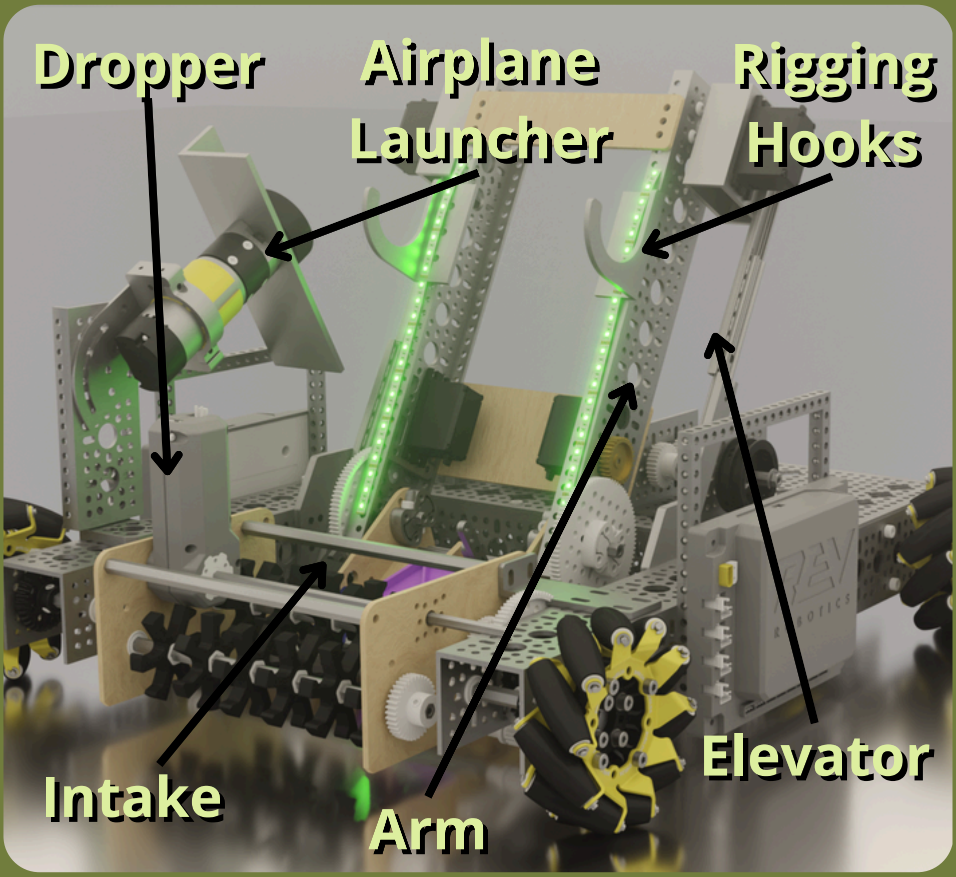
TITAN ROBOTICS FTC 3543



Presents...

ENGINEERING PORTFOLIO 2023-2024

Team Introduction

Hi! We're team 3543 Titan Robotics from International School of Bellevue. Welcome to our Engineering Portfolio!

This is our 14th year in FTC, and we're excited to introduce our robot, Chronos, for Centerstage this year!

Our team is composed of 16 members a diverse range of members from 7th to 10th grade. This year, we have two 7th graders, four 8th graders, nine 9th graders, and two 10th graders. Despite our team this year being almost completely new, our team was ready to take on the challenges of FTC.

Titan Robotics Club's FRC team was formed in 2001 as a Senior Project and expanded to include FTC in 2009 to reach more age groups in our community. We remain one of Washington State's oldest active FTC teams.

Currently, our club consists of one FTC team, one FRC team, and three FLL teams. Our sister team 6541 is unfortunately inactive this year**,** but we were luckily able to return for Centerstage and consolidate our past years' unique experiences and insight into 3543.

Our Team

We are a mix of different talents and passions united under FIRST. It is crucial to us that our robot is a result of everyone's work, from our mentors who guide us through building and programming, to our members who push themselves throughout the season to grow.

Our Values

We always focus on the fundamental principles of FIRST, such as **Gracious Professionalism, cooperation,** and **growth**. We also nurture the core values of International School: **integrity, growth, collaboration,** and **creativity**.

We support other teams, whether it be lending them parts at competitions or advising them online about how to troubleshoot and implement code.

Team Member Profiles

Elliana

Alyssa

Anya

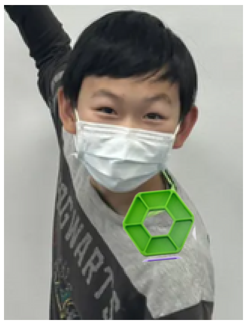
Daniel

Noah

Lucas

Ricky

Ashwin


Anthony

Benjamin

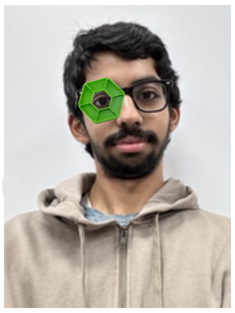
Charlie

Krishan

Kinghang

Nidheesh

Jason


Avijit

Sam (Mentor)

Andy (Mentor)

Dave (Mentor)

Mike (Mentor)

School Interaction

Titan Robotics Club

Titan Robotics Club is a diverse family that creates a sustainable pipeline by including multiple teams through every grade level at our school. Our three FIRST Lego League (FLL) teams introduce middle schoolers to the world of robotics and STEM. As these students progress to high school, many of them continue their involvement in the FIRST program by joining either our FIRST Tech Challenge (FTC) or FIRST Robotics Competition (FRC) teams. This allows them to build upon the skills and knowledge they gained in FLL and continue to explore the exciting world of robotics.

Our School

Our school is a 6-12 choice/lottery-based college preparatory school within the Bellevue School District. Titan Robotics Club provides the practical application to the theory taught in classroom, allowing students to study science and technology outside of school while enjoying the fun of robotics! Although there are no high school sports at our school, we are proud to say that **robotics is our sport**, with nearly **1/6** of the school population participating in the club this year.

Recruitment

We heavily expanded on the recruiting efforts this year. We started at the 5th-grade barbeque, a special event where prospective 5th graders come to our school to eat lunch, to garner interest in those who will join the school in the future. We then participated in back to the business day and club fairs, where we were able to reach more of the student population and publicize the club more. With word of mouth about our success last year, there was even more excitement associated with joining the club.

Team Sustainability

FLL Connection

As a family with multiple FLL teams, there is always a constant stream of young students striving to assemble more than just Legos. Our role as high schoolers mentoring three FLL teams, allows students already exposed to the values of FIRST to seamlessly advance into the FTC team - giving us a chance to pass down our knowledge to younger kids. FLL consists of 3 main components, which include gameplay, core values, and research projects. Each component of FLL prepares our students for FTC by teaching them the very basics of design, coding, and most importantly teamwork.

TRCLib + Documentation

Our robot software is based on TRCLib, our own open-source software library with features such as PID drive, vector-based odometry, and abstraction layer for code compatibility between FRC and FTC robots. This library, which is publicly available on GITHUB, is used by FTC and FRC teams around the world, like Rise of Hephaestus in San Diego and Fixit in Victoria.
The CAD files for each year's robot are also saved in a CAD library, serving as reference for future years' games. We also keep meeting notes and maintain a guide for the various tools available at our shop.

Grants + Sponsorships

The Titan Robotics community connects with Boeing, Perpetual Technologies, and other long-term sponsors to operate. Because we are building robots, offering off-season opportunities, and making constant use of our new warehouse, the donations and grants they offer us are essential. For example, FIRST donates an annual grant to help offset the cost of registration. However, our robotics team also receives community-based funding as well. From school fundraisers to PTSA donations, our team also focuses on maintaining a positive relationship with our locals.

FRC Connection

Through our special club structure, we are able to share resources critical to our club's operation with our FRC team, team 492, such as our workspace and mentors. We are also able to draw on the massive population of the FRC team for special events such as the two Turing League competitions we hosted this year. With the expertise in the FRC team, we are able to host pre-season training and workshops for **shop safety, CAD design, and tool usage**.

Community Outreach

Seafair Summer Festival

We demonstrated both FRC and FTC robots to the people of all ages at the 2023 Seafair Festival

Cherry Crest Eng. Night

We were a popular hit at the Cherry Crest Family Engineering Night - kids loved driving and playing with the FTC bot!

Winter Market/Back to Buisness

We demonstrated our club and robot to our school for recruitment activities.

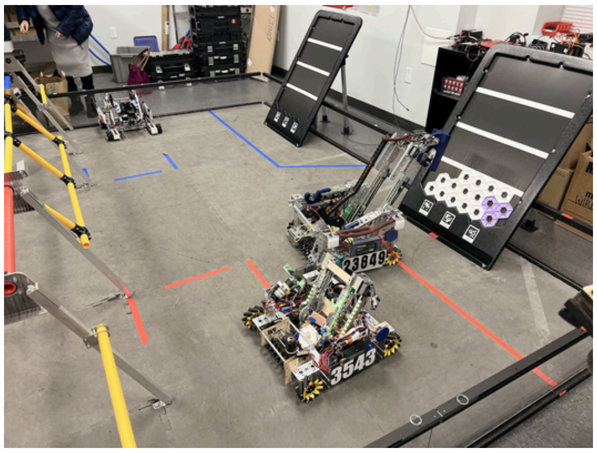
Olympia for WA Legislation Day

We visited Olympia along with our FRC team to share our passion for FIRST Robotics.

Hosting Competitions

Cooperation

As competition hosts for the Turing League, we want every team to have the opportunity to succeed. Multiple members of our team help other teams to solve some of their issues, including securing power switch mounting, and lending our portable battery pack to ensure that their driver station remained charged. We constantly supported the other teams during the competitions, by sharing resources with the other teams and congratulating others on their achievements. We also invited 3 other teams to a practice competition at our warehouse.



Business Team

This competition was an excellent opportunity for our business team as well. We worked hard to set up and sell concessions, raising over 500 dollars, which goes towards our club finances.

Beater Intake

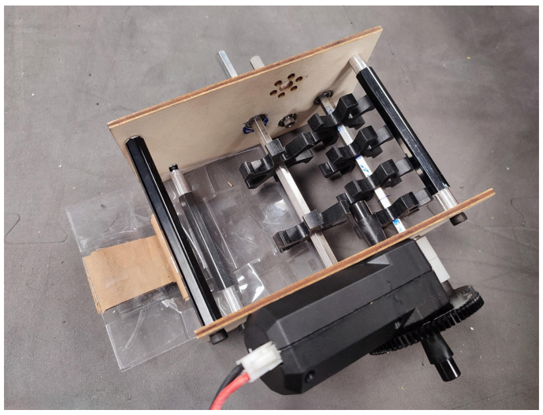
Overview

Our intake is a beater which moves the pixels through a series of rollers and drops them onto the 'end effector' that we call a dropper. It is powered by a REV motor and a system of gears to get everything moving, it also has a funnel to get them centered.

Early Prototypes

In the first week when we were coming up with ideas for the beater intake, we took inspiration from an old design. We put a prototype together and it looked promising, with the pixels getting sucked in smoothly. We found six-spoked wheels worked better than surgical tubing, so we swapped them out. We added a ramp so that the pixels would get pushed up into an end effector, but the ramp would drag against the floor.

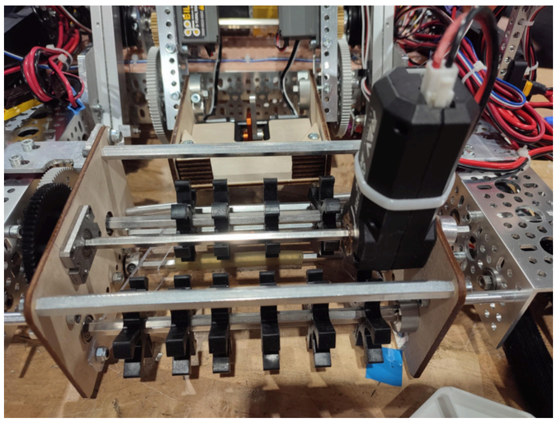
V3: Initial Beater Design



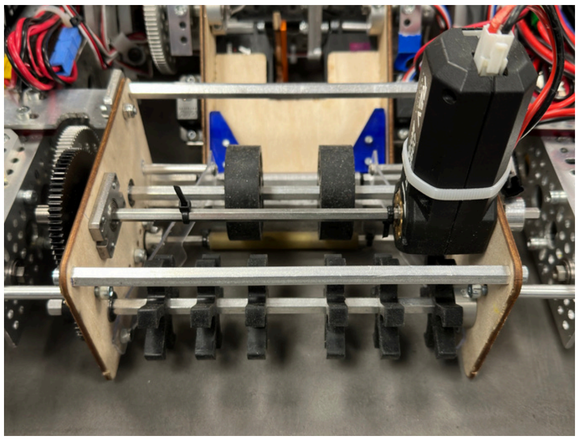
Designing this intake took many iterations, but we ended up creating a reliable intake. When we were designing, we wanted to only use one motor to power the entire intake, so we CADed a matrix of gears to ensure everything turned properly. We also added two polycarbonate funnel pieces to get the pixel centered into the dropper.

After the second qualification match, we realized that the pixels had an inconsistent velocity while going through the intake due to their intricate geometry. Using slow-motion videos, we identified the problem, and we changed the second row of six-spoked wheels to cylindrical wheels so the pressure on the pixel was more consistent. Finally, we changed the number of wheels from four to two.

V9: Qualifications Design



V12: Interleague



Drive Base & Elevator

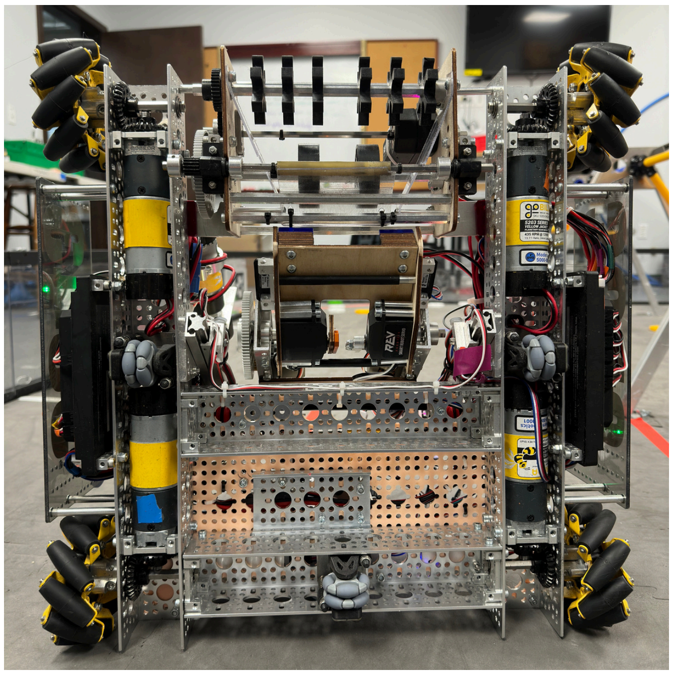
Mecanum Drive

We had developed a prototype swerve drive during the off-season, however we realized that this year we required a more low-profile base. We decided to use mecanum drive base this year. To achieve this, we used a classic Gobilda strafer base with both the motors and the odometry mechanisms tucked into the C-channels. The pseudo H-frame has an open front to create space for the intake mechanism.

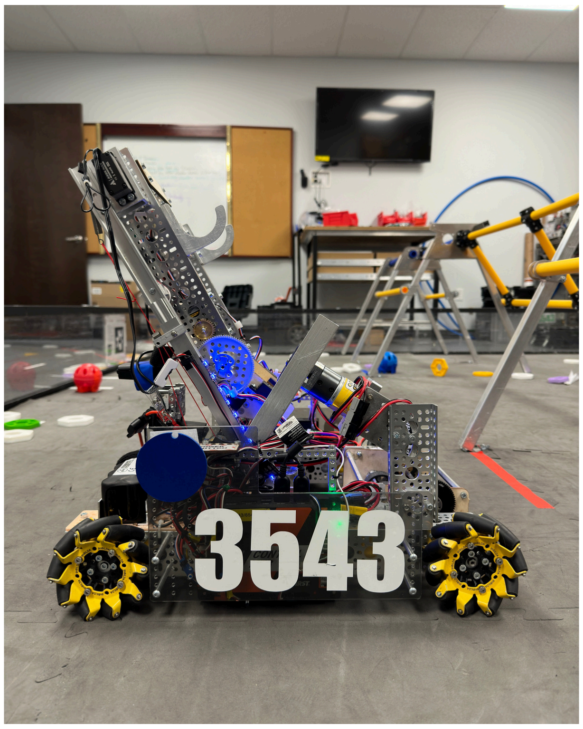
Elevator Design

Our current elevator is a custom two-stage elevator mounted at the same angle as the backboard for easy scoring alignment. The elevator is extended and retracted using a winch mechanism. The arm is attached to the top of the elevator and uses two high torque servos to efficiently reposition the end effector from the front (pick-up) to the back (scoring) positions.

Final design



Angled Elevator

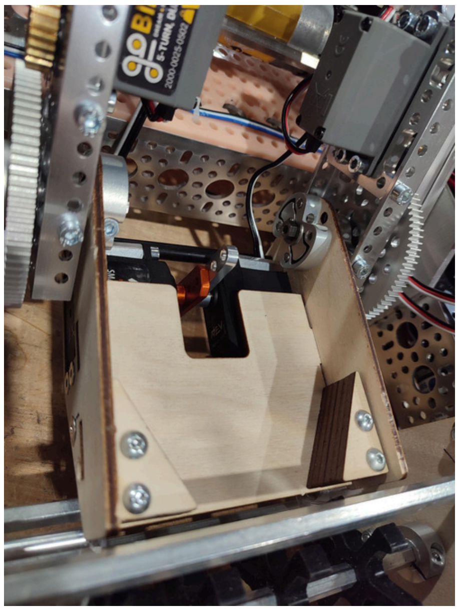


Dropper

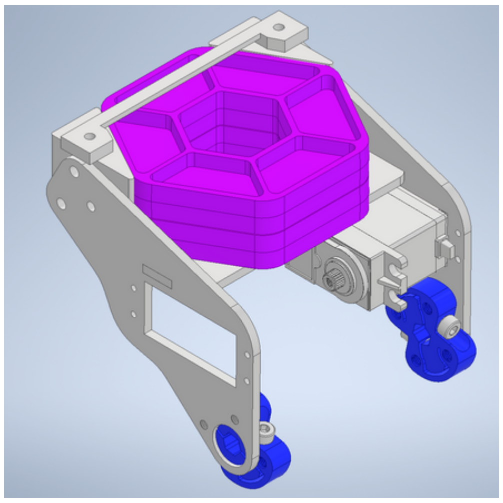
Dropper Design

Our current iteration of our dropper involves a container capable of fitting 2 pixel stacked on top of each other with servo arms to hold each in place. This is a custom made outake design that uses gravity to settle the pixel into the correct position for locking. For ease of use during teleop, we use presets to automatically move the wrist into the correct position for scoring/pickup.

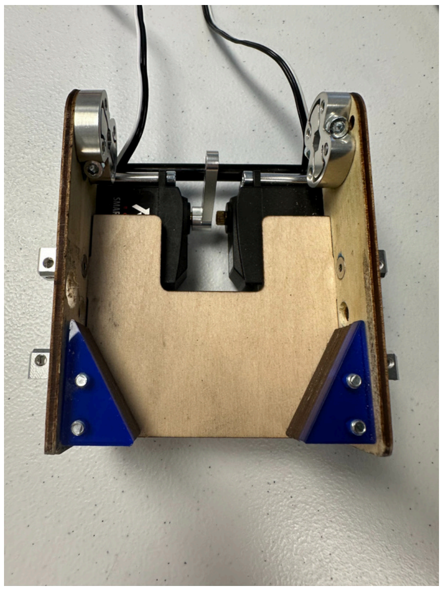
V1: League I



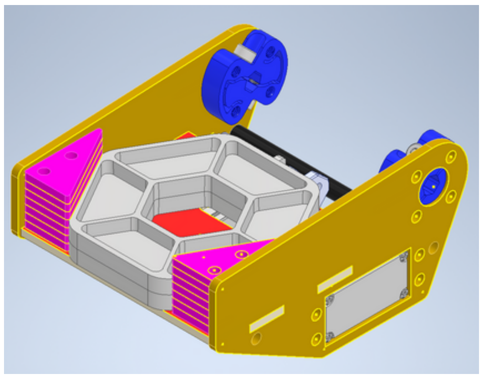
Problem: poorly designed side plates led to side-to-side play



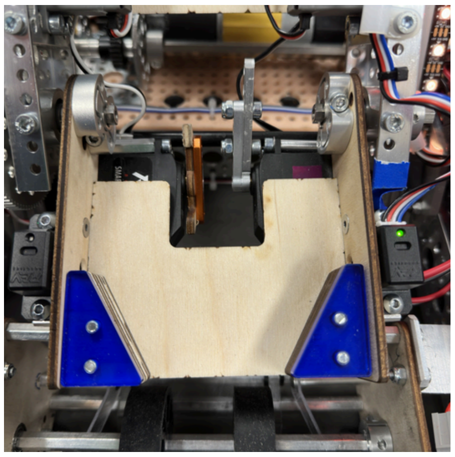
V2: League II



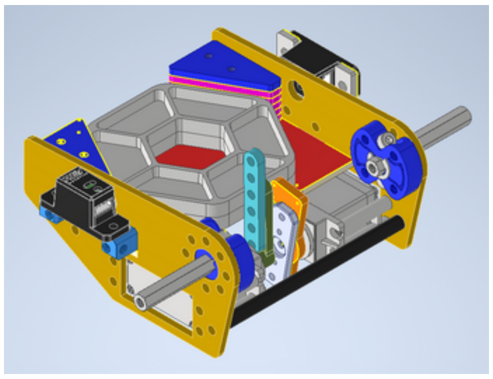
Problem: The Dropper was too skinny so almost half the time, the pixel would not land in the dropper properly, wasting a lot of time.



V3: Interleagues



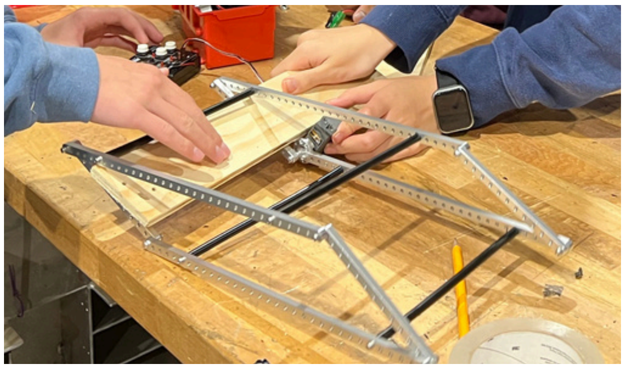
This current iteration now allows for more variation in the pixel intake by making the intake wider, which also helped the pixels settle faster, and more consistently.



Arm & Climber

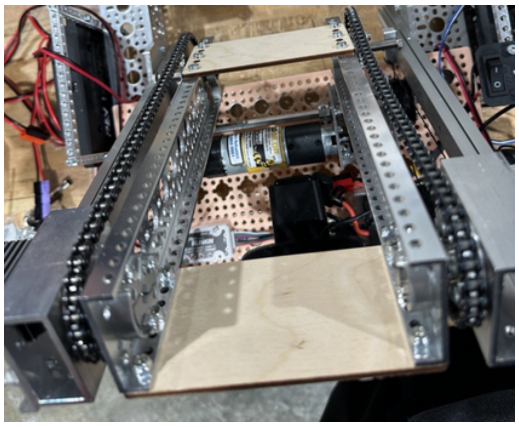
V1: 4 Bar

One of our first ideas for the arm was a four bar. While we were testing and working with the four bar, we recognized a problem with the mechanism. It had a small range of motion which slowed down our scoring.

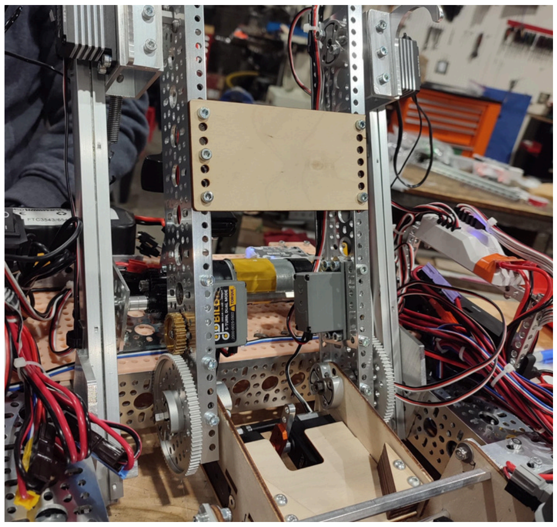


V2: Virtual 4 Bar

After experimenting with the four bar, we decided to test out a virtual four bar. This would enable us to collect pixels from the front, and score them in the back, making scoring faster. However, while building the virtual four bar we encountered many problems. The sprockets were too difficult to align with the chain, and it was even harder to maintain and build. Another issue we encountered was that the mechanism was too big, cluttering our already packed robot.



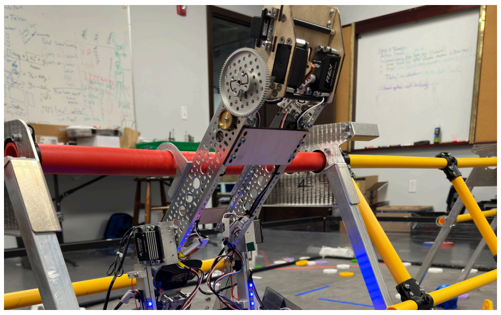
V3(Current): Simple Arm



Since the VBar attempt was unsuccessful, we opted for a simplified version. We replaced the chain with another metal beam. This simplified the mechanism, but kept the features we wanted. A modification to the dropper meant that we would no longer need form of vbar.

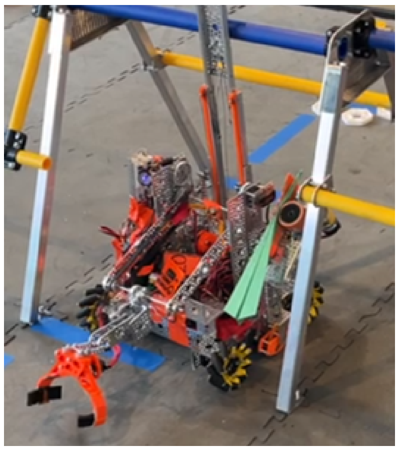
Climber

In order to climb the truss structure, we decided to use a pair of hooks on the elevator. This was unable to reach the bar, so we moved the hooks to the Arm. This presented an issue since climbing would put most of the weight on the Arm Servos, straining them. This was solved by having the servos go limp, putting the weight of the robot on the much more powerful elevator motor.



Airplane Launcher

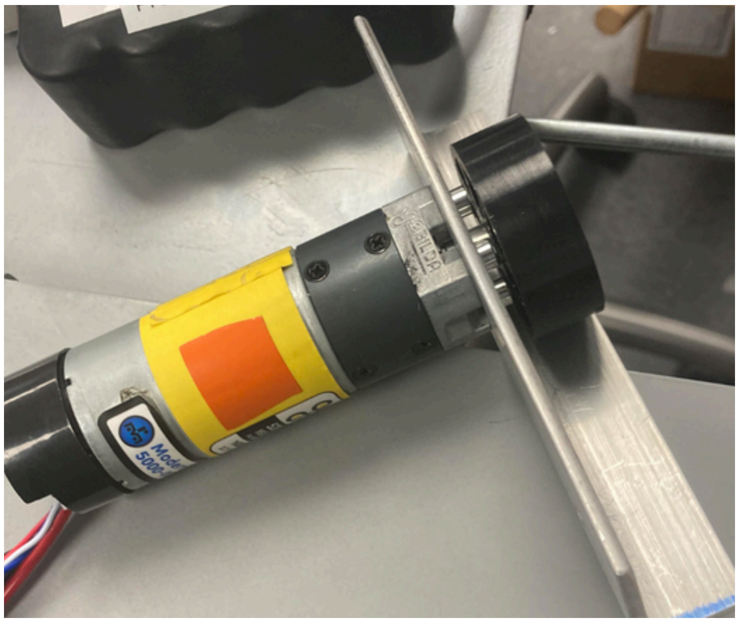
Inspiration



We were able to find this design from team 8365 on Discord, and this became the inspiration for the design.

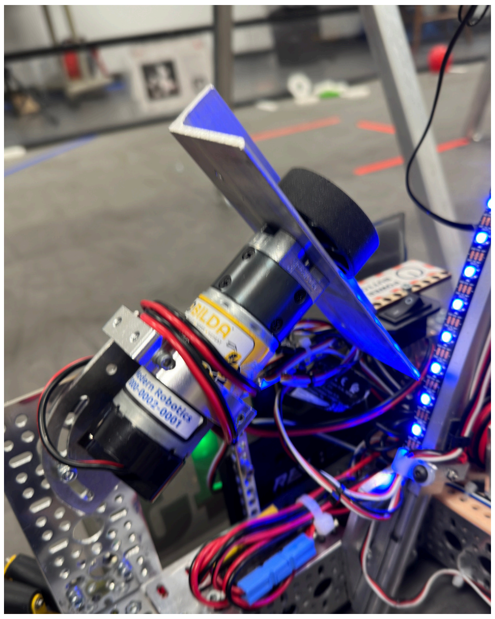
Design Timeline

Our first design was a simple motor with a cylindrical wheel, attached to a piece of scrap metal with the general shape needed. However, a 435 RPM motor Gobilda motor couldn't send the plane far enough, so it was swapped for a 1620 RPM motor that still didn't work, We then swapped it for a 6000 RPM motor that did work and the next problem was mounting.



Robot Placement/ Airplane Design

After testing various angles, motor strengths, and airplane types, we found that the airplane landed decently at 45 degrees, 90% motor power. The airplane design we found that worked well a double folded variation of the standard paper airplane. We mounted it on the robot and saw it worked well just in front of the back wall.



Final launcher on the robot!

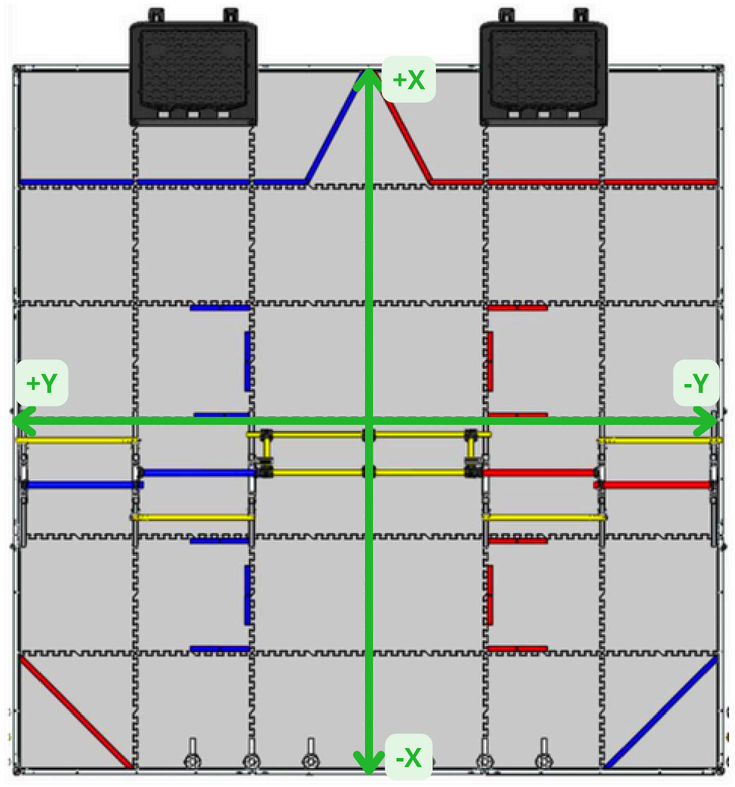
Odometry Background

How We Describe Robot Position

- Coordinates are usually created in tile units for ease of thinking, but we convert them to inches before using them for accuracy.
- Positions are described with (x, y, heading)

What is Odometry?

- Measuring change in position of robot from its previous position using data from motion sensors. (ΔPose)
    - This data can be integrated with time to track robot's absolute field location at a sampling rate of 30/sec (localization).



Why Do We Need Odometry?

- Pure Pursuit requires localization to figure out the robot's location in order to follow the given path, and localization requires odometry.
- Easier to debug - We can print out the robot's position in our trace log so if it executes a pure pursuit path incorrectly, we can check where odometry thought the robot was located.

How Do We Implement Odometry?

- **We used to use:**
    - Live-wheel-odometry: Taking data from the encoders on our drive base motors.
    - **But:** If any of the wheels slip—which is common when driving with mecanum wheels—the odometry gradually become inaccurate.

- **Solution: (Passive or Dead Wheel Odometry)**
    - We now use unpowered Omni wheels with encoders on springs
    - Now, even if the driving wheels slip, we still have these engaged on the ground

Computer Vision

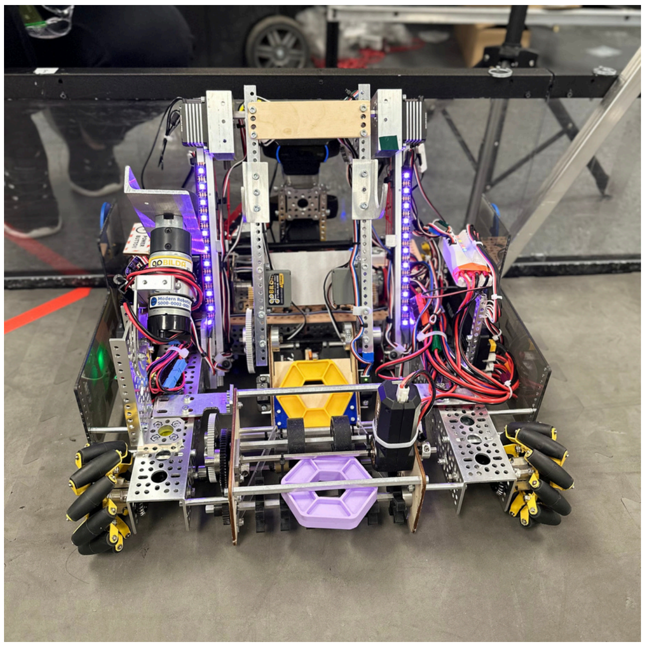
OpenCV

We are using OpenCV's color-blob detection software to locate shapes of different colors. Since this year, there were six objects all of contrasting colors; we didn't need to differentiate between blobs of the same color.

LEDs

During autonomous initialization, our LEDs change color to indicate the detected prop location on spike mark (left-violet, middle-green, right-yellow), see below

During TeleOp, different subsystems display their status with different color patterns. To avoid subsystems fighting for the LEDs, each status color pattern is assigned a priority. For example, our LEDs flash lime if Field Oriented Driving is enabled but if the camera detects an Apriltag, the LEDs will cyan instead of lime. Once we have finished signaling, the LEDs will return to lime.



Pipelines

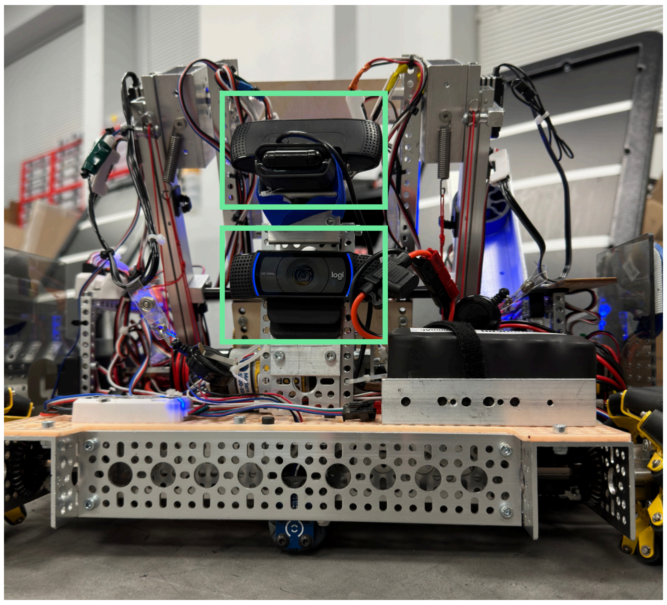
In order to look for an object's position, we use vision pipelines that then takes an image of what the webcam sees and applies a combination of color thresholding and morphological operations to narrow down on our desired subject.

Our pipeline for the Red Team Prop follows these steps:

1. Convert the color space from RGB to YCRCB
2. Color threshold filtering
3. Find and filter contours (This is the blob part of the blob detection, removing stray pixels)
4. Criteria filtering (width, aspect ratio, etc.)

Double Camera Setup

1) Game element detection
2) AprilTag alignment



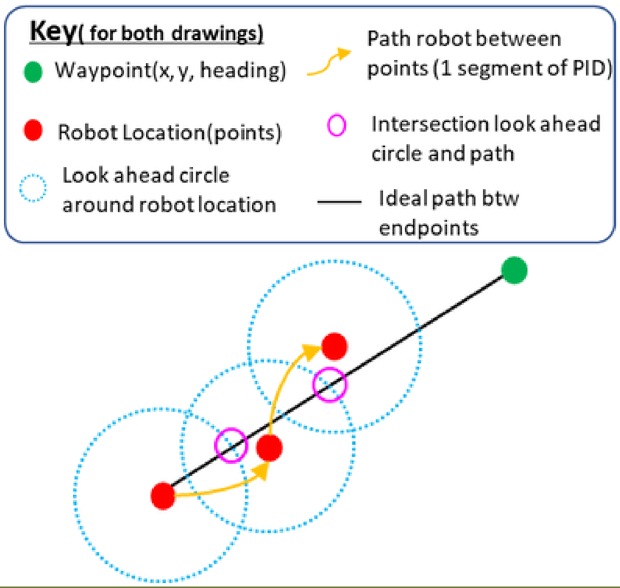
Pure Pursuit + Odometry

What is Pure Pursuit?

A path following algorithm that uses the robot location from odometry to generate curved paths to target points. setting smaller target points in between based on the robot's location. Making the overall journey shorter and makes less obstacles in the way.

We essentially drive to the intersection of lookahead circles (centered around the robot/target points) until we reach the target.

The Standard Pathway (Zig-Zag)


Key(for both drawings)
Waypoint(x, y, heading)
Path robot between points (1 segment of PID)
Robot Location(points)
Intersection look ahead circle and path
Look ahead circle around robot location
Ideal path btw endpoints

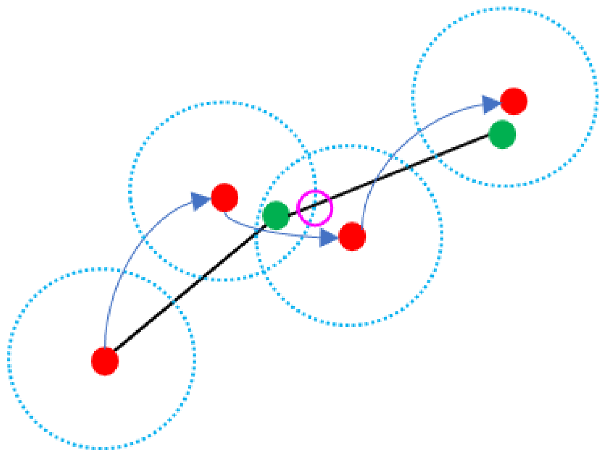
Downside to Our Old Algorithm

The radius of the look-ahead circle is generally about 6 inches and our old algorithm therefore relied on PID-controlled drives with targets of 6 inches or less. This caused the robot to move very slowly: because of drive error, the robot would often overshoot the intersection point and end up zig zagging around the path (i.e. oscillating along the path) instead of following it in a straight line.

Our Improved Algorithm

- To remedy these problems, we implemented v2 Pure Pursuit, which generates even smoother paths
- If the distance of the robot to the next point is more than the look-ahead circle radius, we bypass pure pursuit by setting a PID target to the next waypoint instead of the intersection between the look-ahead circle on a path. Once the distance is inside the radius, we switch back to traditional pure pursuit.
- This solves both problems that occur when the PID target is far away: PID controlled drive is a lot faster getting to the next waypoint -- it's more accurately following a straight line.

Our Improved Algorithm Path



Autonomous & Auto Assists

Context: In our practices, we realized that when scoring, it would be really easy to de-score all the pixels. So we created **AutoScore** an algorithm that will automatically score the pixels.

Algorithm: Locate an April tag to re-localize the robot, drive to the target location, score our pixels.

1. Locate an April tag and re-localize our robot position.
2. Set the elevator, arm, and wrist to scoring position.
3. Drive to the specified location.
4. Stop driving when either it has reached the location, or the distance sensor says it's close enough.
5. Release our pixels.
6. Collapse elevator and arm.

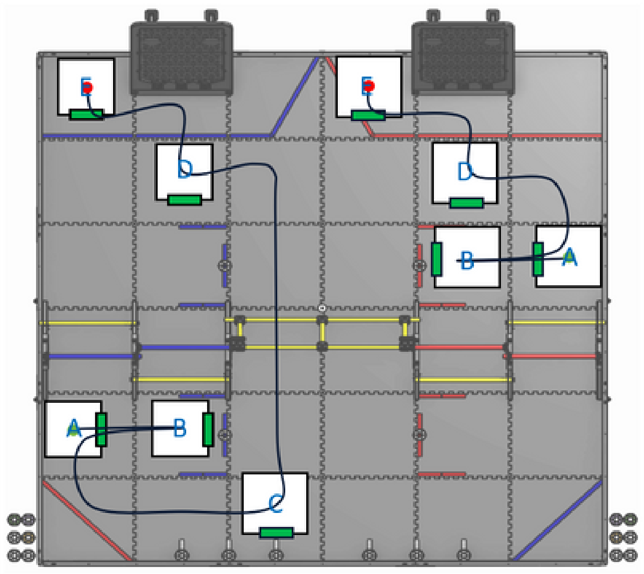
Smart Intake: Automatically intake exactly 2 pixels

1. Operator clicks A, enabling the auto assist and starting the intake.
2. Flash white LEDs!
3. If the sensor detects 2 pixels, then lock the pixels in place.
4. Reverse intake to eject any loose pixels we also picked up.

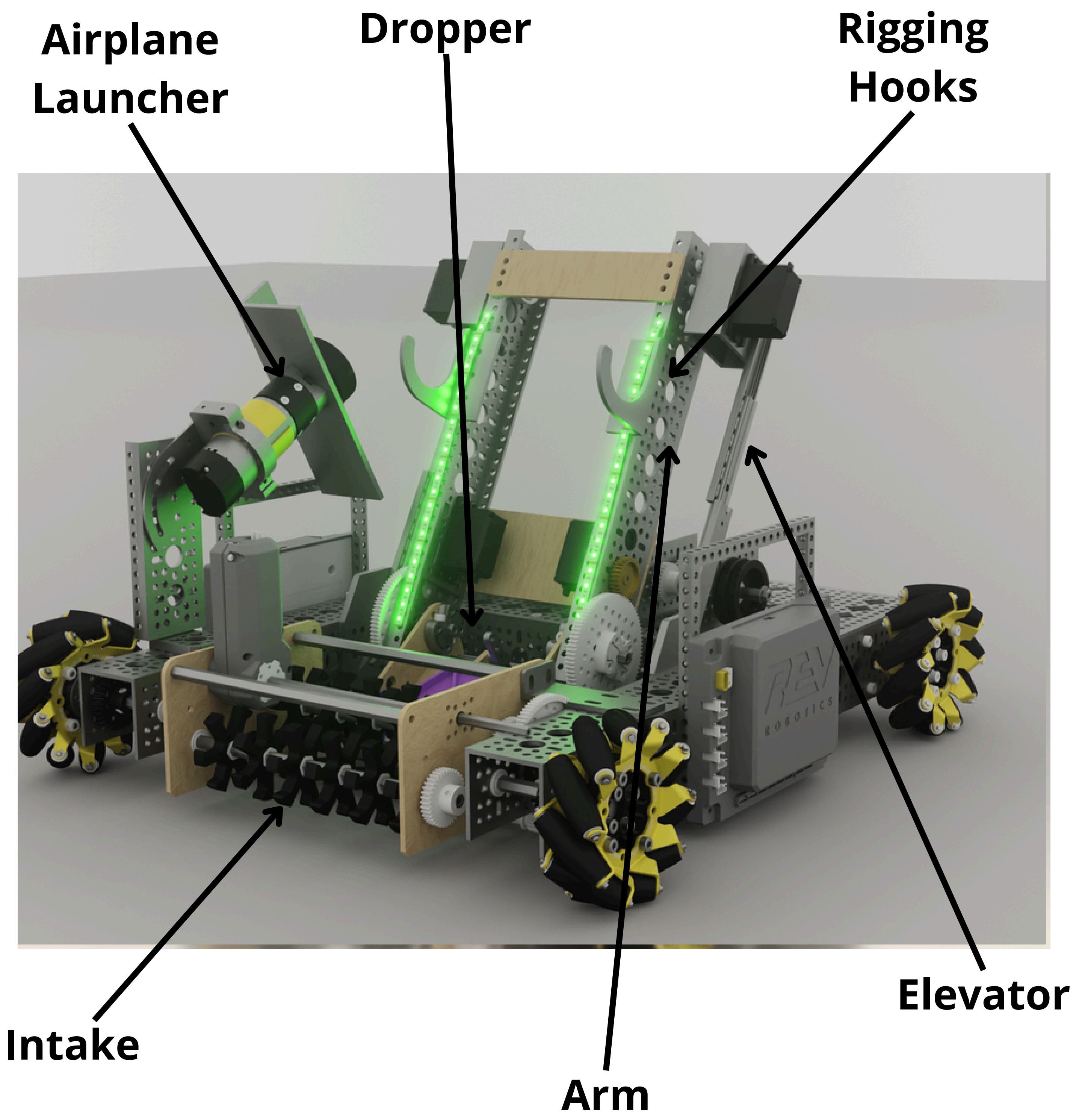
Autonomous Pathing

Autonomous:

1. Front webcam detects randomized position based on team prop location.
2. Drive to place purple pixel on correct spike mark.
3. If we are running 2+1 (automatically audience side) stop at pixel stack to get another pixel
4. Drive to lookout position to see AprilTags
5. Score a yellow pixel using April tags on correct spot
   a. If we are doing 2+1 score the white first away from the yellow pixel to avoid interference
6. Park in the backdrop



Robot Overview


Airplane Launcher
Dropper
Rigging Hooks
Intake
Arm
Elevator