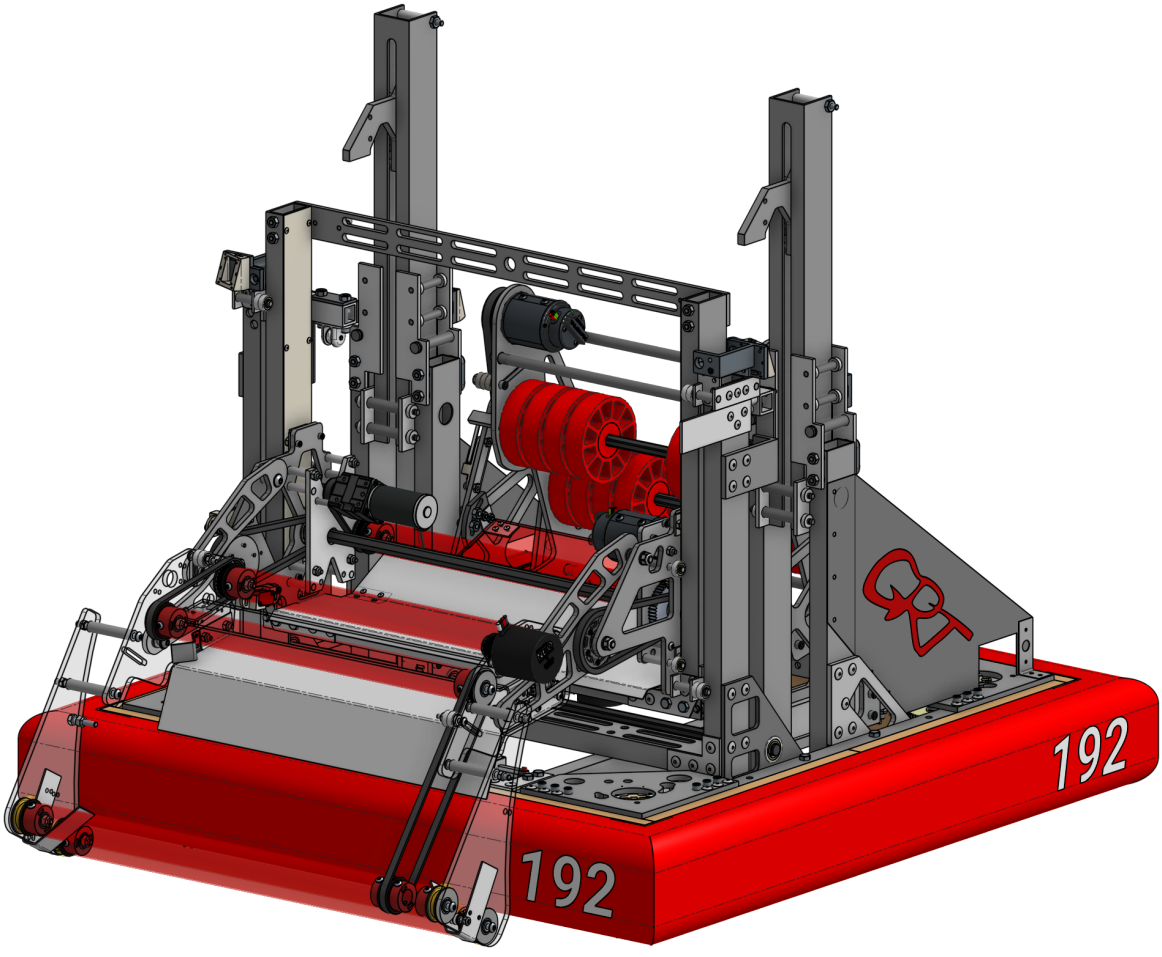
# GUNN ROBOTICS TEAM

## 2024 Technical Binder



# CONTENTS


Game Analysis.....4

Introducing Dissonance.....7

Drivetrain.....9

Mechanisms.....19

| Intake 20

| Elevator 22

| Climb 25

| Shooter 27

Controls.....31

# Thank you to our generous sponsors, who make everything we do possible

BOSCH

PALO ALTO UNIFIED SCHOOL DISTRICT

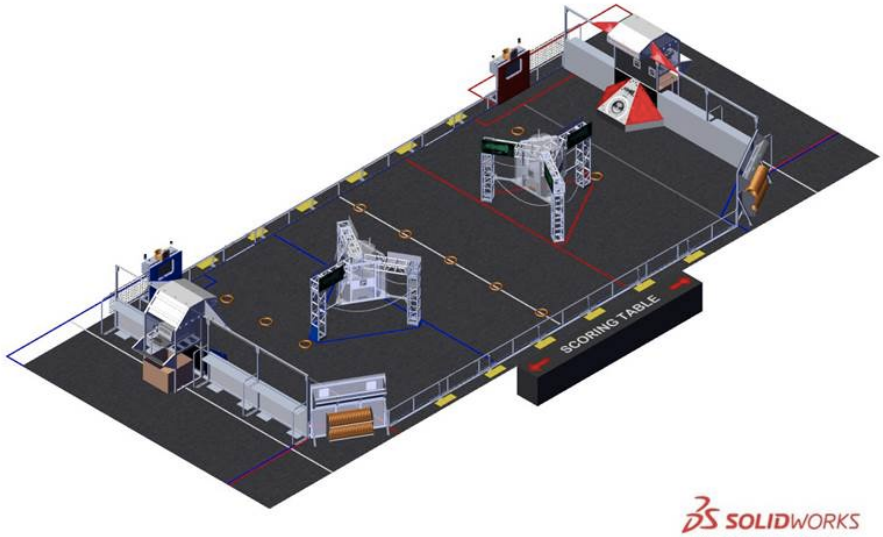
FIRST CALIFORNIA ROBOTICS

PALO ALTO Partners in Education OUR EDUCATION FOUNDATION

# Game Analysis

# CRESCENDO

In the 2024 FRC season game, Crescendo, robots are given the task of collecting Notes (foam rings) from the ground and Source. They earn points by scoring in the Amplifier and Speaker, two locations on the field. In endgame, robots race to lift themselves onstage via the chain, and score in the Trap.




## DELIVERABLES AND STRATEGY

- Determined goals for the robot as a team during kickoff weekend
- Aimed to be able to score into the Amp, Speaker, and Trap
- Prioritized scoring in the Amp over the Speaker, since Amplifying leads to more points than just shooting in the speaker alone
- Considered that other robots would prioritize speaker scoring, so focusing on Amp scoring would give us an edge
- Advantageous to utilize a swerve drivetrain, which we designed and machined in-house to tailor to our needs.

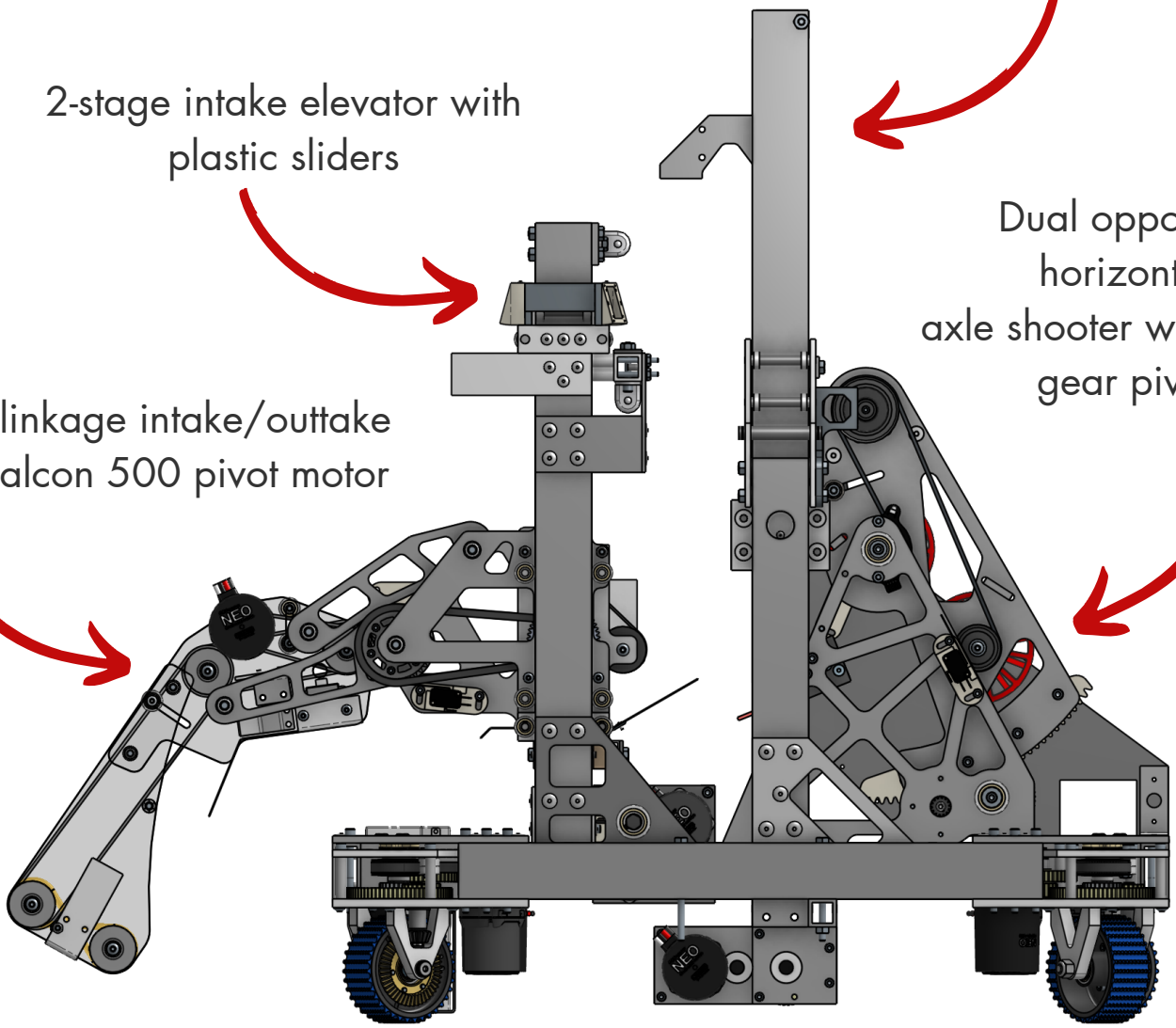
# INTRODUCING
# DISSONANCE

Dual two-stage climb arms designed and machined in-house

2-stage intake elevator with plastic sliders

Dual opposing horizontal axle shooter with sector gear pivot

4-bar linkage intake/outtake with Falcon 500 pivot motor

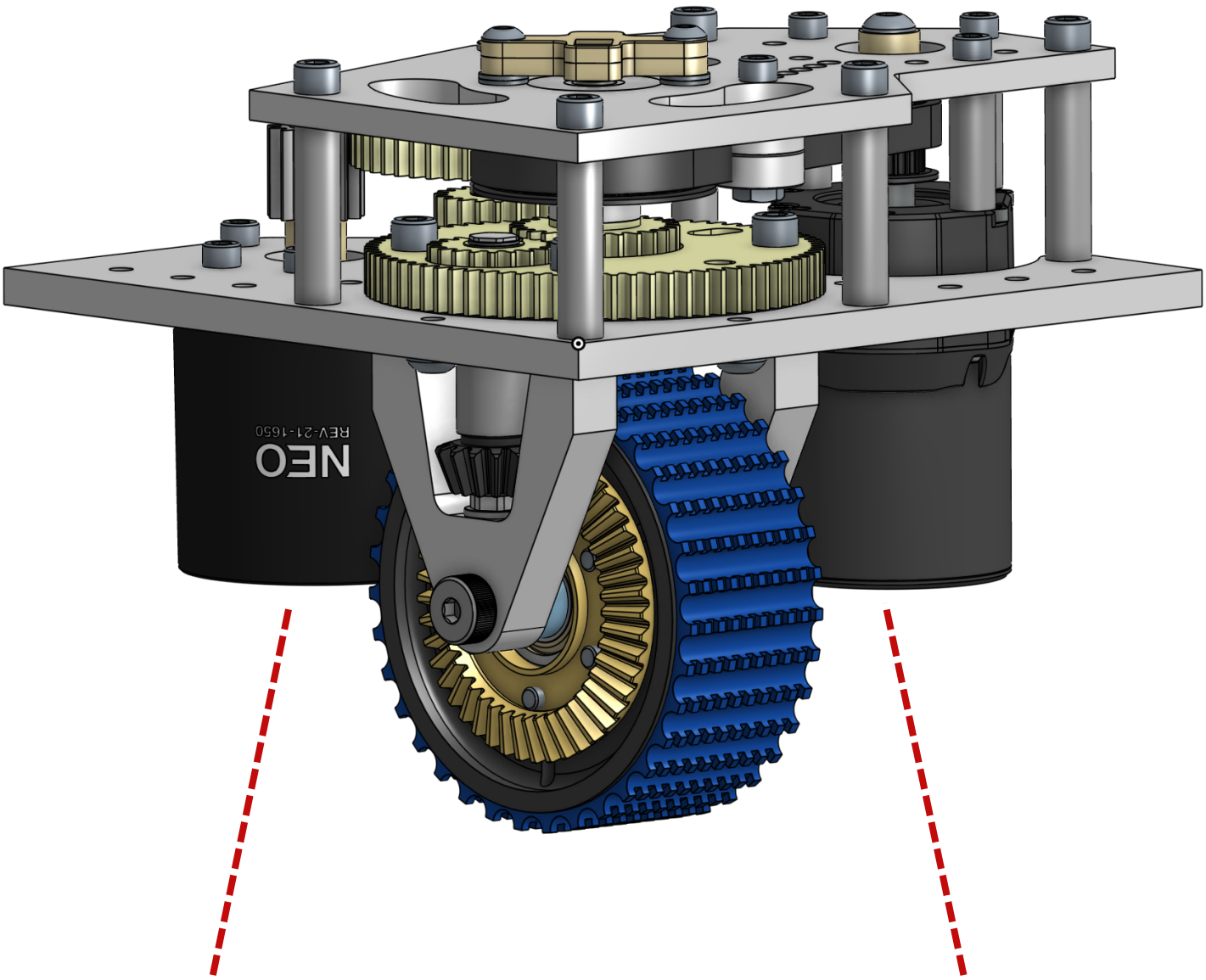



Custom designed and machined swerve

# Drivetrain

# INTRODUCING HERMES

## SWERVE MODULE OVERVIEW




**STEER**

- REV NEO v1.1
- 13.66 gear ratio
- 0.09° resolution (TTB abs mag encoder)

**DRIVE**

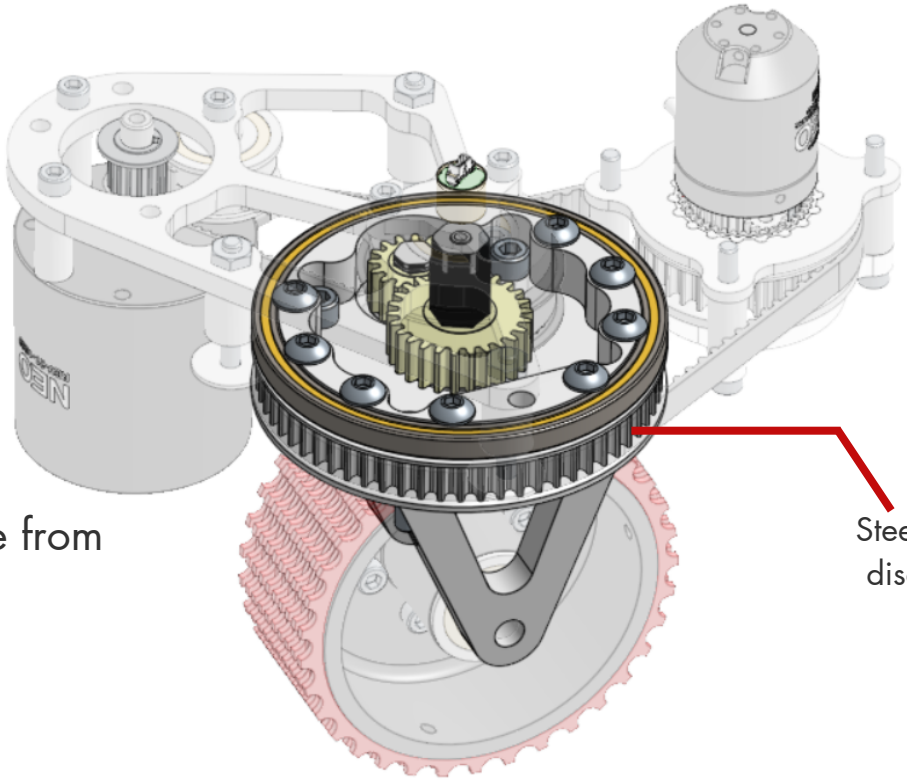
- REV NEO Vortex
- 6.92 gear ratio
- 17.1 ft/s top speed
- 23.6 ft/s^2 low end acceleration

Because we expected heavy defense and short sprints, our gear ratio of 6.92 prioritizes acceleration, while having a solid top speed for field-crossing.

# LEARNING FROM 2023

During the 2023 CHARGED UP season, we also used a custom swerve module design. It was lightweight and drove well, but came with drawbacks:

- Time-consuming and over-complicated to assemble
- Power transfer not robust
- 3D-printed steer pulleys snapped under fatigue
- Large structural issues related to unoptimized force paths.




GRT swerve from 2023

## INITIAL GOALS

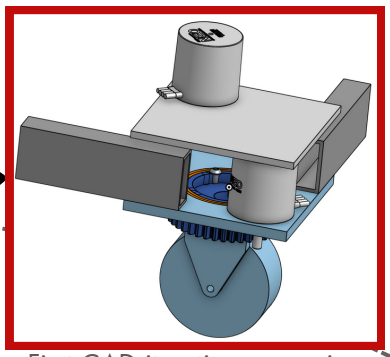
Given what we learned from 2023, we set goals for a 2024 custom swerve design:

- Expect and simulate the forces felt in matches
- Machine all parts in house
- Use an aluminum steer section for durability
- Streamline module assembly and maintenance
- Minimize package space

# ITERATIONS

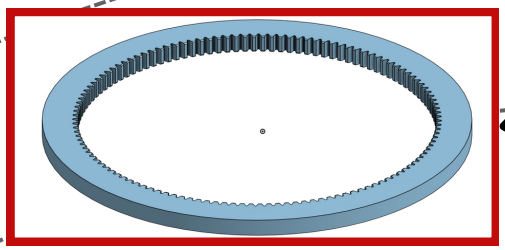
## THROUGH THE SOLAR SYSTEM

### HADES



First CAD iteration - experimented with motor placement.

### POSEIDON

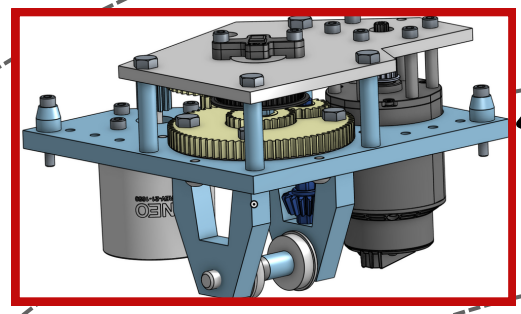


A large internal gear could be used for steer

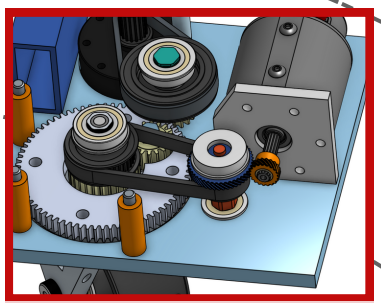
### OURANOS

More experimentation with packaging, steer section begins to take form.

### KRONOS



### ZEUS



Helical gears could be used for power transmission to package drive motor horizontally.

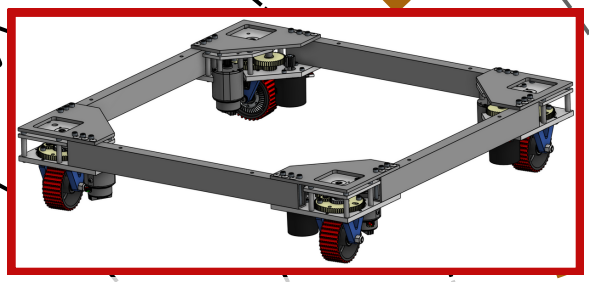
### DESIGN REVIEW

**(ASTEROID BELT)**

### ARES

The first fully viable module CAD, building off the Kronos design.

### GAEA

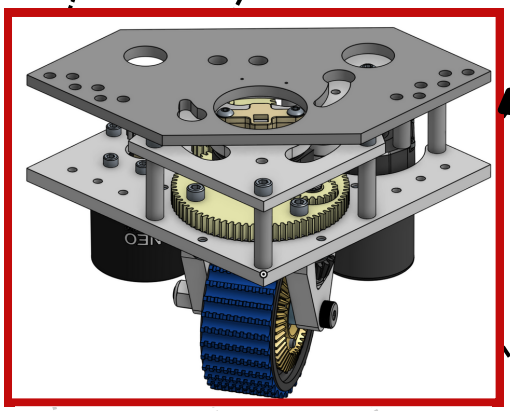


The first full chassis designed and machined for Robot 1.

### APHRODITE

The first module incorporating a NEO Vortex, and minor improvements from R1 testing.

### HERMES



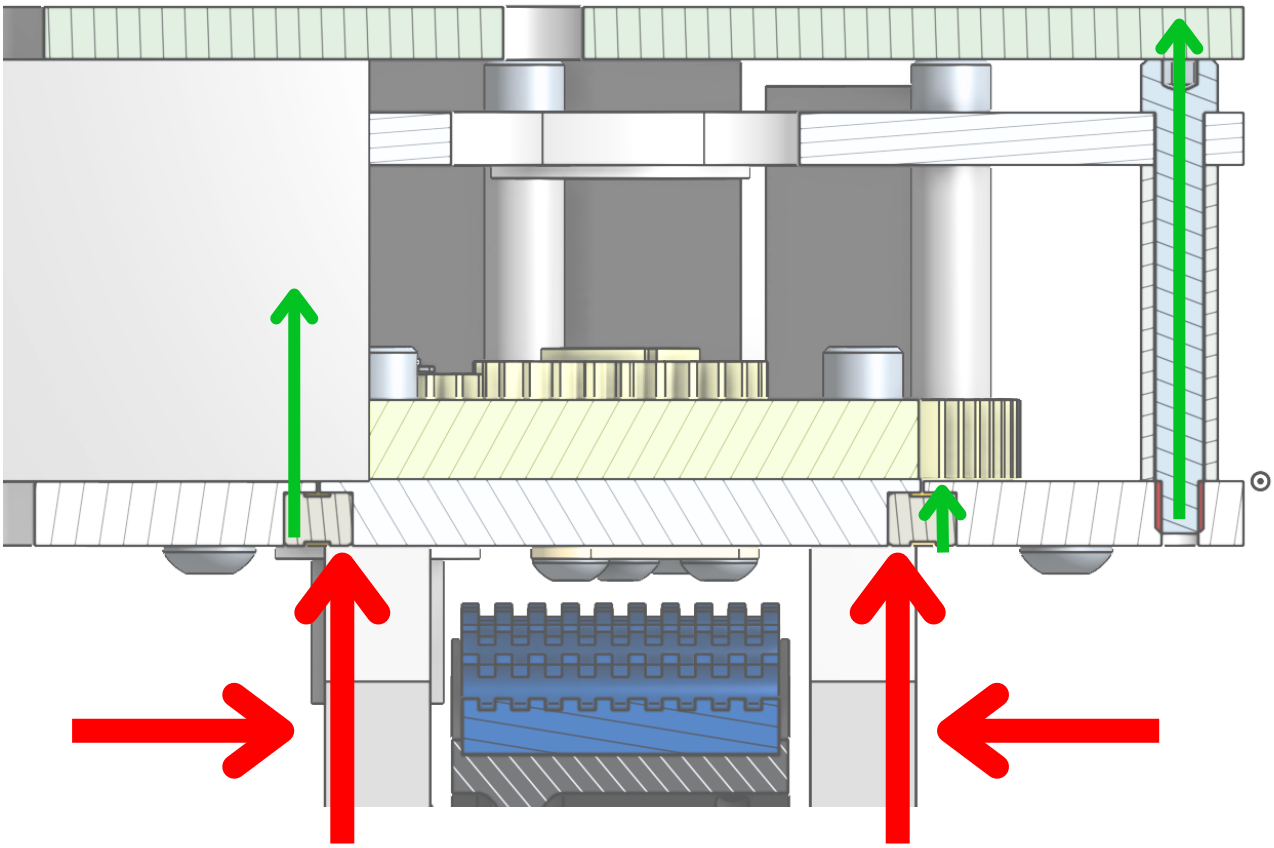
The final, competition-ready module, with weight-reducing pockets.

# FORCE PATHS

When designing our modules we considered the forces that the module would undergo. Our design choices reflect the force paths that would create the most robust modules.

## IMPACTS

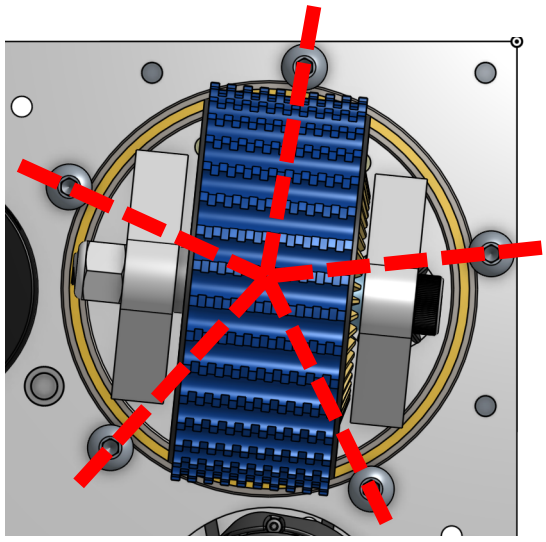
Whether dealing with vertical, lateral, or head-on impacts, it is important to design the module to react forces into the frame as directly as possible.



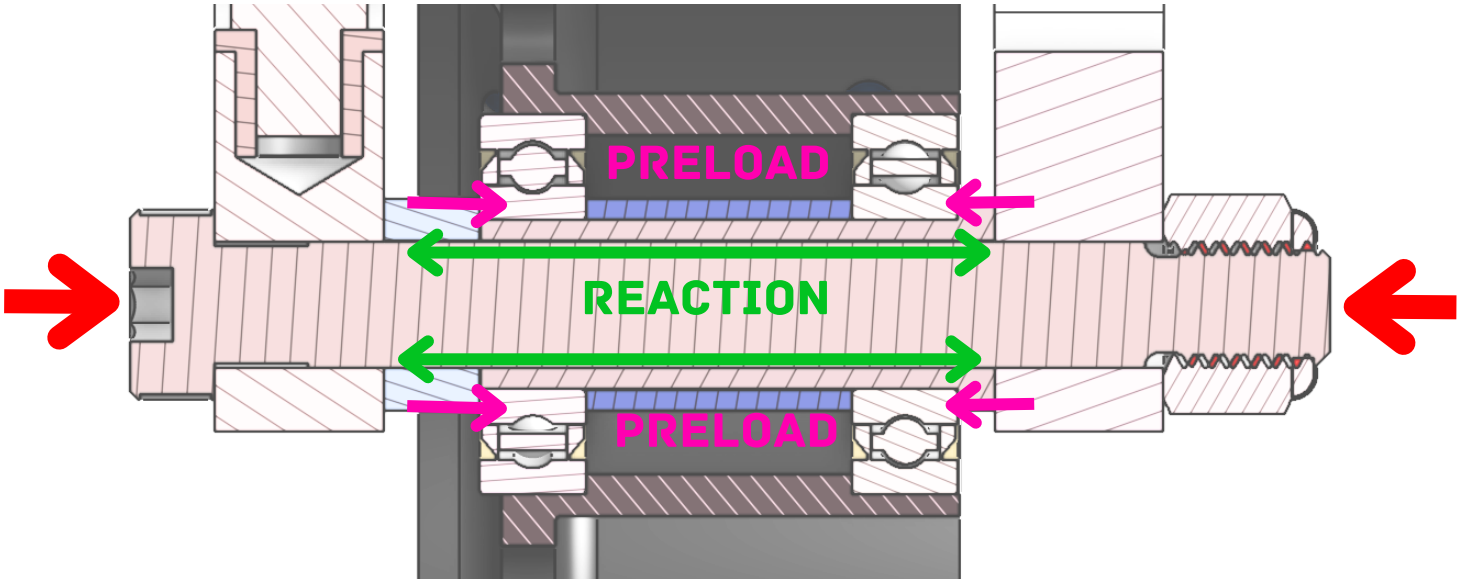
To minimize the number of force transfers, the wheel forks contact the ring bearing directly, which contacts the bottom gusset directly through the machined flange. From there, any force can travel directly into the frame box beam (left) or through the head of the screw contacting the top gusset (right), minimizing the cantilever on the corner of the module.

## BEARING RETENTION

Any side loads on the wheel produce not only an upwards force into the flange, but also a tensile force downward on the retaining screws. Using 5 screws to retain the ring bearing distributes the load while ensuring no symmetrical bending axis. However, under normal load the screws experience no tension, because the bearing is mostly pressed upward into the flange on the bottom gusset, due to the robot's weight.



## WHEEL CENTER BRACING




The wheel uses 2 ball bearings to spin on a dead axle carriage bolt. The clamping force of the bolt must be reacted through the wheel center, which is composed of 3 pieces: A spacer, a T-Pin, and a sleeve. These components allow for a force path directly between the forks, and a path through the inner races of the bearings. The path through the bearings allows for a preload, which improves the rigidity and lifetime of the wheel spindle. This axis was also designed with machining/assembly in mind.

# MACHINING

Instead of allowing our existing machining knowledge to limit our design, we strive to expand our capabilities to meet the challenge.

## GEAR HUB

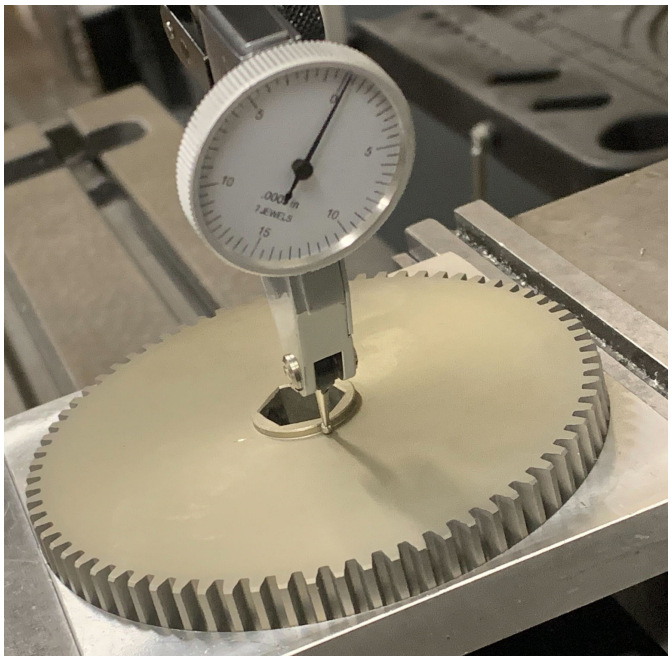
Searching the standard FRC vendors, it became clear that just milling a large gear was insufficient: The gear could only be 3/8" thick, and could not accommodate the necessary features. We needed a separate part to extend it downwards. We called this the gear hub.

First, we start with square stock, mill the hole features, then bandsaw the piece to a rough circular shape.

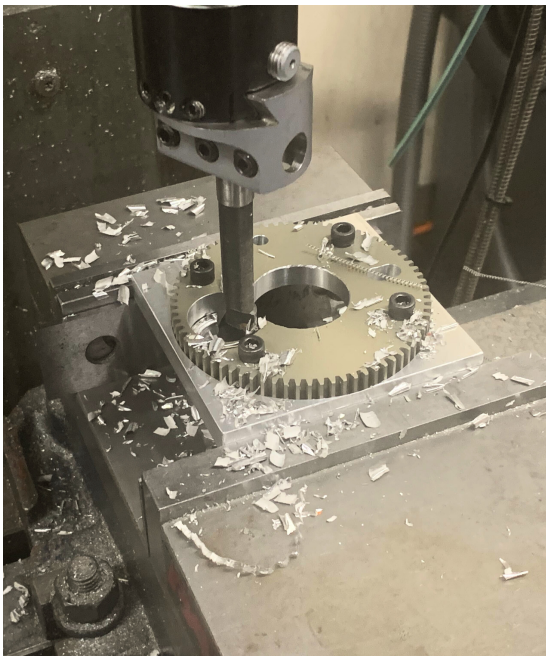
Next, we mount to our lathe fixture, locate to the center, and bore/turn all concentric features.

## BIG GEAR

Designing around our ideal drive gear ratio, we chose a 74t aluminum spur gear from VEX, since it fit our two drive spur gears.



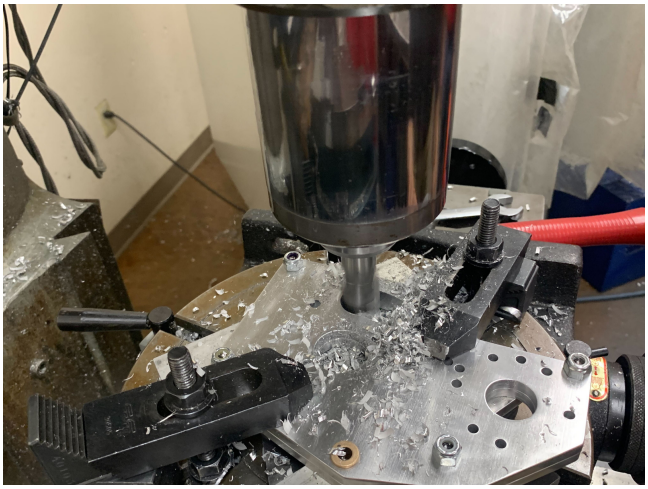
We press-fit the gear into a custom mill fixture, dial indicate to the raised lip of the gear.



We drill and bore all features, measured from the indicated center, clamping and bolting the gear down.

## SPOTLIGHT: ROTARY TABLE

The rotary table allows for curved slots on the mill. We used the rotary table to make stylized weight-reducing pockets, but we hope to utilize this powerful tool to make complex precision parts in future years.

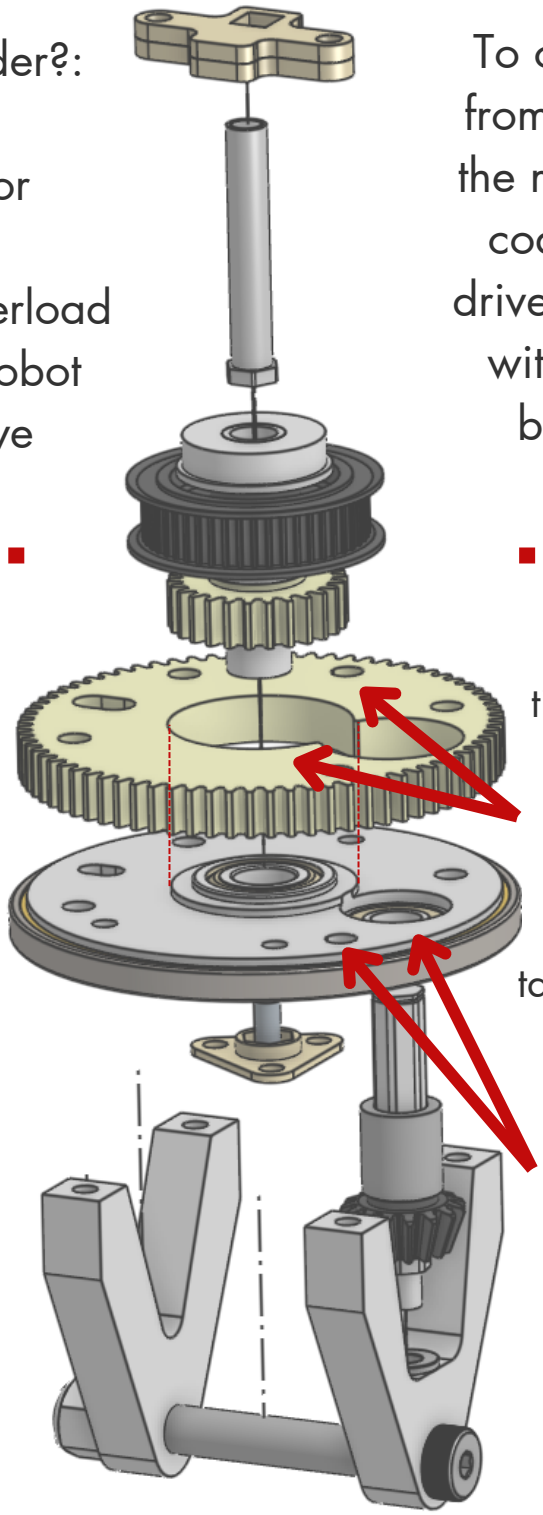


# The Steer Section

## ENCODER

Why use a Thrifty encoder?:

1. Magnetic encoder for ease of disassembly
2. Wiring does not overload CAN chain on the robot
3. Relatively inexpensive



To create a 1:1 reading from the steer, we housed the magnet in a rod that is coaxial with the central drive axle. This rod mounts with a 3D-printed insert below the gear hub.

## LOCATING PARTS

For precision, we used a 4-way 2-way datum structure in multiple locations on the steer section .

By having a tight fitting circular locating feature (4-way) in tandem with a small slot feature (2-way), we ensure that we eliminate all 6 degrees of freedom with zero play and without over-constraining any parts.

A bore in the big gear fits tightly around a pilot feature (4-way) in the gear hub. the rotational freedom is restricted by a short slot (2-way) in the big gear over a tapped hole in the gear hub.

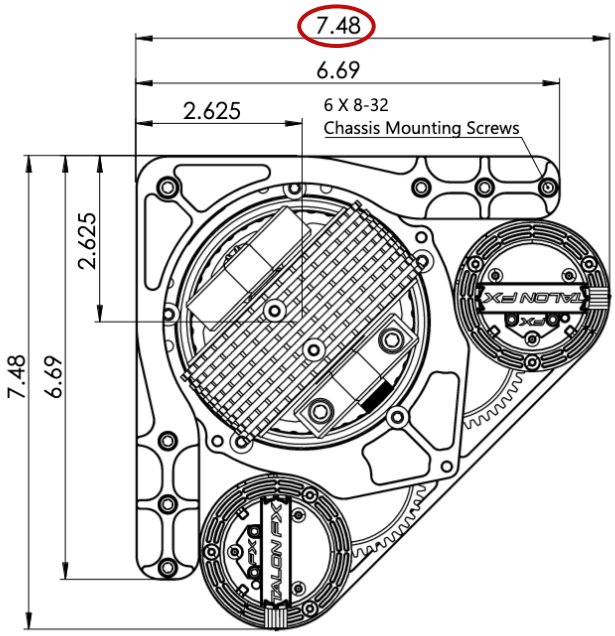
The wheel forks are located by the bevel gear axle and bearing in the gear hub (4-way) along with a short slot in the gear hub (2-way).

Additional screws are used to hold the pieces together, but they do not over-constrain the parts because they clamp through loose clearance holes.

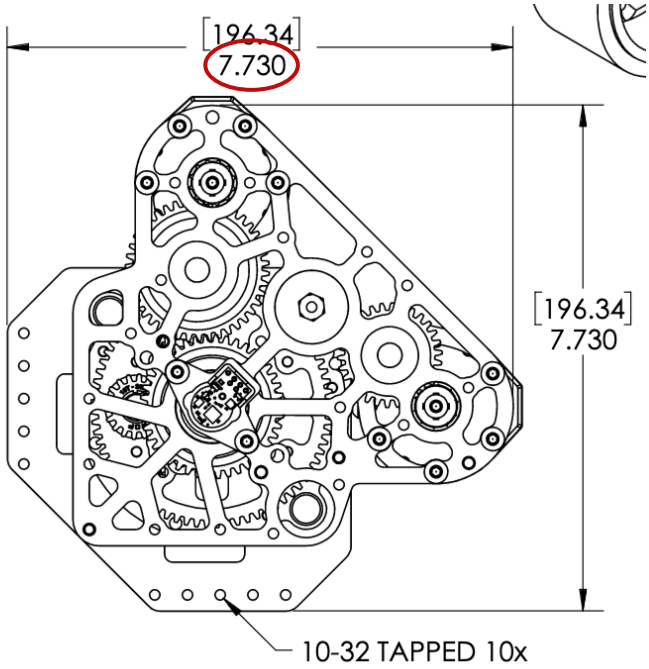
# Package Space

One important goal for Hermes was compactness to give mechanisms as much space as possible. We wanted to mount the swerve module beneath the frame beams, sacrificing ground clearance, but allowing for an intake that could take up the majority of the width of the robot. Additionally, we saved on horizontal package space by shifting the wheel axis into the corner of the frame.

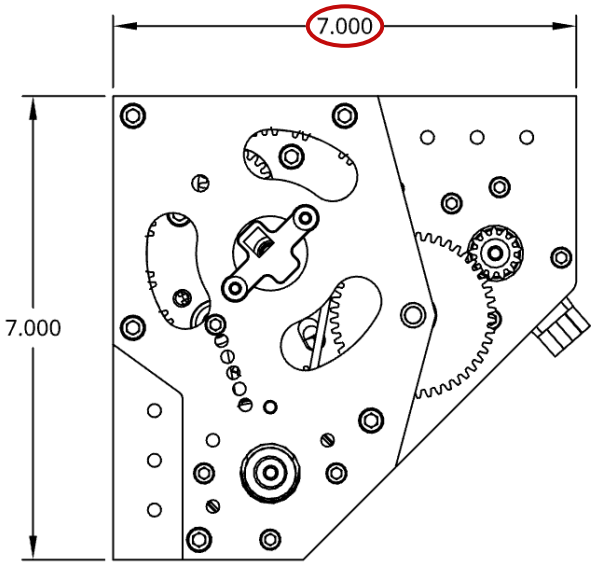
### SDS MK4I




### WCP SWERVE XS FLIPPED




### GRT HERMES 2024



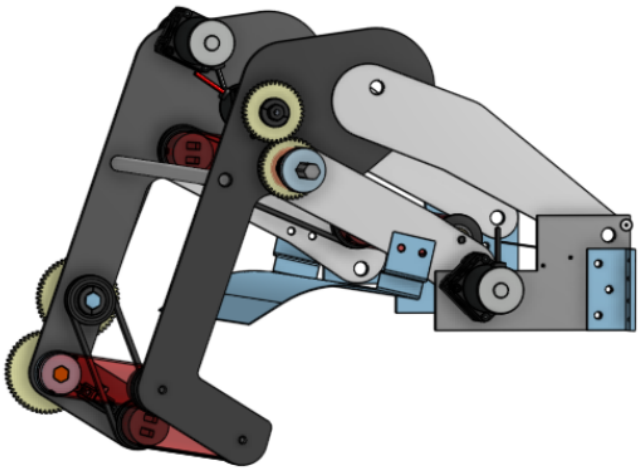

Our swerve design saves on footprint size over popular, off-the-shelf, low-profile swerve options.

# Mechanisms

# Intake

## PROTOTYPING

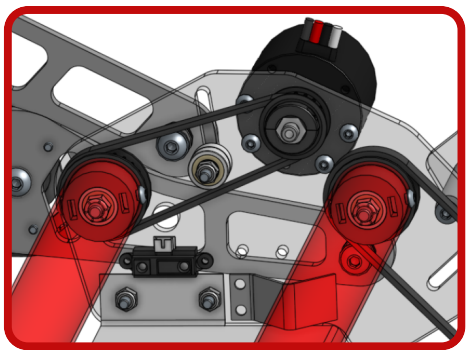
- Dual opposing vertical flywheels powered by hand drills and CIM motors worked well
- Notes often got stuck on the edges
- Also very successful at outtaking into the Amp, which guided our design



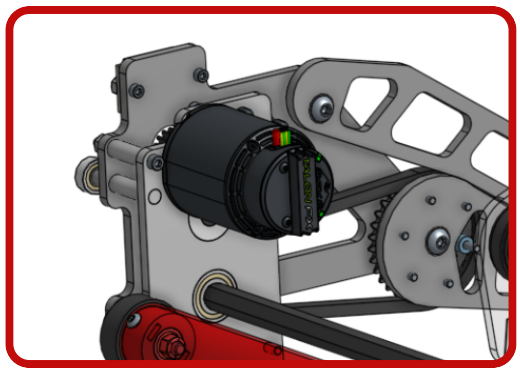
## ROBOT I

- Used large gears to reverse direction
- Used several BAG motors for powering intake rollers
- **4-bar linkage** for better stowing characteristics and a better intaking angle
- Different components on different linkages led to it binding on itself and **requiring extensive modification** to work correctly

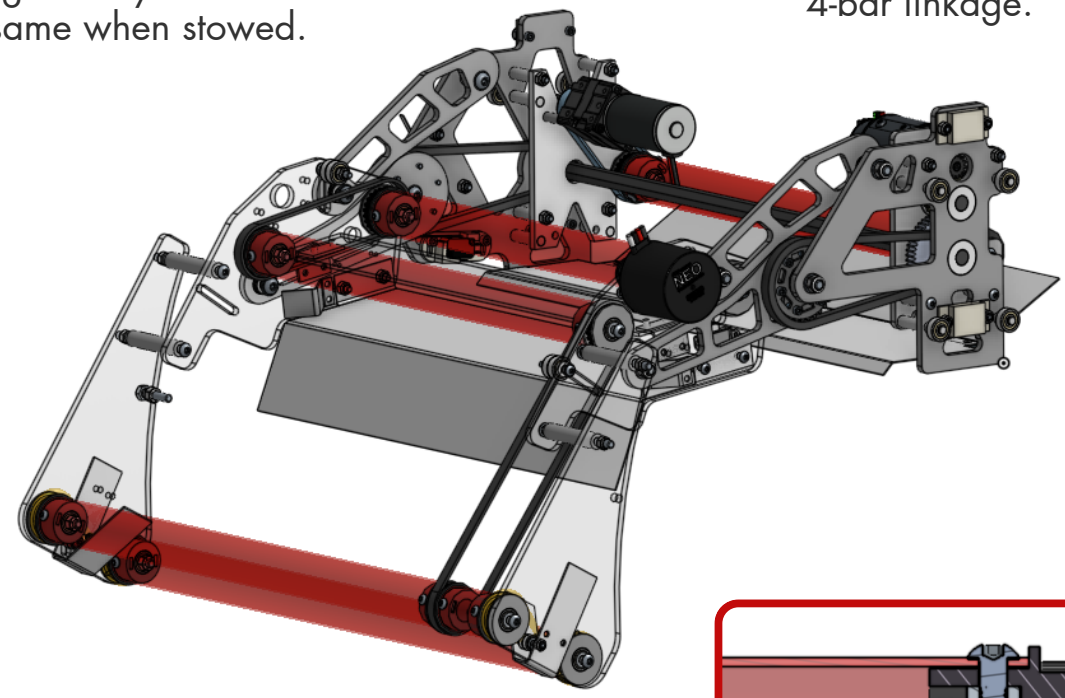
# FINAL PRODUCT

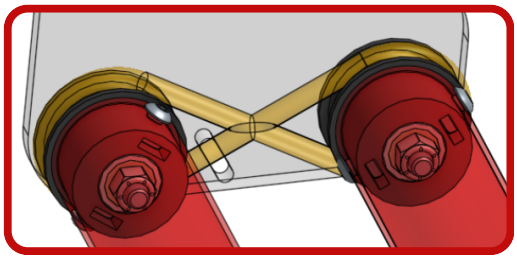


All channeling functions on polycarbonate panels to keep the relative geometry of the intake the same when stowed.

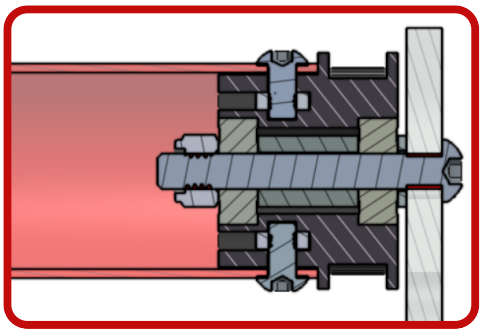


**Falcon 500** motor powers chain & sprockets on both sides to pivot the 4-bar linkage.





**Polycord infinity belts** for simple and lightweight direction reversal



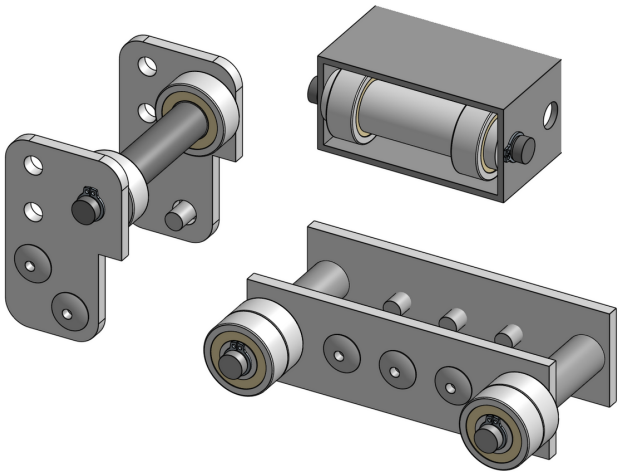
Our **rollers** consist of aluminum tubes that are attached to custom **3D printed pulleys**.

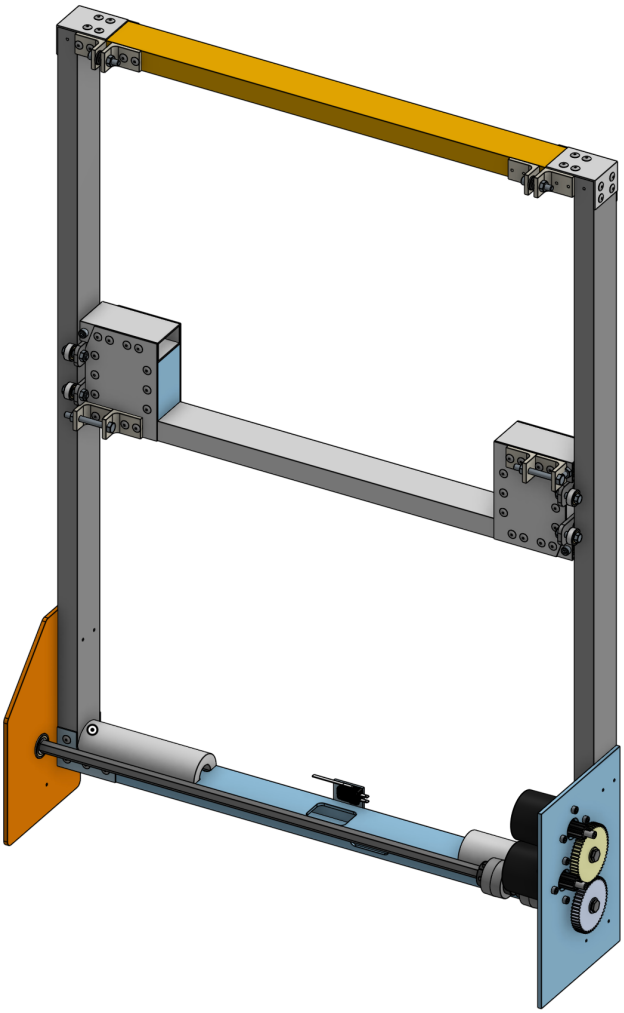
- Uses polycarbonate panels for a high strength, compliant, light weight material
- Stows fully within bumpers and reaches **11.5" when fully extended**
- Intake is also the **second stage of the elevator**, with an integration roller to feed the shooter when the intake is lowered

# ELEVATOR

## DELIVERABLES

- Implement a reliable and maintainable design, **developing upon past years**
- Keep **tighter tolerances** and **streamline assembly**
- Use new techniques to expand the horizons of the team and **set an example**
- Fit **under the stage** and **extend** up to four feet





## ROBOT I

- **One stage** to minimize parts and machining time
- Retained the **full desired range of motion**
- Utilized two separate strings to allow **Note passage** through the elevator
- Experimented with one down and up-string, which caused the carriage to derail to one side
- **A 1:4 gear ratio** compensates for the weight of carriage and intake while keeping high speed

# FINAL PRODUCT

- Upgraded to **two-stage design**, fitting below the stage when retracted and extending fully to Amp height
- The gearbox was moved into the center of the robot for compactness, safety, and better meshing of gears
- Added hard stops, cameras, and plenty of pocket holes for final integration
- Bearing pins press fit into precision drilled holes replaced bolts on which bearings were mounted, **removing play** as the bolt shifted in the clearance hole and allowing much **tighter tolerances**
- String tensioning was redesigned, using bolts to clamp plates onto the string rather than relying on tying knots.
- **Plastic linear bushings** for the second stage

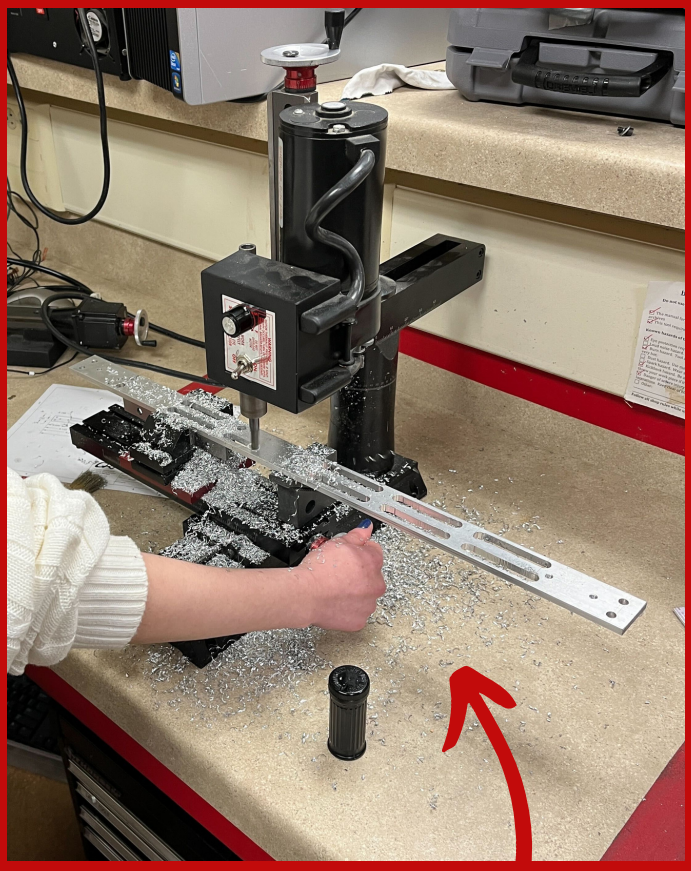
## SPOTLIGHT: PLASTIC LINEAR SLIDERS

When considering possible elevator designs, we analyzed past years' elevators that used plastic sliders. The plastic served as an alternative to bearings because they have naturally low friction and don't need an axis to rotate about. During prototyping, the plastic proved to work without issue. In the end, the plastic blocks and sheets provide a very quiet and reliable linear motion, need little to no precision for assembly, and have no moving parts.

## SPOTLIGHT: SHERLINE MINI MACHINES

While machining our competition robot, one of our two Series 1 manual Bridgeport milling machines broke. A few team members dusted off a neglected miniature Sherline 2000 mill and lathe and got to work.

The machines were so seldom used that nobody had any experience with them, so a small sub-group was created to re-learn and master the mini-machines. The mini mill has no auto-feed nor a digital readout, so the machinist has to count the number of revolutions they made or read the 20th of an inch markings. All in all, there are 15 parts on the robot made by the Sherline mill, as well as a few more on the Sherline mini lathe.

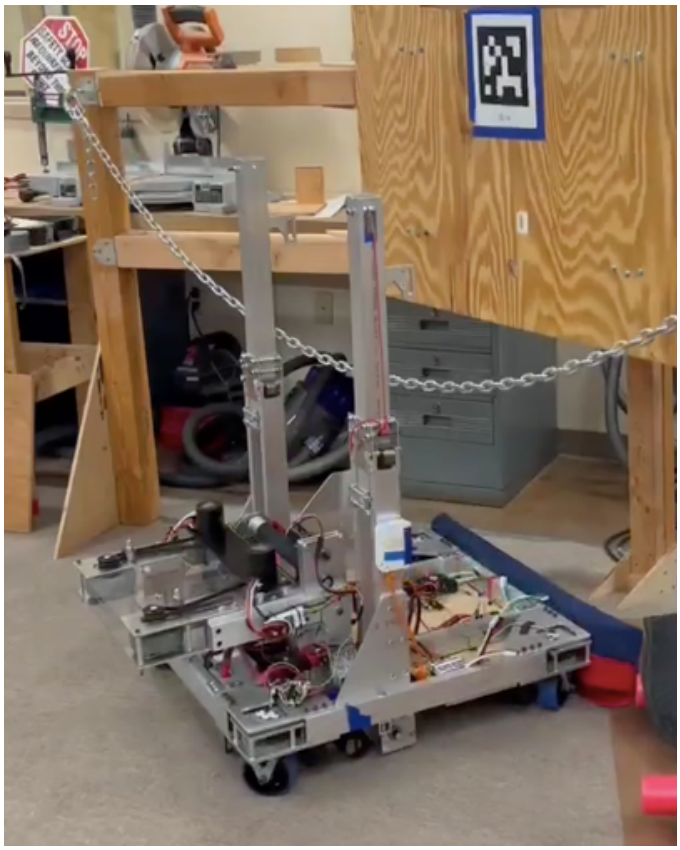


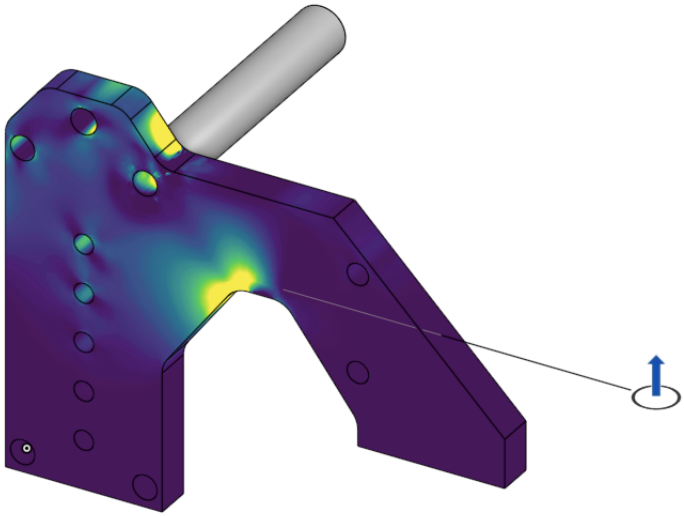
**This elevator part is being pocketed on the Sherline mill to reduce weight**

# CLIMB

## DELIVERABLES

- Capable of **raising bumpers to the chain** in order to score in the Trap
- Capable of fitting **underneath the stage**
- Improve and expand on previous seasons' issues with designing climb mechanisms



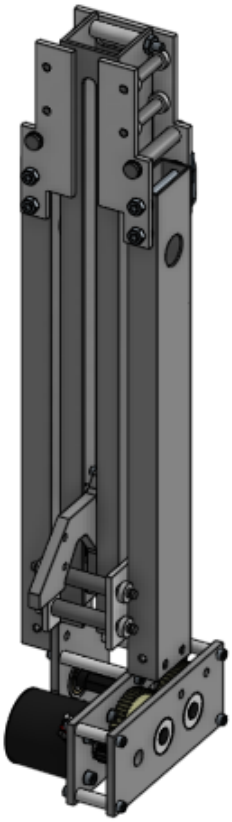


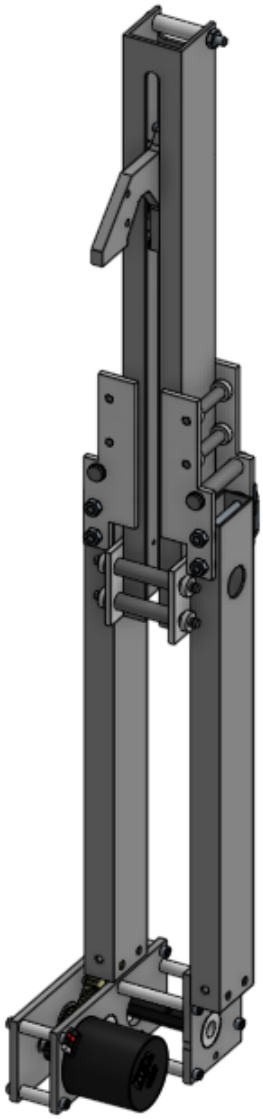
## PROTOTYPING

- Difficult to prototype climb, so we mainly tested hook designs and experimented with **stress analysis**
- **Grappling hooks** were tested, but there was difficulty with lifting the robot and extension limits
- One-point climb prototypes helped with earning **Harmony** points, but were worse for scoring in the Trap

## FINAL PRODUCT

- **Three-stage elevator** mechanism to stay below the stage when lowered but above the chain when about to climb
- One arm on each side of the robot, each identical
- The second stage is a hook with a carriage that slides along the inside of the first stage in order to raise the bumpers to the chain
- The hook and second stage move up using a constant-force spring and down using a winch with a two-stage gearbox powered by a NEO
- Clamping plates to tension pulley strings
- Gearbox with a 1:10 gear ratio

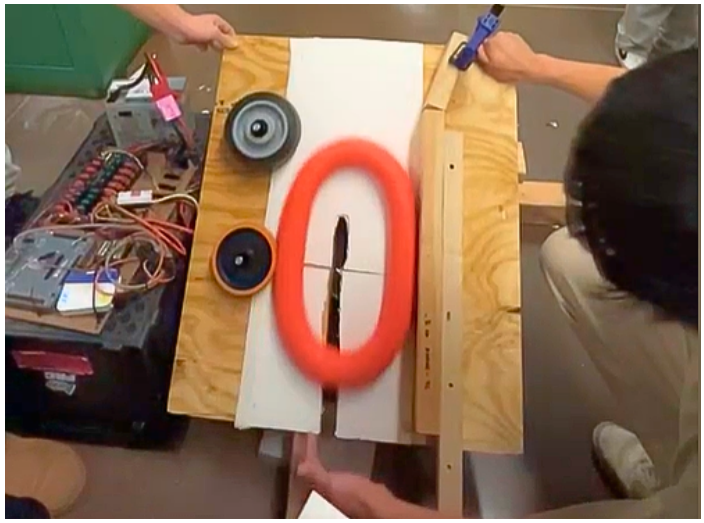


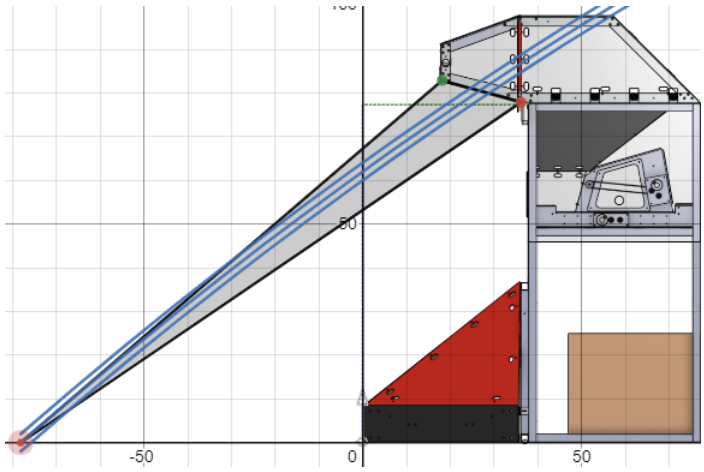


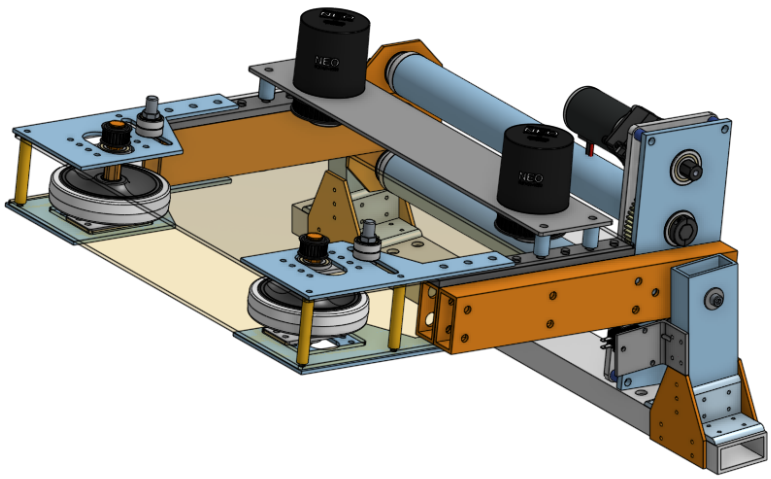
# SHOOTER

## PROTOTYPING

- Created a **Desmos program** to calculate the ideal wheel diameter and gear ratio for a shooter
- Dual one-sided flywheels proved tricky to aim
- Dual vertical rollers were originally seen as too weak because of the **insufficient tangential speed** of the small wheels used in testing
- Dual horizontally spinning flywheels were chosen for robot I because of their **repeatability** in testing and the ability to **adjust aim** with code





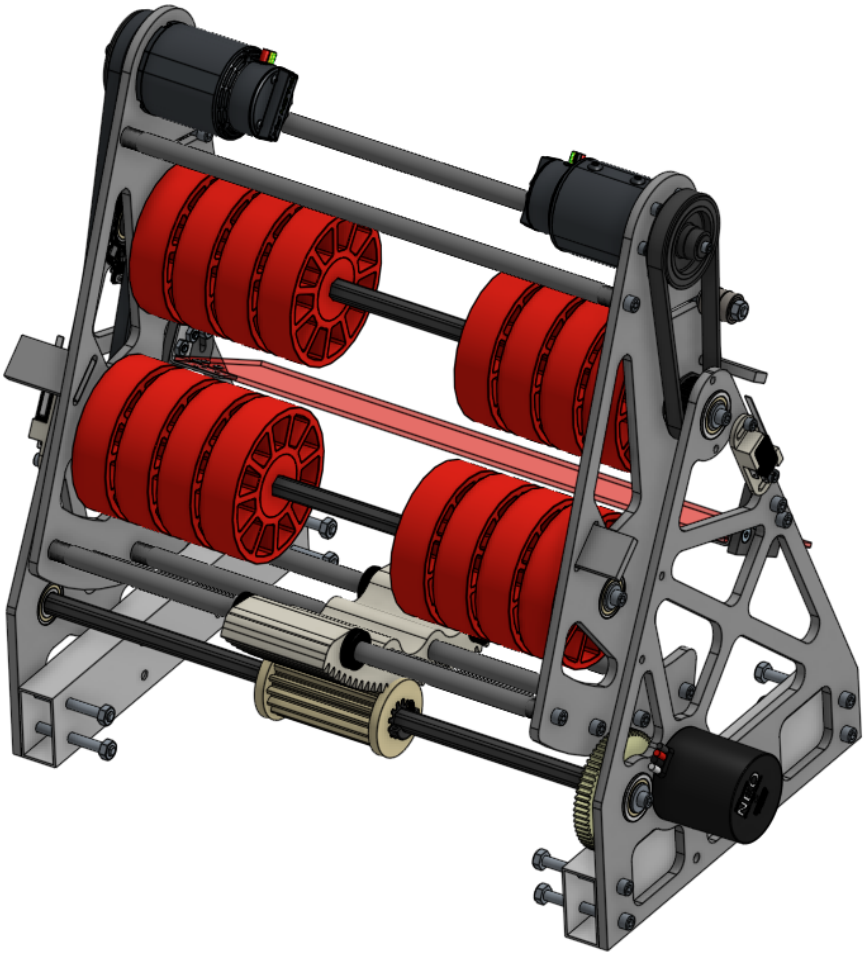



## ROBOT I

- From our testing, dual opposing horizontal flywheels seemed like the best option
- **Dual NEO** flywheel motors
- Built-in intaking mechanism with **automatic Note detection**
- Adjustable compression flywheels
- Chain-and-sprocket pivot with **in-line tensioner**

# FINAL PRODUCT

- Later testing revealed that **vertically spinning flywheels** performed much better
- Uses 4" diameter **35A durometer** compliant wheels because of their grip
- **Dual Falcon 500 flywheel motors** for maximum power and independent top/bottom speed control
- NEO Pivot with a 3D-printed **sector gear and pinion** system for no re-tensioning and **minimal backlash**
- Pivots around the top flywheel axle for design simplicity and reliable Note intaking
- Welded hard stops for **maximum strength**
- The Falcon 500 motors are geared up 3:2 for even higher shooting speeds - capable of launching the Note **in excess of 100mph**



## SPOTLIGHT: WELDING

This season marked the return of the welding subgroup, which temporarily ceased operation due to COVID-19. All of the existing welding subgroup members had graduated, but thanks to our great welding mentor, we learned the chemistry behind tig welding and made great progress toward redeveloping the welding subgroup. Adjusting our welder's spark gaps, coolant lines, and argon gas, we repaired the broken welder and learned how to apply the powers of welding to our robot. You will find the chassis gussets and climb gussets welded on this year's robot.

## SPOTLIGHT: POWERTAPPING

Powertapping is the process of making tapped holes using a powered machine. During the design phase of our custom swerve modules, we decided to set upon learning how to use our milling machines and CNC to powertap.

Traditionally, we have manually used taps designed for powered work, meaning that we already had the tools for powertapping. After proper research and preparation, keeping safety as priority number one, we began our first tests. We optimized spindle speeds, feed rates, and workholding techniques thoroughly. To make sure future years will be able to learn as well, we wrote a curriculum that goes in depth on every step of the process.

**A gear hub being powertapped using a spiral point tap in a key chuck**

# Controls

# Code Structure

This year we focused on designing a fully command-based robot. Almost every operation on the robot is run through WPILib commands, which allow for better control of each specific mechanism without using vast computing power.

Last year we used many state machines and had subsystems run all of their own actions within the period method, a function that is called every 20ms. Since we had multiple periodics doing different things during every loop, we ran into the problem of loop overruns. Loop overruns occur when the next loop is trying to start, but the current loop is still running. This cuts off parts of your code, which can have terrible consequences.

Additionally, each of our subsystems needed complex logic to use the state machines that we had. This logic was illegible and difficult to change, causing many hours of wasted time and preventing us from making quick changes during competitions.

Each mechanism is independent and has several tasks they can be set to do in the form of commands. We use sequences of these commands to automate full robot movements, which removes work from the drivers.

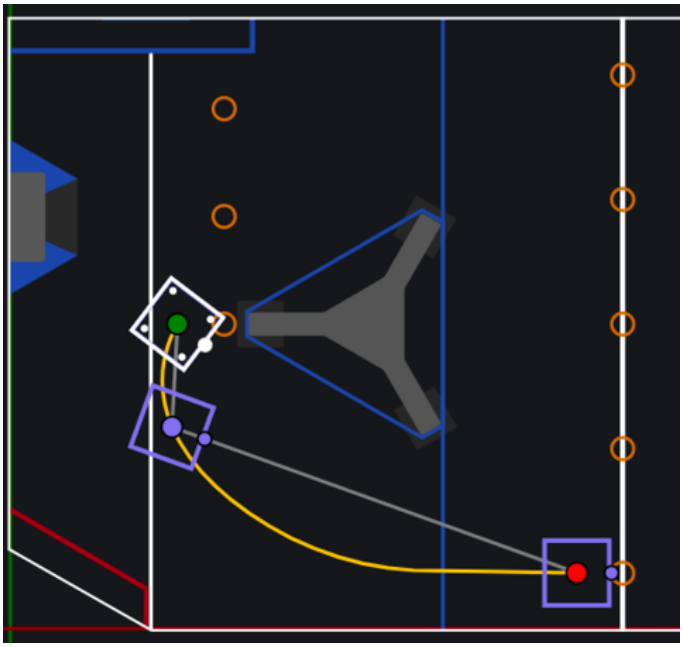
*All code is publicly available a github.com/grt192/GRT2024.*

# AUTONOMOUS

The autonomous structure uses an external library called Choreo for trajectory generation of predetermined paths. We chose Choreo because it takes into account all of the robot specs, including weight, max acceleration, and motor types to account for momentum and other physical influences during path following.

All general auton functions are located in the parent class, BaseAutonSequence, which abstracts the Choreo swerve command into a followPath that only takes in a pre-generated trajectory file. Alongside this are command sequences goShoot and goIntake, allowing paths to be easily strung together in a series of these functions.

The resulting full auton sequences are the child classes of BaseAutonSequence, and are selected through an AutonChooser in robotContainer.



*Here is an example auton path*

# Vision

April tags were critical to our autonomous routines in the 2023 season. However, our performance was limited by our subpar hardware and software setup, which introduced high amounts of latency and noise into our position estimation.

From April to December 2023, we worked to use our existing Nvidia Jetson GPUs to run highly optimized Apriltag pipelines with the goal of integrating them into control systems for both the autonomous and teleoperated periods.

After learning how to work with GPUs, Linux, and ROS (Robot Operating System), we were able to get an Apriltag pipeline running on our coprocessors. Due to its inefficient processor utilization, its performance was at best comparable to the current FRC standard. However, as it was not bound by hardware or software limitations, and used devices with more industrial support, the pipeline opened doors to further leaps in autonomy.

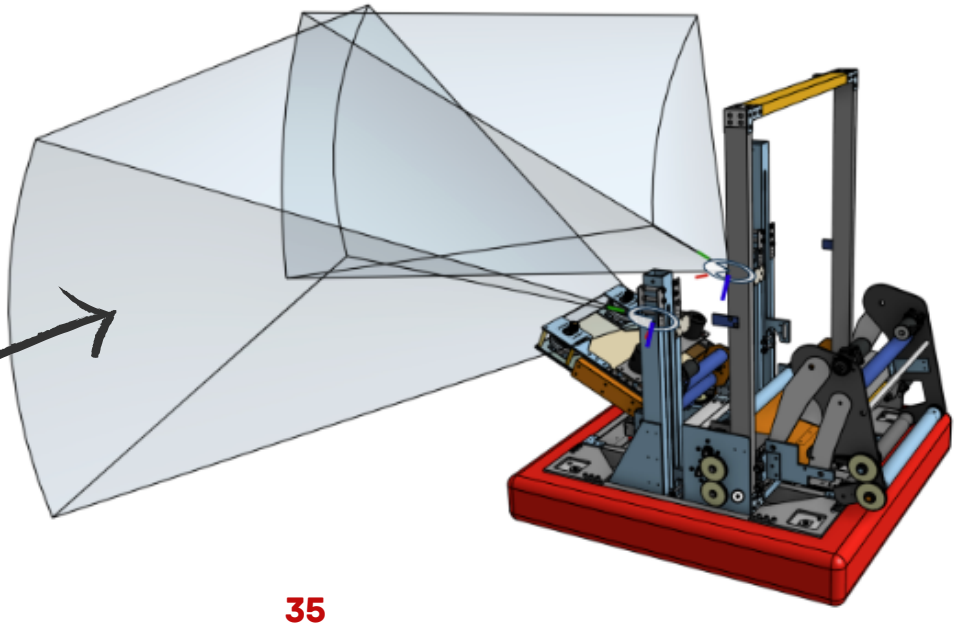
To make it worthwhile, however, we would need to use our processors' GPUs, instead of just their CPU. Using GPUs would drastically optimize image processing, reducing the computational expense associated with Apriltags. Bringing GPU computing to FIRST would enable us to set our sights on more advanced autonomy, setting a higher bar for controls technology in the competition.

However, our efforts to utilize GPU computing were not as successful. In our development process, we began to run into compatibility issues with Nvidia and ROS software development kits, as both organizations had officially ended support for the operating systems used by our GPUs.

GRT has been working with Nvidia to regain access to the versions we need to make GPU Apriltags a possibility in FRC. However, with our time limitations, we made the decision to continue with the FRC standard Apriltag system for another year, using multiple Raspberry Pi coprocessors to run our Apriltag and Note Detection vision pipelines.

The 8 months we worked on this vision system aren't going to come to fruition in the 2024 season. However, we've laid the groundwork for our future seasons to use industrial-grade technology to build powerful autonomous routines that bring us closer to our ultimate goal of full robot autonomy during matches.

Moreover, the project helped students on GRT's controls subgroup develop advanced skills in software development and computer vision. Through building systems for use in FRC, we're developing skills that are highly relevant in real world engineering.



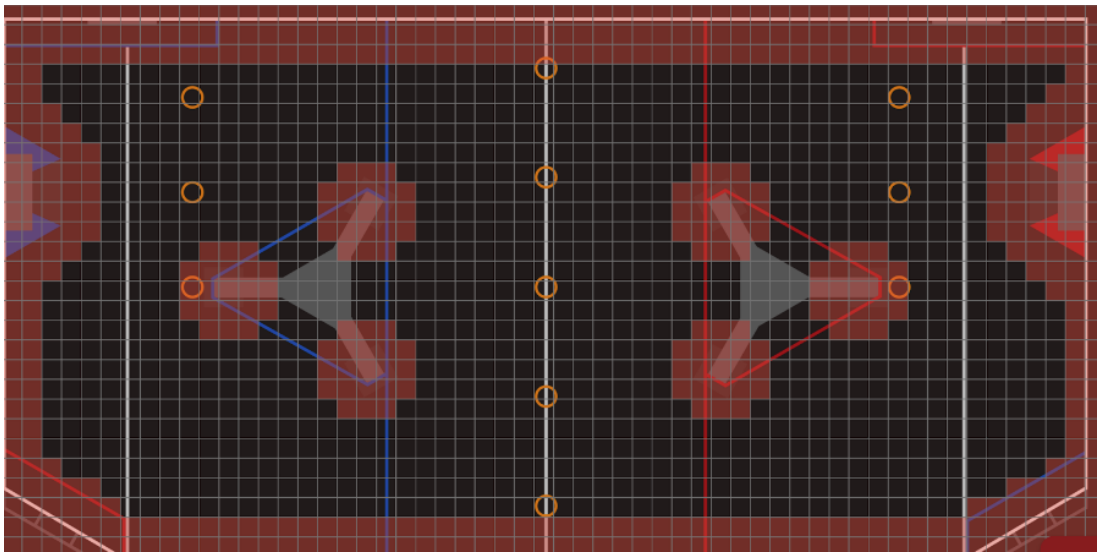
Calculating camera placement to maximize vision system FOV

# Auto-Align

In 2023 Charged Up, GRT saw the opportunity to incorporate elements of our autonomous routines into the teleoperated section, allowing drivers to automate precision tasks. This took the form of a custom interface that can automatically align the robot to the source, speaker, or amp.

For 2024, we expanded the scope of our Auto-Align system to make it an integral part of our scoring cycle. Instead of only close range autonomous control, we are now able to autonomously plot and follow paths between any two points on the field. Using an AD-Star path planning algorithm, we're able to avoid field elements such as the Stages. When game pieces or other robots enter the path of the robot, our drivers simply have to release a trigger on their controller to take over control and make corrective maneuvers.

With these capabilities, our drivers aren't always zoomed in on our robot; they are able to see what's going on around the field (e.g. bottlenecks, defended positions, etc.) and make high-level decisions that can improve our performance.



*One of our robot navigation grids, distinguishing traversable areas from obstacles*

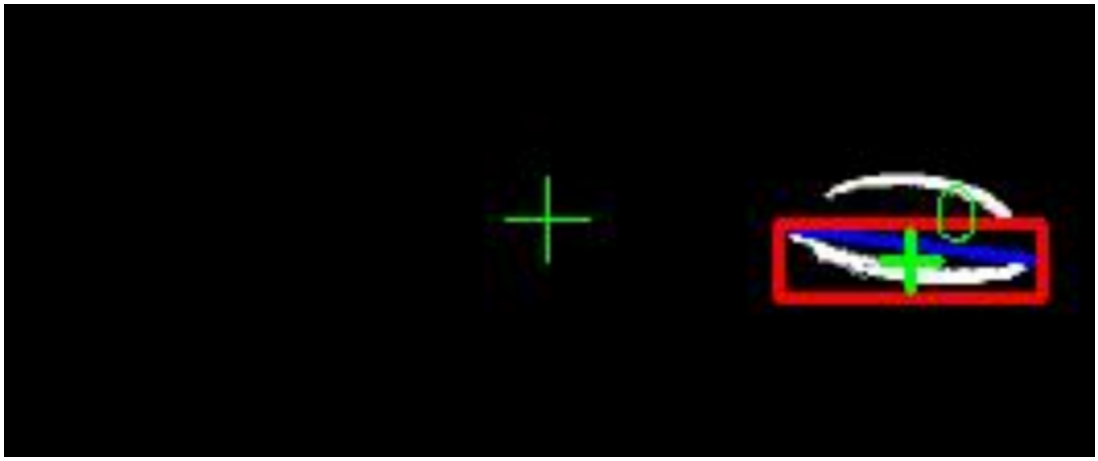
# Note Align

In 2023, we found that it was often difficult for our driver to align to game pieces to intake them, especially when intaking across the field and with poor visibility.

For 2024, we decided to implement automatic Note alignment to solve this issue. A forward-facing camera feeds into a Raspberry Pi running PhotonVision, in which we have a custom-configured colored shape detection pipeline to consistently detect Notes' presence and position within the camera frame.

Our code is then able to access this detection information and gives the driver the option to automatically intake it by first using a rotation-based PID controller to align the note with the camera frame center and then initiating an intake command sequence to move the robot forward until the note is inside the robot.

Note align allows for faster, more consistent cycles, as it cuts down on time spent aligning to Notes and allows our drivers to focus on higher-level game elements.



Detecting a note at an angle

# User Interface

For the 2024 season, we built custom driver interface that can handle the selection of routes, displays telemetry and camera feeds, enables the driver to change robot parameters (such as disabling computer vision if it goes awry mid-match), and shows a map of the field with robot positions.

This year's driver UI is completely redesigned in Python using the PySide6 library. The UI consists of 4 main parts:

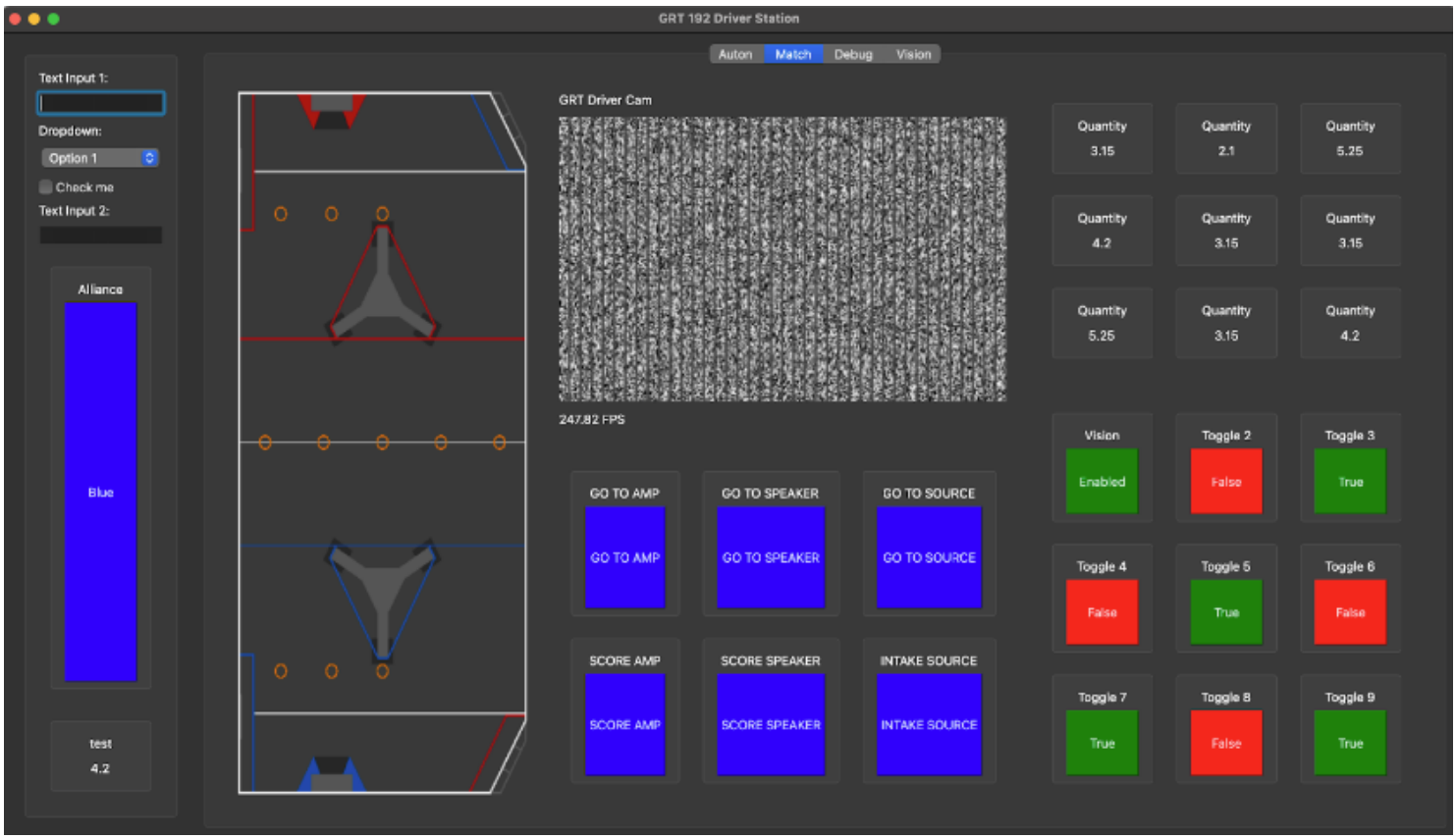
- Control Panel + Control Buttons
- Map Display
- Video Stream
- Info Display

**Control Panel:** The control panel switches the color and the direction of the map according to the alliance color. It will flip the map to fit the driver's point of view from the driver station. It also provides an option to lock itself in case of mistouch during the matches. The buttons are custom-built classes with labels and push buttons. They send changes to the networktable once clicked and also change color based on the status of the action.

**Map Display:** The map display displays both the field and the robot so that the driver can tell where the robot is at all time. The location data is captured through listening to the networktables from the robot.

**Video Stream:** The video stream displays the cameras' video streams on the driver UI. The streams are captured from mjpg servers through OpenCV and then converted to pixmaps to be displayed. The video stream can be switchable, either manually or automatically through buttons on the UI or changes of values in the networktable (e.g. automatically switch to vision camera when trying to auto intake notes).

**Info Display:** We built 2 types of fundamental info display widgets: boolean displays and number displays. They both get the values from networktables. The boolean displays change their color based on their values to make it easier to visualize. The number displays also fluctuates the number once new value is received through networktables.




*Early version of our 2024 Driver User Interface*

# Mechanism Code

## Intake

The intake needs to be able to take in a note and either feed it to the shooter, or outtake it into the amp. A distance sensor is used to detect if a note is being stored. For intaking, the rollers stop moving once a note is detected. For feeding to the shooter and outtaking, a timer is added so that the rollers run a little longer after the note is no longer detected to ensure that it fully leaves the intake. Additionally, the intake has a pivot that allows it to extend when intaking or scoring in the amp, and retract when the robot is moving. When it needs to be fully extended or retracted, the pivot motor runs until the intake hits a limit switch. To score in the amp, the intake isn't fully extended, so the motor runs until an encoder returns a set position.

## Elevator

The elevator has 2 modes: manual mode and auto mode. The manual mode is developed for Trap scoring and as a safety net in case something goes wrong. It allows the driver to move the elevator directly through the controller to any position they want.

In auto mode, the elevator moves between the 4 predefined states: ground, intake (just above ground to make the intake rollers not touch the ground), AMP (amplifier), and trap, which are defined by an enum class. For movements, the elevator uses encoders to measure the current distance (height) and uses PIDs to move to different positions.

One of the problems was that the elevator moves against gravity on its way up and in the same direction with gravity on its way down, which makes it hit its base really hard. To avoid that, we increased the derivative part in PID to account for errors and dampen the speed by applying a counteracting force to slow down the movement. Also, we added an arbitrary feedforward to make sure the elevator doesn't get pulled down by gravity while it's up in the air.

## SHOOTER

The code behind the shooter mechanism involves two separate subsystems (classes). One class, the pivot subsystem, controls functions and variables for the shooter's pivot.and includes soft and hard limits and tuned the PID for the pivot motor.

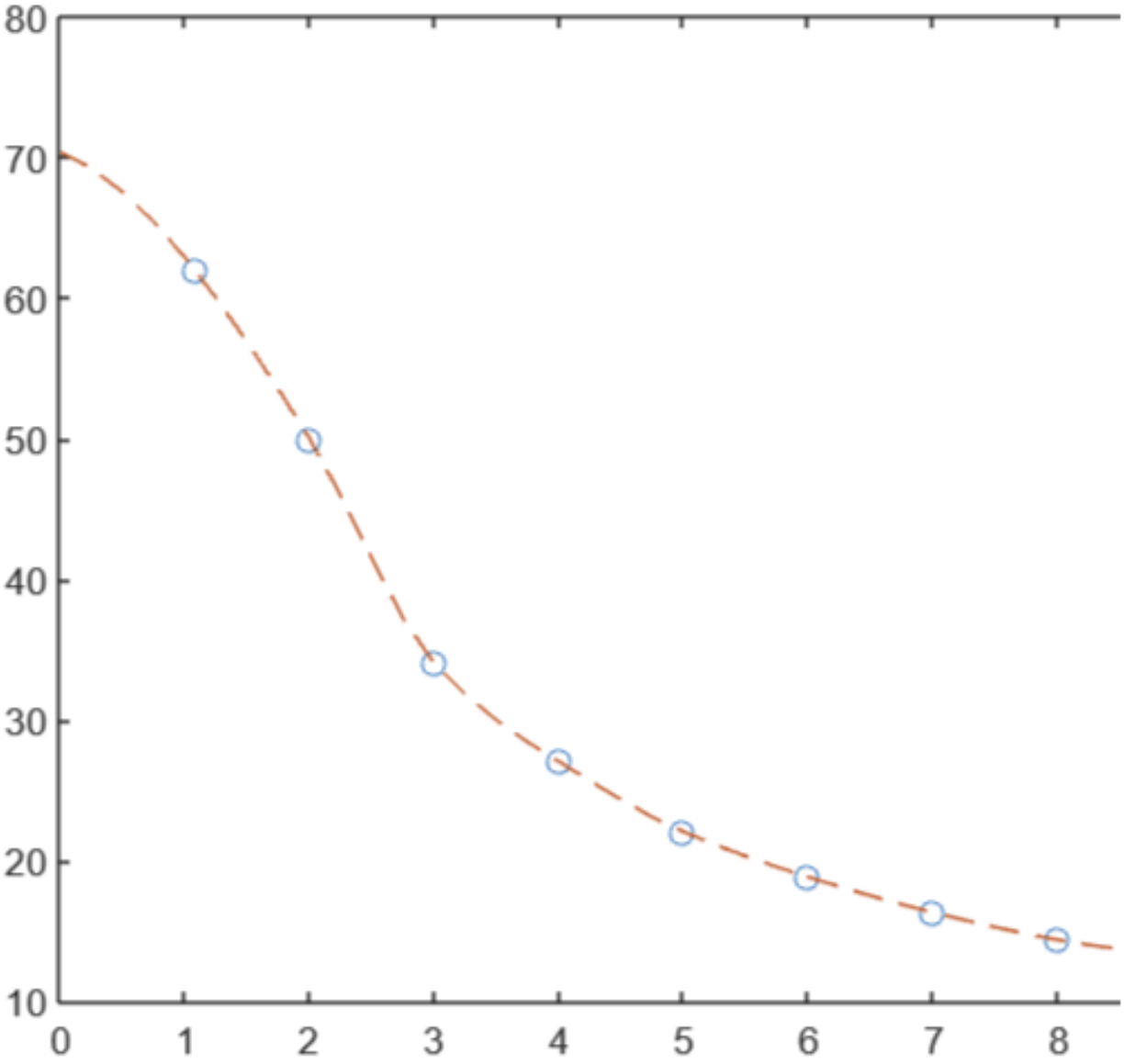
The second subsystem, the flywheel class, was used to control the actual motors that run the flywheel and do the shooting.

Then, using these two subsystems, we built a command-based system which implemented functions from each subsystem in a specific sequence. We bound these commands to buttons on our driver controllers.

One problem we encountered for the shooter was finding the correct aim angles. At the beginning we thought aiming directly at the speaker and adding a number of degrees would be good enough to shoot on target, but that turned out to be ineffective.

Instead, we decided to use linear interpolation to find the correct angle and flywheel speeds. To do this we took data at different distances to the speaker, finding the optimal angle and corresponding flywheel speeds for each increment of one meter. This allows us to shoot accurately from close, far, and anywhere

in between. The interpolation also allows us to vary top and bottom flywheel speeds independently




*Here's an example spline of our shooter angle in degrees based on distance to the speaker*

## CLIMB

The climb mechanism consists of two independent arms controlled by a subsystem. Each arm is equipped with a motor to act as a winch, a limit switch for zeroing, and a solenoid to secure the hook in the down position.

Each is independently controlled by a position-based PID controller to make for a simpler interface to both the rest of the code and the drivers. A "lower" command and a "raise" command each set the hooks' PID target to bumper level and Chain height, respectively. Independent control over each arm should also allow each arm to be set to different heights, a feature which we could use to balance the robot on an uneven chain if necessary.

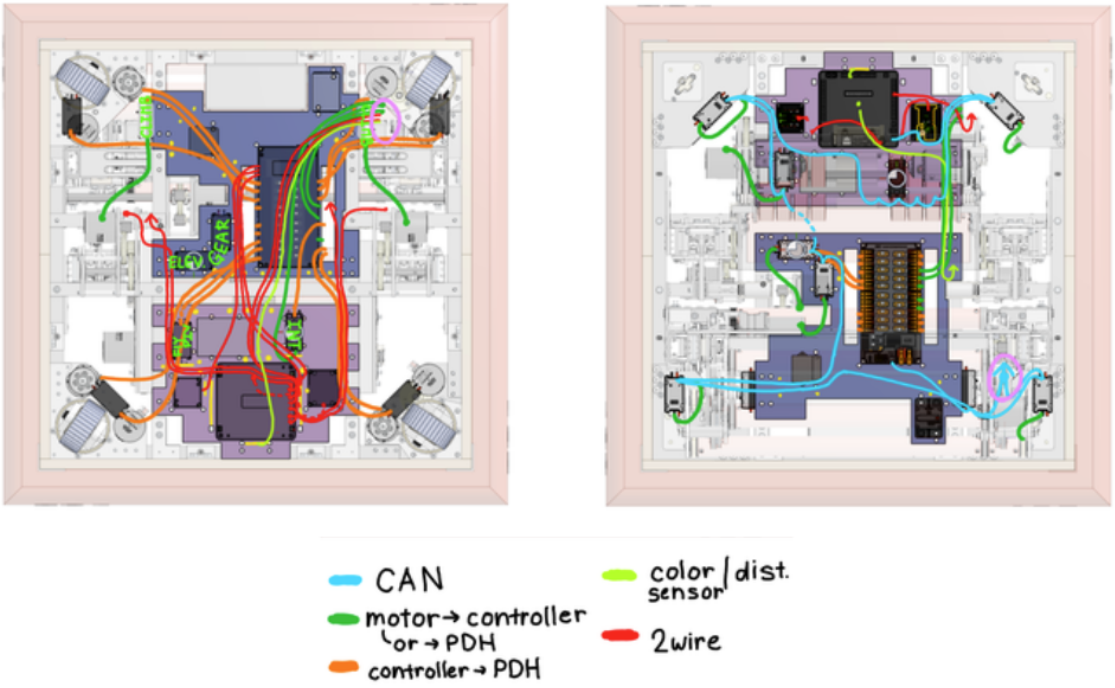
We also have a second, manual mode for operating the arms, which feeds analog inputs from our mechanism driver directly to the motors as power. While this mode is theoretically more difficult to operate, it serves as a barebones, backup system using solely the winch motors so that we can still climb even if a part of the PID system fails.

# Controls Board

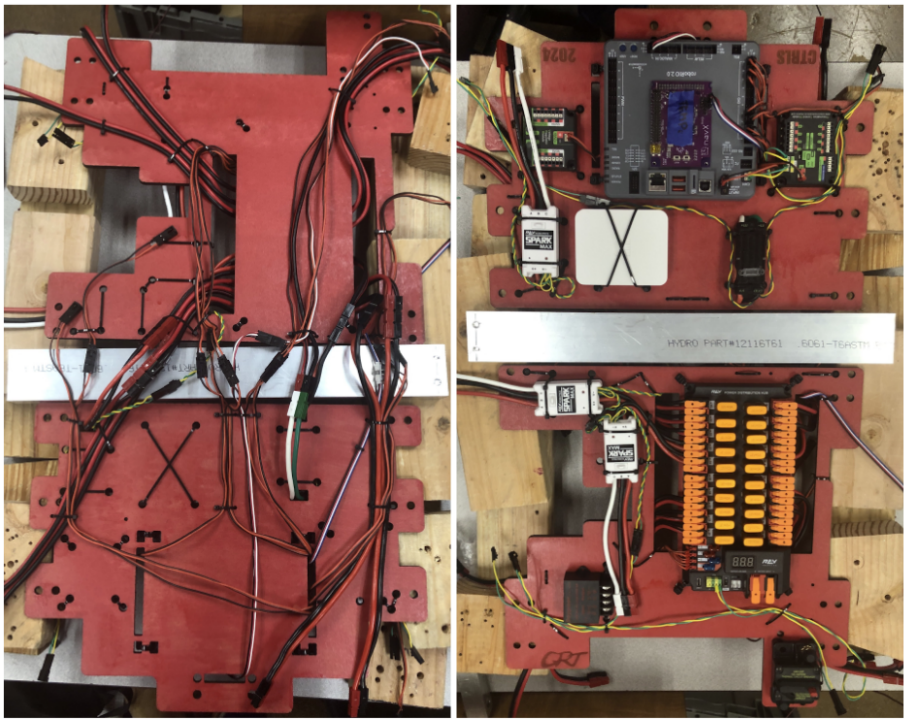
One goal for this year's controls board was to have all of the components on the top side of the board. This was because the previous year had all components mounted on the underside of the robot, resulting in pit crew having to constantly flip the robot on its side between matches to ensure everything is wired and correct. With components on the top, the robot can stay upright at all times, so that mech and controls can work in parallel, instead of one having to wait for the other.

To have a cleaner, and easier to navigate controls board, the board is littered with access holes, which allow almost all of the wiring to be on the underside of the robot, with only the CAN chain and necessary wires on the top.

This allows the CAN to be traceable and the components to be easily accessed. Because the board is split into two parts due to a crossbeam, all the wiring in between the boards was also directed to go underneath the robot, allowing easy to reach assembly.




*Wiring diagram of the controls board, including a top and bottom view that was used during assembly*



*Final wiring of controls board*