Engineering Portfolio

FTC Team #14503

The Robo Sapiens

2020-2021

9th Grade

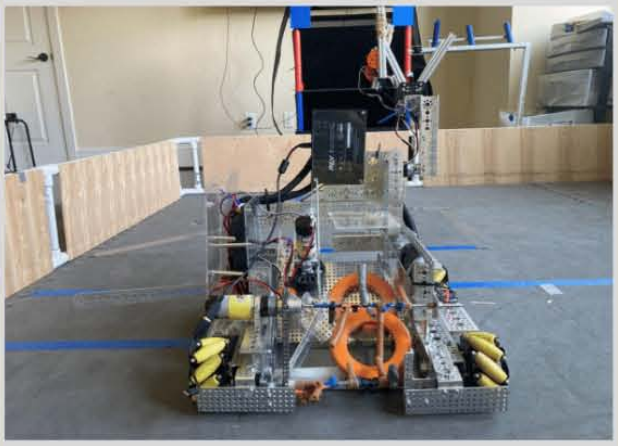
View Judging Materials at: bit.ly/RoboSapiens14503

Team Overview

We are FTC Team #14503: **the Robo Sapiens**, a team consisting of five high school students attending various high schools in the Greater Houston Area. Our coach inspired us to compete in FIRST Tech Challenge, and although we are only a **five-person team** of **freshman** students, we believe that our work this season speaks to our **diligence** and **resilience** in not letting challenges get in the way of **success**.

So far, we have placed **first at all three league meets** by an average margin of **867 total points**. We are one of the **only teams** in the Houston area to **consistently score above 200**. Further, we placed **sixth** at a **global-level competition** called March Mechanical Madness, competing against some of the best teams in the world.

ftcrobosapiens@gmail.com
(832) 361-8084
www.robo-sapiens.net



Robot Overview

This year, we used many **advanced systems** on our robot. To collect rings, we use a two-stage set of **surgical tubing flappers** that push the rings up a **ramp**. From there, up to three rings stack in a **magazine**, ready to be shot. We then have a **super-speed servo** that will, one at a time, push the rings through the **6,000 RPM** flywheel and to the target. For our wobble goal mechanism, we use a **double v-grasp** that **kinematically cages** the wobble goal in **planar translation and rotation**, while **friction** from rubber grips cages it vertically. This caging grasp is attached to a **high-torque motor** that lifts the wobble goal over the field perimeter.

Next, in autonomous, we began to use more **complex coding modules** this year. To detect the number of rings in the starting stack, we use **OpenCV**. For our autonomous movement, we use a **motion profiling library** called **RoadRunner**, which configures predetermined **trajectories** composed of paths to follow while maintaining **controlled velocity** and **acceleration** for an overall significantly more accurate autonomous program.

FIRST Values

As a team, we seek to **uphold the values of FIRST** and **spread its ideas** across our community. FIRST is a great organization that has important ideas. Through our **outreach, mentorship, and connections** within our communities, we believe we have created an **environment more representative** of those values. Below are the **six core values** of FIRST, which are also known as the *FIRST* philosophies of *Gracious Professionalism* and *Coopertition*, and some examples of how we have applied the ideas in our lives and towards those around us.

Discovery: *We explore new skills and ideas.*

- Talked to many different teams via the FIRST Tech Challenge Discord in order to **learn about new technologies** and opportunities in FIRST
- Sought out **numerous mentors** in order to improve ourselves as a team, such as Dr. Neel Doshi, Ph.D., who has a **Ph.D. in Robotics**
- Discussed growing robotics fields with one of our mentors, Dr. Neel Doshi, Ph.D

Innovation: *We use creativity and persistence to solve problems.*

- Created **unique mechanisms** to solve the problems of this year's game
- Analyzed major issues from previous seasons in order to determine the **best possible option** for this year (e.g. using odometry due to previous autonomous inaccuracy)
- Spread our innovative ideas to other teams in FIRST and individuals in the community

Impact: *We apply what we learn to improve our world.*

- Held **many outreach events** to teach others about FIRST and our robotics project
- **Gave opportunities** to visit our pit and drive our robot, which engaged both younger and older participants in FIRST's world
- **Mentored students** from various robotics teams in explaining some of the **robotics concepts** implemented on our robots, which then helped them with their own.

Inclusion: *We respect each other and embrace our differences.*

- **Listen** to everyone's ideas at team meetings
- Support **diversity** both in the **community** and in **FIRST** as a whole
- Take **all ideas** seriously and implement/test all in order to find the best solution

Teamwork: *We are stronger when we work together.*

- **Rely on each other** for help and ideas
- Combined multiple possible designs offered by different members into **one better design** using **synergy** (1+1=3)
- Work on all tasks with more than one person for **uniqueness and security**
- Taught about and **encouraged teamwork** in our mentorship and outreach

Fun: *We enjoy and celebrate what we do!*

- Make our meetings fun by talking and **celebrating** competition wins
- Create a **positive experience** and environment in the robotics pit
- Bond our members through a **common love of STEM and FIRST**
- **Teach others** about the many ways that robotics is **enjoyable** in outreach/mentorship

Team Goals

Since this is our 3rd year participating in FTC, **our goals are very important to us** as a team because they help us make progress and better **exemplify the traits of FIRST** while also succeeding in competition. Below, you can see some of those goals separated into three main categories as well as **the steps we took to achieve them**.

Mechanics

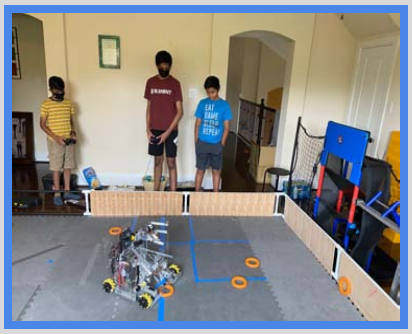
- Create **innovative mechanisms** to be able to complete all scoring opportunities with **well-integrated** parts - Accomplished (steps to accomplishment below)
    1) Brainstormed the best possible routes for completing each task
    2) Narrowed down ideas based on **practicality and efficiency**
    3) Implemented and tested designs
    4) **Analyzed** results and **improved** accordingly
- Integrate **CAD** designs - Accomplished (steps below)
    1) Researched the best **CAD** software (Autodesk Fusion 360)
    2) Attended training courses to learn about the software
    3) Drew design aspects
    4) Noted flaws/weaknesses and **redesigned accordingly**

Coding

- Implement new autonomous technology using **Java** - Accomplished (steps below)
    1) Researched possible technologies on our own (e.g. RoadRunner, odometry, PID)
    2) **Brainstormed** implementation ideas for integration onto our robot
    3) Tested and **tuned** after implementation
- Implement teleop **automations** for more **consistency** - Accomplished (steps below)
    1) Considered the most **optimal** and **tedious** driver tasks by **analyzing** game strategy (found to be aligning to the high goal/powershots and shooting quickly)
    2) Brainstormed ideas for making those tasks easier
    3) Using RoadRunner, odometry, and PID (above), created **teleop automations** that let drivers **align** to the high goal, shoot rings quickly, and shoot power shots

Outreach and Mentorship

- Spread **FIRST's values** and mission to our community - Accomplished
    1) **Reach out** to potential communities (robotics clubs, science fair hosts etc.)
    2) Brainstormed the best **presentation techniques** for the given audience
    3) Set up our outreach materials and teach FIRST's important ideas!
- Mentor other FIRST teams - Accomplished
    1) Reach out to **potential teams** (schools, community, etc.)
    2) Identified possible teaching points for **younger communities**
    3) Presented information that helps teams using our experience
    4) Addressed specific issues they are having on their robot
- Receive mentorship from notable members of the robotics community - Accomplished
    1) Identified **experts** in the robotics fields
    2) Reached out to those experts, explaining FIRST and requesting mentorship
    3) **Met with the mentors** to discuss both our robot and the specific fields in which they specialize



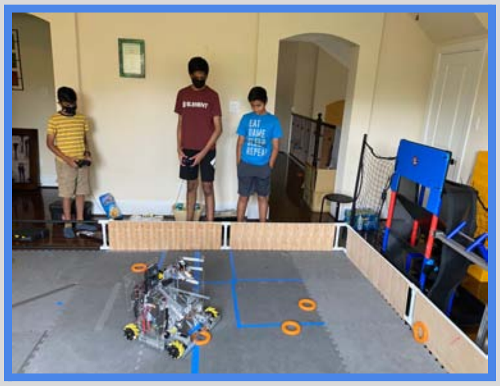
Outreach

As a team, we believe that outreach is one of the most essential parts of *FIRST* because it allows us to **spread our passion for robotics** with people who may not have had a chance to pursue opportunities in STEM. We can also **open new options to existing STEM students**. We have participated in outreach opportunities to both Technical and Non-Technical communities throughout our years in FTC.

Technical:

Robotics Clubs

- Reached out to **local MS Robotics Clubs**
- **Introduced FIRST** with fun robot minigame competitions
- Presented about FIRST and FTC
- Encouraged students to become more engaged in **robotics/STEM**
- Taught about and supported FIRST's essential values
- Helped with technical concepts such as **SeaPerch/FLL**

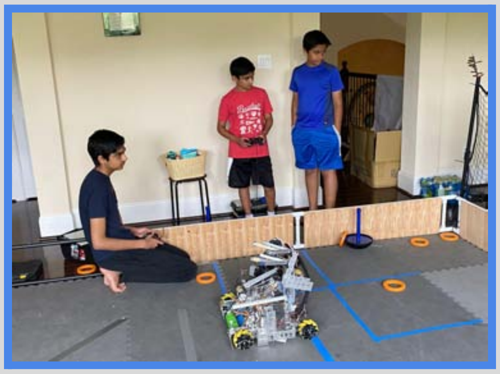


FLL Club Mentorship (see next page)

Non-Technical

Community Advertisement

- Created enriching opportunities to **STEM and non-STEM students** by offering inside views of our team
- Reached out to over **50 local businesses** for sponsorship and to spread the **FIRST message** and competition
- Exemplified FIRST's core values
- Informed participants about FIRST's **opportunities**



Pit Visits

- Reached out to over 20+ students to give an **overview** of our team
- Invited people from the community weekly over to our pit to show them our robot and **give them ideas** for their own STEM endeavors
- Explained to them the values and morals of FIRST
- Demonstrated the **core values of FIRST**
- Allowed the students to drive our robot and observe matches in order to **spark their interest**
- Encouraged the students to pursue their own FIRST teams

Science Night

- Held a booth at the **World of Science Night** at our MS
- Informed the public about all aspects of **FIRST**
- Held **driving demonstrations** with our robot for students

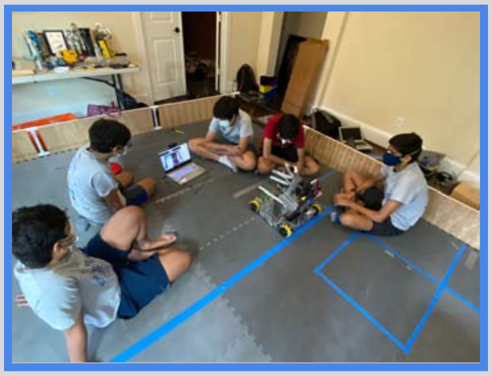
Mentorship

Another important aspect of our team is mentorship. Over our years in FTC, we have both **sought out mentors** to acquire new expertise and **mentored other FIRST teams** ourselves.

Giving Mentorship - FLL (*FIRST* Lego League) Teams

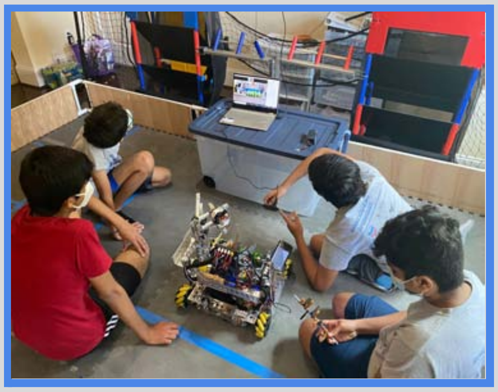
- Taught Sartartia Middle School FLL teams about FTC and important concepts in **both programming and design** that helped them succeed in their competition
- Mentored another younger team involved in FLL, including team Orion Force, which has now **decided to join FTC**
- Helped FLL team Orion Force with issues they were having involving gear ratios using our prior **knowledge from FTC**

Receiving Mentorship - post-doctoral fellow from MIT with a Ph.D. in Robotics from Harvard (full bio below*)



- Zoom call with an **MIT** fellow with a **Ph.D. in Robotics** from **Harvard** University
- Discussed technical aspects of our robot such as the optimal **angular velocity** for shooting rings - then **applied what we discussed** by creating a more constant guide for the rings into the magazine, which increased accuracy
- Discussed growing fields in robotics such as **perception** (deep learning, neural networks, etc.)

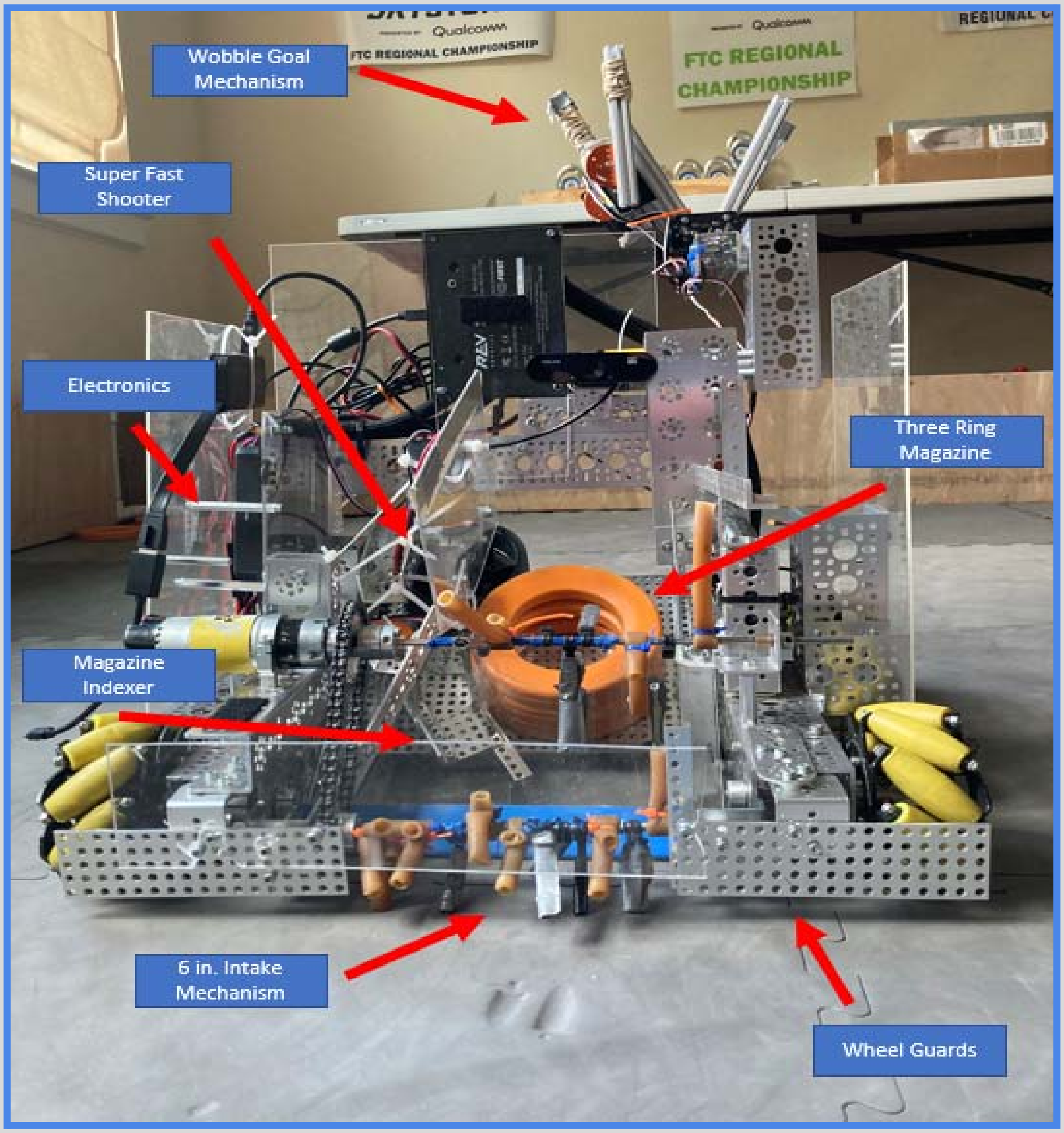
Receiving Mentorship - Mechanical Engineers



- Meeting with **community experts** as part of pit visits
- Discussed troubleshooting and **mechanical aspects** of our robot
- Have been crucial to our success - including helping us with numerous issues and ideas such as when Tetrix hubs were stuck on our motor axle.

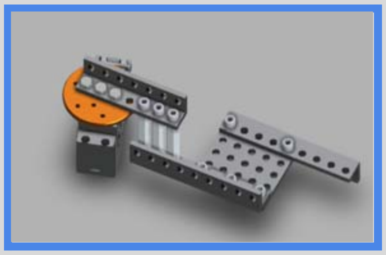
* Neel Doshi is an Intelligence Community post-doctoral fellow advised by Professor Alberto Rodriguez at the Massachusetts Institute of Technology, where he focuses on developing methods for planning and controlling contact-rich robotic manipulation and designing novel robot end-effectors. He graduated with a B.S. in Mechanical Engineering (2012) and a M.S. in Robotics (2013) from the University of Pennsylvania. He earned his PhD (2019) from Harvard University under the supervision of Professors Robert Wood and Scott Kuindersma, where he worked on the design of millimeter-scale robots, including the Harvard Ambulatory MicroRobot (HAMR), and created algorithms for planning and controlling the locomotion of legged robots at this scale.

Robot Overview - Major Components



Design Progression

Computer-Aided Design (CAD)

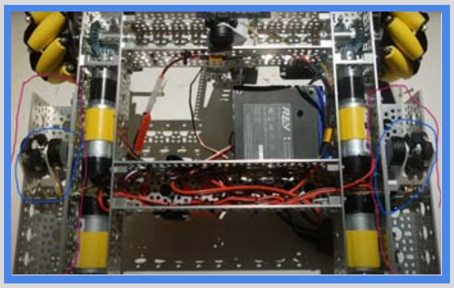


Something that we discovered is very important in the design process was **CAD drawings**, so we prioritized adding those in as a visual representation of our design process. **We drew out multiple iterations** of each design aspect using CAD for mechanisms such as our **odometry, shooter, intake and wobble goal.** This was invaluable in determining which design would be most effective in our robot and allowed us to create the many **iterations** seen in this section. For example, after drawing out CAD for the wobble goal mechanism, we noticed that it would be a bit flimsy and insecure when gripping the wobble goal. So, we decided to add a second V-grip (detailed later).

Overall, CAD has been **crucial** in our design process. You can see examples of our CAD drawings throughout our portfolio, and we have added some here as well.

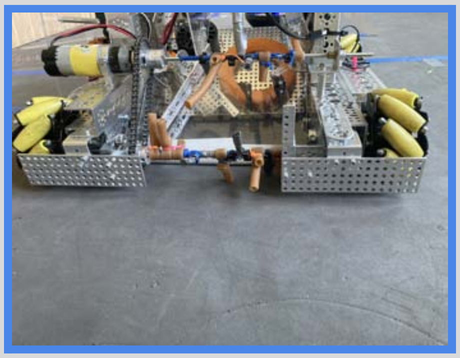
Basic Design Elements

Iteration 1: Strafer Chassis



This year, we knew that the rings would be taking up a lot of space on the inside of our robot. We also knew that we would need significant room for our intake and shooter mechanisms, which is why we needed a chassis that would be spacious. For this reason, we decided to use the **strafer chassis from GoBilda**. This chassis stores the drivetrain motors inside of the chassis channels, which allows for more room on the interior of the robot. It also fits within the sizing cube, although just barely. With **mecanum wheels** as well, the chassis has been a great success for our team. To fit this year's game, we decided to slightly modify the gobilda chassis. To allow space for the rings, we decided to replace one of the horizontal channels with a **Low Side U-Channel.** This way, we were able to allow space for our magazine and our shooter. To fit our odometry onto our robot, we needed to add an extra channel on both sides of the robot so that our odometry pods could fit into the chassis.

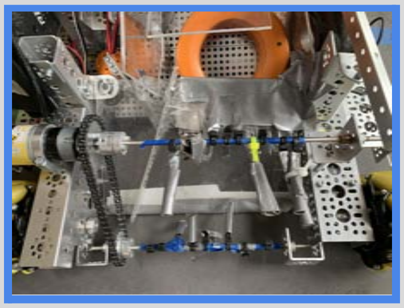
Iteration 2: Adding Wheel Guards



In addition, during teleop, there are always many rings near the front of the robot around the intake, but there is only a small range of flappers that actually propel those rings onto the robot. Our wheels sometimes drive over the rings and get stuck, which limits our movement. To fix this problem, we decided to add **two metal plates** so that the rings can not slide under the front of the robot. These plates also guide the rings to the center of the robot so that they can **efficiently travel up the ramp and into the magazine.**

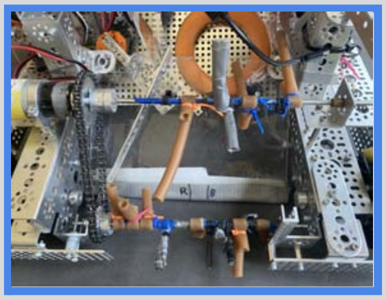
Intake Mechanism

This year, we knew we would have to implement an **inclined ramp** as well as a **dual stage intake mechanism**. Our overall intake design is to have a rod attached to a motor and a gear that allows us to continuously rotate and ensure the upward movement of the rings onto our robot. We decided to go with a chained mechanism with a **gear ratio of 2:1 in a two-stage intake.** The speed of the motor controlling the surgical tubing would have to be twice the speed of the robot due to physics, and this ensures **speed and efficiency** of intaking the rings when the robot is moving around.



Iteration 1: Zip Ties

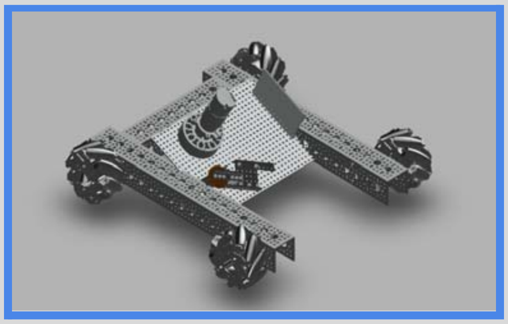
At first, both stages of our intake consisted solely of **zipties** wrapped in duct tape. This acted as a powerful force to propel the rings up the ramp. However there were two issues. The first issue was that the zip ties were slapping against the field and damaging it, and overall, the design was **messy** and **consistent** in intaking the rings.



Iteration 2: Surgical Tubing + Zip Ties (Second Stage)

Because of the issues above, we decided to add **surgical tubing** to our intake mechanism on both stages. which has decreased our cycle time by about 10 seconds. The surgical tubing was far more effective due to its ability to **grip on the rings and its flexibility** while spinning. With the addition of the surgical tubing, we were able to pick up rings while moving in any direction and without the support of the walls, allowing for more efficient gameplay. Now, with the addition of wheels on our lower level, we have decided to only keep surgical tubing and zip ties on the second stage of the intake.

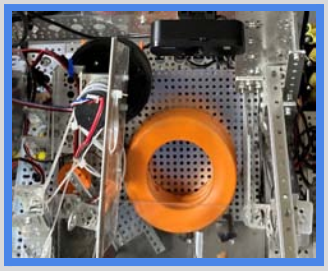
Shooter Mechanism



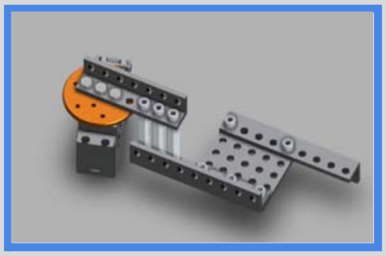
Iteration 1: One-Wheel Design with Gecko - **CAD Drawings**

Because this year's game relies heavily upon shooting rings into the goals we felt that we needed to spend a lot of time on developing an accurate and efficient shooting mechanism. We decided on using a **single wheel shooting mechanism** for two reasons. The first reason is space; by only using one wheel we were able to fit all of the components that were necessary on our robot without leaving the size limit. The second reason is spin. A one wheel design allows the rings to spin which and travel farther similar to a frisbee.This mechanism allows us to shoot **farther and more accurately** than other designs we tried which allows us to maximize our point potential. Originally on our shooter we had implemented a **gecko wheel** which we found to be very ineffective and inconsistent. For this design, we decided to create a **CAD drawing** in order to optimize the angle of the ramp in relation to the wheel, calculating the best possible ring trajectory.

Iteration 2: Stealth Wheel - After testing the gecko wheel, we decided that it was time to change to a **stealth wheel**. This was due to the fact that the gecko wheel, when powered by the rotation of our **high speed motor**, would spin very fast causing it to expand. It caused an inconsistency in our shooting mechanism as it was very variable, and it caused a lot of sound. Our new stealth wheel is bigger than the gecko wheel and **doesn't bloom when powered** which leads to more consistency in our shooting. Additionally, the stealth wheel has greatly increased our speed when outtaking rings, allowing us to score more rings in the high goal than we usually would have. Now, we only shoot at 50% of the motor's max velocity. This is because the velocity of the ring is dependent on the radius and the angular velocity of the wheel.This way, we have more room for error at the competition.



Constant: Guides and Compression - the Physics of Shooting, CAD
The final aspect of our shooting system is our guides and compression. First, in order to keep our rings' path as constant as possible, we added **guides** so that our rings always are in the same place. We also added a straight bar opposite to our flywheel so that rings compress on their way out, which provides more force on the **trajectory**.



In order to calculate the **optimal angle** and size for these mechanisms, we used advanced **AP-level physics**, which can be seen below. For example, we calculated the ring's trajectory and the **optimal RPM** for our motor with the given ramp angle. This allowed us to really dive deep into strategy and think about the **underlying science** of the game.

Vertical Components

$a_g = 9.8m/s^2$

$V_{yf} = 0m/s$

$\Delta y = 1m$

Horizontal Components

$a_x = 0$

$V_{xi} = V_{xf}$

$\Delta x = 2m$

Vertical Component Calculations

$v_f^2 = v_0^2 + 2a_g\Delta y$

$0 = v_0^2 + 2(-9.8) * 1$

$v_0 = 4.427m/s$

$v_f = v_0 + a_g t$

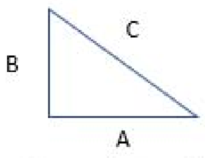
$t = 0.452sec$

Horizontal Component Calculations

$\Delta x = v_{ix} + (1/2)at^2$

$2 = v_ix * 0.452 + 0$

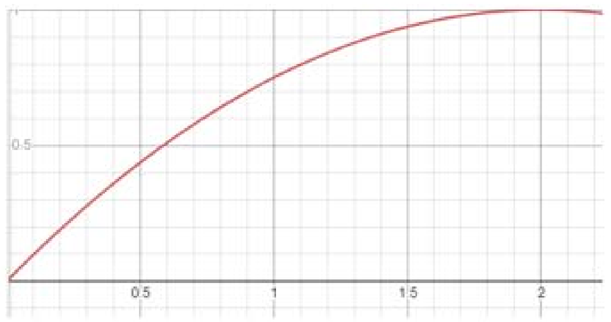
$v_{ix} = 4.425m/s$



$v^2 = v_{ix}^2 + v_{iy}^2$

$v = 6.259m/s$

Ring Trajectory



Calculating Ideal RPM

$w = v/r$

$w = 125.18radians/second$

$radians/second -> rotations/minute$

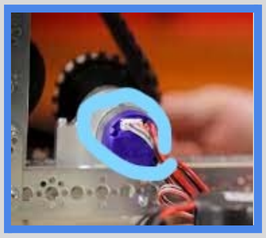
$w = w * 60/2\pi$

Accounts for motor load $RPM = 1195.381 * 2 * 2$

$\boxed{= 4781.524RPM}$

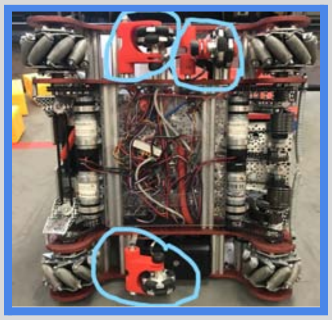
Autonomous Movement Mechanisms

Iteration 1: Motor Encoders



Last year, we used motor encoders in order to move around autonomously. We tried that again at the beginning of this year, and it was much more accurate than time-based movement, but it was still not too accurate since our robot was unbalanced. This caused it to **curve** when making supposedly straight motions. Because of this extreme inaccuracy, which ended up ruining our motions, we decided to switch to more accurate, **motion- and position-tracking mechanisms**.

Iteration 2: Three Odometry Wheels - **Math of Localization/Odometry**



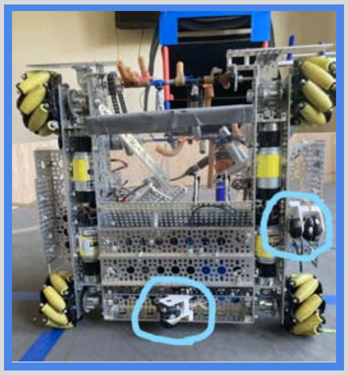
This year, we first tried using **three odometry wheels** on the bottom of our robot in order to more accurately track the motion of the robot. These odometry wheels are not connected to any motors that power motion, but instead, they just spin with the robot. By **attaching an encoder** to these wheels and noting the wheels' circumferences, one is able to track exactly how much the robot has moved in a certain direction. With 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**. Also, there is the issue of heading, or orientation. Since there are two odometry wheels in the parallel direction, this allowed us to **trigonometrically** map the difference between the two wheels' motions and thus the amount the robot has turned. The calculations below were examples of us using our previously-learned **math skills** and applying them to robotics in order to successfully localize the robot with **trigonometric odometry calculations**.

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

Kinematics to localize robot

However, we had one major issue with **odometry** wheels that was not ever resolved. In the beginning, all of the original three encoders that we bought broke. The encoders were $30 dollars each (not cheap), and we were on a tight budget, so we could not afford to buy more. So, we ended up having to move on to two-wheel odometry. To check and prevent the problem, we had to open the caps and make sure the disk wasn't rubbing against the sensor.

Iteration 3: Two Odometry Wheels + Rev Hub IMU

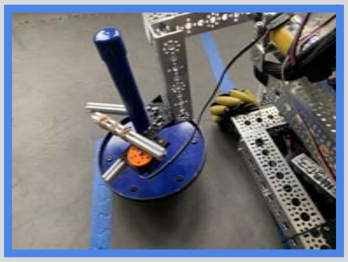


Since the odometry wheels kept breaking, we decided to use only two instead of three as we had previously planned. However, using two odometry wheels in the same direction allowed us to make note of differences between the two wheels' motions, which results in rotation of the robot. With only one wheel for X and one for Y, there is no way to do this with two odometry wheels. The solution for this that we found was instead using the **Rev Expansion Hub built-in IMU sensor**, which can track the orientation of the robot. Surprisingly, we found that this was extremely more accurate because the IMU is more accurate than the odometry wheels, and errors were magnified when using three odometry wheels.

Wobble Goal Mechanism Iteration 1: Servo-less Mechanical Mechanism

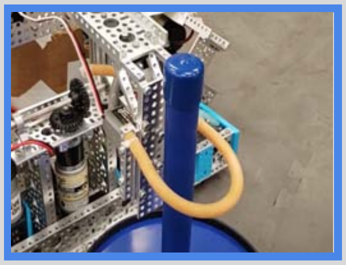
Our very first idea was to use a purely mechanical mechanism that did not rely on any servos. We noticed that the top of the wobble goal was thicker than the body, causing there to be a little slip to grab onto. As the robot drives up to the wobble goal, the wobble goal is guided to the base of the V. Then, the two sides of the V are used as **supports**, with the wobble goal cap resting on those two edges. However, this was a very flimsy idea, so we decided not to use it.

Iteration 2: V-shaped Servo Grabber



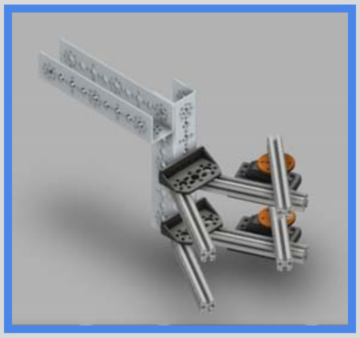
Picking up the wobble goal and placing it around the playing field was a problem that we needed to tackle. We thought the most efficient way was to use a **pinch mechanism**. The pinch consisted of **two steady REV bars** that would guide the wobble goal to the center of the grasp. We then had another REV bar that was attached to a servo, causing it to move and lock onto the wobble goal, making the connection on all the bars firm. In order to lift the wobble goal off of the ground, we attached our pinch mechanism to a **tetrix bar** on the side of our robot. However when we tried to lift the wobble goal we realized that the motor pulling up the bar was not strong enough, so we had to buy a stronger motor with **high torque**.

Iteration 3: Surgical Tubing Caging Mechanism



Our next idea was to have a loop of **surgical tubing** that goes around the wobble goal pole. When we tighten the surgical tubing around the pole, it would squeeze very hard and grip the wobble goal well, allowing us to pick up the goal. However, we decided not to use this mechanism due to **difficulty** in execution/implementation and a better option (below).

Iteration 4: Double V-Shaped Servo Grabber - CAD Drawing



In our original V-shaped servo grabber, we had only one servo restricting **planar translation and rotation**. In addition, friction was only present at one point of contact. To increase the grasp of the wobble goal, we decided to add another point of contact for friction to **cage** the goal vertically. Now, we can keep the wobble goal during teleop without the fear of dropping it. For this design element, we created a **CAD drawing** in order to visualize the grip on the wobble goal.

Programming

Roadrunner

This year, in order to increase our accuracy and potential in autonomous, we decided to use a new 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. Roadrunner incorporates numerous advanced paths and strategies that can maximize the efficiency of the robot's **motion profiling**.

Use of Odometry

To make our autonomous period more accurate, we decided to switch to **odometry** from motor encoders. Odometry is a way that we can track the robot's position using two odometry wheels that are not connected to any motors. Instead, they just spin with the robot. By attaching an **encoder** to these wheels, we track exactly how much the robot has moved in a certain direction. The odometry wheels track the exact **coordinate position** in terms of X and Y. For orientation, we use the built-in Rev Expansion Hub **IMU** (a **gyroscope**).

The reason we switched from motor encoders is because when the robot would hit a robot or wall, the motors would still be moving and changing encoder counts even though the entire robot is not moving. This would cause large amounts of errors which we can not get rid of. With odometry, we used a wheel which has an **E8T encoder** attached to it to measure where the robot is. Since the wheel has no motor attached to it, when the robot stops moving, the odometer stops moving as well. Assembling and coding two odometers was very tedious, but combined with Roadrunner, our autonomous has been very accurate and powerful.

Trajectories

Trajectories are paths with **motion profiling** and are the main components of programming with RoadRunner. They map our robot coordinates, which helps us to accurately control the acceleration, velocity, and **angular acceleration**. Roadrunner contains many different trajectories that can be used to create a motion profile. The ones that we used include, .forward, .strafeLeft, .strafeRight, .splineTo, .lineTo, .lineToLinearHeading, .splineToConstantHeading, and .splineToLinearHeading.

Structure of a Trajectory

A **trajectory** is essentially just a **series of movements** compiled together to form an autonomous path for the robot to move along. Using odometry, we are able to make the entire playing field a coordinate plane by tracking the amount of movement in both the x and y directions.The trajectory is composed of multiple **different paths put together into one**. However, of course, we also need interruptions for servo/motor movements in autonomous mode. For this, we used markers.

Markers

Markers are interruptions in a trajectory that allow for servo and motor movements, which are necessary in the autonomous period. There are three different types of markers, but the main one that we used was **temporal markers**, which are based on time. This means that the code specified in the marker will run after a certain amount of time has elapsed in the trajectory. We use markers to turn on our shooting motor, move our **wobble goal arm**, turn on our intake, and **secure the wobble goal** with a servo.

Trajectories in Teleop

Because of the **versatility of Roadrunner**, we were also able to implement trajectories into our teleop this year. The way this works is we initialize a certain path at the beginning of teleop, and once a certain button is pressed, we run the code to follow the path. This has been useful with powershot automation as well as aligning to the high goal.

Autonomous Sensing

OpenCV(Computer Vision)

After trying various methods, we found out that OpenCV is the easiest to code, and it is also the most accurate autonomous **computer vision** system. Although we at first had trouble using the code, we eventually made it work with the help of our experience in previous coding classes and **object-oriented** programming knowledge. Now, it has a 100% accuracy rate in detecting the number of rings in the stack. We used it for determining where to place the wobble goal, which then affected the splines, strafing, and rotation of the rest of our program.

The one downside to OpenCV is that we have to align the camera at the correct angle for the blocks. To **solve this**, we at first tried using a **phone holder and using the phone camera**. However, this became very **tedious**, and there was too much **inaccuracy** at our regional competition, so instead, we decided to use a **webcam**. This has allowed us to keep a **constant position** for the camera between different autonomous programs, which has made the alignment time period to be quick and efficient.

Game Strategy and Play

Autonomous Period

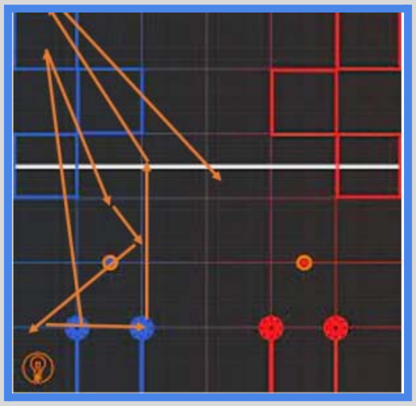
Detection using OpenCV

The very first thing that our robot does during autonomous driving is look at the stack of rings through the **webcam** and determine how many rings are in the stack. Based on this, it determines which depot the wobble goal should go in.

Customizability

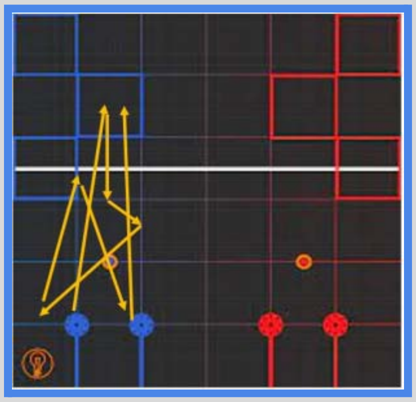
Below are the default autonomous paths that we have. However, with **RoadRunner**, we are easily able to alter the code paths to better **cooperate with our alliance** partners at the contest.

Four Ring Randomization



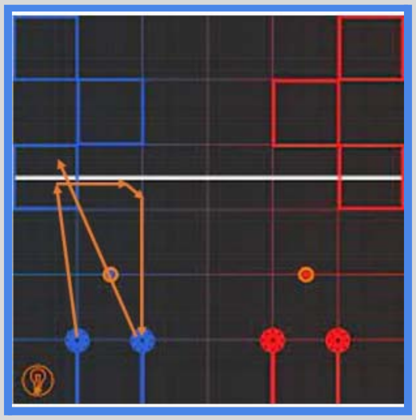
- Spline to drop zone C to deposit the wobble goal.
- Spline to shooting position to shoot 3 preloaded rings
- Strafe to face the four ring stack (Optional)
- Intake as many rings as possible while moving forward (Optional)
- Slight spline to pick up the second wobble goal (Optional)
- Spline to shooting position to shoot ring stack (Optional)
- Strafe to drop zone C to drop the second wobble goal (Optional)
- Move straight forward with remaining time and park.

One Ring Randomization



- Spline to drop zone B
- Move to shooting position and shoot three preloaded rings
- Strafe to face the one ring stack (Optional)
- Intake the one ring (Optional)
- Spline to shooting position (Optional)
- Strafe and turn to face the second wobble goal (Optional)
- Move forward and pick up the wobble goal (Optional)
- Spline to drop zone B and park

Zero Ring Randomization



- Spline to drop zone A
- Strafe to shooting position and shoots three preloaded rings
- Strafe and turn to face the second wobble goal (Optional)
- Move forward and pick up the wobble goal (Optional)
- Spline to drop zone A and park

Autonomous TOTAL: 71-107 Points

Driver-Controlled Period

Playing Strategy

We wanted to get most of our points from the teleop part of the games. Our strategy in the teleop period is that we will continually collect rings and shoot them into the **high goal**, as this gives us the most points. We will also make sure that the wobble goals are in an easily **accessible position**. Once the endgame begins, if we have rings, we will shoot them at the **power shot** targets. Otherwise, we will quickly put both wobble goals outside of the field. If there is extra time, we will get more rings and shoot them at the powershots if not already knocked down or continue to aim at the high goal.

Teleop Automation

This year, with our **highly advanced** autonomous systems, including **RoadRunner**, we were able to implement **automation** in our teleop to make it more efficient and easier for the drivers. The first thing was **auto-alignment for shooting**. With this, the driver is able to reset the pose of the robot (due to drift, odometry gains about 1-2 inches of error after autonomous), and afterwards, with the **push of a single button,** the robot will move to the correct position, ready to align to the high goal. This technology has **improved our cycle time** by nearly 5 seconds.

Another feature was **automatic shooting**. With this, we can click a single button, and the robot will **shoot all three rings consecutively**. The code has already calculated the most optimal break time between shots in order for the servo to be able to move and reset, which makes this mechanism very efficient. Although a small improvement, our automatic shooting has **decreased each cycle time** by about 1 second.

The final, and by far most useful, feature in teleop is our **powershot automation**. In our earlier matches, we used to use **all 30 seconds of endgame** for depositing the wobble goals. However, as our driving improved, we eventually improved to the point where we could deposit both with about 15 seconds remaining. We also noticed that many teams got a **significant number of points** from hitting the powershot targets. So, we decided to begin aiming for the powershot targets in teleop with the extra remaining time. At first, we tried having the driver **manually align** to hit the targets, but we soon discovered that this was too inaccurate. So, we created code that would initialize and run a **Roadrunner** trajectory in teleop. Now, with the push of a single button, we can **hit all three powershots** accurately in the endgame, scoring us an **additional 45 points per game**.

Summary

Along with our autonomous and our endgame, we will target getting at least 210/220/240 points per game (solo, depending on the randomization). However, we realize that this may not happen every time, so our minimum goal will be **200 points**.