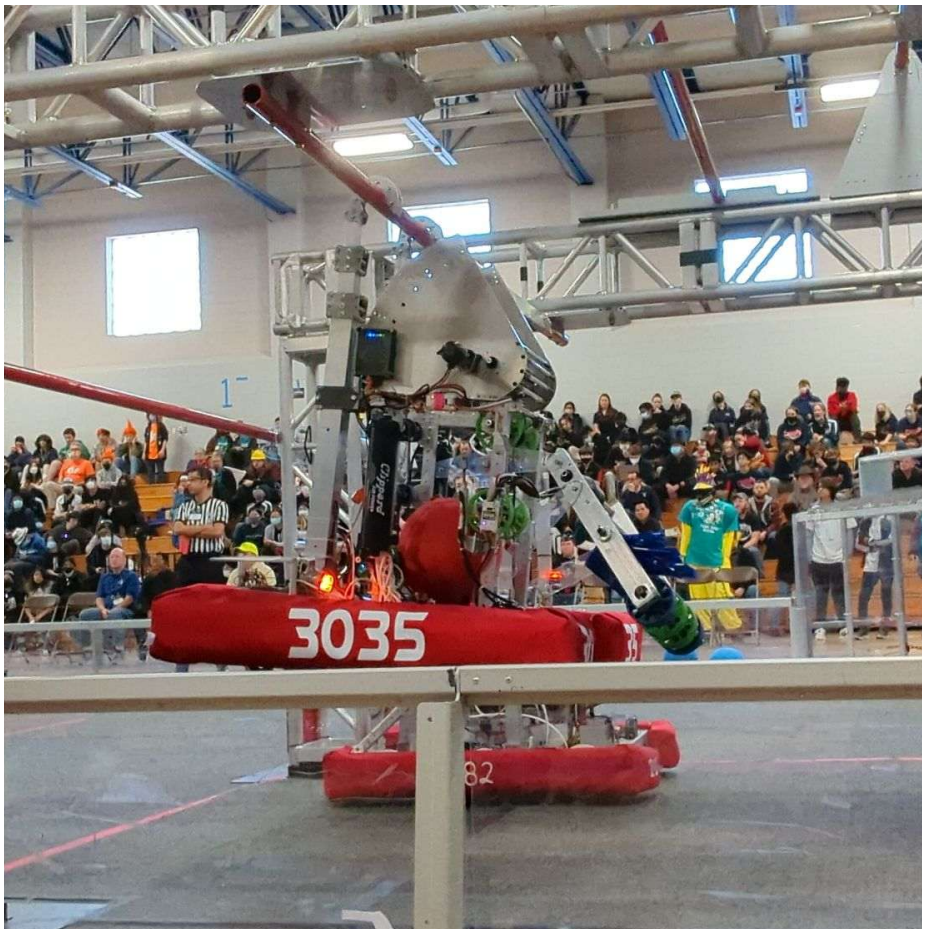
COLLEGIATE HS ROBOTICS

DROID RAGE

# Engineering Portfolio

## Team 3035




RAPID REACT℠

# *3035 Summary*

## Team Structure

**Engineering Team:**
**Design, CAD, and Build Robot**
- Christian
- Chris
- Raul
- Bella

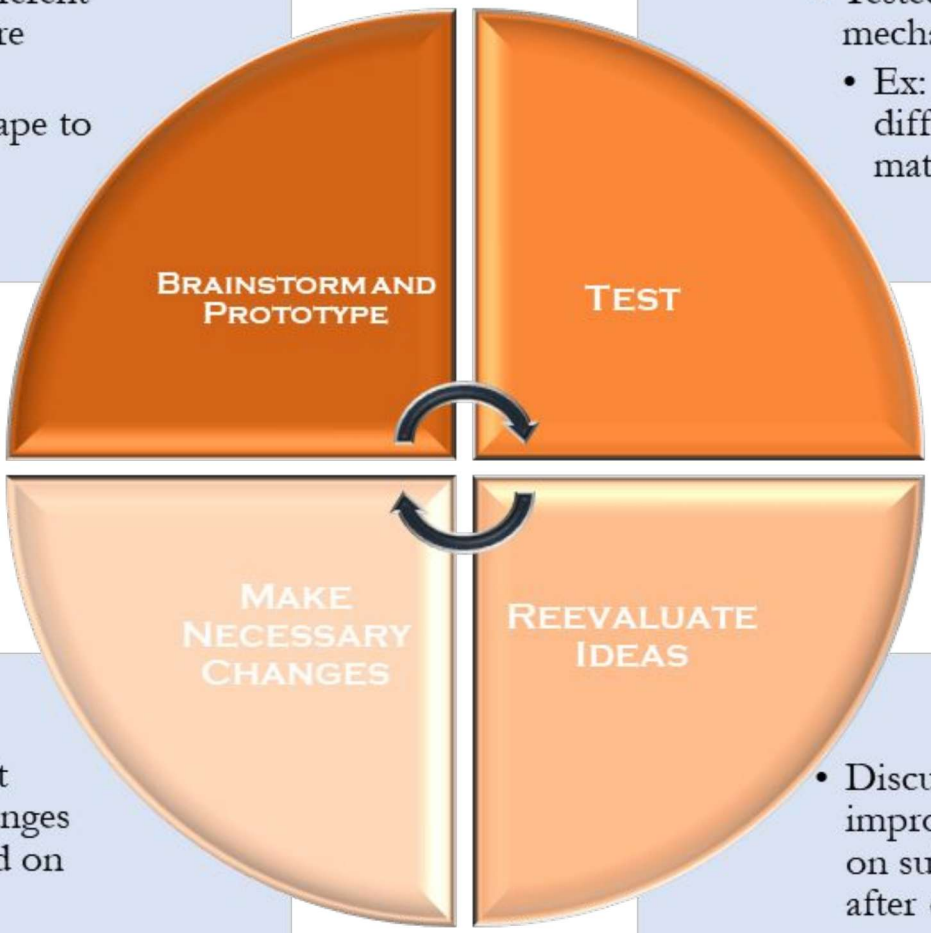
**Programming Team:**
- Justin
- Sai

**Outreach/Scouting Team:**
- Andrea
- Citlali
- Ashley
- Joshua
- Marley

## Design Philosophy

- Dicuss different ideas before CAD
- Use Onshape to CAD

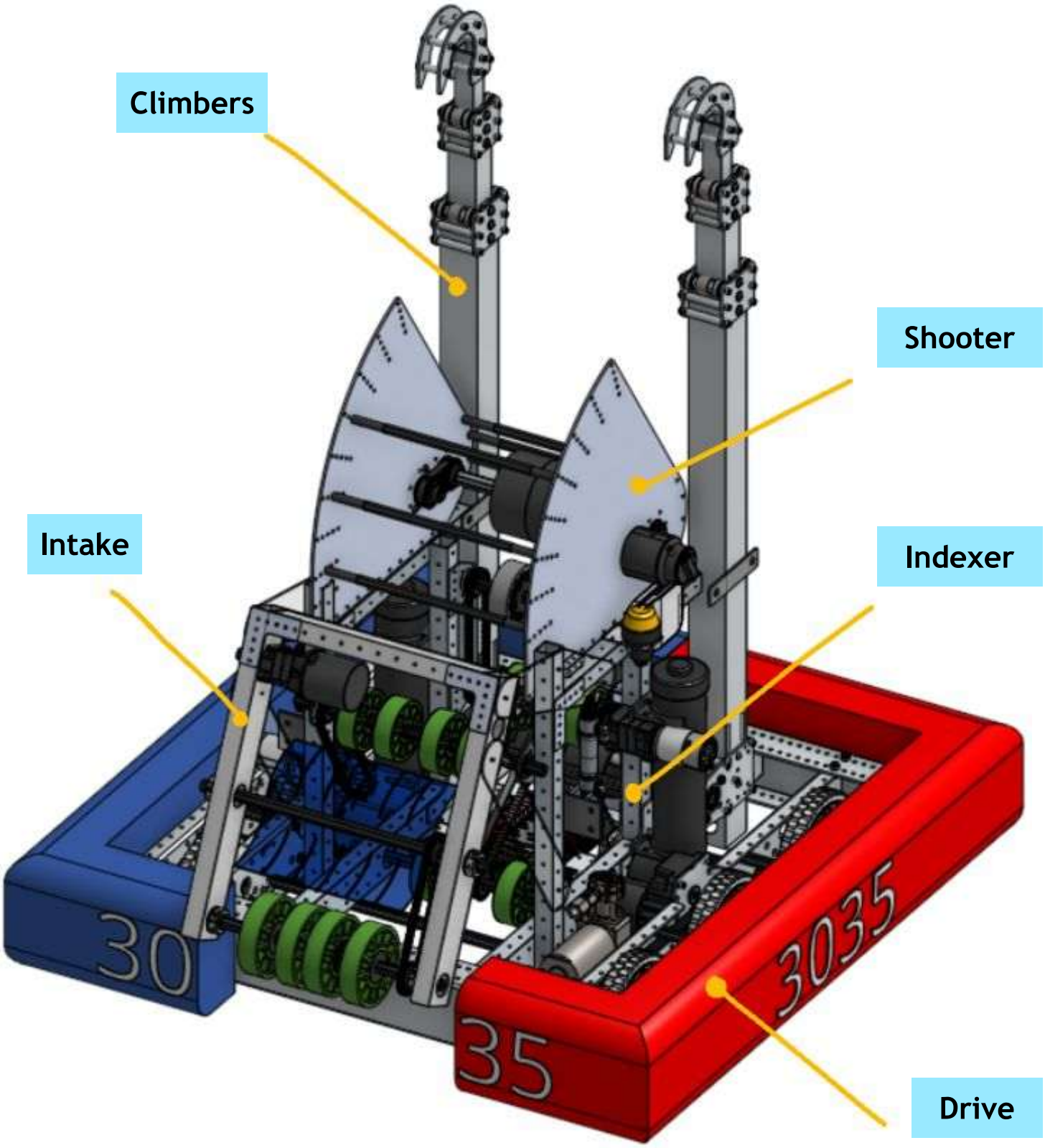
- Tested different mechanisms
  - Ex: Tested different intake material




- Implement design changes we decided on

- Discuss improvements on subsystems after competition

# *Overview of Droid Rage*

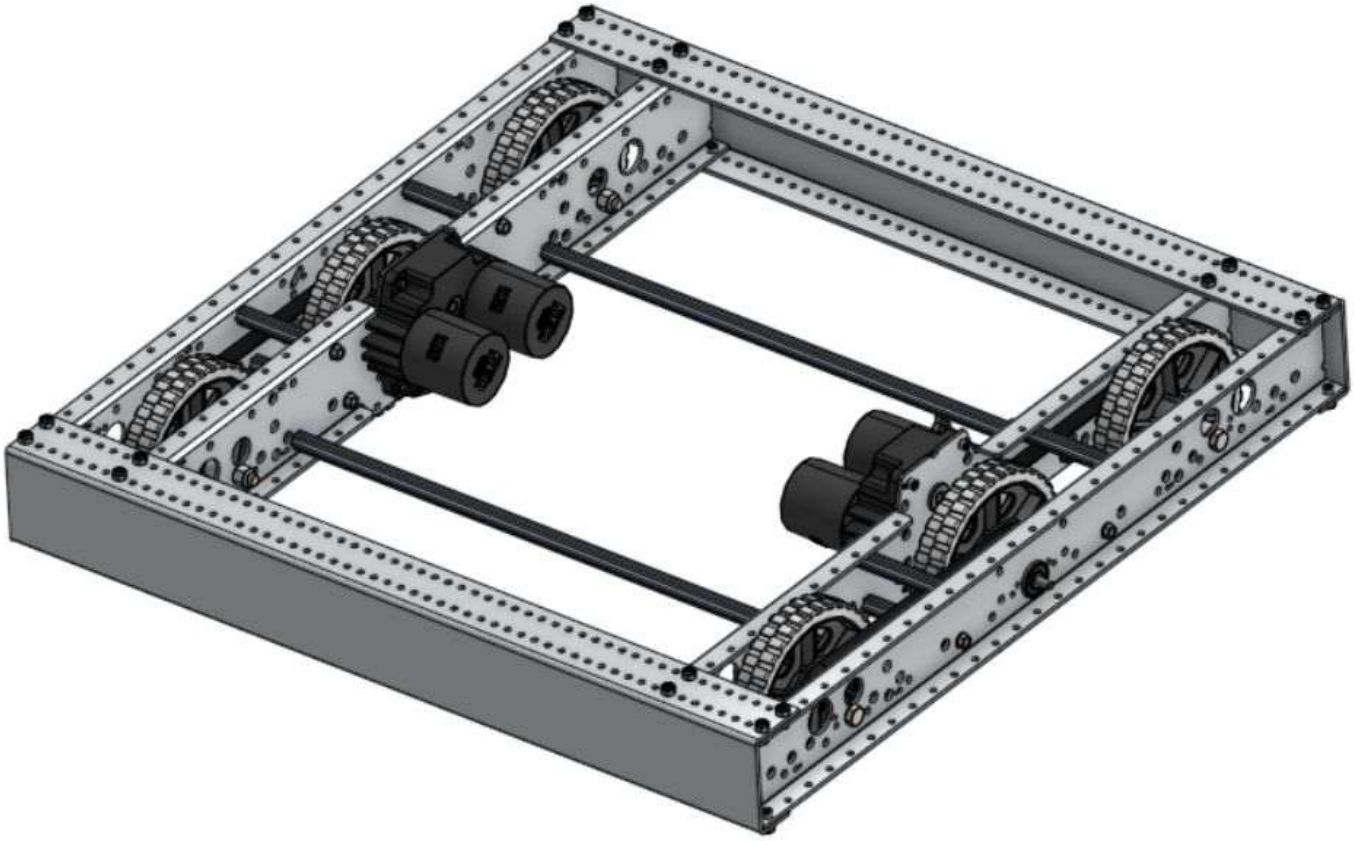



Size: 28.5 x 28.5 x 45”

Six Wheel Drive, 10 Motors

# *Drivetrain*

**Length: 28.5” and Width: 28.5”**



## Programming

- ❏ Used **Ramsete controller** to control during autonomous
- ❏ During telop, the drivetrain is controlled using arcade drive and has multiple speed buttons to use for different parts of the game

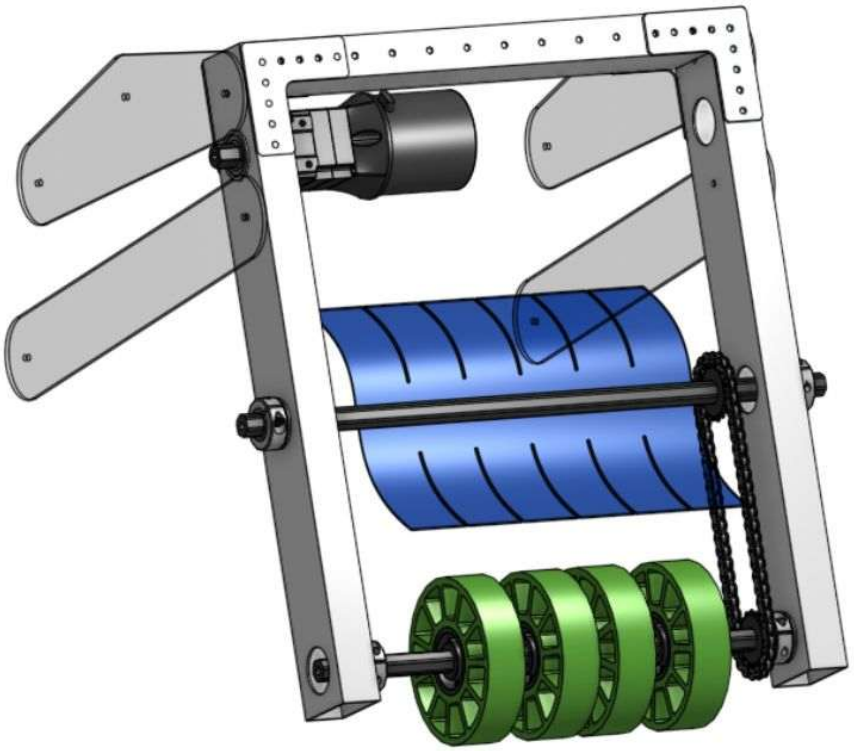
## Goal

- ❏ Capable of defense and offense strategy
- ❏ Balanced with indexer mounting options

## Solution

- 6 wheel tank drive
- AndyMark Square Toughbox Mini drive
- Powered by 4 Rev Neo motors on a gearbox

# *Intake*



## Programming

❑ Driver 1: Set speed by left and right trigger

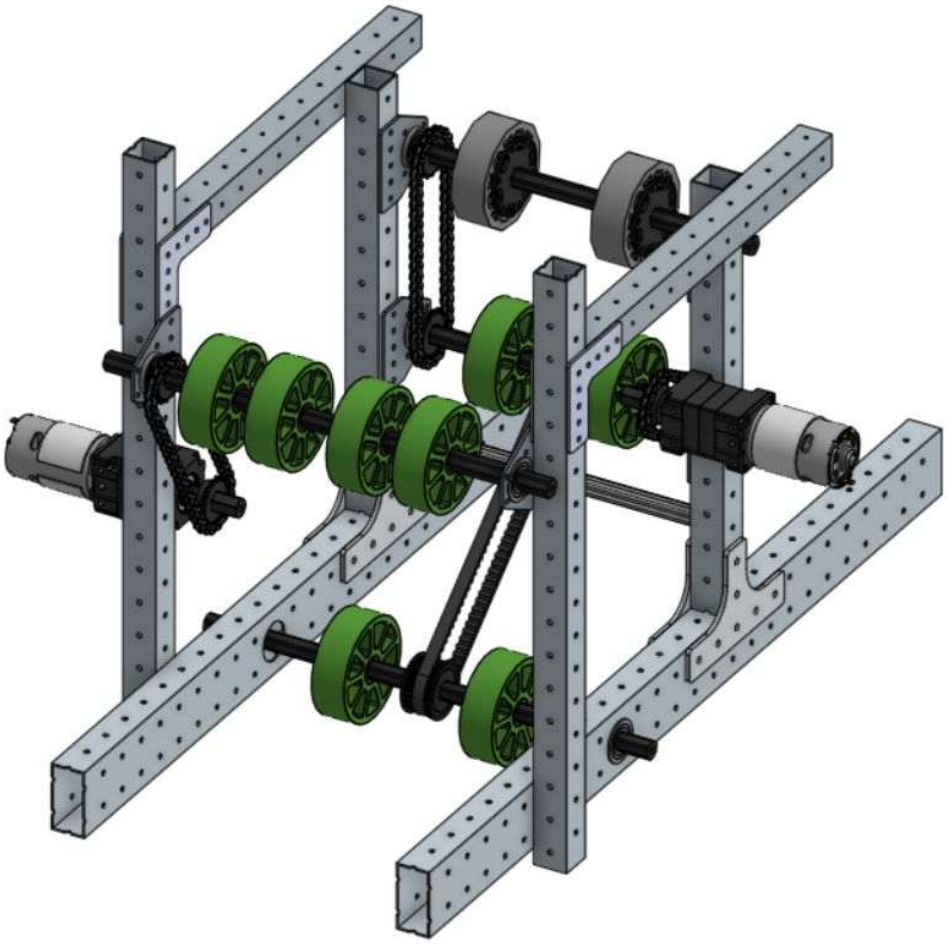
## Goal

- Mechanism that can extend or retract when needed
- Retrieves *Cargo* using an over-bumper method
- Transports *Cargo* to Indexer

## Solution

- Extension/Retraction
  - 2 pneumatic cylinder actuators
  - 4 polycarb plate support
- Stages
  - 4, 3” Compliant Wheels spaced at 1” apart
    - Set [ ] off the ground for sufficient *Cargo* compression
  - PVC vinyl sheets
    - Transfers *Cargo* to Indexing mechanism
- Powered by a NEO Brushless Motor

# *Indexer*



## Programming

❑ Goes up or down with the control of the operator

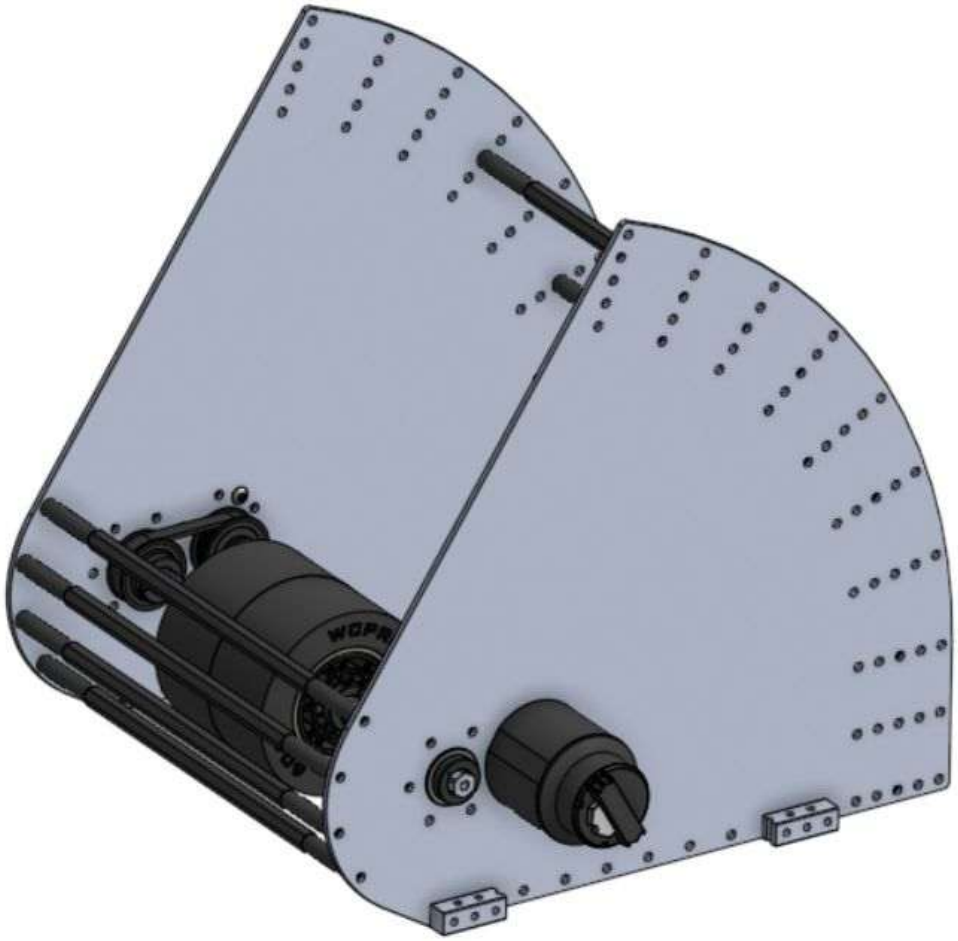
## Goal

- Transfer *Cargo* between intake and Shooter
- Hold 2 *Cargo* pieces at a time
- Capable of both intaking and outaking *Cargo*

## Solution

- 5 Stage vertical system
  - 4” Compliant wheel arrangements
  - Front stages (2 rollers)
  - Rear stages (3 rollers)
- Front and Rear stages set at 8.5” for sufficient *Cargo* compression
- Belted on bottom rollers for additional transportation
- Powered by 2, 775 motors with VersaPlanetary Gearboxes

# *Shooter (Outtake)*



## Programming

- ❑ Using PID to control the speed of the shooter
- ❑ Added weights help with the inertia of the wheel to run it at same RPM

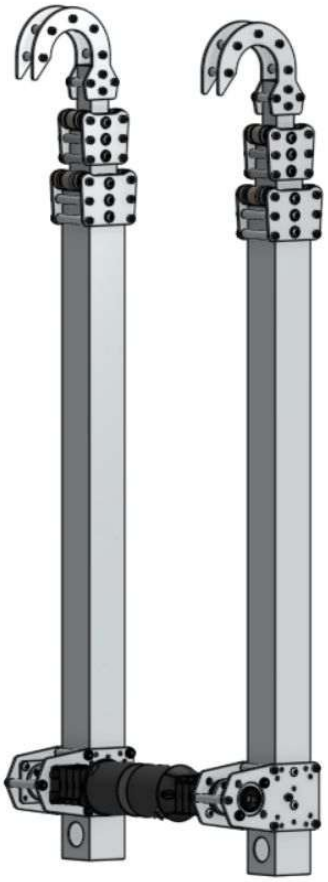
## Goal

- Deliver *Cargo* to both high and low hubs
- Capable of defense strategy
- Consistent, reliable mechanism

## Solution

- GreyT Shooter from WestCoast Products
  - 2 Falcon 500 motors
  - 2 rubber wheels spin 8,000 rpm max and compress *Cargo*
  - Hex shafts & standoffs help guide *Cargo* at around 60˚
- Capable of scoring high and low hubs

# *Climbers*



## Programming

- ❏ When d-pad up or down is pressed, the climbers are given a set power which makes them either go up or down.

## Goal

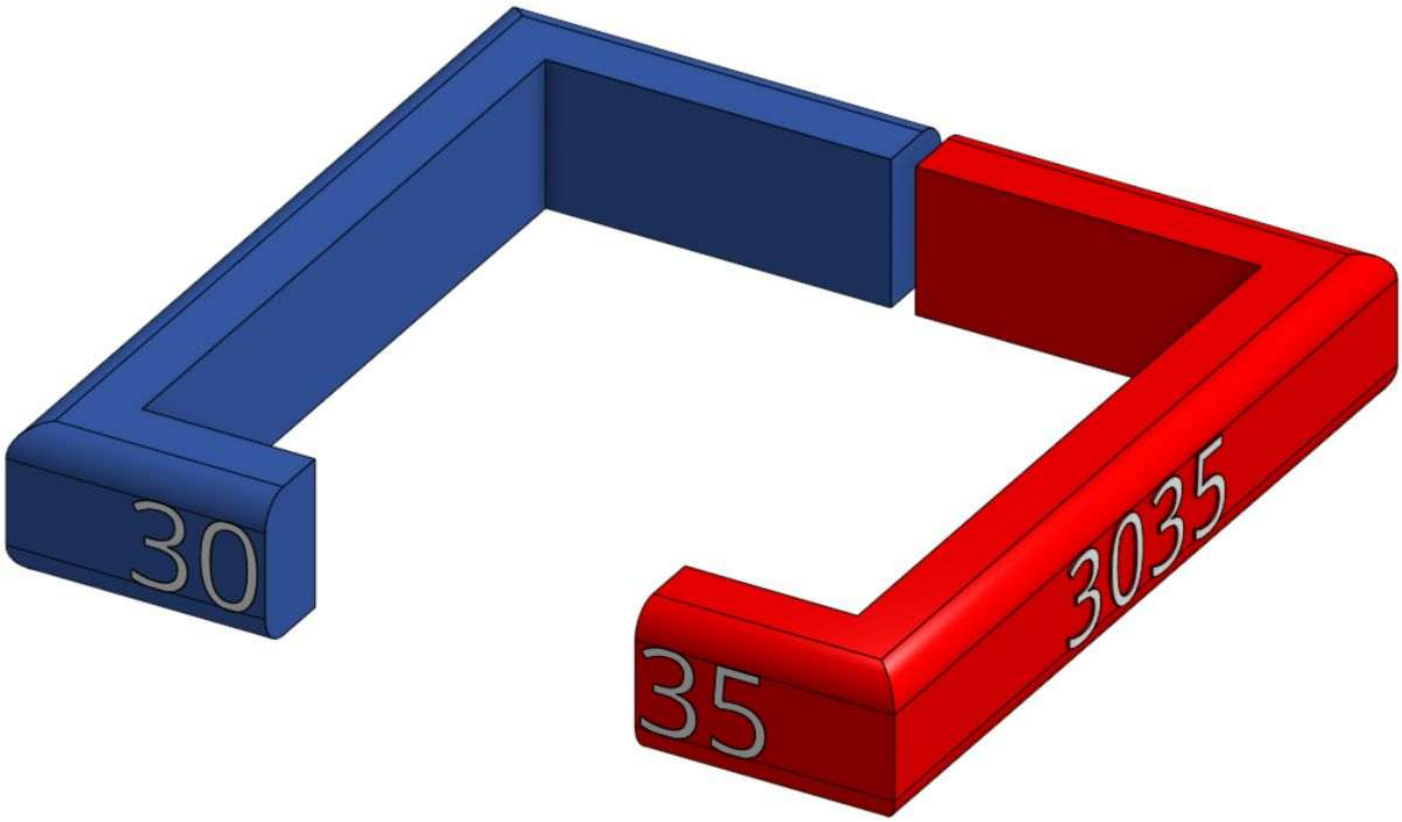
- Lift robot during *Endgame* up to the Middle rung
- Climb from Middle rung to High Rung using traverse arms

## Solution

Originally, we used the AndyMark *Climber In A Box* 2 Stage. However, we modified it with a double sided claw that we made using a cnc.

- ❏ Used 2 REV Neo Motors on a 16:1 Gearbox to control each side of the climb independently.
  - ❏ Tested to carry at least 140 pounds comfortably.
- ❏ 2 Neos controlled using encoders for ease of driver use.
- ❏ Claws 2’’ diameter to fit on the bar comfortably.

# *Bumpers*




## Solution

Materials

- RED AND BLUE Slick Nylon bumper fabric
- Plywood
- Red and Blue pool noodles
- Duct tape
- L shape brackets
- Front bumper brackets
- Side bumper brackets
- White adhesive vinyl

The first thing we did was decide the type of bumpers we wanted; we chose to use two C-shaped bumpers to be able to attach and detach them quickly onto the drive train. The next thing to do was the assembly of the bumpers. Then lastly was the mounting of the bumpers onto the drivetrain.

# *Software I*

## Programming:

**Command-Based Programming (from WPILib)**

- Separates code into subsystems and commands
  - Subsystems separate different mechanisms on the robot; It also holds a group of robot hardware to operate together.
  - Commands combine different parts of the subsystem

**Java**

- High-level
- Object-oriented Language

## Ramsete Controller:

- A trajectory tracker that is a part of **WPILib**
- Uses values: b, zeta, k-heading, and k-linear to control the velocity of the robot by calculating and **processing kinematics**
- Helps deviate back to its original trajectory and pathing, if it strays off
- Better than the PID controller, because it has the robot deviate back to the intended pathing

## Key Commands:

We use different commands to make the code simpler and easier for programmers. Some of examples of this are:

- ShootForSeconds - This command is used for the autonomous period where the shooter will shoot for the specified amount of time.
- DriveByTime - Makes drivetrain drive front or back for a specified amount of time
- TurnByTime - Makes drivetrain turn for a specified amount of time

# *Software II*

## Driver Control Enhancements:

Driver One will be having control of our drive and intake.

- A **slow mode trigger** was added so that the driver can drive more accurately near the hanger and other places.
- A **fast mode trigger** was added so that the driver can drive at full speed if the
-

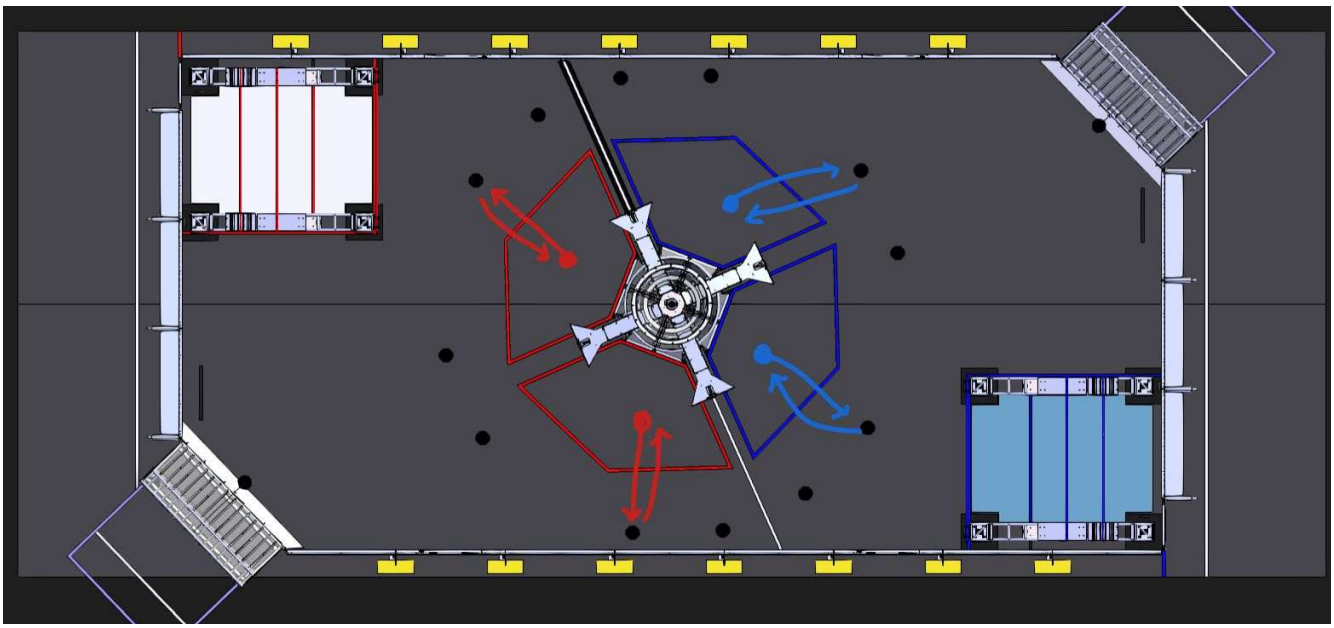
Driver Two on the other hand will have control of the climb, indexer, and shooter

- **Button to reset** entire outtake mechanism.
- The outtake box also comes up as soon as the color sensor recognizes that the robot have a freight in the box.

```
.add("slowMode", RB)    //Slow Mode
  .whenActive(drivetrain::slowDrive, drivetrain)

  .whenInactive(drivetrain::normalDrive, drivetrain)

.add("turboMode", LB)   //Turbo Mode
  .whenActive(drivetrain::turboDrive, drivetrain)

  .whenInactive(drivetrain::normalDrive, drivetrain)
```

## Color Sensor:

- **Color Sensor to index balls quickly**
  - Used the RGB values to correctly check which ball to index during the match for ease of use for drivers
  - Also used to play strategically to push balls to the opposite side of the field



# *Season Progression*

Our designs, experience, and ideas have greatly improved over the course of our competitions, which has helped us improve our games from shooting low  in first comp to high

## Irving Improvements

- ❑ Reversed Shooter (Saves 4 seconds per cycle)
- ❑ Added a reverse roller for shooting ball at a neutral spin
- ❑ Made Indexer Shorter (Allowed for more ease for shooting in the Upper Hub)
- ❑ Climber from 2 stage to 1 stage for ease of use

Managing our time has helped us both in and out of the field, from helping with the swap from driver control, to end-game, and being able to make any repairs much more quickly in between matches.