Issaquah Robotics Society

2023 Engineering Notebook

Our Robot

Retracted State

Extended State

Table of Contents

Engineering Story

One of the goals of *FIRST* is to encourage students to apply the engineering process to various problems. Team 1318 believes that engineering is about developing and implementing a design both effectively and efficiently. We incorporate the engineering process into the design process for the entire robot, integrating engineering into entire meetings instead of just using this process for specific tasks.

We used the same adaptability and change required to be a good engineer to organize and run our team this year and accommodate the new FRC game challenge. To keep the design of the robot progressing, we organized into various sub-teams, each responsible for their own tasks, which were then integrated into the whole robot.

Our Product Cycle

To ensure efficient and effective engineering we follow a product cycle. This allows the IRS to continuously improve our robot while following a reliable process.

Brainstorming
1. Establish a strategy for the game.
2. Determine the most important tasks.
3. Conceive mechanisms that fulfill the tasks.

Design
1. Build CAD and physical mechanism prototypes.
2. Arrive at consensus on preferred designs.
3. Fine-tune and iterate prototype designs.

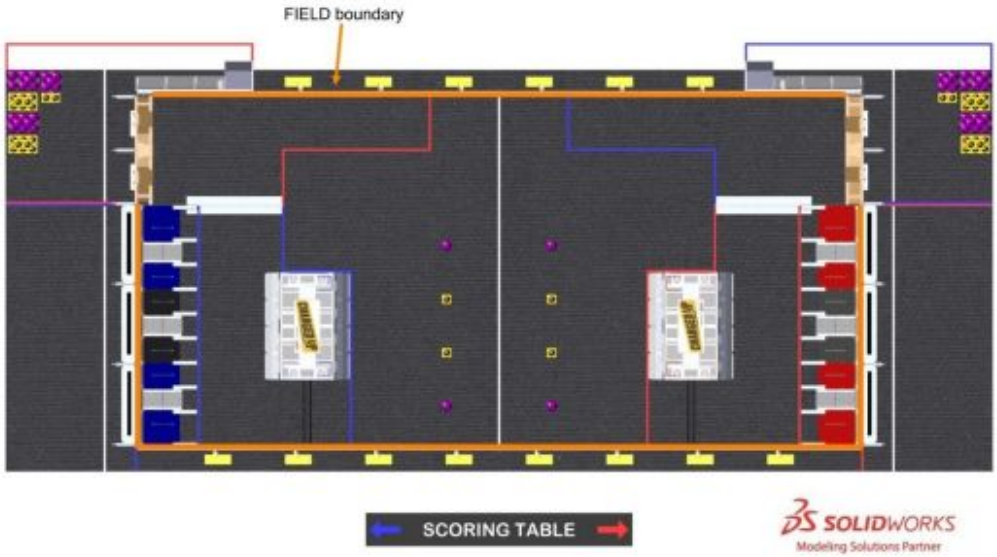
Build
1. Assemble the robot based off of prototype designs.
2. Fabricate & assemble mechanisms.
3. Build competition robot.

Robot Evaluation
1. Test each mechanism separately.
2. Run robot through integrated tasks.
3. Practice with competition robot at field.

Post-Competition Analysis
1. Analyze recruitment, training, and build season.
2. Evaluate student leadership models.
3. Review our design and engineering processes.

The Game

Summary



In CHARGED UP presented by Haas, two competing alliances are invited to process game pieces to bring energy to their community. Each alliance brings energy to their community by retrieving their game pieces from substations and scoring it into the grid. Human players provide the game pieces to the robots from the substations. In the final moments of each match, alliance robots race to dock or engage with their charge station!

Each match begins with a 15-second autonomous period, during which time alliance robots operate only on pre-programmed instructions to score points by:

- leaving their community,
- retrieving and scoring game pieces onto the grid,
- docking on or engaging with their charge station.

In the final 2 minutes and 15 seconds of the match, drivers take control of the robots and score points by:

- continuing to retrieve and score their game pieces onto the grid
- docking on or engaging with their charge station.

The alliance with the highest score at the end of the match wins!

The summary above was extracted from the 2023 FRC Game Manual

The Game

Tasks & Points

Task	Autonomous	Teleop
Mobility	3	-
Scored on bottom ROW	3	2
Scored on middle ROW	4	3
Scored on top ROW	6	5
Link	-	5
DOCKED and not ENGAGED	8 per ROBOT (1 ROBOT max)	6 per ROBOT
DOCKED and ENGAGED	12 per ROBOT (1 ROBOT max)	10 per ROBOT
PARK	-	2
SUSTAINABILITY BONUS	-	1 RP*
COOPERTITION BONUS	-	1 RP*
Tie	-	1 RP*
Win	-	2 RP*

*RP stands for Ranking Point

Strategy Development

Game Analysis - Design

Critical Robot Functions

- Autonomous
  - Reliably score 1 field piece on the GRID
  - Back up out of the COMMUNITY or balance on CHARGE STATION
- Teleop
  - Fast maneuverability while remaining balanced
- Intaking
  - Reliably pick up and grip field pieces from the ground and the SUBSTATION
- Scoring
  - Reliably place field pieces on all positions of the GRID
- Endgame - balancing
  - Reliably balance on the CHARGE STATION with 2 other ROBOTS

Strategy Development

Approach To Design

Team Prototyping Decisions

Main Arm CAD Model

Drivetrain Fabrication (electrical and mechanical mockup)

Intake Prototype Fabrication

Full Robot Design

Our robot began in prototyping. To achieve our final robot, we focused heavily on prototyping in CAD, then, with a semi-finalized design, we moved into physical prototyping and then tested our final robot mechanisms, making adjustments if necessary.

Task Identification

Team Prototyping

Prototype Evaluation

Focused Prototyping

Robot Mechanism

A summary of our prototyping process

Drive Train

Needs	Wants
Navigate different paths on field	Optimize maneuverability
Maintain stability	Able to move at high speeds
Able to climb CHARGE STATION ramp	

Possible Ideas

MK4 Swerve Module	MK4i Swerve Module
Fast, maneuverable	Fast, maneuverable
More exposed motors	Motors protected
Higher center of gravity	Lower center of gravity
Lighter	Heavier
Separate mounting needed	Integrated with frame

Notables

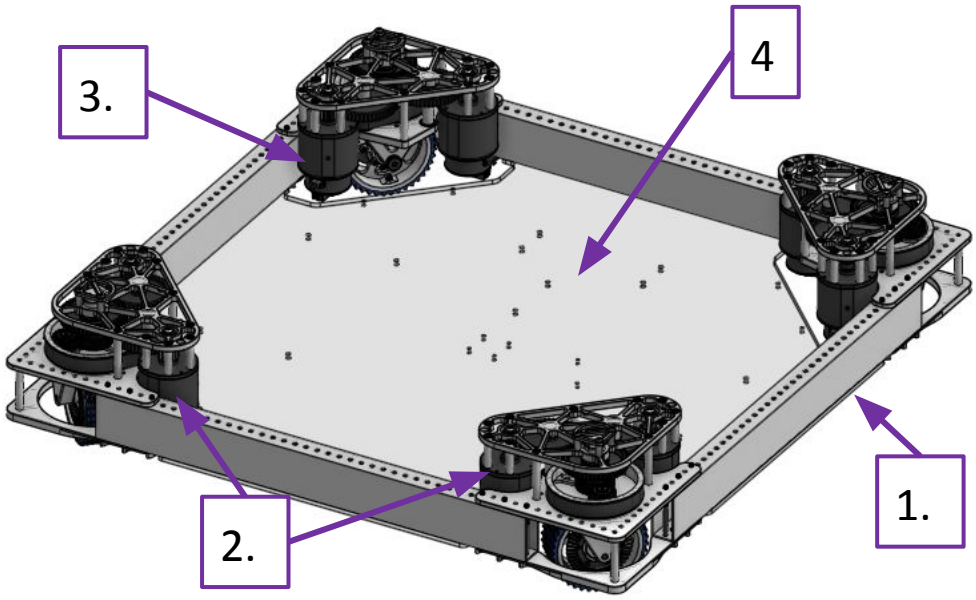
- The drive train is powered by eight Falcon 500 motors
- The drivetrain has PID control for position and velocity control, as well as a PID-based brake mode
- Frame dimensions: 28” x 28”

Drive Train

Swerve Drive Train Design

The drive train this year is a 4-wheel swerve drive. It has 8 Falcon 500 motors coupled with a 6.75 : 1 gear ratio (about 16.3 feet/second max) within a SDS Swerve Drive Module.

The Final Product



Our modified Rev MaxTube Frame

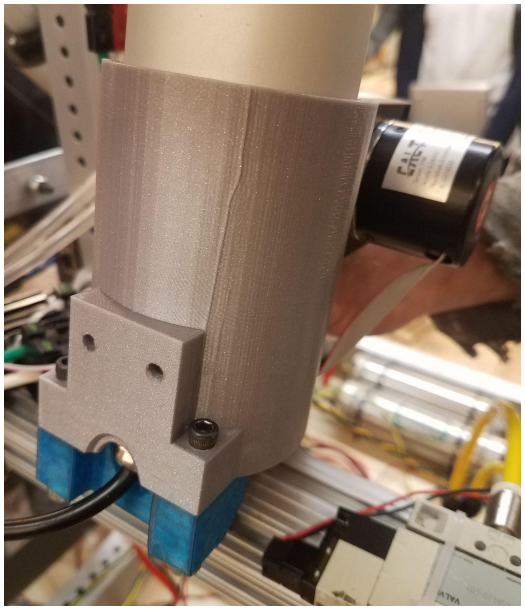
Parts:

1. VersaFrame 2x1 1/16” wall thickness
2. 6.75 : 1 SDS Swerve Modules (4)
3. Falcon 500 motors (8)
4. Polycarbonate belly pan

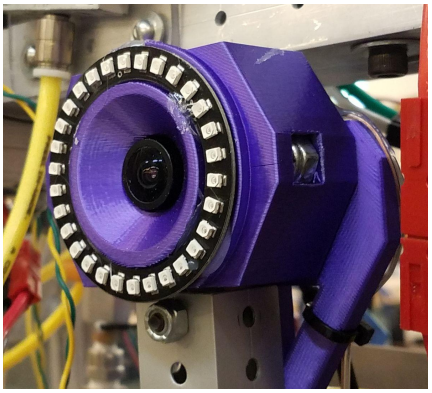
3D Printing

Cone Flipper Pivot

Constant Force Spring Rollers



String Encoder Mount



Camera Housing

Hardware

We designed and printed many of the pieces of hardware used on our robot this year. The rollers for the constant force springs on the main arm, the mounting sleeves for the linear actuators' string encoders, the pivoting parts of the cone flipper arms, polycord pulleys, and the housing for the mk2 camera are just some of the custom parts we've printed.

End Effector

Needs	Wants
Grab both cone and cube	Intaking with little driver accuracy
Limited jamming and easy way to remove jams	Outtake preserves orientation of cone and cube
Easy way to accomplish ground and PORTAL pickups	Does not require precise driver movements

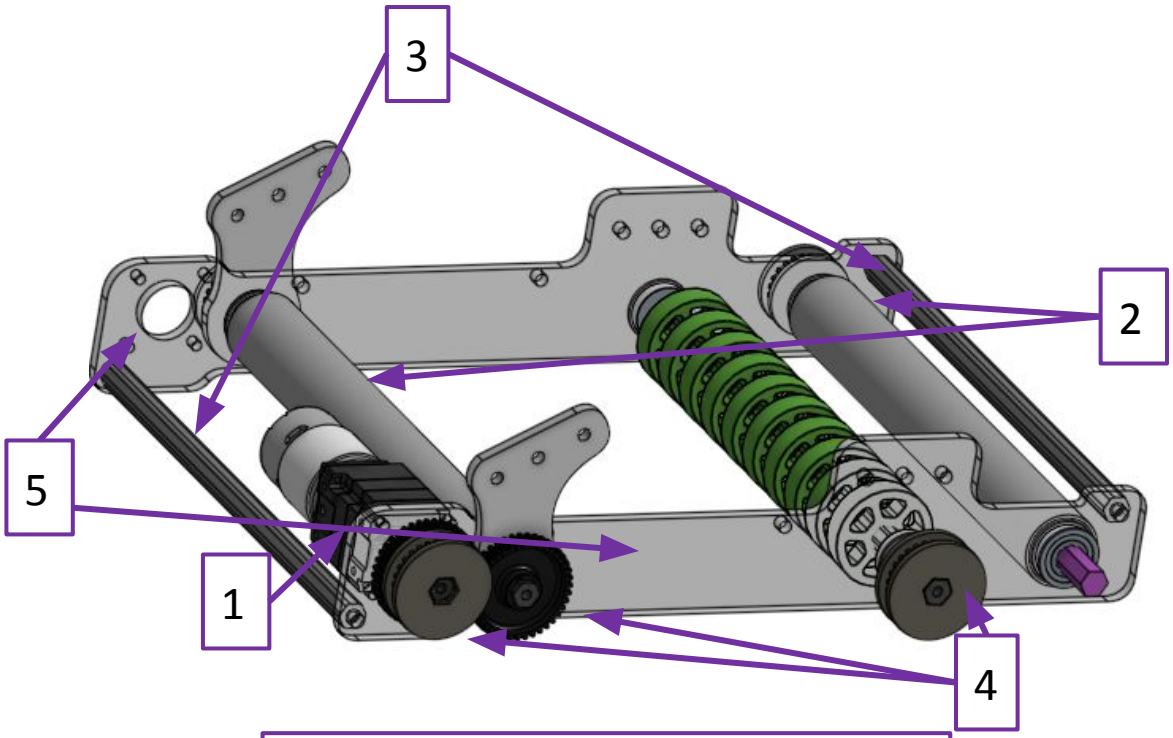
Possible Ideas

Rollers + belts	Rotating Claw + Rollers
Can hold both CONE and CUBE	Can hold both CONE and CUBE
Enables faster intake	Preserves CONE and CUBE orientation
Can orient field pieces to be almost vertical when scoring	Can only pick up upright CONES

Notables

- Main polycarbonate pieces machined in house
- Designed and 3D printed pulleys
- Manufactured aluminium pivot plates

End Effector



Our end effector assembly

Features:

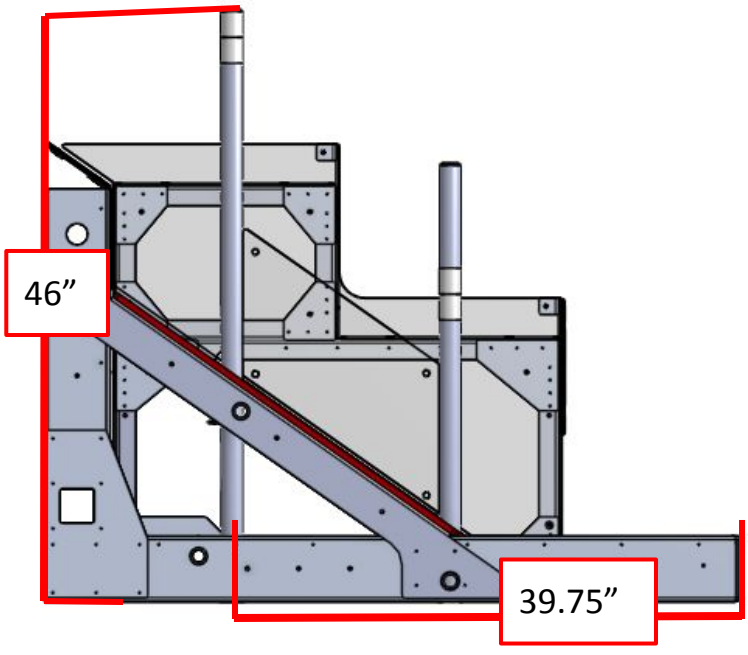
- Intake versatility: capable of holding both types of game pieces
- Single motor powers both rollers with polycord
- Compliant rollers allow for better grip on CONE and CUBE

Parts

1. Redline Motor
2. Rubber Rollers (3)
3. Hexshaft
4. Toothed pulleys
5. Polycarbonate Side Panels

Main Arm

Needs	Wants
Can reach all NODES on GRID	Maneuverable
Can reach game pieces on ground and PORTAL	Able to move quickly
Can be stored completely inside frame perimeter	Light for lowered center of gravity
Doesn't extend outside the frame perimeter in more than one direction	Robust, durable design
Precisely control position of end effector	



Main Arm

Design Process: Possible Prototypes

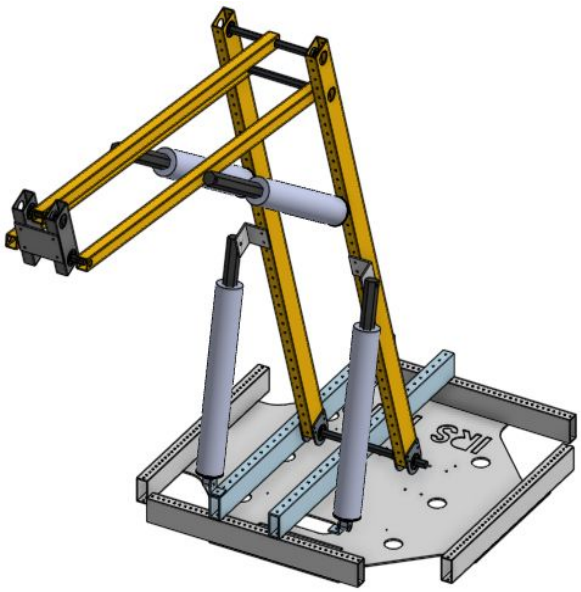
Elevator + Linkage

Able to reach ground and PORTAL

Able to reach all NODES of GRID

Mechanisms:

1. Single carriage elevator
2. Linkage
3. Pivoting base



Linear Actuators + Linkage

Able to reach ground and PORTAL

Able to reach all NODES of GRID

Mechanisms:

1. Linear Actuators (4)
2. Linkage
3. Pivoting base

Main Arm

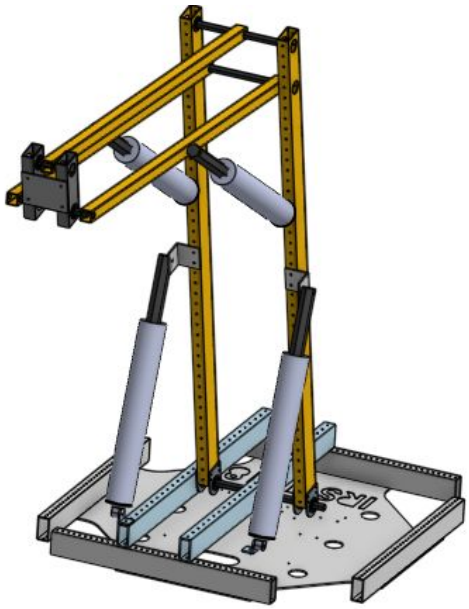
Evolution of Linkage Design

Original Design

Design features:
- 4 Linear Actuators
- 1 upper linkage

Unable to reach full 154 degree range (90 degree max)

Strength of mechanism not sufficient at full extent



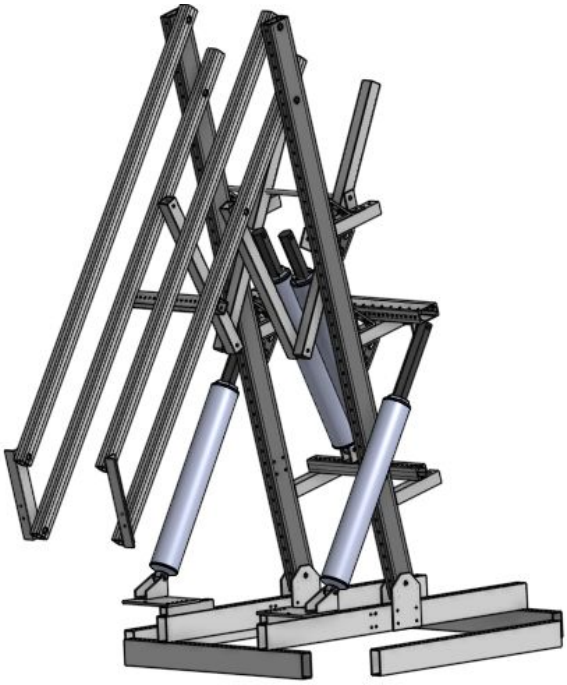
Transfer Linkage Design

Design features:
- 4 Linear Actuators
- 2 upper linkages

Able to reach full 154 degree extension

Arm angle is almost exactly proportional to actuator extension

Allows for optimal constant force spring mounting



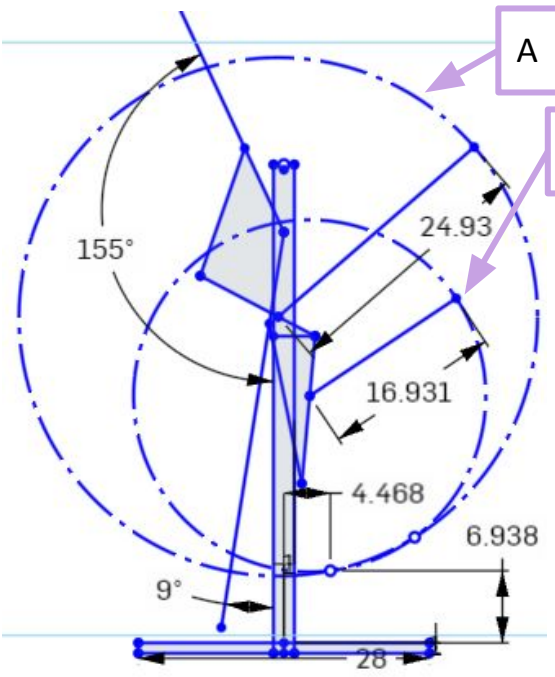
Main Arm

Linkage Geometry

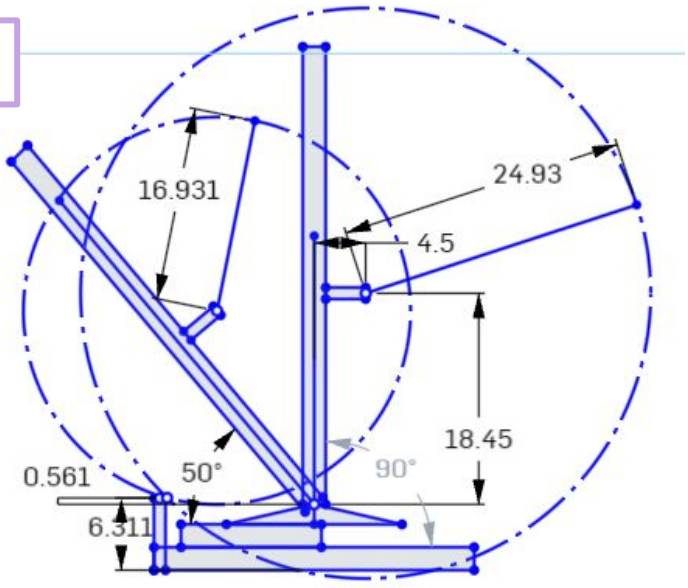
We used 2D diagrams to find the lengths and pivot point locations of the 4-bar linkages and the linear actuators.

Intersection Circles for Locating Linear Actuator Pivots
The dashed circles represent all possible locations of the linear actuators' end pivot points. The larger Circle (A) represents the actuator at its furthest extension and the smaller circle (B) represents the actuator in its retracted state. The intersection of the two circles represents the optimal actuator pivot in that specific design

The arms are modeled in both the fully extended and fully retracted states so we can verify the actuator pivot points will work for both states.



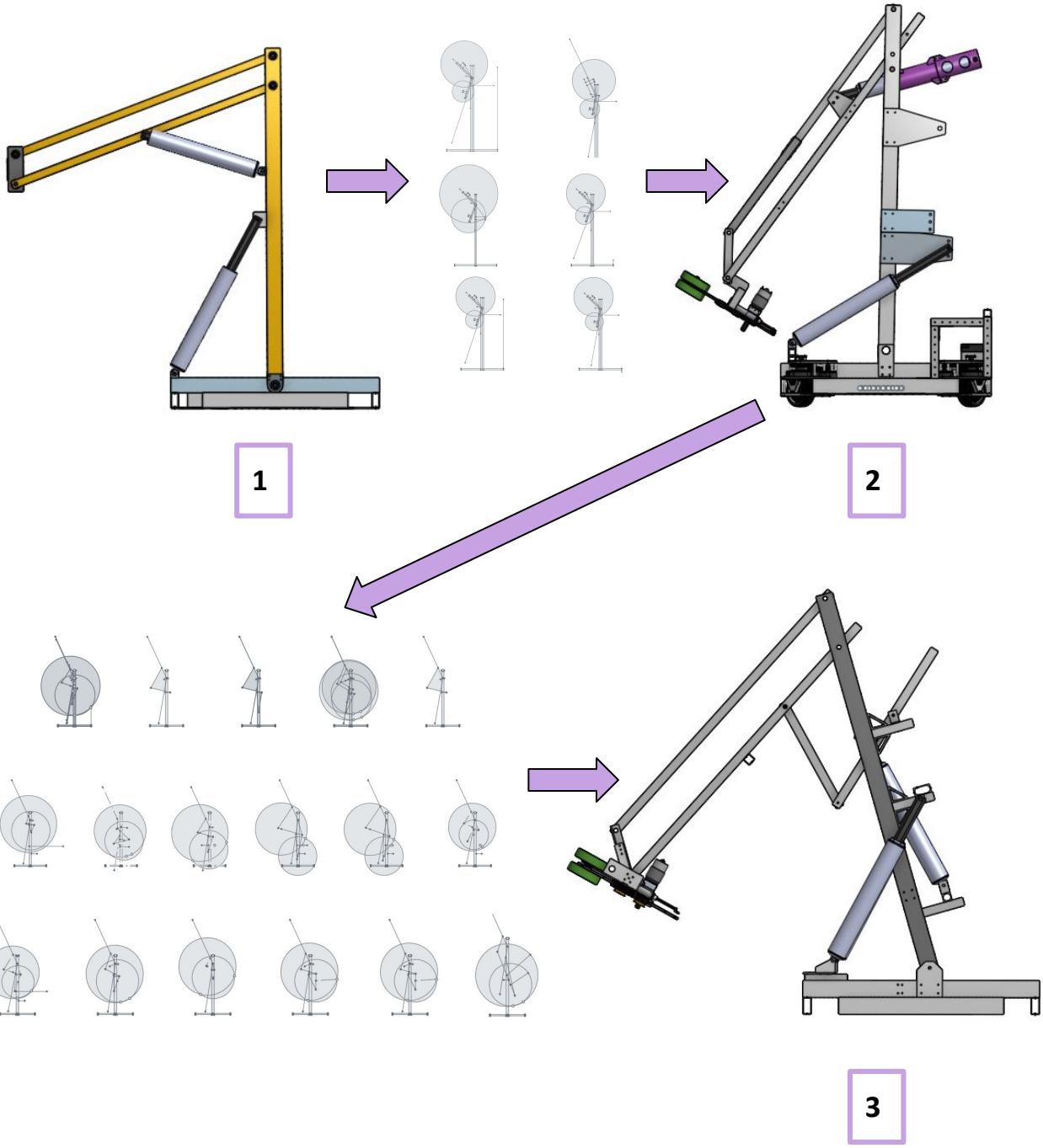
Early Upper Linkage Geometry



Lower Arm Geometry

Main Arm

Linkage Evolution

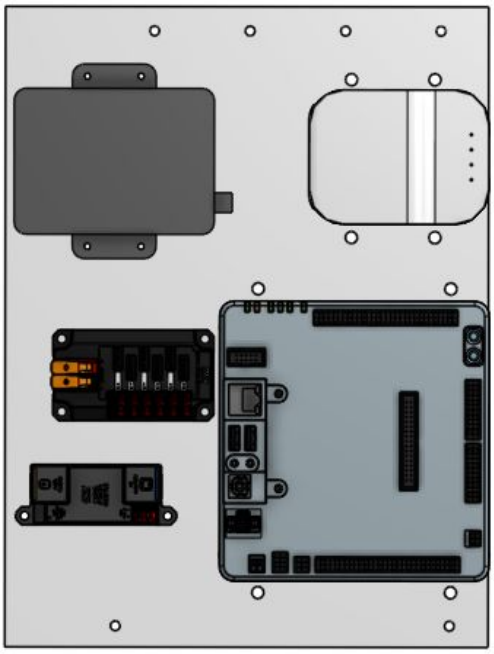


1

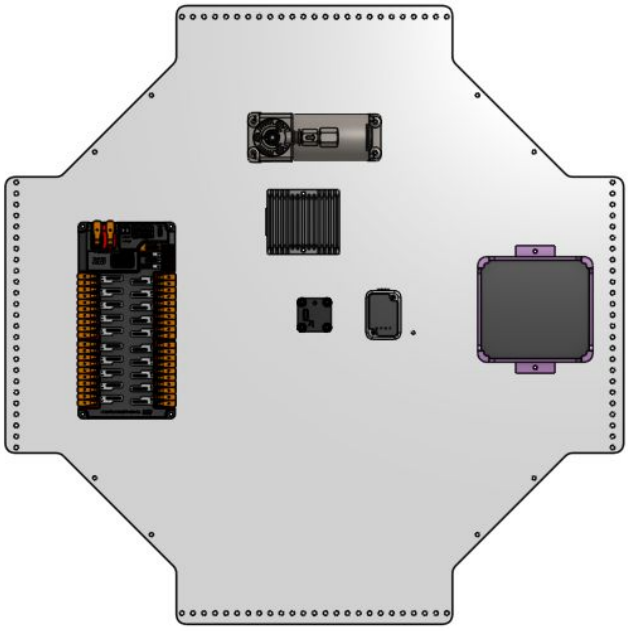
2

3

Electronics & Pneumatics



Right side electrical panel



Belly pan electronics layout

Electronics Notables:

- 2 main electrical boards
- Belly pan area: dedicated/planned space for electronic systems
- Beelink computer for vision
- 5 motor controllers

Pneumatic Notables

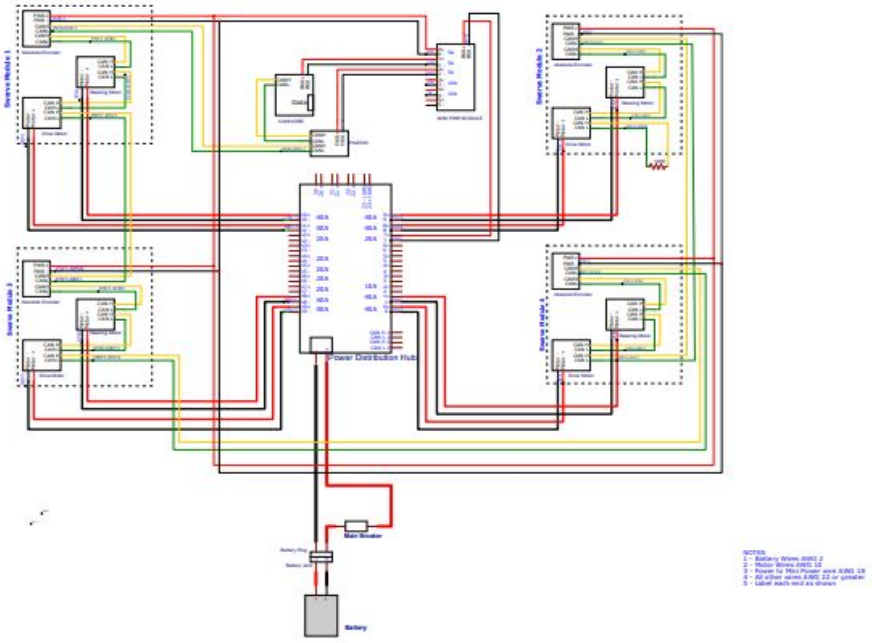
- 3 solenoids
- 2 Clippard Minimatic metal air tanks
- The end effector uses two 2” stroke pistons
- The cone flippers use two 6” stroke pistons

Electronics & Pneumatics

Left board	Right Board
Our left board contains 2 solenoids, the PH, an analog sensor, a digital sensor, and the shut-off valve.	The right board contains the RoboRIO, an MPM, the radio, the RPM, and the Ethernet Switch.

Belly pan

PDH mounted on the left, Beelink computer mounted on the right, Pigeon (IMU) mounted in the center, compressor mounted in the back, air tanks mounted in the front.



Main Power and Drive Train Schematic

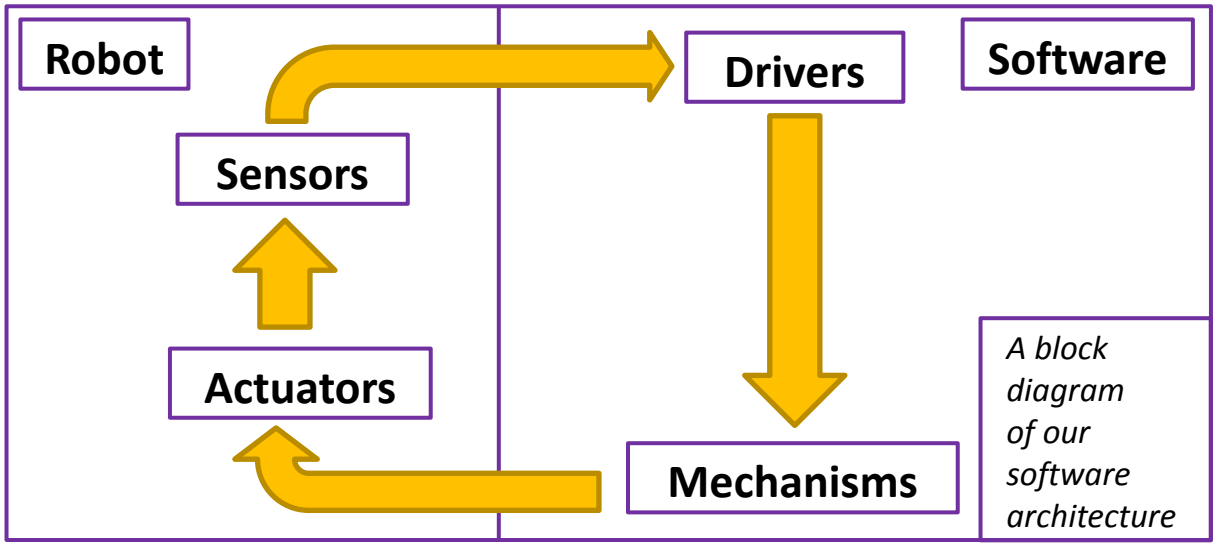
Software Architecture

Notables

- ✓ Macros
- ✓ PID feedback control
- ✓ Automated leveling
- ✓ State machines
- ✓ Vision System

Dual input controllers with motion control

Our team uses Java for robot programming and utilizes Github as our source code repository.



A block diagram of our software architecture

Vision

Vision allows us to detect the retro reflective tape on GRID nodes to align ourselves automatically. This year, we are using a Beelink for vision instead of the RoboRIO, using parallel processing to run the image processing from OpenCV at the same time as the robot control system. Additionally, we added April Tag vision tracking in order to make game piece placement and retrieval automated.

Software Architecture

Proportional Integral Derivative (PID)

- Velocity PID control allows us to regulate the velocity of the drivetrain to deal with inaccurate movement.
- Positional PID control gives us the ability to move our drivetrain to a specific position and maintain that position. Also, with the string encoders, positional PID allows us to determine the exact length of the linear actuators, enabling more precise control.

Smart Dashboard

Smart Dashboard receives information from the robot and displays selected data. This allows us to identify and/or solve problems more quickly.

CHARGE STATION Leveling

We use the pitch from the Pigeon IMU to find the distance to move while on the CHARGE STATION during AUTO.

Swerve Drive

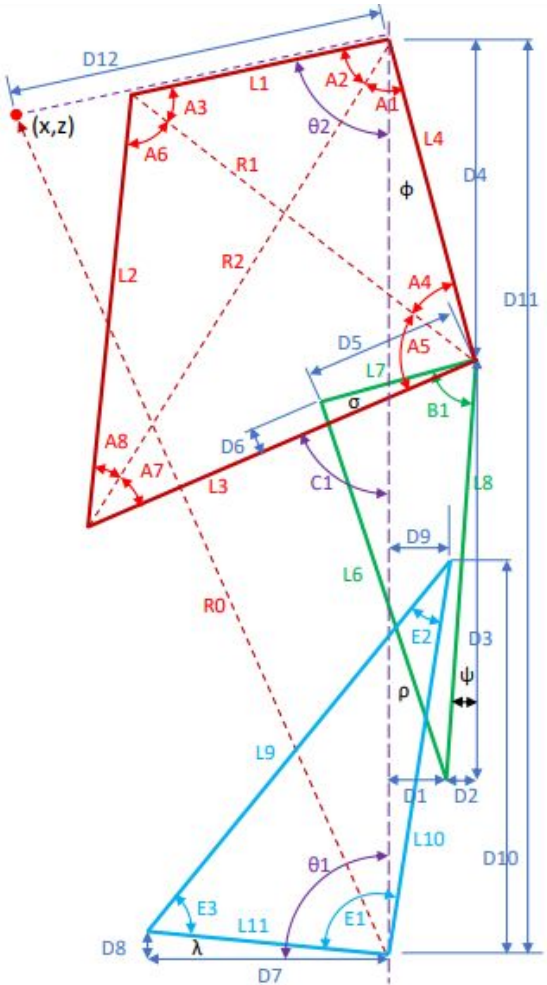
The swerve drive is controlled by kinematics and code we wrote and derived from scratch. We used linear algebra to calculate the angle and speed each of the modules needed to go to drive and rotate at a certain speed.
Using our own code instead of an external library allows us to have more control over possible maneuvers.

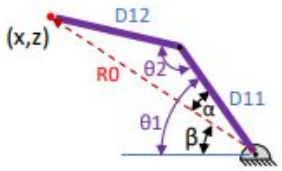
Software Architecture

Main Arm Control

We use positional PID to control the arms motion to avoid overshooting

We used Inverse and Forward Kinematics to determine the extension of the linear actuators required to move the arm to a desired point in space and to limit the arm's movement to within legal limits. We also used set point tables to move the end effector to a certain point in space by using the actuator length value keyed to those coordinates.





Inverse Kinematics

$R_0^2 = x^2 + z^2$

$\theta 2 = \pm \text{acos}\left[\left(D_{11}^2 + D_{12}^2 - R_0^2 \right)/(2*D_{11}*D_{12}) \right]$

$\beta = \text{atan2}(z,x)$

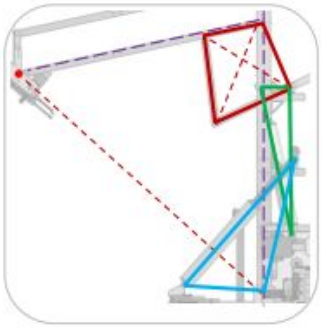
$\alpha = \pm \text{acos}\left[\left(D_{11}^2 + R_0^2 - D_{12}^2 \right)/(2*D_{11}*R_0) \right]$

$\theta 1 = \alpha + \beta$

Forward Kinematics

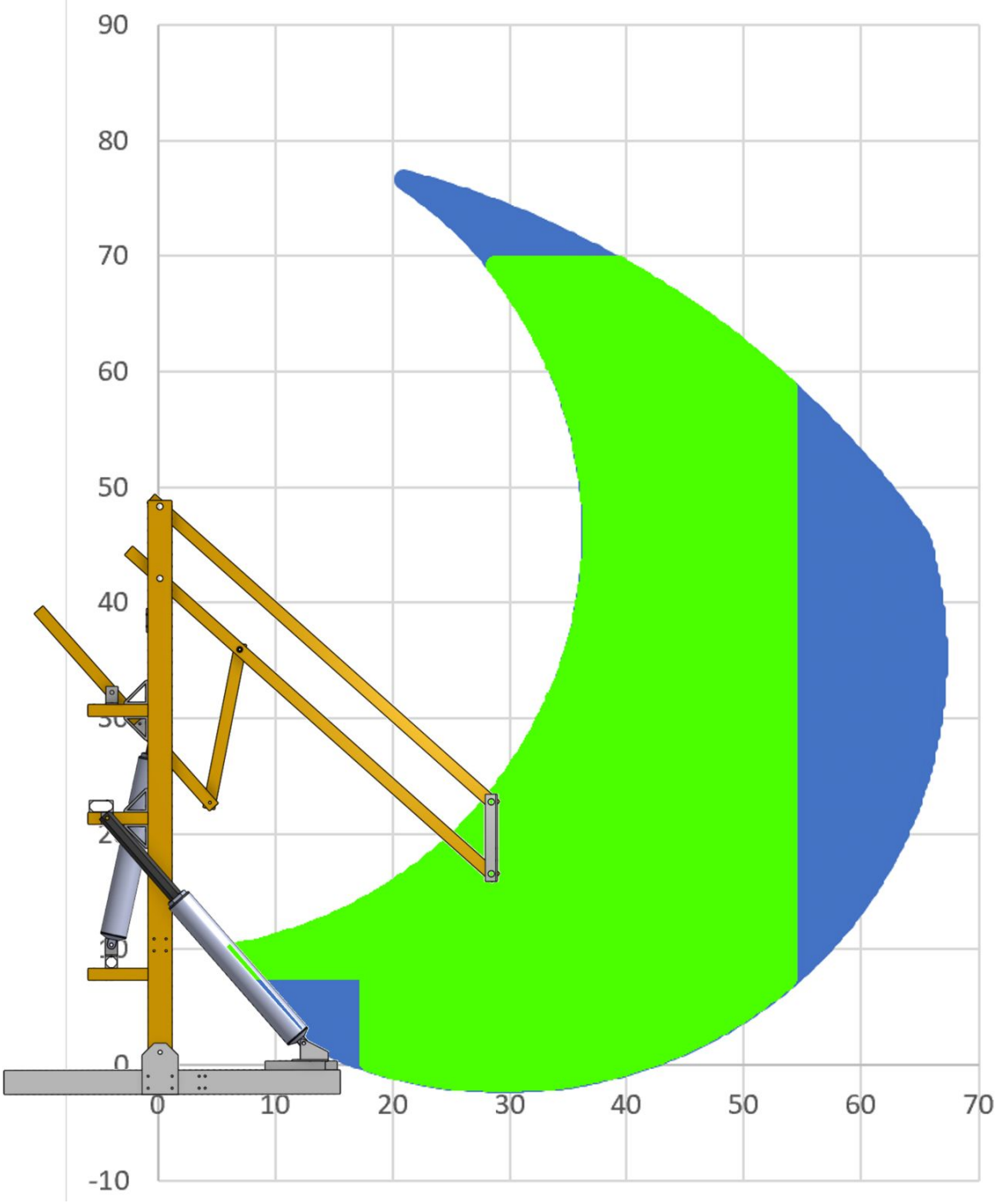
$x = D11*\cos(\theta 1) + D12*\cos(\theta 1 - 180^\circ + \theta 2)$

$z = D11*\sin(\theta 1) + D12*\sin(\theta 1 - 180^\circ + \theta 2)$



Software Architecture

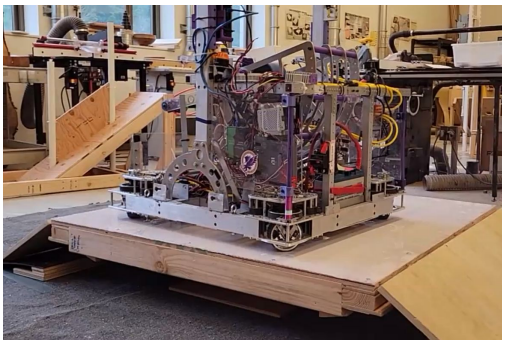
A plot of the main arm's total positional reach used to evaluate the arm geometry's viability and introduce programmed limits. Green shows the legal extension limits.



Charge Station Balancing

Initial Plan

Initially we thought that braking when the pitch was at zero would be enough to balance the robot on the charge station. However, there was delay between when the CHARGE STATION began tipping and when the robot's center of mass was centered.



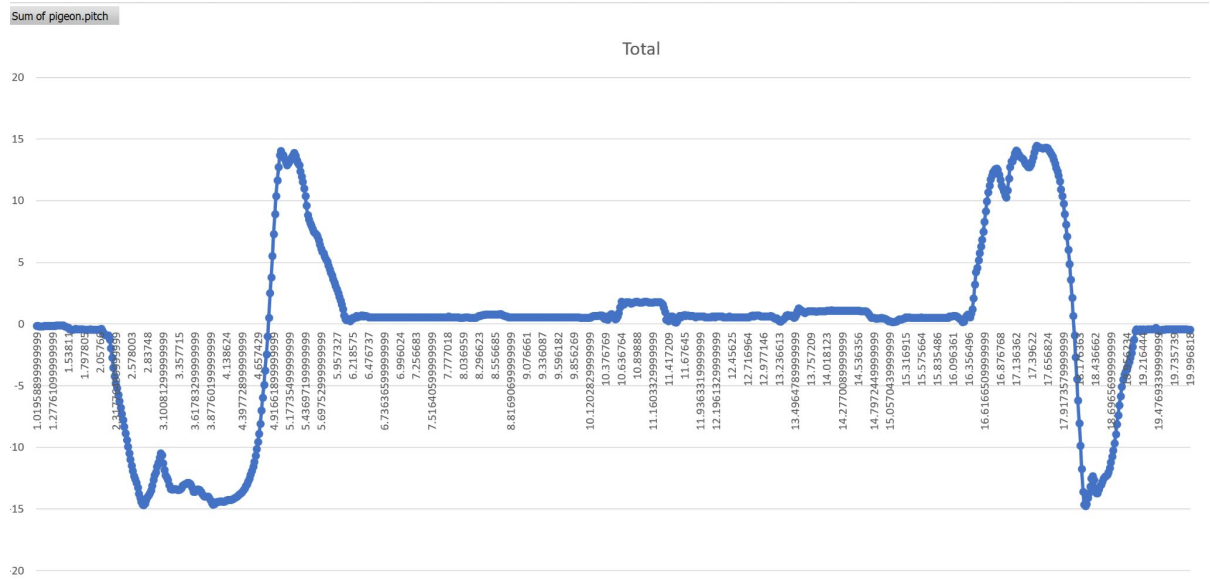
Testing autonomous balancing capabilities on our practice robot

Problem Solving Steps

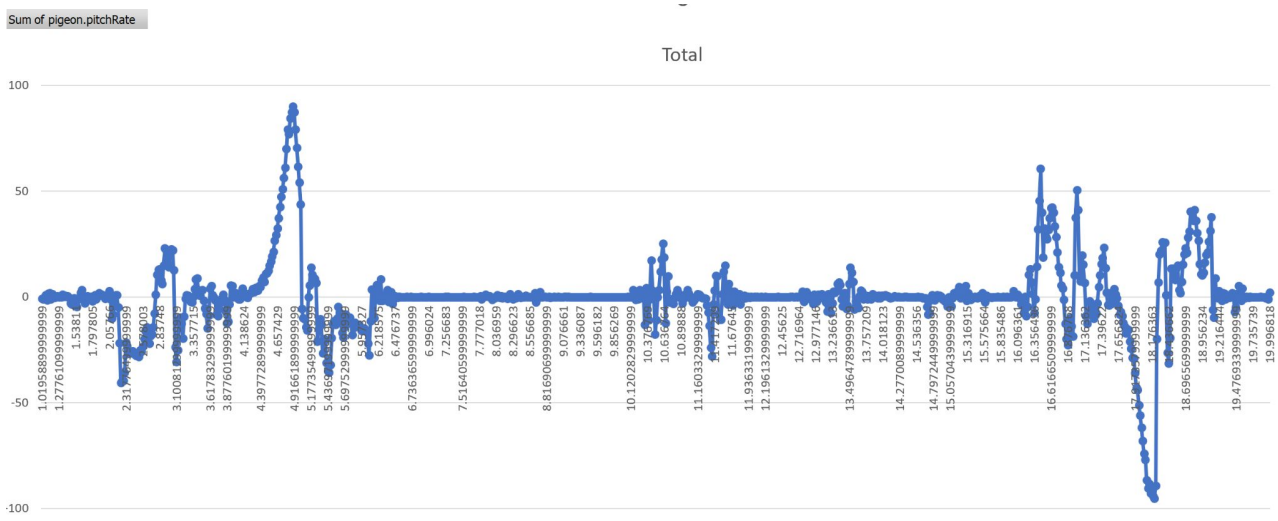
We decided to approach the charge station at a slower speed, then detect when the pitch was changing at a certain rate, meaning that the charge station was tipping the other way. Using a rate would require less tuning and fewer moving parts in the code.

Raw pitch rate from the IMU was experimented with but ultimately not viable because of the amount of statistical noise. We decided to use a change in pitch of 2.0 over 0.5 seconds as a balancing threshold through experimentation.

Charge Station IMU Data



Pitch (integrated from pitch rate) when robot goes forwards then backwards over whole charge station



Pitch rate when robot goes forwards then backwards over whole charge station

Scouting Network

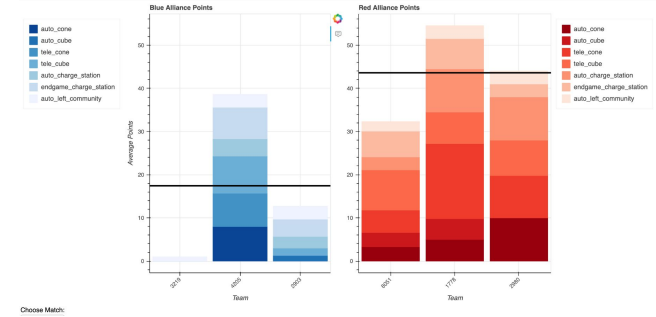
About

Our scouting network allows us to collect large amounts of data on the abilities of other robots. We then use this data in making better game strategies as well as for choosing teams during alliance selection.

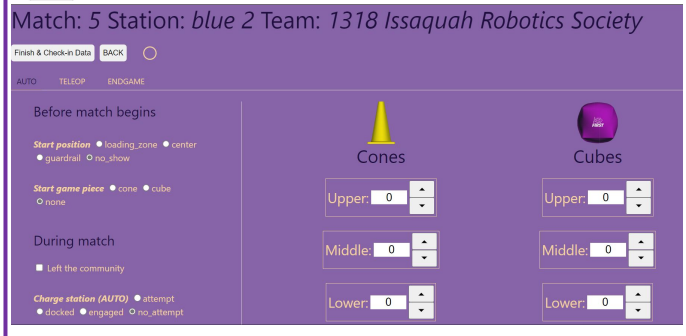
Programming

We have developed a web-based scouting platform that runs on a Flask WSGI server. Our user interface is coded using HTML, JavaScript, and CSS. We store scouting data in a SQL database through a Python server. We also use Bokeh, a python library, to create interactive visualizations in order to view and analyze our data.

Data Analysis



Interactive visualization generated through the Bokeh package



2023 UI Design for the Autonomous page of the Scouting System

Extras

Fall Training

Before kickoff day students and mentors prepared for the build season ahead. Classes included electronics, data analysis, programming, mechanical systems, and CAD.

Machining

We made many custom parts for our robot this year, out of both aluminum and polycarbonate (end effector structure, belly pan, electrical side panels, etc.). These were both designed and machined by our team with our CNC router and CNC mill.

Field Pieces

To organize for the game without a field, we built various field pieces out of wood to practice with a more accurate representation of the field.

CAD

We use CAD to design parts, assemblies, and geometries to make a completed robot model before its fabrication.

Issaquah Robotics Society

Thank you to all of our sponsors!

BOEING

Microsoft

ISSAQUAH SCHOOLS FOUNDATION

BECU

Washington Office of Superintendent of PUBLIC INSTRUCTION

THE PRINTING COMPANY

First Tech federal credit union

RYVER

PACCAR

Signarama

ISSAQUAH SCHOOL DISTRICT 411