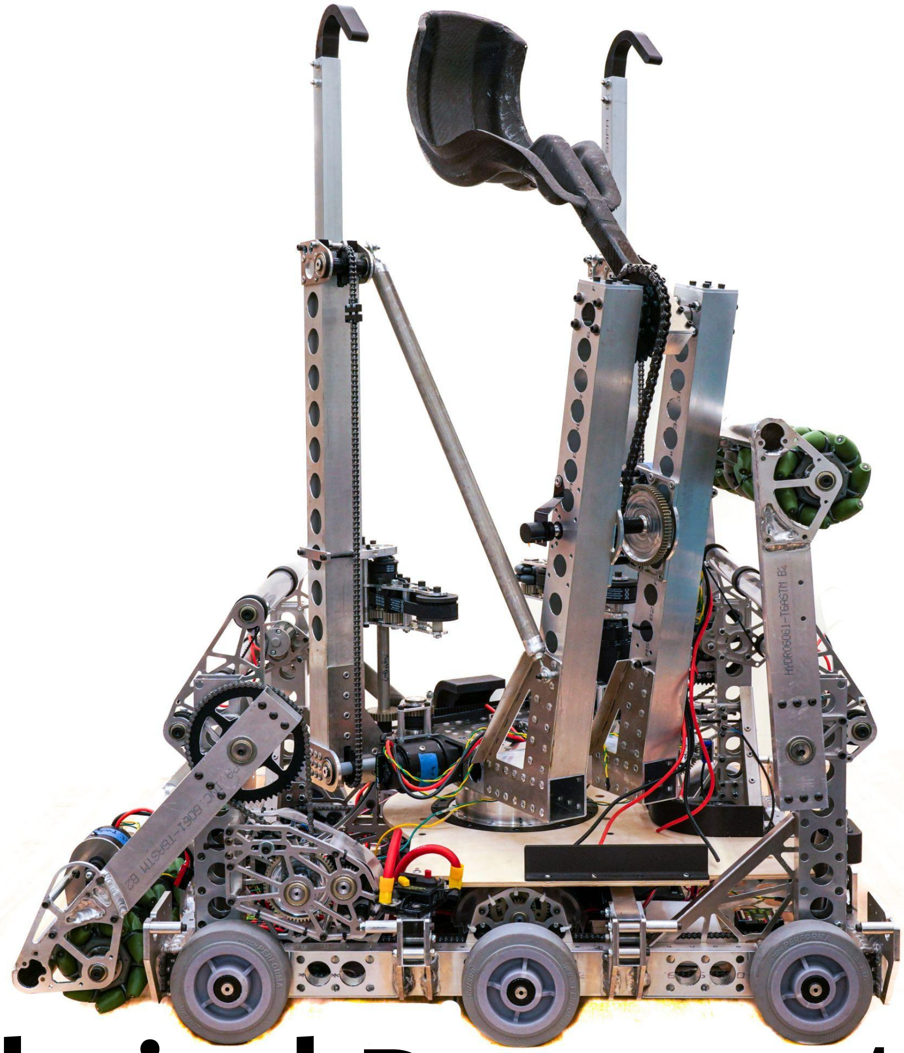
SPARTAN ROBOTICS

FRC 971



Technical Documentation 2022

971 Robot Overview

Flippy Arms:

➔ Feeds cargo from intake to catapult

➔ Holds cargo in place in catapult, releases ball when shooting

Climber:

➔ Lifts robot to the high climb position

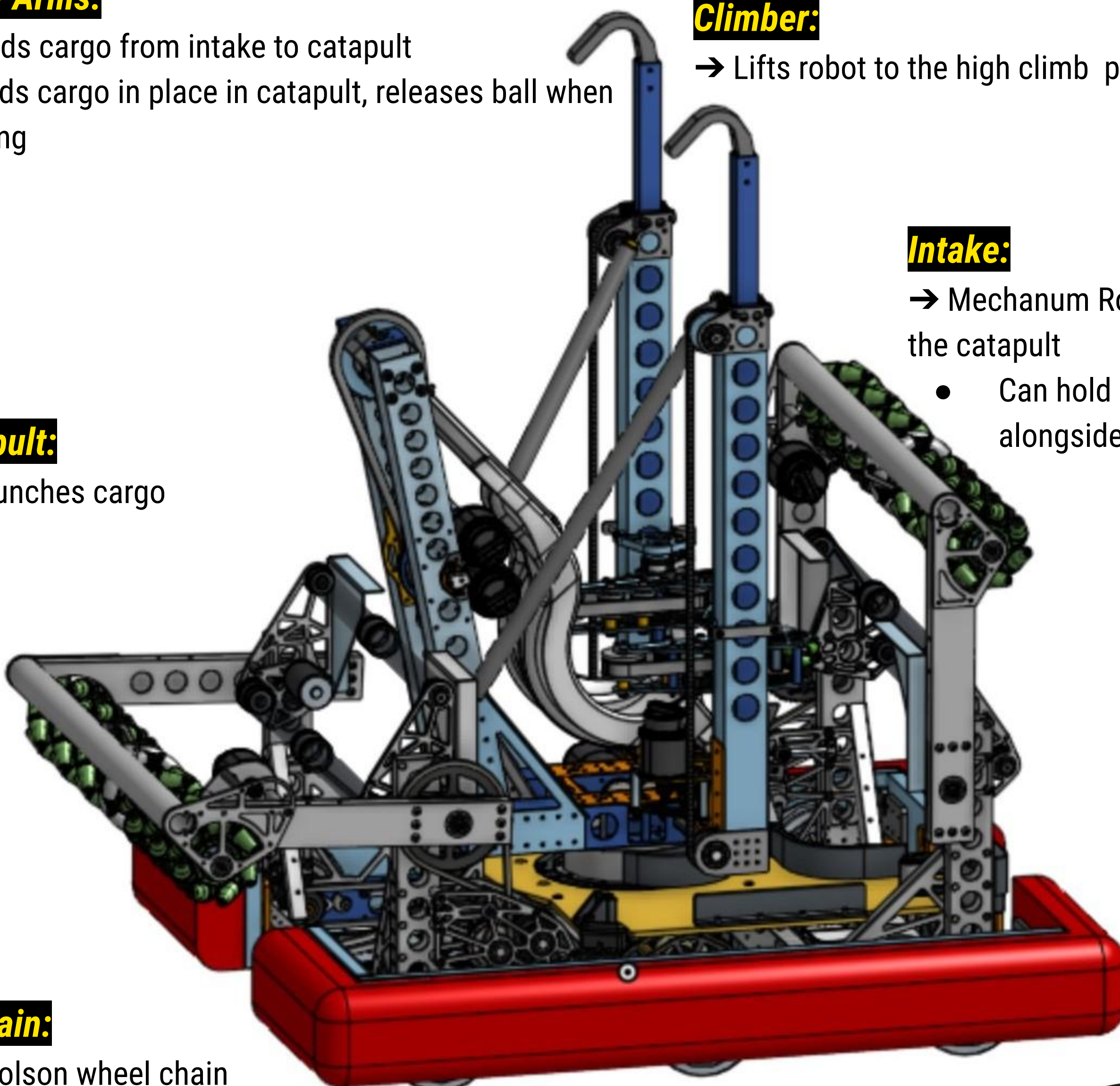
Intake:

➔ Mechanum Rollers feed balls to the catapult

- Can hold cargo in place alongside the flippy arms

Catapult:

➔ Launches cargo



Drivetrain:

➔ Six colson wheel chain driven West Coast Drive

➔ Each side is driven by a central gearbox

Turret:

➔ Turns catapult and climber to face correct direction

971 ON DESIGN

Proposed Complexity Index

Scale each DOF from 0-3 on level of effort (says the most complex DOF takes 3X the effort as the simplest DOF)

- 0: Pure duplicate of something else (account for as combined unit)
- 1: Simple DOF, High leverage of know design and COTS components. Little MFG. EG simple versa planetary DOF or simple pneumatic actuator. Or passive DOF
- 2: Average DOF. Minimum creation/invention required, reasonable interdependencies to other DOF and only moderate space constraints. Average amount of machining required.
- 3: Complex DOF: High level of prototyping required get right and/or high level of design analysis required to get right. Complex interdependencies to other DOF and/or tight space constraint. Large amount of machining

Have a 1, 1.5, 2X multiplier based on how likely to need re-work/revision

Sum up

Design Process

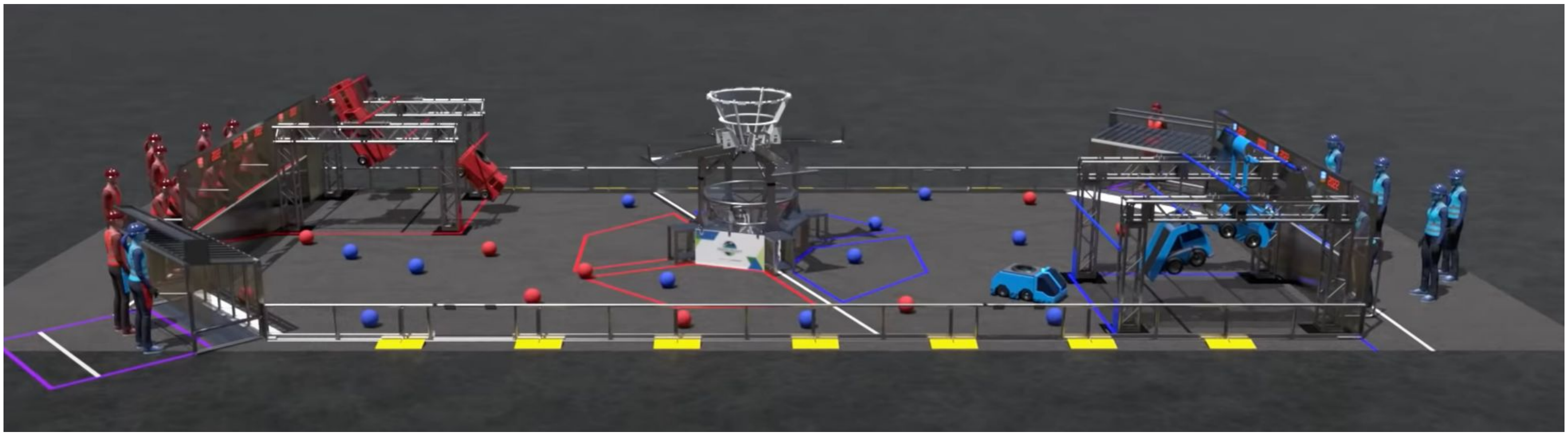
Subsystem DOF	Complexity for 2022
Drivetrain	2
Intake Arm	2
Intake Wheel	1
Intake Transfer	1
Turret	3
Flippy Arms	2
Catapult	3
Climber	2
Total	**16**

Results

Year	DOF	Complexity Index
2014	12	16.5
2016	20	26.5
2016 initial scope	14	19.5
2017	10	28.5
2018	16	29.5
2019	11	20.5
2020	12	29.5
2022		Target ≤ 16

In this year's design, we decided to aim for a less complex robot than previous years. As COVID impacted the experience we had in our team, we needed to limit our works compared to previous years. This limit was needed in order to increase our efficiency with our design process and to accept and understand that we will need to simplify our designs to achieve our goals in time. While difficult, and had a tight time completion in which we were able to complete a promising robot just in time for our first competition in SFR.

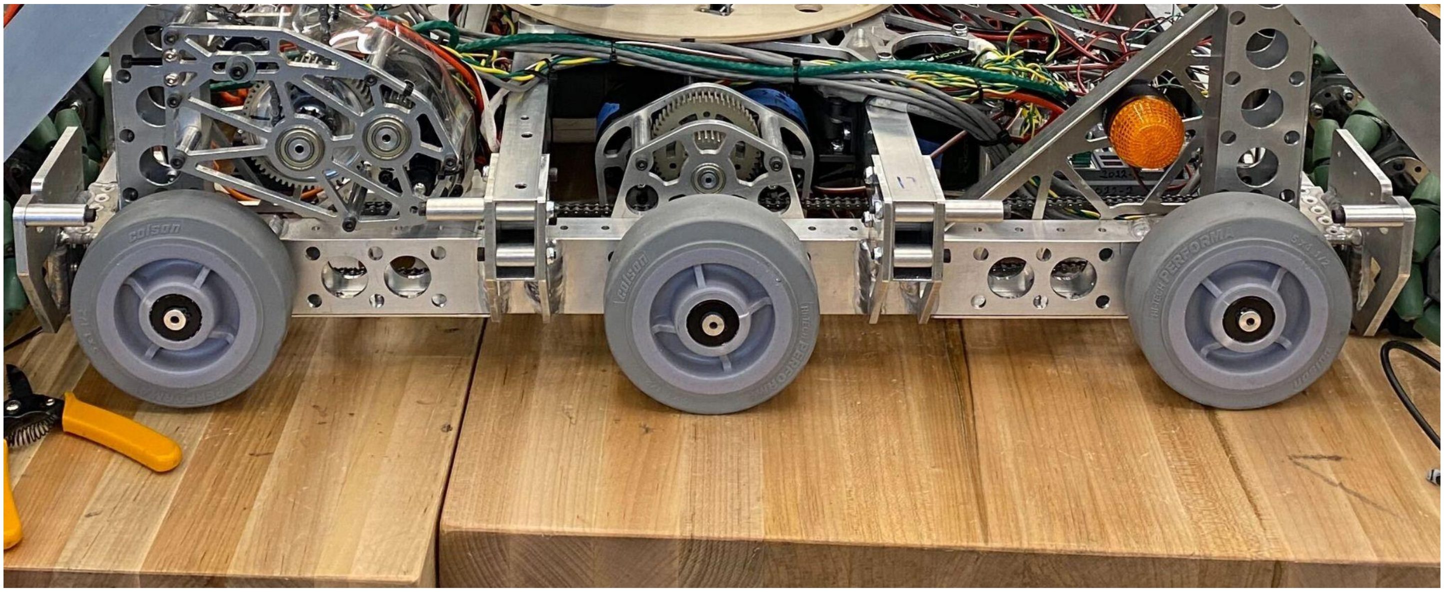
Robot Function Objectives



- Build a simple robot that performs at a high level
- Reliable drivetrain
- Intake balls quickly (2 intakes on opposite sides)
- Store 2 balls
- Shoot accurately and quickly from anywhere on the field
- Climb to the mid rung originally now added a climber to higher rungs

Subsystems

Drivetrain

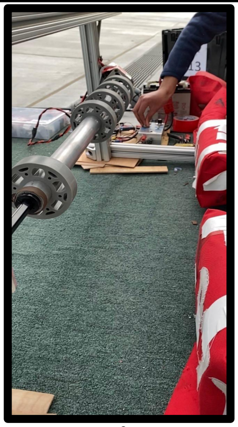


The Drivetrain is a six wheel West Coast Drive that is designed for simplicity and robustness. We chose colson wheels because we wanted to have low maintenance but to still have good performance. Additionally, the gear ratio of the drivetrain was designed for fast acceleration for shorter distances and overcoming defense.

- Two gearboxes, each powered by two falcons
- Drivetrain gearbox ratio: 8:70

Subsystems

Intake



Iteration 1

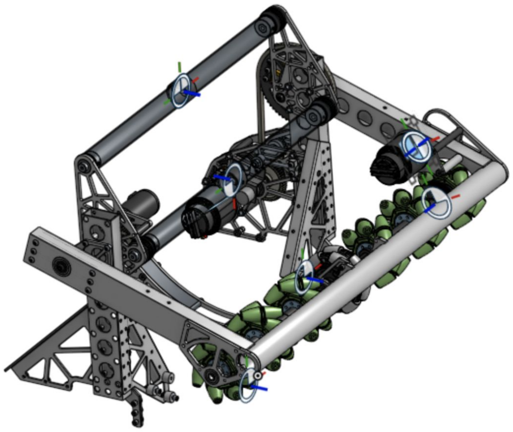
- Our first iteration consisted of flex wheel rollers in order to counter the stiffness of the ball as well as grip it better.

Iteration 2

- Our second iteration was heavily influenced by our 2016 intake design in which there are mecanum wheels which center the ball straight in the robot and has a chain tensioned arm to adapt to the shape of the ball

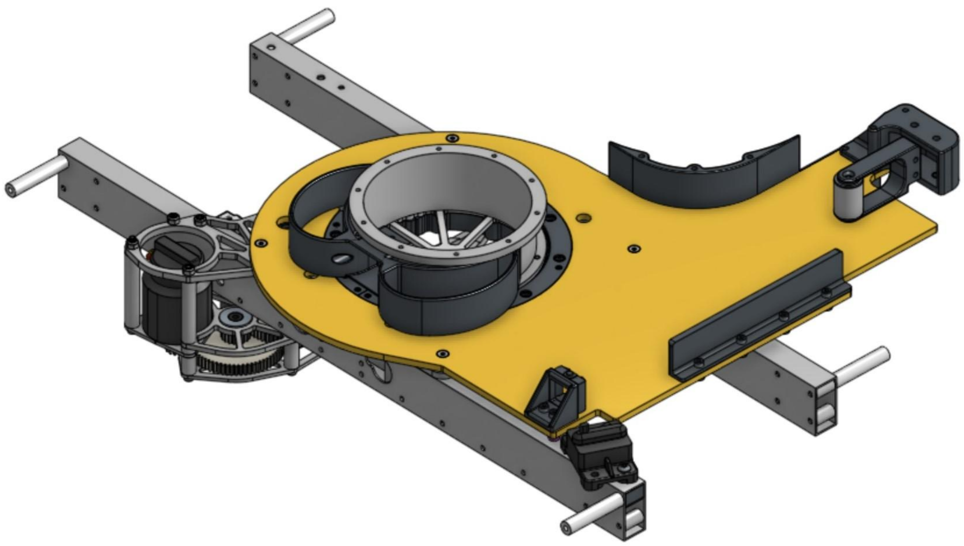
Final Iteration:

Our final design is similar to our 2016 intake design in which we used the mecanum roller function as well as the chain, arm feature. However, we added two new rollers in the upper portion of the intake in order to guide the game pieces to our catapult. This gives us extra support as well as a guide to the next subsystem.



Subsystems

Turret

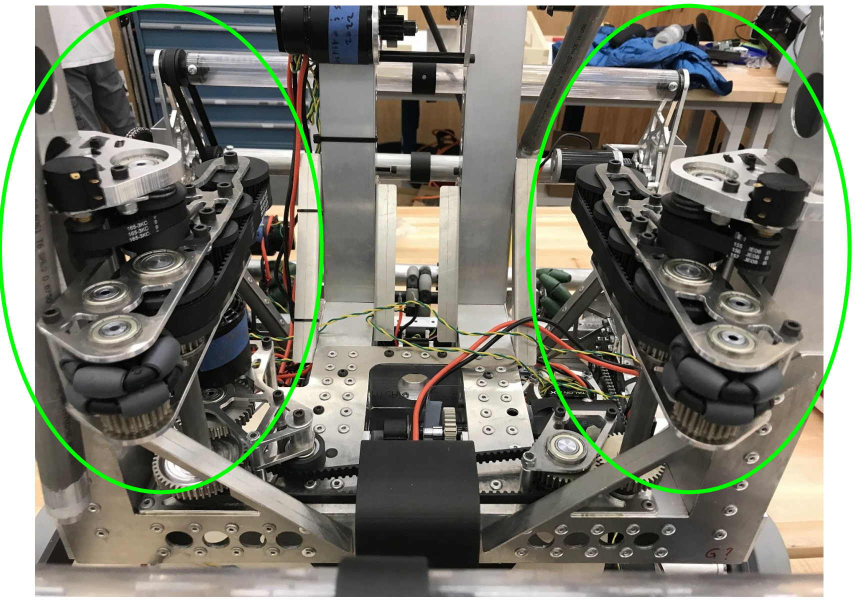


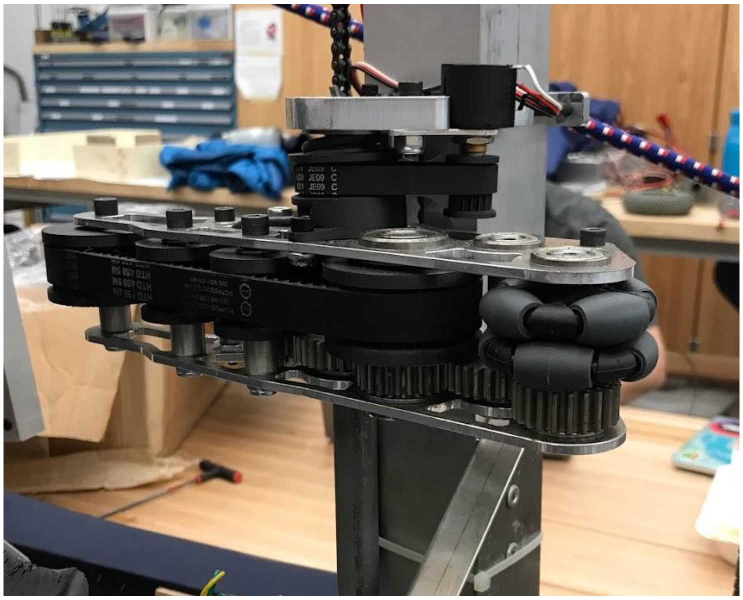
The turret is designed to quickly and reliably turn the catapult, flippy arms, and climber, which are all mounted on top of the turret.

- 102.4:1 gear ratio driven using a Falcon motor
- 12.36 lbf applied using constant force springs to keep the cable carrier organized
- Turret rotates on the combination of a 6.5” ID X-contact and an 1” ID bearing to support forces caused by climbing and firing the catapult

Subsystems

Flippy Arms



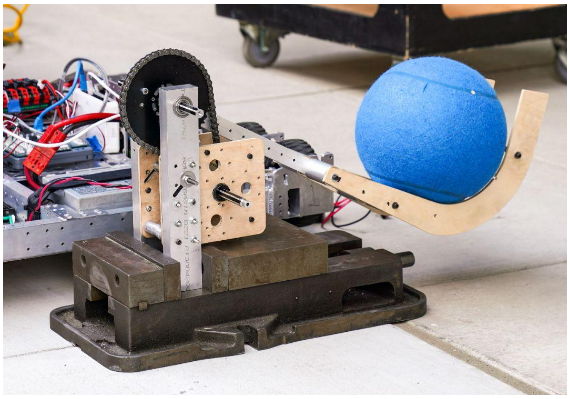


The Flippy Arms were designed to transfer the cargo from the intake into the Catapult. They hold the cargo in place in the catapult as the turret aligns and then open up to release the ball when the catapult fires.

- 2 Flippy Arms per robot
- Powered by a Falcon Motor
- Torsion springs apply 0.75" of compression on the cargo as it's transferred from the intake to catapult
- One way bearing causes the flippy arms to open to release the ball when the wheels are spun in reverse
- Gearbox gear ratio: 7:88
- Flippy arm rollers gear ratio: 32:22

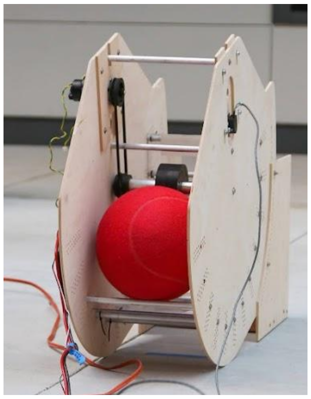
Subsystems

Catapult



Prototype 1 - Catapult

- Extremely consistent and accurate
- Ball variations had little to no impact
- Longer ball acceleration period and lighter catapult significantly increased range



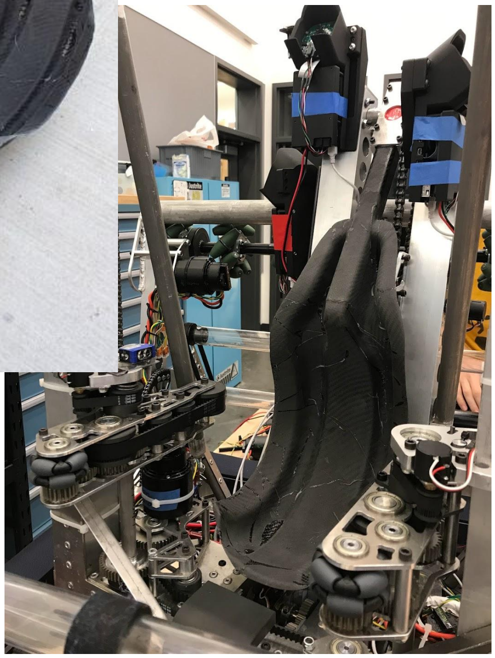
Prototype 2 - Flywheel

- Shot accuracy varied based on ball pressure and surface friction
- Needed higher flywheel RPM and more spinup time to increase range
- Not as consistent as catapult

Final Iteration:

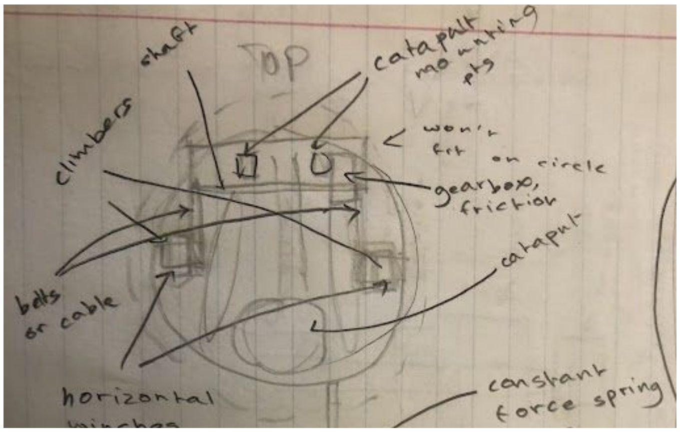
A motor-driven catapult was chosen for accuracy due to it being robust to variations in the ball pressure and surface friction. The design relies on control loops to determine the exit velocity and angle of the catapult when shooting.

- Custom-made carbon fiber catapult is stiff and lightweight
- Powered by 2 Falcon motors
- 7:99 Gear Ratio



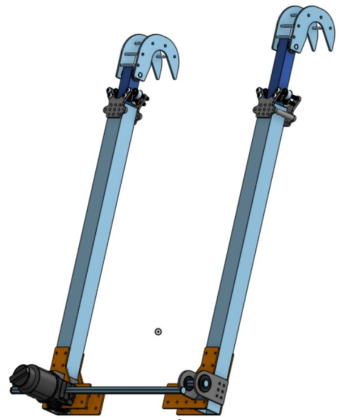
Subsystems

Climber



Iteration 1

- Spool winched climber, motors on 2 sides, friction brake assembly, gear ratio 1:5   packaged on a turret



Iteration 2

- Double spools, powered by a versa gearbox connected by a main shaft, no friction brake assembly instead 1:100 gear ratio

Current Iteration:

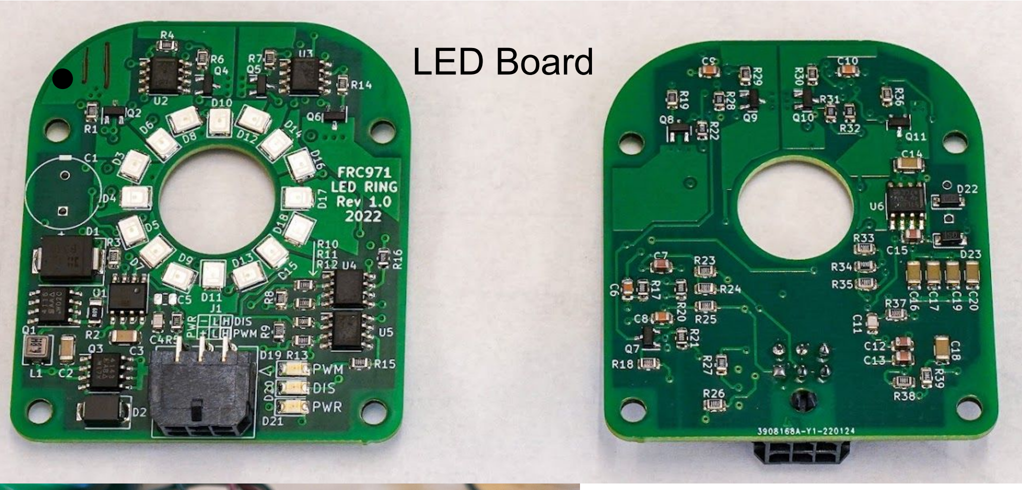
- Compact design, chain and cable combination for weight saving
- Latching hooks on mid bar
- Versa planetary connected to both sides (1:36)
- Air tube extension arms attached with constant force springs for a spring loaded high climber
- Servo + latch assembly holding high bar climb
- Folding hooks, released with a torsion spring for high bar

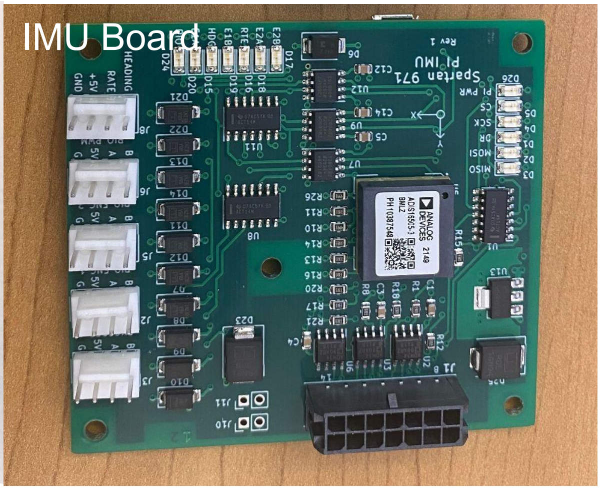
Subsystems

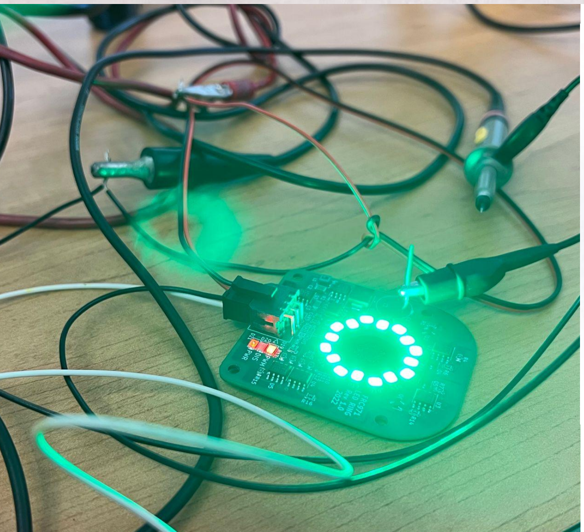
Electrical

3 types of custom PCBs on the robot: IMU, LED rings, a power board for the Raspberry Pis.

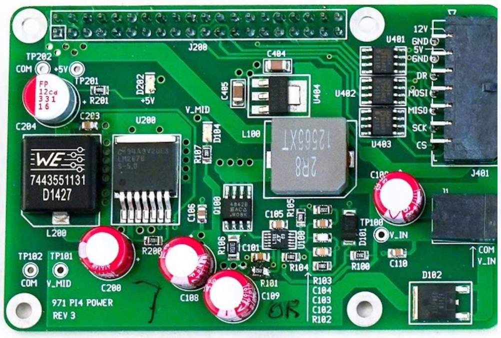
- The design process began when we decided to upgrade our IMU and make a custom board to house it
- As a consequence, the PI Power boards needed to be upgraded to interface with the IMU and LED boards
- The LED Boards were created because we wanted to be able to package a bright, compact led board over a lens effectively for the best retro-reflective tape image captures, which we would not have found off a traditional off the shelf LED board

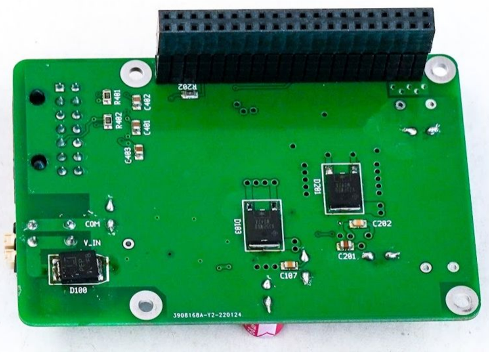






Pi Power Board





Subsystems

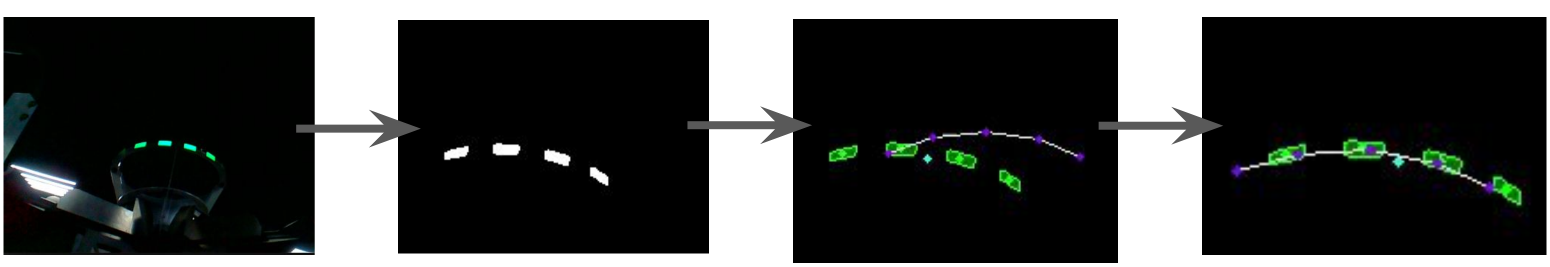
Vision

Vision system used for localization, correcting the drifting IMU measurements.

- Uses our custom LED boards to detect the reflective tape
- 4 Raspberry Pis with cameras and LED boards, giving us 360° vision

- Chooses detected “blobs” of tape with similar dimensions to the real tape, and fits them on a circle

- Figures out what 3D rotation and distance to the target fits the detected blobs’ position and size best
  - Robust to unwanted shaking of the robot, which was a big problem in infinite recharge
  - Uses Google’s Ceres Solver



Raw image | Detected blobs of tape | Solving for rotation and distance

Subsystems

Software

Infrastructure (AOS)

- Layer of code allowing us to communicate between different processes and computers
  - Ex. Autonomous process telling the superstructure to shoot a ball
- Open source, companies use it and contribute to it

Catapult Controller

- Uses model predictive control
  - Can tell it to shoot at a certain angle with a speed in a fixed time
    - Prevents too much acceleration close to release and overshooting
    - Fires in around a 5th of a second
  - Uses OSQP solver to figure out what voltage is needed to get there