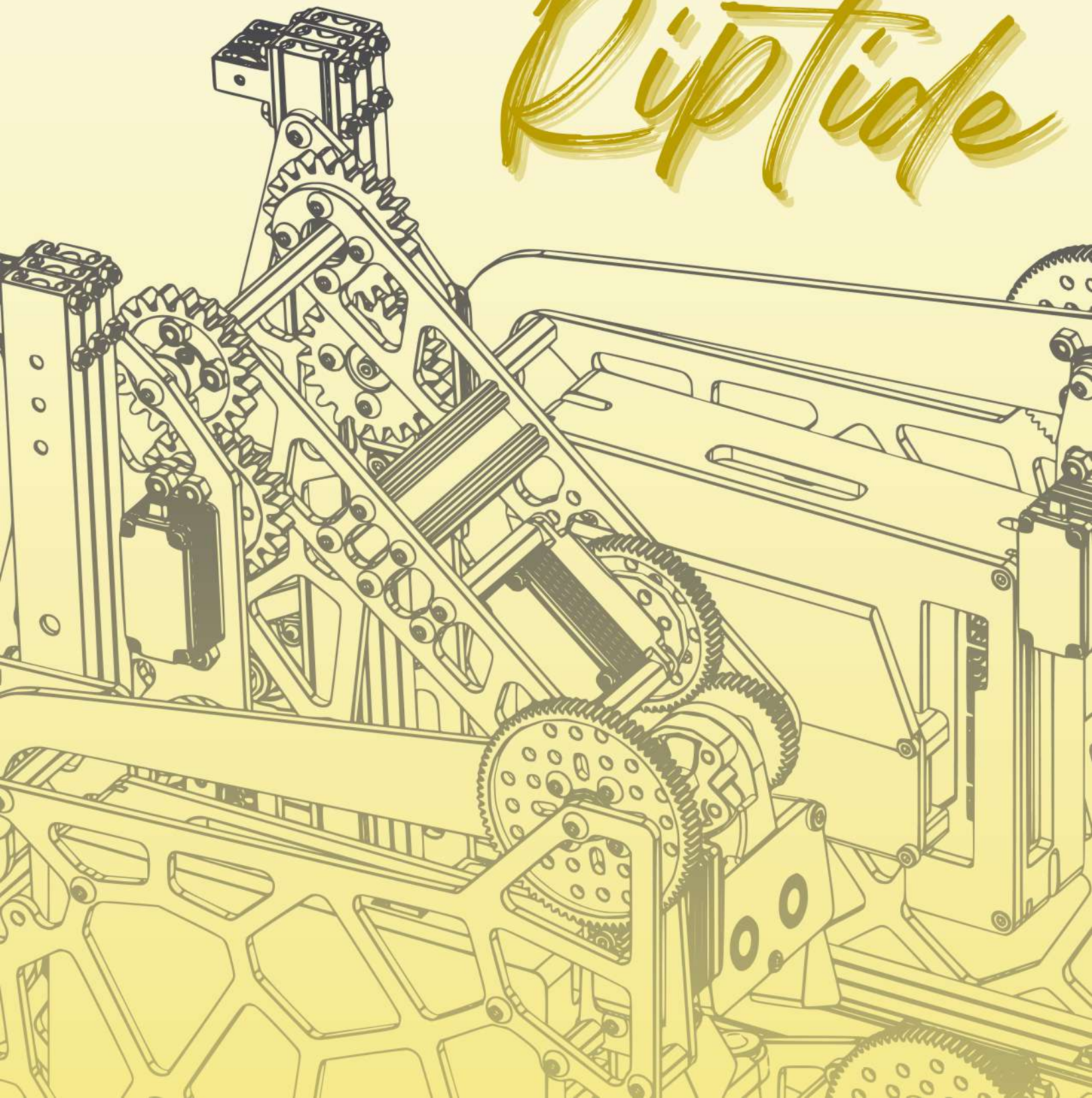
# FTC 23511 - Seattle Solvers

## Into the Deep 2024-2025

## Engineering Portfolio

# Riptide



# Team Organization and Plan

## Mission Statement

The purpose of the Seattle Solvers shall be to **learn new skills**, **promote STEM** in our communities through programs involving engineering and other technical skills, and most importantly, **have fun**!

## About Us

We are the **Seattle Solvers**, a former FLL team, and this is our **second year competing in FTC**, with our members having varying grade levels in middle and high school and all having 2-4 years of FIRST experience.

## Team Structure

Our team is entirely **student-led**, meaning we create our own meeting agendas, coordinate all project management, and season planning ourselves. This **builds our leadership skills** and gives responsibility to specific individuals to complete tasks.

## Team Season Goals

Building onto our overall team mission statement, some of our goals going into the season were getting the chance to go to **worlds**, **growing our FLL classes initiative**, and enjoying being in the FIRST community whether it be with song parodies or silly pictures!

### Ishaan
**Captain/Co-Build Lead**

Planned meetings & project management

### Saket
**Design/Co-Build Lead**

Improved CAD skills and learned how to render

### Arush
**Programming Lead**

Founded and worked to start SolversLib

### Sarannya
**Outreach Lead**

Planned achievable & realistic outreach events

### Erin
**Finance Lead**

Found fundraising and financial opportunities

### Viraj
**Build/Design**

Contributed to building intake and deposit

### Ajay
**Programmer/Driver**

Learned how to plan autonomous paths

**In addition to our team plans, all of our members also had their own personal learning, contributions to the team's performance, and goals improving themselves for the future!**

## Sponsors

We thank our sponsors that allow us to compete in FTC **without having any entry fees** for members!

polymaker

HAAS Gene Haas Foundation

onshape

DUNN LUMBER Building success together.

FIRST WASHINGTON

Brown Bear Car Wash

# Sustainability

We formed our team to be self-sufficient and last longer than an individual

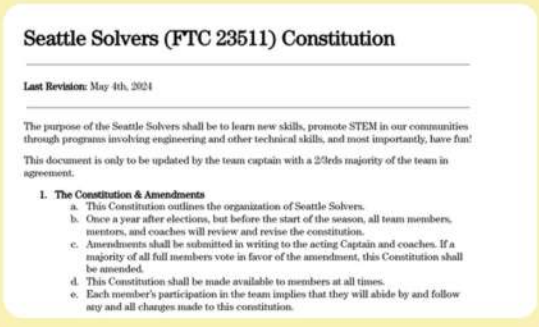
We ran a fundraising event, **selling food, running games**, and talked with interested people about FTC.


## New Members

Any person who wishes to join our team receives a standardized and fair interview. New members are **provided with the resources and opportunities** to learn by their subteam leads, who oversee their training, while still being immersed in team activities.

## Constitution

We created and follow a team constitution to ensure that everyone follows ground rules to keep the team respectful and on track. This is just one of the ways we treat the team as **an organization** rather than a group to ensure its continuity in the future.



**Seattle Solvers (FTC 23511) Constitution**

**Last Revision:** May 4th, 2024

The purpose of the Seattle Solvers shall be to learn new skills, promote STEM in our communities through programs involving engineering and other technical skills, and most importantly, have fun!

This document is only to be updated by the team captain with a 2/3rds majority of the team in agreement.

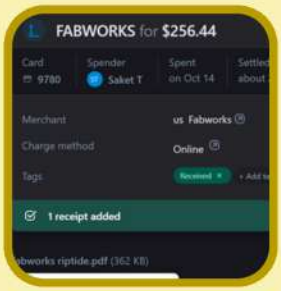
1. **The Constitution & Amendments**
   a. This Constitution outlines the organization of Seattle Solvers.
   b. Once a year after elections, but before the start of the season, all team members, mentors, and coaches will review and revise the constitution.
   c. Amendments shall be submitted in writing to the acting Captain and coaches. If a majority of all full members vote in favor of the amendment, this Constitution shall be amended.
   d. This Constitution shall be made available to members at all times.
   e. Each member's participation in the team implies that they will abide by and follow any and all changes made to this constitution.

## Finances

As a self-sufficient team, we have to **obtain all of our funds by ourselves** through events and sponsorships.
We also keep **track of all purchases** by the team through Hack Club Bank. It allows our members to make **purchases without reliance on adults**, allowing our team to operate more efficiently and independently.

Our team accepting a check from FRC 1294, our sponsor (Left)

An example of an order in Hack Club (Right)




## Task Management

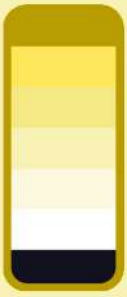
We utilize tools such as **timelines** and **task lists** to **stay on track** during the season. During our weekly meetings that are entirely student-led and organized, we go over the tasks that need to be completed in the coming week and assign them with specific people, priorities, and deadlines to ensure they get done in a timely manner.

| ✓ | Priority | Person(s) | Due Date | Task (in as much detail as possible) |
|---|---|---|---|---|
| ☐ | 2 | Arush + Ajay | 11/10 | Re-zero and tune deposit pivot servos |
| ☑ | ~~2~~ | ~~Arush + Ajay~~ | ~~11/10~~ | ~~Add new specimen intake style for TeleOp~~ |
| ☑ | ~~1~~ | ~~Arush~~ | ~~11/14~~ | ~~Run FTCLib Actions with RoadRunner 1.0~~ |
| ☐ | 1 | | | Tune RoadRunner with OTOS |

An example the software task list from mid-November.

## Standardization

We standardize all **materials, team colors, fonts, logos, and more** to ensure a consistent and continuous marketing styles for future documents and advertisement materials. This allows us to have a more **cohesive branding and be memorable** to the audience and gives members a starting point to promote the team.



Solvers Light Theme

# Learning Process

To facilitate our growth, we learn from others in our community.

## Coaches/Mentors

To help achieve our goal of learning, we get mentors and coaches by contacting them to help us with their knowledge. Our coaches help give feedback on the team planning as a whole, while many mentors are knowledgeable in FIRST and help us learn more technical skills.

**Zheng Z.**
Head Coach
Programming Coach

**Srini Y.**
Programming Coach

**Kai G.**
Business Mentor
FTC 23383 Alumni

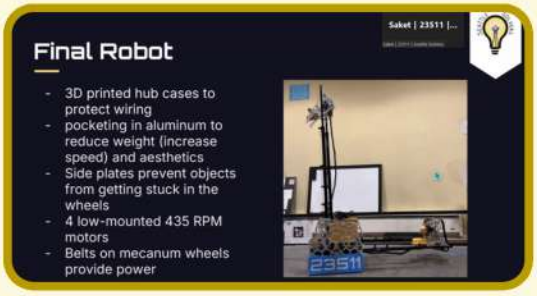
**Anish**
Programming Mentor
FTC 16700 Alumni

**Sanchit A.**
Hardware Mentor
FRC 1294 Alumni

**Aarsh M.**
Programming Mentor
Business Mentor
FTC 12689

**Aaditya A.**
Programming Mentor
FRC 1294 Alumni

Sanchit overseeing our cutting of steel gears for our summer swerve drive.



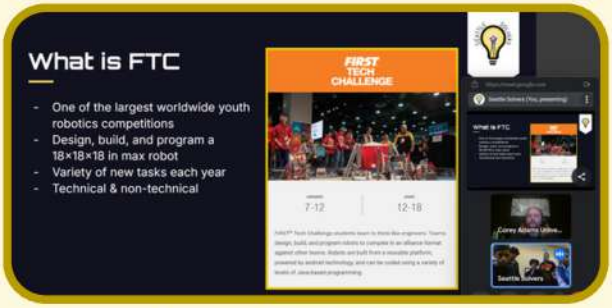

Us presenting at the Unify to Diversify Conference 2024

## Teams

We engage in **sharing and learning** from other teams as well! We met with FTC 14380 Blue Bot Builders to discuss our experiences and learnings at the end of last season and participated in the FTC Unify to Diversify Conference held by 14179 Sushi Squad about robot evolution.

## Universal Robotics

We connected with Universal Robotics, a leading company in the robotics and automation industry. We **receive feedback** from them on our robot design and programming concepts, and we are currently working to **schedule a "Careers in STEM" webinar** with them.



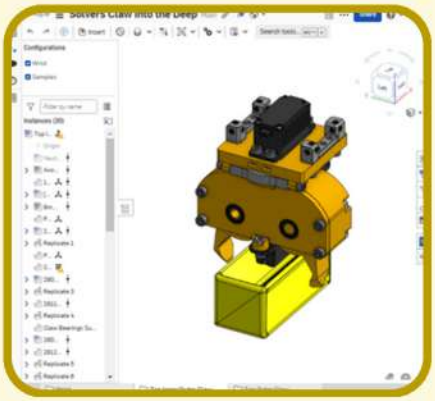

Our initial meeting with Universal Robotics

# Sharing with the Community

We spread the knowledge we gain with the FTC community!

## FTC Open Alliance

As rookies last year, we benefitted a lot from open-source projects and designs from others to inspire our own. This year, we are one of the few early teams to join the FTC Open Alliance that pride ourselves in **open-sourcing all of our code, CAD, and sharing live updates as the season progresses** with the rest of the world. Our in-season robot CAD document for example has **230+ copies and 2400+ links** on Onshape.

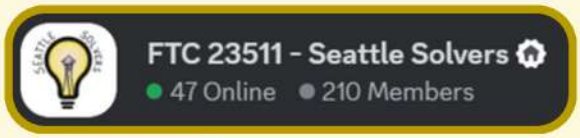


## Solvers Claw

The Solvers Claw is a **continuous project** containing different claws we make each season. Started last season, and having 4 different types of claws, the document is **opensource and available for any to use and learn from**. Currently, the document for just this season's claws has **210+ copies and 1400+ links** on Onshape, not to mention any exports to other CAD software.

## Social Media

We share updates about our progress as a team throughout the season on our team and unofficial global FTC Discord servers. We also post mainly on our team Instagram with **400+ followers**.




@ftc23511



## GDC R104 Animation

Our team created an **animation to explain R104 Extension limits**, eventually being **shown in multiple kickoff events around the world** and even being used in the creation of the **official R104 Recap Video** released in October. The original video on our YouTube channel has over 570 views but has been sent countless times in the format of GIFs and in presentations outside.

# Outreach

We make connections and spread FIRST in a variety of ways.

## West Coast Invitational (WCI)

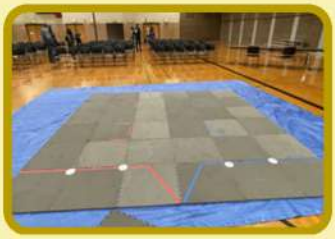
**6/15/2024**

We **co-founded, co-ran, and competed in WCI**, the first offseason invitational for FTC teams across the United States West Coast, **free of cost** to make it accessible for any interested team. Taking multiple leadership roles, we lead the website and award design, helped set up the match stream, and helped set up the competition on-site.

Collaboration while acting **graciously professional** was a key part of our success. We worked with teams FTC 3805, FTC 9880, FTC 19745, and FTC 23383 to start the event as well as working with some FRC demo teams to promote FIRST!

**Reach: ~14 FTC teams**

Man Hours: ~36 hours



Helping set up the fields (left) and one of a trophy we designed (right)

## Mentoring FIRST Teams

**September 2024 - Present**

- FLL Red Puffins (in-person): all parts of FLL
- FTC 22105 (online): design and CAD
- FTC 23521 (online): design and CAD

Note: We have **mentorship messages/letters** from all of these teams for documentation in our pit as well.

**Reach: 3 teams**

Man Hours: ~15 hours

One of our online chats with FTC 23521 doing a design review

Sarannya with the Red Puffins

## Assisted FTC #26015 Otter Bots

**July 2024**

We helped fund and start rookie team **Otterbotz (FTC #26015)** by **donating over $1000** worth of FTC parts (including cataloguing, organizing, and packing parts) and **assisted** them in design topics like pocketing.

**Reach: 1 team**

Man hours: ~6 hours

## High School Robotics Options

**6/12/2024**

In Evergreen Middle School, we worked with Ignite Robotics and FRC 1294 to present at an event to teach graduating middle schoolers about possible high school robotics options.

**Reach: ~20 people**

Man Hours: ~9.5 hours

## Blackwell Elementary STEM Night

**2022 - Present (3/28/2024)**

At Blackwell Elementary School, we **showcased FLL and FTC**, letting kids program and drive the robots on an old FLL mat and presented about FLL to increase interest in FIRST.

**Reach: ~40 people (2024 only)**

Man Hours: ~16 hours

# Spreading LEGO® Robotics

*"As a parent I really loved the enthusiasm and dedication of the Seattle Solvers."*
*"They are the role models for the younger kids."*
*"The members of the team are always willing to help"*
- Parents of students from our classes

## Background

Our team competed in **FLL (FIRST LEGO League) for 2 years** prior to our time in FTC. The first year, our team was the highest scoring team in our region. With our goal of spreading STEM, we decided to start classes in hopes of training kids in good practices for the competition.

## Intro To Lego Robotics Classes

**2021 - Present**

We run our own **custom designed** Intro to Lego Robotics course, teaching **over 70 kids** to date. Using our experience as a former multi-award winning FLL Team, spending **hundreds of hours to design** activities, worksheets, and notes made to help introduce kids to the robotics world, this course serves as a feeder into FIRST programs in Washington and as a paid class, also **helps fund our team.**

One of our classes near the beginning of the program in 2021 (top)

A flyer/advertisement of our summer classes this offseason (below)

*Intro to Lego Robotics*

By Seattle Solvers (FTC #23511)

Go back to the steering wheel analogy. When you turn 'right: 100', you aren't turning right 100 degrees. You are turning the steering wheel right sharper to get a tighter turn. When you turn "left: -50", you aren't turning left 50 degrees. You are turning the steering wheel left less sharply to get a looser turn.

Spin turns are more precise and need less room than pivot turns.

Yaw Angle:

In SPIKE PRIME, the yaw angle is how much your robot has turned (horizontally). It's measured by the gyro sensor, which is built into the hub.

For example, if the robot turns 90, then the yaw angle would be 90.

Roll Yaw Pitch

Coding Turns:

This is the code for a right spin turn:

Let's work through this.

The yaw angle must be set to 0 in the beginning. Here's why: your yaw angle will be at a random number of degrees since you naturally turn your robot when coding and testing. And that will screw your code up since it relies on stopping. If I want to turn 90 degrees and my yaw angle is at 47 degrees, then my robot will only turn 43 degrees instead. Because after 43 degrees of turning, then the robot will hit the 90 degree mark and stop.

Next, your robot spin turns right as it's set to 'right: 100'. This means the robot turns at sharpness 100, which is a spin turn.

The robot turns until the yaw angle is greater than 89 degrees. Not 90. 89! That's because it will stop turning when it's 1 degree greater than the set value. So when it's 89, it'll stop at 90. When it's 90, it'll stop at 91.

Page 3

## Growth

Students fill out a **feedback survey** at the end, and we incorporate that into our next classes. Some changes we have made include:

- More hands-on exercises rather than just theory.
- As classes expanded, we **created worksheets and guided notes** to help students learn (see left for example)


## LEGO Robotics Education in Rural India

The Samata Equal Footing Foundation is sponsoring our **creation of the curriculum and plan for** the starting of LEGO Robotics classes in schools in rural India, specifically targeting areas where this education would be unavailable. The program is being piloted at the nonprofit school RISE, based in the Samiyandipudhur, Tamil Nadu, India.

The classes are scheduled to begin in the **upcoming academic year**.

Man Hours (to date): ~20 hours

RISE
Rural India Student Education

# Robot and Design Overview

To solve any problem, we follow the engineering design process.

### Identify

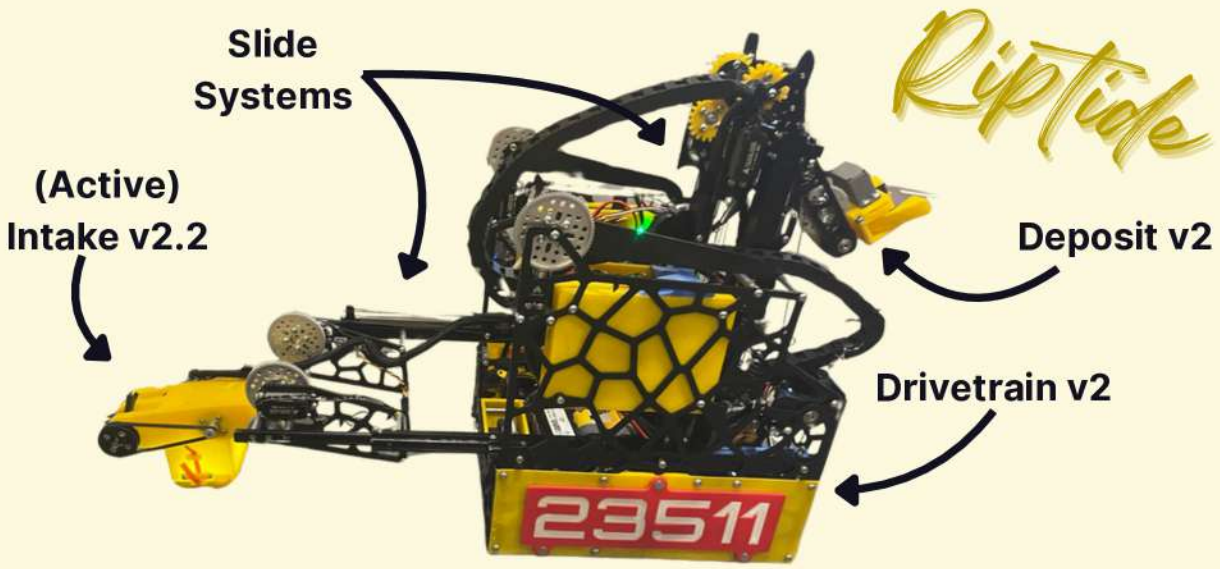
**Identify** the problem that needs to be solved/the objective.

### Design

Brainstorm possible solutions to the objective and **design** prototypes.

### Create

**Create** a product based on the most promising design/prototype.




### Test

**Test**, gather data, and analyze how the current design performs when used

### Iterate

**Iterate** and improve on the solution based on the tests and data gathered

### Communicate

**Share** the design and findings to get feedback and help others!

## Keep It Simple, Straightforward (KISS)

A key factor we consider in our design decisions is **simplicity** - whenever possible, we choose the simplest possible solution that doesn't sacrifice any of the design requirements we set. Most often, the simplest solution is the most robust/reliable and easiest to maintain. If needed, we **iterate to add complexity** until we have a competitive design.

## Fast Iterations

Our team makes many changes & optimizations to our robot throughout the season. We utilize tools such as 3D printers and laser cutters to help us make **quick, inexpensive iterations** on our robot mechanisms.

Our BambuLab X1C 3D printer operating

Cutting our v0.9 robot plates at Microsoft's makerspace

# Drivetrain

## Design Requirements

- As **small** as possible (~12"x12")
- Low center of gravity to prevent tipping
- **Simple** for **maintenance** and **reliability**
- **Rigid** to allow other subsystems to be mounted sturdily

## Drivetrain Type

Even though we **developed a coaxial swerve drive** over the offseason (see it in our pit!), we chose to use mecanum because of its simplicity and reliability.

Our summer swerve module design

## Iterations (v0.9 → v1.1 → v2)

**Pros:**

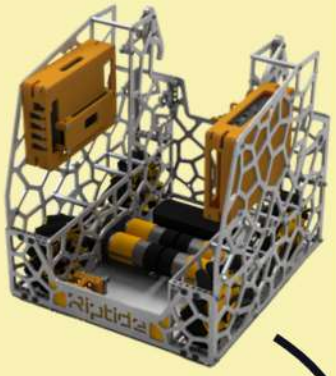
- Easier maintenance and wiring (v0.9 → v1.1)
- Hubs don't collide with intake (v0.9 → v1.1)
- Hardware for Level 3 Ascent (v1.1 → v2)
- Pinpoint hardware for better localization (v1.1 → v2)
  - See programming pages for more info!
- Added sweeper servo (v1.1 → v2)
  - Makes space for intake to pivot into submersible
- Added servo hang for level 2 ascent (v1.1 → v2)

**Cons:**

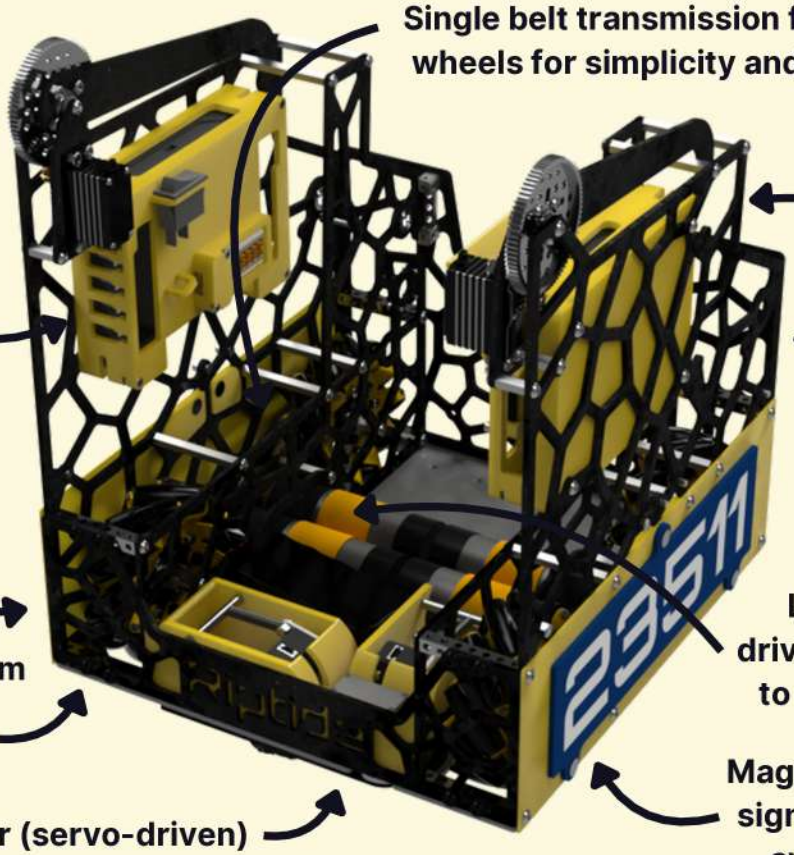
- Increased weight and center of gravity (v0.9 → v1.2)

**v0.9**

**v1.1**



**v2**




**Axon MAX+ driven hooks for level 2 ascent**

**Single belt transmission from motors to wheels for simplicity and maintenance**

**Custom hub cases to protect wiring**

**Pocketing in aluminum to reduce weight and aesthetics**

**Omni-directional (holonomic) drivetrain with mecanum wheels**

**Custom aluminum (parallel) plates**

**Low mounted 435 RPM drivetrain motors and battery to reduce center of gravity**

**Front and back aluminum plates to increase torsional rigidity**

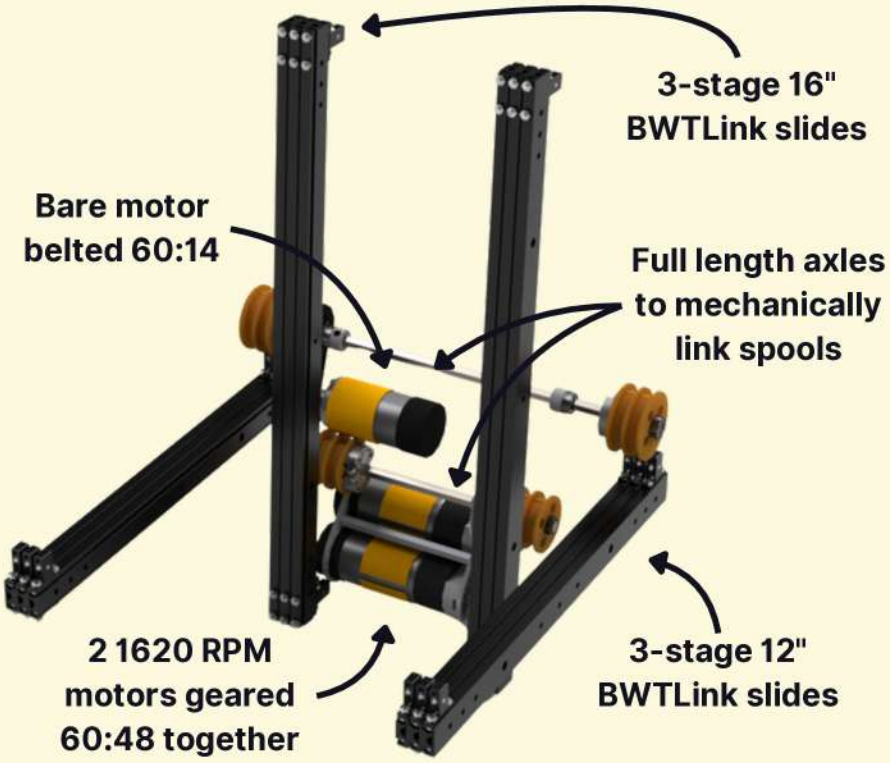
**Magnetic robot signs for easy switching**

**Sweeper (servo-driven)**

# Extension Systems

## Design Requirements

- **Rigid** even while extended for consistency
- **Compact** while retracted to keep robot small
- Intended to have **precise control** (for intake adjustments)
- Be able to lift the robot for level 2 ascent
- Must **fit in a 20" x 42"** rectangle **when extending**



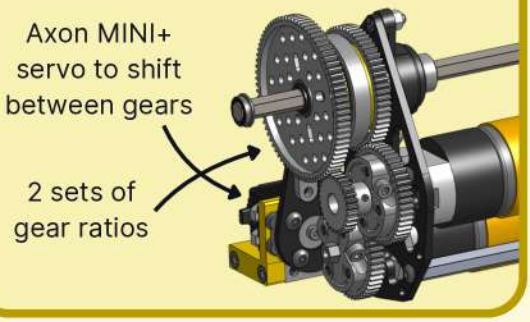

## Calculations (for ascent)

- 12 kg. robot × 175% safety factor → **~21 kgf needed**
- 2 1620 RPM motors = 5.1 kg.cm stall torque × 2 motors
  - 10.2 kg.cm × (80:28 gear ratio) = ~29.2 kg.cm
- ~29.2 kg.cm ÷ 1.4 cm radius spool = **~21 kgf output**

## 2-Speed Gearbox

- Noticed that slides were the **bottleneck** in our cycles on v1
- Need higher speed in tele-op but need to hang in endgame
- Manual shifting between 2 gear ratios/speeds for vertical slides
  - High speed for general use
  - High force/torque for hang

V1 of the gearbox didn't include a clutch and required very precise movement from the servo and gears to shift. V2 with an in-built clutch is **currently in progress** and being tested before being mounted onto robot (seen below).




## Rigging/Stringing

- 2 **continuous strings** winched on a double spool
- Maintaining **tension** is critical to precise control
  - Slack leads to rotary motion not converting to linear

### Top-down view of stringing/rigging extending slides

**Spring** maintains tension in strings even with variation in string released

**Double spool** releases 1 string and retracts 1 string

**Extension string** is winched in, the gap between the pulleys decreases, extending the slides

**Retraction string unspools**, allowing the slide to move a limited amount forward

# Solvers Claw and Intake v0/1

## Design Requirements

- Intended to to pickup samples regardless on how many samples are around it
- Capable **auto-detect** and complete the sequence of intaking
- **Minimizes weight and area** that has to go over the submersible barrier
  - Allows picking up samples close to the edge of the barrier
  - **Lighter is faster** for slide systems

## Claw Iterations

We have many iterations of our Solvers Claw that we couldn't explain in full detail here, so be sure to **come to our pit**!

## Intake Iteration (v0 → v1)

**Pros:**

- Aluminum body and cross bracing makes intake **rigid**
- Servo latch **corrects more variation** of sample for transfer

**Cons:**

- Increased weight and complexity

Intake v0 (prototype)

## Herringbone Gears

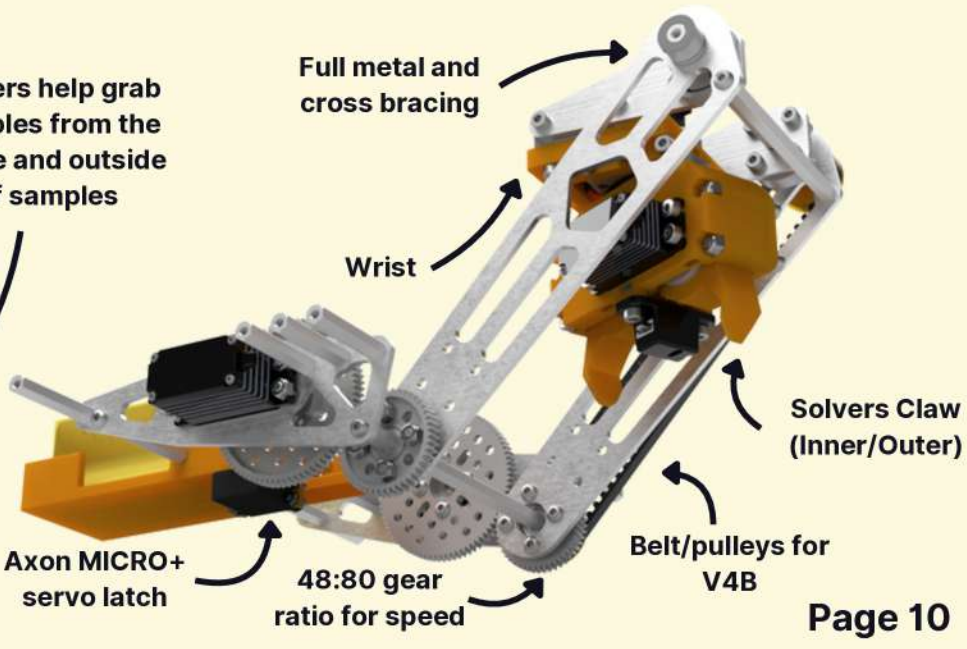
- Gear type with angled teeth
- Gears **self-lock**, removing any **axial load**

Used in our 3D printed Solvers claw fingers!

## Calculations

To ensure we are running our servos at **peak power efficiency to minimize current draw**, we calculated how much torque we need and what gear ratio we would run to ensure we meet that requirement.

- 0.75 kg weight × 22 cm length × 200% safety factor → **33 kg.cm needed**
- 2 Axon MAX+ servos = 28 kg.cm (@ 4.8v) stall torque × 2 servos
  - 56 kg.cm × **48/80 (gear ratio)** = **33.6 kg.cm output**

# Intake v2

## Design Requirements Revisions

After LM1 we **analyzed our matches** and had a few key takeaways:

- Drivers are able to see the submersible, but **took longer to align to samples with fine adjustments** to the yaw and lateral axes
- Intaking samples with many samples around them isn't as critical as we thought before because there are more samples than used in a match and 75% of the samples are legal to be scored

These requirement revisions **led us to iterate** and switch to an active intake.

## Average time to intake sample

v1: ~4.5 seconds
v2.1: ~1.5 seconds
v2.2: ~x seconds

**3 times faster**

## Iteration (v2.1 → v2.2)

**Pros:**

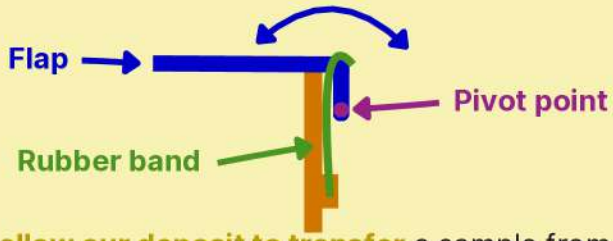
- Smaller and lighter (better for pivot)
- Only 1 set of counter sprung flaps make transfer faster as deposit no longer needs to be as precise to fit
- Less area needed to deploy

**Cons:**

- None (apart from development resources)

**Intake v2.1** which required deposit to grab from the middle

## Counter Sprung Transfer Flaps




To **allow our deposit to transfer** a sample from the intake, we utilize counter sprung flaps. As the flaps open, the tension in the rubber bands increases as the flaps pull the rubber bands with them. After the sample is completely out of the intake, the flaps spring back, due to the tension in the rubber bands.

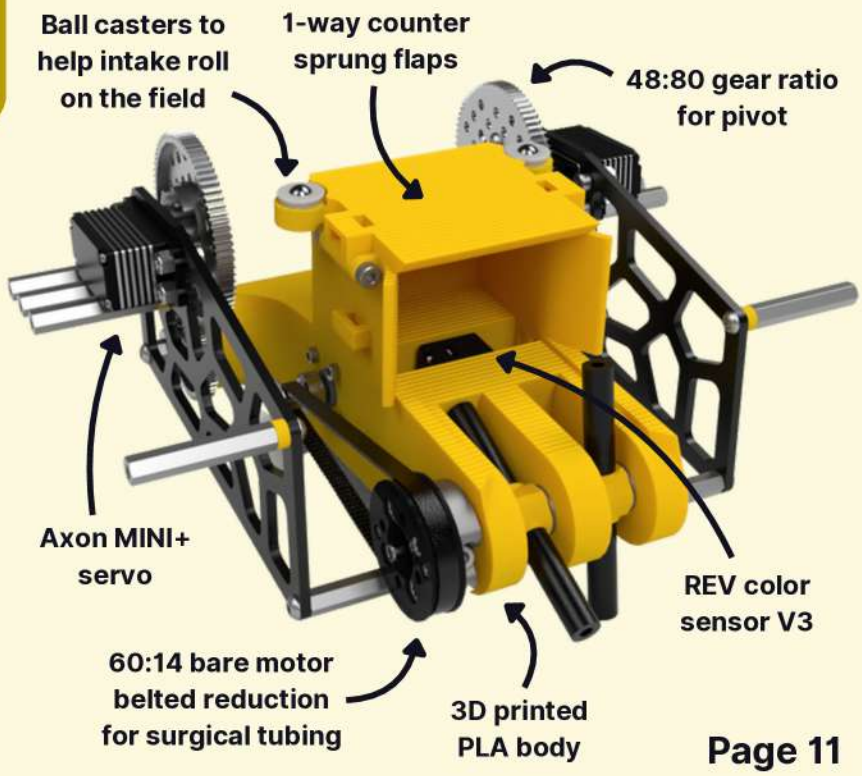
## Programming Automation

**Detecting Samples**

- Uses REV color sensor to detect both distance and RGB for precise information
- Automatically reject wrong color samples to avoid penalties
- Automatically transfer to deposit without any driver input if correct sample

**Transfer**

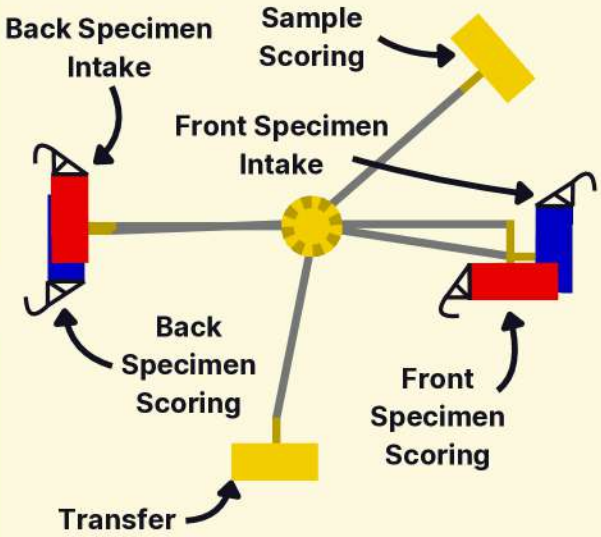
- Automated power to tubing to hold sample in at high retraction/pivot speeds



# Deposit

## Design Requirements

- Able to score **samples** in both baskets and **specimens** on high chamber
  - Able to grab on to samples/specimens with a strong grip
- Able to **transfer** samples from the **intake** system
- Hooks for lifting the robot




## Iteration (v1 → v2)

**Pros:**

- **Longer arm and wrist** allows us to intake and score specimen from **either side**, improving cycles
- Claw able to align to more positions precisely (shown to the right)

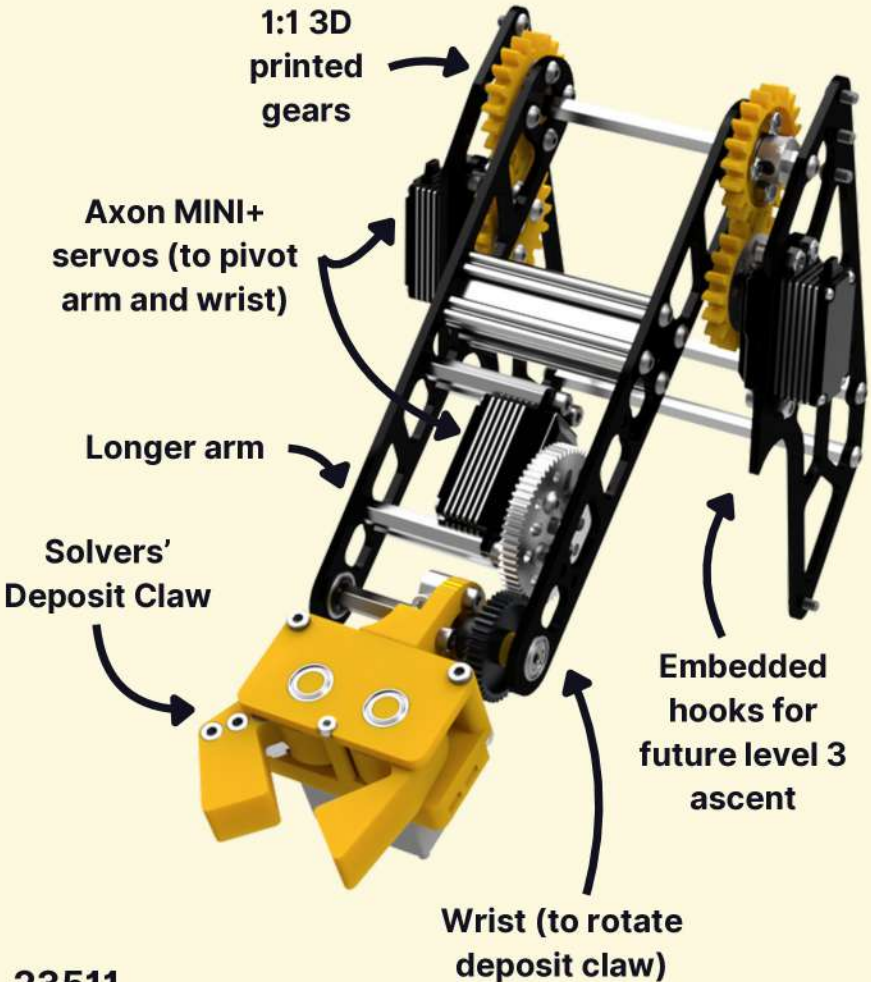
**Cons:**

- None (apart from developmental resources)

Deposit v1

## Programming

- Use **set points** with servos for each position that are easily tuned via test programs and FTC Dashboard
- Move slides, wrist, arm, and claw in a **known, automated order** to prevent collisions
  - e.g. close claw before arm pivoting to sample scoring




## 3D Printed Gears

We opted to try PLA (3D printing material) gears as it allowed us to print **higher module gears**

- Bigger teeth means less chance the gears can skip
- 3D printed gears can be reprinted fast to reduce backlash (gap between gears)
- Theoretically worse for impact loads/durability over time; currently being tested

# Sensors & Localizers

We make our robot more robust and reliable with feedback from sensors.

## Sensors

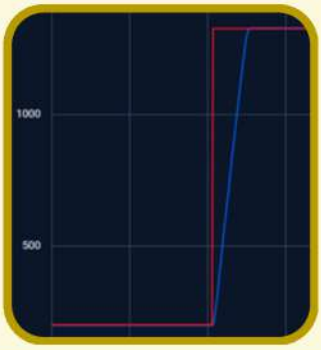
- The REV Color Sensor V3 detects both **color and distance** (accurate up to 10 cm)
  - **Automatically ejects wrong color sample** from intakes
  - Also checks distance to ensure that there is a sample inside the active intake, **automating the transfer sequence** from the intake to the deposit
- External REV through-bore encoders (REVcoder) tracks extension of slide systems and used in **PDF controllers**, and runs at a much **more accurate 4096 ticks per revolution** (the built-in motor encoder runs at 103.8 ticks per revolution)
  - The REVcoder won't slip even if power transmissions (belts/gears) slip

REV Color Sensor V3

REV Encoder

## PDF Controllers

- **P**roportional component to system error
- **D**erivative/dampener component to reduce overshoot with system velocity
- **F**eedforward component to hold the system at a given point (e.g. power to keep slides held)




**Critically dampened** PDF controller on our vertical slides - no overshoot, oscillations, or steady state error, allowing for **fast and precise** control.

**Red** = set point/target
**Blue** = encoder position

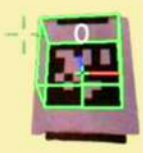
## Localization/Odometry

Localization or odometry is the process of **tracking where the robot is on the field**, and can be **absolute** (returns value of robot on the field based on fixed locations), or **relative** (sums small movements often to track robot movement on the field). By combining both types, we are able to correct any built-up or human error with robot set-up and track the robot at any point in time.

## OTOS + Limelight 3A (v1)

Until Interleagues and on our v1 robot, we used the OTOS and Limelight for localization, but found it to be **inaccurate (±3" of error) over longer autonomous paths** with fewer AprilTags visible at closer ranges and required more maintenance at different fields with varying tile sink.


Optical Odometry Tracking Sensor (**OTOS**)




**Limelight 3A** camera giving absolute positions of the robot based on fixed **AprilTags** (QR code fiducials)

## Pinpoint Odometry (v2)

After Interleagues, we switched to Pinpoint. Pinpoint is a **co-processor** that takes in information to **calculate our robot position faster and more precisely**. Pinpoint runs at 1,500hz (loops per second), while most FTC dead-wheel systems run at 50-200hz.

2 goBILDA Odometry Pods and Pinpoint

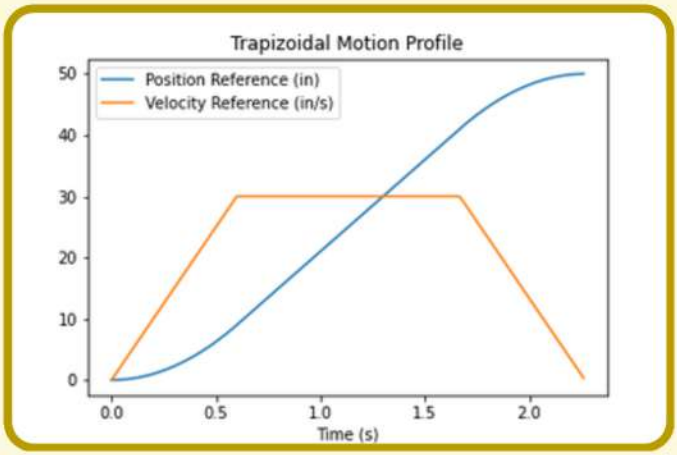
# Motion Control & Path Planning

We use a variety of methods to ensure efficient and accurate movement.

## RoadRunner 1.0

For our first League Meet, we utilized **RoadRunner**, an open-source library used to follow splines. There are pros and cons to using this library however, mainly that it utilizes motion profiles which are an add-on to a PID control loop.

The main limiting factor is that motion profiles limit the acceleration and deceleration of the robot to reduce slip and inaccuracies, therefore **slowing it down** in certain acts of movement as it values time consistency over everything else (most notably speed as mentioned).




LM1

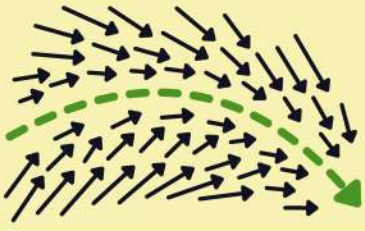
## MeepMeep

MeepMeep is a program that allows you to easily generate paths for RoadRunner, we used it to plan out our routes in an offline environment, and then loaded our paths onto the robot, into an **"online" production environment**.

## Pedro Pathing

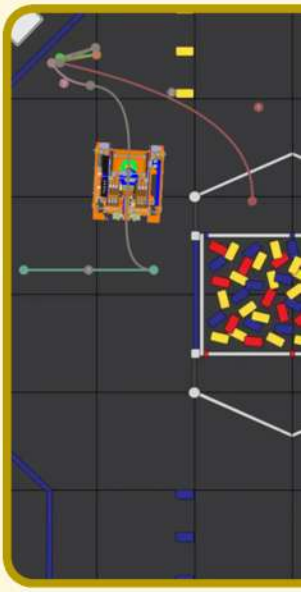
Later, we began using Pedro Pathing, which utilizes a **guiding vector field & lookahead algorithm.** This allows our paths to not be limited by their speed, which is a primary downside of using RoadRunner.

LM2+



To the left is a simplified diagram of a **corrective vector field**, where the black lines represent the **corrective vector** at any specific point in space and the green line is the **ideal trajectory** being followed

It also performs correction for **centripetal force** of our robot, allowing us to "glide" into the Bezier curves we are able to generate with an online path planner similar to MeepMeep.



# Programming Architecture

We use a finite state machine with our Command System to control our robot.

## FTCLib / SolversLib Command Base

We **forked (modified) the library FTCLib** into our own **open-source version called SolversLib** because it was no longer being maintained, which allowed us to add new and modern **features like PS4/5 gamepad support & bug fixes**. We use its command base to control our commands and structure our programs. It handles **resource management** to make sure two commands don't use the same subsystem, as well as making sure the starting, body, and ending of each command is correctly handled.

## Commands

setIntake(IntakeMotorState, IntakePivotState, extendoTarget)

setDeposit(DepositPivotState, clawOpen, slideTarget) → transfer()

Uninterruptable → Interruptable → undoTransfer()

## Finite State Machine (FSM)

By incorporating a FSM with the help of our mentor Polar, each individual subsystem and the robot as a whole is controlled in set states that **prevent collisions** between subsystems and game pieces within the robot, and allows us to easily **handle and set different states autonomously** in both tele-op and auto.

## Using a FSM

Our FSM is at only one state at a time, and can move to another state if certain requirements of sensors & encoders are met. For example, if the intake color sensor detects the wrong color, it will **automatically switch to the ejecting state**. Once there is no longer a sample inside (the color sensor also measures distance), it will then **revert back to the intaking state**.