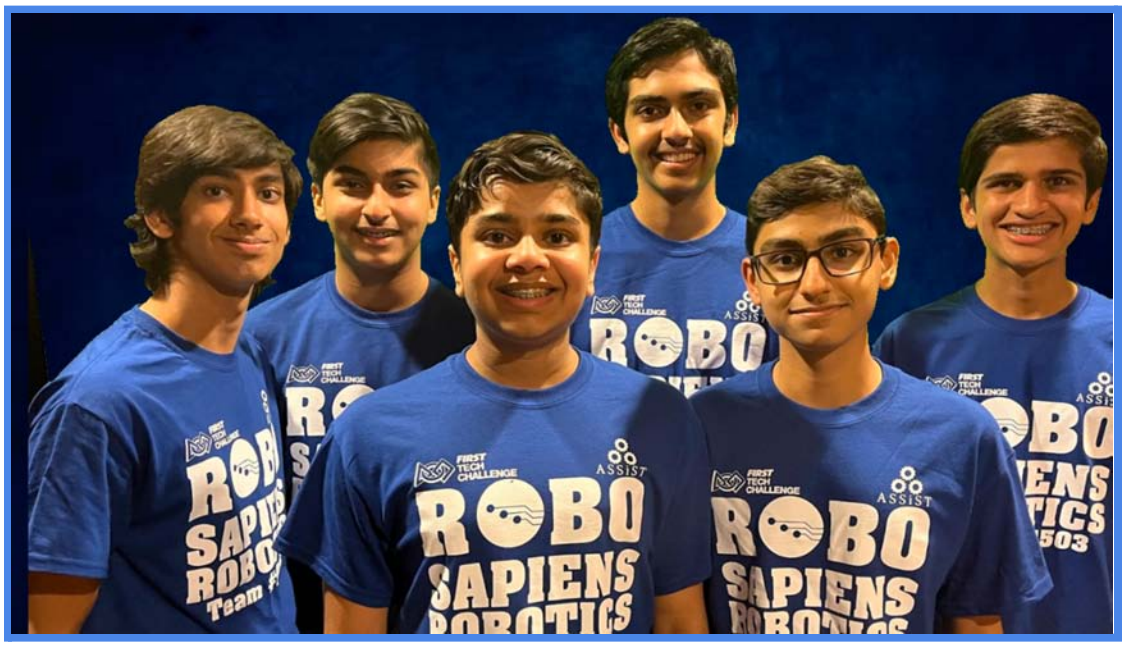
Engineering Portfolio

FTC Team #14503

The Robo Sapiens



2021-2022

10th Grade

Design Progression

Chassis

Design Considerations

- Maneuverable yet **robust**
- Can traverse **barriers**
- Fits the 13-inch **gap** between terrain and wall

Iteration 1: Strafer Chassis

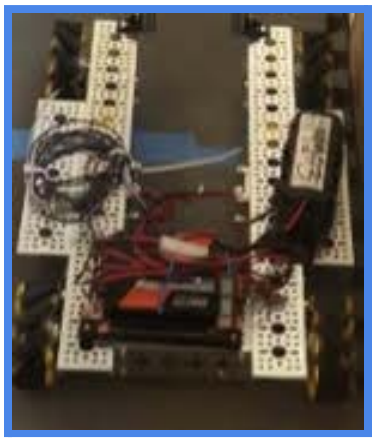
- The original 2022 strafer chassis provided by goBILDA

Pros	Cons
- **Maneuverable** - 5.4 ft/s - Plentiful **space** - 13 in.	- Could not **traverse** the pipes - Could not fit the 13-inch **gap**

Iteration 2: Thirteen Inch Chassis

- Reduced width to **12.5 inches** by cutting center channel

Pros	Cons
- Fits the 13-inch **gap** - **Maneuverable** - 5.4 ft/sec - Increased turn speed	- Very little **space** - Could not **traverse** the terrain - gets stuck

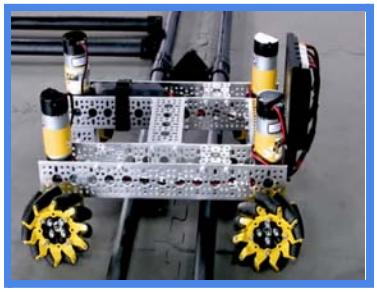


Iteration 2: 13-Inch Chassis

Iteration 3: Raised Thirteen Inch Chassis

- Raised chassis **48 millimeters** by mounting motors **vertically**

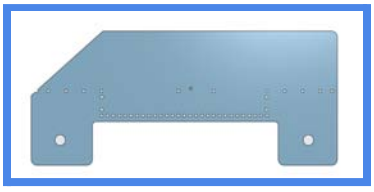
Pros	Cons
- Fits the 13-inch **gap** - **Maneuverable** - 5.4 ft/s - Can **traverse** barrier	- Very little **space** - 8 inches

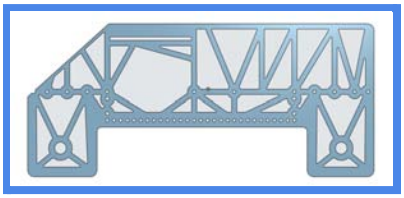


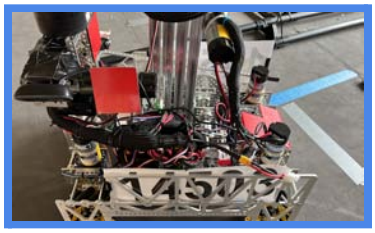
Iteration 3: Raised Chassis

Addition: Side Plates

- Implemented for **protection** after facing defense in the Texas Cup
- **CNC cut side plates** out of **6061 aluminum**
- **Pocketing** removes excess material for a **53% weight decrease**
- Used sponsorship from **Send Cut Send** to manufacture



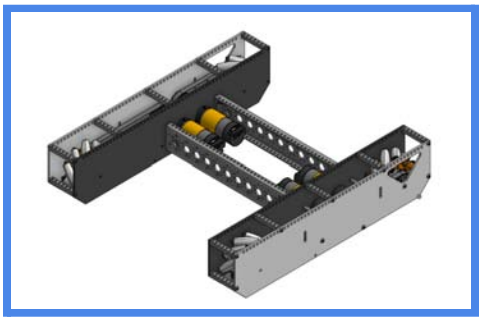




Addition: Side Plates

Computer-Aided Design (CAD)

Something that we discovered is very important in the design process was **CAD drawings**, so we prioritized adding those in as a visual representation of our design process. We drew out **multiple iterations** of each design aspect using **CAD** for mechanisms such as our odometry, transfer, ramp, wheel guards, and team scoring element. Overall, CAD has been **crucial** in our design process. You can see examples of our CAD drawings throughout our portfolio. The software we used is **Fusion 360** and **Onshape**. Additionally, to make sure that our CNC side plates could withstand the rigors of gameplay, we used **MathCad** to **conduct stress tests** on the aluminum.



Custom Belt-Driven Chassis in CAD

Intake

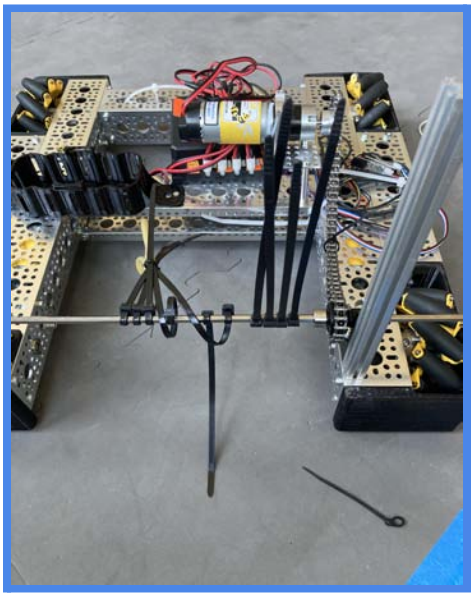
Design Considerations

- Quick and **smooth** pick-up
- Only intake **one freight** at a time
- Wide **range**

Iteration 1: Zip Ties

- Wrapped axle with **zip ties** that were able to grip freight and propel it into our robot at **high speeds**

Pros	Cons
- Could pick up balls and light blocks **very fast** - An extremely **wide range** of pickup	- Could not haul in **heavier cubes** - Intakes 4+ freight

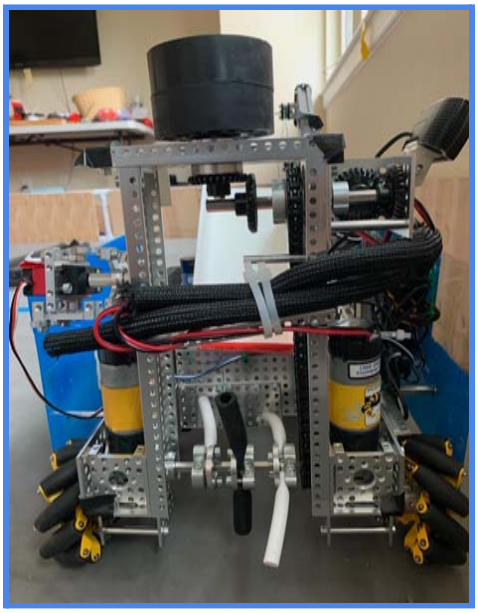


Iteration 1: Zip-Ties

Iteration 2: Surgical Tubing

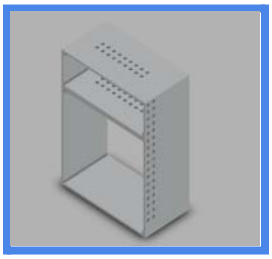
- Change material to **surgical tubing**
- **Narrower** intake area for single intake
- **Chain-driven** rotation

Pros	Cons
- Picks up freight from **tight spaces** quickly - Picks up **weighted** freight - Only intakes one block	- **Bends** over time and needs to be replaced



Iteration 2: Surgical Tubing

Transfer/Outtake



Iteration 4: CAD Box + Guide + Stopper

Design Considerations

- Easy to **replace**
- Efficient to **outtake**
- Can score on both alliance and shared hub **easily**
- Keeps one mineral at a time
- Prevents mineral **lodging**

Four Design Iterations

Iteration	Improvements	Problems
1 - goBILDA Box	- Fits **balls and spheres** - Smooth transition	- **Messy** and unstable
2 - CAD Box	- Neat/**clean** look (3D-printed) - Optimal hole patterns	- Blocks exit inconsistently
3 - CAD Box + Guide	- Uses a **guide** to funnel mineral	- Can hold more than one block at a time
4 - CAD Box + Guide + Stopper	- Only holds **one block** at a time	- Wasn't able to deposit shared hub consistently
5 - Enclosed CAD Box + Ramp Deposit	- Fast shared hub cycle time in order to adapt to Worlds gameplay	- Center of rotation is high

Current Design (Iteration 5)

Attached to stationary linear slide for **manipulation** and scoring
Stopper to prevent intaking of more than one mineral
Small ridge to prevent the mineral from falling out.
Torque servo on the back to drop the freight once ready
Internal guides to allow for accurate deposition

Iteration 5: Enclosed CAD box + ramp deposit

Lessons Learned

- Center of rotation is very high leads to a longer time for the freight to fall out
- Can break easily - have **replacements** ready; 3D printing is not always consistent

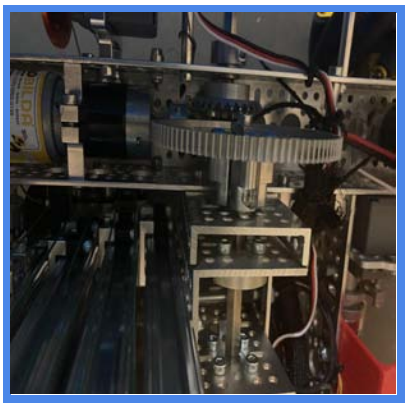
Lift

Design Considerations

- Fast **extension** and **retraction**
- **Adaptability** to different deposit positions
- Reduction in **drive time**

Iteration 1: Angling Linear Slide

- Added a **high torque** motor to **angle** the slides to any desired position
- The **gearbox** to angle the linear slide has as much torque as a **Honda Civic** (**180 lb*ft**)
- **Encoders** for easier autonomous and teleop automation



Iteration 1: Angling Linear Slide (Gearbox)

Pros	Cons
- Fast **extension** and **retraction** - Extension of **3.5 ft** - **6 second** cycle time	- Not very efficient for fast shared hub cycles - Took too much space on the robot

Iteration 2: Fixed Linear Slide

- A **stationary** linear slide at a 55°angle
- Carries the freight, stored in the transfer, from the intake to the shipping hub
- Faster than before using shovel mech as outtake

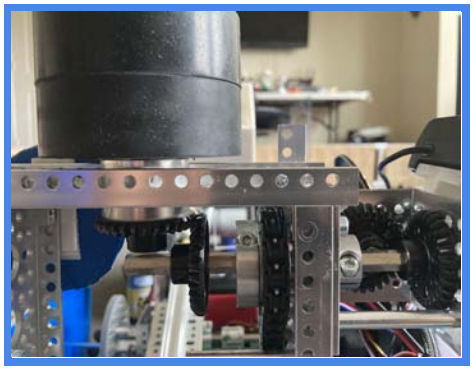
Pros	Cons
- Fast **extension** and **retraction** - Extension of over **3.5 feet** - **5 second** cycle time	- Limited **adaptability** - could only deposit from one single position

Iteration 2: Fixed Linear Slide

Duck Mechanism

Iteration 1: Super-Speed Servo + Gecko Wheel

Pros	Cons
- Compact and **efficient** - **Grips** carousel well	- Too **slow** and low-torque



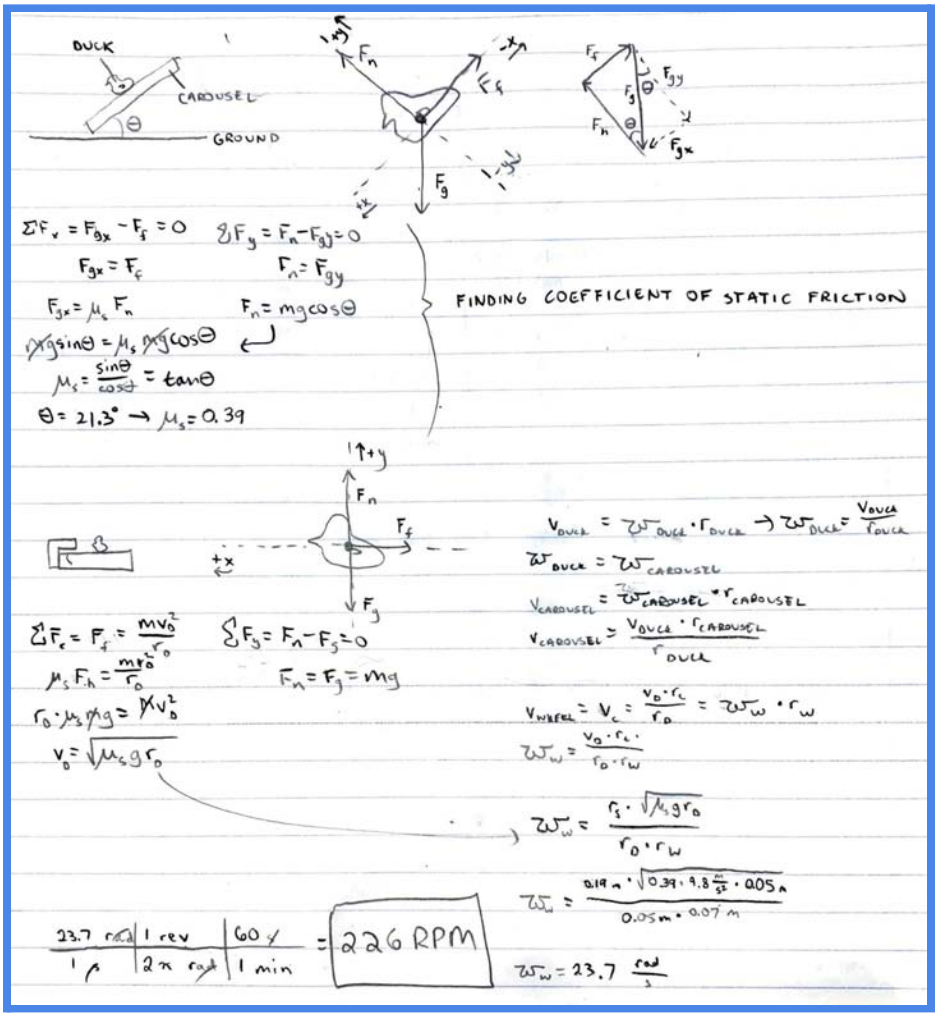
Iteration 2: Intake Motor + Gecko Wheel

Iteration 2: Intake Motor + Gecko Wheel

- Uses **miter-gears** to transmit motor power **perpendicularly**
- **Intake motor** powers both our intake and the duck mechanism with a complex **gear + chain system**
- Saves **space and weight**, both of which are limited
- Motor connects to a **double gecko wheel** with a wide area to account for inconsistencies

Application: Physics of Optimal Motor Speed

- Used the **coefficient of static friction** between the duck and carousel along with **centripetal force** analysis to find the **optimal angular velocity** of our duck wheel



Team Scoring Element

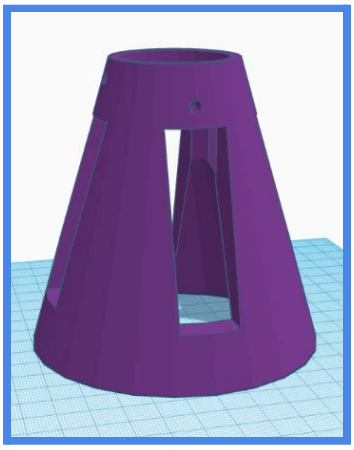
Design Considerations

- **Lightweight**
- Easy to **grab** and place
- Can work with other **alliance partners**

Iteration 1: 3D Design

- **Hollow cone** with a **hole** for an arm to pass through
- Funnel-shaped to allow **room for error**

Pros	Cons
- Works effortlessly with **alliance partner's** capstones - Fits **perfectly** onto alliance hub	- **Heavy** - **Difficult** to grab



Iteration 1: 3D Design

Iteration 2: Magnetism

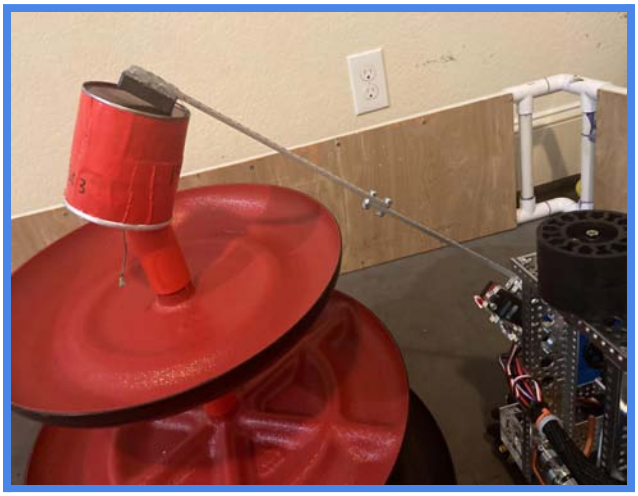
- **Lightweight** cardboard cylinder with a **magnetic** layer on top
- Painted **red** to aid with vision (see OpenCV)

Pros	Cons
- Extremely **light** - **Magnetic** - easy to grab - Easy to replace	- **Sometimes** difficult when placing TSE on top of another team - Can be run over and be **crushed**

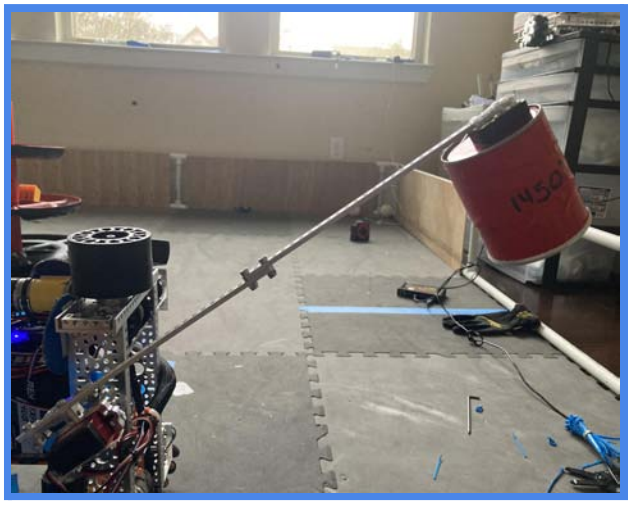
Iteration 2: Magnetism

Retrieval Mechanism: Magnet

- **Magnet** at the tip of the arm that attracts to a magnetic plate on **TSE**
- Long rod attached at a high, **fixed pivot point** on the robot to reach the top of the hub
- Extremely efficient - **10-second cycle** time per TSE capped



Capping Example



Retrieval Mechanism: Magnet

Programming

Odometry

Odometry is a system that uses **dead wheels** to track a robot's position on the field. These odometry wheels are spring-loaded to the floor and spin as the robot moves. By **attaching an E8T encoder** to these wheels and noting the wheels' **circumferences**, we can track exactly how much the robot has moved in a certain direction. With one or two wheels parallel to the robot's motion and one perpendicular, the odometry wheels track the exact coordinate position **in terms of X and Y**. **Heading**, or orientation, is tracked through different methods.

Odometry makes our autonomous significantly more **accurate** because it tracks the ***actual*** movement of the robot, rather than the ***hypothetical*** movement. For example, if the robot hits the wall, the odometry wheels will not move because the robot is not; however, in a time-based code, the programming would **assume** that the robot was actually moving during that period.

The calculations show use of **mathematics** to robotics to localize the robot with **trigonometry**.

$$\begin{pmatrix}\Delta x\\ \Delta y\\ \varphi\end{pmatrix} = \begin{pmatrix}\cos(\theta_0) & -\sin(\theta_0) & 0\\ \sin(\theta_0) & \cos(\theta_0) & 0\\ 0 & 0 & 1\end{pmatrix} \begin{pmatrix}\frac{\sin(\varphi)}{\varphi} & \frac{\cos(\varphi)-1}{\varphi} & 0\\ \frac{1-\cos(\varphi)}{\varphi} & \frac{\sin(\varphi)}{\varphi} & 0\\ 0 & 0 & 1\end{pmatrix} \begin{pmatrix}\Delta x_c\\ \Delta x_\perp\\ \varphi\end{pmatrix}$$

Trigonometric Kinematics

Iteration 1: Three Odometry Wheels - **Math of Localization/Odometry**

- Three dead wheels: two **parallel** (vertical), one **perpendicular** (horizontal)
- Heading calculated through **trigonometric mapping** of the difference between parallels

Setback (Problem): Fragility

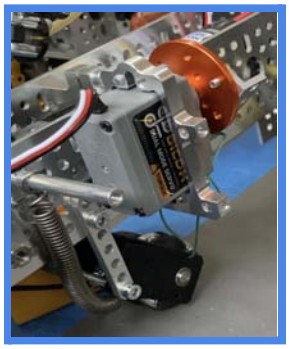
- Encoders were **fragile** - break approximately every month
- **Expensive** to replace - $30 each

Iteration 2 (Solution): Two Odometry Wheels + Rev Hub IMU

- Two dead wheels: one **parallel** (vertical), one **perpendicular** (horizontal)
- REV Control Hub **built-in IMU** measures the robot's heading, or orientation
- More accurate because of fewer magnifications of dead wheel **encoder drift** than three wheels
- Needs fewer **odometry pods**, so we don't need as many replacement parts

Iteration 3: Retractable Odometry

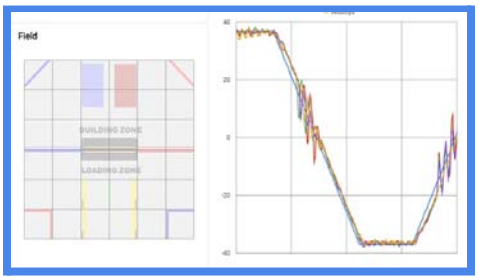
- Need to **traverse barriers** without damaging the fragile odometry pods
- Solution: lift odometry using **servos** at the beginning of teleop, when it is not needed



Standard

Retracted

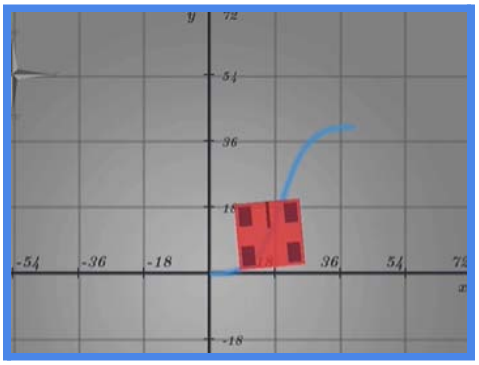
Roadrunner



PID Tuning Graph

This year, in order to increase our accuracy and potential in autonomous, we decided to use a library called **Roadrunner**. Roadrunner is a **motion profiling** library that uses localization to accurately determine the robots **X, Y and theta** coordinates. In addition, Roadrunner implements **velocity** and **acceleration** control which allows the robot to maintain a constant acceleration and accurate velocity throughout the path despite **battery voltage**.

Roadrunner has many different controls that add to the accuracy of the paths. They include a **heading PID**, which accurately corrects the theta angle of the robot, and we also have **translational PID**, which accurately corrects the X and Y coordinates of the robot. The motions also utilize **feedforward** control, allowing for more accuracy with movement (velocity). Roadrunner, when paired with **odometry**, makes our autonomous system very **powerful** overall.



Sample RoadRunner Trajectory

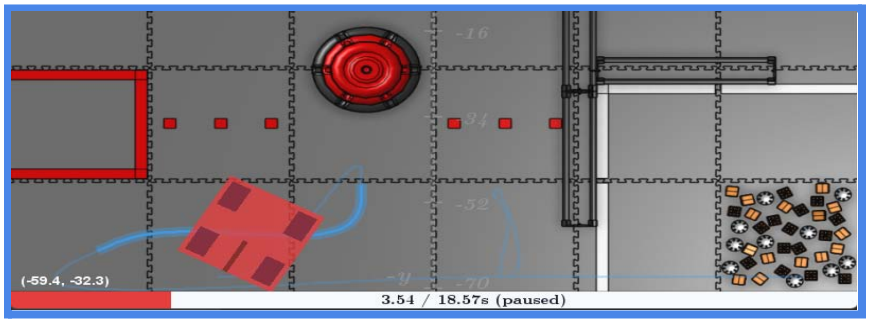
Vision

OpenCV (Computer Vision)

After trying various methods, we found out that **OpenCV** is the most efficient to code, and it is also the most accurate autonomous **computer vision** system. It has a **100% accuracy** rate in detecting the position of the TSE. We used it for determining where to place the starting freight, which then affected the **movements** of the rest of our program.

OpenCV Analysis

The unique challenge this year involved detecting our **own team scoring element** rather than a game piece. Because of this, we had to design a custom-built OpenCV pipeline entirely **from scratch** to specifically match the TSE. Our pipeline draws **three small squares** on each of the three possible positions of the team scoring element on the barcode. In each square, it extracts the **Cr** (red) channel from each pixel using the **YCrCb color scheme** and averages the Cr values for all pixels within that square. This results in three numbers representing the **redness degree** of each possible TSE position, no matter the amount of **light** present. By determining which spot is the most red, we can determine which contains our **TSE**. Although this is tuned specifically to our own TSE, in order to **collaborate** with alliances and be able to **cap** both elements, we have made our vision system such that it can detect **any color of TSE**



Sample RoadRunner Spline Paths