other aluminum slide options. Furthermore, these slides are not designed to have bearings or a second slide attached to them, because they only contain mounting for a standard drawer. Thus, these slides require drilling holes in order to mount the necessary parts for linear extension.

Advantages

- Commonplace at any hardware store
- Not very expensive

Disadvantages

- Heavier than other slide options (steel as opposed to aluminum)
- Sliding is usually good but not great
- Hard to adapt to building systems
- 3D printed spacers may be required

MiSUMI Telescopic Slide Rails

The MiSUMI slide rails[64] are preferred by many top-tier teams because they are sturdy, very reliable, and ridiculously smooth due to the ball bearing system. **MiSUMI slides are able to withstand a significant amount of load with little flex**.

They are also low-profile, and have a M3 mounting pattern, meaning they are easy to attach to REV components. However, MiSUMI slides have a slightly higher price point, and it is often difficult to attach one slide to the next. An easy solution is to attach the end of one slide piece to REV extrusion, and do the same with the next slide. Then attach the REV pulley bearing on the top of the extrusion piece for the string to run through.

To save space, some teams have 3D printed an insert that goes between each slide instead of using the 15mm extrusion piece. In order to attach the slides to anything, teams will need to purchase **countersunk M3 screws** from McMaster-Carr. For attaching to REV extrusion, buy 6mm M3 screws with the M3 nut (**not locknut**) to insert inside the extrusion. As a tip, try to protect chips or sawdust from falling into the slides, as the sliding will have a noticeable difference.

MiSUMI sells two different types of slides: SAR2 and SAR3. The SAR2 is a two-piece slide, while the the SAR3 is a three-stage slide (has intermediate slider to increase the extension of the slide). Teams have used both successfully, and there isn't neither option is clearly superior.

Teams using SAR3 slides will generally need to buy low profile M3 jam nuts from McMaster-Carr to connect the slides together. These nuts fit inside the slides with a tiny bit of clearance when tightened.

Rail length Options:

- 200 mm, part number SAR220 (SAR2) or SAR320 (SAR3)
- 300 mm, part number SAR230 (SAR2) or SAR330 (SAR3)

[64] https://us.misumi-ec.com/vona2/detail/110300072130/?HissuCode=SAR240

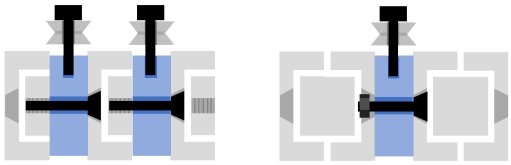


Fig. 39: Left: cross section of a 3 stage lift using 3D printed inserts (blue) and SAR2 slides; right: cross section of a 2 stage lift using a 3D printed insert and SAR3 slides. The left approach is also used for Long Robotics slides (discussed below).

- 400 mm, part number SAR240 (SAR2) or SAR340 (SAR3)

Advantages

- Best slide smoothness due to ball bearings
- Very little slide flex, robust build quality
- Can handle relatively heavy load use cases (within reason)
- Compatible with REV 15mm extrusion system

Disadvantages

- Not cheap
- Tricky to adapt if not using REV
- 3D printed spacers may be required
- Steel ball bearings wear into the aluminum rails over time, introducing play

Viper Slides

goBILDA Viper Slides[65] are a relatively new addition to the goBilda ecosystem. They are COTS steel drawer slides that can be purchased in a kit with all the parts needed to assemble and rig them right away[66].

They use M4 mounting bolts, and are on the standard 8 mm goBilda pattern.

Advantages

- Slide are smooth due to ball bearings
- Compatible with goBILDA ecosystem
- Steel slides can handle more load (can use one set of slides instead of two in many situations)
- Available in a kit with everything you need to start using in one purchase
- Longer stroke (more extension) then the same length misumi slide
- Cheaper then misumi slides

[65] https://www.gobilda.com/steel-viper-slide-14-ball-carriage-336mm-length-244mm-travel/
[66] https://www.gobilda.com/2-stage-viper-slide-kit/

Disadvantages

- Twice as heavy per slide compared to misumi slides
- Recommended to mount slides to channel to prevent bending

Long Robotics Slides

> **Warning:** Due to the tendency of the endstops to fail, as well as inconsistent jamming issues on the slides, Long Robotics slides are no longer recommended to purchase

The Long Robotics[67] slides are also used by teams because they are almost as smooth as MiSUMI slides due to the ball bearing system but slightly cheaper.

They utilize M4 mounting bolts, and can mount directly to goBILDA channel. They are virtually identical to the SAR2 series of MiSUMI slides.

The manufacturer website has CAD files for 3D printed inserts that go between the slides, which V-bearings are mounted to. V-bearings are available from Long Robotics[68]. To attach slides, one will need to purchase **countersunk M4 screws**, which are also available from Long Robotics[69].

These are available in both a 300 mm rail length option[70] and 400 mm rail length option[71].

Advantages

- Slide are smooth due to ball bearings
- Compatible with goBILDA channel
- Cheaper than MiSUMI slides

Disadvantages

- 3D Printed spacers are almost required; while other solutions exist, 3D Printed spacers are the simplest and lowest-risk
- Can be difficult to mount to kit systems which aren't *extrusion* based or don't have holes spaced at 8 mm apart
- Steel ball bearings wear into the aluminum rails over time, introducing play
- The endstops on the slides have been known to fail, spilling ball bearings out of the slide onto the field
- Slides have been known to seize and jam on occasion

[67] https://longrobotics.com/
[68] https://longrobotics.com/product/3x12x4mm-v-bearing-10-pack/
[69] https://longrobotics.com/product/6mm-d-low-head-10mm-m4-screw-10-pack-t10-torx-drive/
[70] https://longrobotics.com/product/lrs-300-aluminum-slide-300mm-black-anodized/
[71] https://longrobotics.com/product/lrs-400-aluminum-slide-400mm-black-anodized/

8.4.2 Extrusion Slides

Extrusion slides are made up of a stack of extrusions that extend by sliding along each other. There are two ways to make this happen: bushings or V-wheels. Bushing slides connect the two slides with two self-lubricating plastic pieces that slide smoothly along the slots in the extrusion. V-wheel slides have V-shaped groove bearings on both sides of the extrusion that bite intro grooves on the extrusion, allowing the stages to slide smoothly.

REV, Actobotics, and goBILDA all sell extrusion slide kits that integrate nicely with existing FTC® kit parts. Beyond this, OpenBuilds sells a V-wheel extrusion slide kit suitable for heavier loads, and Misumi offers a few different sizes of bushing-based extrusion slides. REV furthermore sells a 8020 V-groove bearing slide kit for FRC®, that is not recommended for FTC use cases.

Note: We recommend that newer teams using extrusion slides stick to kits designed for FTC instead of making their own.

REV Robotics 15mm Linear Motion Kit

The REV 15mm Linear Motion Kit[72] is based on the 15mm extrusion system. **This extrusion kit does not perform well without modification.** This has been partially remedied by REV as they have developed a second iteration of their slide kit, which has much better tolerances on the Delrin sliders.

Still, you'll see some competitive teams use this kit with multiple modifications, such as adding lots of lubricant and mounting the sliders differently. Teams have also 3D printed their own sliders, though this is not a great idea for teams without much 3D printing experience. One of the biggest issues with the stock REV kit is the tendency of the slides to bind. Additionally, since the only thing attaching one extrusion to another is the plastic slider, the REV slides are not particularly sturdy, and require crossbeams to keep alignment.

Overall, this kit is lightweight, simple, and cheap. It can be a good start for teams using REV and needing a linear extension, and is generally usable out of the box. However, it is not very smooth and only achieves its maximum potential when modified and tweaked.

Advantages

- REV has complete guide on how to rig the linear slides
- Easily interfaces with REV building system
- Lightweight, should be used for light/medium loads only
- If tweaked, can be a very efficient linear slide

[72] https://www.revrobotics.com/rev-45-1507/

Disadvantages

- Does not perform well out of the box
- Can flex under load, needs additional support
- May need some modifications such as custom sliders

Fig. 40: 11115 Gluten Free, Rover Ruckus, custom REV slides

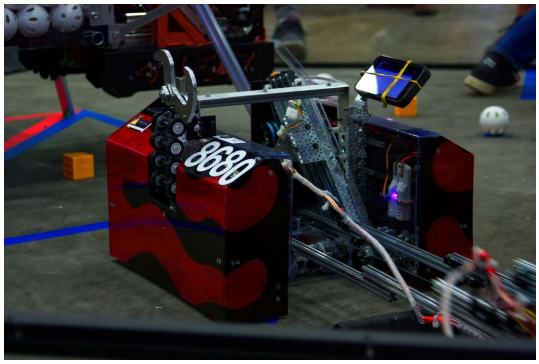


Fig. 41: 8680 Kraken-Pinion, Rover Ruckus, REV slides

OpenBuilds Mini V Gantry Kit

The OpenBuilds Mini V Gantry Kit[73] is designed for use with OpenBuilds' 20mm V-slot extrusion, which can be mounted to other build systems using M3 screws and sliding T nuts. OpenBuilds slides are used for high precision applications such as 3D printers and CNC machines and have a **much higher load capacity** than REV and Actobotics slides. OpenBuilds slides are also much heavier than REV and Actobotics slides, but because they are so strong they can be used in applications where two sets of lighter duty slides would otherwise be required. Out of the box, OpenBuilds slides are tailored to single-stage belt lifts, but they could easily be adapted to work for multi-stage or string lifts given some thought and additional parts.

Advantages

- Sturdy, doesn't flex even under a lot of load
- Interfaces with most other build systems thanks to sliding T nuts

Disadvantages

- Not designed for multi-stage lifts right out of the box
- **Heavy**, overkill for lifting lighter objects unless precision is important
- OpenBuilds only offers M3 and M5 T nuts, although other compatible sizes might be found on Amazon or McMaster-Carr

[73] https://openbuildspartstore.com/mini-v-gantry-kit/

Fig. 42: 4997 Masquerade, Cascade Effect, OpenBuilds slides

goBILDA goRAIL

Linear motion guides[74] are goBILDA's take on linear motion using goRAIL, which is a type of extrusion compatible with V-groove bearings. Similar to a standard 8020 V-groove carriage, used in FRC, goRAIL is a lighter option that is better suited for FTC.

Actobotics X-rail Slide Kit

> **Attention:** ServoCity has discontinued the Actobotics X-rail Slide Kit.

Actobotics' X-rail Slide Kit[75] works well out of the box. However, the main caveat is that the kit has a **very low maximum load (2lb. at maximum extension)**.

Teams will have to keep their designs on this kit particularly lightweight. This slide uses elastic retraction through the use of surgical tubing, which means that instead of having both an extend and return string, there is a retraction force applied at all times. This helps simplify tensioning and spooling, however, limits how fast the slide can be run. Additionally, the plastic end caps have a reputation of breaking regularly because they endure shock load every time the slide extends to maximum. 3D printed alternatives may be more sturdy than the stock end caps.

> **Attention:** It is highly recommended that teams add an additional set of v-groove bearings at the end of each piece of extrusion to give each stage an additional point of support. This will increase load capacity and possibly smoothness.

Advantages

- Easily interfaces with Actobotics building system
- Elastic retraction is a simple way to retract
- **Should be used for light/medium loads only**

Disadvantages

- Will flex under load, needs additional support
- Elastic retraction slows down extension speed and retraction will be slower than string retraction

[74] https://www.gobilda.com/gorail-based/
[75] https://www.servocity.com/cascading-x-rail-slide-kit

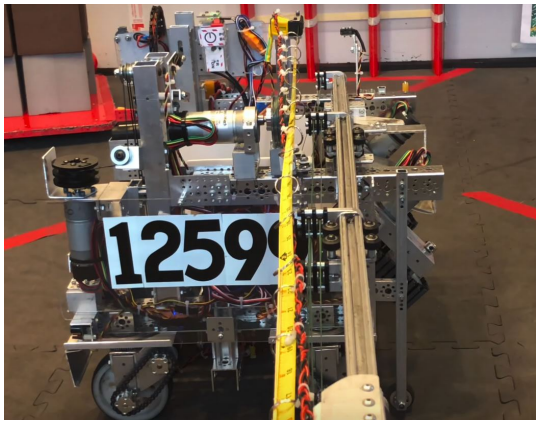


Fig. 43: 12599 Overcharged, Relic Recovery, Actobotics X-Rail slides

8.4.3 Linear Bearing Slides

Linear bearings are specialty ball bearings. Unlike radial ball bearings, they run balls through cyclical tracks along the length of a rail or shaft to create smooth, stable, precise linear motion. They are used in many industrial automation systems, including 3D printers and CNC machines.

> **Attention:** Unlike with most slides, the balls of linear bearings are not well retained when the bearings are removed from their rails. Linear bearings with a few balls missing have slightly decreased performance. Be sure to handle and store the bearings with care and to be prepared if their balls fall out.

There are two types of linear bearings generally used in FTC: round linear bearings and linear bearing blocks.

MGN Series Linear Bearing Blocks

MGN linear bearing blocks are the most common linear bearing blocks in FTC. They are the smallest of a larger classification of slides made by HIWIN Corporation. They come in several sizes, the smallest being MGN7 and the largest being MGN15.

It is **recommended that teams only use MGN7, MGN9, and MGN12 slides.** A pair of MGN12 slides is sufficient to lift a 40lb (20kg) robot. MGN9 slides are similar in weight and scale to MiSUMI drawer slides. MGN7 slides are smaller, but they are slightly more inconvenient to work with because they use smaller M2 screws for mounting.

If ordered directly through industrial supply centers such as HIWIN, these slides can be expensive. For hobbyists, however, they are sold at more reasonable prices through sellers on Amazon and eBay, although these sellers may have questionable quality control. You can find more information with a Google search of “improving cheap MGN slides”

Advantages

- Extremely smooth thanks to usage of ball bearings
- Stiffer and stronger than almost any other type of slide
- Steel rails don't wear out the way aluminum slides do

Disadvantages

- Because MGN rails and sliders are made entirely of steel, they are generally much heavier than aluminum drawer slides. This is not an issue with MGN9 slides, which are marginally heavier, and MGN7 slides, which are marginally lighter.
- Simplest way to connect slides together and mount pulleys is using custom parts
- Cheap slides generally need some attention to clean and re-grease

Fig. 44: 8813 The Winter Soldiers, Rover Ruckus, MGN12 slides connected with custom box tubes

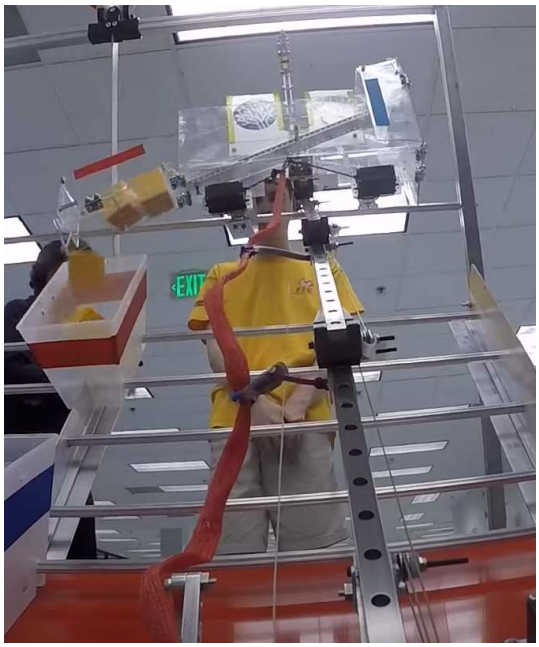


Fig. 45: 8221 Cubix, Res-Q, MGN slides connected with 3D printed parts

Round Linear Bearings

Round linear bearings are similar to bearing blocks and have similar properties, but they can be used on any round shaft rather than a specialized rail. They can also be used in applications where sliding mechanisms have to rotate on the same axis.

Round linear bearings can easily be sourced from most robotics suppliers, including goBILDA and Actobotics. However, there are currently no out-of-the-box lift kits using these bearings available from these suppliers. It is difficult to make a compact slide setup using only kit parts, and for this reason most teams using round linear bearings use custom parts to mount them.

Round linear bearings are often mounted with clamps, but they can also be mounted using inexpensive snap rings that snap onto the grooves on the outsides of the bearings.

Many teams use round linear bearings with inexpensive aluminum and carbon fiber tubes instead of steel tubes to save weight. Although slide assemblies made this way are extremely light, they are prone to wearing out over time as the steel balls wear into the softer materials. Teams should keep this in mind and try to design their slide systems so that the tubes can easily be replaced.

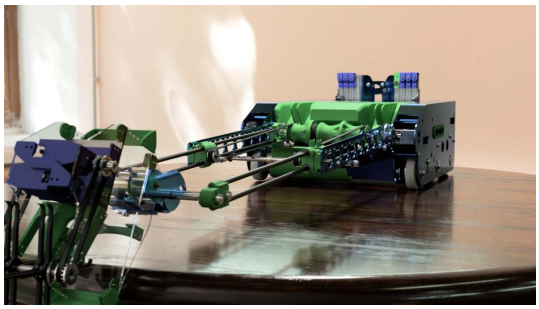


Fig. 46: 8417 'Lectric Legends, Rover Ruckus, round linear bearings and carbon fiber tubes with 3D printed mounts

Fig. 47: 5975 Cybots, Relic Recovery, round linear bearings and carbon fiber tubes with 3D printed mounts

Fig. 48: 11115 Gluten Free, Skystone, round linear bearings and metal shafts with 3D printed mounts

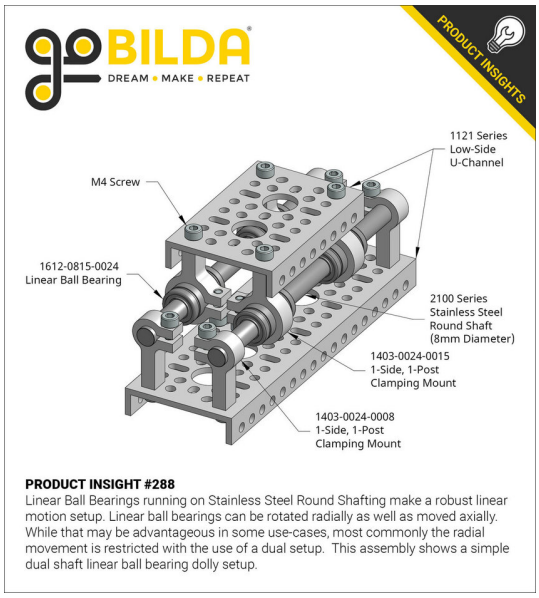


Fig. 49: goBILDA, round linear bearings with metal clamps and shafts

8.4.4 Custom Options

Some teams have opted to build custom solutions for linear motion. Many of these teams borrow concepts from FRC® teams, modeling their elevators off of the tall systems found on the larger robots. There's a reason that many competitive FRC teams build the same type of elevator - at their scale, the box tube elevator has proven to be the most efficient way to get game pieces off the ground at blisteringly fast speeds.

When built correctly, an elevator of this type can withstand hundreds of pounds of load on any axis while barely weighing anything. However, existing off-the-shelf options already fill the linear motion needs of most FTC® teams.

Custom extension systems also require tons of work in CAD, hours upon hours of manufacturing time, and may need multiple iterations before they work correctly. Due to their complexity and how challenging they are to design, less experienced teams may encounter significant challenges.

Note: Lightweight drawer slides (MiSUMI aluminum and Long Robotics Slides) can offer similar performance at a fraction of the complexity.

Fig. 50: 7236 Recharged Green, Rover Ruckus, custom box tube linear elevator

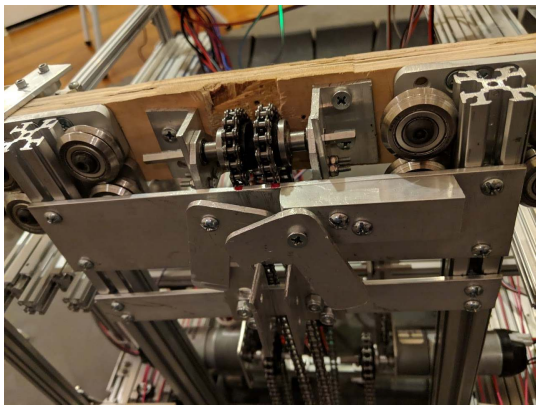


Fig. 51: 13075 Coram Deo Academy Robotics, Rover Ruckus, custom 8020 chain driven hang

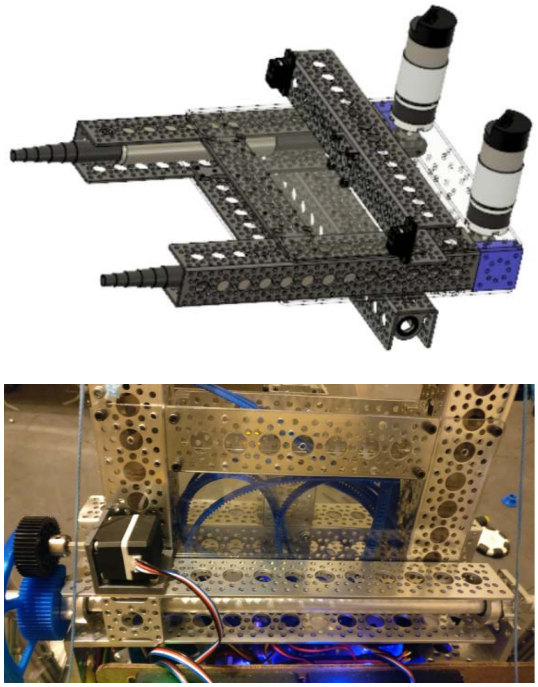


Fig. 52: 7172 Technical Difficulties, Rover Ruckus, fishing pole extension using gear rack

8.4.5 Lead Screws

Lead screws use a threaded rod to create high-torque linear motion. Their primary advantage is being able to handle much higher loads than other unmodified kit options. While this option was popularized in Rover Ruckus for hanging robots, variations have been in use since as far back as Res-Q. Another use case for lead screws is to change the angle of an arm platform, though this is more common in FRC®. However, this ability to handle high loads comes at a cost: lead screws are extremely slow.

By far the most popular lead screw option is the ServoCity Linear Actuator kit[76]. Overall, this kit is lightweight, simple, and cheap. It can be a good start for teams using REV and needing a linear extension, and is generally usable out of the box.

Advantages

- Easy way for high-load, high-torque applications
- Compact form factor
- Typically only requires one motor

[76] https://www.servocity.com/linear-actuator-kit-a-7-4-stroke-x-rail-piston/

Disadvantages

- Not for quick linear extensions
- High torque translates to slow extension speed

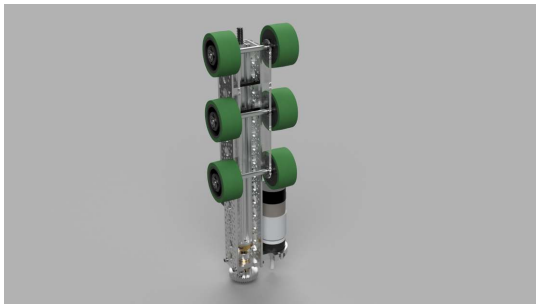


Fig. 53: 11115 Gluten Free, Rover Ruckus, Actobotics lead screw used for hang

8.4.6 Rack and Pinion

Rack and pinion refers to a a toothed linear gear (the rack), meshed with a a circular gear (the pinion gear). When the pinion gear is driven, it will drive the pinion gear upwards or downwards, depending on how the rack and pinion is mounted.

Generally, rack and pinion is a good light-use option for FTC® teams in terms of linear extension. However, there are some disadvantages to rack and pinion compared to the other options of linear extension. Therefore, rack and pinion is generally not recommended for teams.

Advantages

- Easy way to extend upwards.
- Power and linear motion in one package
- Can with proper support, sustain heavy load (e.g. hang robot)

Disadvantages

- Rack and pinion generally is only used in one stage, because multiple mechanisms require other forms of powering it (belt, string, chain, etc.)
- Rack and pinion needs to be supported very well to sustain heavy load, or else the mesh will fail. **It is not advisable to use the Tetrix rack and pinion in high stress conditions**.

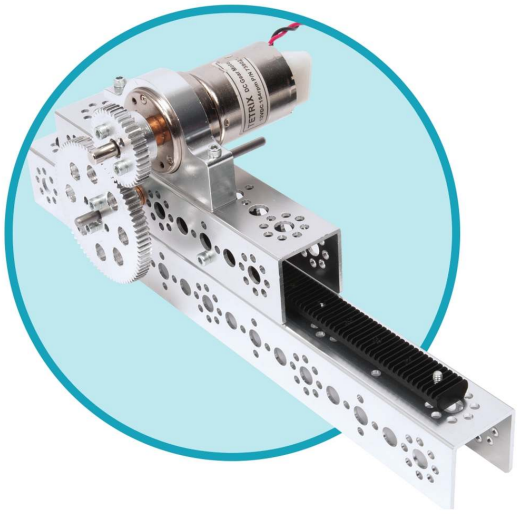


Fig. 54: The Tetrix rack and pinion has a tendency to fail under medium to high load.

8.4.7 Rigging

Rigging refers to the way that string, belt, or chain is set up to extend and retract a linear extension. This is an important and time-consuming requirement for any team that uses linear extension, so be sure to spend adequate time on it.

Fig. 55: Example rigging setup from team 7236 Recharged Green. This setup utilizes a continuous stringing setup for the first stage and a cascaded belt-driven second stage.

General Rigging Tips

- Rig your extension string when the extension is retracted, and your retract string when the extension is at maximum extension
- Clamping a string with a bolt instead of tying it around something makes it much easier to quickly change the tension of a string
- If tying an abrasive material string, like kevlar, around a plastic part, you can tie the string around a washer to distribute the loads across the hole in the pulley instead of tying it directly into a hole in the pulley
- Its generally easier to start from the side with the pulley and go to the end of the slides then the other way around

Continuous Rigging

Generally recommended

Continuous rigging entails rigging one long **extension string**, originating from a motor-powered spool, to the top of the base stage, then to the bottom of the first stage, then to the top of the first stage, then to the bottom of the second stage, etc. A **retraction string**, originating from a second spool on the same axis as the extension spool, is then anchored to the top stage. When the motor rotates one direction, the extension spool reels in the extension string, so it becomes shorter. In doing so, the distance between the top of one stage and the bottom of the next stage decreases, causing the system to extend.

Note: The last stage always extends and retracts before the other stages (this can be either an advantage or a disadvantage, depending on the application).

Once the last stage hits its limit, the next to last stage extends outwards, and so on; the pattern repeats until every stage is fully extended. When the motor spins **in the opposite direction**, the retraction string is reeled in, pulling the top stage closer to its starting position until the system is back where it started. For the retraction string, it is often necessary to add an extra pulley near the back of the extension. This is because the retraction will only retract to the farthest point, which generally is the spool. However, the spool may not be mounted at the very back of the robot - thus an extra pulley is needed. Note that for this to work, the extension string should be wrapped around the spool in the opposite direction of the retraction string. Thus, if the extension is wrapped clockwise, the retraction must be wrapped counterclockwise.

Here are some additional considerations when rigging a continuous system.

- As a general rule, continuous spools can be powered by a system with a relatively low gear ratio.
- The extension string and retraction string **do not need to be separate strings, but it is much easier to tension the system if they are separate**.
- The extension spool and retraction spool should be the same diameter.
- You need to make sure, as with any time you work with strings and pulleys, that you are pulling the string straight. Any amount of misalignment can lead to the string coming off of your pulley.
- The width and diameter of your spool should be enough such that once fully wounded, your string never overlaps. The reason for this is because when the string starts to overlap, this can change the diameter of your spool, causing the tension in the string to change.

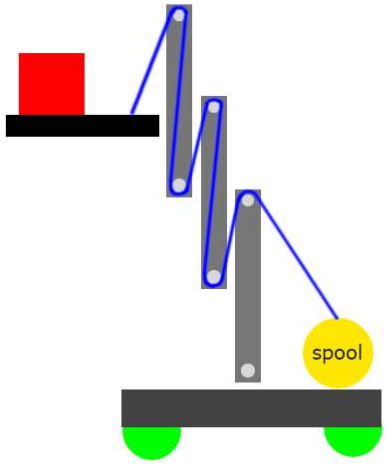


Fig. 56: Continuous rigging diagram

Cascade Rigging

Generally recommended

Cascade rigging is a bit more complicated than continuous rigging. Much like continuous rigging, an extension string originating from a spool is rigged to the top of the base, running down to the bottom of the first stage. However, instead of being rigged to the top of the stage, the extension string is anchored to the bottom of the first stage. A second extension string, anchored to the top of the base, is rigged to the top of the first stage and anchored at the bottom of the second stage. The pattern continues until all stages have been rigged.

Note: The number of strings required to extend is equal to the number of stages in the system.

When the motor rotates one direction, the extension spool reels in the first string, decreasing the distance between the base and the bottom of the first stage. This pushes the second string forward, decreasing the distance between the top of the first stage and the bottom of the second stage, and so on. Note that unlike continuous rigging, **every stage moves at the same time**. The second stage moves 2 times as fast as the first stage relative to the base, the third 3 times as fast, and so on.

A cascaded system can be retracted in three ways: using continuous retraction, elastic retraction, or reverse-cascade retraction.

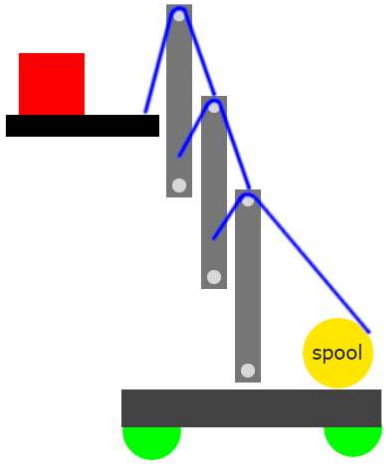


Fig. 57: Cascade rigging diagram

Retraction Options

Continuous Retraction

As the name suggests, continuous retraction utilizes inverted continuous rigging to retract the slides. There are two main methods of doing this, **free floating retraction** where a string is run from the last stage directly back to the spool, and **with slide retraction** where the string is run in an inverse continuous pattern. Generally, **with slide retraction** should be used with extensions going outside the robot frame to prevent entanglement, and **free floating retraction** should only be used with purely vertical slides.

Note: If continuous retraction is being used with cascade extension, the two spools cannot be the same diameter. If the variable N is the number of stages in the system, the diameter of the cascade extension spool must be N **times smaller** than the continuous retraction spool.

Warning: If free floating retraction is used, **make sure your retract string is always parallel to the slides**. If the string is misaligned, it can pull on your slide and cause excess bending forces on the stages.

Advantages:

- Simplest retraction rigging

Disadvantages:

- Can become tangled if free floating
- Requires different sized spools if used with cascade retraction

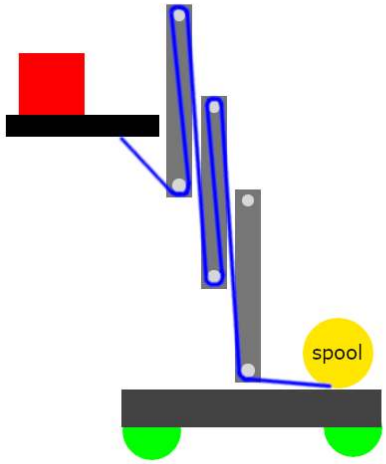


Fig. 58: Continuous Retraction

Elastic Retraction

Instead of retracting using a retraction spool, one common way to retract is to attach a piece of elastic (commonly surgical tubing) to the last stage. The elastic applies a force on the last stage that is counteracted by the motor when extending. However, when retracting, the motor reels the last slide back in. While this is the retraction method recommended by many kit slide manuals, this method is not recommended.

Advantages:

- There is only one string to tension, instead of multiple, so tensioning is simpler.
- The elastic automatically tensions the extension string.

Disadvantages:

- Since the elastic applies a force to the slide at all times, this force opposes the force applied by the motor when extending the slides. **Thus, elastic retraction decreases extension speed considerably**.
- The elastic does not apply a constant force at all times. It applies force proportional to the amount that the slide is extended, so retraction may not be smooth and controlled, like other rigging methods.
- It is very easy to unwind your extension spool when using elastic retraction.

Cascade Retraction

Cascade retraction entails simply rigging another set of cascade string that can retract the system when engaged (see the image below).

Note: If cascade retraction is being used with continuous extension, the two spools cannot be the same diameter. If the variable N is the number of stages in the system, the diameter of the continuous extension spool must be N **times bigger** than the cascade retraction spool.

Advantages:

- Very space-efficient
- All stages retract at the same speed at the same time

Disadvantages:

- Requires more string (less strings to tension, less strings that can become loose)
- Requires different sized spools if used with cascade extension

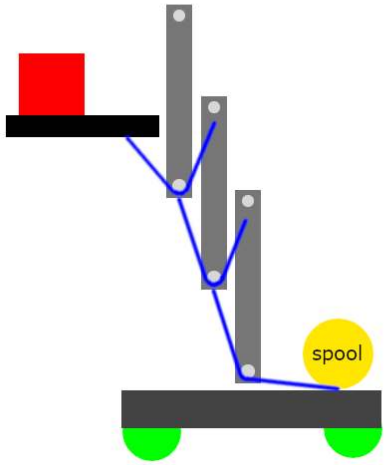


Fig. 59: Cascade retraction

Additional Considerations:

- If the system has only one stage, cascade rigging is **identical to continuous rigging**
- Unlike continuous rigging, each time a new stage is added to a cascaded system, the gear ratio required to maintain the same extension speed increases. For instance, if 2 stages are added to a 1 stage cascaded lift that is geared at a ratio of 3:1, the ratio must increase by a factor of $\frac{2+1}{1}$ to maintain the same speed, changing the ratio to 9:1.
- One disadvantage of cascade rigging is that each string must be kept tensioned. This is still the case with continuous rigging, but you have many more strings to keep track of, as tension must be maintained on all of them.
- You need to make sure, as with any time you work with strings and pulleys, that you are pulling the string straight. Any amount of misalignment can lead to the string coming off of your pulley.

Belt-driven slides

One increasingly popular alternative to traditional string-based rigging is belt-driven slides. This can be done continuously or using cascade rigging.

Advantages over string

Unlike string, belts used on slides never need to be tensioned. As discussed in the linear motion section, in order for string-driven slides to remain efficient, string tension must be maintained. Naturally, string loosens over time, so you either need a mechanism that can provide extra tension (a spring) or manually tighten string, which can get a bit tedious (especially for cascade rigging).

However, belts do not have this issue. They tend not to stretch over time, meaning complex external tensioners are rarely needed. Belts are also on an automatic one to one loop, meaning that for every inch you pull in on the extension side, you always feed back out that inch on the retraction side, and vise versa. While this might not seem like a big deal, if a string winds itself up on a spool and overlaps at any point, the diameter of the spool changes, making the two spools out of sync.

Disadvantages over string

The main disadvantage of belt-driven slides is the amount of space they take up. Simply put, belt pulleys take up much more room than the 4mm thick, 12mm diameter pulley bearings REV sells. When using belt-driven slides, pulleys are at least double that thickness and have a considerably larger diameter, meaning each stage must be thicker.

Fig. 60: 7236 Recharged Green, Rover Ruckus, **continuous** rig

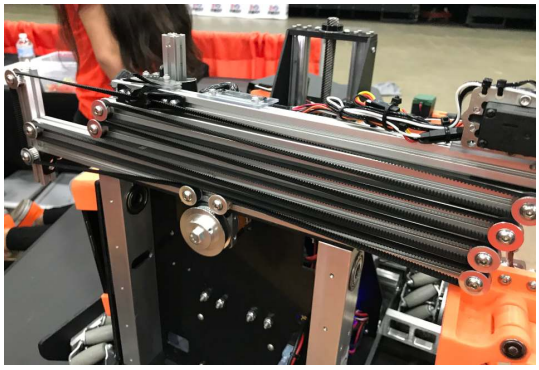


Fig. 61: 11190 Mechadojos, Relic Recovery, continuous belt rig

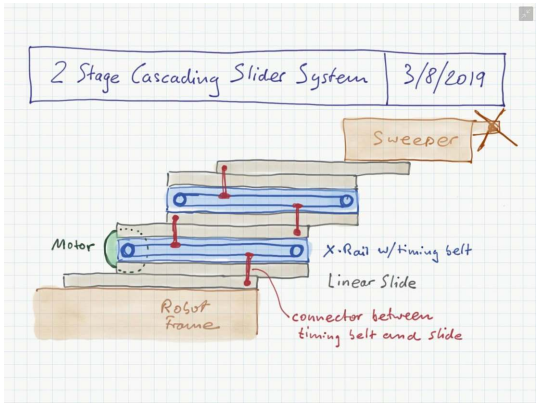


Fig. 62: 5064 Aperture Science Rover Ruckus **cascade** belt-driven proof of concept drawing

Fig. 63: 5064 Aperture Science Rover Ruckus **cascade** belt-driven final version

Belt or chain driven elevators

Almost ubiquitous in FRC®, belt and chain elevators have existed seemingly forever. The basic premise is to have sprockets or pulleys mounted at the top and bottom of the elevator to the robot superstructure. Then, the belt or chain is hard bolted to the elevator near the bottom sprocket or pulley. When the chain or belt is driven, the elevator will move up and down. It is possible for elevators to have multi stage designs, but powering them will be more complex.

Fig. 64: 7236 Recharged Green, Rover Ruckus

Tensioners

Tensioning string is one of the most painstaking tasks for a builder in FTC. Ensuring that both sets of slides are tensioned evenly can be an arduous and annoying job. However, adding tensioners to your strings can help solve the uneven tension and ensure that both sides of slides run together. The most common type of tensioner is a spring that can be purchased at a hardware store. It generally will be placed at the end of the string run, near the part that extends farthest out from the robot's center. By doing so, the string will stretch out when the spool extends the arm, keeping tension so that the string does not detach from the pulleys in the linear slide extension.

Another form of tensioner can be a spring-loaded pulley. Since the pulley is spring-loaded, it will take up the slack in the string. Alternatively, it is possible to mount a pulley on a piece of extrusion, and slide it so the string is taut.

There are two main reasons that tensioners are highly recommended in string based linear slides, the first and oftentimes more important one, is that at its core, a run of string is a series of polygons, and as the slide extends the effective size of that polygon changes. Meaning that when fully extended, your slide may require more or less string compared to when halfway extended. Or partially extended. The closer each set of bearings are, the less this impracts string tension.

Another (but less impactful) thing to keep in mind that spool size changes as string is added/removed from the spool. If the spool radius increases, the speed of the extension will also increase, and torque, which opposes speed, will decrease. Consequently, tension will change as well. Therefore, a bit of slack is inherent in all linear extension designs using string.

Attention: It is highly encouraged that teams have at least one spring tensioner per linear slide set.

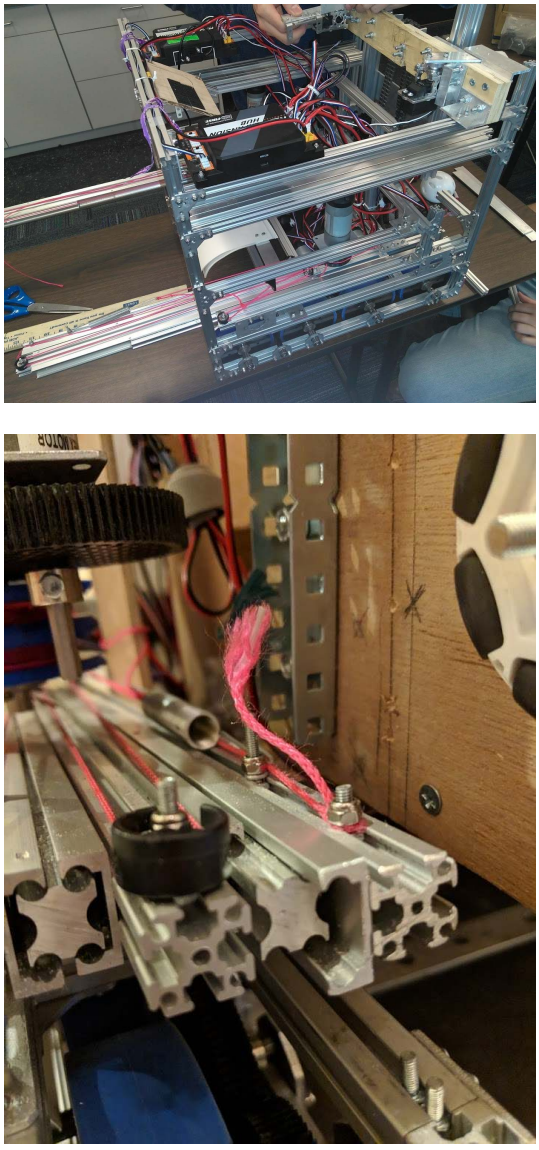


Fig. 65: 13075 Coram Deo Robotics, Rover Ruckus, string tensioner

Picking the right spool size

Spools have a special property that isn't often discussed, but is extremely useful when creating linear slide systems. Just as the system's speed and torque can be changed by changing its gear ratio, speed and torque can also be changed by changing the spool size. The motor rotates the spool at a constant angular speed. Thus, the translational speed (the speed of the slide) is proportional to the radius of the spool, and since torque is inversely proportional to speed, changing the spool size changes torque as well.

This is important to recognize, as changing spool size is often more convenient than changing gear ratio to get the desired combination of speed and torque. To illustrate this, say you have a linear extension system with a 3.7:1 gear ratio. You then decide that a 5:1 gear ratio would provide a more desirable combination of speed and torque than your current 3.7:1 ratio.

In many cases, instead of swapping gearboxes, it makes more sense to swap out spools to a smaller one. If your spool is currently 2 inches, your new size should be $\frac{2*3.7}{5}$ inches to achieve the same result.

You also need to make sure that when fully wrapped on the spool, your cable or string doesn't overlap. Overlapping can result in a change in spool diameter, which will change the tension in your string.

Cable management

When extending outwards, wire management becomes increasingly important. Obviously, it is a necessity to use wires slightly longer than the extension length. However, it is not recommended that these wires are left unprotected, as they can get tangled or caught in the slides much more easily than with protection.

In general, teams should ensure that wires never protrude outside the structural parts of the robot, because they can get caught on other robots or game pieces. This can be accomplished by cable ties or Velcro ties, or by using acrylic plate to keep wires inside.

However, for linear extensions, other forms of cable management are needed. The two types of cable management recommended are cable carrier and retractable coil cord. Refer to the *Electronics and Wiring* (page 211) section for more information.

Cable Carrier/Drag Chain

Cable carrier, the standard wire management method within industry, is plastic chain links with a hollow center. Cables are placed inside the chain, allowing the system to extend indefinitely. The links are somewhat stiff yet flexible, allowing cable chain to bend when the extension is retracted and straighten when extended. They typically are stiff enough not to sag excessively when retracted.

Here are some links to various drag chain products:

- igus cable carrier[77]
- uxcell 10x10mm drag chain, from Amazon[78]

Advantages:

- Difficult to get tangled/hooked onto other objects or robots
- Sturdy and durable
- Protects wires very well

Disadvantages:

[77] https://www.igus.com/info/energy-chains-e2-micro-small-cable-carrier

[78] https://www.amazon.com/uxcell-InnerH-InnerW-Plastic-Carrier/dp/B01LX02PSW/ref=sr_1_1?keywords=drag%2Bchain&qid=1566188144&s=gateway&sr=8-1&th=1

- Large form factor, takes up a lot of space
- Links need to be added if additional extension is needed
- Can be on the heavy side, especially with a long length of drag chain

Fig. 66: 7236 Recharged Green, Rover Ruckus: Cable carrier on the left side of their horizontal slides and the right side of vertical slides

Retractable Coil Cord

While not common within industry, coil cords are still very common (coil cord is a nearly ubiquitous staple of older telephones). Retractable coil cord is more flexible than cable carriers, stretching when extended.

Advantages:

- Very space-efficient
- Flexible and can usually extend to any length needed (unlike cable carriers, no new links ever need to be added)

Disadvantages:

- Can get tangled more easily, as it is less stiff than drag chain

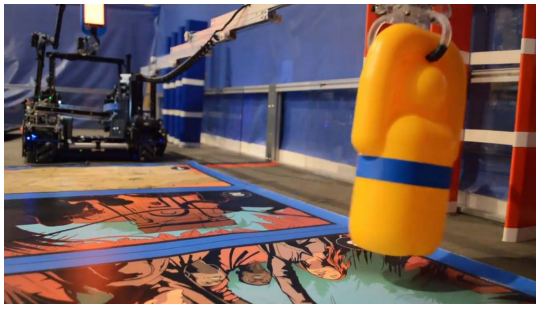


Fig. 67: 10030 7 Sigma Robotics, Relic Recovery: Coil-cord on horizontal extension mechanism

Self-retracting Badge Holder

While an uncommon technique, a self-retracting ID Badge holder can help manage wiring on an extension. These are common items; here is one from Amazon[79].

This really only works for vertical or near-vertical slides, as there is still some sag in the cable which can easily get entangled if it extends outside of the robot frame.

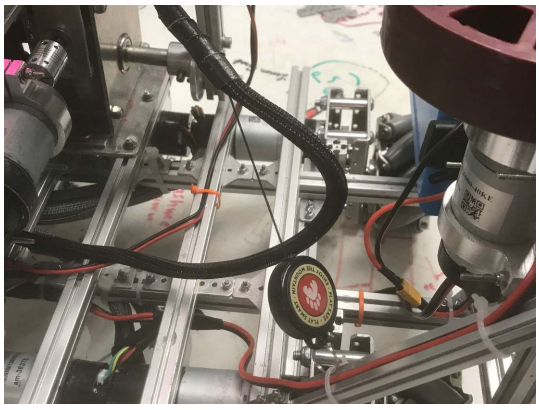
To rig a self-retracting badge holder to a wire, mount the hard plastic part of the badge holder to the base of the slides, and mount end of the holder's string to the wire. The wires should be hard-mounted (using zip ties or whatever your team uses) towards the top of the retracted slide with plenty of slack. For best results, the badge holder should be almost completely retracted (leave a centimeter or two to allow for some slack) when the slide is retracted.

Advantages:

- Extremely simple and easy to implement
- Flexible and can usually extend to any length needed (unlike cable carriers, no new links ever need to be added)

Disadvantages:

- Similar to a coil, self-retracting cable management strings can get tangled more easily, as it is less stiff than drag chain
- Does not work well on long slides (over 2 stages) without substantial engineering work



[79] https://www.amazon.com/OFES-Retractable-Holder-Swivel-Alligator/dp/B07WPRJY9K/

Fig. 68: 248 Fatal Error, Freight Frenzy: Badge retractor cable management on extension mechanism

8.5 Arms

Arms are another way to achieve extension past the 18” x 18” dimension of the robot. Unlike linear extensions, arms require lots of torque - a standard 40:1 or even 60:1 gearbox will not be suitable in most applications. For example, many teams will run a 256:1 gearbox for their rotation motor.

Attention: Such motors must be very well supported, or else the motor may torque itself free from its mount.

Rarely should an arm be directly mounted to the driving motor. Instead, torque should be transferred via gear, chain, or belt. Large arms can also be hard to control (with the addition of momentum adding load to the gearbox, it is hard to stop a three-foot arm that weighs five pounds quickly without breaking a gearbox). In many cases, this issue can be mitigated with software (see the *Control Loops* (page 323) section).

Depending on the application and implementation, arms can be either faster or slower than extension options.

The different types of arms in FTC® include single arm and multi-axis arms.

8.5.1 Advantages

- Single bar arms can be relatively simple to build.
- Arms can be useful in low-load applications; however, most mechanisms in FTC are not very light.

8.5.2 Disadvantages

- Arms require a large amount of torque, and in order to do so, teams must purchase high-torque gearboxes, such as UltraPlanetary gearbox from REV[80] or the high gear ratio goBILDA planetary gearbox motors[81].
- While single arms may be more simple, they cannot provide enough extension for most games.

[80] https://www.revrobotics.com/rev-41-1600/
[81] https://www.gobilda.com/yellow-jacket-planetary-gear-motors

8.5.3 Single Arms

The most simple type of arm in FTC, a single arm refers to an arm on one axis of rotation. While it is possible to successfully build this kind of arm, generally a single axis arm will only afford around 15-16” of extension, which is inadequate for nearly every game.

The reason for this is that the longest a channel can be is 18” (technically you could have a longer channel by placing it diagonally, but this complicates matters). Thus, with a maximum of 18” of extension, a couple inches must be subtracted, since the point of rotation is inside the 18” sizing cube; therefore the extension is around 15-16”.

Therefore, a single arm with further linear extension is optimal. For example, some teams built a single arm with an added linear extension mechanism to reach the desired extension length, which is generally >24”.

The advantages of a single arm are that it is relatively easy to build, and can be a quick way to gain some form of extension outside of the robot cube.

However, there are many disadvantages such as having a high gear ratio, requiring much more support than a linear slide, and being hard to control without the proper software.

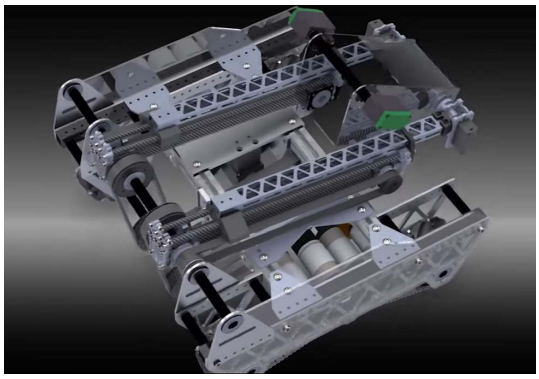


Fig. 69: 8103 Null Robotics, Rover Ruckus, single arm + custom belt driven linear extension

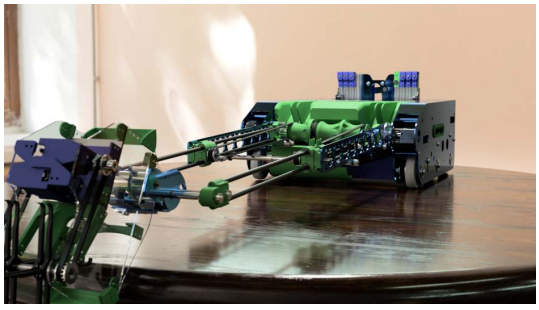


Fig. 70: 8417 ‘Lectric Legends, Rover Ruckus, Finalist Alliance First Pick (Ochoa), single arm + custom belt driven linear extension

8.5.4 Multi-Axis Arms

A multi-axis arm is an arm which has multiple points of rotation. Multi-axis arms introduce many variables that exponentially complicate matters and can really only be modeled through complex kinematic equations.

Warning: This is highly discouraged for inexperienced FTC teams due to the difficulty of building as well as the need for machine tools.

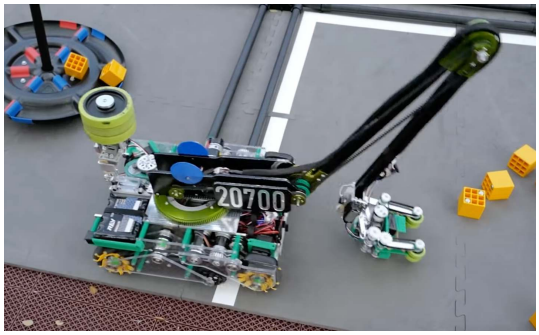


Fig. 71: 20700 Snap, Freight Frenzy

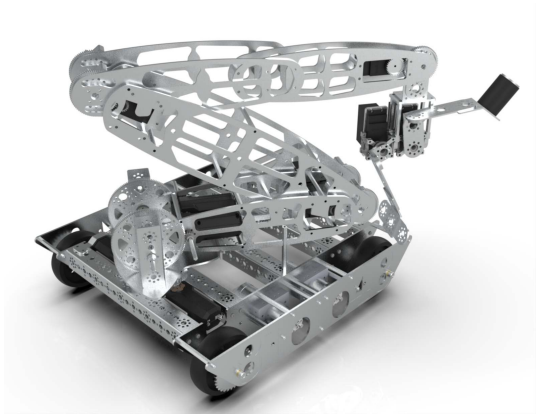


Fig. 72: 8148 Aleph Bots, Relic Recovery

Fig. 73: 15692 Rust In Pieces, Rover Ruckus

8.6 Linkages

There are many different types of linkages. Often, linkages are used to convert rotational motion, such as that from a servo or motor, to linear motion. Linkages can do this efficiently, and also have specialized movement patterns that can make them desirable for certain mechanisms.

Term	
Linkage	A system of solid links or bars connected to two or more other links by hinges, sliding joints, ball-and-socket joints, etc., so as to form a closed chain or a series of closed chains. Generally used to convert linear motion to rotational motion or vice versa.

8.6.1 Considerations

There are several things to consider when constructing a linkage.

- Over-centering is when a linkage is rotated past its center point (usually the point where both bars of the linkage are parallel). Driving a linkage over center can have some benefits, like making the linkage harder to backdrive. See this informational youtube video[82] for more details on this. Essentially, linkages can form a structure where pushing on the linkage moves the robot instead of rotating the linkage.
- Linkages will not have a constant linear speed or force. They generally reach a maximum speed and force when the bars are perpendicular, and the speed and force will decrease as the linkage is rotated farther.
- Cadding the linkage can be useful to check if it will do what you want it to do. See this youtube video[83] for an example of how to set up the mates in OnShape.

8.6.2 Common Linkage Types

Linkage Slides

A common use for linkages is to drive a linear extension. This allows for a compact method of converting the rotational motion from a motor or servo into linear motion moving a mechanism like drawer slides. Generally, a two bar linkage with multi degree ball linkage components are used to construct these linkages.

[82] https://www.youtube.com/watch?v=I7iy8DCNmic
[83] https://www.youtube.com/watch?v=QsAC_seQHJY

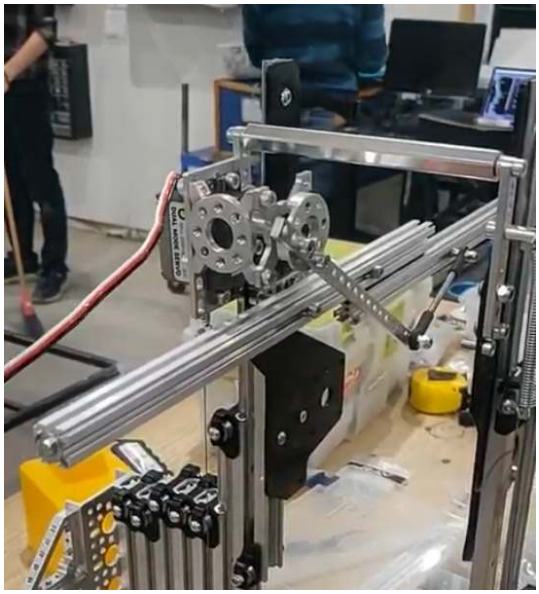


Fig. 74: 7236 Recharged Green, Skystone

Four Bar

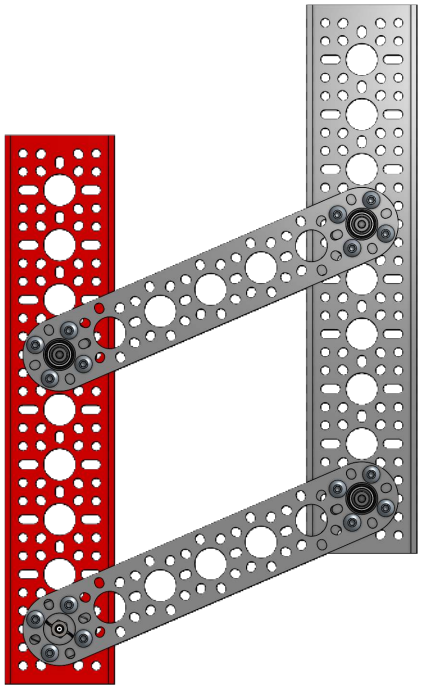
One type of linkage is called a four bar linkage. This is a linkage that keeps the end bar at the same angle to the ground at all times. For example, if the end bar of a virtual four bar is parallel to the ground when retracted, it will be parallel to the ground at all times, even when rotated fully out. This is beneficial for mechanisms like claws, which you want parallel to the ground during operation. In addition, four bars can provide extension outside of the robot frame, and generally provide both vertical and horizontal extension due to the "arc" that the arm follows. Also, the construction material of the four bar can be carefully selected to save weight.

This mechanism is not widely used in FTC® due to the space requirements. The linkage bars that keep the end bar parallel restrict a four bar to under 180 degrees of travel (less then 90 degrees in either direction) without specialized mechanical construction, and the bars also take up space in the robot frame.

CAD Example of Four Bar (Click to expand)

Click here to open this example in Onshape Cad, where you can click and drag parts to see how they move![84].

[84] https://cad.onshape.com/documents/45549489f570f3694569a2df/w/85ff26b9fca4988ebc4df3b4/e/204f2654fb268fb556c1b7b1



Virtual Four Bar

The more popular alternative to the Four Bar linkage is a Virtual Four Bar. While not technically a linkage, a virtual four bar uses chains or belts to create an effect similar to a four bar, where the end effector is kept at a fixed angle to the ground at all times. Because the need for additional bars are eliminated, this "linkage" can travel more then 180 degrees, and also can take up less space then a traditional four bar linkage.

This mechanism is more widely used in FTC, as it is an easy addition to an arm to maintain the end effector's angle relative to the ground.

Note: Both chain and belt can be used to construct a virtual four bar, and there isn't a specific benefit to using either. Due to the fact that the chain doesn't have to completely rotate around the sprocket, a zip-tie can be used to tension the chain, making construction easier.

CAD Example of Virtual Four Bar (Click to expand)

Click here to open this example in Onshape Cad, where you can click and drag parts to see how they move![85].

[85] https://cad.onshape.com/documents/45549489f570f3694569a2df/w/85ff26b9fca4988ebc4df3b4/e/62097ae7e6d154b9232d8957

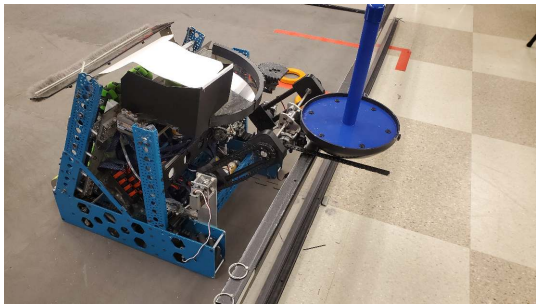


Fig. 75: 7244 OUT of the BOX Robotics, Ultimate Goal

Double-Reverse Four Bar

The double reverse four bar is an extension of the four bar linkage that allows for more extension. A double reverse four bar consists of a four bar linkage with a second four bar linkage mounted to the end of the first. This allows for purely linear extension, as opposed to the "arc" that a single four bar will take. The end effector will still stay at a fixed angle to the ground due to the purely linear extension.

This mechanism is not widely used in FTC due to the generally large space requirements needed, but is a fairly compact method of producing large amounts of linear extension. Care needs to be taken that each side of a double reverse four bar is driven equally so that issues do not arise from asymmetric lifting.

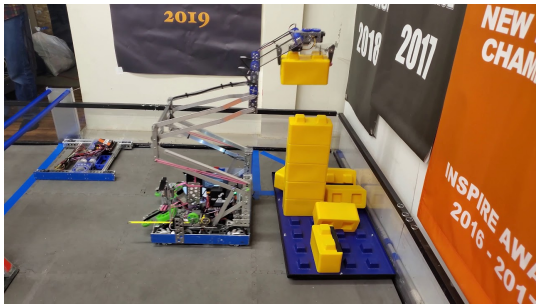


Fig. 76: 11115 Gluten Free, Skystone

Virtual Double-Reverse Four Bar

Similar to the virtual four bar, the virtual double-reverse four bar is a linkage that uses belts or chains to replace the linkage bars of the double-reverse four bar. The virtual double-reverse four bar is simply a virtual four bar where the end effector sprocket/pulley is half the size of the static sprocket/pulley, leading to a 2:1 driving ratio. Then, a bar is mounted to the end effector sprocket/pulley, and another chain/belt is run, which creates purely linear motion with an end effector that is always at the same angle to the ground. This creates much more compact linear motion then a double-reverse four bar, and can also extend in both directions.

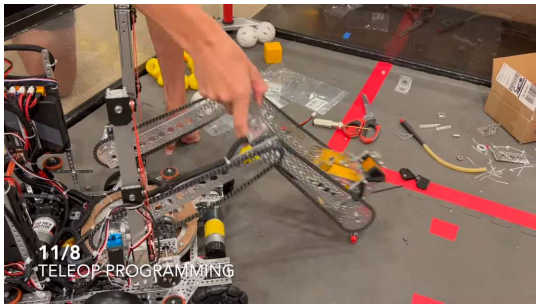


Fig. 77: 8644 Brainstormers, Freight Frenzy

8.7 Passive Intake/Claw

Passive intakes are mechanisms in which the driver must pick up individual items, usually by actuating some sort of gripper, which holds the game element.

Claws are the most common sort of passive intake and probably one of the first mechanisms that jump to mind during brainstorming. Unlike most intakes, claws can be made with only servos, and thus don't require valuable motor slots.

This is not to say there are no disadvantages; notably, unlike intakes, when you pick up something with a claw, it isn't easy to transfer the collected element to a separate scoring mechanism. For this reason, in most cases, claws double as both a collection method and a scoring method.

Claws are extremely useful for picking up large, unwieldy elements that cannot be controlled by active intakes. For example, the relic in Relic Recovery would be much harder to control using a more active intake; so teams used a claw.

Attention: However, claws are outclassed by active intakes for picking up small game pieces in nearly every game.

8.7.1 Advantages

- Much simpler than an intake
- Takes a lot less time to build than a reliable intake + transfer
- Less moving parts = less things that can go wrong
- Usually does not take up a motor slot
- Can pick up odd shaped game elements that active intakes can't

8.7.2 Disadvantages

- Depending on their use case, claws are slower than intakes.
  - The driver must angle the robot, position the claw, and grab the game element.
  - By contrast, with an intake, the driver needs only to position the intake and grab the game elements.
- Claws can be more fragile and break easily.
- In cases in which claws are slower than intakes, they can be much more prone to defensive maneuvers. The more time spent collecting, the more vulnerable you are.

Fig. 78: 11115 Gluten Free, Finalist Alliance Captain (Detroit), Relic Recovery

Fig. 79: 8680 Kraken-Pinion, Relic Recovery

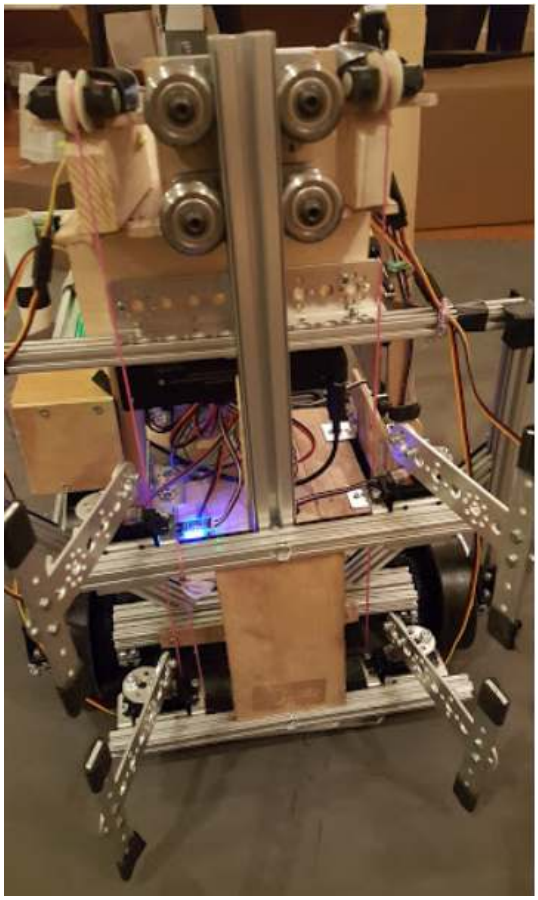


Fig. 80: 13075 Coram Deo Academy Robotics, Relic Recovery (**suboptimal use case**)

8.8 Active Intake

There's one main rule when it comes to intakes in FTC: you need one. Before diving in, what *is* an intake in the first place?

Intake A mechanism designed to pick up game elements using some component of rotational motion powered by a motor.

An intake is differentiated from a claw as an intake does not grasp individual game elements using a pinching motion, but rather sucks the element in through a variety of means such as wheels, tubes, etc.

There are multiple parts of an effective intake: intake geometry, intake type and material, and speed. However, a common rule of them is the faster the intake, often the more effective it is. For example, near the end of Rover Ruckus, many of the top performing teams used a 3.7:1 (1600 RPM) motor or a 5.2:1 (1000 RPM) motor on their intake.

Tip: While not a necessity, it is highly recommended that teams dedicate at least one motor to their intake if possible.

Attention: When it comes to intakes: "**Touch it, own it**" is a really valuable concept.

When your intake comes in contact with game elements, you want it to instantly control them. You should design with a margin for error, both in the orientation of the game piece and in the alignment of the robot. For example, look at FRC® Team 694's ball intake on their 2019 robot:

Fig. 81: FRC 694 StuyPulse, Deep Space

When this robot drives up to a ball, it barely has to point itself at the ball before the ball is instantly locked in between the two rows of wheels. What makes it so effective is the immense amount of testing that went into its design. The team tried a number of intake shapes, initially building with wood and rubber bands, and recorded which shapes were most effective. By testing their intake design out before competition, they didn't have to guess whether it would work as intended: they put it on the field with confidence.

8.8.1 Principles of an Intake

Reliability

- An intake must be reliable, as picking up and scoring game elements is the primary method to gain points in all FTC® games.
- Intakes have many moving parts and are susceptible to breaking, especially at high RPM. Thus, the intake material must be durable to withstand long periods of operation.
- Intakes often stick outside of the robot frame perimeter. In this case, durability becomes extremely important; the intake must be built so that they can withstand impacts/collisions with other robots or parts of the field.

  There are two ways to accomplish this - either by building the intake very robustly (lots of support so it doesn't break), or making the intake flexible (for example out of lexan or spring loading it) so that even though it may bend due to impacts, it will always spring back into place.

  Another way to prevent intake breaking is to make a fully retractable intake that won't protrude outside the 18" cube, though this is seldom needed.

Consistency

- The intake **must** be able to pick up game elements consistently and quickly. For example, in Rover Ruckus each robot could only possess 2 minerals at a time. Therefore, the intake should only pick up two pieces at a time, not zero, one, or three.

  Attention: This is a common mistake that many inexperienced teams fail to take into account.

- Another component is the varying angles that the game objects can be located in. This was especially apparent in the Relic Recovery season, where the "glyphs" (foam cubes) could be oriented in many directions.

  Even though it was relatively easy to make a compliant wheel intake that could effectively intake glyphs in one direction, it was hard to make an intake that could deal with angled glyphs. Being able to intake glyphs in all orientations was especially important for multi-glyph autonomous modes.

Controllability

- The intake must be able to consistently control the game elements. For example, if the intake is too fast and the collection box is not well designed, then game pieces might fly out. If the intake is too slow, it may jam itself when contacting the game elements.
- It is possible for pieces to get jammed at an unreachable angle, especially when using wheeled intakes. If this occurs, ensure that the driver can jar the stuck element loose to avoid having a disabled robot.
- Optimally, the game elements should follow a certain path each time if funneling is done correctly.
- It is best practice that the driver can see the game elements which are being controlled. This can be done through using clear plating such as Lexan.

Efficiency

- The key to any successful robot is cycle time. Reducing cycle time by having an efficient intake will lead to major improvements in score. A good intake should take no more than a few seconds to successfully collect the needed elements.

  For example, in Relic Recovery, the best intakes often had a <3 second collection time for two game elements, and in Rover Ruckus a 1 second collection time was desired at the highest level.

- A key rule to remember in FTC is the shortest distance rule: how can you get scoring elements from A to B in the shortest distance?

  The answer is usually one or two straight lines. The closer the scoring elements follow this path, the faster they will go from collection to deposit. Don't make overly long routes unless needed.

8.8.2 Types of Intake

Horizontal/Top Intake A horizontal intake generally is a wider style of intake that has the intake rotating on a horizontal axis or plane. Horizontal intakes are generally used for intaking smaller game elements, as these types of intakes can control more than one piece at a time. Horizontal intakes have been used successfully in games such as Res-Q and Rover Ruckus, where teams needed to pick up small cubes and balls from the floor.

Vertical Intake A vertical intake typically has wheels or other intake components rotating on a vertical axis (the z-axis). Vertical intakes are more controllable, as they can only pick up one element at a time. Vertical intakes excel at picking up large game elements which would be impossible to control more than one piece at once, such as the foam glyphs in Relic Recovery.

Below are different implementations of horizontal and vertical intakes:

Roller and Wheeled Intakes

Roller and wheeled intakes refer to types of intakes which work by having some sort of hard or pliable object rotate along an axis. Wheeled intakes use different kinds of wheels (solid traction, compliant, or foam wheels) that propel the game element to the collection bin. Roller intakes are wider, and can sometimes intake more than one game element.

Term

Compliant Wheel A compliant wheel is a flexible rubber wheel that is primarily used for intakes. These are **NOT** designed for use in drivetrains.

The available bore options vary depending on the vendor. As with the compliant wheels, durometer (hardness of rubber) affects both traction and longevity, sacrificing one for the other. However, in the case of intakes, a lower durometer is recommended to have maximum grippiness for intaking game elements.

Durometer refers to the hardness of rubber. Having a high durometer translates to a harder rubber surface, more durability, but less traction. A low durometer means a softer rubber, worse durability, but improved traction.

Examples of compliant wheel are Andymark's Compliant Wheels, and goBILDA's Gecko Wheels.

Fig. 82: A 2” compliant wheel

Fig. 83: A 4” compliant wheel

Solid Wheel Intake

Solid wheel intakes use wheels such as grip and traction wheels to pick up game elements. They can be effective in picking up large game elements which do not have much tolerance difference.

However, compliant wheel intakes generally will be more effective because compliant wheels offer more adjustability and forgiveness for the game elements. Compliant wheels also accounting for different game element orientations (the game pieces can get rotated when being controlled) and intake angles (the orientation of the game piece before it reaches the intake).

Compliant Wheel Intake

Spring-Loaded Intakes

A spring-loaded intake is able to pivot in order to accommodate a game element when it is passing through the intake, but will snap back when the element has gone through. A spring-loaded intake requires more thought, but guarantees that the intake wheel will always be in contact with the desired element.

A locked intake simply means that the wheels are locked into place and cannot pivot.

The compliant wheel intake is most commonly used with large game elements such as the glyphs in the 2017-2018 season, Relic Recovery. Compliant wheel intakes excel at controlling elements with flat surfaces such as cubes or rectangular prisms. They suffer at picking up balls.

In this game, robots had to pick up glyphs, which were 6 inch foam cubes, from the center pit and place them in the cryptobox. This game had many wheeled intakes primarily because the wheels had consistent and controllable contact with the glyphs. Wheeled intakes were able to propel the glyphs in a consistent fashion from the point of contact to the deposit plate, which would flip up to deposit the glyphs.

Wheeled intakes can be spring-loaded or locked into place. Teams could choose one or mix; in Relic Recovery, some teams spring-loaded the set of wheels that made contact first, and then had a fixed set in the back. This is up to the design team's choosing.

Wheeled intakes operate at much slower RPM than surgical tubing intakes, as wheeled intakes are meant to pick up **one** element at a time. They generally require more torque than a surgical tubing intake, which is geared for speed.

Term

Surgical Tubing Surgical tubing is generally latex or rubber tubing. Its most common use case is in active intakes, and has been popular among teams for many seasons. Surgical tubing has a hollow center and is sold in different diameters and wall thicknesses. Teams can experiment with different kinds of surgical tubing, as well as adding polyurethane tubing (clear tubing that is stiffer than rubber or latex tubing) in order to make the tubing more stiff.

Advantages

- Very controllable
- Propels elements to desired location
- Great at picking up large elements

Disadvantages

- Picks up only one element at a time
- Elements can get jammed in a bad position
- Not generally used for picking up small elements
- Can generally only pick up one specific element shape

Fig. 84: 9971 LanBros, Finalist Alliance First Pick (Detroit), Relic Recovery, springloaded

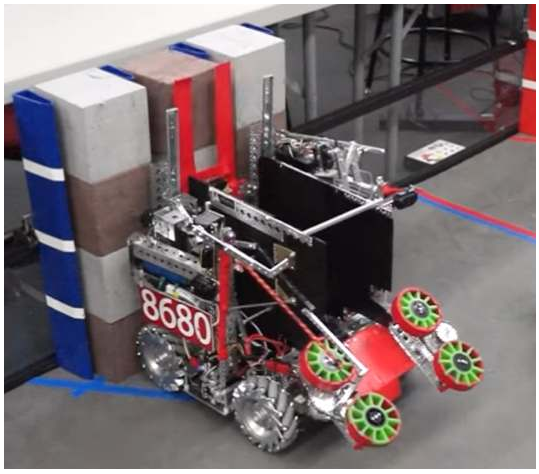


Fig. 85: 8680 Kraken-Pinion, Relic Recovery

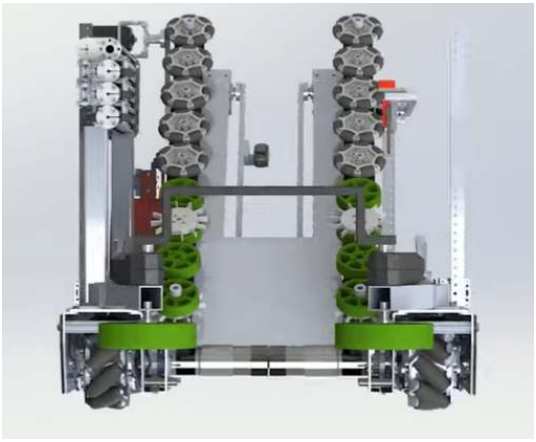


Fig. 86: 2856 Tesseract, Relic Recovery, 2 & 4 in. compliant wheels, 2 in. omni wheels

Fig. 87: 11115 Gluten Free, Finalist Alliance Captain (Detroit), Relic Recovery, springloaded

Foam Wheel Intake

A foam wheel intake has the same principles as a compliant wheel intake, except that it uses foam wheels. It is generally recommended that teams stick to compliant wheels as they are grippier and easier to control.

Fig. 88: 6299 ViperBots QuadX, Res-Q

Rubber Band Intake

Rubber band intakes, commonly used by in VRC, generally feature sprockets, wheels, or gears at two ends, with rubber bands interlaced in between to form a pliable and bendable roller. Generally, it can be actuated or adjustable with a servo, although this is not necessary.

Rubber band intakes are great with intaking balls, but not so great with other types of game elements such as cubes. It generally is slower than a surgical tubing intake, and requires multiple stages to transfer elements from collection to deposit. Zip ties can be added to increase the intake's range to accommodate for smaller balls.

Fig. 89: Ball intake for VRC Game Turning Point

Tubing Intakes

Tubing or noodle intakes typically use some sort of pliable tubing, which is rotated at high RPM to intake game pieces. Tubing intakes are particularly efficient at picking up small objects such as the balls and cubes from Res-Q, Velocity Vortex, and Rover Ruckus.

Surgical Tubing Intake

Surgical tubing or spearfishing rubber tubing, sold by many different manufacturers, is a great option for picking up small game elements such as the minerals from Res-Q or Rover Ruckus.

Surgical tubing intakes can, and often, have multiple sets of tubing in order to move minerals from the collection point to the holding box. This was most often seen in games where robots had to transfer minerals from the field to an elevated location.

Unlike wheeled and rubber band intakes, which can be spring-loaded, surgical tubing intakes are practically always fixed at a certain height and angle.

Surgical tubing by itself is soft and pliable. Teams have two options

1. Increase the RPM to 800-1000+ RPM
2. Use polyurethane tubing at a lower RPM (100-250 RPM).

   Polyurethane tubing can be purchased at a local hardware store and is a clear tubing that is quite stiff. Using some lubricant, insert the clear tubing into the surgical tubing for added stiffness.

It is encouraged that teams test different RPMs and stiffness to develop the optimal intake. You will be surprised how changing one small variable such as the diameter of tubing or how far the tubing extends can affect intake effectiveness.

Surgical tubing intakes are especially good at picking up multiple elements at a time, due to the high RPM (sometimes >1000 RPM) of the rollers. However, it suffers from a lack of controllability, as sometimes the driver may accidentally pick up more than needed, and have to spit it out.

Advantages

- Able to collect multiple elements at a time
- Generally more efficient than wheeled intakes
- Specializes in small and odd-shaped elements

Disadvantages

- Requires high RPM
- Less controllable
- Harder to pick up large elements easily

Zip Tie Intake

Instead of using surgical tubing, some teams opt for heavy zip ties instead. This can work, but we recommend surgical tubing as it is one of the most tried and tested methods for picking up nearly any game element. Zip ties lack the friction of rubber tubing.

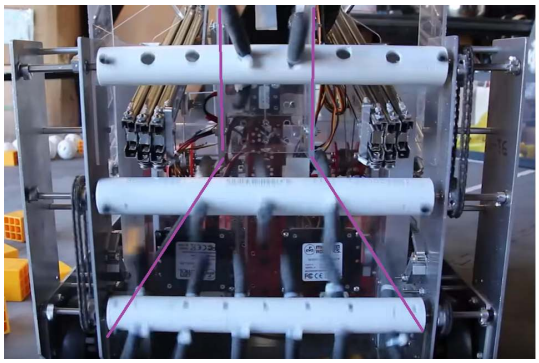


Fig. 90: 8375 Vulcan Robotics, Res-Q - great example of funneling

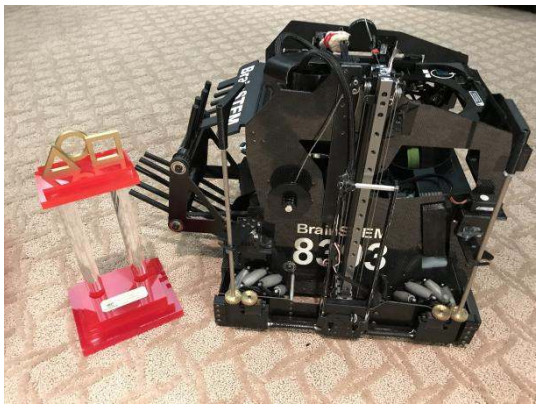


Fig. 91: 8393 Giant Diencephalic BrainSTEM Robotics Team, Semifinalist Alliance Captain (St. Louis), Velocity Vortex

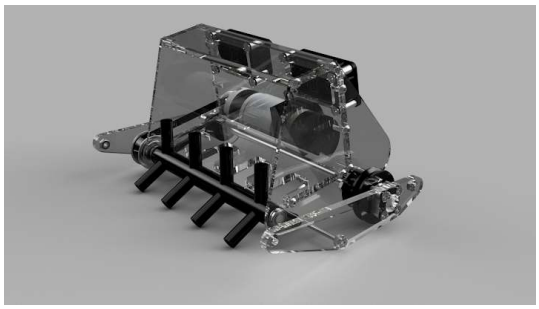


Fig. 92: 11115 Gluten Free, Winning Alliance First Pick (Detroit), Rover Ruckus

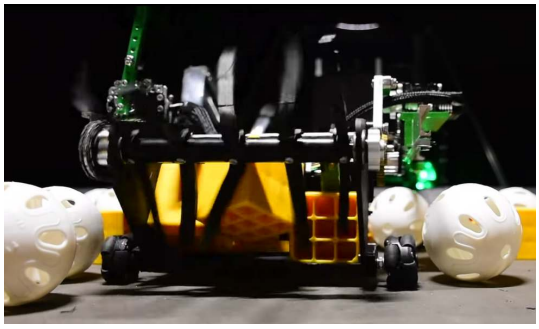


Fig. 93: 7203 KNO3, Rover Ruckus

3D printed intake (NinjaFlex/TPU Filament)

TPU/NinjaFlex 3D printer filament is a great low-RPM intake flap option, if designed right they work well with servos, and several teams have successfully used this configuration in competition.

We recommend 3D printed intakes only if your team has had experience in 3D printing parts.

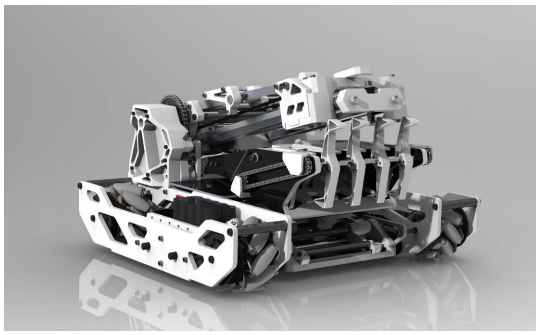


Fig. 94: 731 Wannabee Strange, Rover Ruckus

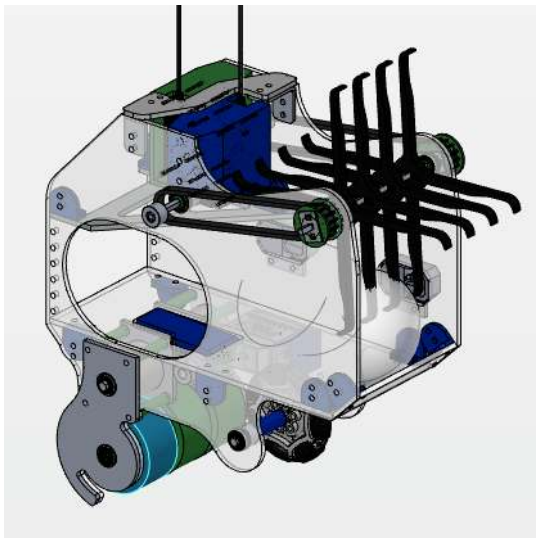


Fig. 95: 8417 Lectric Legends, Rover Ruckus - TPU intake flaps

8.9 Transfers

It is useful to consider how items will be moved inside the robot from one mechanism to another. This is typically done with a “transfer”, which moves something from one mechanism (such as an intake) to another (such as an arm on a deposit).

8.9.1 Transfer Design Fundamentals

There are a few fundamental aspects of the design of any transfer that must be considered. Generally, the most important considerations are:

1. What will be transferred?
2. Where will the item(s) be transferred?
3. How will the item(s) be transferred?

Object Considerations

One important thing to consider when designing a transfer is what object will be transferred by the mechanism. Any design must be able to physically hold the object being transferred, but it is also important to consider how many objects need to be held by the transfer or if different types of items need to be transferred.

Location Considerations

Another important consideration when designing a transfer is where the transfer will be moving items from and to. Moving objects in multiple axes is difficult and error-prone. When mechanisms are aligned, transfers only have to act in straight lines, avoiding unnecessary complexity. A very common use for a transfer mechanism is to move objects from the intake of the robot to a scoring mechanism. In this case, it can make sense to combine the transfer system with the intake system or scoring system, so only two mechanisms are needed.

Tip: Generally, when designing transfers, you want to avoid as much mechanical complexity as possible. The less moving parts in a mechanism, the lower the chance something can fail.

Transfer Type Considerations

There are several different types of transfers, ranging from linear to rotational transfers, and different ways of actually moving objects inside the transfer mechanism. When designing a transfer, a design that has lower mechanical complexity will be prone to less mechanical failures. See *Types of Transfers* (page 181) for examples of common transfer types.

8.9.2 Types of Transfers

There are a lot of different styles and types of transfers, but most are a form of Direct Transfer, Flip Up Transfer, Grate Transfer, or Conveyor Transfer.

Direct Transfer

The simplest form of transfer is a direct transfer. This is the term for when the mechanism that an item is being moved to is simply placed directly behind whatever is gathering the item. The most common form of this is putting an arm or bucket directly behind the intake to collect the item as soon as it enters the intake. Since there is no mechanism between the intake and the outtake system, this is also referred to as a “transferless” system.

Advantages

- Least mechanical complexity of any transfer system
- Doesn’t take up much space
- Fastest transfer system

Disadvantages

- Doesn’t actually move the item internally
- Requires the scoring mechanism to be able to go both behind the intake as well as into whatever position is needed for scoring

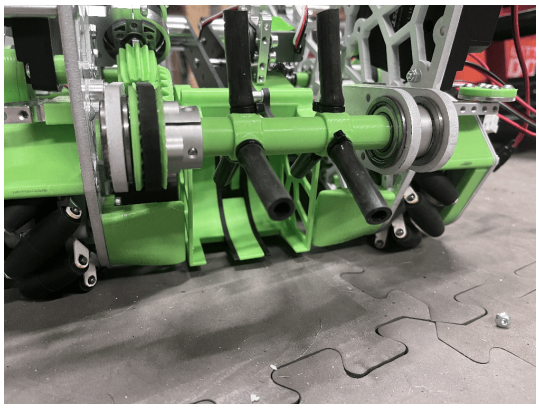


Fig. 96: 16461 Infinite Turtles, Freight Frenzy, Direct transfer intake, where the collection bucket is right behind the intake

Flip Up Transfer

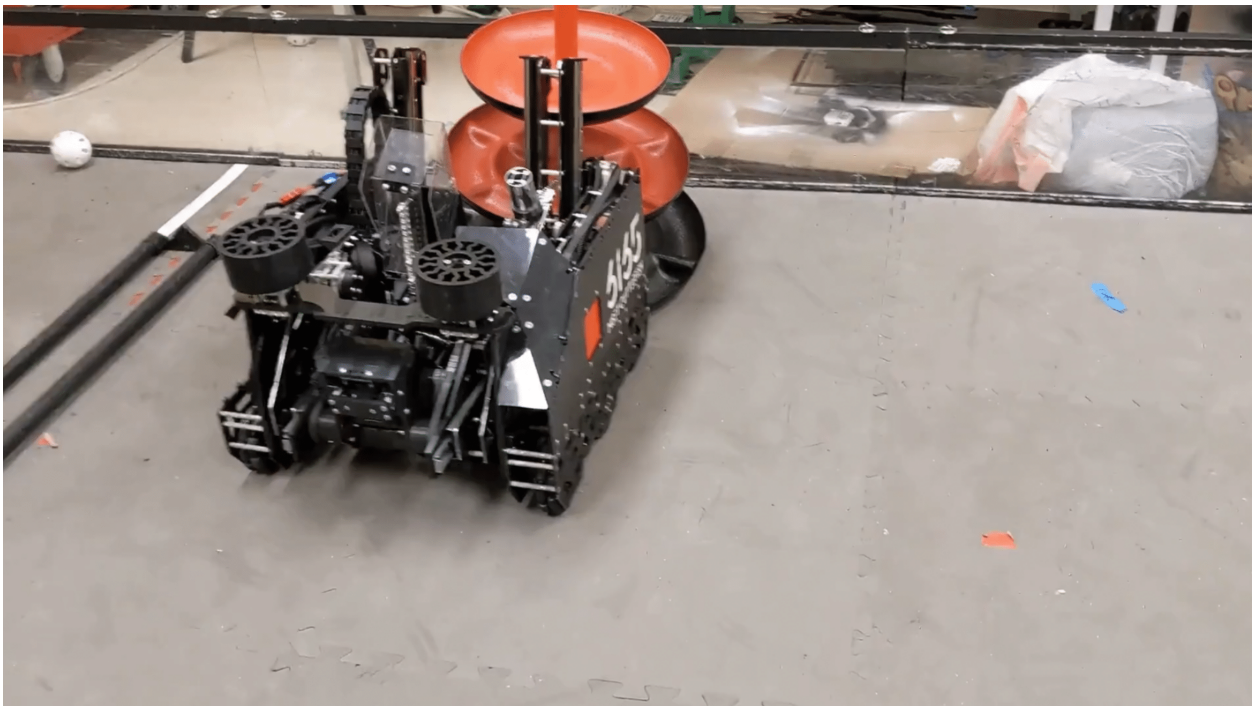
Flip up transfers are a form of transfer where the mechanism holding the item is rotated around a pivot point to align with a second mechanism that then collects the item. This mechanism is very commonly used on intakes to gain vertical height as well as some horizontal distance. In this use case, the intake can simply be mounted onto the transfer system, then spun in reverse to eject the item from the transfer mechanism into the scoring mechanism.

Advantages

- Doesn't take up much space
- Intake can be run co-axially to the flip up system so the motor doesn't have to be moved with the mechanism
- Allows intake to stick outside of the robot and start rotated up to fit inside the 18 inch sizing cube, enabling more flexibility

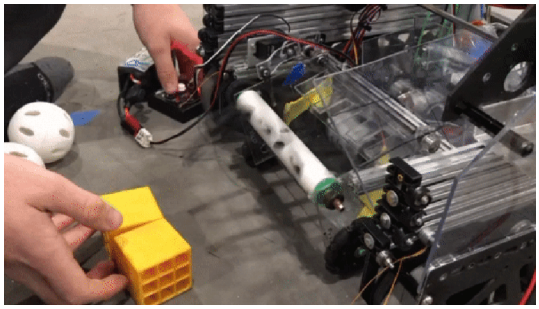
Disadvantages

- Limited in horizontal and vertical movement
- Care must be taken that mechanisms align properly to each other
- Slower form of transfer then grate or direct
- Mechanism can be hit by other robots when outside the robot frame



6165 MSET Cuttlefish, Freight Frenzy, Flip up intake, mounted on linear slides for increased horizontal reach

7236 Recharged Green, Rover Ruckus, Flip up intake



Grate Transfer

Grate transfer is a form of transfer where two sets of “forks” are used on both the transfer mechanism and the scoring mechanism, offset from each other. This means the forks can pass by each other without intersecting, allowing an item to seamlessly go from being supported by one set of forks to the other. This provides really fast transfer times, but also limits how the transfer can operate. Generally, this type of transfer is used with two sets of linear slides at an angle to each other, most commonly with one at a horizontal and one more vertical.

Advantages

- Low mechanical complexity
- Second fastest transfer system (instantaneous)
- Integrates well with linear mechanism (such as extending intakes and outtakes)

Disadvantages

- More complex design, careful alignment needed to prevent the “forks” from colliding
- Doesn’t integrate easily with rotational mechanisms
- Limiting in range of motion, mechanisms have to move in specific orders for the transfer to work correctly

Animated Grate Transfer Example (Click To Expand)

Fig. 97: 6929 Data Force, Rover Ruckus, Grate transfer deposit. Forks are visible on the bottom, which go between forks on the intake bucket.

Conveyor Transfers

Conveyor transfer is a catch all term for transfer system that uses components to linearly move objects internally. Conveyors can use rollers, belts, surgical tubing, rubber bands, and other materials to move the items. These materials then form conveyors that seamlessly move the object linearly from one mechanism to another. Conveyors are generally either roller conveyors, where a series of rollers move parts, or continuous conveyors, where a continuous object moves the items. In addition, a hybrid conveyor can be used which is made up of both rollers and continuous objects in between.

Advantages

- Can create “buffers” where multiple items can be stored
- Items can be moved across complex paths internally
- Items can be continuously transferred instead of having discrete back and forth motions

Disadvantages

- Items can jam on the entrance and exit of the conveyor, especially when multiple elements are entering at once.
- Significant mechanical complexity

There are different types of conveyors that can be used.

Roller Conveyor

Roller conveyors use a series of rollers or wheels to move objects from the beginning of the conveyor to the end. These transfers allow for the use of compliant wheel for different or odd shaped items, but care must be taken that objects don't get stuck in between rollers. In addition, wheels tend to have better grip then many forms of continuous conveyors. Surgical tubing can also be used instead of rollers.

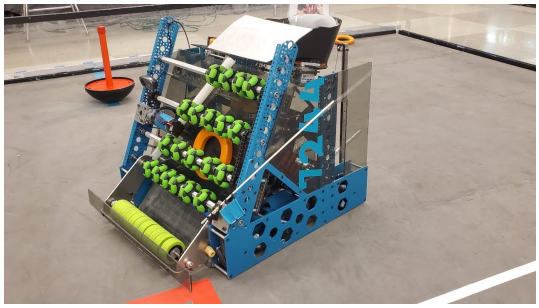


Fig. 98: 7244 OUT of the BOX Robotics, Ultimate Goal, Roller conveyor transfer intake, where a series of omni wheels moves the game element rings at a high vertical angle

Continuous Conveyor

A continuous conveyor uses a continuous object, such as belts, surgical tubing, or rubber bands to move objects. These transfers generally have less compliance than roller intakes, but also continuously contact the object being moved. This can lead to fewer jamming problems. However, the common objects used, such as belts and rubber bands, don't have much grip, so they may slip when transferring objects.

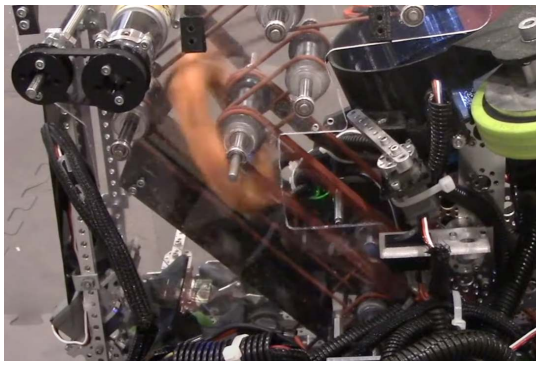


Fig. 99: 8644 Brainstormers, Ultimate Goal, Conveyors which moves rings using o-ring belts internally

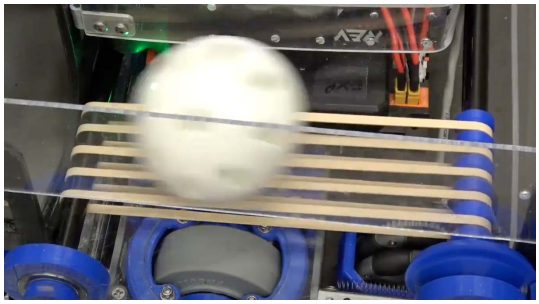


Fig. 100: 11115 Gluten Free, Ball transfer using rubber bands

Hybrid Conveyor

Hybrid conveyors use rollers with continuous objects such as belts and rubber bands between them. This solves the contact issues of roller conveyors while also having more grip then purely continuous conveyors. However, these conveyors have to be designed to allow both the roller and continuous materials to make contact with the item.

8.10 Dead Wheels

Warning: This is a very niche aspect of design in FTC®. Generally it is something done by more experienced teams who have had time to repeatedly test their designs and mechanisms with software during the off-season.

The term dead wheels, tracking wheels, odometry pods, and odometry are often conflated in the FTC community. However, there are a few key differences one must keep in mind. Odometry is an umbrella term and refers to the general use of motion sensors for localization purposes. Meanwhile, dead wheels, tracking wheels, and odometry pods are all synonymous terms.

Term

Dead Wheel A small unpowered wheel (usually an *omni wheel*) that tracks the distance the robot has traveled through the encoder attached to the wheel's axle.

Usually, there will be two or three wheels - one or two on the x and y axis each to track the front-back and left-right position relative to the starting point. Generally, odometry wheels are sprung so that the wheel is in contact with the floor tiles at all times to ensure accuracy.

Odometry refers to the use of motion sensors for localization. Localization is a means for being able to locate the position of the bot at some point in time. Localization is crucial in path following and advanced autonomous modes as one needs to know where they are to generate the necessary movements needed to reach a desired destination. *Localization software* (page 338) plays a major role in odometry; however, in order to produce accurate results, reliable and accurate hardware design is a necessity.

The simplest form of odometry is drive encoder localization. This is the use of encoders measuring the rotation of motors that power the drive train. One is able to read the encoder data and feed it through the kinematic equation for that specific drive train to derive the body's velocity. Drive encoder localization is

generally quite simple and easy to setup as almost all of the FTC legal motors have built-in encoders. Getting drive encoder localization setup is simply a matter of plugging in wires, no additional hardware needed.

Many teams in the community have converged on a unique solution that isn't seen very much outside of FTC: the use of "dead wheels," "tracking wheels," or "odometry pods" (these terms are all synonymous). These refer to small "dead" or non-driven (not powered by a motor) wheels attached to an *encoder sensor* (page 189). Two or three dead wheel pods are often sprung to the ground to ensure accurate tracking. The two-wheel design utilizes one parallel and one perpendicular pod (parallel and perpendicular with respect to the drive wheel axis), measuring x and y movement respectively. Change in heading is measured via a gyroscope. The three-wheel design utilizes two parallel and one perpendicular pod, measuring x and y movement respectively. However, this design forgoes the gyroscope and instead measures heading via the difference with the two parallel wheels. This is often more accurate in the context of the FTC control system because the BNO055 IMU (used for the gyroscope in the two-wheel design) utilizes I2C which is slower than the rest of the I/O on the REV Hub and cannot be bulk read. These two issues lead to minute drift issues which can compound over time, thus leading to a more inaccurate localization system when using the two-wheel design.

However, designing consistently accurate dead wheels proves to be a difficult design challenge. It is often quite pricey. A set of three dead wheels will cost a minimum of $100 for the encoders alone, prior to any hardware.

Let's go through the advantages and disadvantages of each system.

8.10.1 Drive Encoder Localization

- **Pros**:
  - Cheap (the motors you're using most likely already have encoders built in)
  - Accessible
  - Very little configuration necessary
- **Cons**:
  - Drive encoder localization on mecanum drive can be quite inaccurate due to lack on traction on mecanum wheels.
  - Will drift on high acceleration on mecanum drive. Accuracy will be good enough for basic autonomous modes if acceleration is limited

8.10.2 Two-Wheel Odometry Pods

- **Pros**:
  - Cheaper than 3-wheel design
  - Pretty good accuracy
  - No tuning of the heading necessary
- **Cons**:
  - Subject to more drift than the 3-wheel design

8.10.3 Three-Wheel Odometry Pods

- **Pros**:
  - Relatively accurate tracking. Great accuracy in a 30-second autonomous mode
- **Cons**:
  - Quite pricey
  - Tuning of the heading is very important

8.10.4 Encoders

A lot of the localization done in software relies on readings from encoders. *Encoders* (page 230) are sensors that track "counts" or "ticks," which are values that represent a certain amount of a rotation. Different encoders might have a different number of counts per revolution (CPR), which is also sometimes also called ticks per revolution. The greater the number of counts, the more precise the data.

Encoders are plugged into the JST-PH ports in the REV hubs. These encoders can either be built-in to the motors or external. External encoders will still need to be plugged into an encoder port but are not related to the motor in that port. Through software, we can use the motor object to determine the position of the encoder. This should be done with motors that do not use encoders. If you're using dead wheels, you will not need the drive motor encoder ports, so those are potential ports you might want to use.

If one chooses to design dead wheels, there are only two recommended encoders that one should use for FTC: REV Through-Bore Encoders and U.S. Digital S4T Encoders.

REV Through-Bore

Often short-handed to "REVcoders" or "revcoders," the REV Through-Bore encoders[86] has quickly become the de facto option the FTC community. The REV encoders have gained such a reputation due to its relative affordability, much improved reliability, and ease of use. The through-bore design proves to be a *significant* improvement over previous optical disc encoder designs. Optical disc encoders are very fragile, prone to scratching, and are much less tolerant to design flaws.

Advantages:

- Through-bore design is very robust and easy to design with
- Relatively cheap
- High CPR
- Easy wiring

Disadvantages:

- Quite large relative to other encoders. May be challenging to create a compact design
- Many Through-Bores seem to experience slight, uneven resistance when rotating. REV says this is normal and will subside as the encoder wears in
  - To forcefully wear in a REV Through-Bore encoder a 1/2" hex shaft can be spun on a drill through the encoder for a couple of minutes
- Odd mounting points

[86] https://www.revrobotics.com/rev-11-1271/

Fig. 101: REV Through-Bore Encoder

Note: The Through-Bore encoders have a very high CPR (8k). The REV Hub transmits velocity in a 16-bit signed integer. This means it can only communicate a maximum value of 2^15 (which is 32768). Thus, it only takes 4 rotations a second (32k / 8k = 4) for the velocity value on the REV Hub to experience an integer overflow[87]. This is primarily a concern when dealing with motion profiling. The popular, existing tools (Road Runner and FTCLib) have mechanisms for dealing with this issue[88] so this is not a concern and should not sway your design decision. Just keep this detail in mind once you start programming.

U.S. Digital S4T

The S4T[89] miniature shaft encoder is another viable option used in dead wheel designs. These encoders are very small and may significantly reduce the footprint of your dead wheel design. Gearing these encoders is ideal to prevent shock loads.

***Advantages:**

- Very compact

Disadvantages:

- More expensive (nearly double the price)
- Less durable
  - Very thin wires. Prone to breaking easily if not secured properly
- Ideally requires external gearing

[87] https://en.wikipedia.org/wiki/Integer_overflow?oldformat=true
[88] https://github.com/acmerobotics/road-runner-ftc/blob/e79f8a900f45c9058b67716b5289a52e17769e40/RoadRunner/src/main/java/com/acmerobotics/roadrunner/ftc/Encoders.kt#L66
[89] https://www.usdigital.com/products/encoders/incremental/shaft/S4T

Fig. 102: S4T Encoder

SRX Mag Encoder

The SRX Mag Encoder[90] from Cross The Road Electronics is a magnetic encoder. It is not used by many FTC teams due to its slightly higher complexity to use and lack of FTC centric documentation. It is more popular in FRC®.

Fig. 103: CTRE SRX Mag Encoder

Advantages:

- Very compact
- Relatively cheap

Disadvantages:

- Requires assembly
- Not much information exists for use in FTC

[90] https://store.ctr-electronics.com/srx-mag-encoder/

U.S. Digital E8T (deprecated)

Once the de facto option for most FTC teams, the E8T[91] optical encoders are no longer recommended as the REV Through-Bores are a superior option at an equivalent price. The open-hole optical disc design of these encoders face a number of frustrating design flaws that made them very fragile and prone to breaking. The only advantage that they have relative to the REV Through-Bores is their smaller footprint.

Fig. 104: E8T Encoder

8.10.5 Design

There are few open source dead wheel designs. Dead wheels are often designed around a team's own drive train and FTC teams seldom publicly release their own robot CADs.

Here are a few publicly available dead wheel designs:

- **Open Odometry by 18219**
  - https://openodometry.weebly.com
  - Utilizes the REV Through-Bore Encoder
  - Most popular and robust publicly available design
  - Compact enough to fit into a goBILDA channel
  - **Things to consider**:
    - Utilizes Rotacaster 35mm wheels from Australia. Shipping may take a while
- **goREVdometry**
  - https://discord.com/invite/Cvz3MbM9dX
  - Utilizes the REV Through-Bore Encoder
  - Compact enough to fit into a goBILDA channel
  - **Things to consider**:
    - Information only available through their Discord channel
    - Hasn't been iterated on in a while
- **11115 Gluten Free Design - 2019**

[91] https://www.usdigital.com/products/encoders/incremental/kit/E8T

  - https://drive.google.com/file/d/16ZQRSiWdzTKSH92VpKrxKpXy3TTh0sA5/view?usp=sharing
  - The above link the entire robot assembly for 11115's CAD for the 2018-19 season
  - **Things to consider**:
    - Uses LEGO® gears
    - Uses US Digital S4T's. Quite pricey
- **9794 Wizards.exe Design**
  - https://www.youtube.com/watch?list=PLICNg-rquurYgWAQGhu6iC0At75vgqFJp&v=OjNvAD350M4&feature=emb_title
  - Compact enough to fit into a goBILDA channel
  - **No longer recommended as it utilizes the E8T**

Spring Tensioning

It is *highly* recommended that your dead wheel design includes some form of spring tensioning that pushes the wheel into the ground. This ensures that the wheel is always in contact with ground and has adequate traction. Sufficient force is required to ensure constant traction to prevent the wheels from slipping. Keep in mind that too much force may lift a light drive train off the ground and disrupt driving.

The most popular method of spring tensioning is to pivot your pod around a point and provide a rotational force via a spring or rubber band.

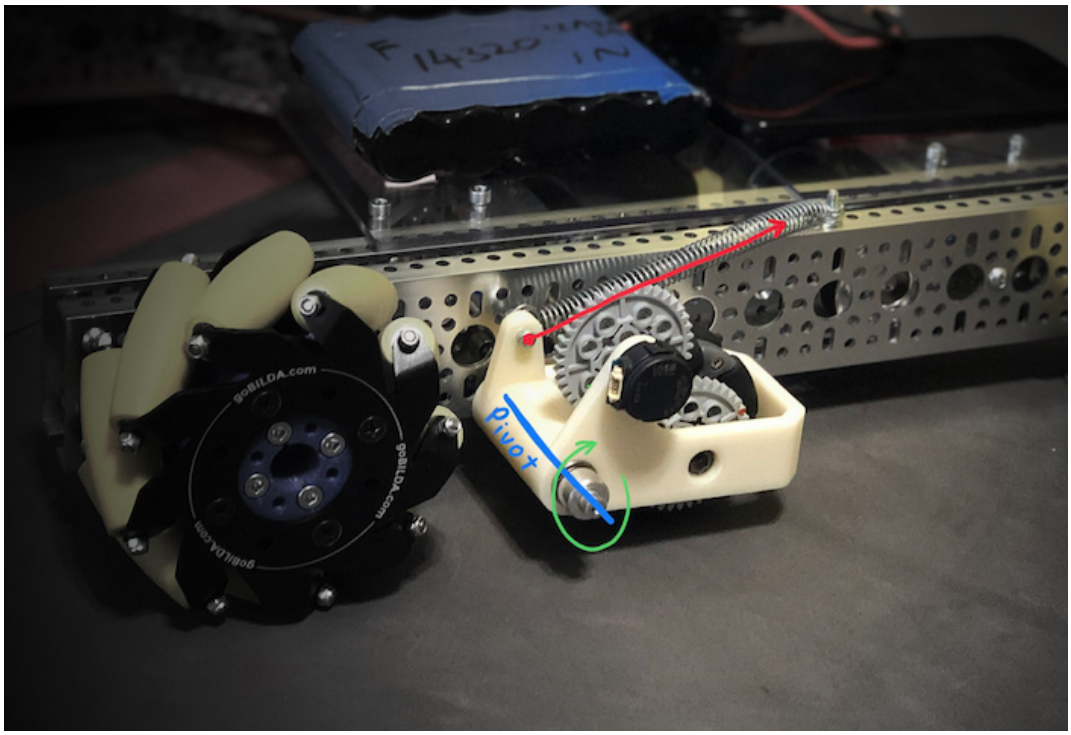


Fig. 105: FTC 14320's spring tensioning

A much more niche option is to vertically spring odometry pods. The idea is that springing around a pivot will cause the dead wheels to move in the axis parallel to the ground if the height of the dead wheels relative to the ground changes. Vertically sprung odometry pods will not experience such an issue. However, this is

not really an issue that most teams will experience. Vertically springing is much harder to design well and is not recommended for the relatively minor improvement in accuracy it yields.

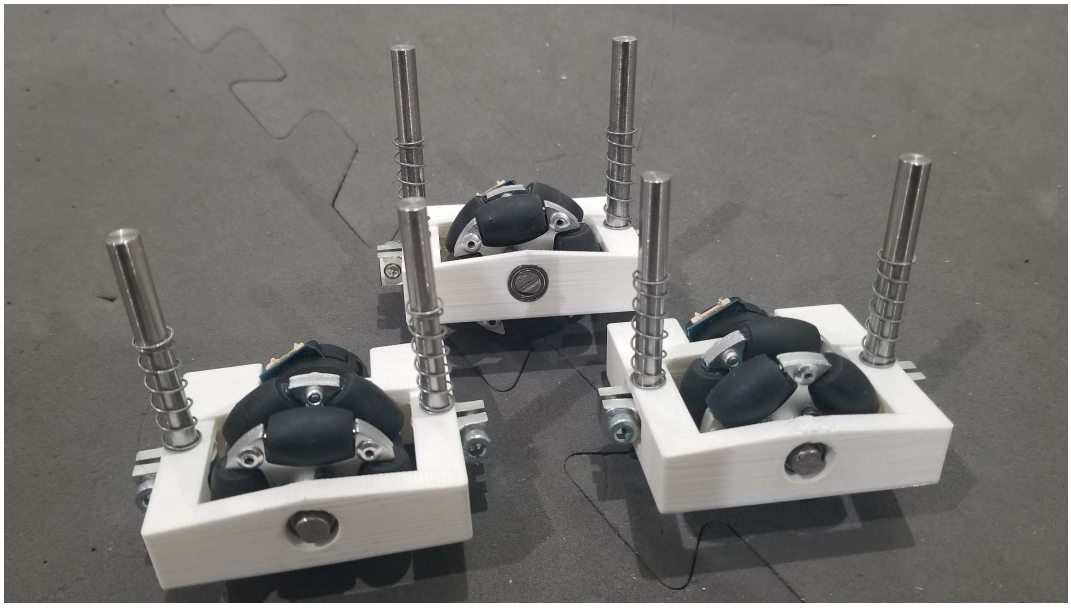
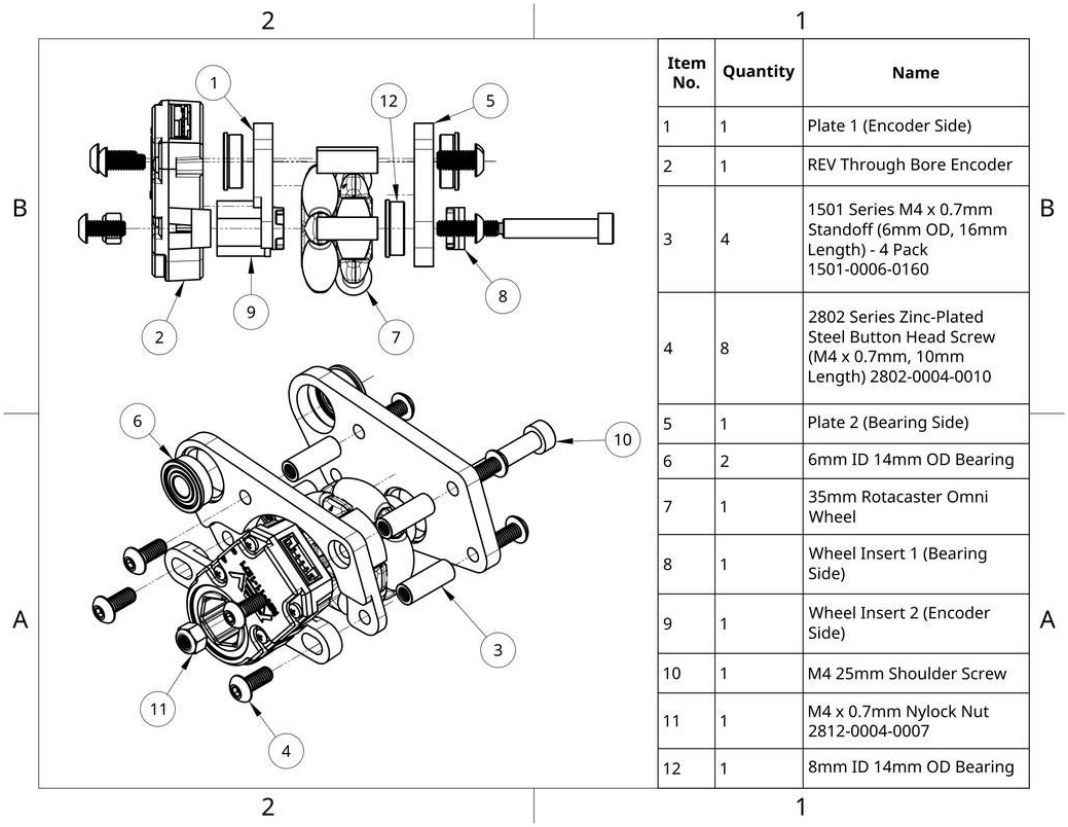


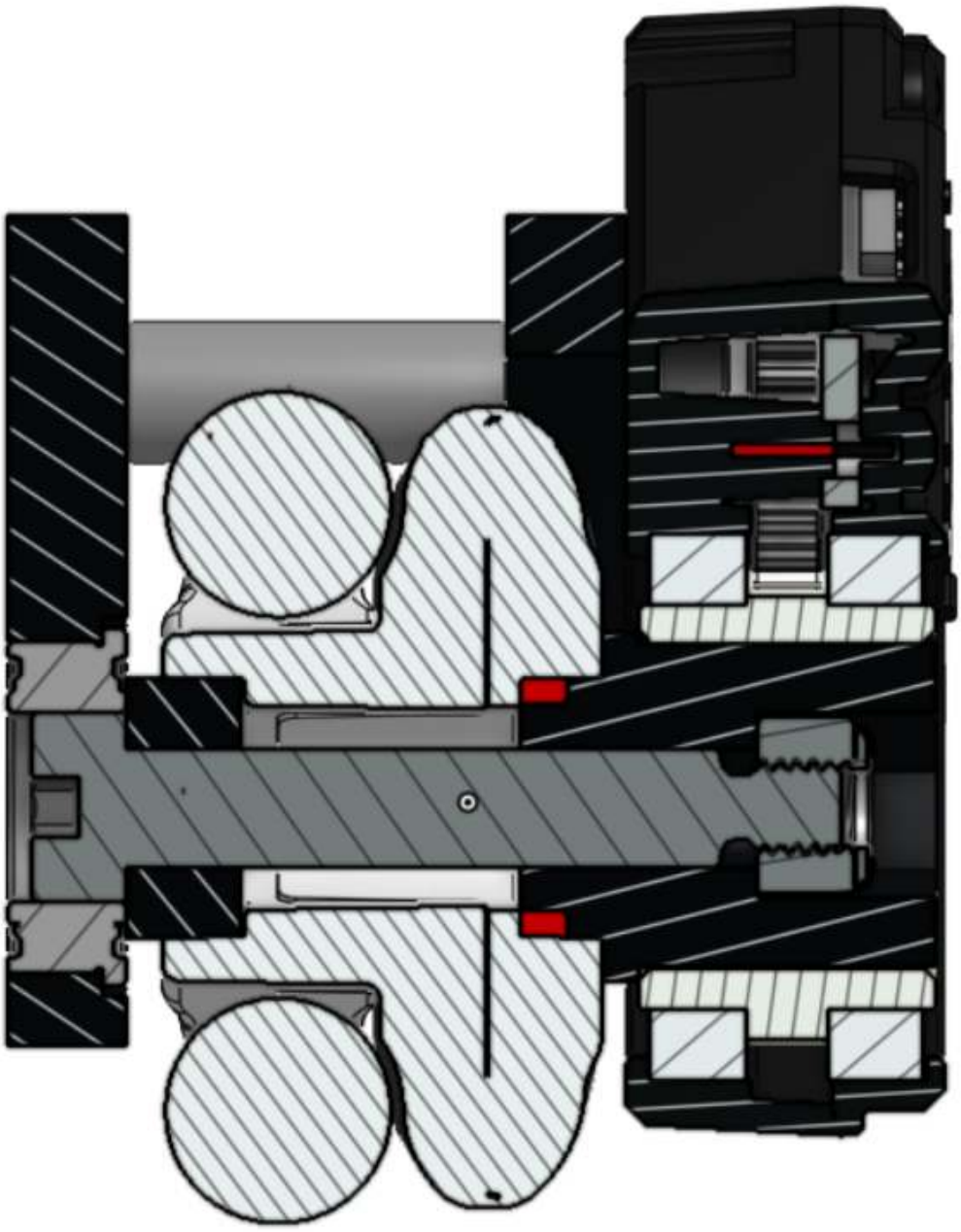
Fig. 106: FTC 18172's vertical springing

8.10.6 Gallery

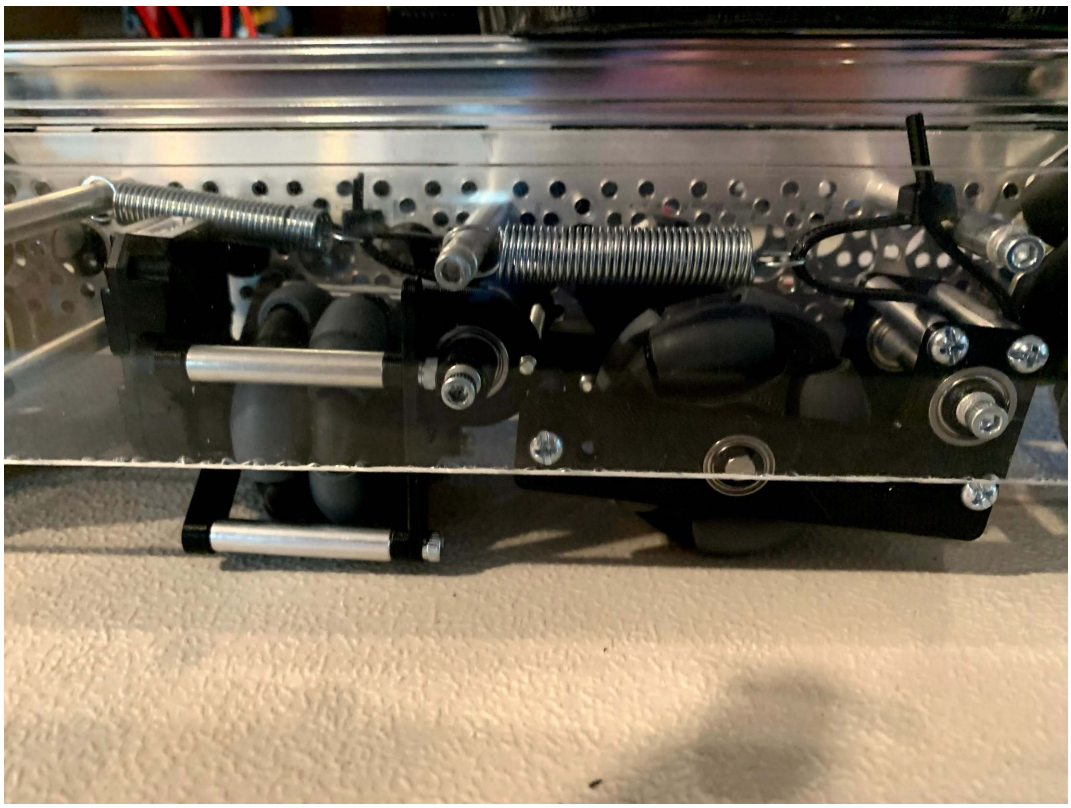
Open Odometry (REV Through Bore Encoder)



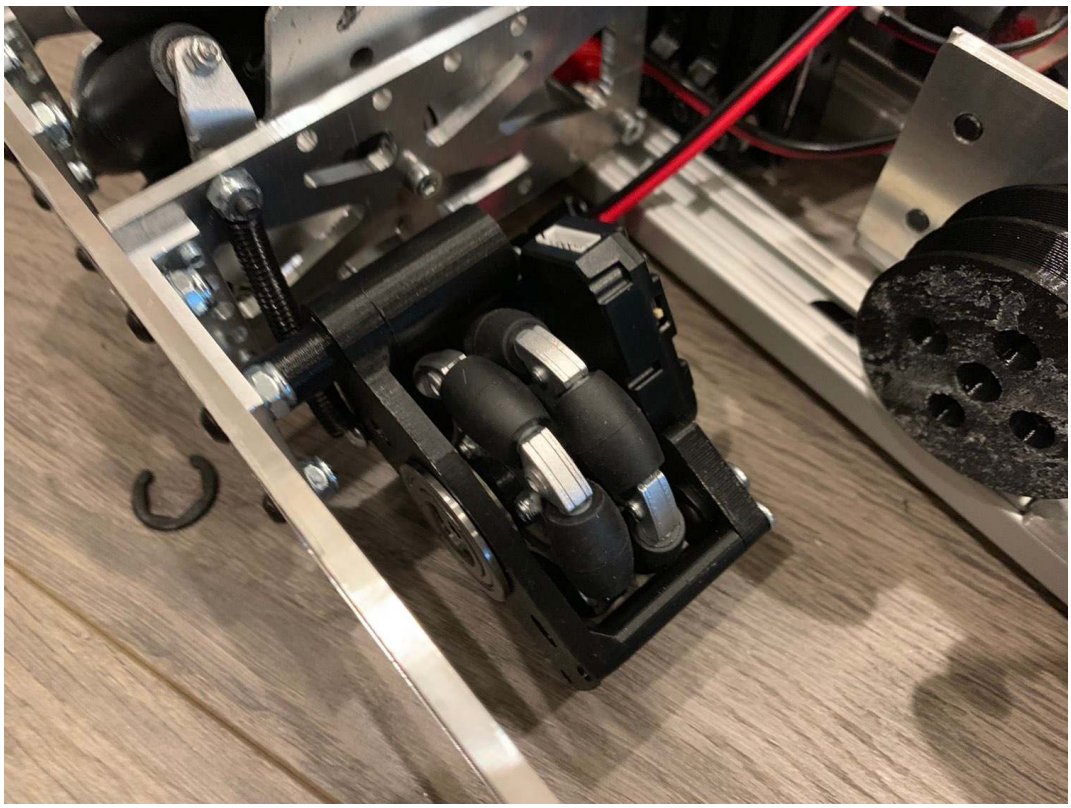
Item No.	Quantity	Name
1	1	Plate 1 (Encoder Side)
2	1	REV Through Bore Encoder
3	4	1501 Series M4 x 0.7mm Standoff (6mm OD, 16mm Length) - 4 Pack 1501-0006-0160
4	8	2802 Series Zinc-Plated Steel Button Head Screw (M4 x 0.7mm, 10mm Length) 2802-0004-0010
5	1	Plate 2 (Bearing Side)
6	2	6mm ID 14mm OD Bearing
7	1	35mm Rotacaster Omni Wheel
8	1	Wheel Insert 1 (Bearing Side)
9	1	Wheel Insert 2 (Encoder Side)
10	1	M4 25mm Shoulder Screw
11	1	M4 x 0.7mm Nylock Nut 2812-0004-0007
12	1	8mm ID 14mm OD Bearing



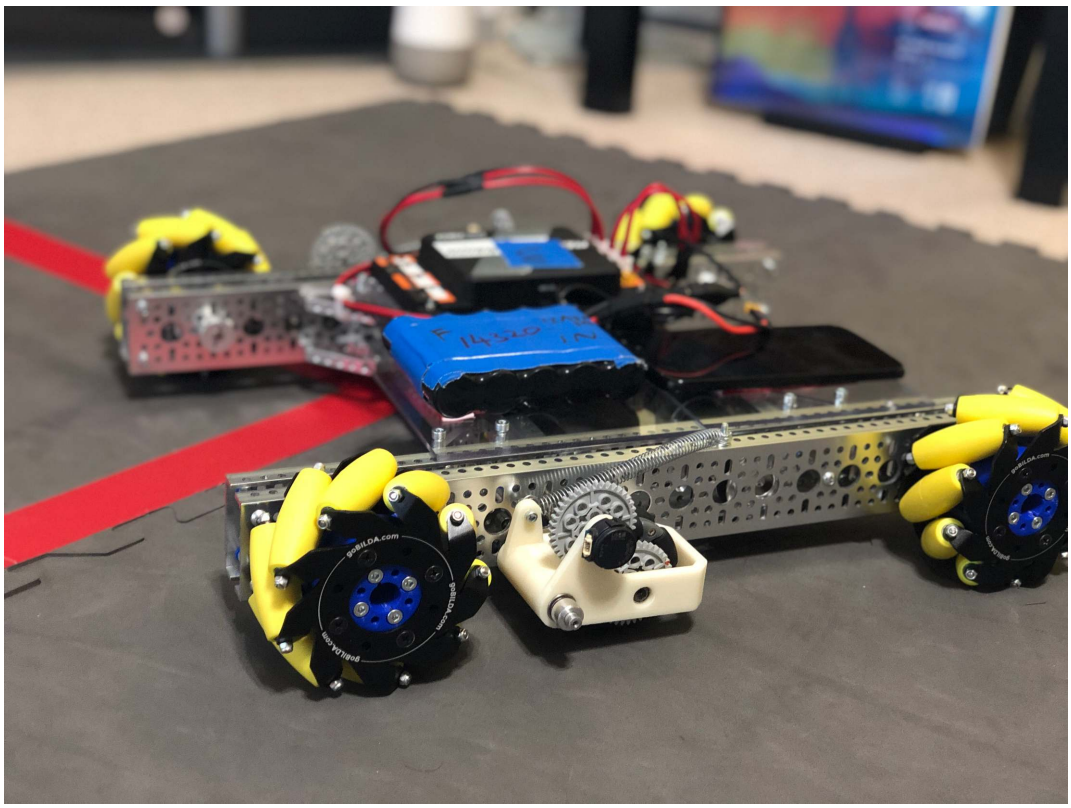
FTC® Team 14310 (REV Through Bore Encoder)



FTC Team 8802 (REV Through Bore Encoder)



FTC Team 14320 (US Digital S4T)

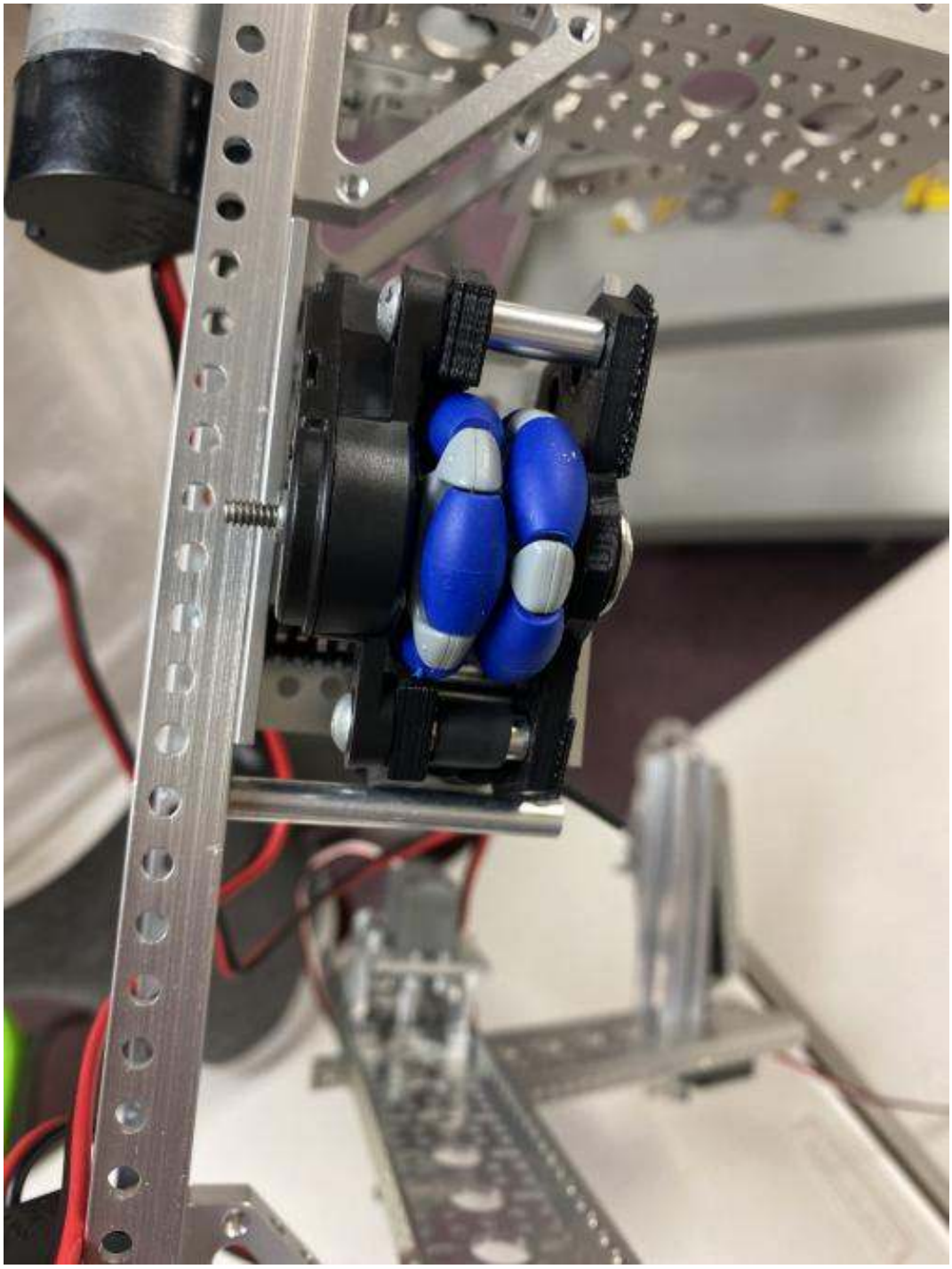


FTC Team 11115 (US Digital S4T)

[92] https://photos.google.com/share/AF1QipPx5inCdVxK6wAqtIznFE-KqvnuzgRq9rFxrhzI50r0DeYYo2o11hWB4hroYObm8A?key=UWwxd3hFdXpYaHFqaFhTSFJnWFlEWjgtV1FTN3Zn

Fig. 107: FTC Team 11115 Photo Album[92]

FTC Team 14481 (REV Through Bore Encoder)



FTC Team 3658 (REV Through Bore Encoder)

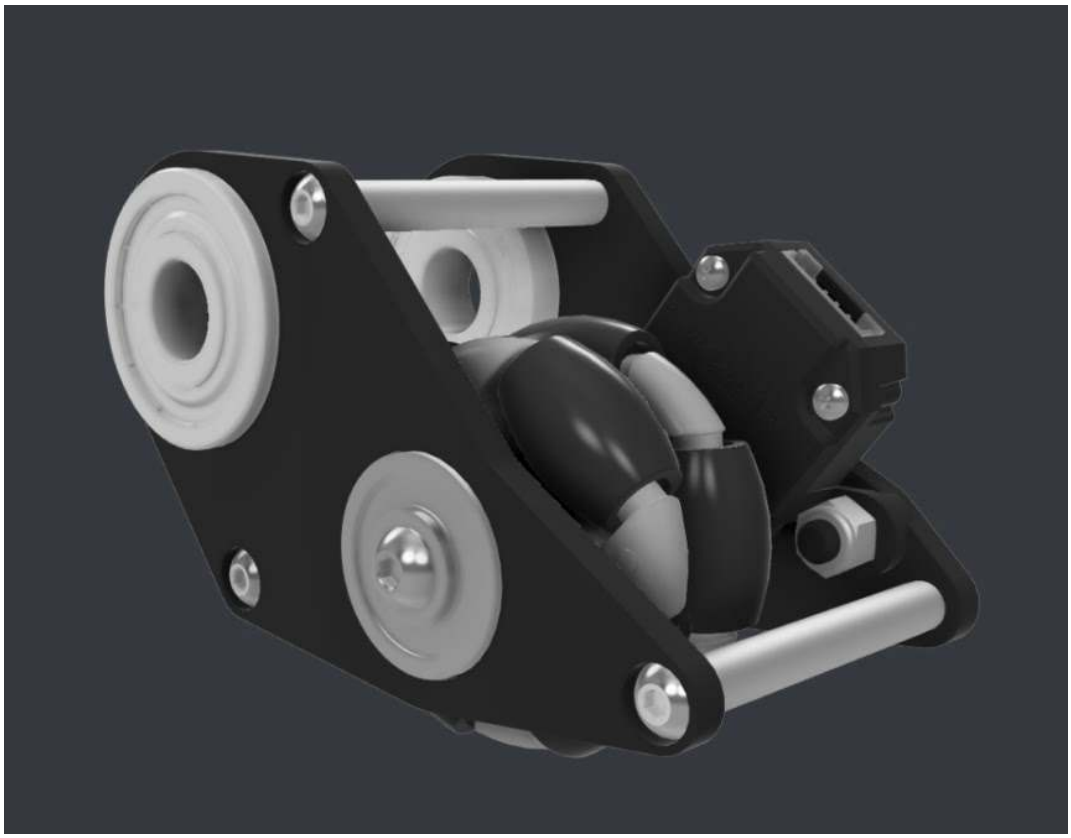
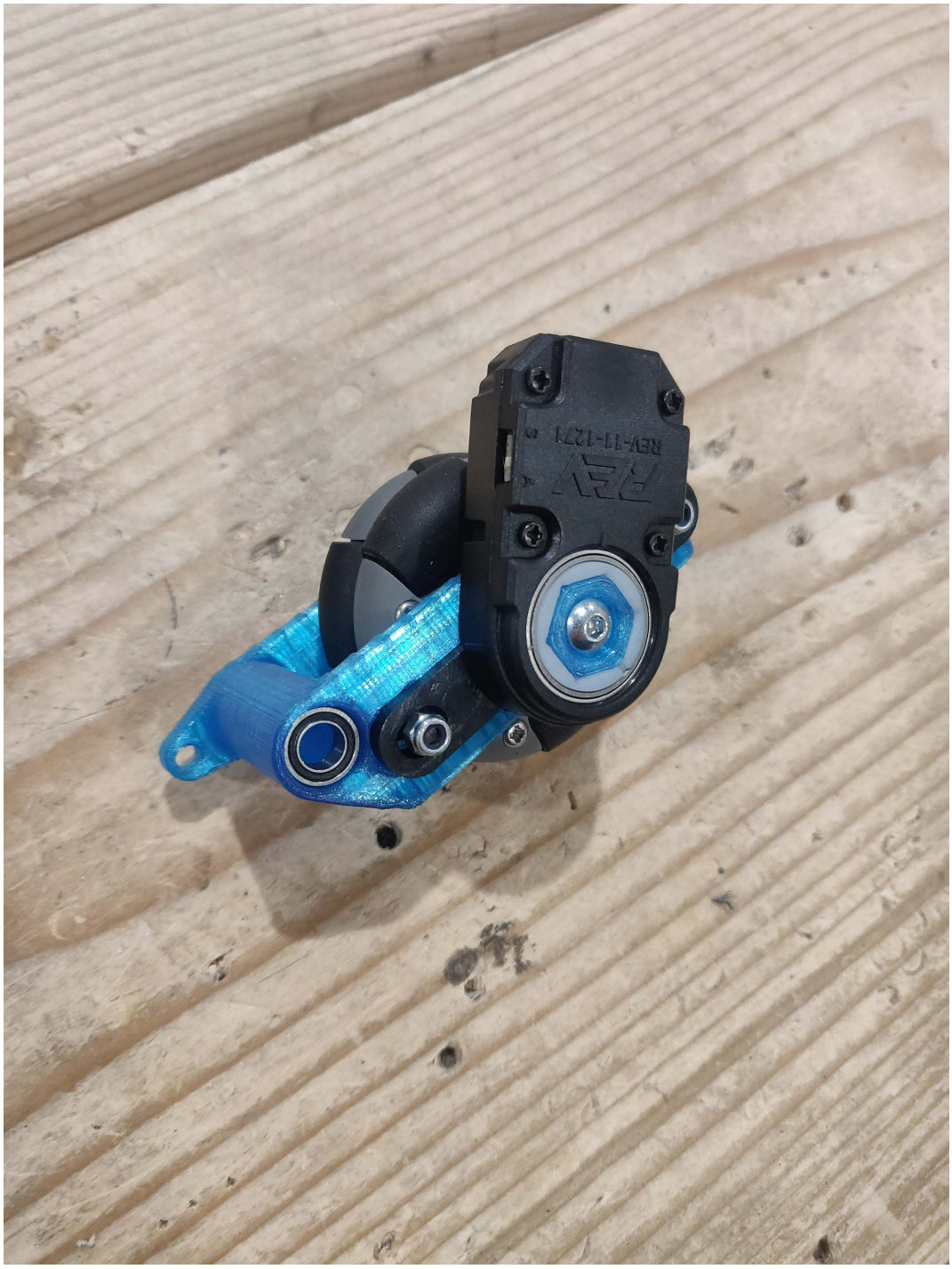


Fig. 108: FTC Team 3658 CAD

Fig. 109: FTC Team 7236 CAD

FTC Team 7236 (REV Through Bore Encoder)



8.11 Turrets

In FTC®, turrets refer to mechanisms that allow for yaw (side-to-side) rotation of another mechanism. This is typically done for the purpose of positioning an intake or scoring mechanism. For example, a shooter could be mounted on a turret to allow for aiming without turning the robot. Turrets typically have a gear, sprocket, or pulley mounted to them, which motors and servos use to rotate them.

The FTC community often splits these turrets into 2 categories, "full turrets" and "mini turrets." Full turrets are typically those powered by motors or multiple servos at the base of the robot, and move large mechanism(s), and what the lone word "turret" generally refers to. Mini turrets are typically powered by one or two servos, are located at the end of an extension of some sort, and only pivot an end effector.

8.11.1 Full Turrets

Listed below are different techniques to implement a full turret. These mechanisms are commonly located at the base of a robot and rotate slide, arm, or shooter assemblies mounted to them.

Lazy Susan Turrets

Lazy susan turrets are based around a type of turntable called a lazy susan, which are available off the shelf.

Advantages

- Lazy susans come prebuilt, meaning you don't need to design or manufacture your own platform for your turret.
- Decent lazy susans are relatively cheap.

Disadvantages

- Lazy susans are prebuilt, meaning they're generally harder to customize than other options.
- Lazy susans aren't designed to handle lateral forces, which means that e.g. the reaction force from a shooter could damage it.

Bearing Stack Turret

Bearing stacks are stacks of bearings, usually consisting of a small bearing sandwiched between two large bearings on a screw shaft or standoff. A bearing stack turret has a disc that rotates smoothly because it is tangent to the small bearing on many bearing stacks, while the bigger bearings constrain the disc vertically. Usually, the small bearing is a radial bearing, while the big bearings may be thrust or radial bearings, depending on which loads the turret experiences.

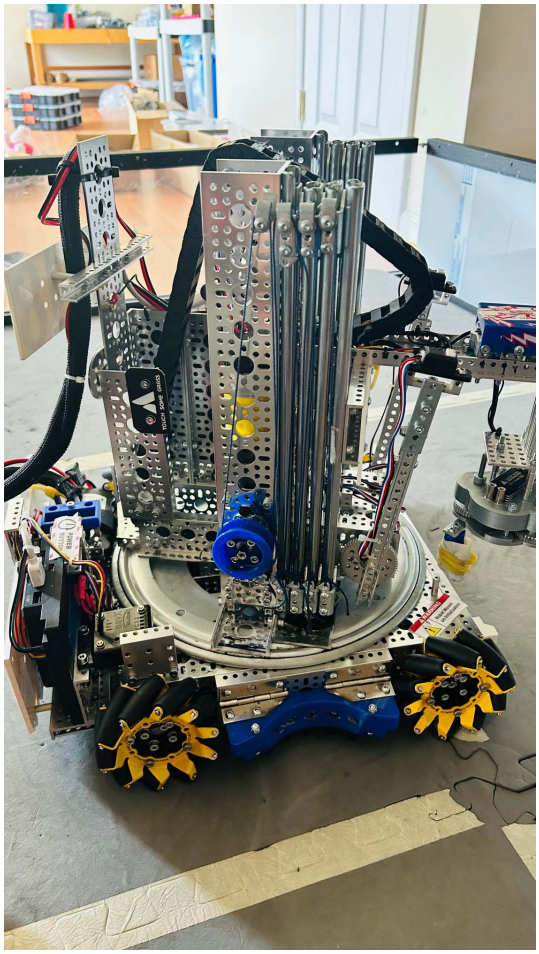


Fig. 110: 19818 Jade Innovations, Power Play, **lazy susan turret rotating slides**

Fig. 111: A sample bearing stack, with a radial bearing sandwiched between 2 thrust bearings on a shoulder screw. A plate (not pictured) rides on the radial bearing while being supported by the thrust bearings.

Advantages

- Highly customizable, as the assembly can be designed specifically for your robot and mechanisms.

Disadvantages

- Requires a lot of design work and precision manufacturing capabilities to build.
- Many potential points of failure, though some bearing stacks can fail without the entire turret failing with redundancy.

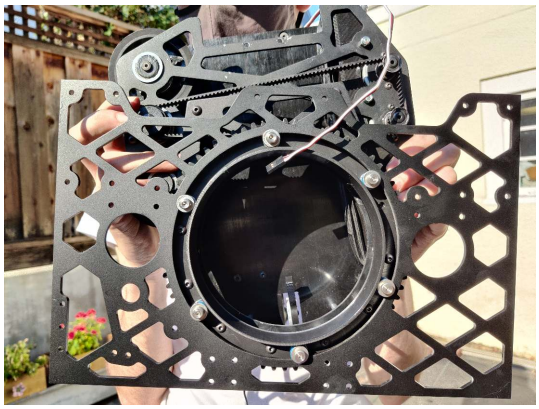


Fig. 112: 18219 Primitive Data, Ultimate Goal, **bearing stack turret**

Center-Bearing Turrets

Center-bearing turrets are based on a bearing (or bearings) coaxial with the turret's axis of rotation.

Advantages

- These turrets are generally simpler than others, as they rotate similarly to any other mechanism on an axle.
- Do not require complex or custom-manufactured components.

Disadvantages

- If a radial bearing is used, thrust loads can cause significant problems. Thrust or x-contact bearings may be better suited for this task, especially in high load situations.
- Due to a bearing being in the center, unless you buy large diameter bearings (which can be very expensive), it may be hard to pass game elements (or other things) through the center. This is often relevant for shooters, where it may be desirable to load them in any orientation.

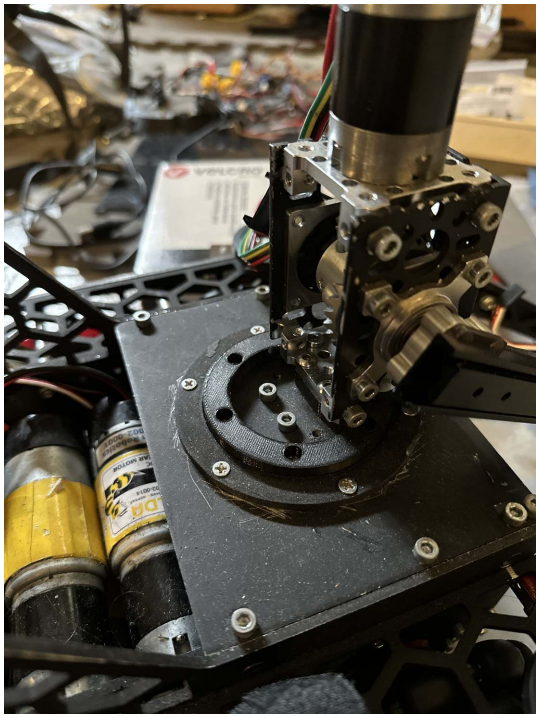


Fig. 113: 16379 KookyBotz, Freight Frenzy, **center-bearing turret using an x-contact bearing**

8.11.2 Mini Turrets

Mini turrets are turrets that pivot something small, typically an end effector. These are often at the end of a linear extension or arm. These are much simpler than full turrets, because they are physically smaller and deal with much smaller loads. There are many ways to implement a mini turret, and generally any of the methods mentioned on the *Full Turrets* (page 205) page will work, but the center-bearing technique is the most common, as it is the simplest.

Advantages

- Allows for accurate mechanism rotation, without drivetrain movement.
- Much simpler than a full turret.

Disadvantages

- Heavier than just an end effector, which is especially important on the end of slides/arms because of mechanism speed and center of gravity.
- More complex than no turret.
- Provide fairly limited range compared to full turrets.

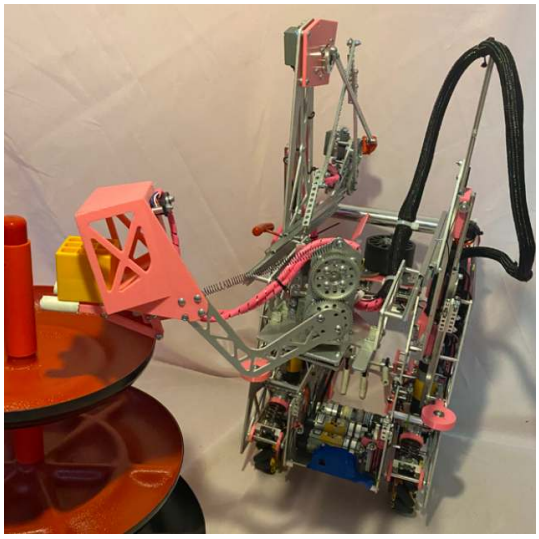


Fig. 114: 12518 Almond Robotics, Freight Frenzy, a **center-bearing mini turret** on the end of slides used to rotate a scoring mechanism with a servo

9 Electronics and Motion Components

This chapter contains information on electric power and electronics of the robot, including information on wiring, batteries, motors, and control system.

9.1 Wiring Guide

9.1.1 Why does wiring matter?

Have you ever looked inside your robot and thought “what a mess of wires”? Wiring is extremely important in FTC®, but is often overlooked or hastily done the hour before competition starts. However, time spent in wiring the robot properly is crucial to the performance and maintenance of the robot.

Attention: It is highly discouraged for teams to overlook wiring, but many new teams seem to disregard it or put little thought into wiring the robot properly.

While tedious and often no fun, wiring can mean the difference between a win and loss. The best robot in the world won’t be able to work if a wire is loose or gets tangled up in the middle of a match. Thus it is imperative that wiring is purposefully thought out when designing and building a robot.

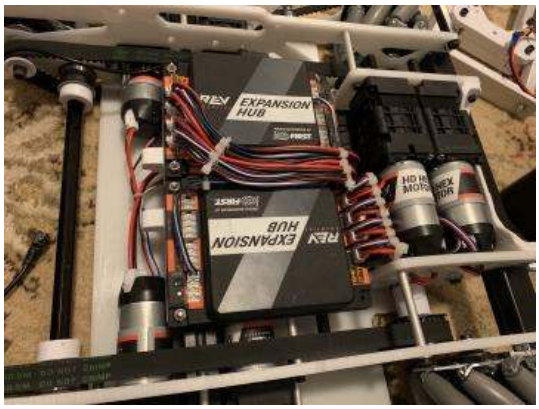
9.1.2 *FIRST*® FTC Wiring Guide

FIRST® has created a wiring guide to help teams with tasks like crimping cables, soldering connections, and ESD mitigation that won’t be covered in this guide. Once you read up on electronics and wiring here, look at the FTC Wiring Guide[93] for the best practices and more tips & tricks. In addition, *FIRST* has written an ESD Mitigation Whitepaper[94] that is worth taking a look at.

[93] https://www.firstinspires.org/sites/default/files/uploads/resource_library/ftc/robot-wiring-guide.pdf
[94] https://www.firstinspires.org/sites/default/files/uploads/resource_library/ftc/analysis-esd-mitigation-echin.pdf

9.1.3 General Advice

- **Always label wires! When bunched up, you may not know which wire goes into which port**.
- Tie together loose wires, and better yet, tie that bunch of wires to a structural component. This will ensure that wires don't interfere with your mechanisms.
- **Pay attention to port numbers!** The rev hub will often have multiple ports per connector on the REV hub. REV has a pinout guide to avoid confusion[95].
- **Treat every wire connection as a point of failure**. Therefore, use electrical tape to tape up and insulate connections and utilize strain relief as much as possible.
- **Strain relief** should be used everywhere possible. It is highly recommended for teams to use products like the *REV USB Retention Mount*, as well as 3D printing strain relief methods for devices such as the Expansion Hub and robot controller phones.
- **DO NOT solder a wire before crimping it**. Solder can "creep" and losing connection is possible, possibly leading to fire.
- **Keep all wire runs as short as possible** to prevent entanglement and improve wire management.
- **When using data/sensor cables, keep them away from motors.** This will reduce electromagnetic interference (EMI). Add a ferrite bead if possible.
- **Crimped connectors are generally better than soldered connectors**, as solder joints can break easier than a crimped connection.
- **Keep wires tucked away from moving mechanisms,** and ensure that you will not be at risk of a mechanism snagging a wire. This is a proper application of materials such as acrylic, which allow drivers to see inside the robot while keeping wires out of the way of other robots/game pieces. It is advised for teams to purchase removable velcro ties or cable ties to aid with cable management.
- **For power wires, lower** *gauge* **(larger size) wires are preferable.** This means a lower resistance across the wire and higher power throughput. This is negligible for data wires, however.
- **Small wires and cables are fragile.** Treat them as such, and don't put them in an area where they will constantly be hit by another object. Larger power cables can take much more abuse.
- **Ensure that your wires are kept out of pinch points** where another mechanism could sandwich the wire. This is especially important in arms or mechanisms that are hinged.



When wiring, also take the time to plan out a specific layout for your wires and how they will run throughout the robot. Take the time to lay out something like an electronics panel if necessary! When building the robot,

[95] https://docs.revrobotics.com/duo-control/control-system-overview/port-pinouts

Fig. 1: Bad example of wiring!

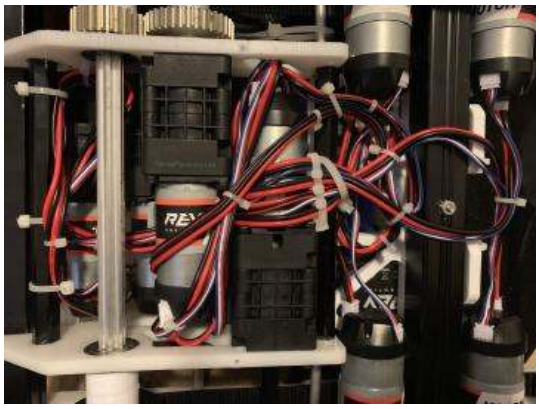


Fig. 2: Good example of wiring by 731 Wannabee Strange, Rover Ruckus

Fig. 3: Good example of wiring by 8417 'Lectric Legends

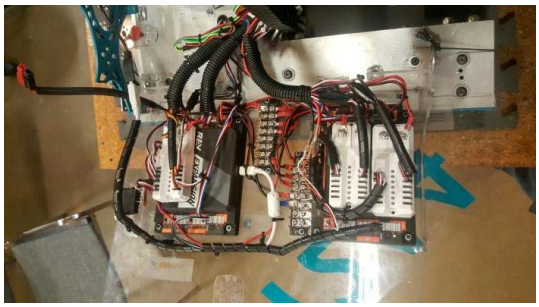


Fig. 4: Good example of wiring by 7244 Out of the Box Robotics

devote adequate space to wiring. This could mean mounting a PVC pipe and running wires from the back end of the robot through it, or simply using velcro or zipties.

Tip: Make sure that wires are as short as possible to reduce the risk of entanglement.

However, as components move out of the robot, the wires move with it. Tying down every loose inch will result in wire disconnecting. Wiring is the art of finding the perfect balance between shortest length and allowing enough freedom as mechanisms need.

It is also recommended for electronics to be mounted on a nonconductive material such as wood to prevent ESD.

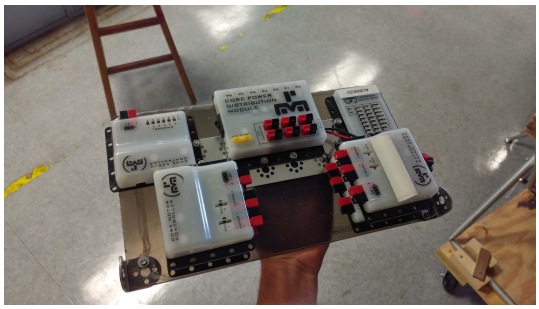


Fig. 5: 3736 Serious Business

9.1.4 Specific Recommendations

Module Power

Be aware, *XT30* connectors can wear out significantly faster than *PowerPole* connectors. Additionally, because *XT30* connectors are soldered, they can break much easier than *PowerPole* connections.

Cabling for module power should be at least 14awg, if not 12awg. Keep in mind that it must be stranded, not solid, wire.

Motor Power

Cabling for motor power should be between 16awg and 12awg. Again, stranded, not solid, wire.

Some motors (like the *REV HD Hex* and *Core Hex*) will have removable power connectors on the back, while other motors (like the *Andymark NeveRests*) will have a cable permanently soldered on the back. It is much more convenient to have a connector on the back, or failing that, have a very short plug on the back of the motor. Once your wires are run and secured, taking them out won't be fun.

Servo Wires

Using heavy-duty extension wires are recommended.

Tape the connections between extension wires and servo wires with electrical tape. This is as the connections can become loose over time and are easy to pull out.

The VEX Motor Controller 29 has the wrong gender connector on the 3-pin end. You are required to either use an adapter cable, or add the right connector to the wires (recommended). **Be sure to protect the MC29, as it is fragile and prone to failure if it takes impact from another object**.

USB

USB is generally a strong connector, but is prone to wearing out over time. Refrain from plugging/unplugging these cables more than necessary, especially on the RC/DS phones.

USB loves strain relief. To keep disconnects low, tie down cables to leave as little loose cabling on the robot as possible.

Sensor Wires/Encoder Wires

Sensor wires and their connectors can be incredibly fragile. Use caution when routing, and keep slack on the connector end when adding strain relief to the cable.

The JST data connectors on the REV Expansion Hub and Control Hub have +5v, GND, and two data pins. If you are using a digital or analog sensor that does not use I2C, you can use a Y cable that gives two sensors off of one port.

9.1.5 Miscellaneous

REV Grounding Strap The *REV Grounding Strap* is currently the only legal way to ground your robot. Attach the end to the metal part of your robot frame, and plug the *XT30* connector end into a free *XT30* port on your robot.

Power distribution blocks/panels The REV Power Distribution Block allows teams to have more than four *XT30* connectors (2 on each Expansion Hub). The block can be connected to the Servo Power Module to boost voltage for servos.

Dryer sheets Dryer sheets can be used to wipe down the robot after every match in order to reduce static buildup. This is not directly recommended by *FIRST* or any vendor, but our empirical evidence throughout the years suggests that it helps, or at the very least, can't hurt to do so. However, dryer sheets may or may not be on the boundary of legality, as grounding the robot to the ground is illegal.

Staticide/static spray Staticide is a spray that helps to keep static off of the robot. Please be sure to spray your robot before an event and not during the event.

Common causes of static

- Every single contact point of your robot to the floor will increase the amount of static buildup.
- Too much turning scrub (or traction when wheels try to turn). This is possible if a 4WD or 6WD (no center drop) with all traction wheels is used.
- A conductive part dragging along the ground. For example, try not to have an intake touch the ground when the robot is moving as much as possible. Foam wheels and foam rollers are a common culprit.

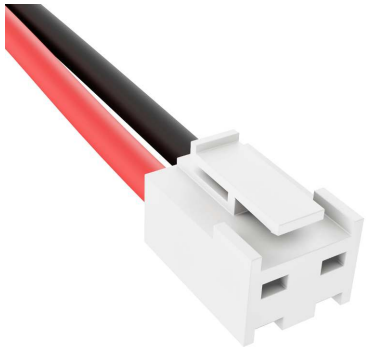
9.2 Connectors and Wires

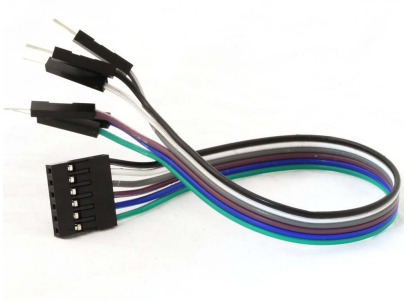
There are many types of connectors for use in FTC®. Here are the most common connectors you will find on an FTC robot.

Anderson PowerPole Anderson PowerPole is a connector used by AndyMark on their *NeveRest motors*. PowerPole connectors are very reliable and recommended for teams. In addition, there are adapters available to other systems.

XT30 The XT30 connector is used in the REV ecosystem through the Expansion or Control Hub. The XT30 through the REV Slim Battery provides power to the Expansion Hub, and teams will need an XT30 cable to transfer power from the main hub to a secondary hub. This is also the connector used on the *REV Grounding Strap*.

JST-VH JST-VH is a type of connector used by FTC motors to interface with the REV Expansion Hub. It is keyed and locks into place for improved reliability.





Dupont 0.1”

A small pinheader connector commonly used on servos and on some sensors.

JST-PH JST-PH is a type of connector. For FTC, the 3-pin and 4-pin options will be used most often. On the REV Control/Expansion Hubs, the 4-pin connector is used for encoder, I2C, analog, and digital connections. The 3-pin connector is used for the RS-485 connection between the Control Hub and Expansion Hub.

JST PH Series JST XH Series

JST-XH JST-XH connectors are used for the encoder connections on goBILDA (MATRIX) and Andymark NeveRest motors. Both vendors sell adapters from JST-XH to the *JST-PH* encoder ports on the REV Control/Expansion Hubs. These connectors are also occasionally found on some third party sensors.

Tamiya Used in some third party boards. Do not use these connectors! The metal connectors are fragile and will lead to random disconnects.

9.3 Control Systems

The FTC® control system is based around a "*Robot Controller*" and a "*Driver Station*". The *Robot Controller* is mounted on the robot. It is either embedded within or connected to special "Hub(s)", which in turn connect to motors, servos, and sensors. The *Robot Controller* is linked to the *Driver Station* through WiFi or WiFi Direct.

REV Robotics is the sole manufacturer of legal FTC control system components. The REV Expansion Hub connects to motors, servos, sensors, and a *Robot Controller*. A REV Control Hub has the same functionality of an Expansion Hub but with a built-in *Robot Controller*.

More information about the FTC Control System can be found below:

- Official Control System Documentation on FTC Docs[96]
- REV Control System Documentation[97]
- Official troubleshooting guide[98]

There are two possible control systems that can be run on an FTC robot legally:

- REV Control Hub + REV Expansion Hub
- *RC Phone* + REV Expansion Hub(s)

Important: Beginning in the 2024-2025 season, an *RC Phone* will no longer be legal to use as a Robot Controller. Teams will be required to use a REV Control Hub as their *Robot Controller*.

---

[96] https://ftc-docs.firstinspires.org/en/latest/programming_resources/shared/control_system_intro/The-FTC-Control-System.html

[97] https://docs.revrobotics.com/duo-control/

[98] https://www.firstinspires.org/sites/default/files/uploads/resource_library/ftc/control-system-troubleshooting-guide.pdf

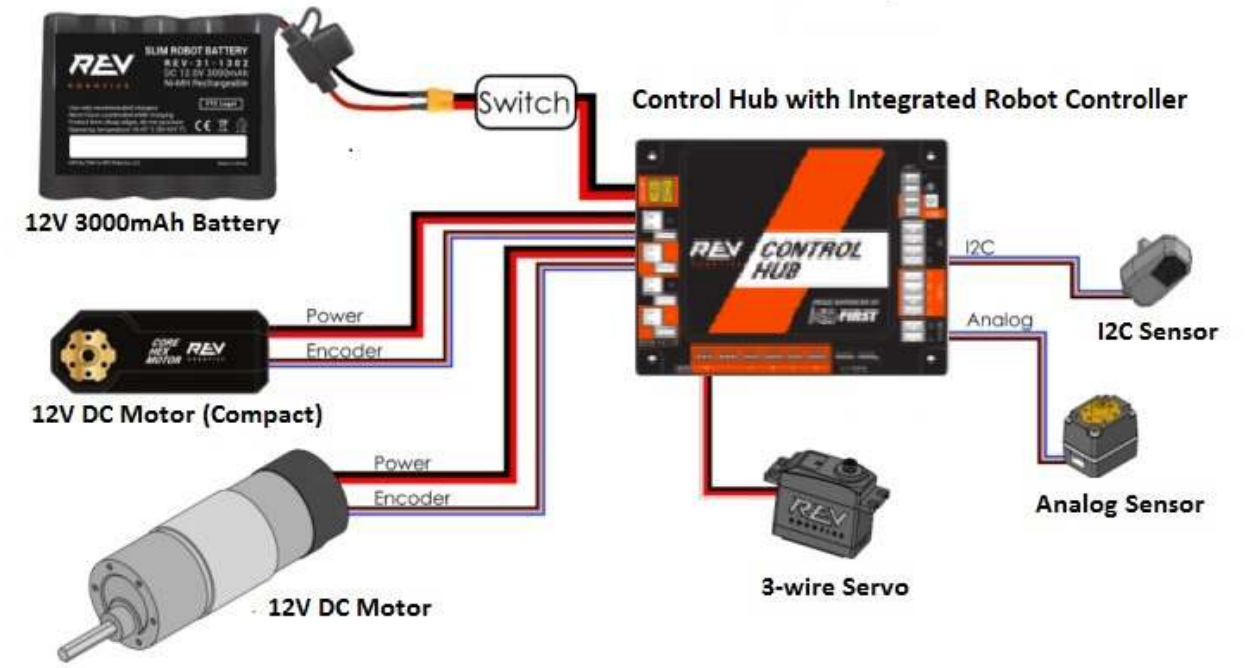
9.3.1 REV Control Hub + REV Expansion Hub

Warning: It is of vital importance to update the firmware on REV Expansion hubs to at least version 1.8.2. It includes better protection against disconnects and improves program performance. See the REV Expansion Hub firmware update docs[99].

This is the standard control system for teams starting out in FTC.

The Control Hub connects to the Expansion Hub through either a RS-485 connection, or a USB-A (Control Hub side) to mini-USB (Expansion Hub-side) connection. In either case, proper strain relief and cable management should be used.

For more information on setting up the Control Hub and configuring the robot, head to REV Robotics' Technical Resources Control Hub Guide[100].



[99] https://docs.revrobotics.com/duo-control/managing-the-control-system/updating-firmware
[100] https://docs.revrobotics.com/duo-control/control-hub-gs

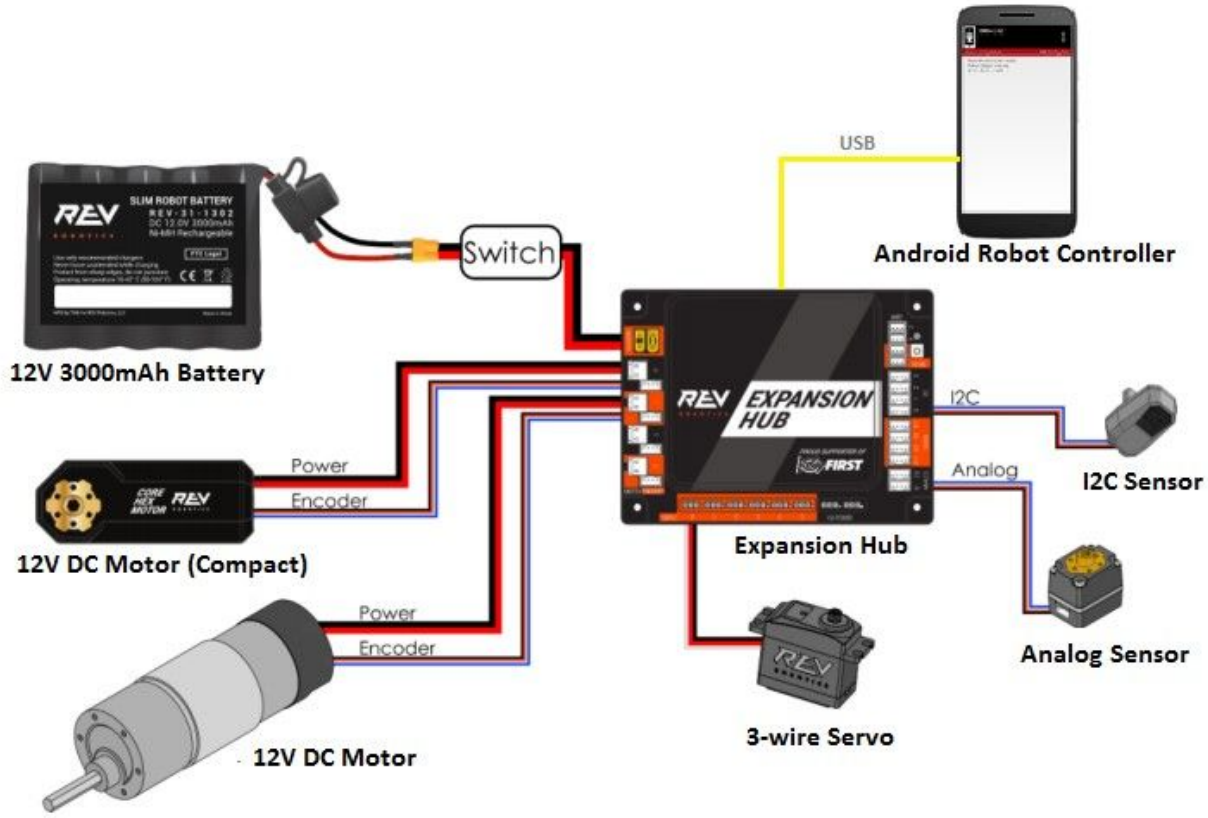
9.3.2 RC Phone + REV Expansion Hub(s)

Warning: It is of vital importance to update the firmware on REV Expansion hubs to at least version 1.8.2. It includes better protection against disconnects and improves program performance. See the REV Expansion Hub firmware update docs[101].

The Expansion Hub connects to the *Robot Controller* phone through its mini USB port. The REV Expansion Hub is reliable, as long as proper strain relief and wiring is carried out. This includes the *USB Retention Mount*, as well as 3D printing *XT30* stress relief mounts.

For more information on setting up the Expansion Hub and configuring the robot, head to REV Robotics' Technical Resources Expansion Hub Guide[102].

- USB Retention Mount[103]
- XT30 Stress Relief[104]



[101] https://docs.revrobotics.com/duo-control/managing-the-control-system/updating-firmware
[102] https://docs.revrobotics.com/duo-control/legacy/expansion-hub-gs
[103] https://www.revrobotics.com/rev-41-1214/
[104] https://www.thingiverse.com/thing:2887045

9.4 Driver Station Guide

The Driver Station (DS) is the way drivers interact with the robot. Through it, you configure settings, configure hardware, start and stop OpModes, use gamepads, and receive telemetry data from the robot. There are two main different choices for a Driver Station: the REV Robotics Driver Hub and an Android phone.

9.4.1 REV Driver Hub

The REV Robotics Driver Hub is effectively an Android phone with built in USB ports. It is purpose built for FTC®, and has several advantages over traditional Android phones.

Advantages

- Full sized USB ports do not require flimsy On The Go (micro-USB connector to USB-A port) cables
- Charges from USB-C including fast charging Power Delivery modes
- Larger screen than many Android phones
- Case is more robust then a standard Android phone

Disadvantages

- Price: A Driver Hub costs $250.00, whereas legal Android phones can be purchased secondhand for less
- Power Issues: Teams have reported random losses of power coming from an improperly sized battery.
- Battery Issues: Teams have reported high battery drain in sleep mode, leading to hubs dying faster then expected
- Wifi Issues: The wifi driver will occasionally crash when the device goes to sleep, requiring a device restart

---

Important: If you purchase a Driver Hub and experience issues, refer to the REV Driver Hub troubleshooting page[105]. The unexpected power off fix may be done preemptively to prevent a shutdown during a match.

9.4.2 Android Phones

As of 2023-2024, Android phones are still legal for competition use.

[105] https://docs.revrobotics.com/duo-control/troubleshooting-the-control-system/driver-hub-troubleshooting

Advantages

- Android phones can be very cheap when purchased secondhand
- Android phones generally are reliable out of the box

Disadvantages

- The lack of a full size USB A port means that flimsy On The Go cables must be used
- The phones themselves should be protected with a case to prevent damage

A 5 GHz phone is preferred over the standard 2.4 GHz phone to reduce ping issues. Have you ever noticed that your team's robot seems to lag at competitions only? This is because most school WiFi bands run on 2.4 GHz, which means that 2.4 GHz channels will be more crowded on competition day. This can lead to noticeably higher ping/lag which adversely affects driver and robot performance.

USB and *OTG* connections can be a possible source of disconnects during a match. This is generally caused by poor quality cables/adapters between the driver station and the gamepads. **It generally is worth it to purchase higher quality cables and adapters as opposed to the cheapest options.**

Recommended USB to OTG Cable[106]

Term

Micro USB On The Go (OTG) Cable The Micro USB OTG cable connects the *Driver Station* phone with the Logitech controller that the driver uses in order to control the robot.

It is recommended that teams purchase a couple spares due to faulty OTG cable connections and their low price.

---

[106] https://www.amazon.com/gp/product/B00YOX4JU6?pf_rd_r=PY8B4WPEQRQ80XYJCMSH&pf_rd_p=edaba0ee-c2fe-4124-9f5d-b31d6b1bfbee/

9.4.3 Controllers

LEGAL CONTROLLERS: As of the 2021-2022 season, the following controllers are legal for competition:[107]

- Xbox 360 Controller for Windows
- Logitech F310 Controller
- ETPark Wired Controller for PS4
- Sony DualShock 4 Wireless Controller for PS4 (Must be connected wired over USB)

In the past, the Xbox 360 controller was preferred by many top teams over the Logitech F310. This is mostly as the Xbox controller has less of a dead spot. A dead spot is basically the area where the joystick can move but not communicate that the stick has shifted slightly. This means that when the joystick is moved to a position where the controller can detect it, the robot will sometimes have a tendency to lurch forward. Dead spots make it difficult for the driver to execute fine-tuned and precise movements. **This is likely a non-issue for the large majority of teams. Also, keep in mind that there are very specific models of the Xbox controller that are legal, so they can be hard to find new.**

Starting with the 2020-2021 season, the ETPark and Sony PS4 Controllers are legal and provide numerous advantages over Xbox 360 Controllers. Some of these advantages include, better trigger and bumper control, a more usable dpad, and trackpad support as of the 2021-2022 season. Use of the PS4 controller is slightly different because the locations of the Left Stick and DPAD are swapped and some button names are different.[108] **Pro Tip: Make sure your drivers have experience with the controller they plan to use for competition.**

Generally it is the opinion that the PS4 controllers are better than the other options. Among the options for PS4 controllers, the choice is really up to you. The Sony controller has slightly better buttons, but the ETPark controller is half the cost and a more solid connecting wire. **Keep in mind that the drivers can use different controllers, so pick the one most comfortable to you.**

A phone holder and *OTG* strain relief connector can be useful as it may help to prevent connections from loosening. Especially if teams are running dual controller configuration, ensuring that the USB hub is secure won't hurt.

9.5 Motor Guide

12 volt DC motors are main drivers of FTC® robots. They are used for powering the drivetrain, intakes, lifts, and other mechanisms.

9.5.1 Choosing a Motor

Legal Motors

12 V motors are strictly controlled by FTC® rules. As of 2020-2021 season, the only FTC legal motors are the following ones:

- TETRIX 12V DC Motor (regular[109], Torquenado[110])

[107] https://www.firstinspires.org/sites/default/files/uploads/resource_library/ftc/legal-illegal-parts-list.pdf
[108] https://github.com/OpenFTC/OpenRC-Turbo/blob/2d1e527d3d53c3ac7da701a73d342b85cf407835/RobotCore/src/main/java/com/qualcomm/robotcore/hardware/Gamepad.java#L884
[109] https://www.pitsco.com/TETRIX-DC-Gear-Motor
[110] https://www.pitsco.com/TETRIX-MAX-TorqueNADO-Motor-with-Encoder/

- AndyMark NeveRest series 12V DC Motors[111]
- Modern Robotics/MATRIX 12V DC Motors (this also includes goBILDA motors[112], which are MATRIX motors with a different gearbox)
- REV Robotics HD Hex 12V DC Motor[113]
- REV Robotics Core Hex 12V DC Motor[114]

With the exception of *REV Core Hex Motor*, which is discussed separately, all other motors above have very similar structures. They consist of the following components.

- **Bare motor**. In all cases above, this is a 12V motor of *RS-555 type*, with free speed around 6,000 RPM and stall current around 10A. The motor specs posted by different vendors might be slightly different, but the difference is mainly due to different testing methods. In real life, the bare motors used by *AndyMark NeveRest motors*, *REV Robotics HD Hex motors*, and *goBILDA motors* are virtually identical. The most reliable specs can be found in the *Peak Power and Motor Curves* (page 232) section.
- **Gearbox**. The gearbox is attached to the front of the motor and reduces the speed while increasing the torque. For example, a 20:1 gearbox reduces the speed by a factor of 20, resulting in a no-load speed of around 300RPM. See *Gear Reduction*. A gearbox also contains the output *shaft* (typically 6mm D profile; REV motors use 5mm hex shafts) and mounting holes. The gearbox can also be replaced; this is FTC legal but requires some skill.
- **Encoder**. Attached to the back of the motor and protected by a plastic cap, the *encoder* contains a sensor registering motor *shaft* rotation. It can be connected to REV hubs and used for precise control of motor speed or rotating to a specific position.

Since the bare motor is similar for all motors discussed above, the choice of the right motor is dictated by the gearbox: the *gear ratio*, output *shaft* type, and ease of mounting.

Gearboxes

There are two kinds of gearboxes: spur and planetary (also known as orbital). Their inner structure and difference is discussed in detail in *Gearbox Internals* (page 227) section. For new teams, it suffices to know that planetary gearboxes are slightly more expensive, but more reliable. Spur gearboxes can strip under shock loads (for example, when your robot hits a wall), requiring you to replace the gearbox. For this reason, it is advised to use planetary gearbox in high-load use cases such as drivetrains.

Available Spur Gearboxes and Motors

> **Danger:** Spur gearboxes are NOT recommended due to their shorter lifespan and lower mechanical resilience compared to planetary gearboxes. If you are purchasing new motors, it is highly suggested to purchase planetary gearbox motors instead. Care should be taken to not put load on the output shaft of a spur gearbox. In particular, spur gear motors should NOT be used in high load applications, such as a drivetrain

Motors with spur gearboxes include *AndyMark NeveRest Classic motors* (in 40:1 and 60:1 ratios), the *REV HD Hex 40:1 Spur motor*, and goBILDA 5201 Series Yellow Jacket Spur Gear Motors. All of them offer similar

[111] https://www.andymark.com/categories/mechanical-gearboxes-gearmotors
[112] https://www.gobilda.com/motors/
[113] https://www.revrobotics.com/rev-41-1301/
[114] https://www.revrobotics.com/rev-41-1300/

performance and reliability, so your choice is primarily dictated by the convenience of mounting and connecting to the rest of your design (e.g., if you use REV kit, you should probably choose *REV HD Hex motor*, as it uses the same *5mm hex shaft* as the rest of REV system).

- goBILDA's 5201 series spur gearboxes are much cheaper than either REV's or Andymark's; whether that's a good or bad thing is up to you. They utilize the rather uncommon (in the FTC world) bullet connection for power. However, these are now discontinued. The output shaft is a 6mm D-shaft.
- *REV HD Hex Planetary* 40:1 motor - This motor comes only in a 40:1 ratio, but does use the same connections (JST VH) as the REV Expansion and Control Hub for power which means no adapter cables. The output *shaft* is a 5mm hex *shaft*. *REV UltraHex* has a 5mm hex *bore* running through the middle of a 1/2" hex *shaft*, which makes adapting this motor to any length of *UltraHex*, and by extension, 1/2" hex *shaft*, very easy.
- *Andymark NeveRest* Classic motors come in a few different ratios, which are 40:1 and 60:1. The output shaft is a *6mm D-shaft*, and like all NeveRest motors use the *Anderson PowerPole* to connect to power. This connector is perhaps the most robust of the ones listed here.

Planetary Gearboxes

Standard planetary gearboxes include *Andymark NeveRest Orbital motors*, *REV 20:1 Planetary motor*, and *goBILDA's 5202/5203/5204 Series Yellow Jacket Motors*.

Any of these "standard" gearboxes are more robust than spur gearboxes. Like the spur gearboxes, the gearboxes from different vendors, while not interchangeable, are very comparable in terms of robustness. Once again, the main thing to consider here is your desired reduction, your desired motor connections, and your desired output *shaft* type.

- goBILDA Yellow Jacket[115] has the most varied selection of gearbox ratios with too many to list here, but utilize the rather uncommon bullet connection for power. The output *shaft* is a *6mm D-shaft* or 8 REX (7 mm hex rounded to 8 mm).
- REV HD Hex Planetary motor - This motor comes only in a 20:1 ratio, but uses the same connections (*JST-VH*) as the REV Expansion and Control hub for power which means no adapter cables. The output shaft is a *5mm hex shaft*. REV *UltraHex* has a 5mm hex *Bore* running through the middle of a *1/2" hex shaft*, which makes adapting this motor to any length of *UltraHex*, and by extension, *1/2" hex shaft* very easy. The ratio of the *HD Hex Motor* is 20:1.
- Andymark NeveRest Orbital motors come in two *ratios*, 3.7:1 and 20:1. The output shaft is a *6mm D-shaft*, and like all *NeveRest motors* use the *Anderson PowerPole* to connect to power. This connector is perhaps the most robust of the ones listed here.
- REV UltraPlanetary[116] gearbox - The UltraPlanetary is a customizable planetary gearbox that is designed for FTC. The three gearbox options are 3:1, 4:1, and 5:1, and can be mix & matched to create a custom ratio. It is possible to use from one to three gearboxes for a minimum ratio of 3:1 and a maximum of 125:1.

Note: While REV lists the UltraPlanetary stages as 3:1, 4:1, and 5:1, their actual gear ratios are slightly different. Consult the REV UltraPlanetary User's Manual for the exact gear ratios[117].

The UltraPlanetary was intended to give teams maximum customization without the typical limiting factor - high cost. The total cost for the three stage gearbox and motor is *exceptionally* good value

[115] https://www.gobilda.com/yellow-jacket-planetary-gear-motors/
[116] https://www.revrobotics.com/rev-41-1600/
[117] https://docs.revrobotics.com/ultraplanetary/ultraplanetary-gearbox/cartridge-details

for a customizable motor. In addition, the UltraPlanetary has a female 5mm hex output *shaft* which allows teams to customize the shaft length.

Choosing The Right Gearbox

For regular use, any of the planetary gearboxes will fit your needs. Planetary gearboxes are just a tiny bit more expensive, but boast better backlash and efficiency, higher load capacity, and better capacity for shock loads than spur gearboxes. The tradeoffs, cost and mechanical noise, are almost never a factor.

Tip: Because both gearbox types are so similar in price for similar ratios, we generally recommend the use of a planetary over a spur gearbox.

If you already own spur gearboxes, try to use them in lower-load situations and use planetary motors on your drivetrain.

For small reductions, it may be more economical to choose a motor you already own and build an external reduction using gears, chain, or belts. It should again come down to your desired output shaft, desired gear ratio, and for the UltraPlanetary, whether you want the ability to swap parts out on the fly.

9.5.2 Gearbox Internals

Fundamentally, a gearbox is just a collection of gears and an enclosure that connects them. Gearboxes have an **output ratio**, the final Gear Reduction between the motor input and the final output *shaft*.

Term

Gear Reduction Also known as a gear ratio. In any rotational power transmission system (typically involving motors and *servos* in FTC®), a gear ratio defines both the number of rotations of the system's input and the number of rotations of the output.

For instance, a NeveRest 20 gearmotor consists of an unmodified *NeveRest Motor* and a planetary gearbox that has a gear ratio of 20:1 (or, when spoken, "20 to 1"). This means that in order for the output shaft of the gearbox to rotate 1 time, the input shaft of the motor must rotate 20 times. Gear ratios are one of the most important design considerations about a power transmission component.

Any FTC motor or servo has two properties: speed and torque (or rotational force). These two properties are inversely proportional, meaning that increasing speed decreases torque, and vice versa. For instance, if one wishes to make a mechanism faster at the expense of torque by doubling the speed of that 20:1 gearbox, they would decrease the gear ratio by a factor of 2. Since 20 divided by 2 is 10, the new desired ratio would be 10:1 (this is referred to as gearing up). However, if one wishes to double torque instead, making the system more powerful and robust at the expense of speed, they would increase the gear ratio by a factor of 2, leaving them with a 40:1 ratio (this is referred to as gearing down).

The most common ways of gearing up or down are using gearboxes, gears, sprockets and belt-driven pulleys, all of which exist in various sizes.

In FTC, gearboxes may be more common than you think - every motor has a gearbox attached to it. These gearboxes are one of the following two types: spur or planetary. Below we give a detailed analysis of each of these gearbox types. **Just for clarification, the gearboxes below are separate from the base motor.**

Spur Gearboxes

Term

Spur gearbox A spur gearbox has spur gears which are stacked on top of each other. Gear reduction is achieved through different size gears on the same plane.

Spur gearboxes are an arrangement of *gear ratios*, often stacked to achieve a large compound ratio (e.g. 40:1). Each individual ratio only has two *gears*- one may be 8:1, another may be 5:1, but the final ratio will be 40:1. These gearboxes are used in the Andymark NeveRest Classic series and goBILDA's 5201 series motors, as well as *REV HD Hex Motors*. Due to the nature of how these gearboxes are built, each reduction only has a few teeth from each *gear* engaged, and those teeth carry the entire load of the gearbox. It's easy to damage a spur gearbox from shock load, and if one *gear* breaks, the entire gearbox will stop functioning.

Tip: Using spur gearboxes on high-load applications such as drivetrains or arms is not recommended. Instead, use planetary gearboxes.

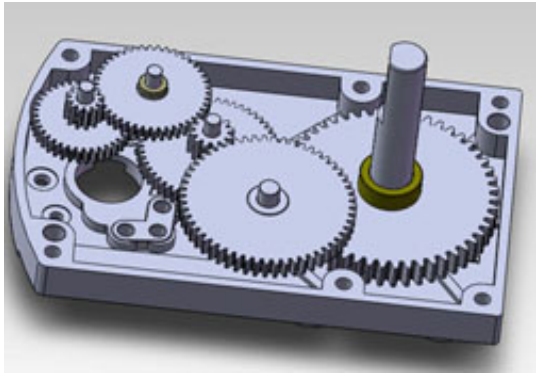


Fig. 6: Example of a spur gearbox. Note how all gears mesh with only one other gear.

Advantages of Spur Gearboxes

Generally, spur gearboxes are cheaper than planetary gearboxes. However, in FTC that price change is often minimal. A 20:1 planetary gearbox from REV is only $4 more than a spur 20:1

Spur gearboxes from different vendors are not interchangeable. However, they are comparable and practically indistinguishable in performance. The main thing to consider here is your **desired reduction, your desired motor connections, and your desired output shaft type**.

Planetary Gearboxes

Planetary gearboxes use a more complex system of gears to achieve a robust reduction in a compact space. In automotive engineering, planetary gear sets can achieve a few different ratios without changing gear size, but all planetary gearboxes that you will see in FTC only achieve one gear ratio.

Term

Planetary Gear Planetary gearing consists of a center gear (sun gear) which has smaller gears (planet gears) revolving around it. The outer radius has a ring gear which holds the other gears in place.

Planetary gearboxes are used in the Andymark Orbital series, some REV HD Hex Planetary and UltraPlanetary Motors and goBILDA's wide selection of *planetary gear motors*. Additionally, AndyMark sells a few aftermarket planetary gearboxes called NeveRest Sport and 57 Sport. As you can see from the graphic below, there are more teeth meshing per stage than in the spur gearbox.

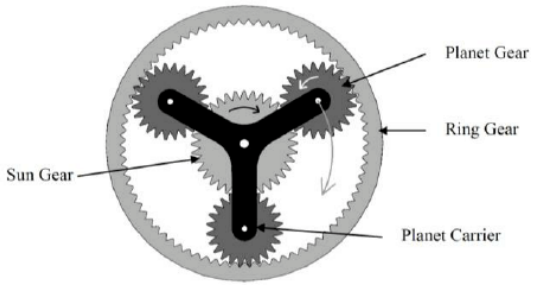


Fig. 7: Example of a planetary gearbox stage. Note how the sun gear meshes with more than one gear.

Advantages of Planetary Gearboxes

- Backlash is lower than spur gearbox equivalents. Backlash is defined as the clearance or lost motion caused by gaps between parts. This can easily be explained through putting a wheel or gear on a motor shaft and lightly rotating it. The part should be able to wiggle around a little without having considerable force imparted on it. This is caused because it is impossible for the gear teeth inside the gearbox to mesh perfectly, and is the same for *chain* and *sprockets*, or any other form of power transmission. However, planetary gearboxes have less backlash as they have less stages of gears.
- Efficiency is better than spur gearboxes. A typical two-stage spur gearbox is about 85% efficient, whereas most two stage planetary gearboxes are 94% efficient.
- Load capacity is higher for planetary gearboxes. This is due to having multiple teeth engaged per stage, which spreads the load.

Tip: This means planetary gearboxes will not break as easily when used in high-load applications such as drivetrains.

9.5.3 Motor Wiring And Mounting

Power Connections

Depending on the vendor, the motors can come with one of the following connectors:

- *JST-VH* connector (REV motors)
- *Anderson PowerPole* (AndyMark's NeveRest)
- 3.5mm Bullet connector (goBILDA)

JST-VH is probably easiest to use, as it is the same connector used by REV Expansion Hub. The *JST-VH* connection has a locking mechanism for peace of mind. *PowerPoles* are very reliable and sturdy but somewhat bulky. Bullet connectors are very compact, making it easy to route the cable through openings, but can disconnect if someone pulls on the wire, so you need to be careful with them.

Since REV Hubs use *JST-VH* connectors, to connect a motor with *Anderson PowerPole* or bullet connectors you need adapters which you can purchase from REV Robotics and goBILDA.

You can extend or shorten motor power cables by soldering additional length of cable. This is legal as long as you use 18 gauge or thicker cables. Teams can also purchase extender cables to chain multiple pieces of cable together.

For more tips on wiring the robot, see *Wiring Guide* (page 211).

Encoders

Note: Encoder cables are very fragile. Take care to protect them from snagging and sharp impacts! It may be prudent to inspect encoder wires once in a while.

If using encoders, you need to connect them to the REV hub by a 4-wire cable. REV Hub uses 4-pin *JST-PH* connector for encoder ports. REV motors also use *JST-PH* encoder ports, so you can connect them to the hub by *JST-PH* 4-wire cable, available from REV Robotics.

goBILDA motors use JST-XH 4-pin encoder port (**note the difference: XH vs PH**), so to connect them, you need a JST-PH to JST-XH cable, available from AndyMark or goBILDA.

AndyMark also use JST-XH encoder port; however, an additional problem is that encoders of these motors require 5v power, whereas the encoder port of REV hub only provides 3.3v. Thus, it is recommended that you connect them using level shifters, available from REV Robotics. For details please check the REV's documentation[118].

Mounting Motors

There are two ways to mount a motor: using a *clamping mount* (such as 32mm goBILDA clamping mount[119]) or *face mounting*, using threaded holes in the front face of the gearbox.

[118] https://docs.revrobotics.com/duo-control/sensors/5v-sensors#connecting-5v-encoder
[119] https://www.gobilda.com/1400-series-1-side-2-post-clamping-mount-32mm-bore/

Clamping Mount

- Easiest way to mount a motor, as only one screw is required.
- Not as secure as face mounting, as clamp friction is looser than face mounting using screws.
- Some gearboxes (particularly spur gear) do not place the output shaft in the center of the gearbox.

  Thus, motors with offset shafts are particularly sensitive to clamp mounts, as any rotation of the motor will alter the shaft position. This may have the consequence of losing *chain* or *belt* tension.
- It is possible to double clamp a motor - one in front, and one in the back.
- To increase friction and reduce the chance of loosening, one can wrap electrical tape around the area of the motor that will be clamped down. Use a couple wraps of tape.

Face Mount

- Slightly more tedious and uses more screws.
- Repairing a broken gearbox or swapping a motor is slower than if using clamp mounting.
- Face mounting is much more reliable than clamp mounting, as the screws hold the motor in place very tightly.
- Teams can use **BLUE** *Loctite* on high-vibration motors to ensure the motor does not jar loose.

Tip: *Face mounting* is recommended for high-load and/or high vibration use cases such as drivetrain. This is as *clamp mounted* motors can shift and come loose easier than *face mounted* motors. It is also recommended to use **BLUE** *Loctite* when *face mounting*, if possible.

Note that the pattern of *face mounting* holes is different for different vendors. For example, goBILDA uses 4 M4 holes in a square with side 16 mm, whereas AndyMark classic motors use 6 M3 holes on a 31 mm diameter circle. Thus, face mounting NeveRest motors to goBILDA parts requires use of special adapters, and vice versa. Similarly, gearbox diameter also varies between motors, so when choosing a *clamping hub*, make sure to use the right diameter.

9.5.4 Motor Power

Just like any electrical device, motors draw current and consume a certain amount of power to function. Motors convert some portion of the input power into spinning the shaft; this portion is the output power. Thinking of motors in terms of energy instead of speed and torque can make it easier to calculate how it should be used and provide tools for choosing the optimal gear ratio for an application.

What is Power?

Fundamentally, power is defined as the amount of energy transferred in a given time period. Practically, this means power is how much "energy" you can get out of the motor. **The output power of a motor varies depending on how much load is on it, but does not change as the gear ratio of the motor changes**. What this means is that a free-spinning 1:1 motor will output the same amount of power as a free spinning 100:1 motor, but the actual speed and torque will be different. Output power is proportional to speed times torque, so as speed and torque are changed with a gearbox, the output power must remain the same.

Another piece of information, although not always as useful, is the *input* power of the motor. The input power of the motor is how many watts of power the motor consumes, which is equal to the voltage sent to the motor times the current consumed by the motor. **Motors are not 100% efficient, so to get a certain output power you often need double or even triple the input power**. For example, a standard FTC® motor can consume up to 65 watts just to produce 29 watts of output power.

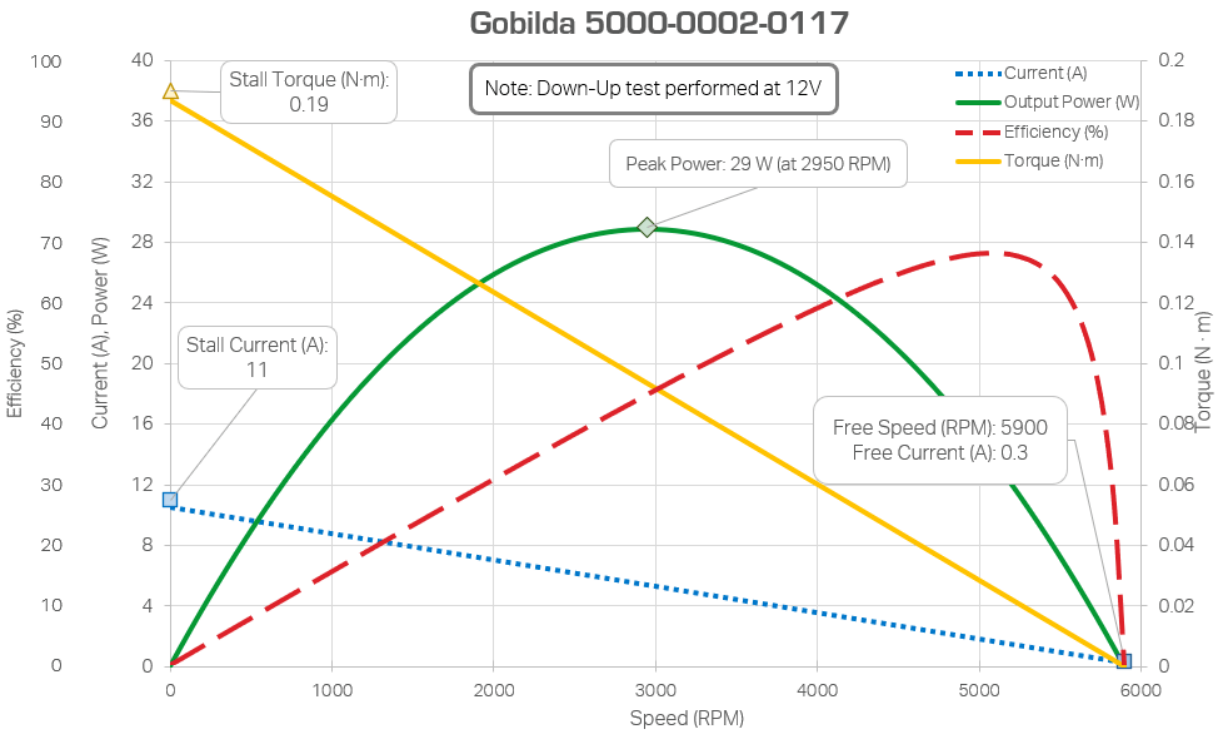
Peak Power and Motor Curves

In order to figure out how much power your motor will be consuming or outputting, its helpful to reference a motor curve. These are data sheets that plot the motor's output speed, output torque, output power, and efficiency all on one graph. Every FTC legal motor except the Core Hex has been dyno tested, and their data are below. It is safe to assume other non-current limited motors (such as servos) follow similarly-shaped curves, although with different speed, torque, and power outputs.

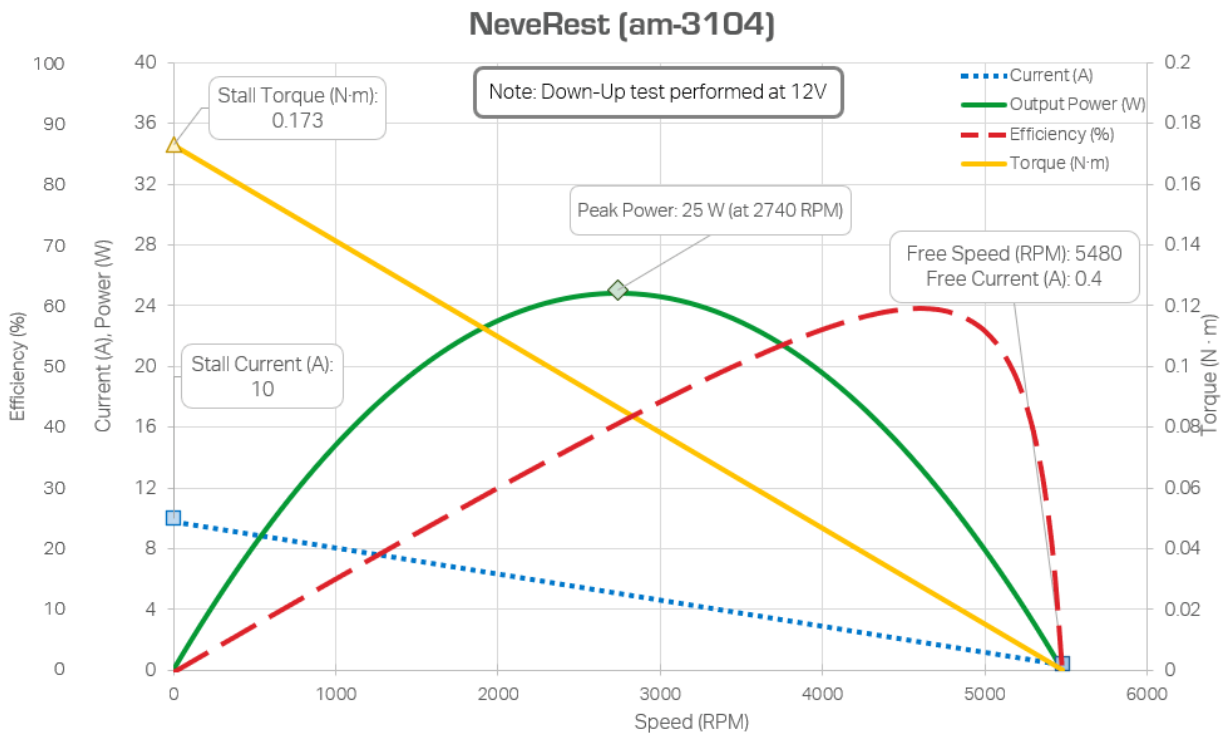
	Free Speed (RPM)	Free Current (A)	Maximum Power (W)	Stall Torque (N*m)	Stall Current (A)
goBILDA (MATRIX)	5900	0.3	29	0.19	11
NeveRest	5500	0.4	26	0.17	9.8
REV Core Hex[120]	125	0.2	10	3.2	4.4
REV HD Hex	6000	0.3	28	0.18	11
TorqueNado	5900	0.2	26	0.17	9.8

[120] Data taken directly from the product page[121], no motor curve available.
[121] https://www.revrobotics.com/rev-41-1300/

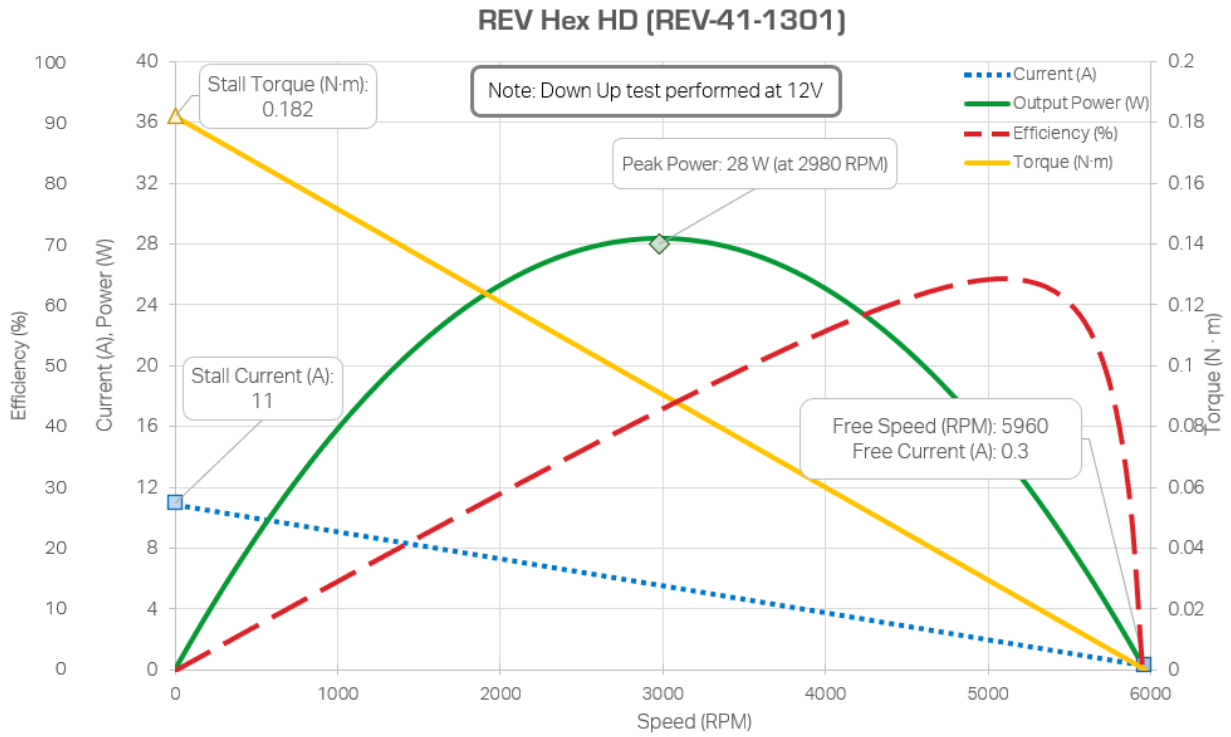
goBILDA (MATRIX)



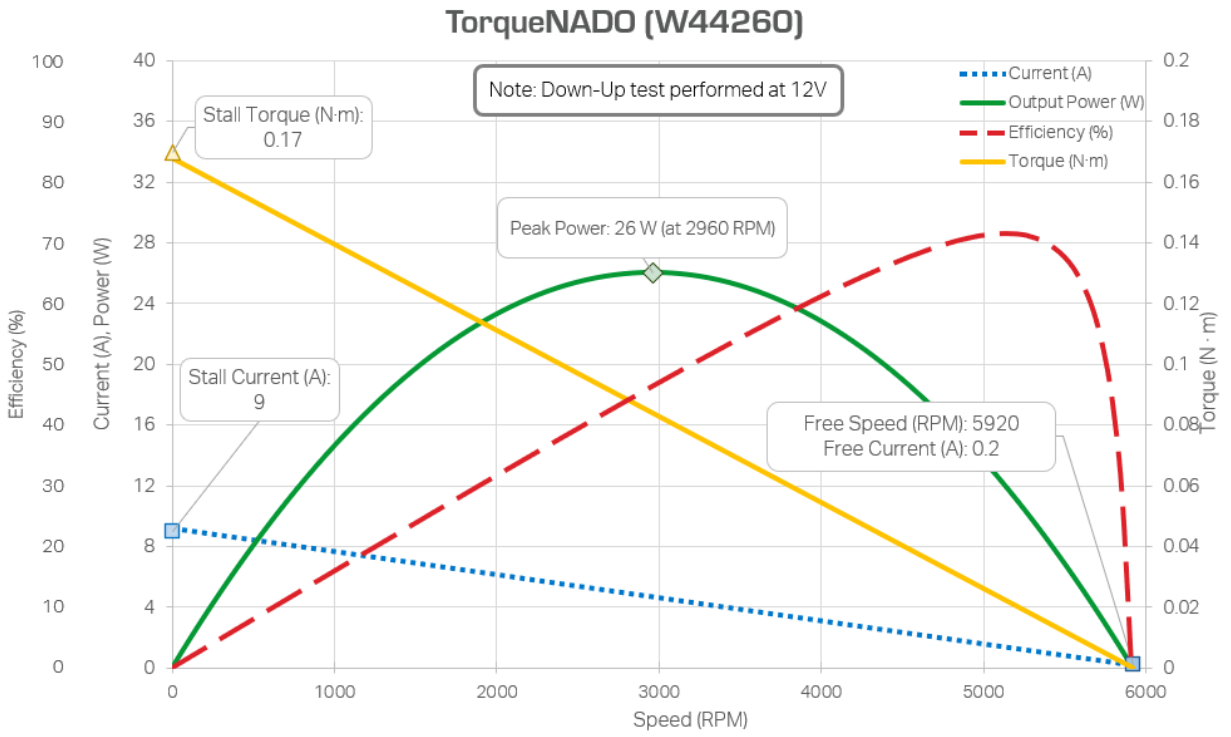
NeveRest


NeveRest (am-3104)
Note: Down-Up test performed at 12V
Stall Torque (N·m): 0.173
Peak Power: 25 W (at 2740 RPM)
Free Speed (RPM): 5480
Free Current (A): 0.4
Stall Current (A): 10
Current (A)
Output Power (W)
Efficiency (%)
Torque (N·m)
Efficiency (%)
Current (A), Power (W)
Torque (N · m)
Speed (RPM)
0
1000
2000
3000
4000
5000
6000

REV HD Hex



TorqueNado



A motor curve represents a motor at 12 V (equivalent to setting the motor power to 1 in software) with various amounts of load applied to the axle. As you can see, the power output from the motor is not constant, instead rising until about 50% load, before falling again. This point at 50% load is called the **peak power output** of the motor, and is at a similar point (50% load) across all FTC legal motors.

The varying power of a motor means that speed and torque output do not change linearly when more load is applied onto the axle. Counterintuitively, placing 50% stall load on a motor doesn't halve its speed, but will rather reduce it to slightly above 50% speed. Similarly, placing more then 50% load on a motor will cause the speed to fall faster then linearly.

In addition, you can see that efficiency rises as speed goes up. This means, if current draw is a concern, one should always be running their motors with loads below 50% of their stall torque. These two properties of a motor, the peak power output being 50% of the stall torque and the efficiency of a motor being higher the lower the load is, guide the selection of the gear ratio of a motor. **Ideally, gear ratios should be chosen such that the stall torque is twice the average torque load on the motor, and should skew towards providing more torque than needed rather then less**.

Note on Current Consumption

You may see while looking at motor curves that the stall current of FTC motors can be as high as 11 amps per motor. FTC batteries can only provide 20 A of current output before blowing the fuse. However, even if the 20 A limit isn't reached, drawing too much current can cause other motors to feel sluggish or unresponsive. Care should be taken that more then two motors are never stalled at the same time.

Note: You may ignore this exception when dealing with mecanum drivetrains, as they will generally slip before the motors actually reach their stall current. However, placing very low gear ratios or more than 4 motors on traction drivetrains can exceed the current limit of an FTC Battery.

Motors can pull "transient current" where they pull a large amount of current for extremely brief amounts of time. This often happens when the motor starts moving or when a momentary load is placed on it. While transients generally cannot cause a fuse to pop, they can cause other issues, such as sluggish control if pulled by a motor, or low voltage if the transient is pulled by a servo (goBILDA Super Speed servos have been observed to do this occasionally).

9.5.5 Motor Glossary

Core Hex Motor The Core Hex Motor, sold by REV, is different from the standard *RS-555 series motors* that are generally used by FTC® teams. It features a 90 degree orientation and does not contain an output shaft. Thus, teams will have to cut 5 mm hex shaft to length as needed. The Core Hex motor has a slow gear ratio (72:1), and is not as powerful as the RS-555 series motor.

> **Warning:** We advise teams to go against the Basic Bot Guide provided by *FIRST*®, as Core Hex Motors should NOT be used to power drivetrains.

HD Hex Motor The HD Hex motor, sold by REV Robotics, is a *RS-555 series motor* with spur gear and planetary gearbox options. The motor has a 5mm hex output shaft compatible with REV's motion system.

NeveRest Motor The NeveRest Motor, sold by AndyMark, is a *RS-555 series motor* that is available in spur gear and planetary options. It has a 6mm D-shaft output compatible with Actobotics motion system.

Yellow Jacket Motor Yellow Jacket motors are the *RS-555 series motor* and *planetary gearbox* sold by goBILDA. It has a *6mm D-shaft* and is available in many different *gear ratios* from 3.7:1 up to 188:1.

UltraHex UltraHex is 1/2" aluminum hex *shafting* sold by REV Robotics. There is an inner 5 mm hex *bore* in the middle, which allows compatibility with REV's 5 mm hex shaft motion system. The 5mm hex bore also allows for a 1/4-20 or M6 screw to be tapped into it. 1/2" hex is also compatible with many FRC® vendors.

RS-550 Series Motor The RS-555 series motor is the standard motor in FTC. It forms the base for the *Andymark NeveRest*, *REV HD Hex*, and *goBILDA Yellow Jacket* motors.

9.6 Servo Guide

Term

Servo a small DC motor attached to servo gears that is very finely controllable and interfaces via a 3 wire PWM connector. Servos are used in FTC® for high-precision applications that are low-load - for example, opening a trapdoor for balls to fall through. Typically servos have limited range of rotation (180° is common). The output has splines, which are the rigid teeth that are on top of the servo.

Servos are commonly used in RC cars (steering) and RC planes (moving ailerons). In FTC, servos are typically used for claws, grabbers, and arms.

Note: Servos are **NOT** replacements for DC motors, and should not be used as such. Servos are made for fine-tuned and accurate movement, not high-load or fast rotation applications.

There are many servos from different manufacturers, which vary widely in price, performance and value. Fortunately, virtually all servos use the standard 3-wire connector, and accept the same kind of controlling signal (PWM signal at 50 hz). Each REV Expansion Hub provides 6 servo ports, so you can plug in a servo from any manufacturer. Also, there is a standard size for servos for FTC use, so mounts can be interchanged between manufacturers.

Fig. 8: A common servo, HS488-HB from Hitec

Commonly used servos used in FTC are the REV Smart Servo and goBILDA Dual mode Servos (25-2[122]) and (25-3[123]), but you should check out other servos as well. Picking the right servo for your application is a question that's almost impossible to give a blanket answer for. To learn more, please read the *Choosing a Servo* (page 239) section.

The most prevalent problem with servos is durability. Internal servo gears in cheaper servos strip easily when subjected to shock loads. Servos are also poor at handling lateral loads or bending of the shaft. To avoid having to frequently replace servos, choose ones with metal gears and use *Servoblocks* or your kit's equivalent to prolong longevity.

9.6.1 Choosing a Servo

Choosing a *servo* can seem daunting at first, given the number of options to consider. This guide is intended to provide a starting point to compare different servo options, and also has some hand picked recommendations at the end.

Important: It is very important to keep the reliability of a given vendor in mind when choosing servos. It is not uncommon for manufacturers and resellers on Amazon and other similar sites to exaggerate their servos specs, or pick unrealistic best case scenarios for measuring the specs. We have limited our recommendation only to vendors who historically have been reliable with publishing servo specifications.

As a rule of thumb established manufacturers (HiTech, Savox, ServoCity, Gobilda, Andymark, etc) will usually publish accurate numbers, and servos from marketplaces (Amazon, AliExpress, etc) should be viewed with some skepticism.

---

[122] https://www.gobilda.com/2000-series-dual-mode-servo-25-2/
[123] https://www.gobilda.com/2000-series-dual-mode-servo-25-3-speed/

Servo Type: Regular or Continuous

Servos that can rotate to a given position based on PWM input signal are called **regular servos**. In addition, there are also **continuous rotation servos**, which are effectively just small motors in a *servo* form factor. They have no position control; instead, PWM signal is used to control their rotation speed.

Many servos from FTC® vendors are Dual Mode, meaning they can switch between continuous and regular modes (often requiring the use of a sold-separately servo programmer). These servos can be used as either continuous or regular servos.

Servo Torque And Speed

Servo output power is measured in both **speed** and **torque**. Speed (normally in seconds per 60°) refers to how fast the *servo* turns 60 degrees in Standard Rotation mode.

Why seconds per 60 degrees?

Historically, the servos commonly used in FTC were created for RC (Radio Controlled) car setups. These cars often had steering linkages with a maximum side to side travel of 60 degrees so manufacturers would often advertise their servos using seconds for 60 degrees.

Torque (usually measured in oz-in or in kg-cm) refers to the amount of force the *servo* can apply to a lever. For reference, if you put a 1" bar on a servo, then put a force gauge on the end, the torque rating of the servo (in oz-in) will be measured.

As you may know, speed and torque will generally have an inverse relationship. You can find some insanely powerful servos that are pretty slow (slower than 0.20 s/60°) or some less powerful ones with faster ratios (anything faster than 0.12 s/60° is considered very fast).

Finding the right *servo* for your application can be tough, but a good way is trying to decide if you need more speed or torque, and if your *servo* will experience shock loads or not.

Durability and Servo Gear Material

The two things that threaten a *servo's* longevity are the internal motor burning out and more commonly, the *gears* stripping inside the *servo*. A motor burning out is pretty uncommon, but it can happen under large loads for a prolonged amount of time.

> **Caution:** You should **never** stall a servo against an immovable object.

Gear stripping is a very common problem which occurs when the torque needed to actuate a component exceeds that of the *servo's* maximum torque. There are two main cases when this can occur.

- Shock load from external force can strip the *gears* easily, regardless of which material the *gears* are made from. An example could be the component slamming into the field wall or another robot.
- Shock load from reversing directions on an object that is too heavy can strip the *gears*. Torque increases with mass and also distance from the center of rotation. If the component being actuated is far from the *servo*, the long lever arm means larger torque. Furthermore, if the component is moving, reversing direction also will require more torque. Thus, the principle is that components should be light and not reverse direction suddenly to prolong *servo* life.

Shock load resistance is impacted directly by the material the *gears* are made from. This ranges from plastic to titanium, so let's go down the list, starting from the weakest.

- **Plastic**: with low power *servos*, these are normally okay. Generally used for applications in model airplanes such as ailerons. FTC applications include light load mechanisms which will not have direct contact with any game elements or the field. An example could be a servo that opens a trapdoor or moves game elements within the robot.
- **Karbonite**: Hitec's *gear* plastic is a very durable and long lasting plastic and is very good under long use and low load. Be aware that it can strip easily under the shock loads found in FTC. Karbonite is more durable than plastic but still suffers from shock loads.
- **Brass**: Brass *gears* are stronger than plastic but also suffer greatly when faced with shock loads in FTC like intake wrists and deposit buckets. It's found on slightly higher end servos such as the REV Smart Servo.
- **Steel**: This is where we start getting big. Steel *gears* are very durable and you'll have a tough time stripping these. In general, expect to pay a premium. The goBILDA Dual mode servos (v2) is an example of steel *gear servo*.
- **Titanium**: Titanium is where you get into really high end, virtually unbreakable *servos*. Starting from $75, they can reach over $150. A common misconception is that titanium is stronger than steel, however its advantage is in strength to weight ratio (as in, titanium gearboxes will often be lighter than steel gearboxes).

Servo Size

Servos come in different sizes. Fortunately, almost all manufacturers use the same standard set of *servo* sizes, ranging from sub-micro to large. The two sizes commonly used in FTC are *standard size* (which includes REV Smart Servo and goBILDA Dual Mode Servos) and *large size* (also known as quarter-scale) *servos*. However, large *servos* are quite uncommon.

Note that while in general, the larger the size, the more powerful the *servo*, it is not a strict rule. You can buy very powerful standard size *servos* - just expect to pay more for them.

Servo Spline

The output shaft of the *servo* is commonly called the **spline**. Most servos have industry standard 25 tooth spline (also known as F3); in particular, this is the spline used by REV Smart Servo and goBILDA Dual Mode servos. However, Hitec servos using 24 tooth spline are also very popular.

Andymark servos are an exception to this, as they use a 5mm hex shaft as their output instead of a 24 or 25 tooth servo spline.

> **Attention:** Please check the spline type before you buy the *servo* - otherwise, your *servo* attachments will not fit.

For more info about servo splines, please check https://www.servocity.com/servo-spline-info/.

Servo Range

The angle over which a non-continuous *servo* can rotate while retaining position feedback is called the range. When choosing a servo, it is important to make sure you have enough range for the movement you need.

By default, the FTC Control Hub and FTC Expansion Hub output 600-2400 microsecond signals. However, this range can be expanded to 500-2500 microseconds. When choosing a servo, it is important to make sure that its range will be usable for your application inside of 500-2500 microseconds.

Note: The default 600-2400 range of the FTC Expansion Hub and FTC Control Hub can make it appear that popular servos like the goBILDA Dual Mode servos and REV Smart Robot Servo have less range than advertised. You can use the following code to expand the range to 500-2500 microseconds.

```
ServoImplEx servo = hardwareMap.get(ServoImplEx.class, "myservo");
...
servo.setPwmRange(new PwmRange(500, 2500));
```

Cost

Servos range from cheap $7 *servos* for light applications, all the way up to some Hitec or Savox *servos* for close to $200.

By far the best bang for your buck *servos* out there are going to be **goBILDA dual mode** and **REV SRS** servos. In addition, the **Andymark High Torque/Speed** servos on paper are the best bang for your buck servo, but at the time of writing have not been released and tested.

The biggest downside to the REV SRS are their brass *gears*. Coupled with high output power, this meant that stripping *gears* with any shock load was commonplace.

The next big name in FTC *Servos* is Hitec, who is a huge name in hobby *servos* for decades and are very well trusted. Their low end *servos* are inexpensive but easily broken.

A mid-priced Hitec *servo* is the HS 485-HB/488-HB servo, with Karbonite *gears*. While it shouldn't be used in high load applications, it is fine for general use such as claws or trapdoors. 485HB uses 24 tooth spline; 488 HB uses 25 tooth spline (recommended).

Where Hitec really shines is the high end market. If your budget is over $100, you can get into some very powerful Hitec *servos*. Most have titanium *gears* and are programmable, so you can dial in the performance and range to exactly what you need.

Axon Robotics, a relatively new company, offers programmable, titanium-geared servos in the $75+ range.

The last big player in the *servo* market in FTC is Savox, which produces great mid-high range *servos* (think $60-$100+). They are made with titanium *gears* (close to bulletproof) and are **fast**. Savox *servos* are mostly brushless and coreless, so they do tend to scream a little under load, but they're definitely worth it if your budget allows for it.

Recommended Servos

We no longer recommend low priced servos. Due to their low strength, they end up requiring multiple replacements over time, negating any cost benefits.

Bang for Your Buck

- goBILDA Dual Mode Servo (Torque) (25-2-torque)[124] - A very good price to performance servo. It is dual mode, has a higher than average output torque (and correspondingly lower speed), and steel gearbox.
- goBILDA Dual Mode Servo (Speed) (25-3)[125] - A very good price to performance servo. It is dual mode, has a higher than average output speed (and correspondingly lower torque), and steel gearbox.
- REV Smart Servo[126] - While very good price to performance, its brass gearbox makes it less recommended than goBILDA Dual Mode Servos
- Andymark High Speed/Torque Servo[127] - A newcomer to the market, this servo is extremely promising as a price to performance servo, with a unique 5mm hex output and imperial half inch mounting pattern. The high speed variant has more power output than the commonly used goBILDA Dual Mode servo. The high torque servo has a higher efficiency than the goBILDA Dual Mode servo. **It is unreleased at the time of writing, so these claims have not been verified.** This servo has been included due to Andymark's historical reliability.

Premium Options

- Axon Robotics MAX+[128] - The best price to performance high performance servo. It has a high efficiency and a high power output. In addition, this servo can track its absolute position via an analog output wire.
- Hitec titanium servos - A reliable choice, Hitec has a large variety of servos making it a good option for super specific servos such as non-standard form factors or specific qualities desired such as high speed or very high torque.

Specialty Servos

- **goBILDA 5 Turn Servo**[129]
  - goBILDA manufactures all three of their Dual Mode servos (Speed, Super Speed, Torque) in 5 turn varients, which can rotate 5 turns while still tracking position. These servos have high range, making them ideal for use with external gearboxes, but are more expensive and have a lower precision than the normal varients.

REV and goBILDA *servos* can be purchased from REV and goBILDA websites respectively. For all other servos, some good sources are ServoCity[130] or Amazon[131].

[124] https://www.gobilda.com/2000-series-dual-mode-servo-25-2-torque/
[125] https://www.gobilda.com/2000-series-dual-mode-servo-25-3-speed/
[126] https://www.revrobotics.com/rev-41-1097/
[127] https://www.andymark.com/products/programmable-servos
[128] https://axon-robotics.com/products/max
[129] https://www.gobilda.com/2000-series-5-turn-dual-mode-servo-25-2-torque/
[130] https://www.servocity.com/
[131] https://www.amazon.com/

9.6.2 Servo Usage Tips

Below are some tips on using servos in FTC®.

- Do not backdrive *servos*. Forcibly rotating a powered *servo* away from its position risks damaging the internal *gears*.
- Pay attention to a servo's travel range! The FTC API, by default, outputs 600-2400 µs. `ServoImplEx` can be used to increase the range to a maximum of 500-2500 µs.

```
ServoImplEx servo = hardwareMap.get(ServoImplEx.class, "myservo");
// ...
servo.setPwmRange(new PwmRange(500, 2500));
```

- *Servo* wires usually are black-red-white. Matching the colors is an easy way to check if the servo is plugged in correctly. *Servo* connectors provide no protection from plugging them the wrong way: if you rotate it 180 degrees, it will still fit - but the *servo* would not work. (It wouldn't be damaged, though). Thus, if your *servo* is not working, first check if they are plugged in correctly. Then check it again.
- When using *servo* wire extensions, use retaining clips[132] to prevent the connection from coming apart when someone pulls on the wire. Alternatively, electrical tape will work in a pinch.
- Do not use socket head screws to attach *servos* - when tightened, they will damage the plastic. Use button head screws or socket heads with a washer.
- *Servos* break very easily when subjected to lateral loads or bending of the *shaft*. For example, if you mount an arm or a claw directly on the *servo* without any additional precautions, it is very likely that you will break the *servo* first time you drive into the wall with the arm extended (and this will inevitably happen).

  To avoid that, use additional supports. The easiest way to do it is by using *Servoblocks* from Actobotics or goBILDA. These assemblies act as exoskeletons for the *servo*, providing additional support. They are expensive, but worth every penny. Additionally, REV offers the inside and outside channel *servo* bracket, which when paired with the aluminum servo *shaft* adapter and *bearing* assembly, fulfills the same function.

There are also some alternative designs of servo supports; one of them, which is not as strong as the original *Servoblock* but much more compact, is shown below (CAD[133] is also available):

- Use linkages. Instead of mounting some rotating piece directly on a *servo*, mount it so it can rotate around a pivot point and then connect it to the *servo* using linkage as shown below:
- If you need more power, use a REV Servo Power Module[134].

Term

Servo Power Module A Servo Power Module (SPM) is a device made by REV Robotics that boosts the voltage that the Expansion Hub provides to a *servo*. The Expansion Hub's output for servos is 5V at 6 amps, and the SPM boosts the voltage to 6V and up to 15amps.

This is important for servos under high load conditions such as the Savox servo.

[132] https://www.gobilda.com/servo-connector-clip-yellow-6-pack/
[133] https://myhub.autodesk360.com/ue2801558/g/shares/SH56a43QTfd62c1cd968b8829158db7626b9
[134] https://www.revrobotics.com/rev-11-1144/

Fig. 9: A servo in a ServoBlock

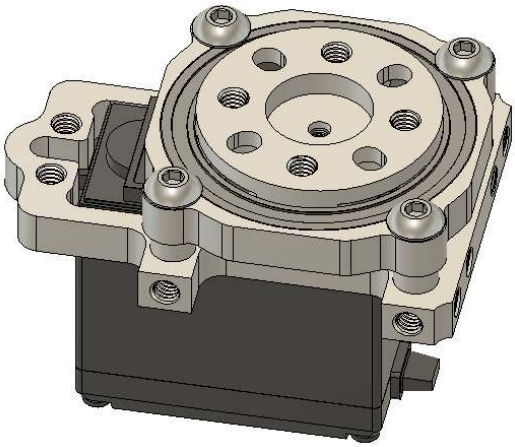


Fig. 10: Alternative *servo* support block

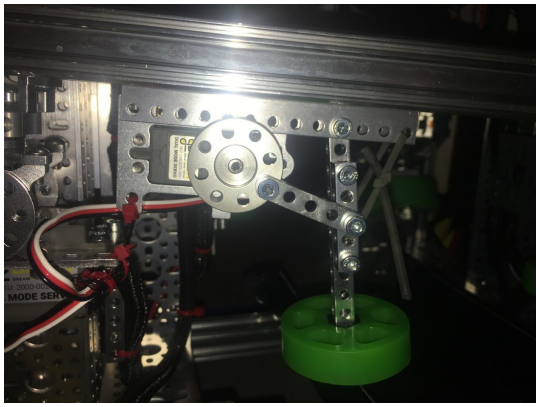


Fig. 11: Linkage example, courtesy of team 4137 Islandbots. A goBILDA flat beam is used as the link.

9.7 Sensor Glossary

Sensors are used in a variety of applications within FTC®. Sensors can give external feedback regarding the position of the robot (for example, relative to the field wall or to a vision target) or internal feedback (velocity, distance traveled, voltage, etc.). Sensors can also be used to determine the rotation of a mechanism and detect color.

9.7.1 Encoders/Potentiometers

- **Rotational**
  - **Absolute (+ Potentiometers)**
    - MA3 (am-2899[135])
    - Potentiometer (REV-31-1155[136])
  - **Relative**
    - REV Through Bore Encoder (REV-11-1271[137]). It has absolute support but over PWM, which is inaccessible within the FTC control system.
    - E4T (am-3132[138])
    - Generic (Sparkfun Rotary Encoder[139])
- **Positional**
  - Linear Potentiometers Slide Pot (Sparkfun Slide Pot[140])

[135] https://www.andymark.com/products/ma3-absolute-encoder-with-cable
[136] https://www.revrobotics.com/rev-31-1155/
[137] https://www.revrobotics.com/rev-11-1271/
[138] https://www.andymark.com/products/e4t-oem-miniature-optical-encoder-kit
[139] https://www.sparkfun.com/products/9117
[140] https://www.sparkfun.com/products/9119

9.7.2 Contact

- **Physical**
  - Limit Switches (3103-0001-0001/3103-0001-0002[141])
  - Endstops Generic (Sparkfun[142])
  - Touch Sensor REV (REV-31-1425[143])
- **Magnetic**
  - Hall Effect Sensor REV (REV-31-1462[144])

9.7.3 Optical

- **Color**
  - Adafruit RGB
  - REV Color
  - MR Color
- **Computer Vision**
  - **Hardware**
    - PixyCMU
  - **Software**
    - OpenCV (EasyOpenCV)
    - Vuforia
    - TFLite

9.7.4 Distance

- ToF
- Ultrasonic

9.7.5 Other

- **IMU**
  - Accelerometer
  - Gyroscope
  - Compass
  - Magnetometer

[141] https://www.gobilda.com/limit-switches/
[142] https://www.sparkfun.com/products/13013
[143] https://www.revrobotics.com/rev-31-1425/
[144] https://www.revrobotics.com/rev-31-1462/

9.7.6 Logic Level Converter

The old Modern Robotics system run on 5v sensor logic. The new REV Robotics system uses 3.3v. For most off the shelf sensors, this doesn't cause any problems, but for some existing FTC sensors it does. To solve this REV sells boards, called logic level converters[145], that convert the sensor data to be readable by the REV hubs. The REV Expansion Hub[146] guide has a chart detailing what adapters are needed for what sensors.

Attention: According to REV testing, goBILDA, REV and TorqueNado motors don't need logic level converters, but only some NeveRest motors worked with no discernable reason why.

It is ideal to not use logic level converters to simplify your wiring. If you need to, there is a best practice. Electrical tape the connectors on either end, this helps with static, and it keeps it from being physically disconnected. This does produce a very noticeable effect with encoders on fields with lots of static.

The second tip is to never tape over the middle or LED. The board generates a very small amount of heat, and it's very easy to overheat if it can't ventilate, also don't fully enclose it in any cases without holes.

9.8 Tips and Tricks

In addition to what is written in the official resources, there are a couple of additional tips.

The traditional *XT30* connector that is used to REV is prone to breaking. It is highly recommended that teams replace *XT30* connectors with *Anderson PowerPole*, or put adapters on their current wires. An example of an adapter is this[147]. In lieu of this, teams can also 3D print strain relief connectors on the Expansion Hub to prevent *XT30* disconnects. The file can be found on Thingiverse or through this link[148].

Fig. 12: An example *Anderson PowerPole* (A) to *XT30* (B) adapter. They can be found at ServoCity[149] or REV Robotics[150].

The Tamiya connectors found on many of the FTC® legal batteries are very weak and prone to becoming unreliable after many repeated plug/unplug cycles. It is recommended that teams crimp new *Anderson PowerPole* connectors onto the battery.

[145] https://www.revrobotics.com/rev-31-1389/
[146] https://docs.revrobotics.com/duo-control/sensors/5v-sensors#logic-level-converter
[147] https://www.servocity.com/anderson-powerpole-to-female-xt30-adaptor
[148] https://www.thingiverse.com/thing:2887045
[149] https://www.servocity.com/anderson-powerpole-to-female-xt30-adaptor/
[150] https://www.revrobotics.com/REV-31-1385/

It is highly recommended for teams to use the REV grounding strap[151] and the REV USB strain relief[152] to help prevent disconnections.

To protect wires, teams often use wire loom[153], and to help wires extend far, teams often use cable carrier (also known as cable chain).

9.9 Glossary

Gauge Wire gauge refers to the diameter of wire. AWG stands for American Wire Gauge, the general system used in the US. The larger the gauge number, the smaller the wire diameter. Generally, *servo* wires are 22 AWG and motor wires are 18 AWG.

Grounding Strap The REV Grounding Strap is used to ground the metal frame of the robot to the *XT30* port of the Expansion Hub. It is currently the only legal way to ground your robot.

USB Retention Mount The USB Retention Mount, sold by REV, is a plastic part affixed to the Expansion Hub that relieves stress on the USB Mini port. This is especially important because if the USB cable is loose or disconnected, the robot phone cannot communicate with the Expansion Hub, causing a disconnect.

Note: For teams using an expansion hub, it is highly recommended for teams to purchase the USB retention mount.

[151] https://www.revrobotics.com/rev-31-1269/
[152] https://www.revrobotics.com/rev-41-1214/
[153] https://www.amazon.com/Black-Split-Tubing-Cover-Marine/dp/B00J7RD6ZI/ref=sr_1_13?keywords=wire+loom&qid=1562452458&s=gateway&sr=8-13

10 Software

This section covers the basics of software for FTC®.

10.1 Getting Started

Basic information for starting out on your programming journey.

Check out the official FTC|reg| SDK Example Programs here![154]

10.1.1 Fundamental Concepts of Programming

For almost any programming language, whether it's Java, Python, or Blocks, there are concepts in coding that transfer across languages. These ideas are foundational when learning to program and should be applicable in FTC® and beyond.

This section is primarily for people with limited Java experience. However, even if you are more experienced, it may still be helpful to skim through the section, as you might find concepts that have not yet been introduced to you.

Examples will mostly be in Java, where `//` indicates a comment which the program ignores and is used for people to read.

```
int number; // Declaring that number will contain an integer.
number = 5; // Setting a value so that the variable holds something.

int secondNumber = 6; // Doing both above.

int total = number + secondNumber; // Math.
System.out.println(total); // Printing, it will show up as 11.
```

[154] https://github.com/FIRST-Tech-Challenge/FtcRobotController/tree/master/FtcRobotController/src/main/java/org/firstinspires/ftc/robotcontroller/external/samples

Java-Specific Exploratory Questions

- If I didn't set a value for number and then I printed it, what would it print?
- What other operations can I do with number and secondNumber?
- Can I set a decimal to number? If not, what happens?
- What is `System.out.println();`?
- Delete one character in the code. Remember the error (if any), and then undo it. Delete another part. How many different errors can you get?

There are different types of variables

- Numbers (Integers, Floats, Doubles)
- Strings (Text) or characters
- And a lot more depending on the language (Ex: Arrays)
- They help tell the program know the basis of what it should do with a variable.

```
String coolName = "Gluten Free";
String restOfSentence = " is epic.";

// Prints out the sentence by combining the strings, unlike adding if they // were␣
↪integers
System.out.println(coolName + restOfSentence);

// Fun fact: Using + to add strings is called String Concatenation
```

Java-Specific Exploratory Questions

- Replace the text in `coolName` to something else. Your name, a phone number, your favorite anime. What about emotes and copypastas? What about characters in other languages?
- Try adding a number and a string, what happens?
- Is it possible to add multiple strings and numbers together?

Important Control Structures

Be sure to familiarize yourself with basic control structures (if/else statements, for loops, while loops, and for-each loops). These control structures are by far the most commonly encountered, and thus, familiarizing yourself with these principles is extremely important (not just for FTC, but programming in general). However, there are a few control structures that are far less common that are extremely useful in FTC; namely *Finite State Machines* (page 330).

Data Structures (Arrays)

Data structures are a method of organizing and storing large amounts of data. There are a lot of different types of data structures that mostly differ in the relationships between data points, and we would recommend that you read into them. We will only go over a few here.

Arrays[155] Arrays are the most basic and simple data structure. When an array is initialized, its size must be set, and it cannot be changed.

If you wish to expand an array, a new one must be created and all of the old data copied over. Elements of an array are stored adjacent to each other in memory, so when they are accessed the number you want to access times the amount of bits in the object in the array is added to the starting address, and data is accessed from there.

This means that arrays are incredibly efficient at reading data in a nonlinear order.

ArrayList[156]

Object Oriented Programming in Java[157]

10.1.2 Options for Programming

There are three recommended options for teams to use when programming a robot to do various tasks: Blocks, OnBot Java, and Android Studio.

Blocks

The Blocks Programming Tool is a simple way to code simple actions for the robot. It has a colorful and lego-like design and includes sounds of block pieces of snapping together.

Using a device and a robot controller phone, connect your device to the phone's Wi-Fi by clicking the three dots in the top right of the *RC* and select "Program & Manage". This will display the WiFi SSID and password for you to connect to. Go onto your browser (preferably Chrome, though other modern browser will likely work) and enter the address shown on the *RC*.

Advantages

- Good for beginners: one of the easiest and simplest options to use.
- Easy to set up: requires a device (laptop, chromebook, tablet) and a phone in the Program & Manage screen
- Programs can be saved directly to the phone
  - Changes can be made quickly
- Doesn't require an external Wi-Fi connection

[155] https://www.geeksforgeeks.org/arrays-in-java/
[156] https://www.geeksforgeeks.org/arraylist-in-java/
[157] https://www.geeksforgeeks.org/classes-objects-java/

Disadvantages

- Not recommended if you're already familiar with some type of programming
  - Primarily a teaching tool for people with no programming experience
- This will change your primary Wi-Fi network to the *Robot Controller's* Direct Wi-Fi network, and as such, you will be unable to access the internet while connected to this particular network.
  - Teams can get around this by purchasing an external Wi-Fi module that allows computers to run 2 Wi-Fi networks at once.
- Sacrifices flexibility and application for simplicity
- Don't ever use it on a phone, unless you're out of options. (Not phone-friendly)
  - You may make a bigger mess if you try to.

OnBot Java

OnBot Java uses a similar method of using a browser to code and save directly to the phone. The difference being that a programming language called Java is used instead.

Advantages

- Recommended if you're learning or have learned some programming, even better if you know a little Java.
- Greater flexibility than Blocks
- There are a lot more resources available in case you need help
- More applicable to the real-world than Blocks
- Maintains most of the advantages of Blocks

Disadvantages

- Connecting to the robot controller's Wi-Fi network will prevent you from using anything on the internet including video tutorials and online communication
- Using external libraries is difficult and borderline impossible
- Steeper learning curve than Blocks

Android Studio

Android Studio is a comprehensive Integrated Development Environment (IDE) that uses Java to program the phones. Instead of using a browser to upload code, Android Studio will compile your Robot Controller code into a .apk file (an app installer), and install that on the phone.

Advantages

- Recommended if you're learning or have learned some programming, even better if you know a little Java.
- Much greater flexibility than Blocks.
- Much easier to integrate libraries like EasyOpenCV[158], FTC Dashboard[159], FTCLib[160], and Road Runner[161].
- Can use plugins like Road Runner[162].
- Can use either a USB connection to the *RC* phone, or a wireless connection to upload code.

Note: Deploy times can be sped up by using OpenRC Turbo[163].

- Can debug in real-time
- Many resources for Java, Android Studio, and IDEA
- Can use other programming languages

Disadvantages

- Connecting to the robot controller's Wi-Fi network will prevent you from using anything on the internet including video tutorials and online communication, unless you have a second Wifi adapter (cheap and easy)
- Relatively easy setup process, but time consuming and is a hefty install (3GB of files between Android Studio, ftc_app, and other libraries)
- Issues can be difficult to diagnose and solve

Other Programming Languages

Kotlin

Kotlin is a relatively new and rapidly growing programming language from Jetbrains, the creator of the IntelliJ IDE, which Android Studio is based off. Kotlin was made to be completely compatible with Java but be easier to work with. Google recently announced it as an official Android language, then announced they are going "Kotlin first" but still keeping Java support.

[158] https://github.com/openftc/easyopencv
[159] https://github.com/acmerobotics/ftc-dashboard
[160] https://github.com/ftclib/ftclib
[161] https://github.com/acmerobotics/road-runner
[162] https://github.com/acmerobotics/road-runner
[163] https://github.com/OpenFTC/OpenRC-Turbo

Advantages

- Concise, readable, easy to edit code
  - Easy to write as it takes much less code to do the same thing
- Both optional type inference and a stronger type system than Java
- Null safety
- Thread safety
- Functional programming
- Seamless integration with Java code and libraries
- Very easy to transition from Java

Disadvantages

- Not widely used in FTC® yet
- New and has fewer community resources for training
- Not recommended for programmers who need large amounts of help from other teams

C and C++

C and C++ are native programming languages compatible with Android. Very few teams have used C++. This is typically used for only part of the code, with the majority being Java or Kotlin.

Advantages

- Fast execution for extremely resource-intensive applications.
- Supports more libraries

Disadvantages

- Rarely needed
- Very difficult to set up
- Difficult to debug code
- Very few teams can help you
- Very few online resources

10.1.3 LinearOpMode vs OpMode

There are two OpMode classes within the FTC® SDK: `OpMode` and `LinearOpMode`. The one you use affects how you write the program. For examples of how to use OpMode and LinearOpMode, refer to the example OpModes in the SDK[164].

LinearOpMode Methods

- `runOpMode()`: Code inside this method will run exactly once after you press the INIT button. This is where you should put all code for the OpMode.
- `waitForStart()`: This method pauses the Op-Mode until you press the START button on the driver station.
- `isStarted()`: returns `true` if the START button has been pressed, otherwise it returns `false`.
- `isStopRequested()`: returns `true` if the STOP button has been pressed, otherwise it returns `false`.
- `idle()`: calls `Thread.yield`, allowing other threads at the same priority level to run.
- `opModeIsActive()`: returns `isStarted() && !isStopRequested()` and calls `idle()`.
- `opModeInInit()`: returns `!isStarted() && !isStopRequested()` and does not call `idle()`.

OpMode Methods

- `init()`: Code inside this method will run exactly once after you press the INIT button on the driver station.
- `init_loop()`: Once the code in `init()` has been run, code inside this method will run continuously until the START button is pressed on the driver station.
- `start()`: Code inside this method will run exactly once after you press the START button on the driver station.
- `loop()`: Once the code in `start()` has been run, code inside this method will run continuously until the STOP button is pressed on the driver station.
- `stop()`: Code inside this method will run exactly once after you press the STOP button on the driver station.

Note: As of SDK version 8.1, when executing `OpModes` there is a negligible delay of one millisecond between calls of `loop()`. Previously, it had unpredictable delays, however since 8.1 it is similarly performant to `LinearOpMode`.

---

[164] https://github.com/FIRST-Tech-Challenge/FtcRobotController/tree/master/FtcRobotController/src/main/java/org/firstinspires/ftc/robotcontroller/external/samples

Conclusion

Overall, the use of `LinearOpMode` or `OpMode` is up to preference. Game Manual 0 uses `LinearOpMode` everywhere for consistency.

10.1.4 Reading and Writing to Hardware

When using the FTC® SDK, there are a variety of built in hardware classes which can be used to communicate with hardware on the robot such as DC Motors, *Servos*, and Sensors.

Creating and Instantiating Hardware Objects

The first thing required to properly create an object is to import its class. In Android Studio, if the class is referenced without being imported Alt+Enter can be pressed to automatically import it. After it is imported, the next step is to create the object:

```
private DcMotor liftMotor;
```

After the object is created, it must be instantiated. Part of the `OpMode` superclass is something called `hardwareMap`. `hardwareMap` is used in the FTC SDK to instantiate objects rather than calling a constructor.

It contains all of the information entered into the configuration on the Robot Controller, such as names of hardware and what port it is plugged into. Here is an example of instantiating the motor we created above:

```
liftMotor = hardwareMap.get(DcMotor.class, "Lift Motor");
```

Whatever sensor you are using, you will pass that class into the spot where `DcMotor.class` is. For example, if liftMotor was a Servo, `Servo.class` would be passed instead.

For the second argument, you pass whatever the device is named in the Robot Controller configuration. `hardwareMap` will then go find what port the device with that name is plugged into, which allows the hardware to be accessed.

Examples of Using Common Hardware Components

DC Motor

```
DcMotor leftMotor = hardwareMap.get(DcMotor.class, "Left Motor");
DcMotor rightMotor = hardwareMap.get(DcMotor.class, "Right Motor");

DcMotor elevatorMotor = hardware.get(DcMotor.class, "Elevator Motor");
DcMotor intakeMotor = hardware.get(DcMotor.class, "Intake Motor");
```

After a `DcMotor` is instantiated, there are a few variables you can set to affect how the DC Motor runs. The first of these is direction:

```
leftMotor.setDirection(DcMotor.Direction.REVERSE);
rightMotor.setDirection(DcMotor.Direction.FORWARD);
```

Changing the direction of the motor does exactly what should be expected, it changes the direction. If a power of 1 is applied to the motor while it is in forward mode, it will turn one direction. If it is in reverse, a power of 1 will spin it in the other direction. If you face the shaft of the motor towards you, forward is counterclockwise (with the exception of NeveRest motors).

Next, there are two zero power behaviors that can be adjusted:

```
leftMotor.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.BRAKE);
rightMotor.setZeroPowerBehavior(DcMotor.ZeroPowerBehavior.FLOAT);
```

Changing this variable affects how the DC Motor behaves while a power of 0 is applied. BRAKE will cause the motor to try and slow itself down if it is moving (it will NOT cause the motor to hold its position if not already moving), while FLOAT causes the motor to glide to a stop, letting friction do all the work.

Finally, there are four different run modes that can be used with DC motors:

```
leftMotor.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);
rightMotor.setMode(DcMotor.RunMode.RUN_USING_ENCODER);

elevatorMotor.setMode(DcMotor.RunMode.RUN_TO_POSITION);
intakeMotor.setMode(DcMotor.RunMode.STOP_AND_RESET_ENCODER);
```

It is important to note that encoder values can be read in any of these modes provided an encoder is properly plugged in. These modes just change how the motor reacts to these encoder values. The REV Robotics documentation has an explanation of all four run modes[165].

Warning: `RUN_TO_POSITION` can be a convenient way to control a single-motor mechanism, as it offloads all control work; however, since every motor is dealt with independently, it is inadvisable to use this on mechanisms with multiple motors, especially drivetrains.

Encoders

Term

Encoder An encoder refers to a device that tracks (generally) rotational movement around an axis.

> There are both absolute and relative encoders. An absolute encoder will report at exactly what angle the shaft is compared to its absolute "zero". A relative encoder will report how far the *shaft* has rotated since it started tracking (for example, when autonomous starts). Relative encoders will have a quadrature output, whereas absolute encoders generally have analog or i2c outputs.
>
> Encoders are used to help find the position of where the robot, or one of its mechanisms, is.

While all FTC legal motors contain built in relative quadrature encoders, they must be wired separately and are not required for use. External encoders may be used and plugged into an encoder port so long as they use the quadrature communication protocol.

Accessing encoders requires calling one method on the DcMotor object, `getCurrentPosition()`, which returns the current position of the encoder plugged into the port. This number may be arbitrary at the beginning of an opmode, and is not reset to 0 unless `STOP_AND_RESET_ENCODERS` is used or power is cycled to the expansion hub.

[165] https://docs.revrobotics.com/duo-control/programming/using-encoder-feedback#choosing-a-motor-mode

Important: There is no real standardized terminology when dealing with quadrature encoders. The SDK uses "CPR" or Counts Per Revolution by default,. You may also see some datasheets list "PPR" or Pulses per Revolution. One pulse can be equivalent to anywhere from 1 to 4 SDK "counts". Be careful when reading datasheets!

Warning: Encoders with high numbers of Counts per Revolution, such as the REV Through Bore Encoder, can lose steps if plugged into ports 1 or 2. In addition, calls to `getVelocity()` on a DcMotorEx object may overflow with high counts per revolution encoders, due to the returned number only being a 16 bit signed integer.

Servo

```
Servo relicServo = hardwareMap.get(Servo.class, "Release Servo");
```

After instantiating a `Servo`, there are two main functions that can be called: `setPosition()` and `getPosition()`.

```
releaseServo.setPosition(0.75);
telemetry.addData("Release Servo Target", releaseServo.getPosition());
```

`setPosition()` sets the position of the *servo*. The SDK will use a built-in control loop with the *servo's* potentiometer to drive the *servo* to that position and hold that position. `setPosition()` takes in a double between 0 and 1, where 0 is the *servo's* lower limit of rotation and 1 is the *servo's* upper limit of rotation. Everything between is directly proportional, so 0.5 is the middle, 0.75 is 3/4 the way up, etc.

`getPosition()` does not return the *servo's* current position, rather its current target position. If a variable for the *servo's* current target position is stored properly, this function should never be needed.

Continuous Rotation Servo

```
CRServo intakeServo = hardwareMap.get(CRServo.class, "Intake Servo");
```

A CRServo has one main method; `setPower()`. This works very similarly to `DcMotor` 's `setPower()`, meaning that passing it 0 makes it stop, passing it 1 makes it go forward at full speed, passing it -1 makes it go backwards at full speed, and everything in between.

```
intakeServo.setPower(0.75);
```

Digital IO

```
DigitalChannel digitalDevice = hardwareMap.get(DigitalChannel.class, "digital device
↪");
```

A DigitalChannel has a couple main methods. `setMode()` is used to set the port as either an OUTPUT or INPUT port, `getState()` returns the current state of the port (only works in INPUT mode), and `setState()` sets the state of the port (only works in OUTPUT mode)

Tip: Digital ports start by default in INPUT mode

Danger: Digital ports are pulled UP to prevent floating. This means that there is a resistor between the port and 3.3V so the port reads HIGH by default when nothing is connected. As a result digital devices MUST connect the digital pin to ground when closed, then leave it unconnected when open. For limit switches, this means connecting one lead to ground and the other to the digital port. **Connecting this wrong (connecting 3.3V to the digital port) may cause instability and can cause your expansion hub to crash**

Analog Input

```
AnalogInput analogInput = hardwareMap.get(AnalogInput.class, "analog input");
```

An `AnalogInput` has one main method: `getVoltage()` which is used to get the current input voltage to the port.

Note: Although `getMaxVoltage()` returns 3.3v, the expansion and control hub analog input ports can safely handle up to 5v.

A Note on Hardware Call Speed

Every hardware call you make, (whether it be setting the power for a motor, setting a *servo* position, reading an encoder value, etc.) will take approximately 3 milliseconds to execute, except for I2C calls which can take upwards of 7ms. This is because behind the scenes, the SDK may need to make multiple hardware calls in order to perform the I2C operation.

Note: When using a Control Hub, you may see considerably faster hardware call times because the Control Hub uses a direct UART connection to the Lynx board instead of going through USB and a middle-man FTDI as happens when using a phone.

These times may seem fast, but they add up quickly. Consider a control loop to drive forward for N encoder counts while maintaining heading using the IMU. This would require 5 normal hardware calls (4 set power + 1 read encoder) and an I2C call (IMU) which means that the loop cycle would take approximately 22ms to execute, and thus run at approximately 45Hz.

This means that it is critical to minimize the amount of hardware calls you make in order to keep your control loops running fast. For instance, do not read a sensor more than once per loop. Instead, read it once and store the value to a variable if you need to use it again at other points in the same loop cycle.

Using a bulk read hardware call can help with this problem. A bulk read takes the same 3ms to execute as any other normal hardware call, but it returns far more data. In order to be able to use bulk reads, you must be running SDK v5.4 or higher. See *Bulk Reads* (page 319) for more information

10.1.5 Using Android Studio

Android Studio[166] is an integrated development environment (IDE) for Android app development based on IntelliJ. It compiles your code to an apk which is then installed onto the Robot Controller: either the Control Hub or a legal Android phone.

Downloading Android Studio

If you've already downloaded Android Studio, you can move on to the next step, which is *setting up the SDK* (page **??**).

The steps to download and setup Android Studio are:

1. Check to make sure your system meets the necessary requirements[167]
2. Install the *latest* version of Android Studio from https://developer.android.com/studio/index.html
3. Run the executable, follow the setup wizard, and use any and all recommended development kits

Setting up the SDK

Now that you have Android Studio installed, you're going to want to use the current season's SDK (software development kit) where you will create your team's code.

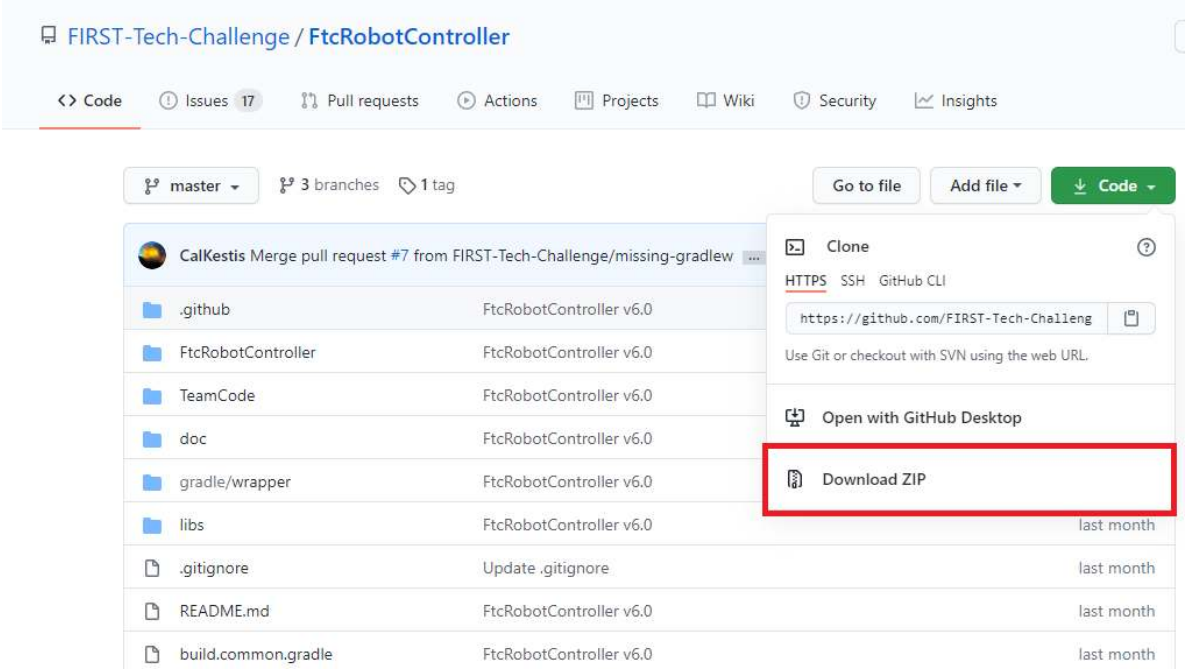
Downloading the SDK

The SDK is publicly released to a GitHub repository every season. The current season's SDK can be found in the FtcRobotController[168] repository.

Downloading the ZIP

1. When you're at the repository, click the green "code" button. Then, select "Download ZIP."

[166] https://developer.android.com/studio/intro
[167] https://developer.android.com/studio#Requirements
[168] https://github.com/FIRST-Tech-Challenge/FtcRobotController



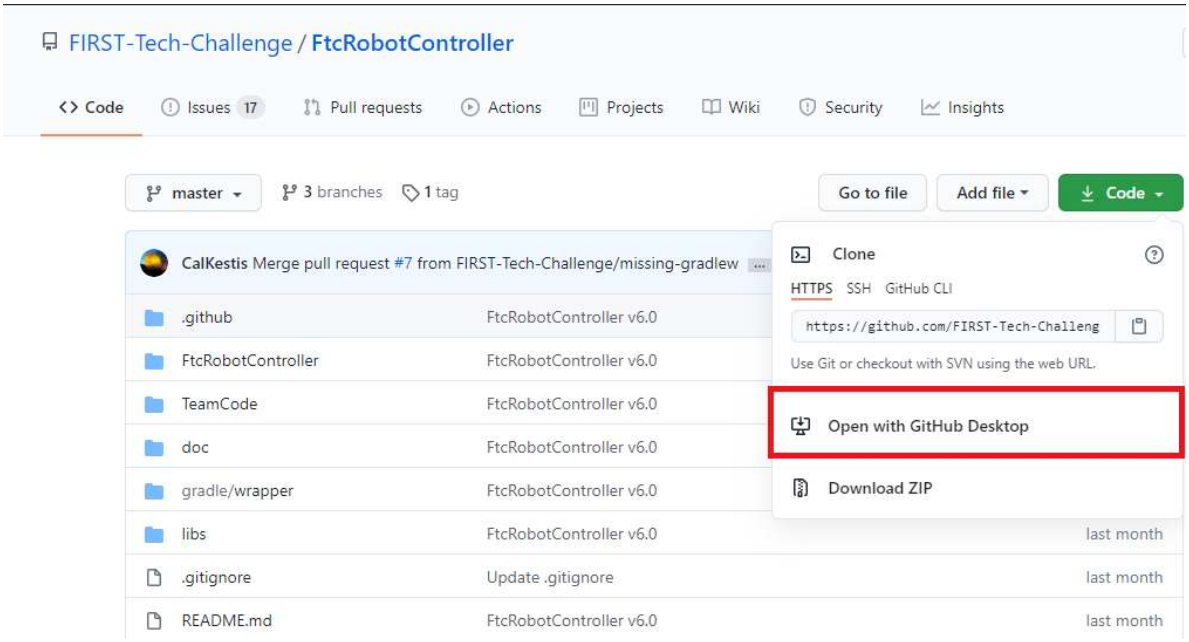
2. Then, save it to the desired location in your computer.

3. After it is saved, extract the contents of the ZIP and place them into whatever desired location. You should see the contents of the SDK inside of the folder location.

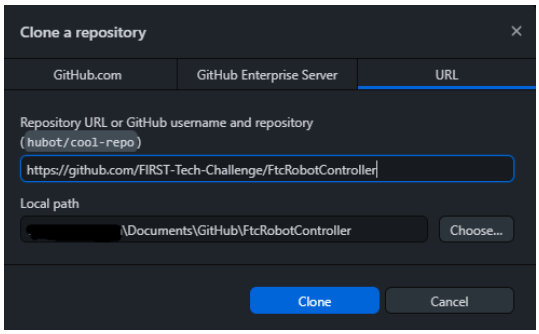
Using GitHub Desktop

1. Install GitHub Desktop[169]
2. Open the SDK repository[170] in a browser.
3. Click the green “code” button, and then select “Open with GitHub Desktop.”

[169] https://docs.github.com/en/desktop/installing-and-configuring-github-desktop/installing-and-authenticating-to-github-desktop
[170] https://github.com/FIRST-Tech-Challenge/FTCRobotController



4. Clone the project.



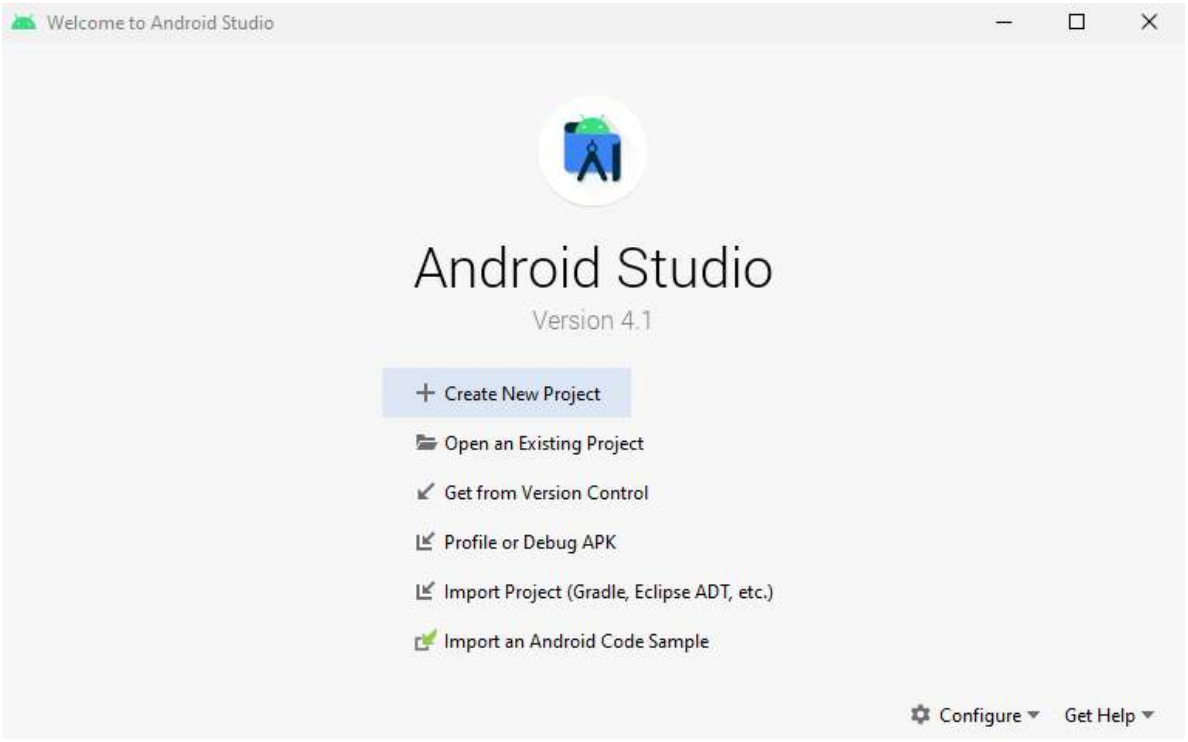
From the Command Line

1. Install git[171]
2. Open the terminal (probably bash) in the desired resource location.
3. Use `$ git clone https://github.com/FIRST-Tech-Challenge/FtcRobotController.git`

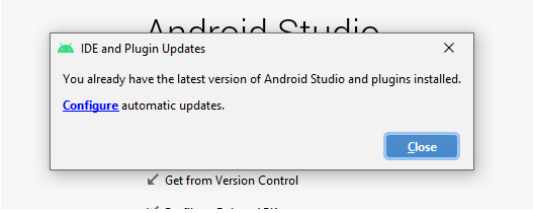
[171] https://github.com/git-guides/install-git

Opening the SDK on Android Studio

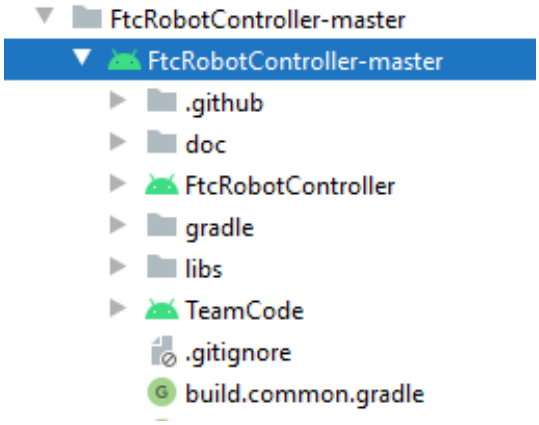
1. Open Android Studio. If you have another project open, close it.



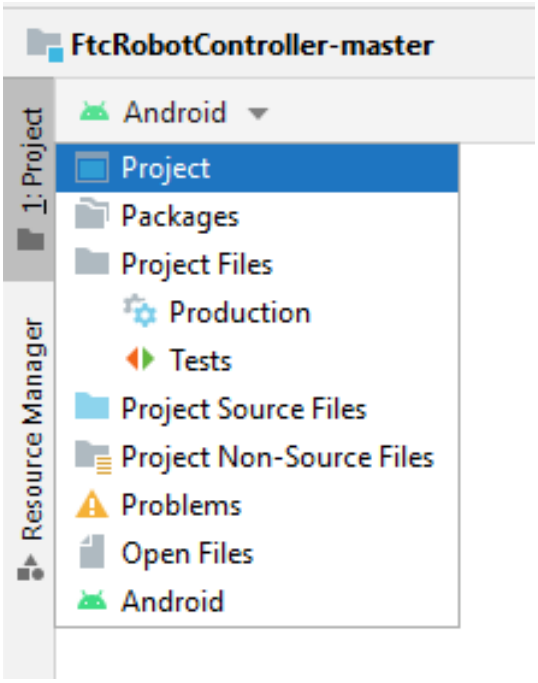
2. Check for updates. Click on the “configure” dropdown and select “check for updates.” If you do not have the latest version, download the updates.



3. Select “Import Project.” Navigate to where you have the SDK saved on your computer. Choose the directory that has the Android logo.



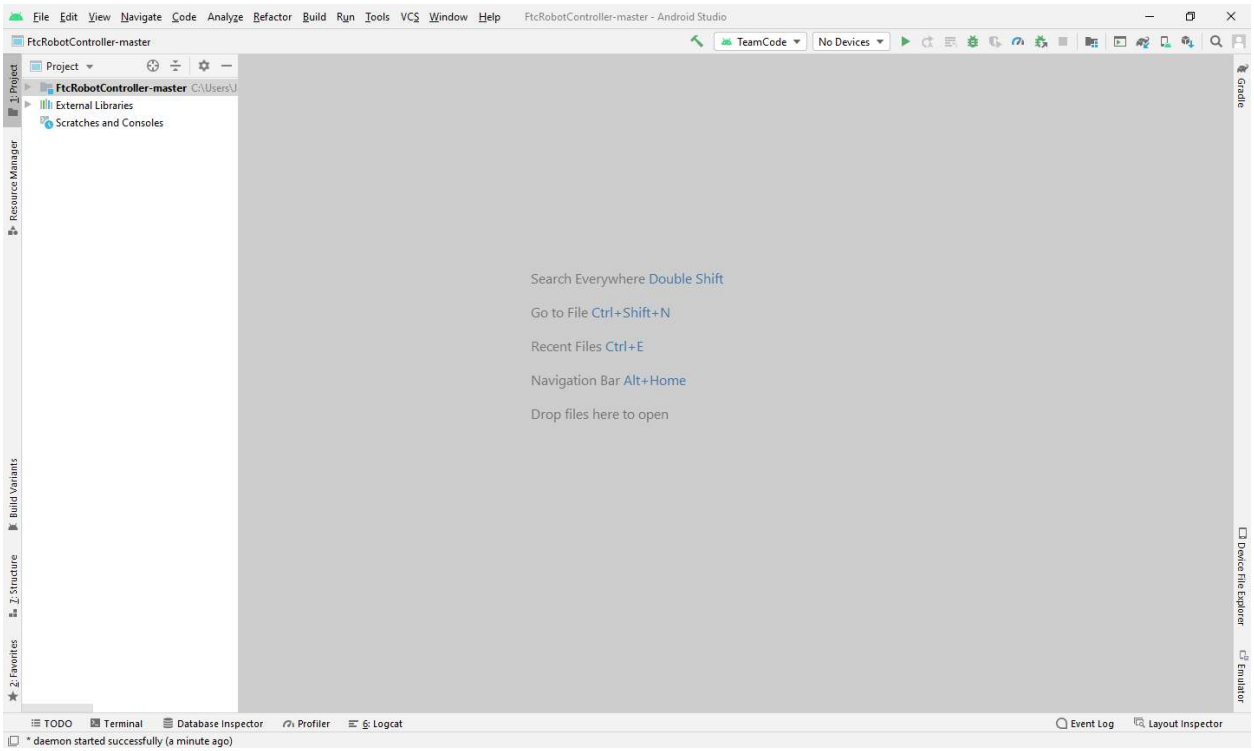
4. Change to project view. In the top left corner should be a dropdown that allows you to change the way you are looking at your project.



5. Wait for *Gradle* (page 270) to complete the build. This indicator should be located at the bottom of the window by default.

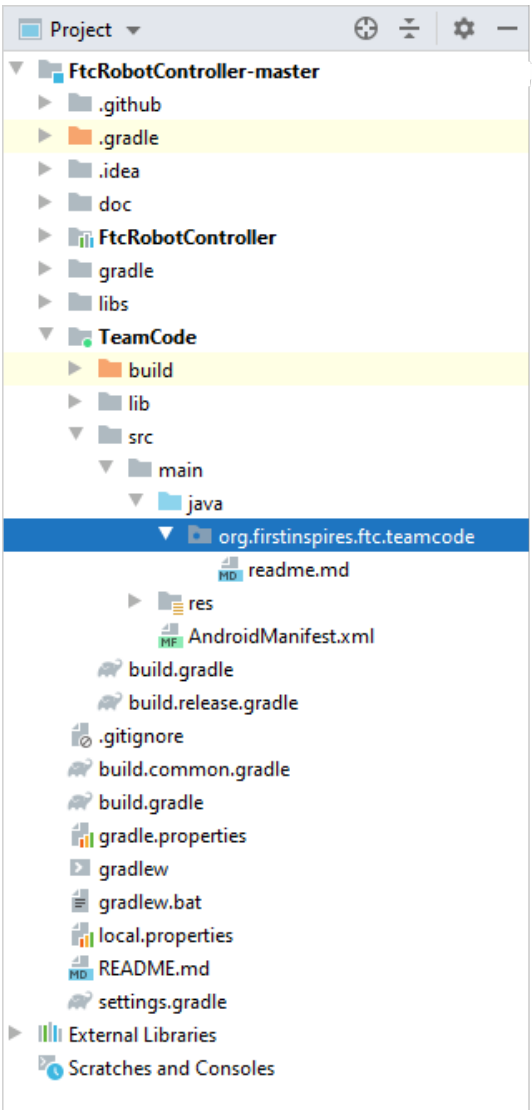
Layout

Android Studio can look intimidating at first glance, but there are only a few features needed to use it properly.

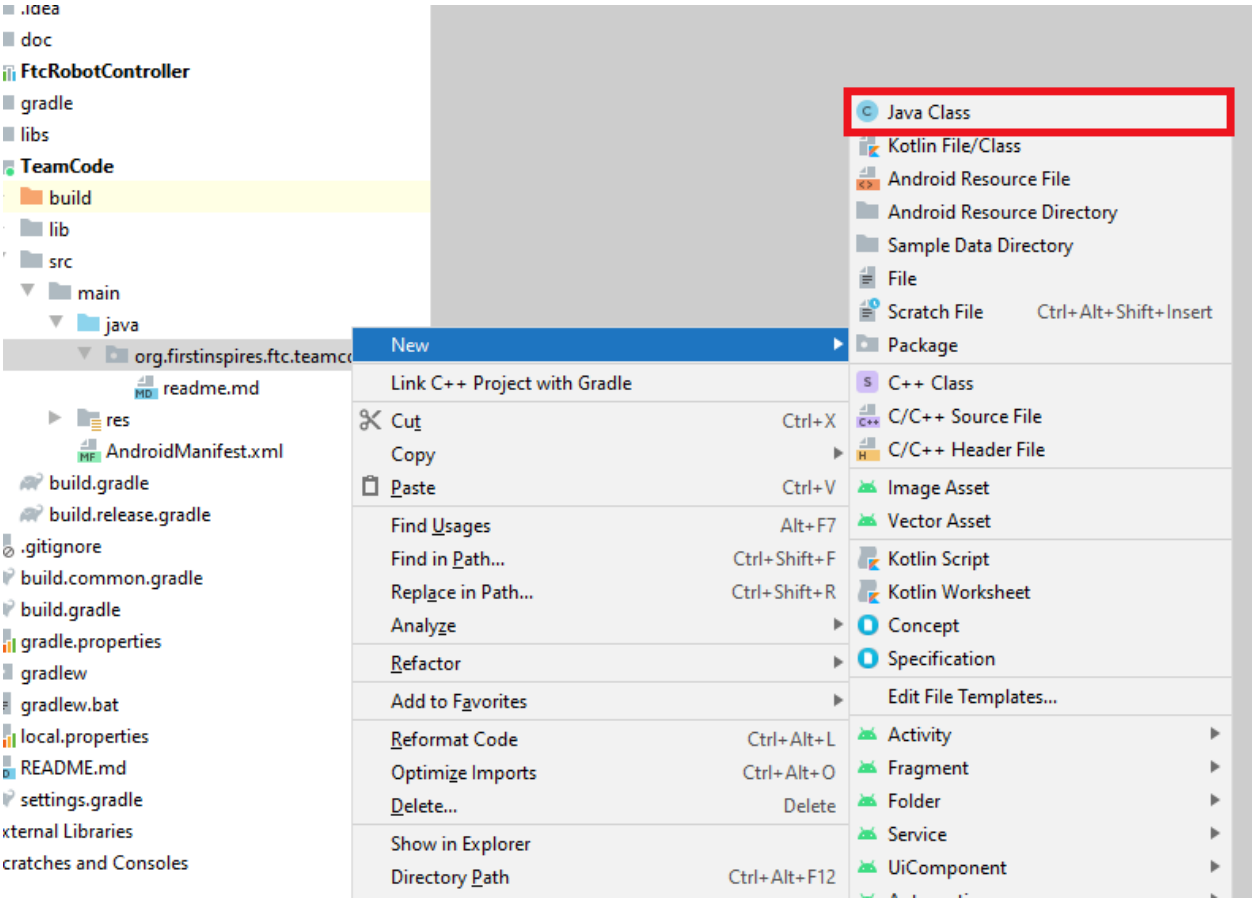


Creating Classes

The first thing to note in the project view is the dropdown with the name of the project. If you drop that down, you will see all of the Gradle files and directories. Navigate to the TeamCode folder. In the teamcode folder you will see an `org.firstinspires.ftc.teamcode` package.



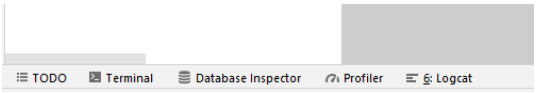
This is where you will create your code for the robot. To create a new Java class, right click on the package, select New, and then choose “Java Class.”



Alternatively, you can select the "Package" option if you want to create a subfolder for organization purposes. Then, you can create classes in those packages.

Terminal and Logcat

Near the bottom left of the application, you will find tabs for the local terminal and logcat. These are useful tools for working with your program.



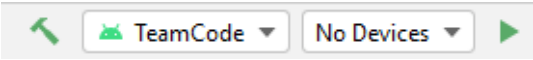
Some useful information on using the logcat can be found here[172].

Note: You can build your program through the commandline via the local terminal. Click on the terminal tab and then input `gradlew :TeamCode:clean :TeamCode:build`. This will delete the previously compiled files and build your TeamCode module.

[172] https://developer.android.com/studio/debug/am-logcat

Installing Your Program

To install your program onto the Robot Controller, you will use the play button located near the top right of the application window.



Next to it you will see a dropdown for devices. When you connect your Robot Controller to your computer (using the correct cable), the device should appear in the dropdown after some time. Then, click the play button and your program will install onto the device.

Tip: Occasionally the app will fail to start on the robot controller, leaving the driver station in a disconnected state. If this occurs you can open the terminal and run

```
adb shell am start -n com.qualcomm.ftcrobotcontroller/org.firstinspires.ftc.robotcontroller.internal.PermissionValidatorWrapper
```

to remotely start the Robot Controller app.

If you run into any problems with this process, refer to the official REV documentation[173]. Some useful pages from the REV site are:

- Troubleshooting the Control System[174]
- Deploying Code Wirelessly[175]

If you're still stuck you can ask for help in the FTC|reg| Discord[176].

Gradle

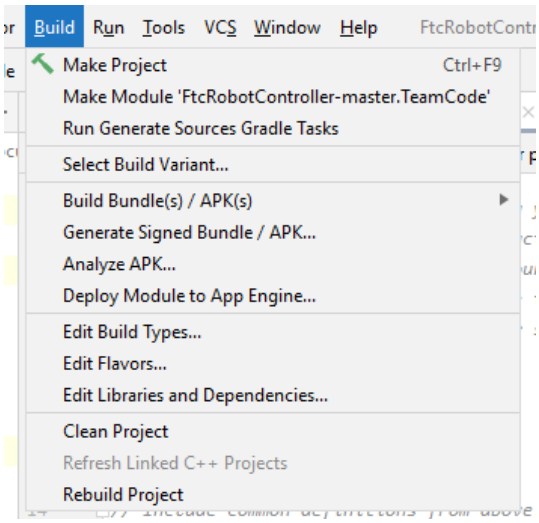
Gradle is a build tool for software development. In the scope of FTC, it is used to build and manage dependencies for your project.

When you update any of your Gradle files you will need to perform a Gradle sync, which syncs your project to the changes and rebuilds it. In your `build.common.gradle`, you will find information for how your robot controller application is built.

Rebuilding

You can rebuild your project easily with the build dropdown.

[173] https://docs.revrobotics.com/duo-control/
[174] https://github.com/FIRST-Tech-Challenge/FtcRobotController/wiki/Android-Studio-Tutorial
[175] https://docs.revrobotics.com/duo-control/programming/android-studio-using-wireless-adb
[176] https://discord.com/invite/first-tech-challenge



To rebuild from a clean project, press the clean project option. This removes old compiled files from your project so you can completely rebuild your project. It clears any production files, generated files, etc. This is useful for making sure old artifacts don't break anything when you build your code. When you next build your project, it will do so from scratch with no old compiled files to which it can refer. To rebuild your project, press the rebuild option.

Invalidate and Restart

Sometimes you can get errors after moving things around, refactoring, etc. The first step is to try cleaning the project and doing a rebuild. If this doesn't work, you might have confused Android Studio because it caches information about your project structure.

The most common way to fix these errors is to do an invalidate and restart. In the file dropdown, there will be an option for this and then you will choose Invalidate and Restart. This clears the cache and restarts your Android Studio, which then should perform a Gradle rebuild.

Adding Dependencies

If you want to add dependencies to your project, you can do so in the `build.gradle` file in the TeamCode directory.

There should be a dependencies block at the bottom of the file.

```
// Include common definitions from above.
apply from: '../build.common.gradle'
apply from: '../build.dependencies.gradle'

dependencies {
     implementation project(':FtcRobotController')
     annotationProcessor files('lib/OpModeAnnotationProcessor.jar')
}
```

Some dependencies require changes to your other Gradle files. Make sure to read the installation instructions for whatever dependency you want to add.

Next, you add a line in the dependencies block to implement the dependency. This is generally done with `implementation 'com.package.name'`.

```
dependencies {
    implementation project(':FtcRobotController')
    annotationProcessor files('lib/OpModeAnnotationProcessor.jar')

    implementation 'com.package.name:name:version'
}
```

Refer to the instructions of whatever library you are using for the correct package name and version.

Finally, perform a Gradle sync.

Upgrading to Java 8

By default, the SDK's version of Java is set to 7. Java 8 is also supported. You might want to upgrade your version of Java from 7 to 8 if you want to use features such as lambdas or generics. Some libraries may also require you to change your Java version.

To upgrade to Java 8, navigate to your `build.common.gradle` file. Scroll down until you find this block:

```
compileOptions {
    sourceCompatibility JavaVersion.VERSION_1_7
    targetCompatibility JavaVersion.VERSION_1_7
}
```

Change the 7 to 8, like so:

```
compileOptions {
    sourceCompatibility JavaVersion.VERSION_1_8
    targetCompatibility JavaVersion.VERSION_1_8
}
```

Then, perform a Gradle sync.

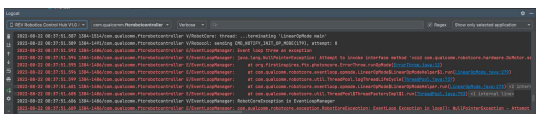
Android Debug Bridge

Note: On macOS, Linux, or using PowerShell you will have to change any commands that start with `adb` to start with `./adb` if you are in the `platform-tools` directory.

Logcat

Logcat is extremely useful for debugging issues with your code at runtime or figuring out what went wrong. For example, if your app activity crashes and you pull up the log seeing 5000 lines of the same error, there is probably infinite recursion in your code!

To use logcat, plug in your device (or connect via ADB). Then, select the app you want to view the logs for. Your window should look like this.



If you have an issue you don't understand, you can take a screenshot of the log or select and copy the error and ask a question in the FTC discord[177].

Wireless Communication

Android Debug Bridge (ADB) is a command-line tool that allows for wireless communication between the robot controller (phone or Control Hub).

ADB should come with the platform tools in Android Studio. Navigate to your `local.properties` file in the root of your project and you should see a path to the Android SDK on your computer, such as `C:\Users\Woodie\AppData\Local\Android\Sdk`. Then navigate to `platform-tools` and an application called adb should be there. To use it, open CLI (like PowerShell or command prompt) and run either `adb devices` or `./adb devices`.

For more information on ADB, you can look at the developers page[178].

Setting Up ADB

1. Ensure USB debugging is enabled on your device and it is in developer mode.
2. Make sure you have ADB installed. If you do not, follow the instructions at this link[179]

Note: You can use logcat via ADB with the `adb logcat` command. This is extremely useful for debugging as it allows you to look at the logs wirelessly which saves time. Remember, logcat is the *best* way to debug your software.

Add ADB To PATH

Adding variables to PATH:

- Windows[180]
- Linux/Unix (bash)[181]
- macOS (zsh)[182]

If you want to use ADB from any directory, add it to PATH. Follow an online tutorial for adding to PATH and set the PATH to the `platform-tools` directory. Once you do that, you can run ADB commands from anywhere on your system.

[177] https://discord.com/invite/first-tech-challenge
[178] https://developer.android.com/studio/command-line/adb
[179] https://www.xda-developers.com/install-adb-windows-macos-linux/
[180] https://docs.alfresco.com/content-services/latest/admin/troubleshoot/
[181] https://unix.stackexchange.com/questions/26047/how-to-correctly-add-a-path-to-path
[182] https://koenwoortman.com/zsh-add-directory-to-path/

Connecting to a Phone Wirelessly

1. Plug the robot controller phone into your computer.
2. Run the command `adb devices` in the `platform-tools` directory and see if the phone shows up.
3. Run `adb usb` and then `adb tcpip 5555`. You can then unplug the phone.
4. Connect to the same WiFi network the device is either hosting or on. The WiFi direct network created by the phone should be called "TEAMNUMBER-RC" or some small derivation of that. It may include extra letters if you have multiple devices per team. Refer to RS01 in Game Manual Part 1 for further details on the network naming scheme.
5. Connect to the phone using `adb connect 192.168.49.1:5555`. If this doesn't work, recheck the IP address of the phone and try again with that IP address if it is different.

Connecting to a Control Hub Wirelessly

1. Connect to the WiFi hotspot hosted by the Control Hub. The hotspot should be called "TEAMNUMBER-RC" or some small derivation of that. It may include extra letters if you have multiple devices per team. Refer to RS01 in Game Manual Part 1 for further details on the network naming scheme.
2. Once you're connected to a Control Hub's network, you simply need to connect to it using `adb connect 192.168.43.1:5555`.

Once a connection is established, it should appear in the device dropdown in Android Studio.

Wireless Configuration

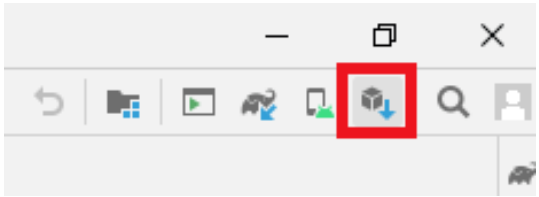
You can set up a configuration on the Driver Station or Robot Controller like usual. However, you can also create a valid configuration XML file in `TeamCode/src/main/res/xml`. You can find your configuration files in the `/sdcard/FIRST` folder as an XML file with the same name as the configuration.

To get these XML files wirelessly, you can use `adb pull /sdcard/FIRST/config.xml /fully/qualified/path/res/xml`.

If a valid configuration XML file is in `res/xml` it will show up as a configuration you can use for the robot when you push it to the Robot Controller or a Control Hub.

SDK Manager

You can find the SDK manager in the top right corner of your Android Studio.



Accepting Licenses

If you get a warning complaining about licenses not being accepted, follow these steps:

1. Go to the SDK manager and under SDK Platforms.
2. Select the version with the API level specified by the warning.
3. Click "Apply" and wait for the components to install.
4. Once this finishes, press "Finish," then "Ok." Wait for Android Studio to index if it is.
5. Restart Android Studio.

Installing SDK Tools

To install any SDK tools such as platform tools or build tools, open the SDK manager and go to SDK Tools. Select the tools you want to install and install them the same way you would for the SDK platforms.

Version Control

Version control is an extremely useful tool. It allows for looking at (and reverting to) previous versions of code, easy collaboration, having multiple versions of code that can be merged together, etc.

As far as version control systems go, we strongly recommend git, especially when used with a GUI like Android Studio's built in VCS tools or Github Desktop. While a git tutorial is out of scope for Game Manual 0, here are some git resources:

- The official git tutorial[183]
- GitHub's collection of git resources[184]
- GitHub's guide to installing git[185]
- GitHub Desktop, a git GUI[186]
- Android Studio's/IntelliJ's git integration documentation[187]

10.1.6 Common Issues

> **Warning:** Be careful what code you take as a reference from this page! Some of it is intentionally buggy to demonstrate potential easy-to-make errors.

Exceptions

Exceptions are events that occur during the execution of a program, disrupting the normal flow of instructions, used in error events or problems that arise during runtime. A exception can be catched to avoid propagation, otherwise any exception that's not handled will cause the program flow to stop immediately.

[183] https://git-scm.com/docs/gittutorial
[184] https://docs.github.com/en/get-started/quickstart/set-up-git
[185] https://github.com/git-guides/install-git
[186] https://desktop.github.com/
[187] https://www.jetbrains.com/help/idea/version-control-integration.html

Some common types of exceptions include:

- **NullPointerException**
  - It occurs when trying to call a method or getting a property of an object from a variable with a *null* value, which basically means that the variable doesn't hold a value *yet*, or the value doesn't exist.
  - This exception is one of the most common in FTC®, below is an example that throws a NullPointerException:

```
public class CrashyOpMode extends OpMode {

    // This call to the "get" method here will throw a NullPointerException.
    //
    // The value of the "hardwareMap" variable is null at this point, due to
    // the way the SDK is limited to define the value of this variable, its
    // value is defined right before the init() (or runOpMode() in LinearOpModes)
    // method is called.
    Servo clawServo = hardwareMap.get(Servo.class, "claw");

    @Override
    public void init() {
        // This statement won't ever be reached due to the
        // thrown NullPointerException, explained above,
        // since it happens before the OpMode starts execution.
        clawServo.setPosition(0.5);
    }

    // ...

}
```

  - This can be fixed moving the "Servo" variable value definition to the init (or `runOpMode()` in LinearOpModes) method as follows:

```
public class WorkingOpMode extends OpMode {

    Servo clawServo = null;

    @Override
    public void init() {
        // This won't throw a NullPointerException since the value of the
        // "hardwareMap" variable is defined at this point, but note that
        // the "get" method will return null if the name "claw" isn't
        // configured. (consult the "Using the SDK" section)
        clawServo = hardwareMap.get(Servo.class, "claw");

        // This statement should be reached and executed now.
        //
        // Note that if the "claw" servo is not configured, the value returned
        // by the hardwareMap will be null as explained before, therefore,
        // a NullPointerException would be thrown here if that happens.
        clawServo.setPosition(0.5);
    }

    // ...

}
```

- **TargetPositionNotSetException**

  - This exception type is a custom one from the SDK. It means you changed the motor RunMode to RUN_TO_POSITION before setting a target position:

```
// This will throw a "TargetPositionNotSetException" here!
motor.setRunMode(DcMotor.RunMode.RUN_TO_POSITION);

// And this statement won't be reached.
motor.setTargetPosition(1120);
```

  - It is fixed by simply switching the order of the statements; setting target position first, then changing the RunMode:

```
// Setting the target position first
motor.setTargetPosition(1120);

// Then switching the RunMode
motor.setRunMode(DcMotor.RunMode.RUN_TO_POSITION);
```

- **ArithmeticException**
  - Occurs when performing any illegal arithmetic operations such as dividing by zero:

```
int number = 128 / 0; // This will throw an ArithmeticException!
```

  - It can be handled by enclosing the code likely to throw this type of exception with a try catch block[188]:

```
int number; // Declaring the variable in the outside scope

try {
   // Giving it a value that will possibly throw an ArithmeticException
   number = 128 / 0;
} catch (ArithmeticException e) {
   // Do something when the ArithmeticException happens.
   // (The value of the "number" variable will remain 0)
}
```

- **InterruptedException**
  - It means that the SDK requested the OpMode to stop, and it's considered part of normal operation. An interrupt means that the current thread has been requested to end, so don't panic when you see a spam of those in *logcat* (page 272)!
  - If you call a method that possibly throws an InterruptedException (such as `Thread.sleep()`) it should be handled like this, with the try catch syntax mentioned before:

```
try {
   // Block for 500 milliseconds
   Thread.sleep(500);
} catch(InterruptedException e) {
   // Tells the current thread (OpMode) to
   // end the execution as soon as possible
   Thread.currentThread().interrupt();
}
```

  - Note that LinearOpMode already contains a shorthand sleep()[189] method that does this under

[188] https://www.w3schools.com/java/java_try_catch.asp
[189] https://github.com/OpenFTC/Extracted-RC/blob/f47d6f15fa1b59faaf509a522e0ec04f223ec125/RobotCore/src/main/java/com/qualcomm/robotcore/eventloop/opmode/LinearOpMode.java#L96

the hood. (And you shouldn't be using sleeps in OpMode since they're more strictly controlled. Read next sections for further information)

How the SDK handles exceptions

Except for InterruptedExceptions and some other internal special cases, which simply cause the OpMode to end, the FTC SDK performs an "emergency stop" routine when an exception is thrown and not handled properly. This halts the OpMode and displays the full stacktrace on screen. The stacktrace can also be viewed through *Logcat* (page 272) when using Android Studio.

Note: Before SDK 8.0, only the first line of the error would be displayed and selecting "Restart Robot" from the menu would be required before running an OpMode again.

It's generally a good idea to debug all OpModes extensively before any official match, as these exceptions are disruptive.

Stuck in start, loop, stop...

OpModes are *strictly controlled programs*, in the sense that the SDK requires them to flow in a certain way with the methods `init()`, `loop()`, etc. If you take more than a specific time (5 seconds, or 900 milliseconds in stop commands[190]) executing an action in any of these methods, the SDK will perform the "emergency stop" routine explained before, with the "stuck in `action`" error message.

```
public class StuckyOpMode extends OpMode {

   // ...

   @Override
   public void loop() {
       // Don't do this in a normal iterative OpMode!
       // This will cause a "stuck in stop" error after
       // 5 seconds, since iterative OpModes shouldn't
       // be blocked by loops or any lengthy operation.
       while(true) {
         // ...
       }
   }

}
```

If you need to run any sort of lenghty action in your OpMode, another option would be using a LinearOpMode instead.

LinearOpModes are less strict since their single `runOpMode()` method can flow more freely, but they still need to be cooperative to stop requests. Take the following code as an example:

```
public class StuckyLinearOpMode extends LinearOpMode {

    @Override
    public void runOpMode() {
```

(continues on next page)

[190] https://github.com/OpenFTC/Extracted-RC/blob/f47d6f15fa1b59faaf509a522e0ec04f223ec125/RobotCore/src/main/java/com/qualcomm/robotcore/eventloop/opmode/OpMode.java#L189

(continued from previous page)

```
        // Wait for the driver to press PLAY on the DS
        waitForStart();

        while(true) {
          // Do stuff infinitely
        }
    }

}
```

This code isn't cooperative to stop requests, since the `while` loop doesn't have an exit condition to cooperate with the OpMode stopping, therefore, this code will cause a "stuck in stop" error once it's stopped in the Driver Station.

To cooperate with the stopping of the OpMode, an `opModeIsActive()` or `!isStopRequested()` condition is required to be added to all the blocking loops executed in the `runOpMode()` method. Consult the *LinearOpMode vs OpMode* (page 257) page for more information about these methods.

An example for a cooperative LinearOpMode would be as follows:

```
public class CooperativeLinearOpMode extends LinearOpMode {

   @Override
   public void runOpMode() {
       while(someCondition && !isStopRequested()) {
           // Do something while the "someOtherCondition"
           // is true and the OpMode is not stopped.
       }

       // Wait for the driver to press PLAY on the DS
       waitForStart();

       while(someOtherCondition && opModeIsActive()) {
           // Do something while the "someCondition" is true
           // and the OpMode is running (started and not stopped).
       }
   }

}
```

10.2 Tutorials

Tutorials for common FTC® programs.

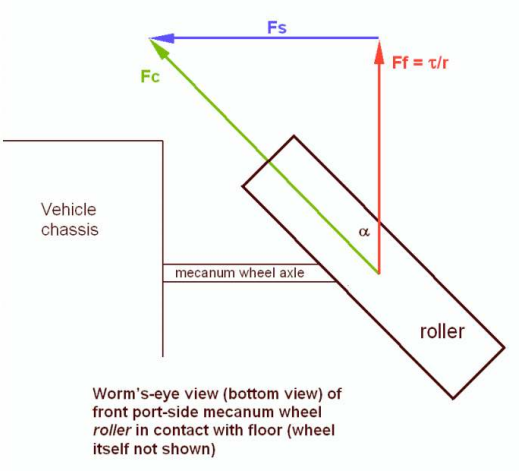
10.2.1 Mecanum TeleOp

Mecanum Physics

Mecanum drive is a very popular drivetrain type in FTC®. Mecanum drivetrains enables holonomic movement. This means that the drivetrain is able to move in any direction while rotating: forwards, backwards, side to side, translating while rotating, etc. Here is a neat video[191] demonstrating such movement.

Note: Some common COTS mecanum drivetrain kits are the goBILDA Strafer Chassis Kit[192] and the REV Mecanum Drivetrain Kit[193].

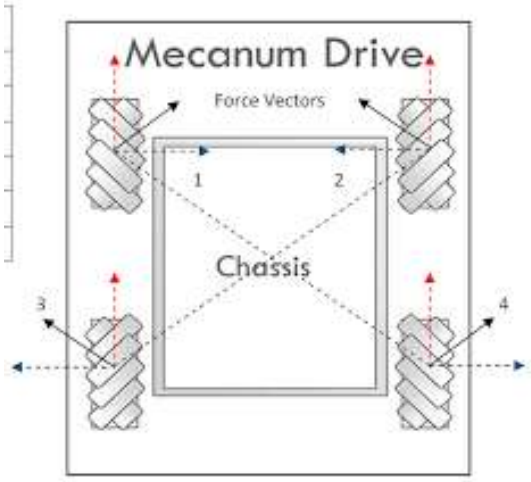
Mecanum wheels have rollers at a 45° angle to the rest of the wheel. Since these are in contact with the ground instead of something solid like in a *traction wheel*, instead of the wheel creating a force parallel to the orientation of the wheel, it creates one 45° from parallel. Depending on how the wheels are driven, X or Y components of the force vectors can cancel which allows movement in any direction.



Using Vectoring to Create Omnidirectional Movement

A standard mecanum drive configuration possesses 4 mecanum wheels oriented in an "X" shape. This means that the rollers are angled towards the center when looking at it from above. This configuration allows one to add up the force vectors generated by the offset rollers and derive movement in any direction. It is important to note that because of friction, perfect movement isn't possible in every direction, so a *mecanum drivetrain* will be able to drive slightly faster forwards/backwards than any other directions. Combining translation and rotation will also result in slower movement.

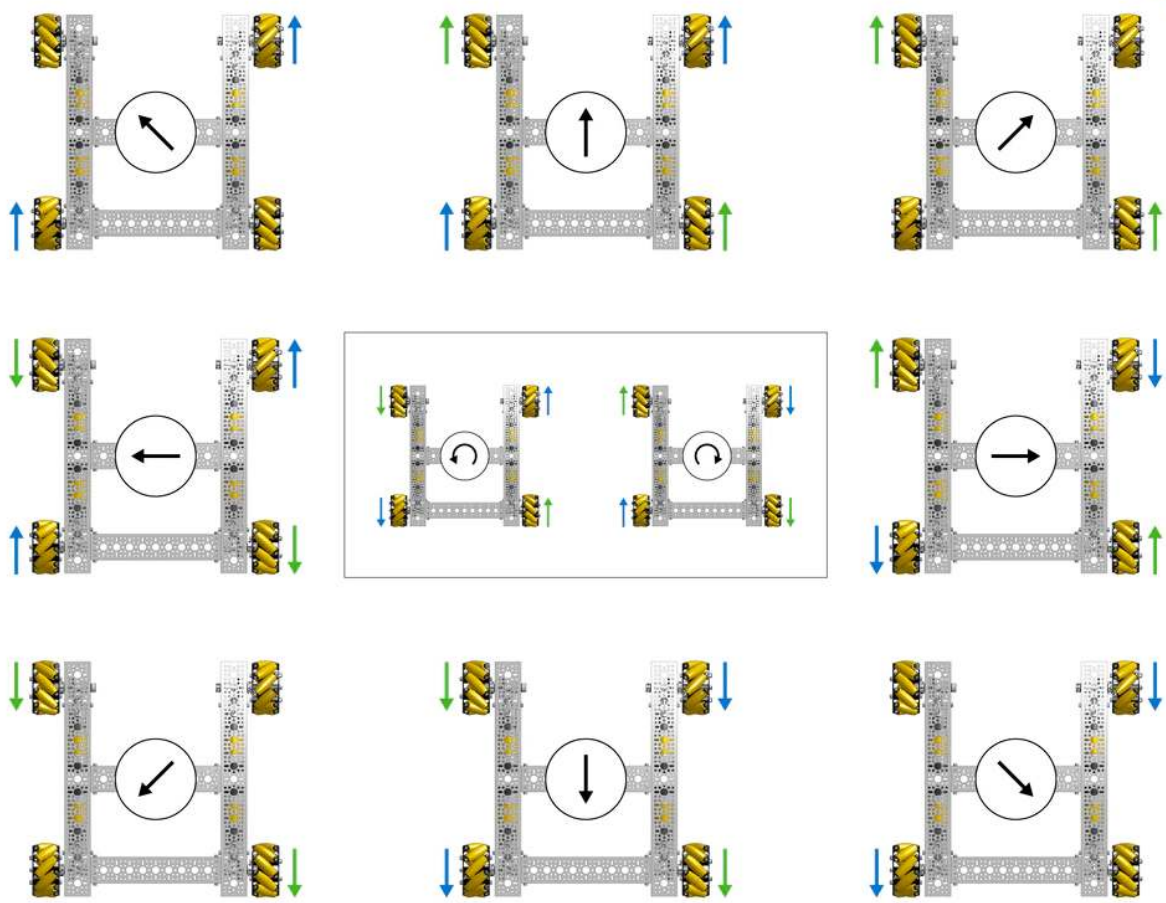
[191] https://www.youtube.com/watch?v=pP8ajNMx84k
[192] https://www.gobilda.com/strafer-chassis-kit-v5/
[193] https://www.revrobotics.com/rev-45-2470/



In the image above, vectors 1, 2, 3, and 4 are the force vectors created by the *mecanum wheels* when the chassis is instructed to drive towards the top of the image. All motors are driving forward. The blue and red lines are their X and Y components, respectively. Here are a few examples of how the wheels must be driven to achieve different movements:

goBILDA®
DREAM • MAKE • REPEAT

PRODUCT INSIGHTS



Attention: It is strongly advised to not hardcode these movements in; there is a much better way described below that allows for true holonomic movement and is much more elegant.

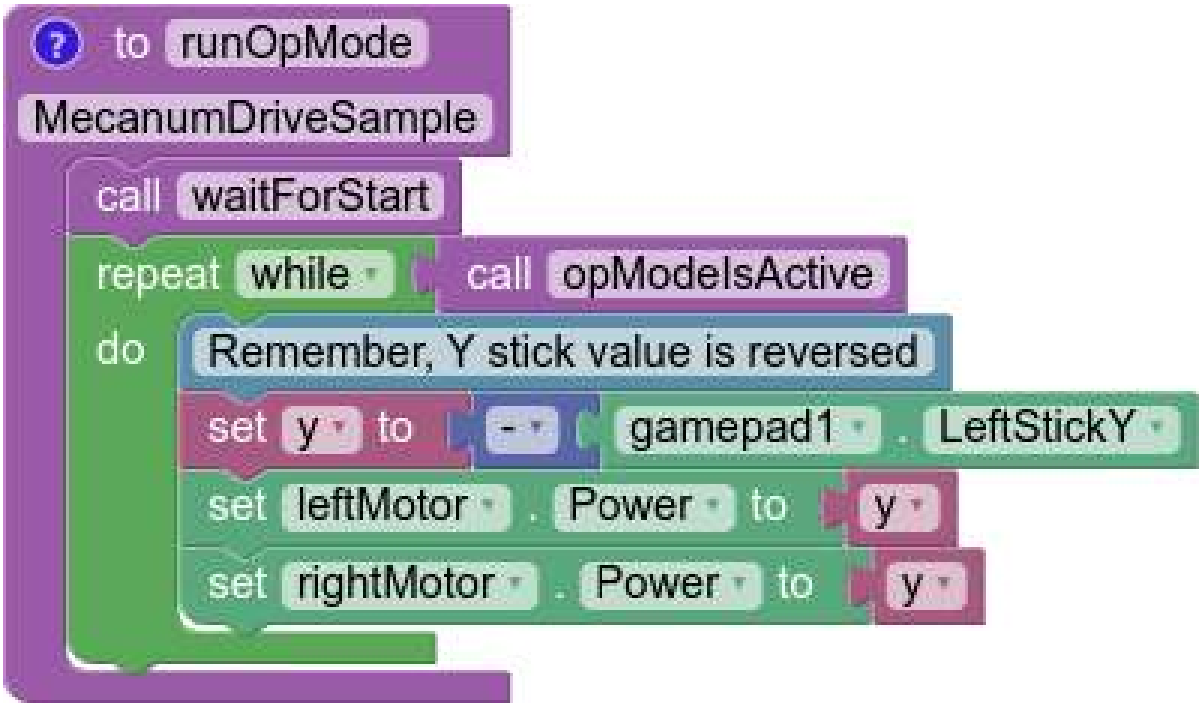
Deriving Mecanum Control Equations

Before thinking about mecanum, envision a scenario where you have a 2 motor tank drivetrain which you want to control using the left stick Y axis for forward/backward movement, and the right stick X axis for pivot turning. The motors are configured so that positive is clockwise for the right motor when the body is facing away from you, and the left motor is the opposite. To control only forward/backward movement, you simply need to set the motor powers to the Y stick value (flip the sign since Y is reversed):

Java

```
double y = -gamepad1.left_stick_y; // Remember, Y stick is reversed!

leftMotor.setPower(y);
rightMotor.setPower(y);
```

Blocks

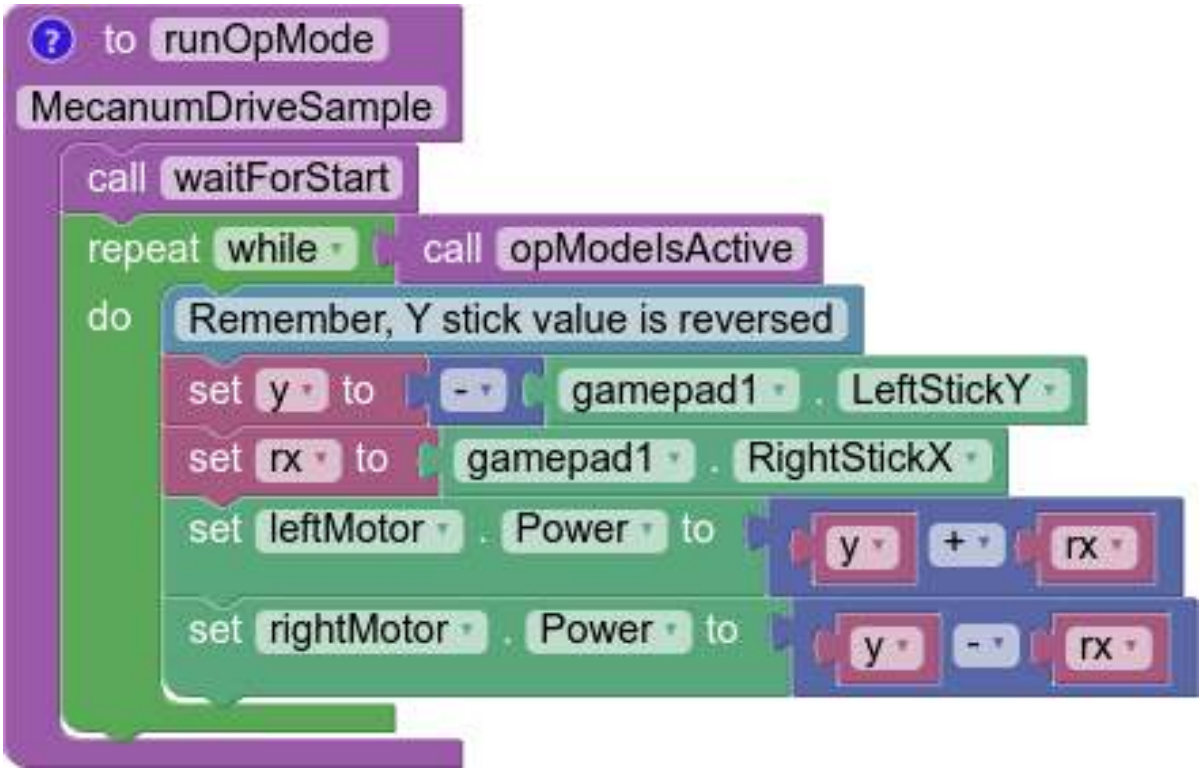


Although at first adding rotation might seem like a difficult task, it's actually super simple. All you need to do is subtract the right X stick value from the right wheels, and add it to the left:

Java

```
double y = -gamepad1.left_stick_y; // Remember, Y stick is reversed!
double rx = gamepad1.right_stick_x;

leftMotor.setPower(y + rx);
rightMotor.setPower(y - rx);
```

Blocks



Here, if the left stick is pressed upwards, both of the motors will be fed a positive value, causing the robot to move forward. If it is pressed downwards, both of the motors will be fed a negative value, causing the robot to move backwards. A similar principle applies for rotation: if the right stick is pushed rightward, the left wheels will spin forward while the right spin backward, causing rotation. The opposite applies for pushing the stick left. If both sticks are pushed at the same time, say the left Y stick is at 1 and the right X stick is also at 1, the value of the left wheels will be $1 + 1 = 2$ (which gets clipped to 1 in the SDK) and the right wheels will be $1 - 1 = 0$, which causes a rightward curve.

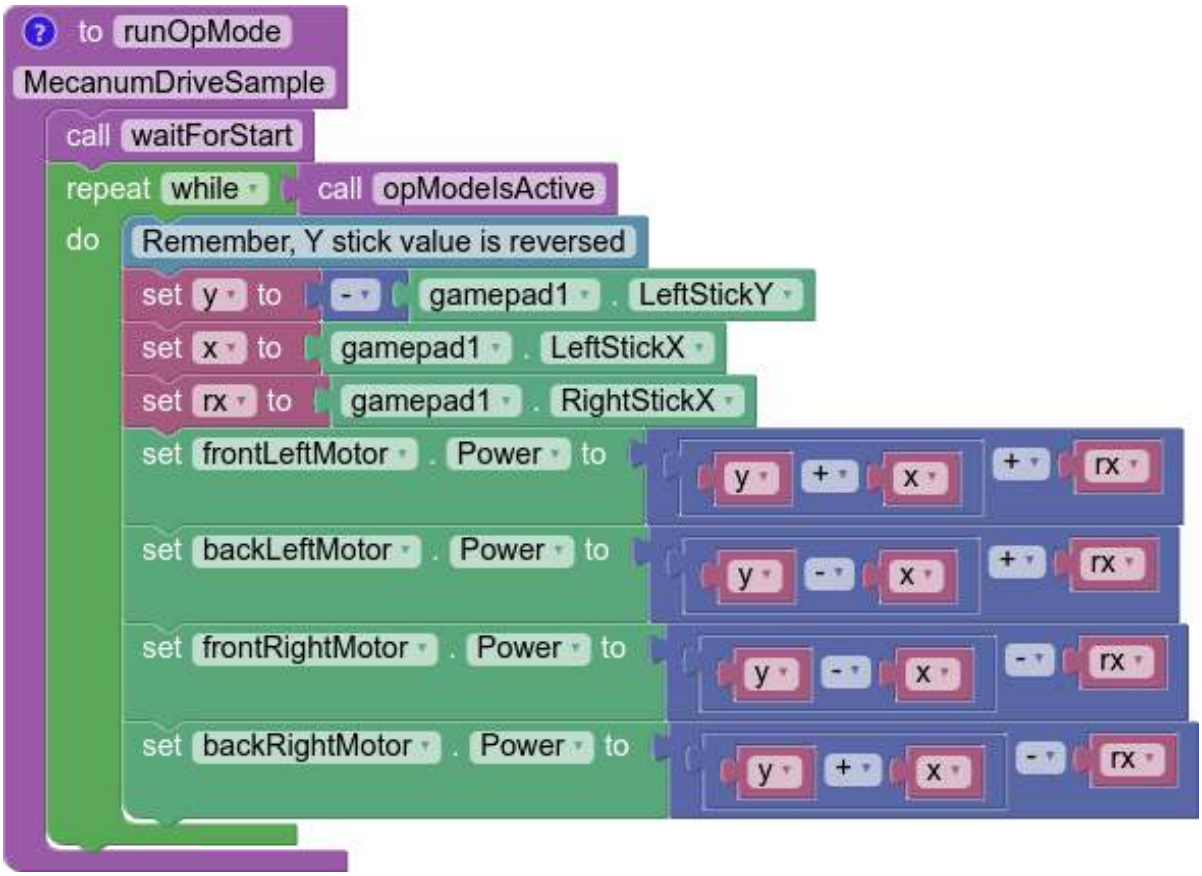
Applying omnidirectional movement with *mecanum wheels* operates under the same principle as adding turning into the tank example. The left stick X values will be added or subtracted to each wheel depending on how that wheel needs to rotate to get the desired movement. The only difference from turning is that rather than wheels on the same side being the same sign, wheels diagonal to each other will be the same sign.

We want a positive left stick X value to correlate to rightward strafing. If we refer back to the vectoring image, this means that the front left and back right need to rotate forward, while the back left and front right need to rotate backwards. So, we should add the x value to the front left and back right and subtract it from the back right and front left:

Java

```
double y = -gamepad1.left_stick_y; // Remember, Y stick is reversed!
double x = gamepad1.left_stick_x;
double rx = gamepad1.right_stick_x;

frontLeftMotor.setPower(y + x + rx);
backLeftMotor.setPower(y - x + rx);
frontRightMotor.setPower(y - x - rx);
backRightMotor.setPower(y + x - rx);
```

Blocks



Important: Most FTC motors spin counterclockwise when viewed from their face when given positive power by default, with the exception of NeveRests. If your drivetrain uses an even number of gears, this will reverse the direction the motors spin in.

On most drivetrains, you will need to reverse the left side for positive power to move forwards with most motors, and reverse the right side with NeveRests. The presence of gearing between the motor gearbox and the wheel may swap this, which is the case for the goBILDA Strafer and the REV Mecanum Drivetrain Kit.

This is the same as the tank example, except now with 4 motors and the strafing component added. Similarly to the tank example, the Y component is added to all wheels, and the right X (rx) is added to the left wheels and subtracted from the right. Now, we have added a left X component (x) that allows us to strafe rightward.

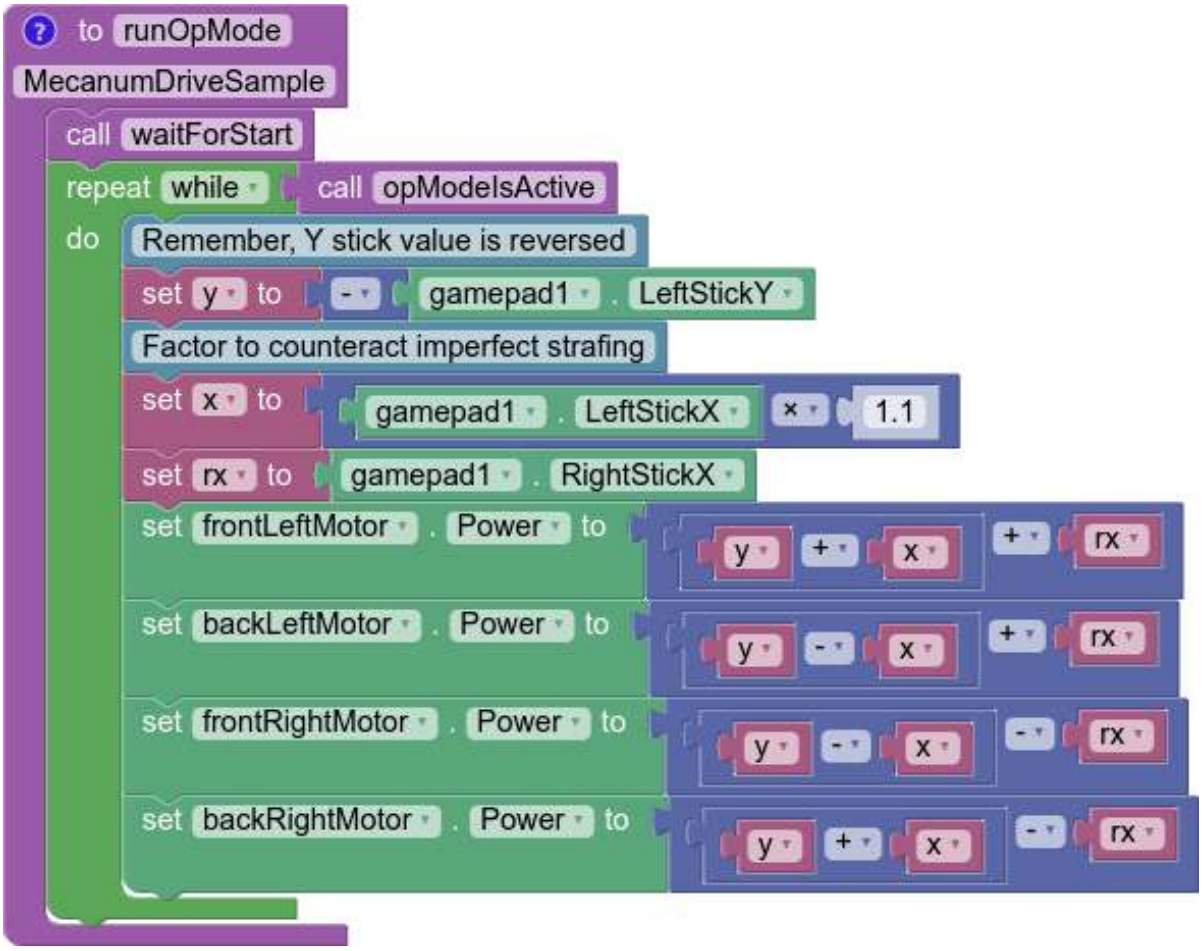
In doing that, however, we have actually allowed for strafing in any direction. If you think about it, pressing the left joystick to the left will do the same thing in reverse, which is what is needed to strafe left. If it is pressed at 45 degrees, the x and y components of the joystick will be equal. This will cause two diagonal motors to cancel, allowing for diagonal movement. This same effect applies to every angle of the joystick.

Now that we have a functioning mecanum driving program, there are a few things that can be done to clean it up. The first of these would be multiplying the left X value by something to counteract imperfect strafing. Doing this will make the drive feel more accurate on non axis aligned directions, and make field centric driving more accurate. In this tutorial, we will use 1.1, but it's really up to driver preference.

Java

```
double y = -gamepad1.left_stick_y; // Remember, Y stick is reversed!
double x = gamepad1.left_stick_x * 1.1; // Counteract imperfect strafing
double rx = gamepad1.right_stick_x;
```

Blocks



The other improvement we can make is scale the values into the range of -1 to 1.

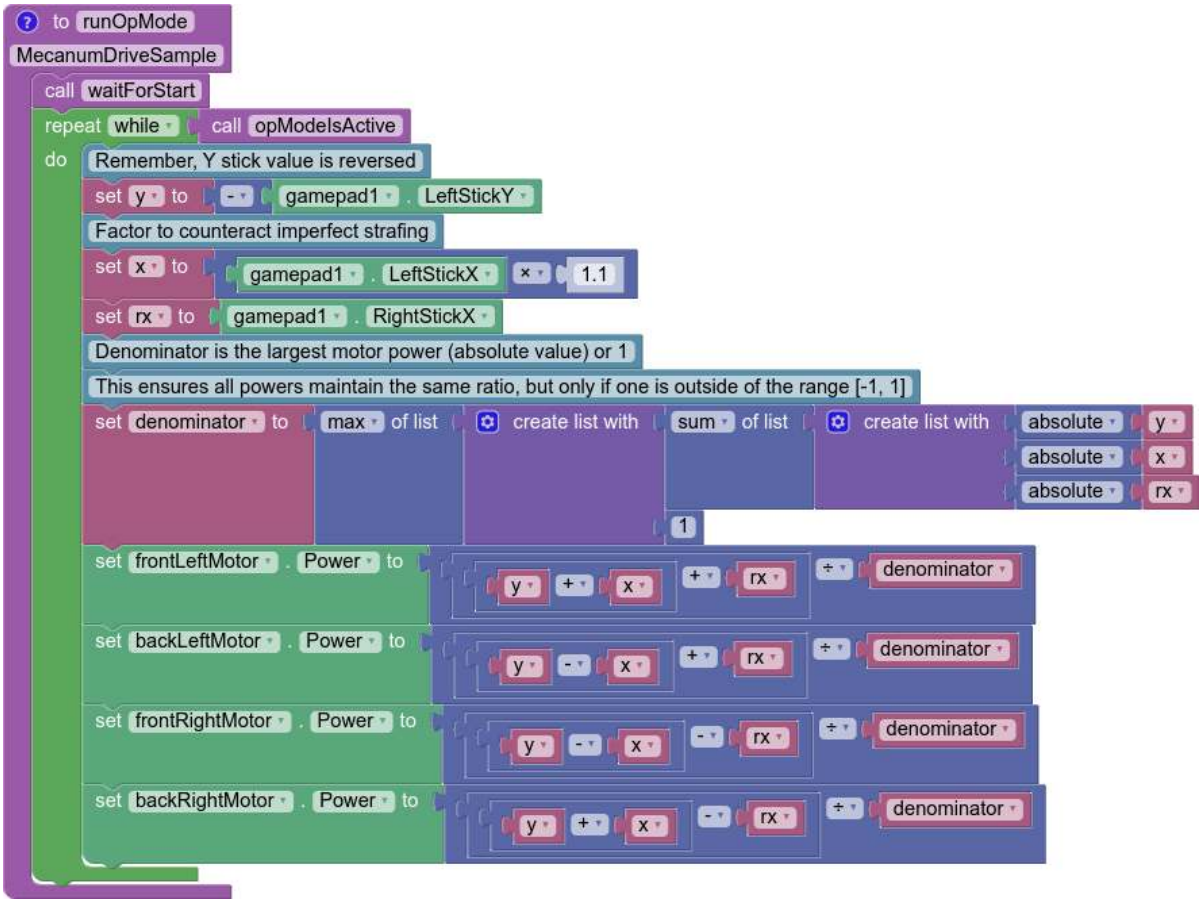
Since the SDK simply clips (limits) the powers to that range, we can lose the ratio we are looking for unless we proactively put all the numbers back in that range while still maintaining our calculated ratio. For example,

if we calculate values of 0.4, 0.1, 1.1, and 1.4, they will be clipped to 0.4, 0.1, 1.0, and 1.0, which is not the same ratio. Instead, we need to divide all of them by the largest power's absolute value when it exceeds 1:

Java

```
// Denominator is the largest motor power (absolute value) or 1
// This ensures all the powers maintain the same ratio, but only when
// at least one is out of the range [-1, 1]
double denominator = Math.max(Math.abs(y) + Math.abs(x) + Math.abs(rx), 1);
double frontLeftPower = (y + x + rx) / denominator;
double backLeftPower = (y - x + rx) / denominator;
double frontRightPower = (y - x - rx) / denominator;
double backRightPower = (y + x - rx) / denominator;
```

Blocks



Make sure to set the powers on your motor and update this every loop in an opmode!

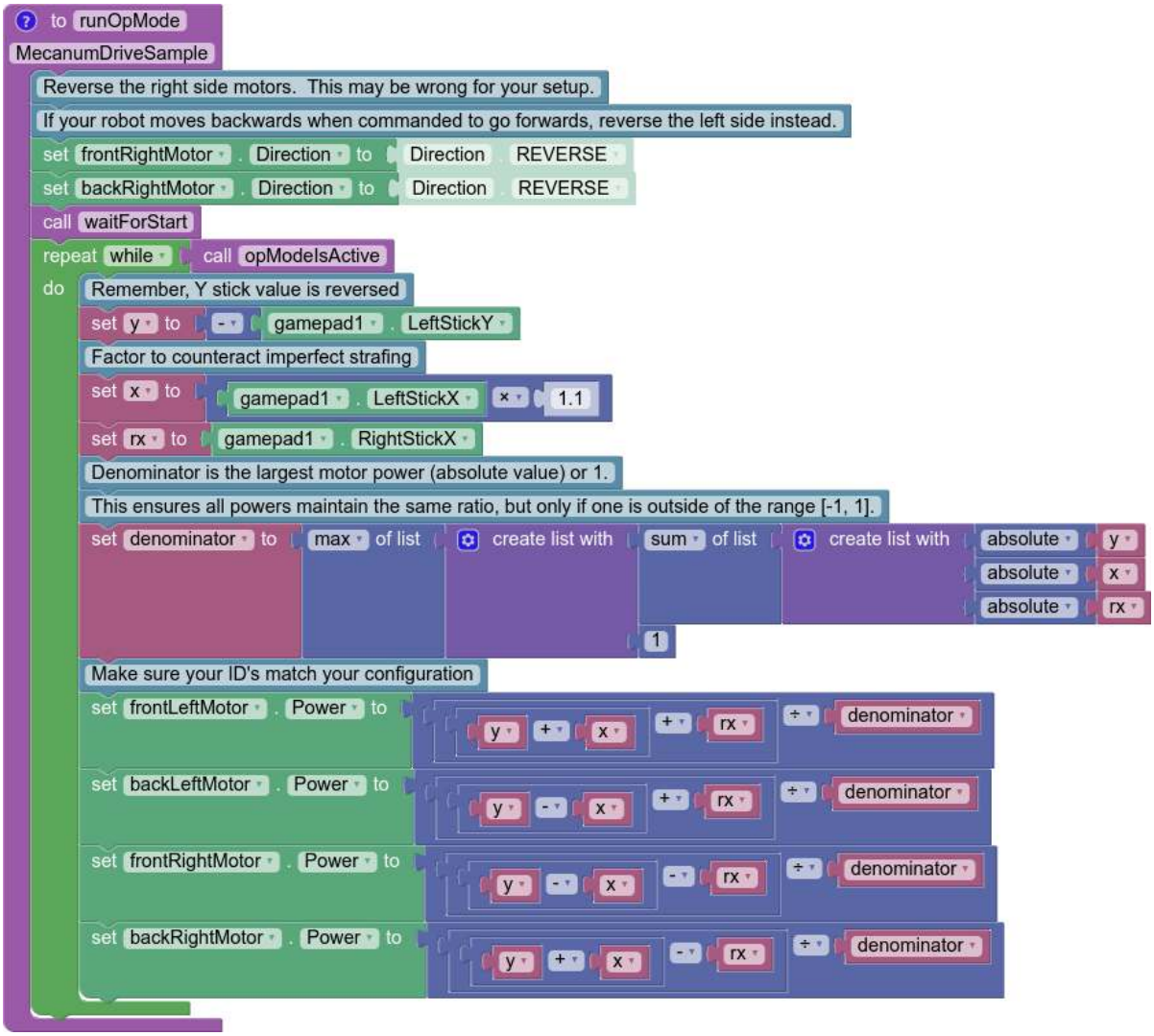
Robot-Centric Final Sample Code

Java

```
package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.DcMotorSimple;

@TeleOp
public class MecanumTeleOp extends LinearOpMode {
    @Override
    public void runOpMode() throws InterruptedException {
        // Declare our motors
        // Make sure your ID's match your configuration
        DcMotor frontLeftMotor = hardwareMap.dcMotor.get("frontLeftMotor");
        DcMotor backLeftMotor = hardwareMap.dcMotor.get("backLeftMotor");
        DcMotor frontRightMotor = hardwareMap.dcMotor.get("frontRightMotor");
        DcMotor backRightMotor = hardwareMap.dcMotor.get("backRightMotor");

        // Reverse the right side motors. This may be wrong for your setup.
        // If your robot moves backwards when commanded to go forwards,
        // reverse the left side instead.
        // See the note about this earlier on this page.
        frontRightMotor.setDirection(DcMotorSimple.Direction.REVERSE);
        backRightMotor.setDirection(DcMotorSimple.Direction.REVERSE);

        waitForStart();

        if (isStopRequested()) return;

        while (opModeIsActive()) {
            double y = -gamepad1.left_stick_y; // Remember, Y stick value is reversed
            double x = gamepad1.left_stick_x * 1.1; // Counteract imperfect strafing
            double rx = gamepad1.right_stick_x;

            // Denominator is the largest motor power (absolute value) or 1
            // This ensures all the powers maintain the same ratio,
            // but only if at least one is out of the range [-1, 1]
            double denominator = Math.max(Math.abs(y) + Math.abs(x) + Math.abs(rx),␣
↪1);
            double frontLeftPower = (y + x + rx) / denominator;
            double backLeftPower = (y - x + rx) / denominator;
            double frontRightPower = (y - x - rx) / denominator;
            double backRightPower = (y + x - rx) / denominator;

            frontLeftMotor.setPower(frontLeftPower);
            backLeftMotor.setPower(backLeftPower);
            frontRightMotor.setPower(frontRightPower);
            backRightMotor.setPower(backRightPower);
        }
    }
}
```

Blocks

`Blocks file download`



Field Centric

With field centric mecanum drive, the translation joystick controls the direction of the robot relative to the field, as opposed to the robot frame. This is preferred by some drivers, and make some evasive action easier, as one can spin while translating in a given direction easier. To do this, the x/y components of the joysticks are rotated counter to the robot's angle, which is given by the IMU.

There is an IMU inside of Control Hubs (and older models of Expansion Hubs). Unlike most other hardware, it is recommended to do more than `hardwareMap.get()` to begin using it. Note, this is configured when creating a new configuration by default as `imu`. See the FTC doc page covering the IMU interface and its parameters[194] for more information. The way the IMU will be initialized here is:

[194] https://ftc-docs.firstinspires.org/programming_resources/imu/imu.html

```
// Retrieve the IMU from the hardware map
imu = hardwareMap.get(IMU.class, "imu");
// Adjust the orientation parameters to match your robot
IMU.Parameters parameters = new IMU.Parameters(new RevHubOrientationOnRobot(
        RevHubOrientationOnRobot.LogoFacingDirection.UP,
        RevHubOrientationOnRobot.UsbFacingDirection.FORWARD));
// Without this, the REV Hub's orientation is assumed to be logo up / USB forward
imu.initialize(parameters);
```

The angle needs to be read every loop. In addition to this, while the IMU keeps a consistent zero position between OpModes (relevantly, including between autonomous and teleop), adding a bind to reset the angle is important to counteract drift and because the zero can change due to some types of disconnects.

Note: BNO055 objects will reset the IMU zero when `initialize` is called. The BNO055 class is not recommended for new development. The IMU class does not have this behavior, and is the appropriate replacement as of SDK v8.1.

```
// This button choice was made so that it is hard to hit on accident,
// it can be freely changed based on preference.
// The equivalent button is start on Xbox-style controllers.
if (gamepad1.options) {
    imu.resetYaw();
}

double botHeading = imu.getRobotYawPitchRollAngles().getYaw(AngleUnit.RADIANS);
```

Then, the translation joystick values need to be counterrotated by the robot heading. The IMU returns heading, however we need to rotate the movement counter to the robot's rotation, so its negative is taken. The joystick values are a vector, and rotating a vector in 2D requires this formula (proved here[195]), where x_1 and y_1 are the components of the original vector, β is the angle to rotate by, and x_2 and y_2 are the components of the resultant vector.

$$x_2 = x_1 cos\beta - y_1 sin\beta$$
$$y_2 = x_1 sin\beta + y_1 cos\beta$$

```
// Rotate the movement direction counter to the bot's rotation
double rotX = x * Math.cos(-botHeading) - y * Math.sin(-botHeading);
double rotY = x * Math.sin(-botHeading) + y * Math.cos(-botHeading);
```

Then, these rotated values can be put into the mecanum kinematics shown earlier.

```
double denominator = Math.max(Math.abs(rotY) + Math.abs(rotX) + Math.abs(rx), 1);
double frontLeftPower = (rotY + rotX + rx) / denominator;
double backLeftPower = (rotY - rotX + rx) / denominator;
double frontRightPower = (rotY - rotX - rx) / denominator;
double backRightPower = (rotY + rotX - rx) / denominator;
```

[195] https://matthew-brett.github.io/teaching/rotation_2d.html

Field-Centric Final Sample Code

```
package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.hardware.IMU;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;
import com.qualcomm.robotcore.hardware.DcMotorSimple;
import com.qualcomm.hardware.rev.RevHubOrientationOnRobot;
import org.firstinspires.ftc.robotcore.external.navigation.AngleUnit;

@TeleOp
public class FieldCentricMecanumTeleOp extends LinearOpMode {
    @Override
    public void runOpMode() throws InterruptedException {
        // Declare our motors
        // Make sure your ID's match your configuration
        DcMotor frontLeftMotor = hardwareMap.dcMotor.get("frontLeftMotor");
        DcMotor backLeftMotor = hardwareMap.dcMotor.get("backLeftMotor");
        DcMotor frontRightMotor = hardwareMap.dcMotor.get("frontRightMotor");
        DcMotor backRightMotor = hardwareMap.dcMotor.get("backRightMotor");

        // Reverse the right side motors. This may be wrong for your setup.
        // If your robot moves backwards when commanded to go forwards,
        // reverse the left side instead.
        // See the note about this earlier on this page.
        frontRightMotor.setDirection(DcMotorSimple.Direction.REVERSE);
        backRightMotor.setDirection(DcMotorSimple.Direction.REVERSE);

        // Retrieve the IMU from the hardware map
        IMU imu = hardwareMap.get(IMU.class, "imu");
        // Adjust the orientation parameters to match your robot
        IMU.Parameters parameters = new IMU.Parameters(new RevHubOrientationOnRobot(
                RevHubOrientationOnRobot.LogoFacingDirection.UP,
                RevHubOrientationOnRobot.UsbFacingDirection.FORWARD));
        // Without this, the REV Hub's orientation is assumed to be logo up / USB␣
↪forward
        imu.initialize(parameters);

        waitForStart();

        if (isStopRequested()) return;

        while (opModeIsActive()) {
            double y = -gamepad1.left_stick_y; // Remember, Y stick value is reversed
            double x = gamepad1.left_stick_x;
            double rx = gamepad1.right_stick_x;

            // This button choice was made so that it is hard to hit on accident,
            // it can be freely changed based on preference.
            // The equivalent button is start on Xbox-style controllers.
            if (gamepad1.options) {
                imu.resetYaw();
            }

            double botHeading = imu.getRobotYawPitchRollAngles().getYaw(AngleUnit.
↪RADIANS);
```

(continues on next page)

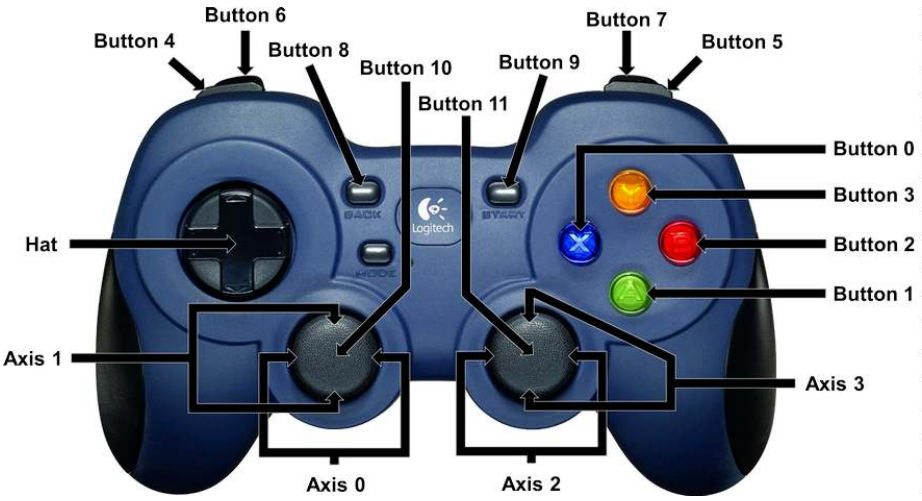
(continued from previous page)

```
            // Rotate the movement direction counter to the bot's rotation
            double rotX = x * Math.cos(-botHeading) - y * Math.sin(-botHeading);
            double rotY = x * Math.sin(-botHeading) + y * Math.cos(-botHeading);

            rotX = rotX * 1.1;  // Counteract imperfect strafing

            // Denominator is the largest motor power (absolute value) or 1
            // This ensures all the powers maintain the same ratio,
            // but only if at least one is out of the range [-1, 1]
            double denominator = Math.max(Math.abs(rotY) + Math.abs(rotX) + Math.
↪abs(rx), 1);
            double frontLeftPower = (rotY + rotX + rx) / denominator;
            double backLeftPower = (rotY - rotX + rx) / denominator;
            double frontRightPower = (rotY - rotX - rx) / denominator;
            double backRightPower = (rotY + rotX - rx) / denominator;

            frontLeftMotor.setPower(frontLeftPower);
            backLeftMotor.setPower(backLeftPower);
            frontRightMotor.setPower(frontRightPower);
            backRightMotor.setPower(backRightPower);
        }
    }
}
```

10.2.2 Gamepad Usage

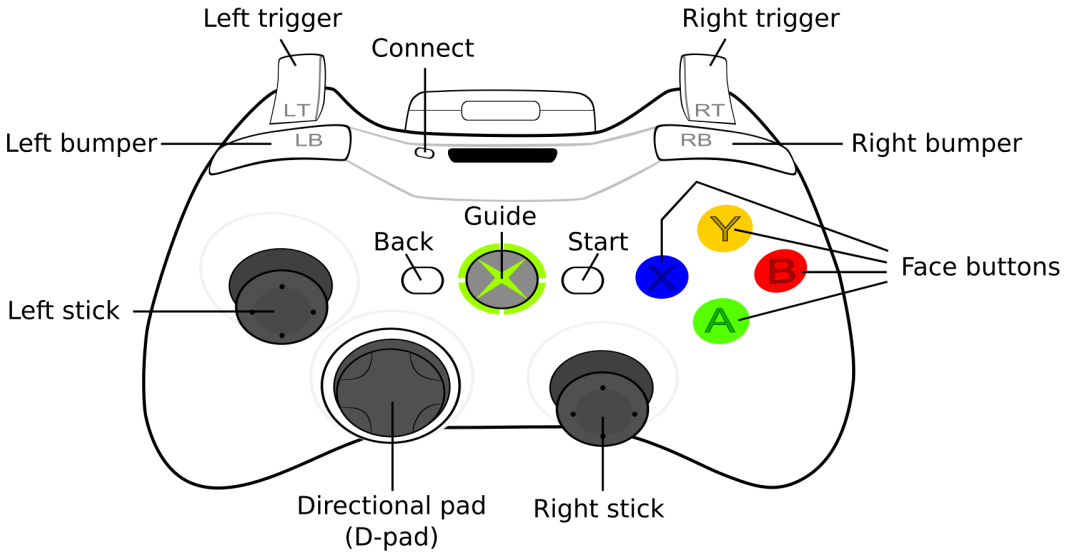
Gamepad Layout

Logitech F310

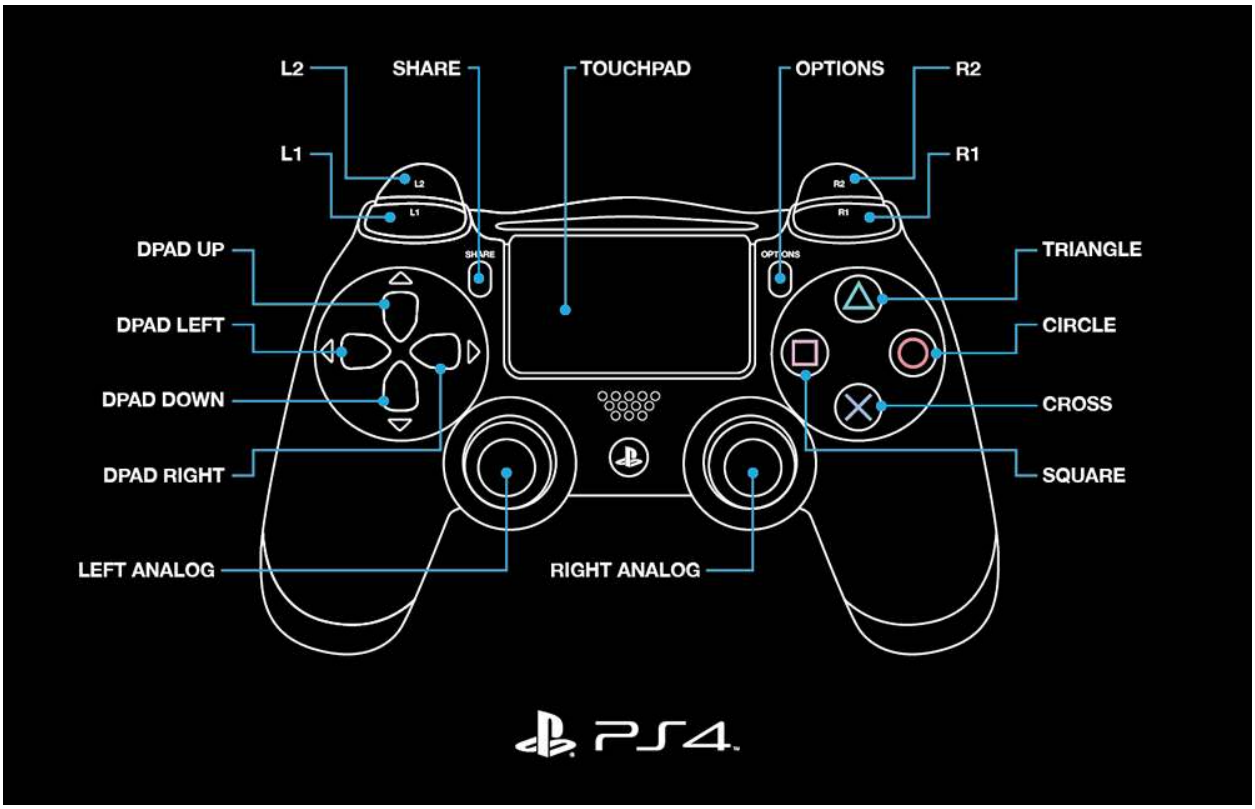


Axis 0	Left Joystick X
Axis 1	Left Joystick Y (inv)
Axis 2	Right Joystick X
Axis 3	Right Joystick Y (inv)
Button 0	X
Button 1	A
Button 2	B
Button 3	Y
Button 4	Left Bumper
Button 5	Right Bumper
Button 6	Left Trigger
Button 7	Right Trigger
Button 8	Back
Button 9	Start
Button 10	Left Joystick (click)
Button 11	Right Joystick (click)
Hat	Directional Pad (X, Y)

Xbox 360



PS4/Etpark



Note: L1 is `left_bumper`, L2 is `left_trigger`, R1/R2 are the right-hand equivalents.

Button Aliases

Since both PS4-style and Xbox-Style controllers are FTC® legal, there are aliases in the FTC SDK[196] between PS4-style and Xbox-style button naming.

PS4	Xbox
circle	b
cross	a
triangle	y
square	x
share	back
options	start
ps	guide

Boolean Inputs

TeleOp OpModes are generally written in iterative style, with a loop that contains code that is called over an over. Under this paradigm, a simple handling of user input could look like

```
if (gamepad1.a) {
    motor.setPower(1);
}
else {
    motor.setPower(0);
}
```

In this situation, this likely does what the driver wants: while the button is held, the motor's power is set to 1, and otherwise the power is set to 0. Since writing the same power to a motor multiple times has no effect on the motor's behavior, this works perfectly fine. However, issues arise when wanting to do something once when a button is pressed. For example, it is tempting to write something like this to get a press of a or b to adjust a servo.

```
if (gamepad1.a) {
    servo.setPosition(servo.getPosition()+0.1);
}
else if (gamepad1.b) {
    servo.setPosition(servo.getPosition()-0.1);
}
```

However, this will behave unpredictably, as each time the button is pressed, the `setPosition` method will be called multiple times as the loop frequency changes as does the button press length. There are a few techniques to avoid this, however they all require comparing the gamepad state to the gamepad state in the previous loop; therefore, it is necessary to store it.

[196] https://github.com/OpenFTC/Extracted-RC/blob/c960dd7de34d49a66c00a345636175392f936b9e/RobotCore/src/main/java/com/qualcomm/robotcore/hardware/Gamepad.java#L892

Storing Gamepad State

While each gamepad input's previous state could be stored individually in a variable, e.g. `boolean previousA`, this very quickly gets annoying. Luckily, the FTC SDK provides a way to copy gamepad states, with `gamepad.copy(gamepadToCopy)`.

Note: In addition to storing the gamepad state for the previous iteration of the loop, the gamepad state for the current iteration of the loop is also stored. This is necessary because if the state of a button was always read from `gamepad1`/`gamepad2`, it could change between reading the value and storing the previous value. This is because `gamepad1`/`gamepad2` update concurrently for `LinearOpMode`, and so can change during a loop iteration.

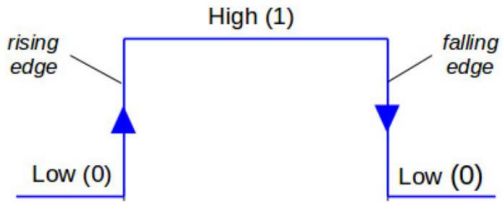
In a `LinearOpMode` based TeleOp program, storing both current and previous gamepad state could look like:

```
public void runOpMode() {
    // By setting these values to new Gamepad(), they will default to all
    // boolean values as false and all float values as 0
    Gamepad currentGamepad1 = new Gamepad();
    Gamepad currentGamepad2 = new Gamepad();

    Gamepad previousGamepad1 = new Gamepad();
    Gamepad previousGamepad2 = new Gamepad();

    // other initialization code goes here

    while (opModeIsActive()) {
        // Store the gamepad values from the previous loop iteration in
        // previousGamepad1/2 to be used in this loop iteration.
        // This is equivalent to doing this at the end of the previous
        // loop iteration, as it will run in the same order except for
        // the first/last iteration of the loop.
        previousGamepad1.copy(currentGamepad1);
        previousGamepad2.copy(currentGamepad2);

        // Store the gamepad values from this loop iteration in
        // currentGamepad1/2 to be used for the entirety of this loop iteration.
        // This prevents the gamepad values from changing between being
        // used and stored in previousGamepad1/2.
        currentGamepad1.copy(gamepad1);
        currentGamepad2.copy(gamepad2);

        // Main teleop loop goes here
    }
 }
```

Rising Edge Detector

Why is it called a rising edge detector?

A signal edge is a transition in a digital signal. In this case, the digital signal is the gamepad input, which is low when not held and high when held. The rising edge is the transition from low to high, and the falling edge is the transition from high to low.



The most commonly used technique is a rising edge detector. It allows code to be run only once when the button is initially pressed, as opposed to every loop while it is held down. It works by checking that the button is currently pressed, but was not pressed in the previous loop. For example, inside of a TeleOp loop:

```
if (currentGamepad1.a && !previousGamepad1.a) {
    servo.setPosition(servo.getPosition() + 0.1);
}
```

This will increase the servo position by 0.1 exactly once per press of a.

Falling Edge Detector

A very similar technique is a falling edge detector. It allows code to be run only once when the button is released, as opposed to every loop while it is held down. It works by checking that the button is currently not pressed, but was pressed in the previous loop. For example, inside of a TeleOp loop:

```
if (!currentGamepad1.b && previousGamepad1.b) {
    servo.setPosition(servo.getPosition() - 0.1);
}
```

This will decrease the servo position by 0.1 exactly once per release of b.

Note: One button can run different code on the rising and falling edge. This is mainly useful for fairly complex interactions and so is not demonstrated here.

Toggles

One common use case for rising edge detectors is to control toggles. Toggles can be used to have a button for the robot to switch between states; for example, turning an intake on and off. This can be done for any number of states but is most commonly done between two. To make a toggle between two states, a rising edge detector is used to set a boolean to its opposite and then that boolean is used to control an action.

Example

Within the initialization code:

```
boolean intakeToggle = false;
```

Inside of the corresponding TeleOp loop:

```
// Rising edge detector
if (currentGamepad1.a && !previousGamepad1.a) {
    // This will set intakeToggle to true if it was previously false
    // and intakeToggle to false if it was previously true,
    // providing a toggling behavior.
    intakeToggle = !intakeToggle;
}

// Using the toggle variable to control the robot.
if (intakeToggle) {
    intakeMotor.setPower(1);
}
else {
    intakeMotor.setPower(0);
}
```

This will turn on the intake when a is pressed, and leave it on until it is pressed again.

Note: The less a driver has to keep in their head about the state of the robot the less they can screw up. Since toggles mean that a button does different things every time it is pressed, consider alternate solutions. This is especially true for toggles with more than two states.

Gamepad Feedback

Gamepad feedback (i.e. rumble and LED control) can be a helpful way for robots to communicate status to a driver during a match. The degree to which the legal gamepads support this functionality varies:

Logitech F310

- Rumble: none
- LED Control: none

Xbox 360

- Rumble: large (whomp whomp) and small (bzzz)
- LED Control: none

DualShock4 (PS4)

- Rumble: large (whomp whomp) and small (bzzz)
- LED Control: control of RGB lightbar (solid color or pattern)

EtPark

- Rumble: contains both left and right rumble motors, but both seem to be only small weight (bzzz)
- LED Control: control of RGB LED (solid color or pattern). LED is fairly small and dim and may not be a good choice.

Tip: Gamepad feedback can be used to alert drivers of: start of endgame, intake loaded, automatic aligment complete, etc.

Rumble

The SDK provides both a simple and more complex API for controlling rumble according to the desired use case.

Note:

- Rumble power is specified as a floating-point value in the range [0.0, 1.0].
- Rumble duration is specified in milliseconds as an integer. The constant `Gamepad.RUMBLE_DURATION_CONTINUOUS` may be used to indicate that the rumble should continue until another rumble action is commanded.

Note: All rumble actions are completed asynchronously; i.e. the function calls will return immediately. Any call to a rumble API will immediately displace any currently running rumble action. That is, if you command a gamepad to rumble for 750ms and then immediately command a rumble for 250ms, the gamepad will rumble for 250ms from the time the second command was issued.

Simple API

The simplest way to command rumble: rumble motor 1 at 100% power for a specified duration:

```
gamepad1.rumble(int durationMs);
```

If control over both rumble motors and rumble intensity is desried:

```
gamepad1.rumble(double rumble1, double rumble2, int durationMs);
```

To make a gamepad rumble for a certain number of “blips” (the notion of what a “blip” is being predefined by the SDK):

```
gamepad1.rumbleBlips(int numBlips);
```

Helper functions:

The `public boolean isRumbling()` function provides an educated guess about whether there is a rumble action ongoing on this gamepad. The Robot Controller does not know for sure whether a rumble action is ongoing or not, because once the command is sent to the Driver Station, the Driver Station handles running the gamepad effects and the Robot Controller is "hands off".

The `public void stopRumble()` function may be used to stop any ongoing rumble action for a gamepad (perhaps most useful in conjunction with a rumble of continuous duration).

Advanced API

To create more advanced rumble behavior, a `RumbleEffect` may be created, which is composed of "Steps" which specify the power and duration each rumble motor should operate at. When a gamepad is commanded to run a `RumbleEffect`, it will perform each of the "Steps" in series.

To create a `RumbleEffect`, the `RumbleEffect.Builder` class must be used. The builder provides the `addStep(double rumble1, double rumble2, int durationMs)` function which is used to add a step to the sequence, and the `build()` function to create a `RumbleEffect` from the sequence of steps.

Using an anonymous instance of the builder class is the cleanest way to construct a `RumbleEffect`, for example:

```
Gamepad.RumbleEffect effect = new Gamepad.RumbleEffect.Builder()
       .addStep(0.0, 1.0, 500)  //  Rumble right motor 100% for 500 mSec
       .addStep(0.0, 0.0, 300)  //  Pause for 300 mSec
       .addStep(1.0, 0.0, 250)  //  Rumble left motor 100% for 250 mSec
       .addStep(0.0, 0.0, 250)  //  Pause for 250 mSec
       .addStep(1.0, 0.0, 250)  //  Rumble left motor 100% for 250 mSec
       .build();
```

Once a `RumbleEffect` has been created, it can be sent to a gamepad by calling:

```
gamepad1.runRumbleEffect(effect);
```

LED Control

Note:

- RGB LED component (i.e. red, green, blue) intensity is specified as a floating-point value in the range [0.0, 1.0].
- LED duration is specified in milliseconds as an integer. The constant `Gamepad.LED_DURATION_CONTINUOUS` may be used to indicate that the LED should remain the specified color until another command is issued.

Note: All LED actions are completed *asynchronously;* i.e. the function calls will return immediately. Any call to an LED API will immediately displace any currently running LED action. That is, if you command the

LED green for 750ms and then immediately command purple for 250ms, the LED will light purple for 250ms from the time the second command was issued.

To set the LED color for a fixed duration:

```
gamepad1.setLedColor(double r, double g, double b, int durationMs);
```

To create more advanced LED behavior, an `LedEffect` may be created, which is composed of "Steps" which specify a color and the duration for which to maintain it. When a gamepad is commanded to run an `LedEffect`, it will perform each of the "Steps" in series.

To create an `LedEffect`, the `LedEffect.Builder` class must be used. The builder provides the `addStep(double r, double g, double b, int durationMs)` function which is used to add a step to the sequence, and the `build()` function to create an `LedEffect` from the sequence of steps.

Using an anonymous instance of the builder class is the cleanest way to construct an `LedEffect`, for example:

```
Gamepad.LedEffect rgbEffect = new Gamepad.LedEffect.Builder()
       .addStep(1, 0, 0, 250) // Show red for 250ms
       .addStep(0, 1, 0, 250) // Show green for 250ms
       .addStep(0, 0, 1, 250) // Show blue for 250ms
       .addStep(1, 1, 1, 250) // Show white for 250ms
       .build();
```

Once an `LedEffect` has been created, it can be sent to a gamepad by calling:

```
gamepad1.runLedEffect(rgbEffect);
```

10.2.3 Using Telemetry

The SDK has "telemetry" that can be sent from the robot controller to the drivers station to display basic text. This telemetry is accessible through the `Telemetry` class, and the `telemetry` variable accessible in both OpMode and LinearOpMode.

Building Telemetry

There are two main methods used to add things to telemetry. `addData()` takes two parameters, a string for the caption, and then a value that can be any object. This is then printed on the phone screen in the format `caption : value`. These lines are keyed, so adding a secondary `addData()` with the same caption will override the value set.

```
telemetry.addData("Caption 1", 2.5);
telemetry.addData("Caption 2", "value");
```

The second method used to add things to telemetry is `addLine()`. Add line can be used to add a line with no parameters, or can take a single string to add as a line.

```
telemetry.addLine("This is a line!");
```

Updating Telemetry

Once all telemetry items have been added (typically at the end of an opmode loop), `telemetry.update()` must be called. Calling this method is what pushes the values to the phone, so without it telemetry will not appear on the phone screen. Note that `OpMode` will call `telemetry.update()` automatically once per call of its `loop`.

Tip: By default, telemetry is only refreshed on the phone every 250 ms. Any calls to `telemetry.update()` during this window will be saved and overwritten if update() is called again before 250 ms have elapsed. `setMsTransmissionInterval()` may be used to change the amount of time the SDK will wait between sending updates to the driver station.

10.2.4 Encoders

What Are Encoders?

Very commonly in FTC®, you want to know where something is. Whether that is how many times your drivetrain wheel has rotated, what angle your arm is at, or how far your string slides have gone, rotational encoders can help you. In FTC, an encoder refers to any sensor that tracks the rotational angle of a mechanism.

There are two types of encoders commonly used in FTC, relative and absolute encoders. Relative encoders are the ones covered here as they are the more common type.

Relative Encoders

Ranging from the built in encoder in every FTC legal motor to common external encoders like the REV Through Bore encoder, these encoders track the relative position of the shaft or mechanism they are attached to. **What this means is that the position output is relative to the initial position at robot power on, meaning position information is lost between power cycles**.

Tip: Relative encoders don't reset to zero at the beginning of OpModes! You can use STOP_AND_RESET_ENCODERS to ensure that your encoders are always at zero at the beginning of an OpMode (see below).

All relative encoders in FTC use the "quadrature" protocol to send position information to the expansion hub. As a result, relative encoders must be plugged into the encoder ports located near the motor ports in order to function.

Terminology

Count: A "count" (sometimes referred to as a "tick") refers to one increment of the encoder's position. Relative encoders report their position as a number equal to the number of "ticks" or "counts" the encoder has moved from its starting angle.

Counts Per Revolution: The number of "counts" that an encoder reports after it has gone exactly one revolution. This value is commonly used to convert encoder "counts" into degrees or revolutions.

Warning: Quadrature terminology can get very confusing! Some encoders may report "pulses per revolution." One pulse can either equal 1 or 4 counts. Other encoders may report "cycles per revolution", which confusingly have the same acronym as counts per revolution. The best way to check is to plug the encoder into the REV Hub and turn it 1 full revolution, then check what it reports.

Programming Encoders

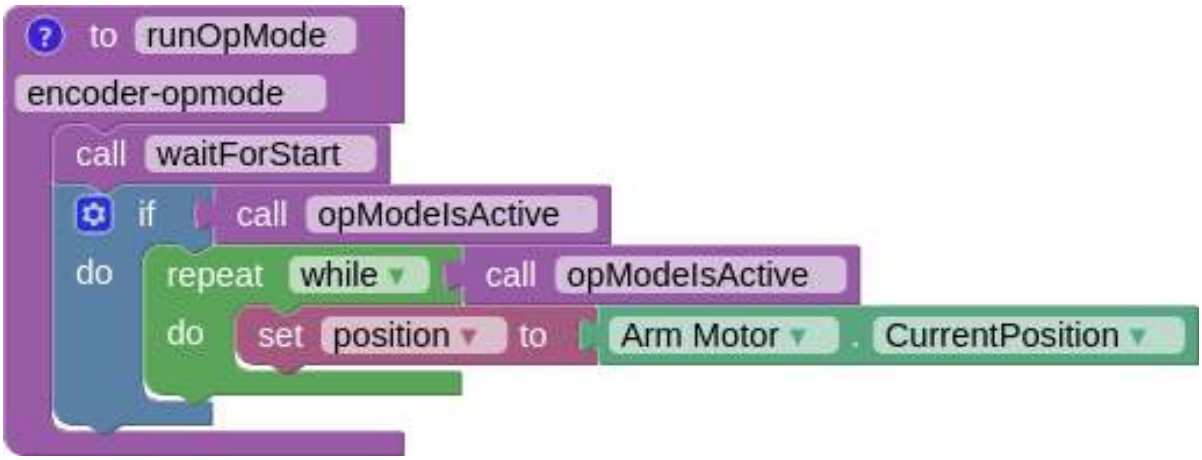
Reading Encoders

In FTC software, quadrature encoders and motors are accessed with the same motor object. What this means is that we can access an encoder's position like so:

Java

```
int position = motor.getCurrentPosition();
```

Blocks

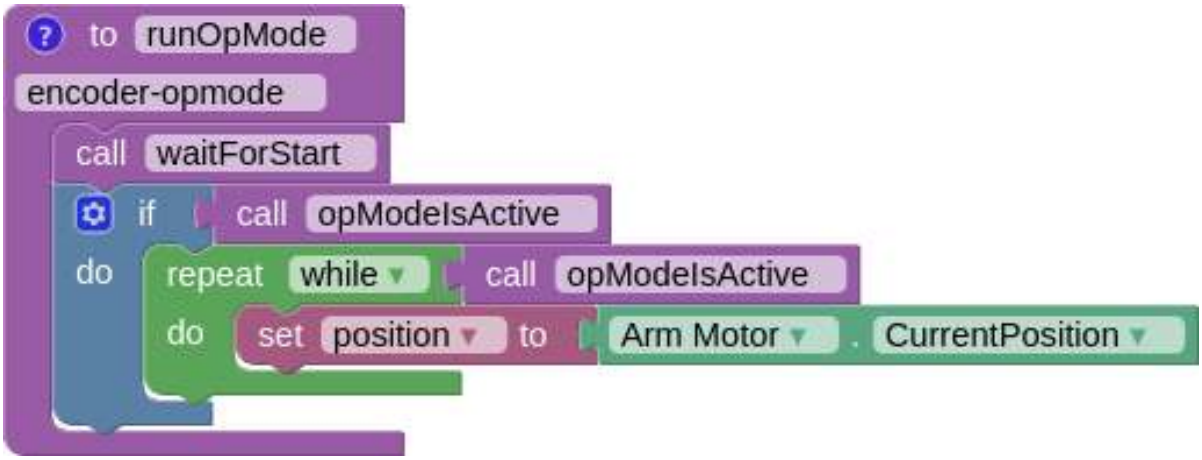


While convenient if one uses the built-in motor encoder, this can easily become confusing if using external encoders. To use external encoders, you must use the motor object associated with the port. For example, if there is a motor in port 1 named "Arm Motor" and an external encoder plugged into encoder port 1, you must do the following to get the encoder's position:

Java

```
DcMotor motor = hardwareMap.dcMotor.get("Arm Motor");
double position = motor.getCurrentPosition();
```

Blocks

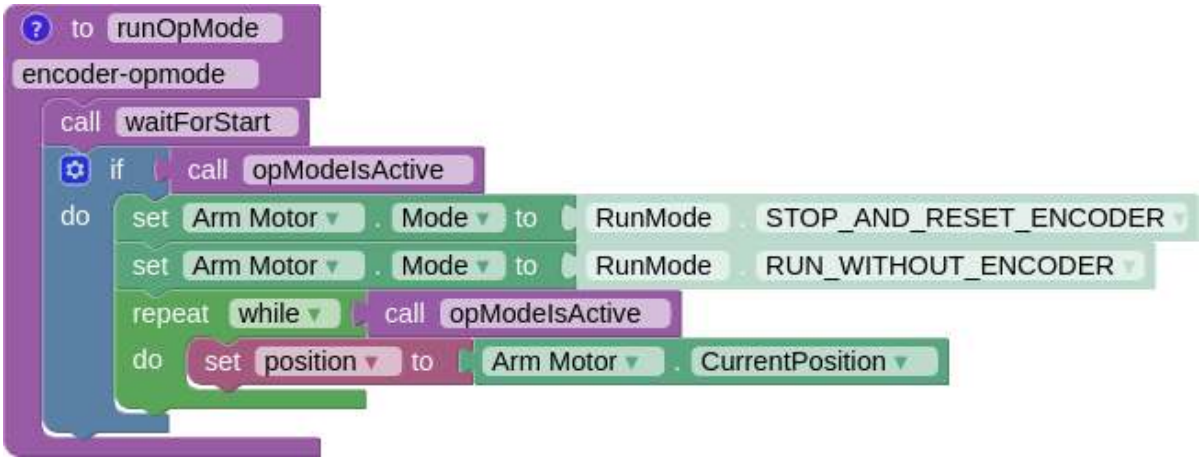


Great! We now have the relative position of our encoder, reported in the number of "counts" it is from what it considers to be zero. However, it is often convenient to have the encoder start at zero at the beginning of the OpMode. While it technically does not change anything, it can help with debugging and simplify future code. To do this, we can add a call to reset the encoders before we read them.

Java

```
DcMotor motor = hardwareMap.dcMotor.get("Arm Motor");
motor.setMode(DcMotor.RunMode.STOP_AND_RESET_ENCODER); // Reset the motor encoder
motor.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER); // Turn the motor back on when we␣
↪are done
int position = motor.getCurrentPosition();
```

Blocks



As a note, **RUN_WITHOUT_ENCODER does not disable the encoder. It instead tells the SDK not to use the motor encoder for built-in velocity control**. We will go over what this means in a later section, but for now, know that it turns the motor back on so we can use it after the encoder is reset.

Now we have our position (in counts) relative to the starting angle of the encoder. We can make a quick program to see this in action. Here, we use a motor encoder plugged into a port named "Arm Motor" in the hardware configuration.

Java

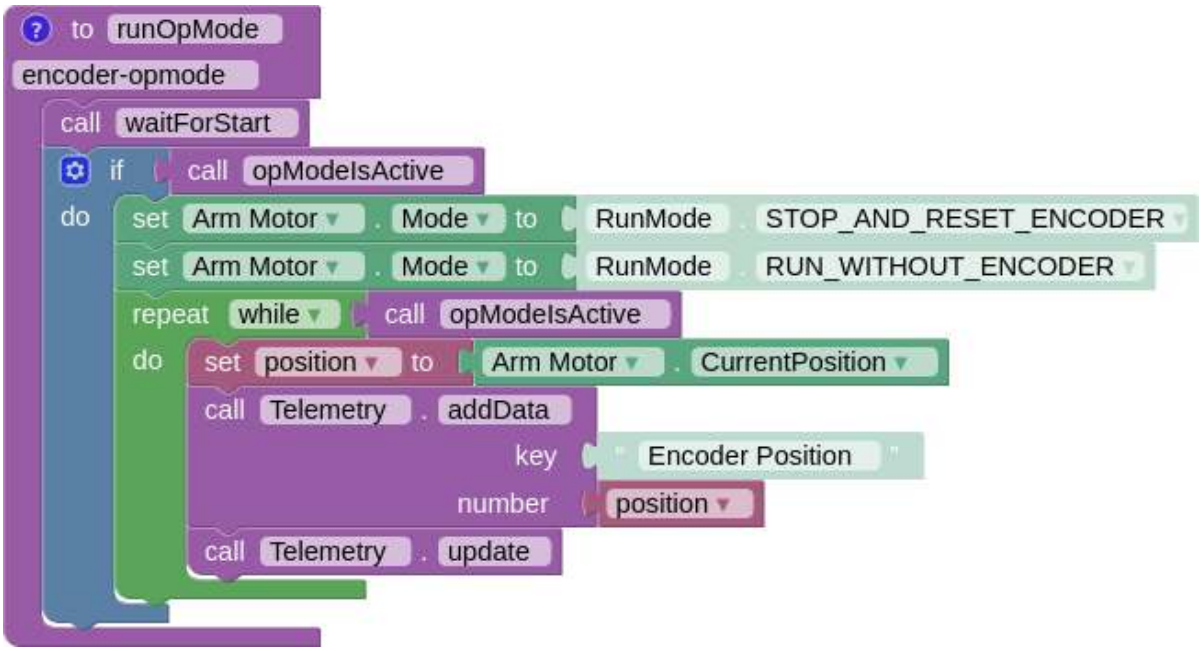
```
package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;
@TeleOp
public class EncoderOpmode extends LinearOpMode {
    @Override
    public void runOpMode() throws InterruptedException {
        // Find a motor in the hardware map named "Arm Motor"
        DcMotor motor = hardwareMap.dcMotor.get("Arm Motor");

        // Reset the motor encoder so that it reads zero ticks
        motor.setMode(DcMotor.RunMode.STOP_AND_RESET_ENCODER);

        // Turn the motor back on, required if you use STOP_AND_RESET_ENCODER
        motor.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);

        waitForStart();

        while (opModeIsActive()) {
            // Get the current position of the motor
            int position = motor.getCurrentPosition();

            // Show the position of the motor on telemetry
            telemetry.addData("Encoder Position", position);
            telemetry.update();
```

(continues on next page)

(continued from previous page)

```
        }
    }
}
```

Blocks



If you run the above OpMode and turn the encoder, you should see the values change as you move. If you rotate the shaft back to where it started, you will see the number return to (very close to) zero. As an exercise, rotate the shaft one full revolution (360) degrees and note down the number.

There is one more thing we can do with encoders. While knowing the number of counts something moved is useful, oftentimes one will need a different number, like the number of revolutions the encoder has turned or the angle it is at. To determine these, we need a constant, the encoders Counts Per Revolution or CPR. For external encoders, this number is often provided in a datasheet. For motors, it will generally be on the product page, although some motors (most notably the Rev Ultraplanetary Gearbox) do not provide it plainly.

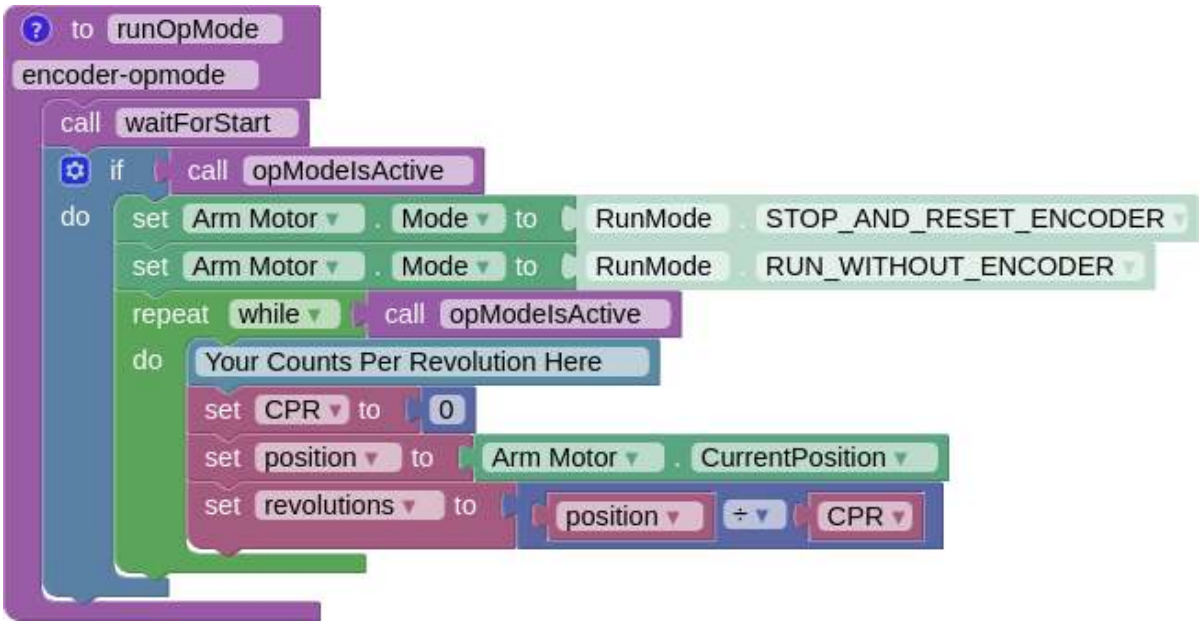
Tip: You can calculate a motor's Counts Per Revolution by taking the base motor's Counts Per Revolution and multiplying it by the gearbox ratio. Be careful to use the actual gearbox ratio when doing this! For example, a 5:1 Ultraplanetary motor would have a count per revolution of 28 * (5.23) = 146.44 because the base motor has 28 Counts Per Revolution and the 5:1 gearbox actually has a 5.23:1 gear ratio. Remember, when using two gearboxes on top of each other, you multiply the gear ratios together.

In the following example, we divide the encoder position by its counts per revolution to obtain the number of revolutions the encoder has rotated. You have to replace [Your Counts Per Revolution Here] with the counts per revolution of your motor, found on its product page or calculated using the above tip.

Java

```
double CPR = [Your Counts Per Revolution Here];

int position = motor.getCurrentPosition();
double revolutions = position/CPR;
```

Blocks

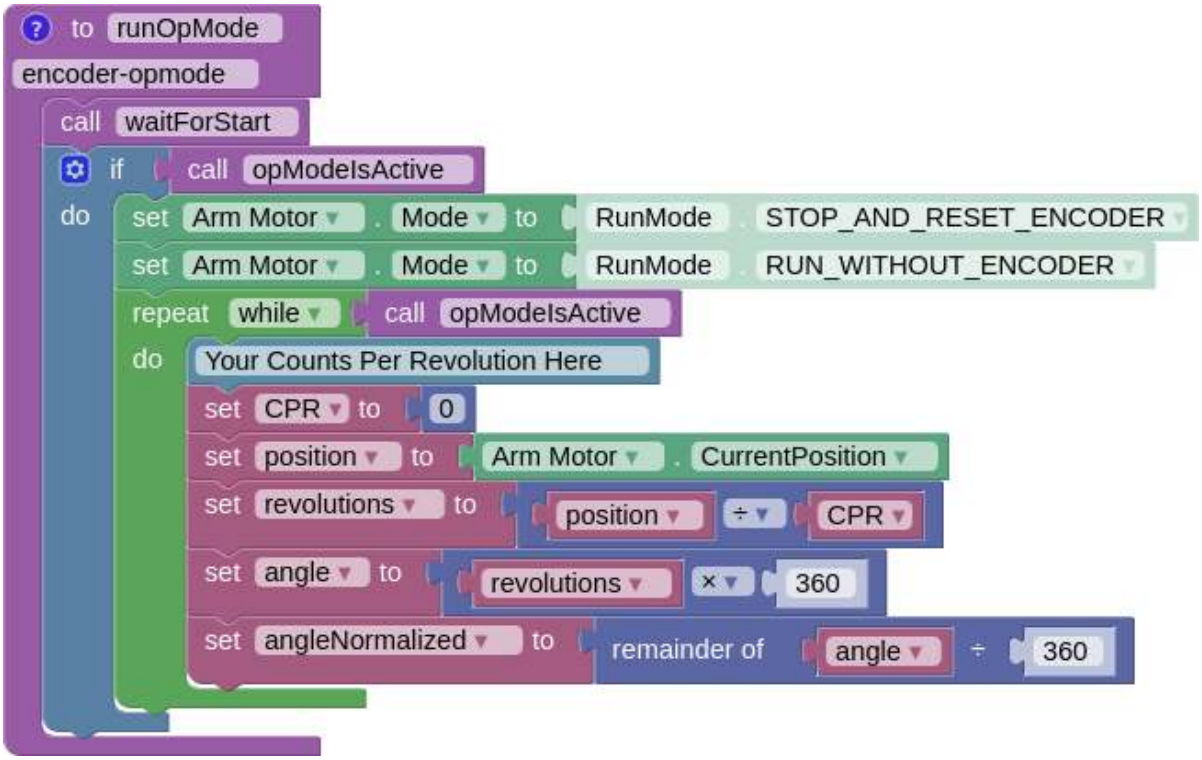


There is one more number we can get: the angle of the shaft. Calculating this number is very simple. We can multiply the number of rotations by 360 (since there are 360 degrees in one revolution). You might notice that this number can go above 360 as the shaft rotates multiple times. As such, we introduce angleNormalized, which will always be between 0 and 360.

Java

```
double CPR = [Your Counts Per Revolution Here];

int position = motor.getCurrentPosition();
double revolutions = position/CPR;

double angle = revolutions * 360;
double angleNormalized = angle % 360;
```

Blocks



Putting it all together, we get the following testing program.

Java

```
package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;
@TeleOp
public class EncoderOpmode extends LinearOpMode {
    @Override
    public void runOpMode() throws InterruptedException {
        // Find a motor in the hardware map named "Arm Motor"
        DcMotor motor = hardwareMap.dcMotor.get("Arm Motor");

        // Reset the motor encoder so that it reads zero ticks
        motor.setMode(DcMotor.RunMode.STOP_AND_RESET_ENCODER);

        // Turn the motor back on, required if you use STOP_AND_RESET_ENCODER
        motor.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);

        waitForStart();

        while (opModeIsActive()) {
            double CPR = [Your Counts Per Revolution Here];
```

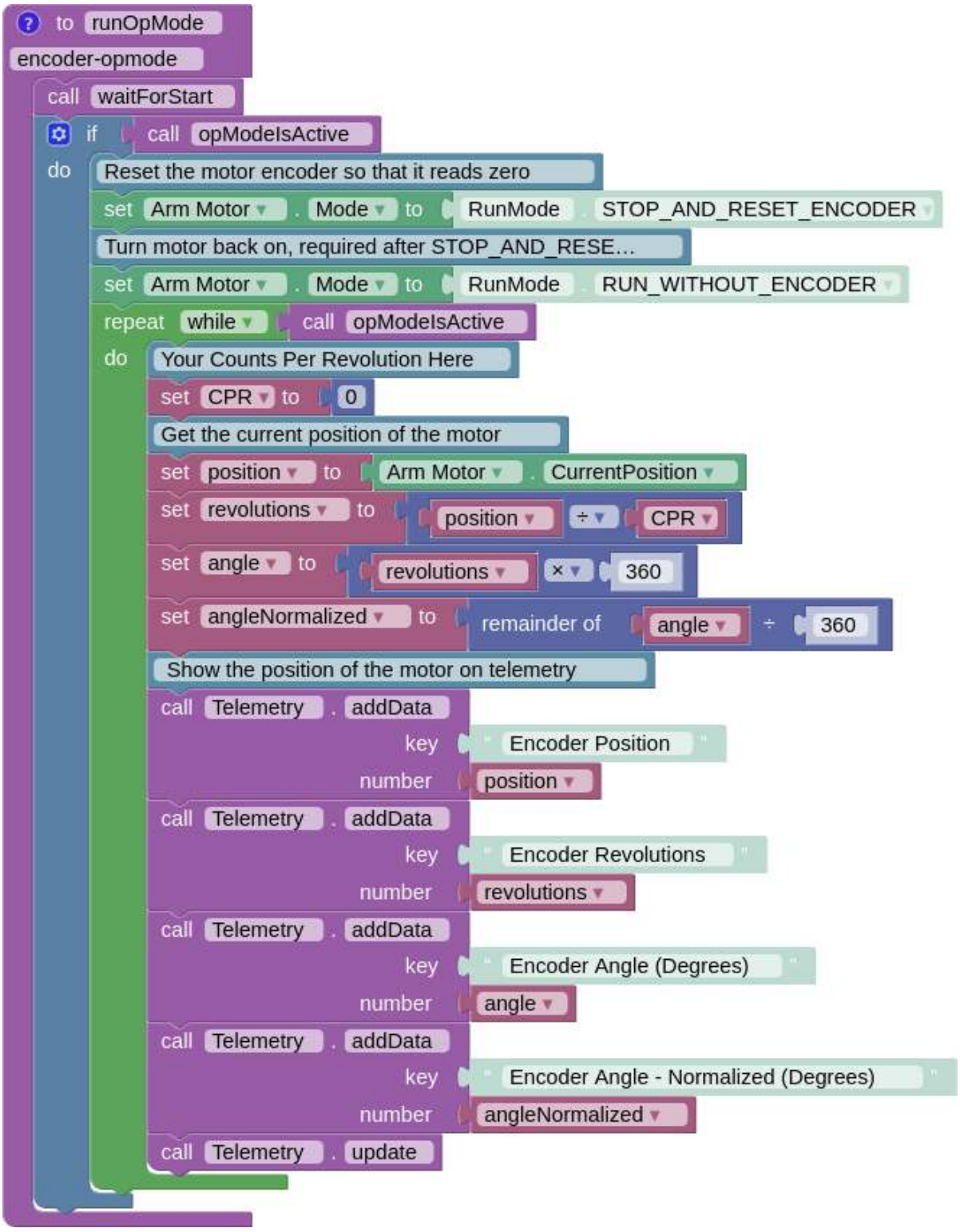
(continues on next page)

(continued from previous page)

```
            // Get the current position of the motor
            int position = motor.getCurrentPosition();
            double revolutions = position/CPR;

            double angle = revolutions * 360;
            double angleNormalized = angle % 360;

            // Show the position of the motor on telemetry
            telemetry.addData("Encoder Position", position);
            telemetry.addData("Encoder Revolutions", revolutions);
            telemetry.addData("Encoder Angle (Degrees)", angle);
            telemetry.addData("Encoder Angle - Normalized (Degrees)",␣
↪angleNormalized);
            telemetry.update();
        }
    }
}
```

Blocks

Blocks file download


to runOpMode
encoder-opmode
call waitForStart
if call opModeIsActive
do
Reset the motor encoder so that it reads zero
set Arm Motor . Mode to RunMode . STOP_AND_RESET_ENCODER
Turn motor back on, required after STOP_AND_RESE…
set Arm Motor . Mode to RunMode . RUN_WITHOUT_ENCODER
repeat while call opModeIsActive
do
Your Counts Per Revolution Here
set CPR to 0
Get the current position of the motor
set position to Arm Motor . CurrentPosition
set revolutions to position ÷ CPR
set angle to revolutions × 360
set angleNormalized to remainder of angle ÷ 360
Show the position of the motor on telemetry
call Telemetry . addData
key " Encoder Position "
number position
call Telemetry . addData
key " Encoder Revolutions "
number revolutions
call Telemetry . addData
key " Encoder Angle (Degrees) "
number angle
call Telemetry . addData
key " Encoder Angle - Normalized (Degrees) "
number angleNormalized
call Telemetry . update

Tracking Wheels and Spools

Up to this point, we have mostly been working with motors rotating something. However, many mechanisms in FTC are linear, and it can be desirable to measure these in a linear unit as well. Fortunately, this is very straightforward. All we need to know is the diameter of the object we are measuring.

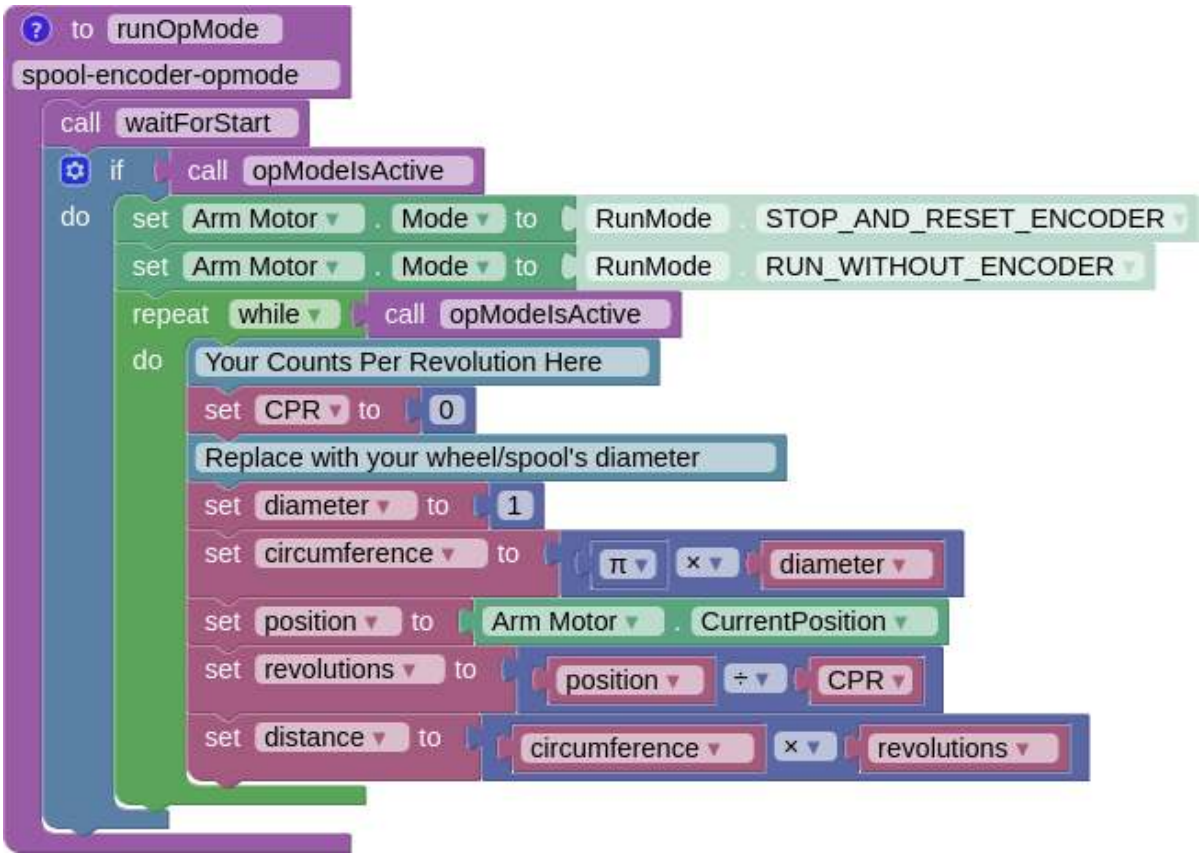
Be careful when selecting your diameter. For wheels, it is the outer diameter of the wheel, but for spools, it is the inner diameter of the spool, where the string rests. For chain and belts, it is the "pitch diameter" of the sprocket or pulley.

From here, we can calculate the circumference (the length of the arc of the circle, or the distance the wheel/spool will travel in one rotation)

Java

```
double diameter = 1.0; // Replace with your wheel/spool's diameter
double circumference = Math.PI * diameter;

double distance = circumference * revolutions;
```

Blocks



Note: Units are very important when dealing with FTC programming, so make sure they are always consistent! Whatever units you use for the diameter are the units for your calculated distance. So if you measure

your diameter in inches, the reported distance will also be in inches.

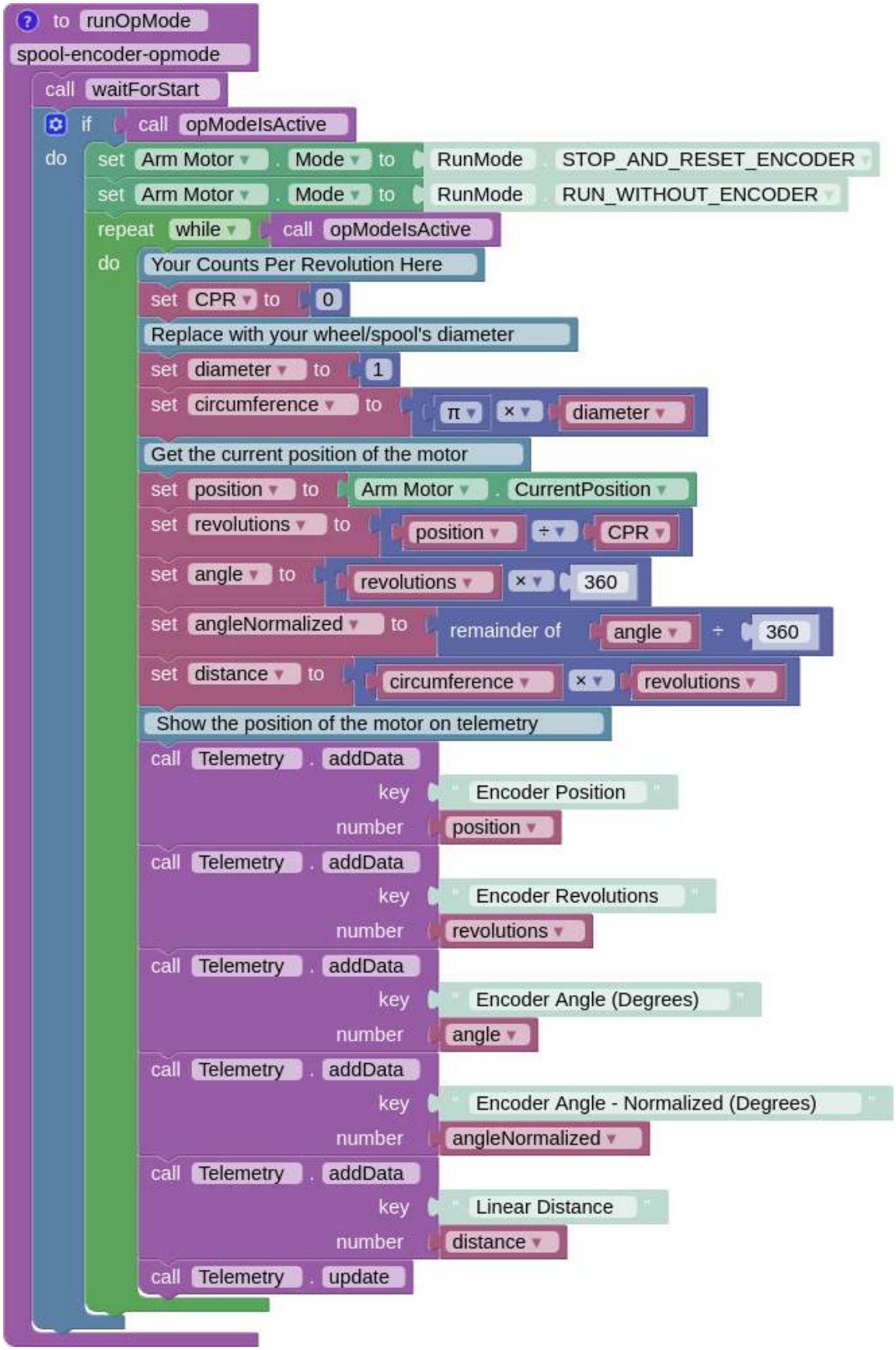
Putting this all together with our previous testing program, we get

Java

```
package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;
@TeleOp
public class SpoolEncoderOpmode extends LinearOpMode {
   @Override
   public void runOpMode() throws InterruptedException {
      // Find a motor in the hardware map named "Arm Motor"
      DcMotor motor = hardwareMap.dcMotor.get("Arm Motor");

      // Reset the motor encoder so that it reads zero ticks
      motor.setMode(DcMotor.RunMode.STOP_AND_RESET_ENCODER);

      // Turn the motor back on, required if you use STOP_AND_RESET_ENCODER
      motor.setMode(DcMotor.RunMode.RUN_WITHOUT_ENCODER);

      waitForStart();

      while (opModeIsActive()) {
            double CPR = [Your Counts Per Revolution Here];

            double diameter = 1.0; // Replace with your wheel/spool's diameter
            double circumference = Math.PI * diameter;

            // Get the current position of the motor
            int position = motor.getCurrentPosition();
            double revolutions = position/CPR;

            double angle = revolutions * 360;
            double angleNormalized = angle % 360;

            double distance = circumference * revolutions;

            //Show the position of the motor on telemetry
            telemetry.addData("Encoder Position", position);
            telemetry.addData("Encoder Revolutions", revolutions);
            telemetry.addData("Encoder Angle (Degrees)", angle);
            telemetry.addData("Encoder Angle - Normalized (Degrees)",␣
↪angleNormalized);
            telemetry.addData("Linear Distance", distance);
            telemetry.update();
      }
   }
}
```

Blocks

```
Blocks file download
```


to runOpMode
spool-encoder-opmode
call waitForStart
if call opModeIsActive
do set Arm Motor . Mode to RunMode . STOP_AND_RESET_ENCODER
set Arm Motor . Mode to RunMode . RUN_WITHOUT_ENCODER
repeat while call opModeIsActive
do Your Counts Per Revolution Here
set CPR to 0
Replace with your wheel/spool's diameter
set diameter to 1
set circumference to π × diameter
Get the current position of the motor
set position to Arm Motor . CurrentPosition
set revolutions to position ÷ CPR
set angle to revolutions × 360
set angleNormalized to remainder of angle ÷ 360
set distance to circumference × revolutions
Show the position of the motor on telemetry
call Telemetry . addData
key " Encoder Position "
number position
call Telemetry . addData
key " Encoder Revolutions "
number revolutions
call Telemetry . addData
key " Encoder Angle (Degrees) "
number angle
call Telemetry . addData
key " Encoder Angle - Normalized (Degrees) "
number angleNormalized
call Telemetry . addData
key " Linear Distance "
number distance
call Telemetry . update

Running Motors With Encoders

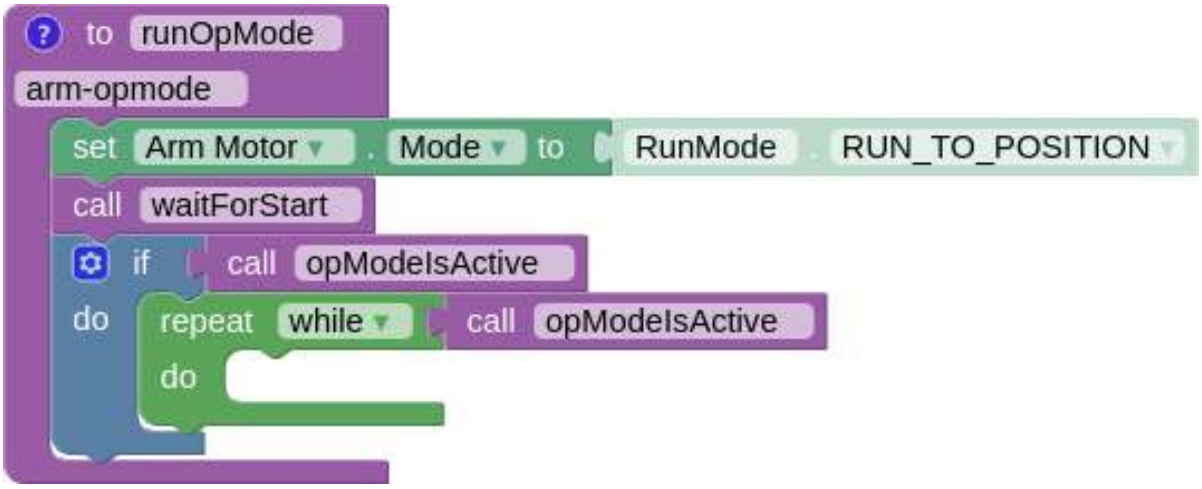
We've learned how to read encoder values, but how do you set where you want to go and tell the motor to go there?

Earlier, we learned about the RUN_WITHOUT_ENCODER mode for the motor. We can use another motor mode, RUN_TO_POSITION, to tell the motor to run to a specific position in ticks, like so:

Java

```
DcMotor motor = hardwareMap.dcmotor.get("Arm Motor");
motor.setMode(DcMotor.RunMode.RUN_TO_POSITION); // Tells the motor to run to the␣
↪specific position
```

Blocks



Tip: You can find out more about run modes at the official REV Robotics Documentation page[197]

However, before we tell the motor to go to a position, we have to tell the motor what position to run to. **Note that this value must be an integer**. Let's amend the above code to do that.

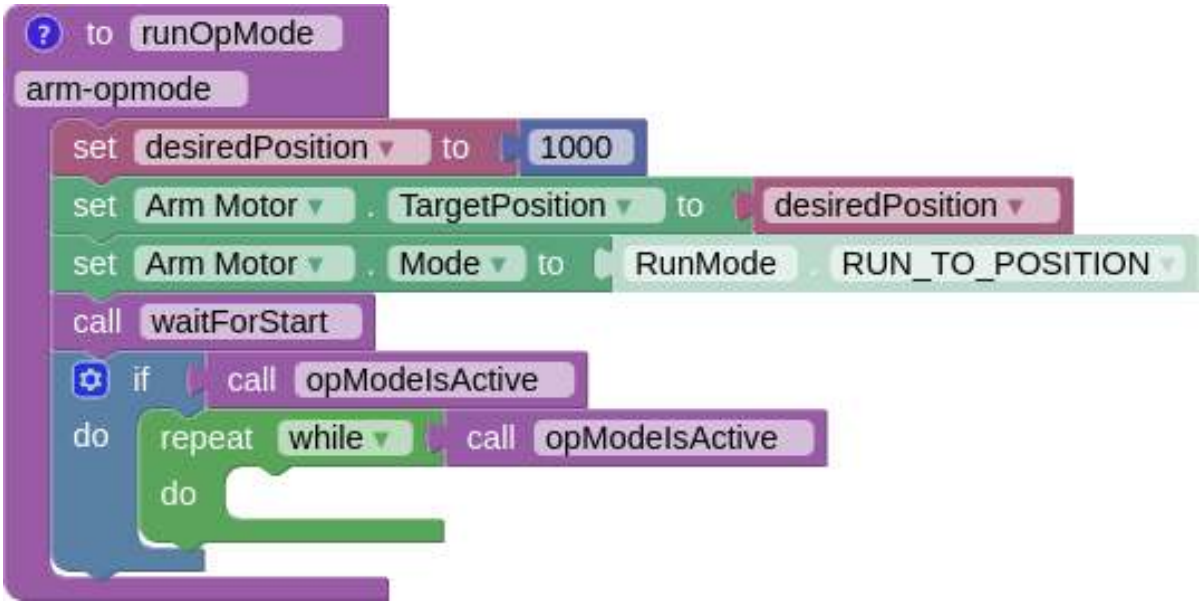
Warning: Setting the motor to RUN_TO_POSITION mode before setting a target position will throw an error. Be careful not to do that!

[197] https://docs.revrobotics.com/duo-control/programming/using-encoder-feedback

Java

```
DcMotor motor = hardwareMap.dcmotor.get("Arm Motor");
int desiredPosition = 1000; // The position (in ticks) that you want the motor to␣
↪move to
motor.setTargetPosition(desiredPosition); // Tells the motor that the position it␣
↪should go to is desiredPosition
motor.setMode(DcMotor.RunMode.RUN_TO_POSITION);
```

Blocks



This code tells the motor to move to 1000 ticks, using a PID loop to control the motor's position. You can read more about PID loops here.[198]

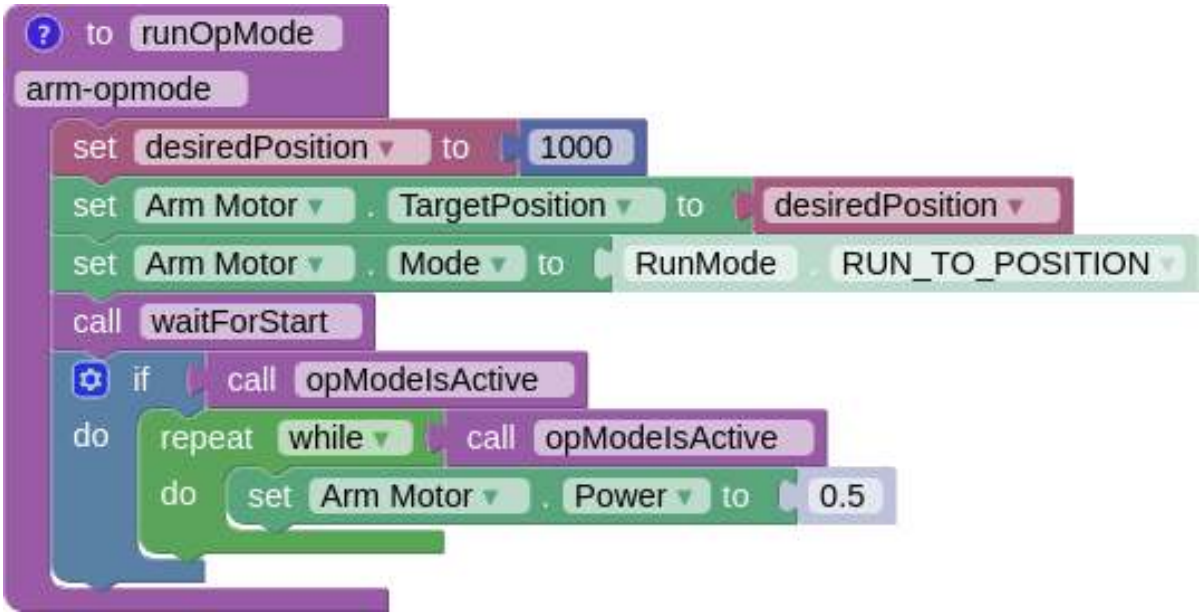
We can cap the speed that the motor runs at using the following code:

Java

```
DcMotor motor = hardwareMap.dcmotor.get("Arm Motor");
int desiredPosition = 1000; // The position (in ticks) that you want the motor to␣
↪move to
motor.setTargetPosition(desiredPosition); // Tells the motor that the position it␣
↪should go to is desiredPosition
motor.setMode(DcMotor.RunMode.RUN_TO_POSITION);
motor.setPower(0.5); // Sets the maximum power that the motor can go at
```

[198] https://gm0.org/en/latest/docs/software/concepts/control-loops.html#pid

Blocks



Now, let's use this information to control an arm in an OpMode.

Java

```
package org.firstinspires.ftc.teamcode.Tests;

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.DcMotor;

@TeleOp
public class ArmOpMode extends LinearOpMode {
   @Override
   public void runOpMode() throws InterruptedException {
      // Position of the arm when it's lifted
      int armUpPosition = 1000;

      // Position of the arm when it's down
      int armDownPosition = 0;

      // Find a motor in the hardware map named "Arm Motor"
      DcMotor armMotor = hardwareMap.dcMotor.get("Arm Motor");

      // Reset the motor encoder so that it reads zero ticks
      armMotor.setMode(DcMotor.RunMode.STOP_AND_RESET_ENCODER);

      // Sets the starting position of the arm to the down position
      armMotor.setTargetPosition(armDownPosition);
      armMotor.setMode(DcMotor.RunMode.RUN_TO_POSITION);

      waitForStart();
```

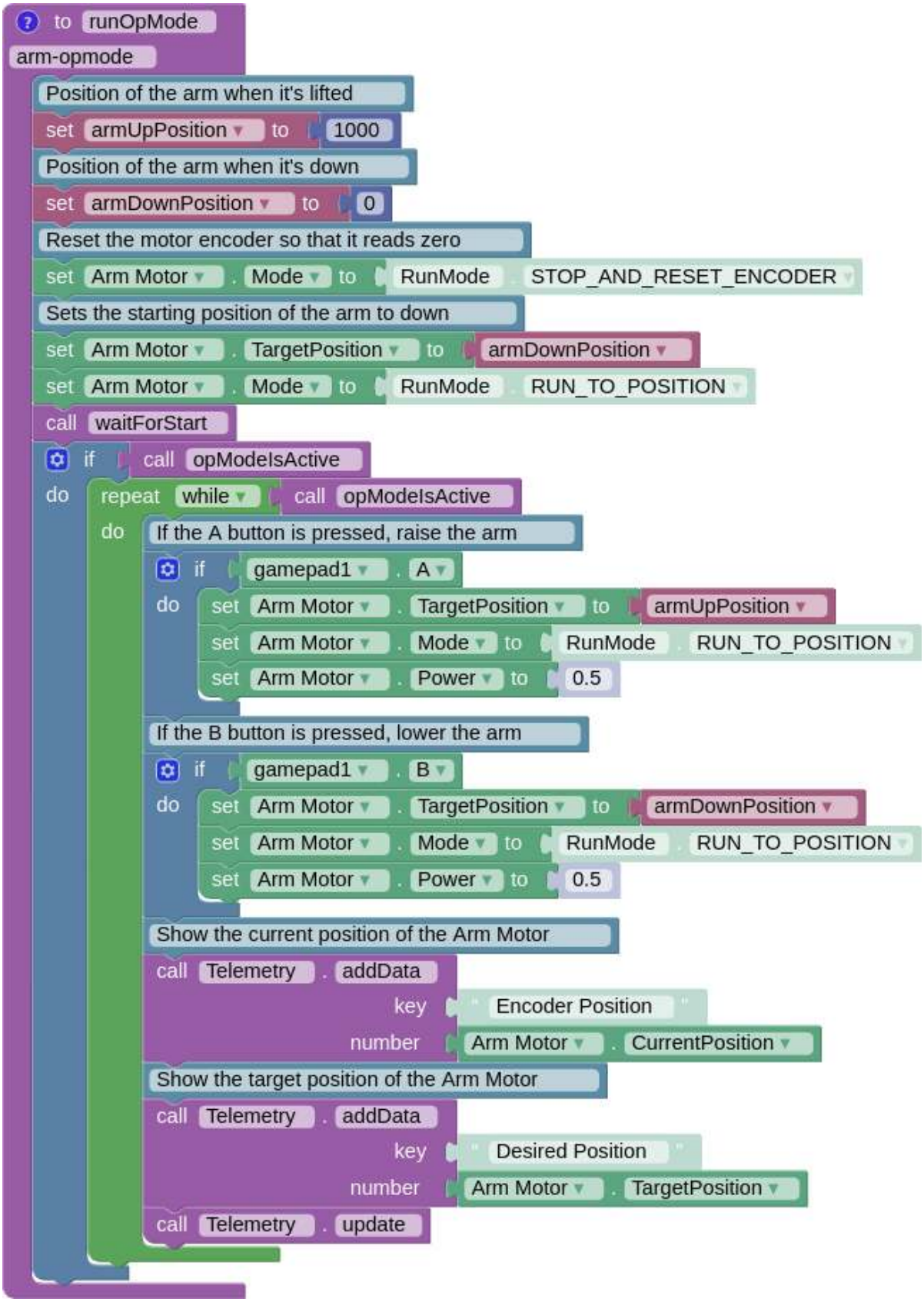
(continues on next page)

(continued from previous page)

```
      while (opModeIsActive()) {
            // If the A button is pressed, raise the arm
            if (gamepad1.a) {
               armMotor.setTargetPosition(armUpPosition);
               armMotor.setMode(DcMotor.RunMode.RUN_TO_POSITION);
               armMotor.setPower(0.5);
            }

            // If the B button is pressed, lower the arm
            if (gamepad1.b) {
               armMotor.setTargetPosition(armDownPosition);
               armMotor.setMode(DcMotor.RunMode.RUN_TO_POSITION);
               armMotor.setPower(0.5);
            }

            // Get the current position of the armMotor
            double position = armMotor.getCurrentPosition();

            // Get the target position of the armMotor
            double desiredPosition = armMotor.getTargetPosition();

            // Show the position of the armMotor on telemetry
            telemetry.addData("Encoder Position", position);

            // Show the target position of the armMotor on telemetry
            telemetry.addData("Desired Position", desiredPosition);

            telemetry.update();
      }
   }
}
```

Blocks

Blocks file download


to runOpMode
arm-opmode
Position of the arm when it's lifted
set armUpPosition to 1000
Position of the arm when it's down
set armDownPosition to 0
Reset the motor encoder so that it reads zero
set Arm Motor . Mode to RunMode . STOP_AND_RESET_ENCODER
Sets the starting position of the arm to down
set Arm Motor . TargetPosition to armDownPosition
set Arm Motor . Mode to RunMode . RUN_TO_POSITION
call waitForStart
if call opModeIsActive
do repeat while call opModeIsActive
do If the A button is pressed, raise the arm
if gamepad1 . A
do set Arm Motor . TargetPosition to armUpPosition
set Arm Motor . Mode to RunMode . RUN_TO_POSITION
set Arm Motor . Power to 0.5
If the B button is pressed, lower the arm
if gamepad1 . B
do set Arm Motor . TargetPosition to armDownPosition
set Arm Motor . Mode to RunMode . RUN_TO_POSITION
set Arm Motor . Power to 0.5
Show the current position of the Arm Motor
call Telemetry . addData
key " Encoder Position "
number Arm Motor . CurrentPosition
Show the target position of the Arm Motor
call Telemetry . addData
key " Desired Position "
number Arm Motor . TargetPosition
call Telemetry . update

10.2.5 Bulk Reads

Bulk reads are a LynxCommand that reads all sensor values (except I2C) on a hub at once. This takes the same amount of time to execute as any other LynxCommand, and can therefore save a lot of time in the execution loop; with a bulk read, reading ten sensors takes as much time as reading one sensor (if they are not I2C and are on the same hub).

This became much simpler to do with SDK versions 5.4 and above, with a built-in way to easily access it. Here is the official example[199] on how to use bulk reads.

Off Mode

This is the default, and the most boring; it means bulk reads are not used by the SDK when calling normal hardware-access methods.

Note: Bulk reads can still be accessed by calling the `LynxModule.getBulkInputData()` method, however if one wishes to use bulk reads (which we highly recommend) using `AUTO` or `MANUAL` modes is simpler.

To manually set `OFF` mode, you need to run

```
List<LynxModule> allHubs = hardwareMap.getAll(LynxModule.class);

for (LynxModule hub : allHubs) {
   hub.setBulkCachingMode(LynxModule.BulkCachingMode.OFF);
}
```

Auto Mode

This is the simplest mode to use that utilizes bulk reads; a new bulk read is done when a hardware read is repeated. As an example of this

```
List<LynxModule> allHubs = hardwareMap.getAll(LynxModule.class);

for (LynxModule hub : allHubs) {
   hub.setBulkCachingMode(LynxModule.BulkCachingMode.AUTO);
}

while (opModeIsActive()) {
   // Will run one bulk read per cycle; however, if e.g.
   // frontLeftMotor.getCurrentPosition() was called again,
   // a new bulk read would be issued
   int frontLeftEncoderPos = frontLeftMotor.getCurrentPosition();
   int frontRightEncoderPos = frontRightMotor.getCurrentPosition();
   int backLeftEncoderPos = backLeftMotor.getCurrentPosition();
   int backRightEncoderPos = backRightMotor.getCurrentPosition();
}
```

However, this can be problematic, if the same hardware read is called more than once in a given loop; an example of this

[199] https://github.com/first-tech-challenge/FtcRobotController/blob/master/FtcRobotController/src/main/java/org/firstinspires/ftc/robotcontroller/external/samples/ConceptMotorBulkRead.java

```
List<LynxModule> allHubs = hardwareMap.getAll(LynxModule.class);

for (LynxModule hub : allHubs) {
   hub.setBulkCachingMode(LynxModule.BulkCachingMode.AUTO);
}

while (opModeIsActive()) {
   // Will run two bulk read per cycles,
   // as frontLeftMotor.getCurrentPosition() is called twice
   int frontLeftEncoderPos = frontLeftMotor.getCurrentPosition();
   int frontLeftEncoderPos2 = frontLeftMotor.getCurrentPosition();
}
```

Overall, this is recommended, as it is very unlikely to mess anything up and can give significant performance improvements for little effort. On the user side, one does not need to manually flush the bulk read cache; however, this means you lose some control.

Manual Mode

In manual mode the cache for bulk reads is only reset once manually reset. This can be useful, as it is the way to absolutely minimize extraneous reads, however if the cache is not reset, stale values will be returned. That being said, here's a proper implementation of MANUAL mode

```
List<LynxModule> allHubs = hardwareMap.getAll(LynxModule.class);

for (LynxModule hub : allHubs) {
   hub.setBulkCachingMode(LynxModule.BulkCachingMode.MANUAL);
}

while (opModeIsActive()) {
   // Will run one bulk read per cycle,
   // even as frontLeftMotor.getCurrentPosition() is called twice
   // because the caches are being handled manually and cleared
   // once a loop
   for (LynxModule hub : allHubs) {
      hub.clearBulkCache();
   }

   int frontLeftEncoderPos = frontLeftMotor.getCurrentPosition();
   int frontLeftEncoderPos2 = frontLeftMotor.getCurrentPosition();
}
```

Warning: When in MANUAL mode, if the cache is not cleared appropriately, stale values will be returned. For that reason, if you are not quite sure what you are doing, we recommend AUTO mode; while MANUAL mode can have some performance improvements if AUTO mode is not used optimally, it has less room for catastrophic error.

Tip: Bulk reads are issued per hub, so you don't necessarily need to bulk read both hubs at the same time or every loop.

10.2.6 Computer Vision

Computer vision is the process of using computers to understand digital images, such as photographs and videos. Computer vision offers object detection which is commonly needed in FTC®.

Currently, there are 3 commonly used forms of computer vision: TensorFlow Lite, AprilTags, and OpenCV (via VisionPortal/EasyOpenCV).

Officially Supported Solutions

TensorFlow

TensorFlow is Google's machine learning technology, which can be trained to detect objects. The FTC SDK uses TensorFlow Lite, which is a lightweight version of Google's TensorFlow designed to run on mobile devices. Along with Android Studio and OnBot support, TensorFlow has block support which makes it a good choice for most teams.

Sample OpModes for TensorFlow being used for pixel detection (CENTERSTAGE) can be found here[200] (Blocks), and a Java example can be found here[201].

FIRST® has released a tool called FTC-ML to train your own TensorFlow Lite model for detecting custom objects. Details about FTC-ML can be found on FTC Docs[202].

AprilTags

AprilTags detect specific low-resolution, black-and-white images that are placed on various parts of the field. AprilTags are a great way to detect specific parts of the fields at wide angles, distances, and in a variety of lighting conditions. AprilTags are supported in OnBot Java, Android Studio, and Blocks.

You can learn more about implementing AprilTags on FTC Docs[203]. Supported AprilTag images are available on this pdf[204].

VisionPortal

Introduced with FTC SDK v8.2, the VisionPortal API integrates the EasyOpenCV[205] project into the FTC SDK. With VisionPortal, you can attach multiple VisionProcessors to a single camera. VisionProcessors are an easy way to create OpenCV pipelines, which allows for custom manipulation and processing to be applied to each incoming frame. VisionPortal is the most powerful form of computer vision, but it is also the hardest to use. Therefore, we can only recommend it to more advanced teams.

[200] https://github.com/FIRST-Tech-Challenge/FtcRobotController/wiki/Blocks-Sample-OpMode-for-TFOD
[201] https://github.com/FIRST-Tech-Challenge/FtcRobotController/wiki/Java-Sample-OpMode-for-TFOD
[202] https://ftc-docs.firstinspires.org/ftc_ml/index.html
[203] https://ftc-docs.firstinspires.org/en/latest/apriltag/vision_portal/apriltag_intro/apriltag-intro.html
[204] https://www.dotproduct3d.com/uploads/8/5/1/1/85115558/apriltags1-20.pdf
[205] https://github.com/OpenFTC/EasyOpenCV

Additional Vision Resources

EasyOpenCV Simulator

EasyOpenCV Simulator is a straightforward way to test your pipelines directly on your computer. It supports Windows, macOS, and Linux, and simulates a portion of the FTC SDK structure including the VisionPortal API, allowing you to copy and paste pipelines.

You can find EasyOpenCV Simulator here[206].

FTC® Dashboard

FTC Dashboard runs a dashboard webpage on the Control Hub that, among other things, streams a live preview of a connected camera. This is a very useful tool for testing and debugging vision applications.

> **Warning:** FTC Dashboard is not legal to run during matches, make sure it is disabled during a competition. More information can be found here[207].

You can find FTC Dashboard here[208].

Scrcpy

Scrcpy is an easy way to display and control Android devices connected over USB. It supports Linux, Windows, and macOS, offers low latency, requires low resources, and has great performance. It is extremely useful for debugging vision code on a Control Hub, as it will enable you to see the camera output in near real time. It can also be used with a robot controller phone, however the camera output can be seen on the phone's screen.

You can find scrcpy here[209].

10.3 Programming Concepts

Many of the pages here are on concepts that are not necessarily FTC® specific, and are generally more theoretical then actual tutorials on FTC specific concepts.

[206] https://github.com/deltacv/EOCV-Sim
[207] https://acmerobotics.github.io/ftc-dashboard/competition
[208] https://github.com/acmerobotics/ftc-dashboard
[209] https://github.com/Genymobile/scrcpy

10.3.1 Control Loops

Control loops are software used to operate power transmission systems (such as a drivetrain or linear slide) in a fast and controlled fashion. Not only do control loops let you run mechanisms quickly without fear of losing control, in many cases, they help preserve the longevity of mechanisms by reducing rapid change of applied motor voltage.

What is Error?

The first thing that must be defined when discussing control loops is the concept of error.

Error is defined as the difference between where you are and where you want to be. For instance, say you tell your drivetrain to drive at 30 inches per second, but in actuality, at a time, the drivetrain is driving at 28 inches per second. Since $30 - 28 = 2$, the error of the drivetrain's speed at this time T is 2 inches per second. In other words, at a time $t = T$, $e(t) = 2$.

PID

A PID controller (or Proportional Integral Derivative controller) is a control loop that solely uses error to control the system. PID is a form of a **feedback control loop**, or **closed loop control**. This means that data about the variable you are controlling is required in order for the loop to control that variable. In this case, information about the **error** of the system is required to control the system with a PID controller.

The Optional Calculus

The following equation represents the rigorous mathematical definition of the output of a PID controller f at any given time t:

$$f(t) = K_p e(t) + K_i \int_o^t e(t)\mathsf{d}t + K_d \frac{\mathsf{d}e(t)}{\mathsf{d}t}$$

where K_p, K_i, and K_d are constants and $e(t)$, as previously mentioned, is the error in the system.

If you have no experience with calculus, don't worry; while PID is fundamentally rooted in calculus, you do not need any calculus experience to be able to understand it, only basic algebra. However, you are still urged to read the rest of the section regardless of calculus experience, as the formula alone doesn't tell you why it works.

Simplification of the PID formula

Here is a simplified version of the PID formula: $f(t) = K_p P(t) + K_i I(t) + K_d D(t)$

All we have done is simply take the full formula and replaced part of the terms with functions: $P(t)$, $I(t)$, and $D(t)$.

The Proportional Term

The first component of the function, $K_pP(t)$, is by far the most simple and easy to understand, as $P(t) = e(t)$. For the sake of example, let's pretend that $K_i = 0$ and $K_d = 0$ (a PID controller with only a proportional constant is known as a **P controller**). How will the system behave? Well, if the error is large, the output will be large. Likewise, if the error is small, the output will be small. Also, ideally, given enough time, the system always approaches its destination, assuming K_p is of the correct sign.

Say we apply this to a drivetrain. You want to drive a distance D, and you decide to set your motor powers using a P controller to accomplish this. In this case, your error is how far away the robot is from the desired location. As you start to drive forward, your error is large, so you drive forward quickly, which is desirable. After all, you aren't concerned with overshooting the target yet if you are far away from it.

But as the robot's distance to the target approaches 0, you will start to slow down, gaining more control over the robot. Once the error is zero, ideally, the robot will stop, and you have reached your destination. If you happen to overshoot, the error will become negative, and the robot will backtrack, repeating the process.

The Derivative Term

This term, $K_dD(t)$, is intended to dampen the rate of change of the error. In other words, it tries to keep the error constant. How is this done?

Well, for those of you with calculus under your belt, $D(t) = \frac{de(t)}{dt}$. For those without calculus experience, it represents how fast the error is changing. Graphically, $D(t)$ is simply the slope of the error at any given time t.

This slope can be calculated by keeping track of the error over successive iterations of the control loop. One iteration occurs at time t_n with an error of $e(t_n)$. At the next iteration, the time is t_{n+1} with an error of $e(t_{n+1})$. Thus, to find $D(t)$, simply find the slope of $e(t)$ given these two points.

The Integral Term

Admittedly, the integral term is the least important term for FTC® PID control loops. With a properly tuned K_p and K_d, you often can just set K_i to 0 and call it a day.

However, it can still be useful in some cases. Just like the derivative term, the integral term intends to correct for overshoot. if the system thinks it reached its destination, it will stop, even when, in fact, the error is not yet 0. Perhaps the motor is no longer being supplied enough power to move. Well, given enough time, the integral term will increase the output (in this case, motor power), causing movement towards the destination.

To explain without calculus, the integral term essentially sums the error over a specific interval of time. To do this, error in each loop iteration is added to a variable (in this case, $I(t)$).

However, summing error this way has an unfortunate side effect: the longer the loop takes to complete one iteration, the more slowly this sum increases, which is obviously not desirable, as we don't want lag to affect how the robot moves. To compensate for this, before the error is added to $I(t)$, it is multiplied by how long the previous loop took to-complete, or $t_{n+1} - t_n$, preventing lag from making the system sum more slowly.

So say the robot stops short of the target. The P and D combination aren't strong enough to move it forward to the destination. You can either tune K_p and K_d to compensate (**this is recommended**), or you can add the integral term to increase output (**this works too, but requires more attention and tuning to achieve the same result**).

PID Pseudocode

```
while True:
   current_time = get_current_time()
   current_error = desire_position-current_position

   p = k_p * current_error

   i += k_i * (current_error * (current_time - previous_time))

   if i > max_i:
       i = max_i
   elif i < -max_i:
       i = -max_i

   D = k_d * (current_error - previous_error) / (current_time - previous_time)

   output = p + i + d

   previous_error = current_error
   previous_time = current_time
```

Tuning a PID Loop

The most important thing to know while tuning a PID loop is how each of the terms affects the output. This can allow you to see which gains need to be adjusted.

For example, if the target is not reached but instead the setpoint begins to oscillate around the target, it means there is not enough D gain. If the target is eventually reached, albeit very slowly, that means there is not enough P gain or the D gain is too high.

In brief, the P variable drives the error towards zero, the I variable corrects for steady state error, and the D variable dampens the effects of the P variable, more so as error approaches zero, which prevents overshoot.

The most common method for tuning a PID controller is as follows:

1. Set the I and D gains to zero
2. Increase the P gain until there are oscillations around the target
3. Increase the D gain until no overshoot occurs
4. If there is steady state error, increase the I gain until it is corrected

An important thing to note is that most systems do not need both I and D control. Generally, systems without a lot of friction do not need an I term, but will need more D control. Systems with a lot of friction, on the other hand, generally do not need D control because the friction facilitates deceleration but need I control because the friction prevents the system from reaching the target otherwise.

For a more in-depth explanation, click here[210]

[210] https://blog.wesleyac.com/posts/intro-to-control-part-two-pid-tuning

Built-In PID Controller

For situations where one needs to control the velocity or position of a single motor, the built in PID controller can be used. PID can be enabled by changing the run mode to `RUN_USING_ENCODER`

Hint: Many misunderstand the use of `RUN_USING_ENCODER`, many may mistake that it is necessary to use this mode for the encoders to work at all, but this is not true. Instead, `RUN_USING_ENCODER` enables velocity feedback using the encoder. If you are using an external PID controller such as one that you implement, generally, it is recommended that you use `RUN_WITHOUT_ENCODER`.

For official documentation on the built in PID controller, see here[211]

Debugging Built-In PID Controller

Problem	Solution
Motor goes at Full Speed regardless of velocity setpoint	Most of the time this occurs when one of two things occurs: #1: Your encoder is not connected properly. Diagnosis: Log your encoder position to telemetry, if the position oscillates between 0 - 1 make sure you have the correct cable and it is seated correctly. #2: Your motor is going in the wrong direction. Diagnosis: Log your velocity to telemetry, if you have a positive reference velocity and the output is negative or vice versa then your motor is plugged in backwards.
Motor does not reach full speed with `.setPower`	Use the `.setVelocity` method as part of DcMotorEx or use `RUN_WITHOUT_ENCODER` with an external PID controller.

PID Controller Sample Rate

For teams who desire the most performance out of their PID controller, it is essential to consider the Sample rate of the controller. The Sample rate is when the controller updates its output given new sensor data. Higher Sample rates allow for more stable control and allow for the usage of more significant PID coefficients to reduce settling time. See this video[212] to see how sample rate effects stability in a practical motor control example. The inbuilt PID controller is locked at a 20hz refresh rate (50ms sample rate). Many top FTC teams optimize their robot loops to run at up to 80hz, achieving *much* more stable control with an external PID.

[211] https://docs.revrobotics.com/duo-control/programming/using-encoder-feedback
[212] https://www.youtube.com/watch?v=fusr9eTceEo&t=133s

Feedforward Control

Feedforward control is a method of what is known as "open-loop" control. This is the opposite of closed-loop control and the primary difference is that feedforward does not actively use sensors to control the system. Instead it "predicts" the desired input based on a model.

Typically feedforward is used to control either rates of change or combat known disturbances from your system.

Feedforward is very powerful because it is immune to noise or other sensor errors. This is because it is not actively measuring the system, but instead predicting the desired input. However, this also means that it is not very good at correcting for errors. This is why it is often used in conjunction with a closed-loop controller such as PID.

Kv Ka Feedforward Model

The most common feedforward and the one used by libraries such as road-runner is the Kv-Ka feedforward model:

$$f(t) = K_v \cdot \text{Velocity} + K_a \cdot \text{Acceleration}$$

Where K_v is the velocity gain, K_a is the acceleration gain, and $f(t)$ is the feedforward output sent to your motors.

These gains can be estimated by giving the controller a series of ramp inputs (such as those computed with a motion profile), measuring the output, and then changing these gains till the robot matches the desired motion.

Note: The gains will change based on the robot's mass, friction, and other factors. It is recommended to re-estimate these gains every time you make a significant change to your robot.

Kv Ka Feedforward Pseudocode

```
while True:
     targetVelocity = getTargetVelocity(time)
     targetAcceleration = getTargetAcceleration(time)
     output = targetVelocity * Kv + targetAcceleration * Ka;
```

Static Friction Feedforward

In every system there is bound to be some amount of static Friction. This means that the robot mechanism will not move until a certain amount of power is applied. This can be modeled by adding a constant feedforward term in the direction you want to move.

```
while True:
     error = desire_position - current_position;
     sign = signum(error) # sign of error, will be -1, 0, or 1
     output = sign * staticFriction + PID(error); # PID Controller + Friction␣
↪Feedforward
```

Motion Profiles

Tip: Motion profiles are *not* a specific type of control loop, but rather a technique that works well in combination with other control loops such as PID and feedforward.

Motion profiling is a technique popularized in FRC® that is starting to find its way to FTC. A motion profile is a function used to change the speed of a power transmission system in a controlled and consistent way by changing desired speed gradually rather than instantaneously.

Let's illustrate this with an example: say you want your drivetrain, which is initially unmoving, to drive forward at full speed. Ordinarily, you would set all drivetrain motors to full power in the code. However, this can be problematic because even though you tell the motors to move at full speed instantaneously, the drivetrain takes time to get to full speed. This can lead to uncontrolled movements which have the potential to make autonomous less consistent and, perhaps more importantly, damage mechanisms.

Motion profiling attempts to solve this issue.

Advantages

- More controlled and predictable movements
- Reduces rapid change of applied motor voltage

Disadvantages

- Can be slower

There are two main types of motion profiles: **Trapezoidal** profiles and **S-Curve** profiles. Trapezoidal profiles accelerate the system at a constant rate, and S-Curve profiles assume jerk (the speed acceleration changes) is constant. Given that S-Curve profiles are not optimal for controlling 2d trajectories (such as driving) and exist to reduce slippage (which usually only occurs when driving in FTC), trapezoidal profiles are recommended for most FTC applications.

Trapezoidal profiles get their name from the shape of the graph of velocity over time:

Here is some pseudocode for a trapezoidal profile:

```
while True:
   current_velocity = get_current_velocity()
   current_time = get_current_time()

   direction_multiplier = 1

   if position_error < 0:
       direction_multiplier = -1

   # if maximum speed has not been reached
   if MAXIMUM_SPEED > abs(current_velocity):
       output_velocity = current_velocity + direction_multiplier * MAX_ACCELERATION *␣
↪(current_time - previous_time)
       output_acceleration = MAX_ACCELERATION

   #if maximum speed has been reached, stay there for now
```

(continues on next page)

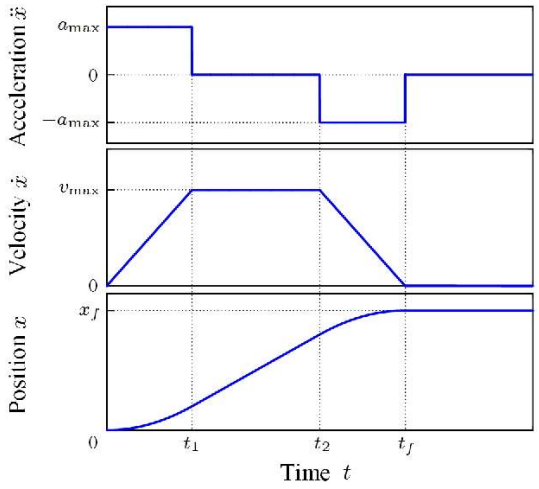


Fig. 1: These are the “magic functions” for velocity and acceleration over time alluded to in the feedforward section.

(continued from previous page)

```
  else:
      outputVelocity = MAXIMUM_SPEED
      outputAcceleration = 0

  #if we are close enough to the object to begin slowing down
  if position_error <= (output_velocity * output_velocity) / (2 * MAX_ACCELERATION)):
      output_velocity = current_velocity - direction_multiplier * MAX_ACCELERATION *␣
↪(current_time - previous_time)
      output_acceleration = -MAX_ACCELERATION

  previous_time = current_time
```

The results of the above pseudocode are then used in a feedforward and / or PID loop to control the position of the system in a smooth and predictable way.

A more advanced example of the math for motion profile generation as used in the Road Runner library[213] can be found in this Jupyter Notebook[214].

[213] https://github.com/acmerobotics/road-runner
[214] https://mybinder.org/v2/gh/acmerobotics/road-runner/HEAD?filepath=doc%2Fnotebook%2Froad-runner-lite.ipynb

10.3.2 Finite State Machines

Finite State Machines (FSM) are often used while programming in order to allow for more complex series of actions. This is especially useful when one needs multiple tasks to run at the same time, because it allows for tasks to depend on each other's execution in a non-linear fashion.

What is a Finite State Machine?

The name of a finite state machine is very descriptive; it's a state machine, with a finite number of states. It can be in one state at a time, and can transition to a different state once something happens. The Wikipedia[215] example of a turnstile explains the concept very well.

Implementation

Naive Implementation

When first learning about FSMs, it is quite common for programmers to try and use them. Often times, they try to apply an FSM to their autonomous programs by segmenting their autonomous into a giant `switch` statement, and it often looks something like this:

```
while (opModeIsActive()) {
    switch (state) {
        case DETECT_SKYSTONE:
            // skystone detection code here
            int position = detectSkystone();

            if (position == 0) {
                state = SKYSTONE_POS_0;
```

(continues on next page)

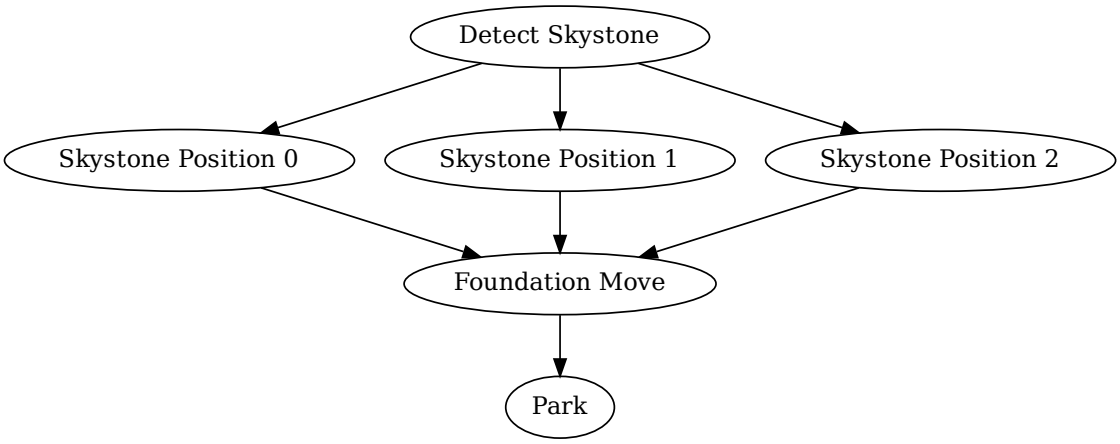
[215] https://en.wikipedia.org/wiki/Finite-state_machine#Example:_coin-operated_turnstile

(continued from previous page)

```
            }
            else if (position == 1) {
                state = SKYSTONE_POS_1;
            }
            else {
                state = SKYSTONE_POS_2;
            }
            break;
        case SKYSTONE_POS_0:
            // skystone position 0 here
            doSkystone(0);
            state = MOVE_FOUNDATION;
            break;
        case SKYSTONE_POS_1:
        case SKYSTONE_POS_2:
            // etc etc
            break;
        case MOVE_FOUNDATION:
            // foundation move code
            state = PARK;
            break;
        case PARK:
            // park the bot
            break;
    }
}
```

This however does not really have any benefits compared to if the programmer had simply put each of the code's segments into functions, and executed them in order. In fact, often times, programmers will structure their code like this instead of splitting their code up into functions. The result is an autonomous that's more difficult to debug, and ultimately harder to fix on the fly during a competition or other time crunch.

If one drew out the state transition diagram for each of the states, for the autonomus above it'd be very linear, and the state transitions always occur because the section of the code finished:



In fact, in many implementations, making state transitions for any other reason is often difficult because the code executes linearly and is only in a loop to rerun the switch statements. (Often times, this means the

code has a hard time reacting to a stop request in the middle of autonomous.)

> **Warning:** It is unadvisable to write code like this. If your autonomous is synchronous, it is preferable to split your code up into functions and run them in order, as this will be easier to understand and edit on the fly.

Useful Implementation

FSMs are the right tool to use when a robot needs to complete multiple tasks at once; a common example of this is when a robot should have automation in teleop, but still have control over the drivetrain.

Often times, teams have issues because their teleop executes in a loop and their servo logic has sleeps in it. But, we can avoid this if we write code in an **asynchronous** fashion - where instead of waiting for a task to complete before doing the next one, tasks are performed at the same time, and each task's state is checked without stopping the other tasks from executing.

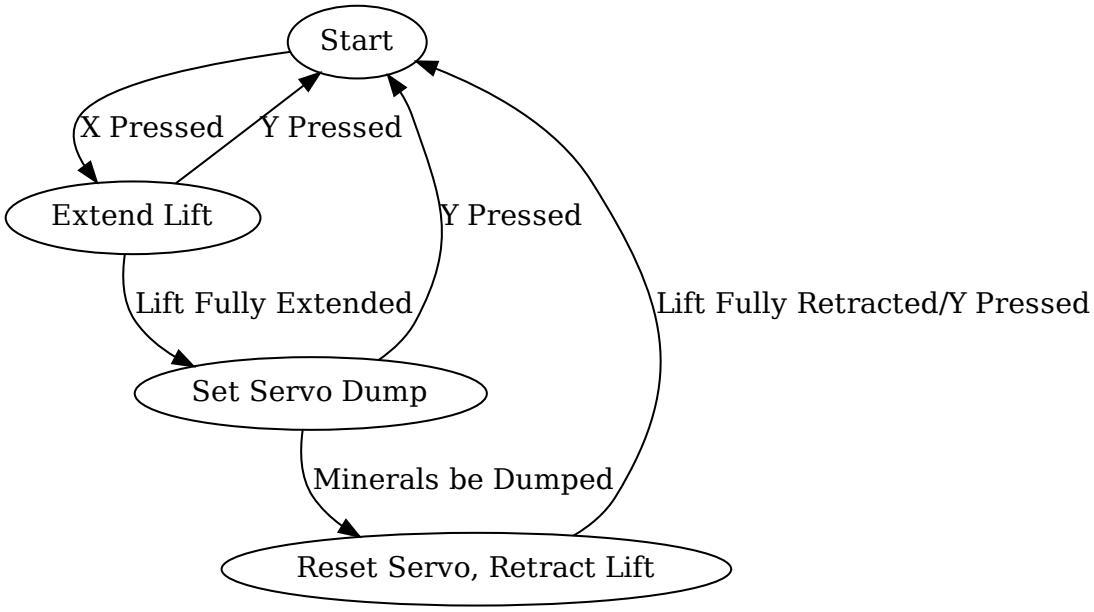
An example of this would be that if one had a robot similar to Gluten Free's Rover Ruckus Robot[216], and one wanted to automate the scoring lift so that the drivers don't have to think while the bot deposits the minerals. There are two parts of the bot that are relevant to us in this exercise: the angled scoring lift, and the servo that tips the dumper so the minerals fall out. The goal is to be able to push a button, and then the bot will:

- extend the lift,
- at full lift extension, angle the mineral bucket servo to deposit the minerals,
- wait for the minerals to fall out,
- reset the servo to the original position
- retract the lift

If the drivers press a specific other button, we will stop executing the actions above as a failsafe - in case the robot is breaking somehow and the drivers need to take manual control. All the while, the drivers should still be able to control our drivetrain so we can make adjustments. Now, of course, this is a bit simplified (and probably not entirely what GF did), but it will do for now.

Before anything is programmed, it may be useful draw out the state diagram for this to get a better understanding of what we the robot should actually be doing. This can also add to a Control Award submission.

[216] https://www.youtube.com/watch?v=NQvhvYJXVMA



Notice how resetting the dump servo and retracting the lift share a state. That's because the robot doesn't need to wait for the servo to reset before moving the lift down; they can both happen at once.

Now, let's get into actually implementing the code for this. In a traditional `OpMode`, which is commonly used for teleop, code runs repeatedly in a `loop()` function, so instead of waiting for a state transition to happen directly, the code will repeatedly check on each `loop()` call if it should perform a state transition. This kind of "update our state" pattern keeps code from blocking the rest of the `loop()` code from running, such as the drivetrain.

```
/*
 * Some declarations that are boilerplate are
 * skipped for the sake of brevity.
 * Since there are no real values to use, named constants will be used.
 */

@TeleOp(name="FSM Example")
public class FSMExample extends OpMode {
    // An Enum is used to represent lift states.
    // (This is one thing enums are designed to do)
    public enum LiftState {
        LIFT_START,
        LIFT_EXTEND,
        LIFT_DUMP,
        LIFT_RETRACT
    };

    // The liftState variable is declared out here
    // so its value persists between loop() calls
    LiftState liftState = LiftState.LIFT_START;

    // Some hardware access boilerplate; these would be initialized in init()
```

(continues on next page)

(continued from previous page)

```
    // the lift motor, it's in RUN_TO_POSITION mode
    public DcMotorEx liftMotor;

    // the dump servo
    public Servo liftDump;
    // used with the dump servo, this will get covered in a bit
    ElapsedTime liftTimer = new ElapsedTime();

    final double DUMP_IDLE; // the idle position for the dump servo
    final double DUMP_DEPOSIT; // the dumping position for the dump servo

    // the amount of time the dump servo takes to activate in seconds
    final double DUMP_TIME;

    final int LIFT_LOW; // the low encoder position for the lift
    final int LIFT_HIGH; // the high encoder position for the lift

    public void init() {
        liftTimer.reset();

        // hardware initialization code goes here
        // this needs to correspond with the configuration used
        liftMotor = hardwareMap.get(DcMotorEx.class, "liftMotor");
        liftDump = hardwareMap.get(Servo.class, "liftDump");

        liftMotor.setTargetPosition(LIFT_LOW);
        liftMotor.setMode(DcMotor.RunMode.RUN_TO_POSITION);
    }

    public void loop() {
        liftMotor.setPower(1.0);

        switch (liftState) {
            case LIFT_START:
                // Waiting for some input
                if (gamepad1.x) {
                    // x is pressed, start extending
                    liftMotor.setTargetPosition(LIFT_HIGH);
                    liftState = LiftState.LIFT_EXTEND;
                }
                break;
            case LIFT_EXTEND:
                 // check if the lift has finished extending,
                 // otherwise do nothing.
                if (Math.abs(liftMotor.getCurrentPosition() - LIFT_HIGH) < 10) {
                    // our threshold is within
                    // 10 encoder ticks of our target.
                    // this is pretty arbitrary, and would have to be
                    // tweaked for each robot.

                    // set the lift dump to dump
                    liftDump.setTargetPosition(DUMP_DEPOSIT);

                    liftTimer.reset();
                    liftState = LiftState.LIFT_DUMP;
                }
```

(continues on next page)

(continued from previous page)

```
                break;
            case LIFT_DUMP:
                if (liftTimer.seconds() >= DUMP_TIME) {
                    // The robot waited long enough, time to start
                    // retracting the lift
                    liftDump.setTargetPosition(DUMP_IDLE);
                    liftMotor.setTargetPosition(LIFT_LOW);
                    liftState = LiftState.LIFT_RETRACT;
                }
                break;
            case LIFT_RETRACT:
                if (Math.abs(liftMotor.getCurrentPosition() - LIFT_LOW) < 10) {
                    liftState = LiftState.LIFT_START;
                }
                break;
            default:
                 // should never be reached, as liftState should never be null
                 liftState = LiftState.LIFT_START;
        }

        // small optimization, instead of repeating ourselves in each
        // lift state case besides LIFT_START for the cancel action,
        // it's just handled here
        if (gamepad1.y && liftState != LiftState.LIFT_START) {
            liftState = LiftState.LIFT_START;
        }

        // mecanum drive code goes here
        // But since none of the stuff in the switch case stops
        // the robot, this will always run!
        updateDrive(gamepad1, gamepad2);
    }
}
```

10.3.3 Kinematics

Kinematics is the application of geometry to the control of various robot mechanisms. Kinematics equations are used to control mechanisms by providing specific inputs to achieve a desired output.

Many of the kinematics equations here were taken from Controls Engineering in the FIRST Robotics Competition (book)[217] and Mobile Robot Kinematics for FTC (paper)[218], which contain the relevant derivations. While only tank (differential drive) and mecanum kinematics equations are shown here, these sources also contain derivations for other mechanisms such as swerve and dead wheel odometry.

[217] https://file.tavsys.net/control/controls-engineering-in-frc.pdf
[218] https://github.com/acmerobotics/road-runner/blob/master/doc/pdf/Mobile_Robot_Kinematics_for_FTC.pdf

Forward vs Inverse Kinematics

Mechanisms may have different sets of equations for their forward and inverse kinematics. Forward kinematics are the equations used to determine the state of a system given the state of its outputs, whereas inverse kinematics determines the output of a system given the desired state. For example, in a drivetrain, forward kinematics would determine body velocity of the robot based on the individual velocities of the wheels, whereas inverse kinematics would determine the required wheel velocities for a desired body velocity.

Tank (Differential Drive)

A tank, or differential drive, is a drivetrain consisting of two sets of wheels on either side of the robot that are independently driven. These are described under further detail in the *Tank (Skid-Steer) Drivetrains* (page 92) section.

Variables

The following variables are used in this section.

- v_r denotes the linear velocity of the right wheel(s)
- v_l denotes the linear velocity of the left wheel(s)
- v_f denotes the forward velocity of the robot, relative to itself
- ω denotes the rotational velocity of the robot in radians/second
- r_b denotes the base track radius, or the distance between the wheel and center of the robot (half of the distance between wheels)

Warning: These variables, with the exception of ω, represent **linear** velocities NOT **rotational** velocities. Wheel rotational velocity in radians/second can be converted to linear velocity by multiplying by the wheel's radius.

Positive rotational velocity (ω) will spin the robot COUNTERCLOCKWISE when viewed from above.

Forward Kinematics

The forward kinematics of a tank drive relate the velocity of the wheels to the forward and rotational velocities of the robot, relative to itself. The forward velocity v_f and the rotational velocity v_θ is:

$$v_f = \frac{v_r + v_l}{2}$$
$$\omega = \frac{v_r - v_l}{2r_b}$$

Inverse Kinematics

The inverse kinematics of a tank drive relate the desired velocity of the robot to the velocity required of the wheels. These velocities are as follows:

$$v_r = v_f + r_d \cdot \omega$$
$$v_l = v_f - r_d \cdot \omega$$

Mecanum Drive

Variables

Mecanum kinematics uses the same variables as differential drive, except with four wheel velocity variables and an additional robot velocity vector for the left to right velocity.

- v_{fr} denotes the linear velocity of the front (leading) right wheel
- v_{br} denotes the linear velocity of the back (trailing) right wheel
- v_{fl} denotes the linear velocity of the front (leading) left wheel(s)
- v_{bl} denotes the linear velocity of the back (trailing) left wheel(s)
- v_f denotes the forward velocity of the robot, relative to itself.
- v_s denotes the strafe (sideways) velocity of the robot, relative to itself.
- ω denotes the rotational velocity of the robot in radians/second
- r_b represents the base track radius, or the distance between the wheel and center of the robot (half of the distance between wheels)

Warning: These variables, with the exception of ω, represent **linear** velocities NOT **rotational** velocities. Wheel rotational velocity in radians/second can be converted to linear velocity by multiplying by the wheel's radius.

Positive rotational velocity (ω) will spin the robot COUNTERCLOCKWISE when viewed from above.

Forward Kinematics

The forward kinematics of a mecanum drive relate the velocity of the wheels to the forward, strafe, and rotational velocities of the robot, relative to itself. These are as follows:

$$v_f = \frac{v_{\text{fr}} + v_{\text{fl}} + v_{\text{br}} + v_{\text{bl}}}{4}$$
$$v_s = \frac{v_{\text{bl}} + v_{\text{fr}} - v_{\text{fl}} - v_{\text{br}}}{4}$$
$$\omega = \frac{v_{\text{br}} + v_{\text{fr}} - v_{\text{fl}} - v_{\text{bl}}}{4 * 2r_b}$$

Inverse Kinematics

The inverse kinematics of a mecanum drive relate the desired velocity of the robot to the velocity required on the wheels. These are as follows:

$$
\begin{aligned}
v_{fl} &= v_f - v_s - (2r_b \cdot \omega) \\
v_{bl} &= v_f + v_s - (2r_b \cdot \omega) \\
v_{br} &= v_f - v_s + (2r_b \cdot \omega) \\
v_{fr} &= v_f + v_s + (2r_b \cdot \omega)
\end{aligned}
$$

10.3.4 Odometry

Odometry is a form of localization that uses data from sensors like encoders to derive an estimated position relative to a starting point. Localization is a means for being able to locate the position of the bot at some point in time. Odometry is especially useful in autonomous programs because it allows for easier implementation of different tasks on the field due to understanding one's position.

Pose

We refer to pose, which is the position of some body (like a bot), normally in the context two-dimensional space, as the movement of the robot is generally constrained to a single plane. We notate the robot's pose as $\vec{x}$. A pose contains two entries: the robot's position and heading; position is generally in Cartesian coordinates, so the pose can be represented with x, y, and θ. A "heading" is a term for the direction towards which the front of the robot is facing. Because of this, the robot's coordinate frame is set up such that the global x-axis is lined up with the 0 heading.

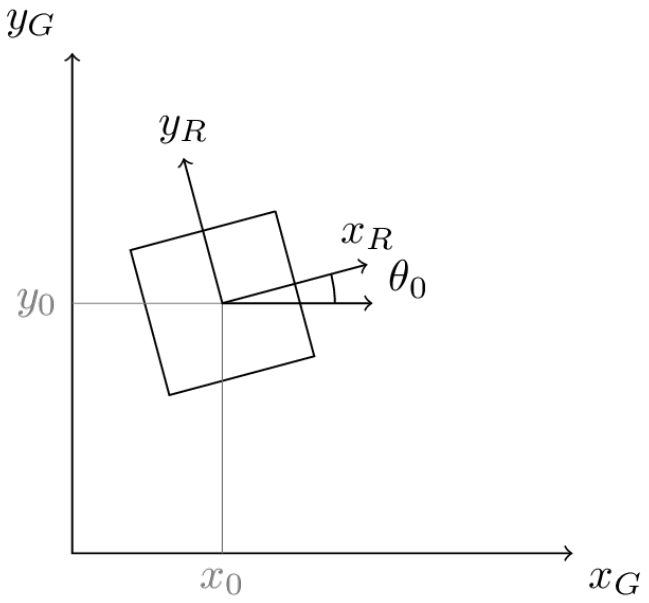


Fig. 2: Road Runner Coordinate Frame Documentation[219]

[219] https://acme-robotics.gitbook.io/road-runner/tour/coordinate-frame

We can refer to the current pose ($\vec{x}_0$) of the robot as $\begin{pmatrix} x_0 \\ y_0 \\ \theta_0 \end{pmatrix}$. This is just fancy notation for a point on the field (x_0, y_0) with a specified orientation of the robot–the heading θ_0. A pose generally has some beginning origin in the coordinate frame.

Updating the Pose

The change in pose over some very small amount of time is $\Delta\vec{x}$. The difference in time between the current pose and the last pose should be as small as possible to improve the approximations for the math. Teams should update their robot pose every cycle of their control loop.

Updating the pose is as simple as adding the transformed change to the previous pose where $\varphi = \Delta\theta$

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \\ \theta_0 \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \\ \varphi \end{pmatrix}$$

The idea of odometry is to use sensor data and math to form an approximation for the robot's pose over time.

Finding the Change in Position

In order to determine the current location of the robot and update its pose, the change must be calculated using data read from the sensors. For a robot, there will be three possible sensors that you can use: two that are parallel with the robot's body in the x-direction and one that is aligned with the y-direction of movement (perpendicular to the drive wheels).

Angle and Displacement

The displacement (or change in position) of the left sensor is Δx_l and the displacement of the right sensor is Δx_r. The lateral distance between these two sensors is called the trackwidth, notated as L. This is very important for determining angle for turning approximations. This value will need to be tuned, which means tested repeatedly and then brought to some converging value that is close to the actual measurement.

Deriving the value of φ then becomes simple:

$$\varphi = \frac{\Delta x_l - \Delta x_r}{L}$$

To perform later calculations, we need to know the displacement of the robot in the x-direction relative to its center rather than the two parallel sensors. To do this, we take the average to derive Δx_c, or the center displacement:

$$\Delta x_c = \frac{\Delta x_l + \Delta x_r}{2}$$

The final displacement we need before we can determine the change in pose is the horizontal displacement $\Delta x_\perp$. This is the displacement of the perpendicular sensor Δx_h with a correction for forward offset F. In order to get accurate approximations, the forward offset needs to be considered. When the sensor is closer to the back, the offset is negative, but when it is closer to the front, it is positive. This is to account for the change in its position based on point-turns.

[220] https://learnroadrunner.com/dead-wheels.html#three-wheel-odometry

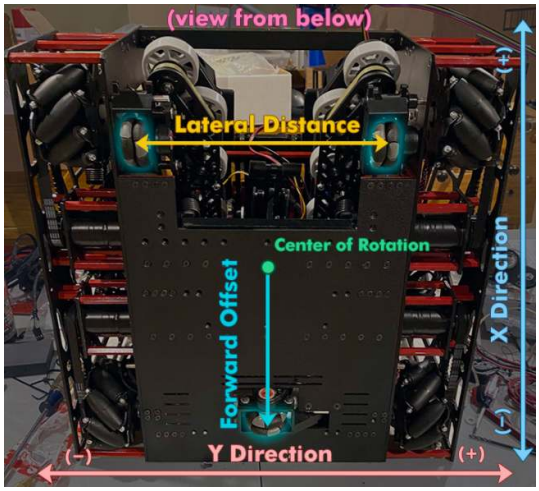


Fig. 3: 17508 Rising Tau's 2019/20 Skystone Bot[220]

As a result of this, we can define our horizontal displacement as:

$$\Delta x_{\perp} = \Delta x_h - (F * \varphi)$$

Note: If you do not have perpendicular sensors, which are not required if the robot cannot move in the lateral direction, $\Delta x_{\perp}$ is not necessary.

For this value, use 0 if you do not have a horizontal sensor.

Robot-Relative Deltas

Let's come up with a simplified, nonoptimal way to calculate our robot-relative pose deltas which we can then transform into field-relative coordinate changes. To perform this we need to transform the robot-relative deltas via a rotation matrix where we rotate the relative pose difference by the original heading. We can derive the values of Δx and Δy.

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \varphi \end{pmatrix} = \begin{pmatrix} \cos(\theta_0) & -\sin(\theta_0) & 0 \\ \sin(\theta_0) & \cos(\theta_0) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \Delta x_c \\ \Delta x_{\perp} \\ \varphi \end{pmatrix}$$

From this, we can calculate our field-relative change in pose:

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \varphi \end{pmatrix} = \begin{pmatrix} \Delta x_c \cos(\theta_0) - \Delta x_{\perp} \sin(\theta_0) \\ \Delta x_c \sin(\theta_0) + \Delta x_{\perp} \cos(\theta_0) \\ \varphi \end{pmatrix}$$

Note: This method of approximating position is known as Euler integration, but we are using it for strict pose deltas instead of integrating the velocity (essentially, this is a very simplified version of the original theory).

Warning: This is for advanced programmers; while implementing this from scratch is a great learning exercise, it is likely not the optimal way to get the best auto. There are several *resources* (page 342) out there for producing great, well-tested, and easy-to-implement odometry.

Odometry Pseudocode

```
while robot_is_active():
   delta_left_encoder_pos = left_encoder_pos - prev_left_encoder_pos
   delta_right_encoder_pos = right_encoder_pos - prev_right_encoder_pos
   delta_center_encoder_pos = center_encoder_pos - prev_center_encoder_pos

   phi = (delta_left_encoder_pos - delta_right_encoder_pos) / trackwidth
   delta_middle_pos = (delta_left_encoder_pos + delta_right_encoder_pos) / 2
   delta_perp_pos = delta_center_encoder_pos - forward_offset * phi

   delta_x = delta_middle_pos * cos(heading) - delta_perp_pos * sin(heading)
   delta_y = delta_middle_pos * sin(heading) + delta_perp_pos * cos(heading)

   x_pos += delta_x
   y_pos += delta_y
   heading += phi

   prev_left_encoder_pos = left_encoder_pos
   prev_right_encoder_pos = right_encoder_pos
   prev_center_encoder_pos = center_encoder_pos
```

Using Pose Exponentials

This method uses differential equations to solve the nonlinear position of the robot given constant curvature. Euler integration assumes that the robot follows a straight path between updates, which can lead to inaccurate approximations when traveling around curves. If you are interested in the math itself, we recommend you check out this book[221] for FRC® controls.

We'll treat the way it is solved in this page as a black box, and derive the formula by implementing a correction for this nonlinear curvature into our Euler integration robot-relative deltas equation:

$$\begin{pmatrix}\Delta x\\ \Delta y\\ \varphi\end{pmatrix} = \begin{pmatrix}\cos(\theta_0) & -\sin(\theta_0) & 0\\ \sin(\theta_0) & \cos(\theta_0) & 0\\ 0 & 0 & 1\end{pmatrix} \begin{pmatrix}\frac{\sin(\varphi)}{\varphi} & \frac{\cos(\varphi)-1}{\varphi} & 0\\ \frac{1-\cos(\varphi)}{\varphi} & \frac{\sin(\varphi)}{\varphi} & 0\\ 0 & 0 & 1\end{pmatrix} \begin{pmatrix}\Delta x_c\\ \Delta x_\perp\\ \varphi\end{pmatrix}$$

[221] https://file.tavsys.net/control/controls-engineering-in-frc.pdf

Resources for Odometry

There are several great resources out there for odometry. We highly recommend Road Runner[222]. For the math behind Road Runner (which utilizes pose exponentials), you can also read Ryan's paper[223]. An additional resource for Road Runner is Learn Road Runner[224] which is a step-by-step procedural guide that explains how to work with the Road Runner quickstart[225].

We also recommend Tyler's book[226] as it goes into great detail about various controls in *FIRST*® robotics.

If you're using other resources, it is important that you do not use ones that utilize Euler integration as it is less optimal for real life approximations of robot pose.

10.4 Advanced Control System

This section covers advanced details about the SDK and control system.

10.4.1 Control System Electronics

This page contains a breakdown of the known electronics in the control system, as well as any notes regarding them

Expansion Hub Internals

> **Warning:** Don't take apart a Control or Expansion Hub unless you really know what you are doing. They can be damaged in the process, especially if one does not know how to properly reassemble it. THIS WILL VOID YOUR WARRANTY!

Lynx Board

"Lynx" is the codename of the board within the Expansion Hub and Control Hub that interacts with hardware. References to "Lynx" are made in the FTC® SDK refer to this board. It appears to have been developed by both REV and DEKA, possibly for use in *FIRST*® Global (judging by the *FIRST* Global silkscreen on the PCB).

> **Warning:** Don't take apart a Control or Expansion Hub unless you really know what you are doing. They can be damaged in the process, especially if one does not know how to properly reassemble it.

[222] https://acme-robotics.gitbook.io/road-runner/
[223] https://github.com/acmerobotics/road-runner/blob/master/doc/pdf/Mobile_Robot_Kinematics_for_FTC.pdf
[224] https://learnroadrunner.com/
[225] https://github.com/acmerobotics/road-runner-quickstart
[226] https://file.tavsys.net/control/controls-engineering-in-frc.pdf

Fig. 4: A Lynx board that was removed from its case

Processor

The main processor of the Expansion Hub is a Texas Instruments ARM Cortex M4 running at 80 MHZ.

Ports

The expansion hub has the following ports

- Two UART Debug Ports
  - The top port outputs a continuous high speed CSV stream of data from the various subsystems in the hub.
  - The bottom port outputs data at a baud rate of 115200 at a user specified verbosity.
- Four I2C Ports
  - These ports are 100/400 kHz compliant and are connected to a separate bus so there is no need to be concerned about address collisions
  - There are integrated pull up resistors on this port
- Eight DIO Ports
  - 3.3V only, current limited, can briefly supply more than the rated current spec. Pulled up internally.
- Four Analog Ports
  - 5V compliant, you can use a level shifter to supply 5v to the sensor, but take care the analog line bypasses the level shifter. The VIN and GND lines must pass through the level shifter for it to work however.
- Six Servo Ports
  - The 5V supply on the servo ports is default OFF, and will only enable once a servo is used. All six 5V pins are switched on and off together, and disabling PWM on one port will switch all of the ports off unless another servo is used.
- Two 5V Power Ports
  - The 5V supply is shared with the servos
- Four Motor Ports

  - Current limited and have overtemperature protection on the chip. The output does not brake the h-bridge during the PWM off cycle. What this means is that there might be linearity issues while deaccelerating, because of momentum in the motor.
  - The motor driver is a ST Microelectronics VNH5050 motor driver IC, which is capabile of handling well over the maximum amount of current an FTC motor can draw. It has integrated current sensing and has been used since the modern robotics era. It has built in thermal and current safety limits. This motor controller has been used for years, and may even date back to the early HiTechnic based control system.

- Four Encoder Ports
  - **IMPORTANT: Only two of the encoder ports (Ports 0 and 3) appear to be connected via hardware and are reliable at high speed**. There are two methods of connecting an encoder internally to the texas instruments microprocessor, through hardware and software. Hardware ports use the integrated quadrature decoder chip and are extremely accurate at high speed, whereas encoders decoded in software are not reliable at high speeds. As a result, high Count Per Revolution (CPR) encoders, those with more than 4000 counts per revolution, should NOT be used on ports 1 and 2, the ports connected in software.
- Two XT30 Connectors
  - Care should be taken when moving around cables as these connectors have been known to fail and break off the board.
  - In addition, XT30s will wear and get loose with time. Hot glue is recommended if cables come loose.
- Mini USB B Connector
  - Capable of full speed USB 2.0, and 5V output to charge a phone. The 5v output may be unreliable, it depends on the hub.
- Internal Connector
  - This connector is what is used to connect to the android daughterboard. It presumably has UART capability as well as power and ground. All expansion hubs have this connector, whether or not they are a control hub. Presumably this is because the control hub and REV hub were developed together at the same time.

Control Hub

The Control Hub is an Expansion Hub with an embedded Android single-board computer daughterboard connected to it. This enables it to not need a separate Robot Controller phone, as the daughterboard functions as the Robot Controller. Internally, LynxCommands are sent over from the daughterboard to the Lynx board over an internal UART connection.

The control hub daughterboard contains a RK3328 Quad-core ARM Cortex-A53 running a custom version of Android/Linux. This software is open source, but has nothing notible except for some software that manages the wifi access point. It has no thermal spreading, such as heat sinks, on the chip, which can cause thermal throttling when heavy applications such as tensorflow are used.

Warning: Don't take apart a Control or Expansion Hub unless you really know what you are doing. They can be damaged in the process, especially if one does not know how to properly reassemble it.

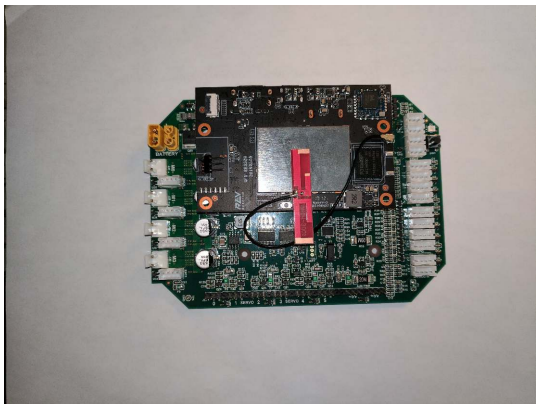


Fig. 5: The single board computer and Lynx board from a Control Hub

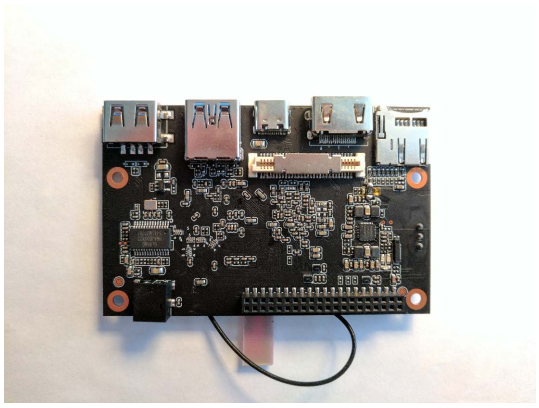


Fig. 6: The android board, removed from a control hub

10.4.2 SDK Communication

When using any method in the FTC® SDK that accesses hardware, be that setting motor power, reading an encoder, a sensor, etc., a LynxCommand is sent.

Note: LynxCommands are not sent directly from the Robot Controller to an Expansion Hub through USB; in an expansion hub they are sent through USB to an FTDI, which converts the USB signal to a UART one. In the control hub, this USB step is skipped, instead the control hub board sends the data directly over UART to the internal expansion hub.

Warning: LynxCommands being blocking (and more specifically a master lock being present on each usb device) means that multithreading hardware calls is at best not helpful and typically harmful to performance.

If an Android phone and Expansion Hub is used, LynxCommands are sent over USB; however if a Control Hub is used, LynxCommands are sent over UART. This is very important, not just because of the increased reliability with UART instead of USB, but also because LynxCommands take approximately 3 milliseconds over USB and approximately 2 milliseconds over UART.

Any expansion hubs connected via RS485 receive their commands via that connector. Lynx hubs do not have to retransmit packets, so the added latency from this process isn't significant, but there will be some added latency. Up to 255 expansion hubs can be connected together in theory.

Note: Interacting with I2C devices takes significantly longer; upwards of 7 milliseconds over USB. However, this is not because each LynxCommand takes longer, but because multiple LynxCommands must be sent to interact with I2C.

Please note that since version 5.5 of the SDK, I2C calls on the Control Hub are much faster than those on the Expansion Hub. This is because the polling rate was dramatically increased, which can cut down on unnecessary wasted time.

10.4.3 SDK Motors

The SDK offers several methods of controlling and communicating with motors, as well as a couple of hidden methods that can be easily accessed.

General Explanation

Motor Controller

All motor ports are controlled with what is called an H-bridge motor controller, a circuit that can be used to vary the output voltage as well as signage (negative or positive) of the voltage. Negative voltage through a DC Motor reverses a motor, whereas positive motor will make the motor go forward. SDK Motor power (from -1 to 1), represents a multiplier of the input voltage that is output through the motor port. The different voltages are created via PWM, where the port is turned on and off rapidly to create a lower average voltage.

In addition, the zero power behavior of the motor, that is the behavior of the motor when no power is applied, can be configured. In FLOAT mode, the motor controller simply turns off, providing minimal additional re-

sistance. In BRAKE mode, the two motor leads are shorted together internally. Due to the inherent property that all DC brushed motors generate electricity when the shaft spins, shorting the leads causes a reverse power that stops the motor quickly and is resistant to external forces.

Note: The motor controller uses the input voltage, this means on a 13 volt battery, setting the power to 1 will create an output voltage of 13 volts. Similarly, on an 11 volt battery, a power of 1 will create an output voltage of 11 volts.

Motor Encoder

Important: There is no real standardized terminology when dealing with quadrature encoders. Here, we use the terms "count" and "tick" to represent a single rising or falling action in the quadrature wave. You may also see some datasheets list "pulses", which can indicate anything from 1 "count" to 4 "counts". Be careful when reading datasheets!

FTC® Encoders use the two wire quadrature format for transmitting relative encoder information. In quadrature, there are two signal wires, A and B. When moving, both A and B generate square waves that are 90 degrees apart, that is one square wave starts half way through the other square wave, and ends half way through the other square wave. When traveling in one direction, the A wire square wave leads the B wire square wave, and in the other direction the B wire square wave leads the A wire square wave. The two waves are combined in XOR to produce the output wave, where each rising and falling action is one "tick", and the faster the wave the faster the encoder is moving.

The REV hub counts the pulses and calculates velocity by using a 5 value "ring buffer", which has a new value added to it every 10 ms. These 5 values are then used to calculate the current velocity.

Warning: It is recommended that quadrature encoders be hooked up to special hardware decoded ports to allow them to be read correctly. The expansion hub contains special hardware for reading quadrature encoders, but because there are only two of those controllers, two of the pins are hooked up in "software" instead (the ports are decoded in software instead of in hardware). Effectively, this means that ports 0 and 3, the two ports connected to the special quadrature ports, will always read accurately. **Ports 1 and 2 are connected to the less accurate "software" ports, meaning that with high CPR encoders (encoders that produce more than 4000 counts per revolution, such as the REV Through Bore Encoder or Talon SRX Encoder) they can "lose steps" and drift.**

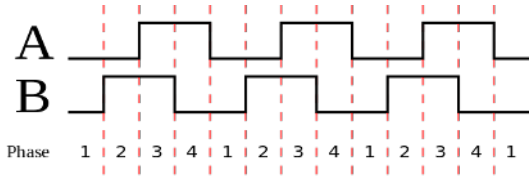


Fig. 7: An example of a quadrature wave, with channel A leading channel B. Each count is a "count" or "tick"

Hidden Methods

DcMotorEx

All REV hub DC motors are instances of `DcMotorEx`, which exposes some more methods to the user, such as velocity control and current draw measurement.

Note: There is no downside to using DcMotorEx, in order to convert a DcMotor to a DcMotorEx the user simply just needs to cast the DcMotor returned by the `hardwareMap` to a DcMotorEx.

Tips

- While current readings are not bulk read, current alerts (`isMotorOverCurrent()`) is bulk read.
- PID/PIDF coefficients use internal units for the output, a two byte short from -32767 to 32767, instead of the user -1 to 1.
- The default `getVelocity()` method returns the encoder velocity in ticks per second.
- The `RunMode` `STOP_AND_RESET_ENCODERS` is not actually a run mode. Instead, it just sets the power to zero and sends a `LynxResetMotorEncoderCommand`. This command can be issued manually if one wants to easy reset a motor encoder without changing the run mode.

10.4.4 SDK Servos

The SDK offers several methods of controlling and communicating with servos, as well as a couple of hidden methods that can be easily accessed.

General Explanation

PWM Explanation

Servos are controlled via a PWM signal. PWM signal is one where the signal turns on for a number of microseconds, then off for a number of microseconds. Servos are controlled by sending variable length pulses every 20 ms, with the length of the pulses dictating what angle the servo should move to (or with CR servos, what speed it should move at). This length of the pulse, expressed in microseconds, is called the PWM pulse width. By default, the sdk generates signals from 600 to 2400 microseconds (with SDK 0 being 600 microseconds, and SDK 1 being 2400 microseconds). However, the expansion hub can actually generate between 500 and 2500 microseconds if the range is set manually.

5V Power

All servos require a minimum of 5V to operate, and so the expansion hub generates 5V to power the servos. The 5V power is shared between pairs of ports (0-1, 2-3, 4-5). By default, this 5V power is OFF, and doesn't provide power. However, doing any servo operation, such as setting the position of a servo, will turn ON the 5V power to both shared ports. Calling disable PWM seems to turn OFF shared power, however if another servo is used or is active the 5V power will remain ON instead.

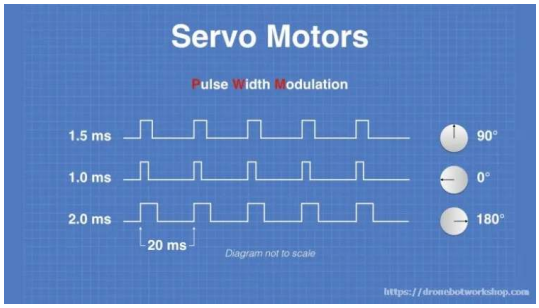


Fig. 8: An example of servo PWM waves. NOTE: Servo angles are arbitrarily selected and will not be accurate to all servos

Warning: Every pair of **two ports**, (0-1, 2-3, 4-5) share the SAME 2 amp limitation, so care should be taken that pairs of servos do not pull more than 2 amps. Putting servos in every other port is recommended when possible.

Hidden Methods

ServoImplEx

All REV hub servos are instances of `ServoImplEx`, which exposes some more methods to the user, such as setting the servo PWM range.

Note: There is no downside to using `ServoImplEx`, in order to convert a `Servo` to a `ServoImplEx` the user simply just needs to cast the `Servo` returned by the `hardwareMap` to a `ServoImplEx`.

Tips:

- You can use `setPwmRange()` to increase the servo range to a maximum of 500 to 2500 microseconds. This changes the SDK mapping, with 0 becoming the minimum microseconds and 1 becoming the maximum microseconds. **Increasing the range to 500-2500 can give more range when controlling servos that use 500-2500 PWM Range, such as goBILDA servos and REV Smart Robot Servo**
- `setPwmDisable()` and `setPwmEnable()` can be used to turn on and off the PWM signal to a servo. On some servos, this will cause the servo to de-energize and "turn off" with no holding power. Other servos may stay on and hold their position. Behavior will vary from servo manufacturer to servo manufacturer. It will also disable 5V power if the servo is the only one in its group of ports.

Tip: Both REV Smart Robot Servos and goBILDA servos will stop holding their position when `setPwmDisable` is called, regardless of if the 5v power is still on or not.

10.4.5 SDK Sensors

The SDK offers several methods of controlling and communicating with sensors.

General Explanation

I2C Explanation

I2C is a two wire serial communication bus that is designed to communicate with multiple devices. The two wires are the SCL, or clock line, and SDA, or data line. The REV hub connector also contains two more pins for 3.3v and GND. The protocol can address multiple devices using addresses, where the master device sends the address it is communicating with, then the data it wants to send. The device it is talking to then responds. This means any number of devices can be connected to one I2C port, so long as no two devices have the same I2C addresses.

I2C has multiple speed specifications, but the REV hub supports both 100khz and 400khz communication standards. The specification also requires the lines to be pulled up, so the REV hub has dual 2.49 kΩ pull up resistors, negating the need for external pull up resistors.

Tip: Devices generally have their I2C addresses hard coded, meaning they cannot be changed. Check the devices data sheet to make sure addresses will not conflict if connecting multiple I2C devices together.

Note: I2C is often called slower than other forms of sensors. It is not inherently slower, but due to the way I2C communication works in the SDK it takes multiple commands to read one I2C sensor, which means one I2C read can take 2-3x longer than one digital or analog read.

Analog Input

Reads the input voltage to the expansion hub. These ports are the ONLY sensor ports that are 5v tolerant.

Digital I/O

Digital I/O can be configured to either act as an INPUT, reading whether the port is HIGH or LOW, or an OUTPUT, sending HIGH or LOW signals. The digital input is pulled HIGH to prevent floating.

Note: Floating occurs when you attempt to read from a digital input pin but the signal is disconnected or not pulled HIGH or to ground. As a result, the pin is considered “floating” and can have inconsistent results. To prevent this, a resistor bridges between the port and 3.3V, so that instead of floating the pin instead reads HIGH when nothing is connected.

Danger: When wiring limit switches or other digital devices, DO NOT connect the limit switch to 3.3V and the digital port, like most datasheets recommend. Instead, the REV hub expects a connection between GROUND and the digital pin. **Connecting between 3.3V and the digital pin may cause instability or the hub to die.**

10.4.6 Lynx Module

The Lynx Module is an object that represents the Control Hub and Expansion Hub. One Lynx Module object is created per connected hub.

General Explanation

Obtaining Lynx Module Object

A list of connected Lynx Module objects can be obtained using `hardwareMap.getAll(LynxModule.class);`. The Control Hub or the Expansion Hub connected to the phone via USB will have `isMaster()` set to true, and the hub connected over RS-485 will have it set to false.

LED

The REV hub's LED can be set using `setConstant()` or `setPattern()`. `setConstant()` sets a constant color to the LED, and `setPattern()` allows the user to define a pattern of colors for the REV hub to follow.

Power

The total amount of current that the REV hub is pulling can be measured using `getCurrent()`. This is the current pull of the 12V input in the specified units, so it includes everything connected to the hub as well as the hub itself. In addition, the input voltage to the expansion hub can be measured using `getInputVoltage()`, which returns the battery voltage in the specified units.

Bulk Reads

Bulk reads can also be managed through the lynx module object. See *Bulk Reads* (page 319) for more information on this.

10.5 Software Glossary

Disconnect A disconnect (DC) is when, for any reason, the robot is not able to be controlled from the gamepad. This can happen for many reasons - static buildup on the robot, a loose cable, or an error in code.

Generally, most DCs are caused by improper wiring, so wire stress relief is encouraged for all teams (*USB Retention Mount*). They can also be caused by WiFi disconnects, or an ESD (electrostatic discharge) shock to the electronics.

Driver Station The Driver Station (DS) refers to both the software and device that is used by the drive team and connects to the gamepad(s). The device can either be one of the Android phones specified in Game Manual Part 1 or a REV Robotics Driver Hub.

Robot Controller The Robot Controller (RC) refers to both the software and device that is on the robot that controls it. This can either be a REV Robotics Control Hub or one of the Android phones specified in Game Manual Part 1 connected to an Expansion Hub via a Micro USB cable.

11

Awards

FIRST® Tech Challenge is more than just building robots, it contains elements of engineering documentation, outreach, and more. While the majority of Game Manual 0 talks about how to make decisions about creating your robot, the portfolio you write is actually showing how you made those decisions and how you impact the STEM community around you in a positive way.

It only benefits you to go for awards. You can't win awards without making a portfolio, and in certain states the only way to advance is awards and in others it can make it significantly easier to advance. In the past an engineering notebook was required for awards, however this is no longer true. Therefore, these are the topics that will be covered in this section of Game Manual 0.

Warning: Judging and awards are extremely subjective and some rules change a bit from state to state. What works in one region may not in other regions. So contact your judges about specifics of some things; for example some competitions allow you to turn a custom control award sheet, others only accept the one listed by *FIRST*. The writers of this part of Game Manual 0 have extensive experience with awards and have even been winners and finalists for the Inspire Award at the World Championships.

For most awards, teams have a 5 minute presentation at the beginning of the day followed by some Q&A time and possibly pit judging later on (read the section on the judging process for more detail). There are 7 awards for FTC® teams (excluding awards that cannot advance a team).

For winning every award, the Engineering Portfolio is a key reference and is quintessential, so make sure that your team focuses on the Engineering Portfolio.

Always remember that regardless of how well you feel you did, awards have a level of subjectivity of the judges, so there is no way to 100% control how your team does for awards.

Overall, awards are one of the core parts of the *FIRST* Tech Challenge competition. They recognize teams at the end of every competition (except league meets) as well as determine many of the teams that advance.

11.1 Award Categories

11.1.1 Advancing Awards

There are 7 awards for FTC® teams (excluding optional awards). For more information on the exact requirements for these awards, see Game Manual Part 1. In order of advancement priority, these are:

Inspire Award The Inspire Award is the top award for an FTC team. It is given to a team that consistently rises to the top in other awards categories. It is also (barring host team advancement), the first advancement slot on the advancement list. This means the winner is guaranteed to advance to the next level in most scenarios, unlike the winning alliance captain in some states.

In order to win the Inspire Award, your team should focus on every other award category, and perform well in the robot game. In the past, the Inspire Award has given much more weight to awards performance, but it seems as if the judges are now taking robot performance into consideration more.

Of all the awards, this one is arguably the most subjective, as it is about how good does a team does overall, including and how much they weigh outreach as compared to robot performance along with documentation.

Official Game Manual Description This judged award is given to the team that best embodies the 'challenge' of the *FIRST*® Tech Challenge program. The Team that receives this award is a strong ambassador for *FIRST* programs and a role model *FIRST* Team. This Team is a top contender for many other judged awards and is a gracious competitor. The Inspire Award winner is an inspiration to other Teams, acting with Gracious Professionalism® both on and off the playing field. This Team shares their experiences, enthusiasm and knowledge with other teams, sponsors, their community, and the Judges. Working as a unit, this Team will have shown success in performing the task of designing and building a Robot.

Think Award The Think Award is based entirely on the Engineering Notebook and Engineering Portfolio. In order to have a chance at winning, the Engineering Notebook should include as much math and physics as possible, document the entire journey of the robot through iterations, and other documentation of design and game strategy.

In addition, the portfolio should be well laid out and contain information on various non-technical aspects of the team, such as sustainability (recruitment of new team member and mentors), training, outreach (especially the impact of the outreach and what was learned from it), and team structure. While these categories aren't as strictly important as robot documentation, they are generally recommended for a strong engineering portfolio.

Official Game Manual Description This judged award is given to the Team that best reflects the journey the Team took as they experienced the engineering design process during the build season. The engineering content within the portfolio is the key reference for judges to help identify the most deserving Team. The Teams engineering content must focus on the design and build stage of the Team's Robot.

The Team must be able to share or provide additional detailed information that is helpful for the judges. This could include descriptions of the underlying science and mathematics of the Robot design and game strategies, the designs, redesigns, successes, and opportunities for improvement. A Team is not a candidate for this award if their portfolio does not include engineering content.

Connect Award The Connect Award is one of the two outreach awards. This award is for teams that work with their local STEM community and corporate community. Unfortunately, the line between the Connect and Motivate award can be vague, and judges may not differentiate which outreach falls under which award.

Look at the section about differentiating connect and motivate for more advice on this. For both outreach awards, a Team Plan is required. Refer to the section about writing a team plan for advice.

A portfolio is also required for this award, and should contain descriptions of the outreach as well as what the team gained from the outreach. In addition, the portfolio should also contain information on plans for recruitment of mentors as well as how the team plans to meaningfully develop their STEM connections.

Be prepared for judges to ask you how your outreaches were meaningful, and try to avoid doing STEM outreaches just for the sake of saying you did a STEM outreach. Successful STEM outreaches are those that are undertaken for a specific purpose and have a clear intent and goal, such as meeting with an expert on computer vision to ask for advice on a vision task for the game. If you are struggling to find STEM outreaches, local colleges and businesses are a good start.

Official Game Manual Description This judged award is given to the Team that most connects with their local science, technology, engineering, and math (STEM) community. A true *FIRST* team is more than a sum of its parts and recognizes that engaging their local STEM community plays an essential part in their success. The recipient of this award is recognized for helping the community understand *FIRST*, the *FIRST* Tech Challenge, and the Team itself. The Team that wins the Connect Award aggressively seeks and recruits engineers and explores the opportunities available in the world of engineering, science and technology. This Team has a clear Team plan and has identified steps to achieve their goals.

Innovate Award The Innovate award is what it sounds like - it's for teams with innovative robots or robot mechanisms.

The Innovate award is mainly for hardware, but some teams have been able to also present software as innovative. Some judges think it's great to present software as part of innovation, but others feel that software only fits under Control.

While it may be tempting to sell your entire robot as innovative, it is often much more effective to focus on one or two aspects of your robot instead. Judges will often ask what the most innovative part of your robot is, and this is your opportunity to focus in on the one or two mechanisms that you can sell.

The engineering portfolio should contain information on your robot's mechanisms, and your presentation should also mention the innovative parts of your robot. However, refrain from over describing the mechanisms you intend to sell as innovative, as you want to leave room for the judges to ask questions, which gives you more opportunities and time to sell your mechanisms. In addition, practice what aspects of the mechanisms you want to sell as innovative, and make sure you are able to thoroughly describe why they are innovative when asked.

Official Game Manual Description The Innovate Award celebrates a Team that thinks imaginatively and has the ingenuity, creativity, and inventiveness to make their designs come to life. This judged award is given to the Team that has the most innovative and creative Robot design solution to any specific components in the *FIRST* Tech Challenge game. Elements of this award include elegant design, robustness, and 'out of the box' thinking related to design. This award may address the design of the whole Robot or of a sub-assembly attached to the Robot. The creative component must work consistently, but a Robot does not have to work all the time during Matches to be considered for this award. The Team's engineering portfolio must include a summary of the design of the component or components and the Team's Robot to be eligible for this award. Entries must describe how the Team arrived at their solution.

Control Award The Control award is meant to recognize a team that has a good software solution to make their robot "intelligent". It's known as the "software award" and is for the team with the best or most innovative software and sensor solution for the game.

Don't be tempted to overlook the control award even though it doesn't advance at most competitions. The control award counts for the inspire award, and strong performance in control counts towards inspire ranking.

This award requires a separate submission sheet which is a condensed summary of a team's software. Check your region's rules, some require the traditional (in person)[227] or remote[228] form, but some accept custom forms. Check with your affiliate partner or judges if you are unsure. **Do not directly put code into your portfolio, control award sheet or notebook, the judges will not care**. Instead, focus on explaining key algorithms that you use, and explain the software in an easy to understand way. Remember, your control award judges may not be software engineers or programmers, so make sure you can explain everything to someone without a software background.

In addition, control award software is more then just your autonomous mode programs. Driver assistance, feedback, and automation all are vital to the control award.

Official Game Manual Description The Control Award celebrates a Team that uses sensors and software to increase the Robot's functionality in the field. This award is given to the Team that demonstrates innovative thinking to solve game challenges such as autonomous operation, improving mechanical systems with intelligent control, or using sensors to achieve better results. The control component should work consistently in the field. The Team's engineering portfolio must contain a summary of the software, sensors, and mechanical control, but would likely not include copies of the code itself.

Some examples of control award sheets are:

- 11115 Gluten Free Rover Ruckus[229]
- 11115 Gluten Free Skystone[230]
- 1002 Circuit Runners Green Skystone[231]
- 9866 VIRUS Skystone[232]
- 5143 Xcentrics Skystone[233]
- 11528 Bots of Prey Skystone[234]
- 9794 Wizards.exe Skystone[235]

Motivate Award The Motivate Award is one of the two outreach awards. It's for teams that work with their local and *FIRST* community. Unfortunately, the line between the *connect award* and motivate award can be vague, and most judges don't know how to differentiate which outreach falls under which award.

Look at the section about differentiating connect and motivate for more advice on this. For both outreach awards, a Team Plan is required. Refer to the section about writing a team plan for advice.

The key aspects to include in your portfolio and presentation for motivate is showing how all team members contribute to the success of the team, how your team is recruiting members from non-stem areas, as well as plans for fundraising, funding, sustainability, and recruitment.

Official Game Manual Description This Team embraces the culture of *FIRST* and clearly shows what it means to be a team. This judged award celebrates the Team that represents the essence of the *FIRST* Tech Challenge competition through Gracious Professionalism and general enthusiasm for the overall philosophy of *FIRST* and what it means to be a *FIRST* Tech Challenge Team. This is a Team who makes a collective effort to make *FIRST* known throughout their school and community, and sparks others to embrace the culture of *FIRST*.

[227] https://www.firstinspires.org/sites/default/files/uploads/resource_library/ftc/control-award-submission-form-traditional.pdf
[228] https://www.firstinspires.org/sites/default/files/uploads/resource_library/ftc/control-award-submission-form-remote.pdf
[229] https://docs.google.com/document/d/1dXtv628kQRlMkslx5xFYXEXGucp7-IyfMthEEfNveQ4/edit
[230] https://docs.google.com/document/d/18laHXP-aKpkPc_QzlaC5b9aeHVzLxlHNPuzaLOYh84Y/edit
[231] https://docs.google.com/document/d/1jwoP1ZpFJdSB36ybrIu1igLV8cwLweD767LLgi7pX6Y/edit
[232] https://drive.google.com/file/d/1hWp07uPvID0qbwyuOulewDEwrAl6lpMA/view
[233] https://docs.google.com/document/d/1HuuHvmBrM-qRmuz3W7KvYm7uiQcRyLXmuo-KRQFgw4E/edit
[234] https://drive.google.com/file/d/1PEFclEL5nApEOcNh-k4O4m94mGgoa35u/view?usp=sharing
[235] https://drive.google.com/file/d/1YS9scvXvqHFiqJL1beXzEUJmslHtX0IS/view?usp=sharing

Design Award The Design Award is one of the robot awards that primarily focuses on the hardware aspect of the robot. It is for robots that are both functional, aesthetic, and use good design practices, including CAD.

In order to be considered for the Design Award, it's recommended that your team uses CAD and designs the robot before it is built, with engineering portfolio sections about the development of the robot through iterations of the engineering design process. Its important to include CAD screenshots and drawings in your portfolio, and your design should be consistent with any team goals listed.

While functionality is what most teams focus on, the Design Award also takes into account aesthetics, and most judges will generally be turned off by an ugly robot for this award (no cardboard on the robot!), so make sure your robot looks presentable.

Official Game Manual Description This judged award recognizes design elements of the Robot that are both functional and aesthetic. The Design Award is presented to Teams that incorporate industrial design elements into their solution. These design elements could simplify the Robot's appearance by giving it a clean look, be decorative in nature, or otherwise express the creativity of the Team. The Robot should be durable, efficiently designed, and effectively address the game challenge.

Connect vs Motivate

Both outreach awards can be hard to differentiate, as the official descriptions are super vague, and most judges don't know the difference perfectly. Since the descriptions are very vague, it is up to you to determine what goes towards which award. Oftentimes, teams will put more outreach in *Connect* than *Motivate*, as Connect is fairly high up on the advancement list unlike Motivate, so putting more into Connect and winning Connect may advance, though make sure to do your research before trying a strategy like this.

What Falls Under Connect?

- Developing relationships with companies
- Getting external mentors (not parents)
- Fundraising from companies (other methods may fall under motivate on a case by case basis)

What Falls Under Motivate?

- Starting/Mentoring *FIRST* Teams
- Community Demos

Tips for Both

- **Present numbers, but only emphasize them if they're large with a wow factor**
  - Make sure your numbers are somewhat accurate! If you are at a large event, you can get a rough headcount from the organizers, but its generally better to know roughly how many people actually stopped and looked at your team.
- Present stories to the judges, not just overviews. Tell personal stories.
- Log all your outreach events, with who went and how many hours each person did in its own place separate from the engineering notebook. This makes it easier to compile total numbers and shows the judges every outreach activity/event in one place.
- For a bigger impact make sure you have more resources then just your team's information at an event, having details for *FIRST* Lego League teams in addition to your *FIRST* Tech Challenge information can broaden how many people you reach.

- Follow up, follow up, follow up! If a person gives you a business card or a student expresses interest, it doesn't hurt to follow up if they don't reach out as promised. People can forget or get busy, sometimes a reminder is useful!

11.1.2 Optional Awards

There are some awards that events are not required to present; these do not advance teams.

Judges Choice Award The Judges Choice Award is meant to recognize a team that doesn't fit into any of the existing award categories, but the judges still felt the team deserved to win an award for their outstanding effort or other experience. This award is very subjective, and doesn't advance teams. It is also optional to give at every competition, but in some regions given at every competition unless the judges don't find a deserving team.

Promote Award and Compass Award The Promote and Compass awards are optional awards that are usually given only at state championships and world championships. These awards do not require an engineering notebook to win, but do not advance teams. They are submitted as a video no longer than 1 minute. The Promote award is for creating a PSA for *FIRST* with a specific video prompt. This prompt changes every year, and is found in Game Manual Part 1. The Compass award is for recognizing an outstanding mentor. Submitting these awards is usually done on a case-by-case basis, where the event organizer sends teams instructions on how to submit.

Some good Promote award submissions include:

- Team 3595 in 2014[236]
- Team 8808 in 2017[237]
- Team 5795 in 2017[238]
- Team 4924 in 2016[239]

Some good Compass award submissions include:

- Team 4855 in 2017[240]
- Team 3595 in 2017[241]
- Team 9879 in 2017[242]
- Team 6510 in 2015[243]

For more information on these awards, take a look at the specific section for each award in Game Manual Part 1.

[236] https://www.youtube.com/watch?v=yYFxuJwtCu0
[237] https://www.youtube.com/watch?v=7yjGMYbtKU0
[238] https://www.youtube.com/watch?v=8gn-URpmXVA
[239] https://www.youtube.com/watch?v=lYaKEnutiR4
[240] https://www.youtube.com/watch?v=r3zK2l7K8Fo
[241] https://www.youtube.com/watch?v=EKKaMAcAfXE
[242] https://www.youtube.com/watch?v=z6M6UYMLujo
[243] https://www.youtube.com/watch?v=E76ij2H3YF4

11.2 Engineering Portfolio

The engineering portfolio is a concise but detailed overview of your team and season. **It is required for all awards and is a critical part of the judging process**.

11.2.1 Portfolio Requirements

Tip: A detailed list of the requirements is listed in section 9.3 of Game Manual Part 1

The engineering portfolio is a 15 page document consisting of standard A4 or letter sized pages and a minimum 10 point font. However, despite the small number of pages, there is a lot of information that needs to be packed into the document to meet the requirements for all awards.

Overall Tips

- Organize your portfolio clearly and carefully! Sections should be easy to follow and read, and have clear headings and explanations. Having your team number somewhere on the page is helpful for judges quickly reviewing portfolios.
- Keep formatting and branding consistent, don't vary your design a ton from one page to another. Similar layouts can help with reading the portfolio.
- Make sure your text is readable! Color choices matter a lot, try to avoid similar color text on a background.
- **Do not exceed 15 pages total, 16 with the cover page.**
- Less is more: Judges are going to lose interest in large unbroken blocks of text.
- Images, Images, Images: CAD Drawings and screenshots can be useful tools, but also have actual pictures of mechanisms and components. While renders can look impressive in covers, a lot of judges prefer actual CAD screenshots or pictures.
- **Make sure your team number is on the cover page of the portfolio.**
- Don't over explain everything. You want to leave things for judges to ask questions about.
- Remember that your portfolio isn't just for judges to read, its a good tool to have in judging as a quick reference to present to judges.

Cover Page

In addition to the 15 pages, you are allowed one cover page. This should contain your team number and any relevant logos, and should also include a picture of your robot and/or team.

Generally, the required information for an engineering portfolio can be split into one of 3 categories.

- Robot Information
- Team Information
- Outreach/Nontechnical Information

11.2.2 Robot Information

The season and robot overview is the core of your engineering portfolio. It contains an overview of your robot design, strategy, iterations, and shows off how your team has progressed so far throughout the season.

Robot

Your portfolio must contain an overview of your robot and its components. However, the portfolio should go a step beyond just describing the components of your robot. Generally, teams split up their robot into 3-4 parts, putting a different mechanism on a new page. The page should have pictures and descriptions of the mechanism, but should also have iterations and explanations of how the team arrived at that version of the mechanism or why that system was chosen, as well as why previous iterations didn't work and/or what was improved between iterations.

Tip: Get in the habit of taking photos of everything! Whether its a quick jig meant to test something or a full on prototype, you should have pictures to put in the portfolio as iterations.

As a quick reference, robot mechanism pages should contain the following:

- Explanation of why that mechanism was needed
- Explanation of what the mechanism is
- Description of iterations the mechanism went through (and a quick few words on WHY it had to go through those iterations)
- Math/equations/sensors used with that mechanism
- Pictures and/or CAD screenshots of the mechanism

In addition to mechanism-specific pages, a page or two dedicated to a high level overview of your bot and sensors can be beneficial as a quick reference, especially in judging.

Season

Your season overview is a higher level explanation of your robot design and how you have proceeded with the season. This doesn't have to be completely separated from mechanism pages, but should still provide an explanation of your overall approach to the season as well as your approach to robot design in general. This section should aim to cover the following:

- The team's approach to game and strategy analysis
- Overall approach to robot design (engineering design process, iteration, etc.)
- Approach to problem solving and decision making

Having specific examples for these sections is generally recommended.

11.2.3 Team Information

An engineering portfolio needs to cover more then just the team's season and robot, it should cover details about the team itself. This doesn't just mean a list of team members. Topics covered in this section should include:

- Basic outline of team members and groups
- Funding, sponsors, and expenses
- Sustainability plans

Funding, Sponsors, and Expenses

An important part of your team is where you get money from and how you spend your money. While perfectly precise numbers aren't required, having an awareness of what your expenses are, and where you get money from can be useful information to have in a portfolio. The portfolio should definitely include a list of sponsors, but can also include other information like team dues and a rough breakdown of expenses.

Sustainability

Sustainability is a very important part of an organization, and it is also a major part of judged awards. Sustainability doesn't just mean funding, it includes plans for recruiting new students and mentors, maintaining and growing connections with the community, and securing new sponsors. It is important to have a sustainability plan or report, an outline of how the team plans to recruit new students, mentors, and sponsors, as well as the progress the team has made on that plan. Specifically pointing out new members, sponsors, and mentors that have joined is important, so judges can see the results your team has had.

11.2.4 Outreach/Nontechnical Information

Outreach is the nontechnical aspect of your team, including connections with the general community, mentoring, and reaching out to the STEM Community. Due to how similar these things are, its tempting to combine them into one section, but there are distinctions that should be made between them.

Connections with the STEM Community

Connections with the STEM community are important to document. These should include basic details like who the person or organization is, and what they do. However, the portfolio should also explain what the team learned from the connection and why it was meaningful. Having a specific and applicable reason for a STEM connection is important, as in recent years judges have started becoming more critical of large numbers of connections that do not benefit the team.

Tip: This should tie into your sustainability plan, as connections can often become mentors or sponsors.

Outreaches with the General Community

Often teams will host events or outreaches with the intent of spreading the idea of *FIRST*® and recruiting new members to their team. Documenting this correctly is important for some awards. Generally, these sections should contain details about the event such as when it took place, what your team did, and how many people you reached.

> **Attention:** Appendix F of Game Manual Part 1 contains **very specific definitions of words like Mentored, Started, Reached, Ran, and more**. Misuse of these words can be held against you in judging, so make sure you meet the definitions of terms if you use them.

When documenting how many people an event reached, it can be tempting to ask the organizers for an official head count and use that number. However, Game Manual Part 1 specifically details that simply being at an event does not count someone as being reached by the team, they have to interact with your team in some way. Keeping a rough count of how many people you interact with can help you keep track of accurate numbers. **You may be questioned on how you know your numbers are accurate, so be ready to answer questions on how you kept track**.

Working with Other Teams

Mentoring, starting, and helping other teams is critical to the program itself as well as some awards. These interactions should be documented in the portfolio, as well as evidence that proves that you meet the definitions of Mentoring, Starting, and Assisting as outlined in Game Manual Part 1. In general, Mentoring requires regular, meaningful communication between you and a team, and Starting requires the new team agreeing that they were started by you.

Tip: An easy way to prove you met the definitions is to have screenshots of emails from mentors on the teams you helped stating that you Mentored, Started, or Assisted that team.

11.3 The Judging Process and Presentation

At the beginning of a competition, teams will have a 5 minute presentation followed by a brief Q&A period where judges can ask questions about the portfolio and presentation. Following this, judges may sometimes do follow up interviews in the pits where they ask teams more questions about specific things to help in their deliberations. This process is as important as your written documentation when it comes to the success of your team in awards.

11.3.1 Overall Tips

- **You are a team, act like one**. You should practice multiple times with each other and act as a team. Don't talk over each other, let everyone have a chance to respond. You should know each other's scripts and also how to answer a broad range of questions.
- Be professional. Always assume a judge is watching you. While you don't have to abstain from fun, be careful of how you act. When you encounter judges, be on your best behavior and be as professional as you can.
- **Be Consistent!** If you say something during the beginning Q&A, make sure you say the same thing during pit judging. If you call out one specific mechanism as innovative, don't talk about a different mechanism when questioned. Always be consistent.
- Vary who answers the questions. **Make sure everybody has an opportunity to speak.**
- Use your portfolio! You can use it to reference things or point to various sections, especially if you can't remember something in the moment.
- Don't ramble but also don't cut it short. Answer questions fully, but there is no benefit to wasting judges time. **Most of the time, judges have a set time allotment with each team so wasting time may decrease the number of questions you can answer.**
- Try not to go off topic. If a judge asks about outreach, don't talk about robot design too much. Judges are usually asking things for a specific purpose and other information may get wasted.

11.3.2 The Presentation

The presentation is a 5 minute presentation during which judges are not allowed to interrupt. At the end of 5 minutes, you will either be asked to stop or be interrupted by the judges. There is no set format for this period, so your team can figure out how they want to present. Generally, most teams will do scripted formal presentations, often with each student tackling a different part of information about the team, robot, outreaches, etc. Its recommended teams cover topics from every award topic.

Tip: While it may be tempting to do something unique, like singing or performing your presentation, be aware that this is usually difficult for little benefit. Unless you know you can pull it off very well, you probably shouldn't try it.

There is generally no penalty for using scripts, so it is highly recommended they are used. In addition, you should practice the judging period repeatedly before the competition. Practice everything from walking in and giving your portfolio to the judges to the question and answer period. **Make sure you are comfortable with your script and with answering questions**.

Presentation Tips

- Practice speaking clearly and precisely. You should practice multiple times and not only know your lines, but what people around you are going to say.
- Use the robot! Have other team members point and move things on the robot as you speak to demonstrate.
- While having a script won't be held against you, try to avoid gluing your eyes to your script and looking down. Looking at the judges occasionally will go a long way in making your presentation more professional

- **Make sure everybody gets a chance to say something or answer a question**. Having students say nothing during judging can look bad.

Q&A Period

The Q&A period is just as important as the presentation and should be practiced multiple times until the team is comfortable answering impromptu questions. A good idea is to practice the sample judges questions (Appendix B)[244]. Mentors should ask questions similar to those in the sample judges question, in addition to difficult to answer questions ("What award do you think you deserve?" is an example of a question that is incredibly difficult to answer on the spot).

Tip: Judges almost always ask something along the lines of "Is there anything else that you want us to know that was not covered?" Make sure you have a response to this question! The best strategy is usually to answer with something that wasn't asked about or covered as well, so pay attention to what topics have been covered during the Q&A.

11.3.3 Pit Judging

During the competition, secondary groups of judges may be walking around pits to ask more questions about specific components of the team, robot, or outreach. These questions are an important part of judging deliberation, and should be treated similarly to the Q&A in the beginning of the competition.

Pit judging can be difficult sometimes since judges might want to talk to your team as you are heading out to a match. While some judges may offer to come back at a different time, often it is better to have your teammates not driving talk to the judge if possible. This can ensure they don't run out of time to talk to you.

When pit judges arrive, one should try to determine roughly what they are there for. Judges will make it clear pretty quickly if they are there for control, robot-based awards, outreach, etc. If a judge is there to talk to you about the robot, don't waste their time with outreach. Similarly, if a judge is asking about your outreach don't divert to talking about the robot. Judges are there for a specific purpose, other information may essentially be discarded by them.

Important: When judges come near your pits, STAND UP and PUT THE PHONES DOWN. Uninterested students can cause a judge's opinion of your team to quickly sour. All students should try to look interested and should participate, get as many different people speaking as you can.

Pit Judging Tips

- Judges have limited amounts of time, don't waste it! Answer questions thoroughly but consciously.
- Have a portfolio or notebook ready for quick reference as needed.
- Try to vary who answers questions, getting everyone involved.

[244] https://www.firstinspires.org/sites/default/files/uploads/resource_library/ftc/judge-manual.pdf

11.4 Notebook Designs and Decisions

This section will cover both what needs to be in the notebook and the decisions you need to make about it affecting both individual sections and it as a whole.

Attention: The engineering notebook is completely OPTIONAL. Having a notebook is NOT a requirement for any award. However, judges can technically still ask for the notebook for more information, so some teams still choose to make one. **This article remains here for legacy reasons for teams that wish to make engineering notebooks, even though it is completely optional and not required for teams to still have a shot at awards**. There have been several instances of teams winning high level awards like Inspire and Think while having no engineering notebook.

11.4.1 Whole Notebook Decisions

Condensed vs Long

A trap many teams fall into is that a longer notebook is a better notebook, especially if it's actually quality writing. Unfortunately, that is not necessarily true.

At a lot of competitions, especially with the notebook being completely optional, judges don't have a lot of time to read your notebooks. They will likely take a cursory scan at a few pages and take a more detailed look at certain pages. Thus, even if you do write a great but long notebook, they might miss the highlights and rank you lower than you should have. The solution to this is to create a condensed notebook.

This is where you make a purposeful effort to get rid of redundancies and make sure everything written in the notebook serves the purpose of being read and getting things checked off of awards.

So if you know your state competitions are going to be more than a couple of days, write a long, detailed filled notebook. But if it is only a day long, write with every section having a purpose.

Handwritten vs Typed

This one is a team decision. Some teams swear judges prefer handwritten notebooks, but there is no proof for that. But if you're a small team or you would prefer to write, do that. Be sure that the handwriting is consistent and legible! Typed notebooks are a lot easier to organize, and judges will definitely be able to read them.

Serious vs Charming

A team's notebook is an extension of its personality and branding. Some teams try to come across as an engineering firm or at least very serious, while others act the opposite.

You should try and reflect that in your notebook. Neither way is better, but it's good early on to figure out your team's way of acting and writing in a manner that fits it.

Generally, humor is acceptable, but try to keep jokes down to a minimum (after all, the engineering notebook is meant to be a professional-grade document).

Note: Having a consistent team brand is important so judges remember you. A large amount of winning awards is being remembered and advocated for by the judges.

Therefore, having consistent branding helps judges know what team they are talking to and what they know about you.

11.4.2 Individual sections

Section 9.2.2 of Game Manual Part 1 lists out the type of things that are expected to see in your notebook.

They are listed here:

- Problem Definition
- Information Gathering
- Brainstorming Solutions
- Concept Design
- System Level Design
- Testing
- Design Improvement
- Production
- Promotion
- Budgeting
- Planning
- Outreach

Specific sections named in Game Manual Part 1, the Judges Manual, and Notebook Guidelines are not plentiful. They include a team plan, which is pretty much all non-robot writing—a sustainability plan, a strategic plan, and a business plan all fall into it, and doing one of these sections is required for Inspire award, and telling us what they want in the notebook (listed above). It falls to the individual notebook writers to figure out what they specifically want to call each section and what to put in each section.

Daily Logs and Other Ways of Documenting

This might be the most common way of documenting in notebooks. Even teams that are so-so about attempting to write a notebook have a couple months worth of logs because overall they are not that hard to do.

Each team approaches it differently, but a standard approach is as follows.

> Date
>
> Members of Team Present

Tasks & Pictures	More Information, if the Task Was Completed
Lorem ipsum	Dolor sit amet, consectetur adipiscing elit
Sed do eiusmod	Tempor incididunt ut labore
Et dolore magna	Aliqua ut enim ad minim veniam

There are other methods such as weekly, pure goals, or other engineering based methods such as agile (if you pursue the latter you can use those hard won mentors you have gotten). Weekly or bi-weekly is the same

as above, but the date becomes a range and you talk about what happened over that period of time. This is better for a team that has only a couple of people doing the logs because overall logs get more and more tedious.

Pure goals is simply as follows.

Date	Goal and Goal Date	Was the Goal Completed in Time?
Quis	Nostrud	Exercitation ullamco
Nisi	Ut aliquip ex ea	Commodo consequat
Duis	Aute irure dolor	In reprehenderit in voluptate

11.4.3 Building Section and Documenting the Robot

You have spent a bunch of time reading the rest of this manual to learn about the robot and how to build it. The building section is about how your robot fulfills the challenge, and what the driving factors were in building your robot.

Information about parts and materials, as well as pictures of every mechanism (plus captions) will help the judges piece together how the robot works mechanically. Explanations of the components will illustrate the thought processes behind the design.

Ample graphics (CAD screenshots/renders, pictures, etc.) will help judges understand how it works and why it is useful. **However, make sure that each graphic has a caption or explanation. Do not expect judges to understand how your robot works through pictures only.**

As with most documentation, using actual CAD screenshots or real world photos is generally better then high quality renders. While renders are interesting from a technological and aesthetic perspective, they usually take more time and effort and also don't convey your engineering process as well. CAD screenshots tend to make the document more like an actual engineering process documentation and can look much more professional, even if they aren't as aesthetically pleasing.

Additionally, use math in these explanations to target the Think award.

The second part is much more documentation and writing heavy, but in some ways it is easier. As you are building a robot, you will not get your final bot in the first attempt.

Think about telling a story of how your team progressed from brainstorm and idea conception to prototyping and final design. The judges *love* to follow a logical sequence of steps as it shows how the team thought through mistakes and improved upon successes.

Each time you iterate upon a part of your robot or move a step further in the engineering design process, document it. Important questions to ask while writing this section are below.

- What prompted this change/why was this change made?
- What was the change?
- How was the modification enacted?
- What were the results (good and bad)?
- How can this design be further improved?

This also includes your first unrealized ideas that your team talked about right when the team came together after the season was released.

11.4.4 Notebook Gallery

- Relic Recovery
  - 4042[245]
- Rover Ruckus
  - 9794[246]
  - 14270[247]
- Skystone
  - 11115[248]
  - 8813 Design Notebook[249]
  - 9656 Technical Binder[250]
  - 9656 Non-Technical Binder[251]

[245] https://drive.google.com/file/d/10TQJd4ioArq-asmswHneY9S-_okcr5vq/view
[246] https://drive.google.com/file/d/1qwtWxqy3eQ7hpiGFmD433G6NOsZ74guo/view
[247] https://qrobotics.blob.core.windows.net/2018/engineering.pdf
[248] https://drive.google.com/drive/folders/1kn8IKYeHo152oeEQ1JJz-Gwenh02U-9a
[249] https://docs.google.com/document/d/1GedNbBgpffHRdZdgFTL2-qsATb4Zrg9NDpWnJrWLZ-M/edit
[250] https://docs.google.com/document/d/1vNHSydZbP434VDVAdoqnoWFLOlDnErbDbVGmIFsGc58/edit
[251] https://docs.google.com/document/d/1qCutM4_qDffwtt5spxjaUO4TVgqlh0ORxaIAm079_a4/edit

12 Useful Resources

Here is a page dedicated to useful resources around the interwebs. Enjoy!

Note: The links with an asterisk (*) are resources that are especially pertinent or helpful. We highly suggest you check out these links.

12.1 General Resources

FTC Discord[252]* — The **unofficial** FTC® Discord server is a discussion-based community server. It is the most active FTC community, which means it generally has the most up-to-date information, and is the easiest way to get quick answers to questions. It also has a channel with direct access to vendors.

Attention: The FTC Discord's primary demographic is FTC age participants, which strongly affects the average quality of its discourse, which is often off-topic and occasionally problematic. We recommend mostly sticking to the help channels.

FTC Docs[253]* — Official *FIRST*® FTC documentation, including programming resources, team management resources and more.

FTC Q&A[254]* — The FTC game Q&A, where clarifying questions about game rules from teams are asked and answered.

Game and Season Materials[255]* — Where to find the information for the current season, including the Game Manuals.

FTC Blog[256] — The official FTC blog; important updates are typically posted here.

FTC Tutorials[257] — (Unofficial) FTC Tutorials, covering the robot, competitions, and team management.

[252] https://discord.com/invite/first-tech-challenge
[253] https://ftc-docs.firstinspires.org
[254] https://ftc-qa.firstinspires.org
[255] https://www.firstinspires.org/resource-library/ftc/game-and-season-info
[256] https://firsttechchallenge.blogspot.com/
[257] http://ftctutorials.com

FIRST Resource Library[258] — The *FIRST* resource library (filtered for FTC resources). These includes robot/field inspection checklists, robot building and programming resources, team management resources, the FTC mentor manual, and more.

Engineering Portfolio Library[259] — A library of past award-winning Engineering Portfolios. Created by Polar from FTC 23396 and managed by the community.

REV Robotics FTC Documentation[260] — Covers REV's hardware and software. The control system documentation includes an introductory FTC programming tutorial.

Spectrum's Recommended Reading[261] — A list of resources collated by FRC® 3847, Spectrum. While these resources are aimed at FRC, many are directly relevant to FTC.

12.2 Team Management

Running a FIRST Team[262]* — A guide by Karthik Kanagasbathy, former lead mentor (and current advisor) of Hall of Fame team FRC 1114, Simbotics, on running a *FIRST* team. While some details are FRC specific (namely the 6 week timeline), much of it is applicable to running a FTC team.

Team Management Resources[263] — *FIRST* FTC team management resources, covering budget, engineering notebook, etc.

12.3 Strategy

Effective FIRST Strategies[264]* — A championship conference presentation from Karthik Kanagasabapathy, former lead mentor (and current advisor) of Hall of Fame team FRC 1114, Simbotics, on effective design and competition strategies. See also the slideshow itself[265].

12.4 CAD

Autodesk Education Account[266] — Autodesk's education account sign up. An education account gives access to both Fusion 360 and Inventor.

Creo Education License[267] — Creo's education license application form.

Onshape Education Account[268] — Onshape's education account creation page.

SolidWorks Sponsorship[269] — A link to the application form for free Solidworks licenses for robotics teams.

[258] https://www.firstinspires.org/resource-library?field_content_type_value%5B%5D=first_tech_challenge
[259] https://portfolios.hivemindrobotics.net
[260] https://docs.revrobotics.com/docs/rev-duo
[261] https://spectrum3847.org/recommendedreading
[262] https://drive.google.com/file/d/0B8Oix1YVtSZgcUJYTUs0QWlnZkE/view
[263] https://www.firstinspires.org/resource-library/ftc/team-management-resources
[264] https://www.youtube.com/watch?v=5fifL47TvzE
[265] https://www.simbotics.org/_files/ugd/81d293_2417ace601d84fb5afaf62f424ad5bd3.pdf
[266] https://www.autodesk.com/education/edu-software/overview
[267] https://www.ptc.com/en/products/education/free-software/standalone-educator
[268] https://www.onshape.com/en/education/
[269] https://app.smartsheet.com/b/form/6762f6652a04487ca9786fcb06b84cb5

12.4.1 Part Libraries

10650 Hazmat Robotics Public CAD Library[270] — A multi-vendor part library that works across CAD software due to its use of STEP files.

2901 Purple Gears Onshape Parts Library[271] — A multi-vendor parts library specifically for Onshape.

REV Robotics Official CAD Library[272] — REV's official CAD library for its FTC parts.

ServoCity Official CAD Library[273] — ServoCity's entire STEP file catalog. This includes Actobotics parts.

12.4.2 Generators

Belt Generator[274] — A belt generator with 5 different belt types. This allows you to create belts with teeth for cad, and to calculate required tensioning for more complex belt runs. Made by Jeremiah from FTC 10641.

HTD Pulley Generator[275] — A HTD3 and HTD5 pulley generator, with many options. This allows you to make custom pulleys which you can 3D print. Made by Henopied from FTC 18255.

GT2 3mm Pulley Generator[276] — A GT2 3mm pitch pulley generator with many options. This allows you to make custom pulleys which you can 3D print.

12.4.3 Rendering

Blender4FTC[277] — A Blender addon and material library designed to make CAD rendering simple and easy, while being fully customizable for advanced users.

FTC Rendering in Fusion 360[278] — A guide to rendering FTC robots in Fusion 360, assuming little experience in Fusion 360.

12.5 Mechanical Design and Build

Mechatronics[279]* — A document about technologies, principles, design, and analysis of complex electro-mechanical systems. It covers topics including fasteners, manufacturing processes, fabrication paradigms, power transmission, mechanisms, design principles, and more.

8644 Brainstormers Tips and Tricks[280] — A playlist of videos comparing different implementations of various mechanisms, gears and chain, and various other FTC robot-related topics.

[270] https://workbench.grabcad.com/workbench/projects/gcpgZgLBwhIdL0FfUKJJfM75cqa9RW1ncXaL-lQ4KOl1wa#/space/gcSzacmSeI-l19BYQNPm422pSHLenRxOxVtmaD-Pzynwsq/folder/6578524
[271] https://ftconshape.com/introduction-to-the-ftc-parts-library/
[272] https://workbench.grabcad.com/workbench/projects/gcEvgrMnw6kRPx7OR6r45Gvb2t-iOdLiNG3m_ALpdGYzK_#/space/gcFd6nwp5Brrc3ks-92gagLZCV2FkceNTX3qGzaMvy2wQD/folder/2906404
[273] https://www.servocity.com/step-files
[274] https://cad.onshape.com/documents/c163c756b5096bcd95e5692a/w/44c5f14084d55dd0388345f0/e/cf391d827826f30c60340bcc
[275] https://cad.onshape.com/documents/cf7b858fb3c2f64bb9c06e22/w/c6c7b1a41995e254c2bc0115/e/392361de7956ba4aab215db8
[276] https://cad.onshape.com/documents/a0b589f74b21e8886d697efc/w/55a240a887adfa7bff84d0b2/e/fa7ce89bdce08e7313f9580b
[277] https://ryanhcode.gitbook.io/blender4ftc/
[278] https://renders360.gitbook.io/ftc-rendering-in-fusion-360/
[279] https://raw.githubusercontent.com/Thaddeus-Maximus/mechatronics_book/master/mechatronics.pdf
[280] https://www.youtube.com/playlist?list=PLoX10e-f5UgIWtNA3mlb_SSozS5w-eAlB

9794 Wizards.exe[281] — A channel containing many useful FTC videos, especially for rookies.

COREFTC[282] — A complete guide for FDM 3D printing within the scope of FTC, providing answers to a lot of the basic questions asked about 3D printing concerning topics such as: bed adhesion, tolerances, designing for 3D printing, tuning, and hardware choices.

Designing Competitive FTC Robots (paid)[283] — This book describes the overall approach to designing a robot including strategy, brainstorming, and prototyping.

Fastener Guide[284] — A guide explaining the different types of fasteners and where they are useful. It also includes printouts with drawings of various to scale fasteners to help identify them.

FIRST Robot Building Resources[285] — A collection of official *FIRST* design and build resources, including TETRIX/REV build guides.

goBILDA with TETRIX[286] — Documentation covering how to use goBILDA together with TETRIX.

NASA RAP Design Guide[287] — A guide for competitive robotics covering topics such as manufacturing, design styles, power transmission, mechanism design, and more.

The Unofficial FRC Mechanism Encyclopedia[288] — A page containing video examples of a bunch of FRC and FTC mechanisms categorized by type.

12.5.1 Power Transmission

SDP-SI Timing Belt and Pulley Handbook[289]* — A detailed handbook about belts and pulleys. Some of the things mentioned also apply to chain.

SDP-SI Timing Belt Drive Design Guide[290]* — A one-page guide to designing belt and pulley drive systems.

Gears Educational Systems Guide to Chain Drive Systems[291] — A guide to roller chain, featuring some useful equations.

Gear Efficiency Comparisons[292] — A comparison between different types of gears (spur, bevel, worm, etc).

How Gears Work[293] — An interactive visual guide to how gears work.

KHK Introduction to Gears[294] — A handbook covering fundamentals about the mechanicis of gears.

[281] https://www.youtube.com/channel/UC988iYaWDOF7Fpv6HqN-wjQ/featured
[282] https://www.coreftc.org/
[283] https://www.amazon.com/dp/B09DN3999Y
[284] https://www.boltdepot.com/fastener-information/printable-tools/printable-fastener-tools.pdf
[285] https://www.firstinspires.org/resource-library/ftc/robot-building-resources
[286] https://gobildatetrix.blogspot.com
[287] https://robotics.nasa.gov/nasa-rap-robotics-design-guide/
[288] https://www.projectb.net.au/resources/robot-mechanisms/
[289] https://www.sdp-si.com/PDFS/Technical-Section-Timing.pdf
[290] https://www.sdp-si.com/Belt-Drive/Designing-a-miniature-belt-drive.pdf
[291] https://web.archive.org/web/20191020193018/http://gearseds.com/documentation/deb%20holmes/2.5_Chain_drive_systems.pdf
[292] https://www.meadinfo.org/2008/11/gear-efficiency-spur-helical-bevel-worm.html
[293] https://ciechanow.ski/gears/
[294] https://www.khkgears.co.jp/kr/gear_technology/pdf/gear_guide_060817.pdf

12.5.2 Calculators

ILITE Drivetrain Simulator (v2020)[295] — A drivetrain calculator that puts an emphasis on time to target. Input a target motor, number of motors, gearing, and a wide range of elements about the drive train and electrical system. The output shows estimated peak speed, estimated sprint time, minimum system voltage, and maximum voltage while the drive train is at full speed.

JuliaDesignCalc[296] — A spreadsheet design calculator. However, it does not include FTC motor data, so you will need to add that manually using the data from the *Peak Power and Motor Curves* (page 232) section.

Pulley Center-to-Center Calculator[297] — A fully featured belt pulley center-to-center distance calculator.

ReCalc[298] — A collaboration focused mechanical design calculator, currently in alpha, which provides sharable links.

Thad's EveryCalc[299] — A mechanical design calculator which covers a wide variety of mechanisms, while also providing some utilities like belt sizing, a trajectory calculator, and more.

12.6 Programming

Learn Java for FTC[300]* — An introduction to FTC programming, assuming no preexisting Java knowledge.

FTC Robot Controller Repository[301]* — The home of the FTC SDK. Also check out the associated wiki[302] and JavaDocs[303].

REV's Introduction to Programming[304]* — REV's introductory programming documentation, covering both Blocks and Java programming. Linked here is also the rest of REV's documentation for the control system.

Controls Engineering in the FIRST Robotics Competition[305] — A book that introduces students to the broader field of control theory.

CTRL ALT FTC[306] — A guide to control theory created by FTC #19376 Thermal Equilibrium.

FIRST Programming Resources[307] — A collection of official *FIRST* programming resources, including introductions to each programming tool.

FRC 4613 Software Workshops[308] — FRC 4613's workshops that are used to teach their new programmers Java and FTC programming. It goes from the very basics of data up to more complex concepts such as Polymorphism and Functional Interfaces.

Intro to Control Theory[309] — A series of blog posts about control theory.

[295] https://www.chiefdelphi.com/t/ilite-drivetrain-simulator-v2020/369188
[296] https://www.chiefdelphi.com/uploads/short-url/uJyrWsJqE8OVqbvMLMnSgJ8QUdP.xlsx
[297] https://sdp-si.com/eStore/CenterDistanceDesigner
[298] https://reca.lc/
[299] http://everycalc.thadhughes.xyz/
[300] https://raw.githubusercontent.com/alan412/LearnJavaForFTC/master/LearnJavaForFTC.pdf
[301] https://github.com/FIRST-Tech-Challenge/FtcRobotController
[302] https://github.com/FIRST-Tech-Challenge/FtcRobotController/wiki/
[303] https://javadoc.io/doc/org.firstinspires.ftc
[304] https://docs.revrobotics.com/duo-control/programming/hello-robot-introduction-to-programming
[305] https://file.tavsys.net/control/controls-engineering-in-frc.pdf
[306] https://www.ctrlaltftc.com/
[307] https://www.firstinspires.org/resource-library/ftc/technology-information-and-resources
[308] https://github.com/Team4613-BarkerRedbacks/SoftwareWorkshops
[309] https://blog.wesleyac.com/posts/intro-to-control-part-zero-whats-this

12.6.1 Libraries

Easy Open CV[310] — A straightforward way to use openCV on an FTC robot. With this library, you can go from a stock SDK to running a sample openCV OpMode, with either an internal or external camera, in just a few minutes!

FTC Dashboard[311] — FTC Dashboard is a websocket-based React dashboard designed for FTC. It is very useful for debugging, including features such as displaying and graphing telemetry live as well as tuning configuration variables in real-time while opmodes are running.

FTCLib[312] — A fairly comprehensive FTC library, notably providing a command-based programming paradigm and vision pipelines.

Road Runner[313] — Road Runner is a motion planning library. Designed primarily for autonomous robotic movement, it allows for complex path following and generation while maintaining control of velocity and acceleration. This enables bots to have more accurate and advanced path following capabilities. Also see Learn Road Runner[314], a guide to setting up Road Runner.

12.7 Electronics

Robot Wiring Guide[315]* — *FIRST* guide to wiring FTC robots, including ESD mitigation options.

An Analysis of ESD Mitigation for the FIRST Tech Challenge[316] — An analysis of different methods to mitigate electrostatic discharge (ESD) issues, which can cause robots to disconnect. The paper has great recommendations on what to do to help mitigate these issues in its conclusion.

12.8 Team/Event Results

FTC Events[317] — Official *FIRST* team and event result database for FTC.

FTC Scout[318] — A community FTC Events frontend with more features and a more slick UI.

The Orange Alliance[319] — A community-run team and event result database, primarily useful for its more comprehensive results before Ultimate Goal (2020-2021).

[310] https://github.com/OpenFTC/EasyOpenCV
[311] https://github.com/acmerobotics/ftc-dashboard
[312] https://github.com/FTCLib/FTCLib
[313] https://github.com/acmerobotics/road-runner
[314] https://www.learnroadrunner.com/
[315] https://www.firstinspires.org/sites/default/files/uploads/resource_library/ftc/robot-wiring-guide.pdf
[316] https://www.firstinspires.org/sites/default/files/uploads/resource_library/ftc/analysis-esd-mitigation-echin.pdf
[317] https://ftc-events.firstinspires.org/
[318] https://ftcscout.org
[319] https://theorangealliance.org/

13 Appendix

13.1 Gallery Of Robot Designs

Collection of robot designs from past seasons. Feel free to borrow some design solutions for your robot!

Note: Copying another team's design usually doesn't work if you don't understand why their design works and why they built it that way. Just like copying your friend's homework doesn't mean you learned the material.

13.1.1 2022-2023 Power Play

11329 I.C.E. Robotics

- CAD[320]
- Robot Reveal Video[321]
- Behind the Bot[322]
- Code[323]
- Portfolio[324]

[320] https://grabcad.com/library/2023-ftc-power-play-cad-team-11329-i-c-e-robotics-glacier-1
[321] https://www.youtube.com/watch?v=deOm05iy3Ak
[322] https://www.youtube.com/watch?v=Bhwif_vSumw
[323] https://github.com/FTC11329/11329-2023-repo
[324] https://drive.google.com/file/d/1Ji07uGThsF0prkGztpE3OcW9LZN0pYVJ/view

19043 CyLiis

- CAD[325]
- Robot Reveal Video[326]
- Behind the Bot[327]

19044 Peppers

- CAD[328]
- Robot Reveal Video[329]
- Behind the Bot[330]

13.1.2 2021-2022 Freight Frenzy

12518 Almond Robotics

- CAD[331]
- Portfolio[332]

13.1.3 2020-2021 Ultimate Goal

13648 Jankbot

- CAD[333]
- Robot Pictures[334]

[325] https://cad.onshape.com/documents/ecc71c6b26b43f044d4b2589/w/a43082b1875fd38bd5f9bcd2/e/83bd8eba2133596a2717cfac?renderMode=0&uiState=64b5b6fb9b2a8d56d422b561
[326] https://www.youtube.com/watch?v=szGZ6emLUhE
[327] https://www.youtube.com/watch?v=PDxPbxG_3LY
[328] https://a360.co/46LcT7E
[329] https://www.youtube.com/watch?v=oouTEb8M56g&pp=ygUUcGVwcGVycyByb2JvdCByZXZlYWw%3D
[330] https://www.youtube.com/watch?v=_PD54AEV-DM&pp=ygUUcGVwcGVycyByb2JvdCByZXZlYWw%3D
[331] https://cad.onshape.com/documents/ebe870041c6727c32e6a81e1/w/9e5b6fc4b42139b9df352731/e/ce77d17c2170332caa7262cc
[332] https://drive.google.com/file/d/1Fe6p13VGeGRCbjY_8PlscKRp1Vw9C9lU/view?usp=sharing
[333] https://cad.onshape.com/documents/ec03c5a1726117b5dd0ef434/v/26fc62a203f44bf75b45a13d/e/863dc892f987c32991536897
[334] https://photos.google.com/share/AF1QipORERv83O2EB2hgFqmfkuEavisH8N4cqhkPNFVCDFGqgaVcj6ED77WXLYPi9yIQIQ?key=VUhvZmxIejYwRUU4b3lPaXZTcmZ0emFsa21yNy13

16537 LOGICoyote

- Engineering Portfolio[335]
- Robot Reveal Video[336]

17077 Adna Robotics

- Engineering Portfolio[337]
- Robot Reveal Video[338]

18253 Beach Bots

- CAD[339]
- Robot Explanation Video[340]
- Robot Description Flyer[341]

18275 SubZero

- CAD[342]
- Remote Match Video[343]

13.1.4 2019-20 Skystone

11115 Gluten Free

- CAD[344]
- Robot Reveal Video[345]
- Former World Record Match Video[346]

[335] https://drive.google.com/file/d/1pJoChbVlvHk76GqQmj4wkOcPWZAHNehL/edit
[336] https://www.youtube.com/watch?v=eSGSAS1RTHQ
[337] https://docs.google.com/document/d/1Gd3HlolZlD26xz__ngC1cJsoUYtvUFR_MTUYkEW_L1g/edit
[338] https://www.youtube.com/watch?v=mSsAVnTCXg0
[339] https://cad.onshape.com/documents/c4258a3b5a1dbcdad41e21f5/w/4f7810069e9b16a173d2bf0a/e/f837c09187d1cca462aaeca2
[340] https://www.youtube.com/watch?v=fZFT6Cdp58g
[341] https://www.canva.com/design/DAEkqnr3g_8/vUqf5zKo3njwY0KRxsmhXg/view
[342] https://gmail455333.autodesk360.com/g/shares/SH56a43QTfd62c1cd968310eba6a86848032
[343] https://www.youtube.com/watch?v=4Y9WguSl4DE
[344] https://myhub.autodesk360.com/ue2b675b9/g/shares/SH919a0QTf3c32634dcf988c313f186aa49c?viewState=NoIgbgDAdAjCA0IDeAdEAXAngBwKZoC40ARXAZwEsBzAOzXjQEMyzd1C0AmAM0YCMAbAA4IAdgC0uEQBNxAFm6cY4vhFydx3IZ2kBmIQO4Lc%2BEAF8QAXSA
[345] https://www.youtube.com/watch?v=i2g_b54MEFI
[346] https://www.youtube.com/watch?v=hL4nYgLUCeg

11503 Viperbots Hyperfang

- Technical Binder[347]
- CAD[348]

14270 Quantum Robotics

- CAD[349]
- Robot Reveal Video[350]

13.1.5 2018-19 Rover Ruckus

8417 'Lectric Legends

- CAD[351]
- Robot Reveal Video[352]
- Behind the Bot Interview[353]

9048 Philobots

- CAD[354]

9872 (In)Formal Logic

- CAD[355]
- Robot Reveal Video[356]
- Behind the Bot Interview[357]

[347] https://docs.google.com/presentation/d/1MtXrXihTsF2XNWUVU9fH8fmdqNRnnIpUPR5ZxJDZaH0/edit?usp=sharing
[348] https://myhub.autodesk360.com/ue2d6cfee/g/shares/SH919a0QTf3c32634dcfc62291ba1fe920f7
[349] https://myhub.autodesk360.com/ue2b699be/g/shares/SH56a43QTfd62c1cd968c54efb8b6d65921b
[350] https://www.youtube.com/watch?v=3d8-TN8YVNU
[351] https://myhub.autodesk360.com/ue2d6cfee/g/shares/SH919a0QTf3c32634dcf9939325e4a438df9
[352] https://drive.google.com/file/d/1O44wlNqllfe16ktQYHCRPb-YUxIXzPUp/view
[353] https://www.youtube.com/watch?v=IW70TEpFtxM
[354] https://myhub.autodesk360.com/ue2d6cfee/g/shares/SH919a0QTf3c32634dcf1857225708295441
[355] https://myhub.autodesk360.com/ue2814ea3/g/shares/SH56a43QTfd62c1cd968250c04221a0d6400
[356] https://www.youtube.com/watch?v=pMI2PXhnlS0
[357] https://www.youtube.com/watch?v=6PjfbOV496c

11115 Gluten Free

- CAD[358]
- Practice Match Video[359]
- World Record Match Video[360]
- Behind the Bot Interview[361]

14270 Quantum Robotics

- CAD[362]
- Robot Description Flyer[363]
- Practice Match Video[364]
- Robot Reveal Video[365]

13.1.6 2017-18 Relic Recovery

9794 Wizards.exe

- Robot Reveal Video[366]
- 3D model of the claw[367]
- Misc CAD models[368]

13.1.7 2016-17 Velocity Vortex

3415 Livingston Lancers

- Robot Reveal Video[369]
- Robot Render[370]
- Engineering drawing[371]

[358] https://myhub.autodesk360.com/ue2d6cfee/g/shares/SH919a0QTf3c32634dcf876fb9be002654e2
[359] https://www.youtube.com/watch?v=NQvhvYJXVMA
[360] https://www.youtube.com/watch?v=Nm3ff5JqvzM
[361] https://www.youtube.com/watch?v=zun--sNljks
[362] https://myhub.autodesk360.com/ue2b699be/g/shares/SH56a43QTfd62c1cd968e7fc6e5b3808809c
[363] https://qrobotics.blob.core.windows.net/2018/mti.pdf
[364] https://www.youtube.com/watch?v=v4Jpfe0eJUc
[365] https://www.youtube.com/watch?v=v4XP_VJ7nZU
[366] https://www.youtube.com/watch?v=wBmb-4cu4Vs
[367] https://www.thingiverse.com/thing:2785600
[368] https://drive.google.com/drive/folders/1Ng-DqcyMdsfpHy7Mc6W0cfxUMahaA2Sn
[369] https://www.youtube.com/watch?v=8jvF94d46cs
[370] https://drive.google.com/file/d/1oCy7M8DCr8fLGUcjR6L4Akm1JUgkqhYt/view?usp=drive_open
[371] https://drive.google.com/file/d/1YQMyEWS5sPdL1YOPntXIR0FdsY30-G6H/view?usp=drive_open

4137 Islandbots

- Technical Binder[372]
- CAD[373]
- Robot Reveal Video[374]
- Match Video[375]

9794 Wizards.exe

- Robot Reveal Video[376]

13.2 Vendor List

- Actuonix[377]
  - Actuonix sells linear actuators and linear motion components. Expensive, but robust. Teams can apply for a *FIRST*® sponsorship.
- AndyMark[378]
  - AndyMark sells the official game field and game sets, as well as individual game parts and the SoftTiles foam tiles.
  - AndyMark also sells NeveRest and NeveRest Sport motors, TileRunner, compliant, stealth, and mecanum wheels, as well as many other items.
  - AndyMark sells the Robits build system, which uses non-metric components similar to the Tetrix build system.
- **Axon Robotics**[379]
  - Vendor for highly customizable servos, with software functionality for range increases to switch between positional mode and continuous rotation mode.
- CopperState[380]
  - Cheap vendor for fasteners/other hardware.
- eplastics[381]
  - Vendor for raw plastics, such as polycarbonate, acetal, HDPE, etc.
- estreetplastics[382]
  - Vendor for raw plastics, including smoked polycarbonate.

[372] https://docs.google.com/document/d/1RMsGYUu_mo943I42diFhakRUgHF-Bi4TcWEwkxHUE9g/edit?usp=sharing
[373] https://myhub.autodesk360.com/ue2801558/g/shares/SH7f1edQT22b515c761ec425b0f17a8d8573
[374] https://www.youtube.com/watch?v=acWoCPkWOZs
[375] https://www.youtube.com/watch?v=myq3DyHqM0w
[376] https://www.youtube.com/watch?v=pJs-R-j0zXg
[377] https://www.actuonix.com/
[378] https://www.andymark.com
[379] https://axon-robotics.com/
[380] https://www.copperstate.com/
[381] https://www.eplastics.com/
[382] https://www.estreetplastics.com/Default.asp

- Fastenal[383]
  - Mid-range vendor for fasteners/other hardware, including tools.
- goBILDA[384]
  - goBILDA sells its own build system, complete with Yellow Jacket motors, channel, motion components, and battery. Note that the cheapest batteries are found here, the MATRIX 12V Batteries are $39.99, with the team discount they are $29.99, nearly half the price of the $50 batteries sold elsewhere.
  - Teams can get a 25% Team Discount from goBILDA.
- McMaster-Carr[385]
  - McMaster-Carr sells hardware and raw materials in bulk quantities. They stock nearly every type of bolt, screw, and nut possible, as well as washers, bearings, springs, etc. Purchase from them for bulk quantities of hardware, as well as the times you need a very obscure part.
  - Don't be turned off by the hidden shipping. Generally, McMaster-Carr's shipping is around the same price as other vendors, and shipping is usually next-day.
- MiSUMI[386]
  - MiSUMI is a Japanese company specializing in industrial and manufacturing components. They sell bulk 15mm anodized *extrusion* similar to the REV Robotics *extrusion*. The 15mm *extrusion* can be cut to length as well.
  - MiSUMI also sells aluminum drawer slides that are popular for linear extensions. They are available in different lengths, but the most common is 400mm.
- OnlineMetals[387]
  - A common vendor for purchasing various raw metals.
- Pitsco[388]
  - Pitsco sells the Tetrix kit with channels, TorqueNado motors, and their own motion system.
- REV Robotics[389]
  - REV Robotics sells the REV build system, which is an *extrusion*-based ecosystem complete with motors (HD Hex, HD Planetary, Core HEX), *extrusion*, *servos* (Smart Robot Servo), brackets, and battery.
  - REV also sells the control system for FTC® (Expansion Hub and Control Hub).
    - REV offers various sensors (Magnetic Limit Switch, Color Sensor, Touch Sensor, Distance Sensor, Potentiometer, etc.)
    - Additionally, electronic components such as the Servo Power Module, SPARKmini, or Blinkin may be purchased.
- Servocity[390]
  - Servocity sells a wide range of *servos*, from Hitec to Futaba, at all price points.
  - Teams can get a 25% team discount from Servocity

[383] https://www.fastenal.com/
[384] https://www.gobilda.com/
[385] https://www.mcmaster.com
[386] https://us.misumi-ec.com/
[387] https://www.onlinemetals.com/
[388] https://www.pitsco.com
[389] https://www.revrobotics.com
[390] https://www.servocity.com

- Studica[391]
  - Studica sells the Studica build system, which has hole patterns compatible with most systems.
  - Additionally Studica owns and sells the navX product, a higher-end IMU than what is provided in the Control Hub.
- West Coast Products[392]
  - Also known as WCP, West Coast Products sells products exclusively aimed toward FRC® use. However many of their parts can be used in FTC, especially in custom robots.

13.3 Vendor Guide

13.3.1 Vendor Identification

When you read some Bill of Materials (BOMs) or part numbers that are references in this guide, it might be confusing to find what vendor the part you need comes from. The obvious thing to do is Google the part number, and generally it can identify what you want and where it comes from. But if that fails, here are some quick tips to identify the vendor selling what you need.

Prefix Identification

- AM- signifies AndyMark (i.e. AM-0447)
- REV- signifies REV Robotics (i.e. REV-31-1155)
- WCP- signifies West Coast Products (i.e. WCP-0117)

SKU Identification

- Actuonix: 3-4 digit SKU, alphanumeric, in the form of XX00 or X00 (i.e. PQ12)
- goBILDA: 12 digit SKU, numerical, in the form of 0000-0000-0000 (i.e. 3213-3606-0001)
- Pitsco/Tetrix: 6 digit SKU, W + 5 numbers, in the form of W00000 (i.e. W44260)
- Servocity/Actobotics: 6 digit SKU, numerical, in the form of 000000 (i.e. 615190)

13.3.2 Where can I get ...?

- Motors and Gearboxes
  - Spur Gear Motors
    - *AndyMark NeveRest* Classic
    - *REV HD Hex* & *Core Hex*
    - goBILDA 5201 Spur Gear
    - TETRIX TorqueNado

[391] https://www.studica.com/ftc-2
[392] https://wcproducts.com/

    - Planetary Gearbox Motors (recommended)
        - *AndyMark NeveRest* Orbital
        - *REV HD Hex* Planetary & *UltraPlanetary*
        - *goBILDA 5202/5203 Yellow Jacket Planetary*
        - (*italicized* = customizable gearbox ratio)
- Servos
    - HiTec (from ServoCity); 24 tooth spline
    - REV Smart Robot Servo; 25 tooth spline
    - Axon Mini & Max; 25 tooth spline
    - goBILDA Dual Mode Servo; 25 tooth spline
    - Futaba (Futaba servos are widely used in RC and hobby aircraft); 25 tooth spline
    - Savox heavy-duty servos; 25 tooth spline
    - Actuonix (linear actuators)
- Sensors
    - AndyMark
    - REV
- Hardware & Raw Materials
    - McMaster-Carr
    - OnlineMetals
    - Amazon
    - Local hardware store
- Aluminum Components
    - Actobotics *channel*, mini *channel*, X-rail *extrusion*
    - REV extrusion, C *channel*, U *channel*
    - goBILDA *channel*, goRAIL *extrusion*
    - TETRIX *channel*
    - MiSUMI *extrusion*
- Gears/Sprockets/Pulleys
    - REV *gears*, *sprockets*, pulleys
    - Actobotics *gears*, *sprockets*, pulleys
    - goBILDA *gears*, *sprockets*, pulleys
    - TETRIX *gears* and pulleys
    - AndyMark *gears* and pulleys
- Wheels
    - AndyMark traction, *compliant*, *mecanum wheel*
    - REV grip, traction, *omni*, *mecanum wheel*

  - ServoCity traction, *omni wheel*
  - goBILDA traction, *omni*, *mecanum wheel*
  - TETRIX traction, *omni*, *mecanum wheel*
  - Nexus *mecanum wheel*
- Linear Slide Kits
  - REV 15 mm *extrusion* slide kit
  - Actobotics linear slide kit
  - goBILDA linear slide kit
- Drawer Slides
  - MiSUMI SAR2 or SAR3 aluminum slide
  - Long Robotics Slides
  - Hafele cabinet slide
  - Steel-rolled drawer slide
  - Igus

13.4 License

[393] https://github.com/gamemanual0/gm0/blob/main/LICENSE

14 Contributor's Guide

14.1 Introduction

Hello, and thank you for taking the time to consider contributing to Game Manual 0. Your knowledge and contributions will help many other teams, and improve Game Manual 0 as a universal resource. This document is designed to give a basic overview of contributing to Game Manual 0.

14.2 Who can contribute?

Game Manual 0 welcomes contributions from basically anybody. The majority of this guide was written by accomplished students of FTC® from a wide variety of teams, alongside mentors and alums. Our editorial control is based purely on content, as long as the content is good, it'll be added to Game Manual 0.

14.3 Goals of Game Manual 0

Before writing, please take the time to familiarize yourself with the goals of Game Manual 0. The primary goals are:

1. To inform and guide newer teams, participants, and coaches of the *FIRST*® Tech Challenge and help them achieve their competitive goals.
2. To act as a living reference for mechanisms, techniques, and community knowledge pertinent to *FIRST* Tech Challenge.

These goals may seem relatively self-explanatory, but they have several important takeaways.

- **These goals do not necessarily build upon each other.** For example, let's look at the *Drivetrains* (page 90) section. If we want to simply guide more rookie teams on a path to become successful, we would discuss the very popular drivetrains that teams may encounter (2WD, 4WD, 6WD, mecanum, X-Drive, and H-Drive) so that teams can quickly get up to speed and choose a competitive drivetrain. However, if the goal is to be a comprehensive reference, we would include every single possible drivetrain ever created for FTC, even if they aren't very competitive or realistic for newer teams to build (swerve, ball, octocanum, etc).

  - We are seeking to accomplish both goals with a new structure: there will be a "For Rookies" page, that will link to a list of curated pages designed for rookies, while pages outside this list can act as more comprehensive references.
  - Keep the goal of the page and whether it is designed to be kept on the "For Rookies" page in mind while writing, and adjust the content accordingly.

- **Game Manual 0 is a guide from primarily a competitive standpoint.** In general our recommendations have the end goal of increasing the points scored by a team while working inside a team's limitations, and building good technical habits that will be useful in a future career.
- **Game Manual 0 seeks to inform, not tell.** Game Manual 0 does not wish to encourage one "meta" design or series of techniques, instead, the guide seeks to present teams with a variety of options, with a pros and cons list of each, and let teams make their own decisions for their own circumstances. If recommendations are given, they must be based on objective, repeatable reasoning. For more information on this, read the *Style Guide* (page 386).
- **Game Manual 0 is generally a high level concept guide.** This means that information is given in a way that is applicable to many situations, rather than a series of steps. However, when possible we link to external documentation to plug any knowledge gaps. For example: in the electronics sections, we do not discuss how to exactly wire the robot using the REV control system, instead we link to the REV documentation for exact wiring and discuss general wiring best practices.

14.4 Getting Started Contributing

1. Read through this document
2. Read through the *Style Guide* (page 386)
3. Review the Project List[394] and GitHub issues[395]
4. Check out the *Methods of Contributing* (page 388)

14.4.1 Style Guide

Don't deal in absolutes.

- Only a Sith deals in absolutes
- Use pros/cons lists to compare options
- Explain **WHY** something is good or bad
  - For example, we all know Tetrix V2 Motors are bad. But don't just say they're bad, say: "Tetrix V2 motors are underpowered and often burn in a few seconds under stall, which is why they are not generally recommended. They also do not have built in encoders. We would highly recommend purchasing RS-550/RS-555 class motors being sold by goBILDA, REV, and AndyMark instead."
  - Similarly, we know the goBILDA motors are generally good. But *explain* why they are good, e.g. "goBILDA Yellow Jacket Planetary motors are generally recommended if you need a 6mm D/8mm REX output shaft motor, and especially if you use goBILDA as your primary build system. These motors have a robust gearbox that generally can handle a decent amount of shock load (but they still must be supported in high torque situations) and conveniently face mount into goBILDA parts. Additionally, goBILDA motors do not require the use of a level shifter to use its encoders

[394] https://docs.google.com/document/d/1_o7SoUFs6OznR0U07-Mxr5CLdYmKN-NuU7TiQX8nL6A/edit
[395] https://github.com/gamemanual0/gm0/issues

with the REV control system for flexibility. Please note that you will need to adapt the bullet connectors to the REV control system power cables, and the goBILDA motor power cables are not removable, unlike the REV Robotics motors."

- Still emphasize that teams are free to explore and innovate, but help set realistic expectations (see the following point)

Game Manual 0 is a guide **from a competitive standpoint**.

- Try to leave out stuff that doesn't work well and is unpopular; if it is popular it is worth explaining the disadvantages (see H-drive vs ball drive; explaining H-drive, as a relatively popular and simple drivetrain makes sense, but ball drive, a drivetrain that barely exists and is ridiculously complicated makes no sense to bring up)
- Try to leave opinions out as much as possible. Do not speak authoritatively on stuff you do not have first-hand experience with whenever possible
- **Keep in mind that FTC design trends are temporary and transient**
  - Just because something is popular one season doesn't mean it's the end all be all. There was a time when 6-wheel drives and coaxial swerve drives were all the rage, but that doesn't mean that we should recommend them in Game Manual 0. Try your best to think about why something is popular, and what benefits in design, function, and execution they actually bring to the table.

Consider that not all teams have the resources to purchase the most optimal parts for their design.

- This means that we should give recommendations on how to mitigate issues with poor quality parts.
  - For example: TETRIX structure has a bad reputation for bending at times, at least compared to other build systems compared to REV, goBILDA and Actobotics. Therefore, we recommend adding extra support when building in TETRIX compared to other systems (this means including extra crossbars and adding standoffs inside channel to increase rigidity).

Include pictures/videos whenever possible.

- Adding examples of mechanisms really helps with knowledge transfer.
  - Credit the teams: Caption Format is: [Team Number] [Team Name], (Relevant Accomplishment), [Season], (description)
    - []s mean all the time, ()s means when relevant
    - If you have multiple pictures by the same part from the same team only credit them on the last one to avoid repetition.
    - Examples
      - 11115 Gluten Free, Finalist Alliance Captain (Detroit), Relic Recovery, springloaded
      - 8417 'Lectric Legends, Rover Ruckus, TPU intake flaps
      - 7236 Recharged Green, Rover Ruckus, Misumi SAR3

Adhere to brand standards when possible.

- Abide by *FIRST*® Trademark guidelines, available here[396]
  - The first instance of *FIRST* and FTC on a page should include ® (i.e, *FIRST*®)
  - Always capitalize and italize the name *FIRST*
  - Do NOT use a possessive on the *FIRST* name (e.g, *FIRST*'s)
- It's gm0 not GM0 damn it; look at the logo.

[396] https://www.firstinspires.org/sites/default/files/uploads/resource_library/UseofUSFIRSTandLEGOGroupTrademarksandCopyrightedMaterials.pdf

- This also applies to team names: spell them how they are officially spelled
  - Check The Orange Alliance[397] or FTC Events[398] if you do not know how to spell a team's name.

You can use "you" when writing, when it makes writing less awkward. However, try avoiding excessively using it.

14.4.2 Methods of Contributing

GitHub Pull Request (Strongly Preferred)

Pull Requests should be made to the gm0[399] repository on GitHub. They will be reviewed and comments may be made. Draft pull requests are strongly recommended when you are working on something, as it allows for feedback during the development process. It also means that the CI can run.

Document Submission

This method is designed to help contributors who do not have an understanding of git and reStructuredText to start contributing. This method only works for adding **new pages**; you **cannot** edit current pages using this method.

1. Write your intended content in a Google Doc
2. Change the Google Doc to "Anyone on the internet with this link can comment"
3. Submit a link to the Google Doc in the following form[400]

14.4.3 Contributors

Game Manual 0 would not have been possible without the many contributors who have sacrificed their precious freetime (or merely found another excuse to procrastinate their work). I am especially grateful to Ben and Abi for helping port over Game Manual 0 to Copperforge, and for all those who have continued to add and revise Game Manual 0 during the busy school year. Game Manual 0 had been my dream for quite some time, and as an alum tremendously grateful for all that *FIRST*® has taught me, I wanted to create Game Manual 0 to preserve and spread the knowledge base that might have been lost with the FTC® class of 2019 and beyond. For this reason, I wanted to recognize all the contributors who have helped to make Game Manual 0 a great resource for *FIRST* Tech Challenge teams. Feel free to contact any of the names listed below via the email (gamemanual0@gmail.com) or through the other contact emails - I'm sure that they would be more than willing to help you out!

Kleptomaniac

[397] https://theorangealliance.org/
[398] https://ftc-events.firstinspires.org/
[399] https://github.com/gamemanual0/gm0
[400] https://docs.google.com/forms/d/e/1FAIpQLSdo2p-2dDjjUF180ggXPjUJGjucxpvLO9_fU1oEUpBBivDP4A/viewform

Managing Editors

- Tom - FTC Alum - 3736 Serious Business- seminole3736@gmail.com
- Abigail - FTC 7026 JDroids - gm0@dogbuilt.net
- Frank - FTC 8581 Aedificatores - fgportman00@gmail.com
- Justin - FTC 9656 Omega - ftc9656omega@gmail.com
- Nathanael/Kleptomaniac - FTC Alum - 13075 Coram Deo Robotics - nathanchu@utexas.edu
- Davy - FTC 16461 Infinite Turtles - davy@mcr.club

Other Contributors

- Adham - FTC 14875 LightSpeed
- Arjun - FTC 9794 Wizards.exe
- Baylor - FTC 10641 Atomic Gears
- Cole - FTC 7548 SPAREPARTS
- Cooper - FTC 19458 Equilibrium.exe
- David - FTC 2753 Team Overdrive
- David - FTC 7236 Recharged Green
- David - FTC 8680 Kraken Pinion
- Dom - FTC 15692 Rust in Pieces
- Eric - FTC Alum - 8417 'Lectric Legends
- Ethan - FTC 7236 Recharged Green - goBILDA
- Frank - FTC 8581 Aedificatores
- Frank - FTC 8581 Aedificatores
- Fulton - FTC 5143 Xcentrics
- Guineawheek - FTC Alum
- Ian - FTC 7842 Browncoats
- Jackson - FTC/FRC® Mentor/Alum
- James - FTC 14298 Lincoln Robotics
- Justin - FTC 9656 Omega
- Karter - FTC 5975 Cybots
- Kelvin - FTC 731 Wannabee Strange
- Keval - FTC 731 Wannabee Strange/FTC 10195 Night Owls
- Kevin - FTC 9048 Philobots
- Nate - FTC 12897 Newton's Law of Mass
- Navya - FTC 7149 ENFORCERS
- Neo - FTC 6710 Sigmas

- Peter - FTC 12533 Inception
- Sam - FTC 2753 Team Overdrive
- Shurik - FTC 4137 Islandbots mentor
- Tom - FTC 3736 Serious Business
- Tyler - FRC 3184 Blaze Robotics
- Wes - FTC 3658 Bosons
- Whimsy - FTC 8417 'Lectric Legends

Hosting

- Benjamin Ward - FRC/FTC Alum - Copperforge - blward@copperforge.cc

Index