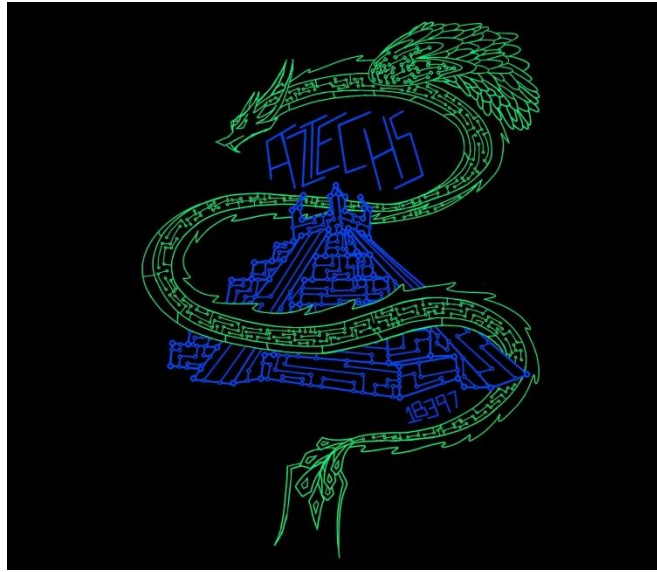


AZTECHS (#18397)

ENGINEERING PORTFOLIO



Inspiring younger and older generations alike

AZTECHS (#18397)

Contact:

Email: ftcaztechs@gmail.com

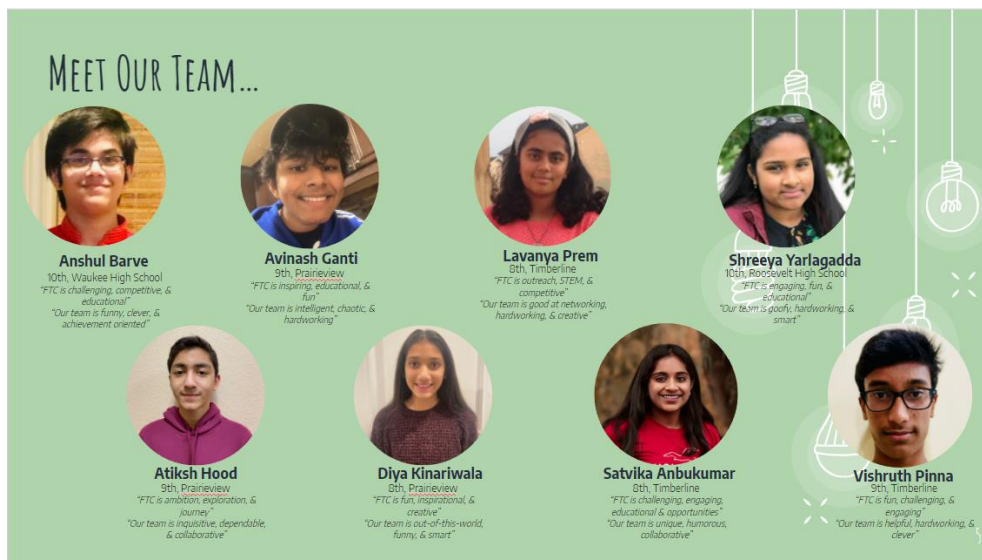
Facebook: [@fllaztechs2019](https://www.facebook.com/fllaztechs2019)

Instagram: [@ftcaztechs](https://www.instagram.com/ftcaztechs)

Phone number: +1 (515)-451-9790

Team History/Introduction

FTC team #18397 Aztechs, is a team comprised of eight 8th, 9th, and 10th-grade students attending Waukee and Des Moines schools. Our team was an FLL team for three years before transitioning to FTC in 2020.



Our team's most distinguishing characteristics are that we are very hard-working, creative, fun-loving, and achievement-oriented.



Coaches & Mentors



Outreach & Fund Raising



1. Celebrasian Festival 2021
2. Roosevelt Trunk or Treat
3. State Fair Lego competition
4. FLL Mentoring
5. Waukee APEX Innovation showcase



@ Celebrasian Festival –

Featured FTC in kids Activities booth. Spoke to ~37 visitors and educated them on our team and Robotics

@ Roosevelt High School Trunk or Treat

Used our robot to hand out candy to ~250 kids and also explained about FIRST and FTC competitions. Received comments as the most engaging Truck on display

@ Iowa State Fair

Participated in the ISU's lego building competition. Showcased and talked about Aztechs team

@ Waukee APEX Innovation Showcase

Talked about FTC to ~100 people and demonstrated our robot

@ FLL Team Mentoring (3 Teams)

Creators of Tomorrow (#?), Micro Mission Bots (#?), Astral Lemurs (#47666)






Shared our experience as a FLL team and talked about FTC

WE THANK ALL OUR SPONSORS FOR SUPPORTING US AND MAKING THIS SEASON POSSIBLE

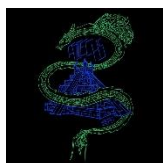


Connect

Our Connect activities involved reaching out to Professionals in the field and learning from their experience.

	<p>Ravi Addanki, Sr. Mechanical Engineer</p> <ul style="list-style-type: none"> - Taught us the importance of structural engineering and Architecture - Discussed control systems
	<p>Salim Chandani (former FTC Coach)</p> <ul style="list-style-type: none"> - FTC Game strategy - Working in Alliances - Using Vuforia & Tensor flow for detection <p>Vijaykumar Anbukumar (Vice President)</p> <ul style="list-style-type: none"> - Use of NFTs for Marketing - Taught us how to drop NFT for our logo
	<p>Barry Nicholson (Director), Nick Embry (Development Manager), Sangeeta Walajabad (VP, Technology delivery)</p> <ul style="list-style-type: none"> - Reviewed our software and taught us the importance of testing - Taught us about the principle of fail-early, fail-often - No need to be perfect at every iteration
	<p><u>Subash Nalluri</u></p> <ul style="list-style-type: none"> - Helped us think about parts of our robot that we overlooked - Discussed the design of John Deere Combines for achieving clearance
	<p><u>Josh Higginbottom</u></p> <ul style="list-style-type: none"> - Josh came on board as a mentor for our team throughout the season - He taught us how to take our ideas and use basic engineering principles to turn these into reality in a methodical way

We were happy to seek help of a mentor Vijaykumar Anbukumaran who taught us how to create an NFT and use it for marketing. We “dropped” an NFT for our logo and intend to pursue selling that as part of our fundraising and sponsorship events.



[Aztechs NFT \(Non-Fungible Token\)](#)

Season goals

1. Build a robot that is capable of doing everything on the FTC challenge field
2. Learn from professionals in the field
3. Teach the robot to “figure skate”
4. Identify ways to work independent of mechanical robot kits (i.e., make or modify parts as needed)
5. Learn to use CAD to 3D print parts for reuse
6. Mentor FLL teams based on our knowledge
7. Learn about how to work in a match as an Alliance

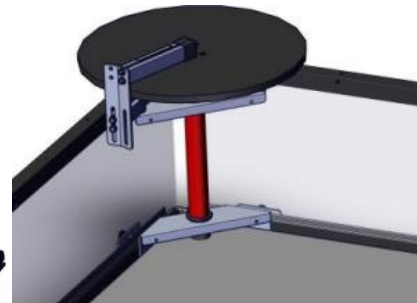
Performance against goals

- ✓ **Goal 1: Exceeded.** Our robot, Marvyn can do every task required on the FTC field for this year’s challenge. We methodically worked through the requirements and designed Marvyn to be a reliable, mechanically intact, smart robot as you will see in the next few sections
- ✓ **Goal 2: Achieved.** We were able to recruit a mentor from Siemens to help us through the design process. We reached out to professionals, both hardware and software, from various companies to have them review our methods and learn from their critical insights.
- ✓ **Goal 3: Exceeded.** When we started this year, we had very little idea of how to move our robot in autonomous. We successfully trained Marvyn using Roadrunner to follow trajectories and use Vuforia and Tensorflow for real-time video based detection. We are particularly proud of participating in the FTC machine learning beta program to take advantage of Tensor flow and Vuforia to train recognition of our custom printed Team Shipping Element. We also, notably achieved our autonomous runs in 10s, with >90% accuracy of completing tasks.
- ✓ **Goal 4: Achieved.** Although Marvyn predominantly uses parts we purchased from GoBilda and REVRobotics, we learned how to modify the metallic parts to fit our needs. Our biggest achievement was learning how to successfully 3D print parts that we would otherwise have struggled to find to spec.
- ✓ **Goal 5: Achieved.** We successfully learned Fusion3D to design parts to required measurements and have mastered 3D printing techniques using slicer software. We convinced our coach and helped raise money to purchase a 3D printer to help with the season.
- ✓ **Goal 6: Achieved.** We connected with three FLL teams throughout the season and taught them from our experience. Proud to claim that two of the teams made their way to the State Championship and have been very gracious in acknowledging our support and insights that helped them.
- ✓ **Goal 7: Achieved.** In 2020, we were a rookie team participating in a remote tournament and felt that there was a lot more to the game than we understood. Through participation in the league events, we learnt how best to utilize our alliance partner and how to contribute best to teams that are more experienced. We have formed successful alliances and are looking forward to learning more and more from all our fellow competitors.

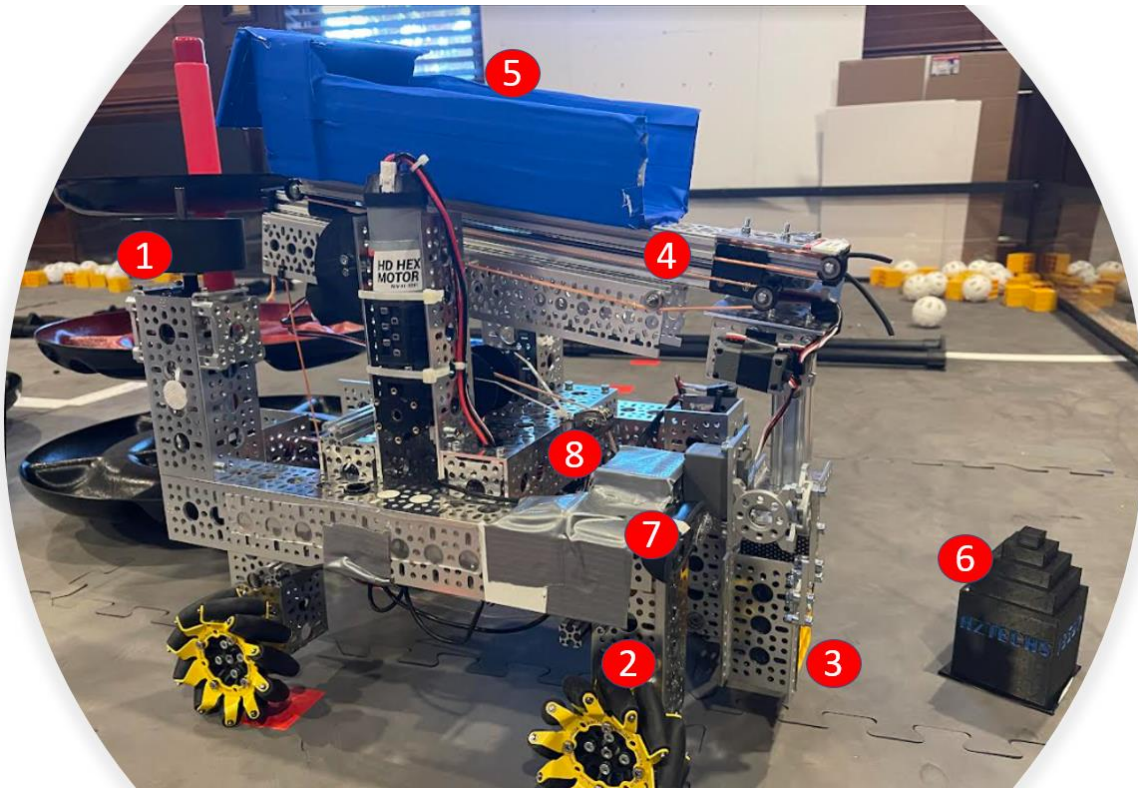
Freight Frenzy - Breaking down the Challenge

Soon after the Season kick-off, we assembled as a team and broke the season's challenges down as follows. This allowed us to write down some requirements as we build our robot

1. **Duck carousel**
 - Need a wheel to spin duck wheels. Sufficient torque and speed to ensure ducks don't fly off the carousel
2. **Climb over barriers**
 - Require high clearance to cross over the barrier
 - Need to be able to pass through track and wall for speedy movement
 - Need to maintain grip on freight when crossing over barriers
3. **Pick up and hold freight**
 - Require a reliable mechanism to pick up cubes and spheres from the floor
 - Keep the freight in possession while moving around to deliver them
4. **Place freight on multiple levels of the Shipping hub**
 - Height adjustable arm mechanism to drop freight appropriately
5. **Place freight while in a tight space (Shared hub)**
 - Navigate in limited space to deliver freight being held in the arm
6. **Cap the Shipping hub**
 - Have sufficient height to go over the shipping hub while holding a Team shipping element
7. **Recognize custom TSE**
 - Recognize custom designed TSE for warehouse level detection
8. **Autonomous navigation**
 - Recognize warehouse level
 - Control mechanisms to raise and lower arm automatically
 - Navigate precisely and quickly to avoid running into other robots



Meet Marvyn – the Freight Master



Marvyn and his Components

S.No	Challenge	Our Solution
1	Spin Carousel	Spin wheels on either side of the robot using torque adjusted motors
2	Climb over barriers	Infinite clearance chassis + Reduced width chassis
3	Pick up freight	Claw to grab freight + Wrist to manipulate it
4	Freight on multiple levels	Linear Actuator + Cantilever arm
5	Freight in tight spaces	Chute for easily delivering freight without requiring a turn
6	Cap the shipping hub	Easy grab TSE design, adjustable height cantilever
7	Recognize TSE	Webcam + Tensorflow Machine learning
8	Autonomous Navigation	Roadrunner Trajectories + Tensorflow/Vuforia Scene navigation + PID encoders driven motors

Solution 1: Duck Wheels

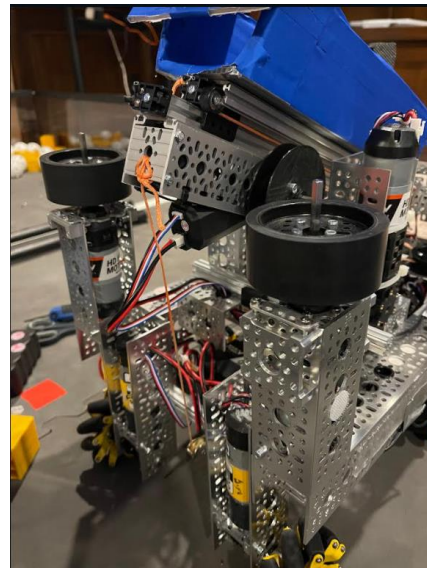
We designed a wheel that would mount on a motor and sit at the back end of the robot to make it easier for drivers and autonomous program to reach and spin the duck wheel.

Some of the problems to consider were:

1. Wheel needs sufficient friction to spin the carousel. We experimented with GoBilda wheels and found one that worked well
2. Going too fast would cause the ducks to fly. We chose a REV Robotics Ultra planetary motor with a 5:1 gear ratio.
3. Later in the design, we added a second Duck wheel on the left side of the robot to allow for easy access while on the Red & Blue sides of the field without having to turn the robot.



Ultraplanetary 5:1



Dual Wheels

Solution 2: Chassis

Iteration 1:

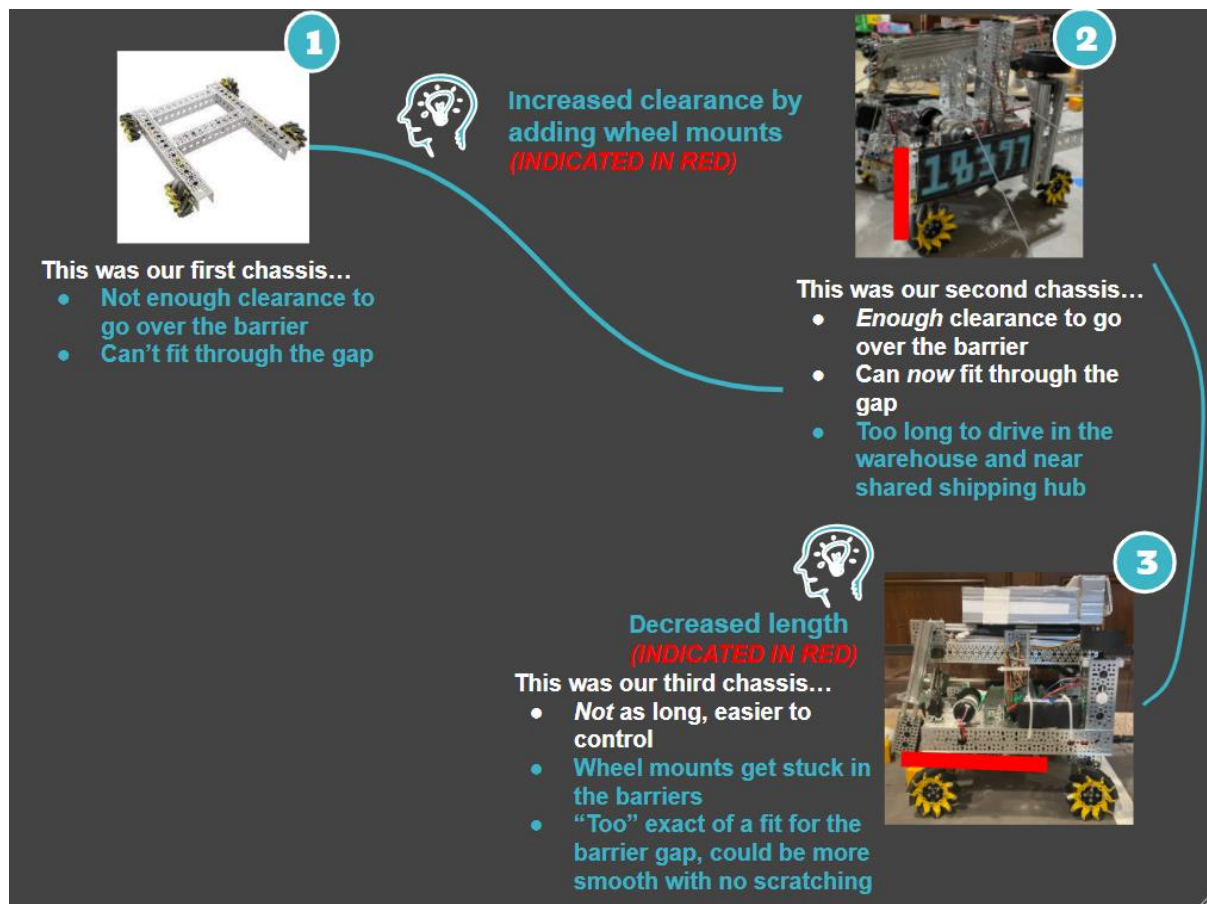
We started with a standard GoBilda strafe Chassis kit. Very soon we realized that this was not going to meet our requirements such as needed clearance to cross the barriers and smaller, less bulky size constraints.

Iteration 2:

Through brainstorming and discussion with our mentor, we figured out that a vertical mount of the mecanum wheels could easily result in “infinite clearance” - i.e., the wheel mount moves with the wheel which means that the chassis cannot get stuck on tracks. The next thing we considered is the overall width. Although we designed to go over the tracks, we believed that fitting within the channel between the track and the wall could add significant benefits. Therefore, we reduced the overall width of the robot in our second iteration.

Iteration 3:

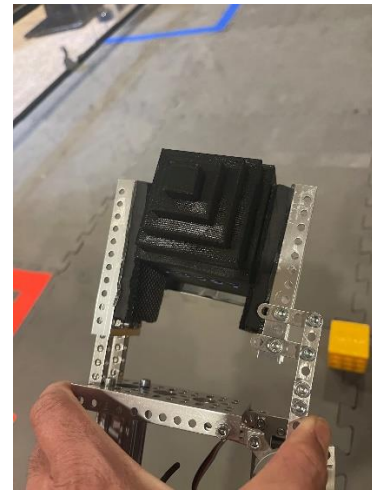
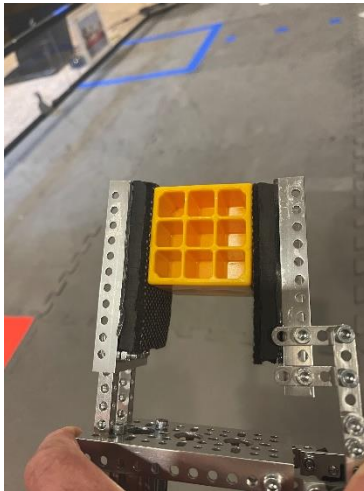
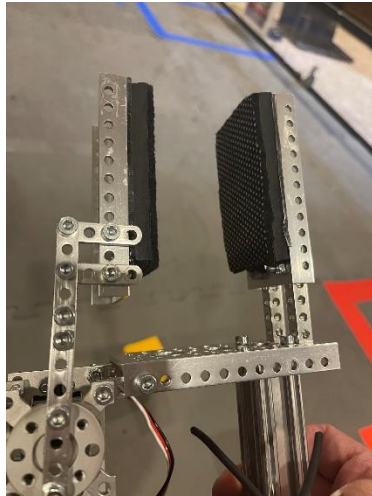
The infinite clearance concept worked almost 90% of the time. However, every now and then during practice and matches we discovered that the wheels do get stuck in the tracks, especially when only one of them is in the tracks. We thought the width of the barriers was causing the issue. However, in a discussion with a John Deere expert, he explained a principle used in designing combines. Per this idea, the lowest point on the robot needs to be higher than the tallest obstacle. As we investigated this, we found out that indeed, our motor mounts were lower than the track cylinders causing the robot to get stuck. We fixed this problem by trimming about 15cm from the motor mounts and that significantly improved the getting stuck problem.



Solution 3: Claw + Wrist to pick up and move freight

Our primary design for pickup and move is a claw. Initially we intended for this to be a free hanging claw that could pick up things from the floor, but through design refinement ended up with a wrist and a claw system as explained below. We chose a claw mechanism rather than a hopper to be able to complete the Capping mission which requires picking up a Team shipping element.

Solution 3a: Claw



S.No	Claw Feature	Purpose
1	Stationary Palm	This palm stays stationary and maintains a straight profile to keep things in place.
2	Movable Pam	This palm is mobile and can open and close to bring things into the grip. This is the primary movable mechanism that helps with grabbing things
3	Servo	The movable palm is mounted on a high speed servo and is controlled by software to open enough to fit upto two cubes next to each other
4	Palm pads	The palms are augmented with rubber padding with a high gripping surface finish to help keep things in place
5	Sizing	The claw is sized such that when the palms are parallel they fit the dimensions of the cube freight element perfectly. With a slight adjustment to width, the claw can easily grab the balls and the team shipping element.

Pros:

- Simple to design
- Firm Grip on objects once picked up
- Can pick up the team shipping element
- Easy to use with simple trigger action

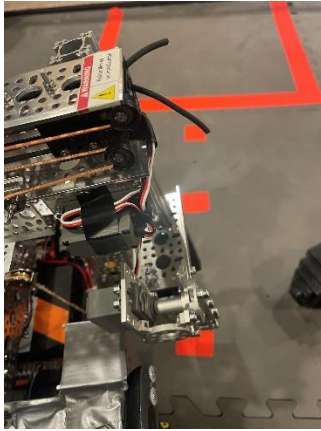
Cons:

- The claw requires heavier/stronger materials to prevent damage during use
- Requires a really long cable (Cable management)
- Needs more time to pick up than other designs

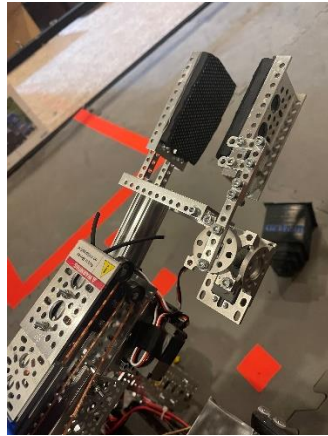
Solution 3b: Wrist

AZTECHS (#18397)

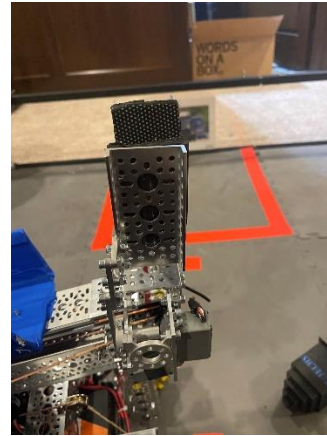
Our initial design of the claw was to simply mount it as a free hanging claw at the end of an actuator. The thinking was that this would naturally allow picking up things from the floor and also use gravity to help drop things on the shipping hubs. However, our first problem was that the free-hanging claw was too easily moved out of place when it collided with objects in the environment. So, to fix that problem we attached the whole claw on a servo so we can move the claw to the ideal angles for picking up and dropping off freight, without sacrificing its rigidity. We used a high torque servo to support the weight of the claw and provide 300 degrees range of motion about the tip of the actuator.



Resting Postion



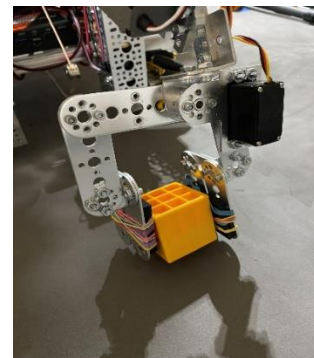
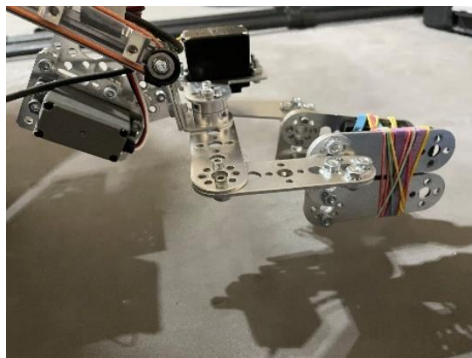
Pickup Position



Chute Drop off position

At the start of the match, the Wrist is in the Resting position facing the claw downwards. Upon Init, it raises the claw parallel to the actuator ready for pick up and drop off. When dropping freight on the Shared hub, the claw swivels to the top to allow drop off in the chute.

Claw & Wrist Iterations:



Solution 4: Varying height extendable arm

Solution 4a: Linear actuator

AZTECHS (#18397)

After finalizing our team scoring plan, it became clear that it was necessary to have a long arm for dropping freight on all three levels of the Alliance Shipping Hub and capping our team shipping element. While we could've just built one long arm, we had to design our arm and robot to somehow fit within an 18 by 18 by 18 inch box. After brainstorming and some research, we concluded that we should build an extendable and retractable linear actuator instead of a static arm. To control the linear actuator we use the left joystick. When the left joystick is pointed up the linear actuator will extend and vice versa.

Goals:

- To give us the vertical and horizontal clearance we needed to drop freight at the top of the shipping hub and cap it with a shipping element
- To have an arm that could fit within an 18 inch box at the start of the game, yet be as long as we need it to after the start of the game.
- Able to support the weight of the full weighted block on the end of the grabber arm

Pros:

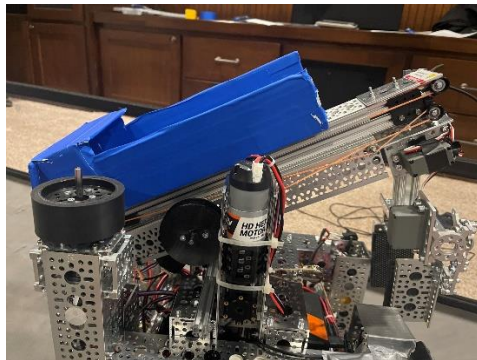
- Allows us to reach targets that are higher and farther away.
- Supports the weight of any freight and does not tilt or affect the robot in any way.

Cons:

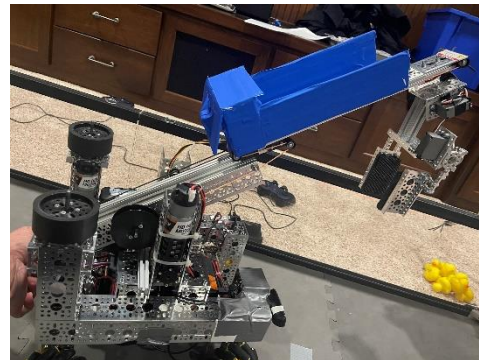
- Cable management gets difficult when the actuator extends
- Over spooling when the driver continues to push the actuator after it can no longer move

Controls:

- To control the linear actuator we use the left joystick.
 - When the left joystick is pointed up the linear actuator will extend.
 - When the left joystick is pointed down the linear actuator will retract.



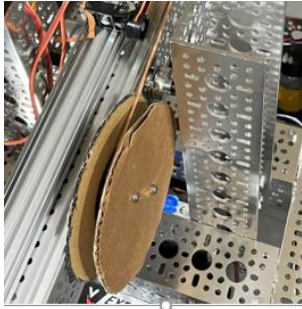
Actuator At rest



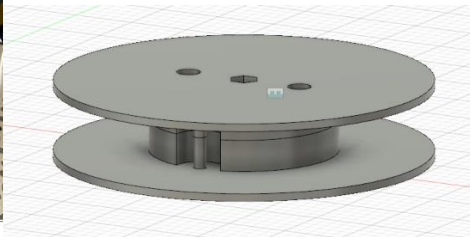
Actuator at Full Extension

CAD Design & 3D Printing:

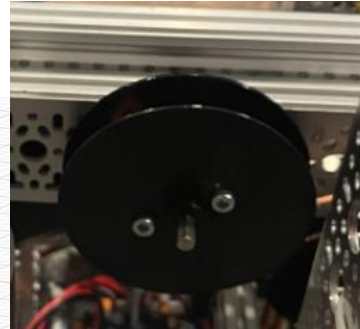
The Linear Actuator is controlled by a Spool attached to a Core Hex Rev motor. The design of this spool required a couple of iterations and helped us learn CAD design and 3D Printing. Our initial solution was to use REV gears with cardboard sidings to build the spool. The cardboard sidings prevented string from running over. Once this prototype proved successful, we used its dimensions to design a spool using Fusion 3D and 3D printed the final spool design.



Prototype



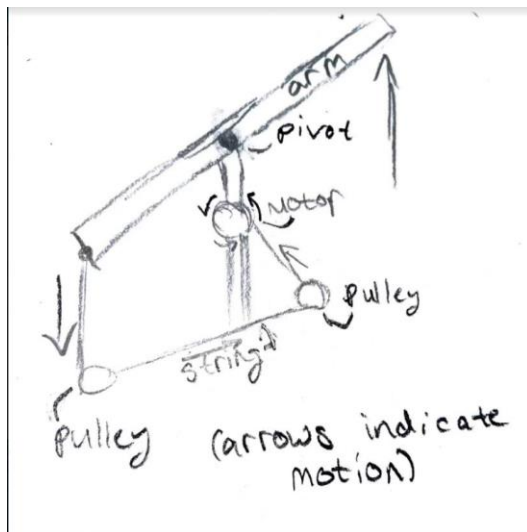
CAD Design



Final Result

Solution 4b: Winch controlled arm

To lift our arm, we use a winch. There is a motor mounted in one of the arms of the Linear Actuator and it uses pulleys and string to move the arm. As the motor winds up the string, the arm moves up, and as it releases, the arm moves down



Weight of claw is weighing down arm, how much torque needed to counter this!

Claw weight = 648g
 Long side = 28 in
 Short side = 7 in

$\frac{28}{7} = 4$

$648g \cdot 4 = 2,592g$ if only using counterweight [NOT VIABLE]

HD Hex motor torque = .105 N/M
 + 48:1 gear ratio
 torque = .105 * 48 = 5.04 N/M

Torque of claw = .648 * 9.8 (Earth's gravity) = 6.3504 N/M

$\frac{6.3504}{5.04} = 1.2592$ ← torque still needed from counterweight

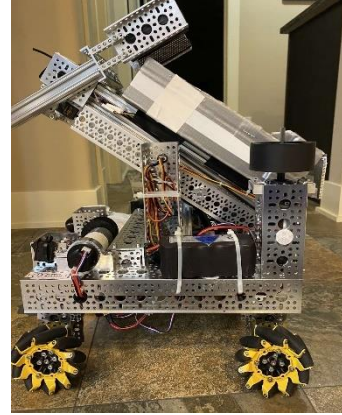
$\frac{1.2592}{6.3504} = \frac{2}{2,592}$

$\frac{6.3504 \cdot 2}{6.3504} = 3,396.5568 \approx 535g$ ← min weight required

Some problems we've faced however, is that the arm would not be able to hold its position if lifted too high, meaning that we had to continuously hold it up if we wanted to drop to the highest position. We countered this by calculating the amount of torque it would require, and adjusting the settings accordingly. There were two solutions that we could think of: a counterweight positioned at the back end of the arm, or a higher torque motor. We quickly realized that a higher torque motor would be required, as we would need a counterweight of over 2kg to sustain the arm at full extension. We ended up using a HD Hex motor, and increasing the gear ratio (37.6:1) to generate more torque (6.04 N-M) . This didn't end up being quite enough, so we compromised and added a counterweight of **540g** at the back, giving us a very precise solution to our issue, while also accounting for any fluctuations in weight we might encounter.

Solution 5: Chute

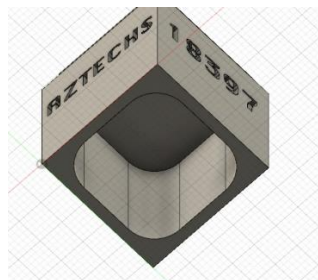
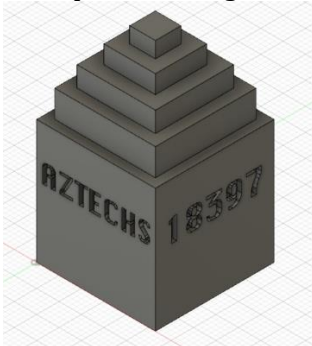
The chute is used to drop freight into the shared shipping hub. We thought of the chute because when we were putting freight in the shared hub we had to turn the whole robot in order to do that, this was taking too much time and effort. The chute helps because we don't have to turn the robot just to drop freight, we can drive backwards into our position and the freight falls into the chute and saves a significant amount of time. We discovered that our original claw design did not have the motion range to support the chute idea, but determined the benefits of the chute made it worth the effort to redesign the claw. We designed the new claw to also rotate 90 degrees upwards so that freight could reach the chute on the top of the robot.



Solution 6: Team Shipping Element

Our design process for the TSE consisted of a few main steps:

- We wanted something to represent our team so we chose a pyramid for the Aztecs
- We sized it to fit the claw when the palms are parallel. We also added a base to the TSE to ensure that it still fits the 3x3x4 minimum sizing constraints
- We wanted the pyramid to be easily grabbable by the claw so we extended the bottom
- We wanted to be able to cap easily so we hollowed out the bottom
- To help Tensor flow training, we colored the text and the numbers blue. This contributed to an improved recognition of above 90% confidence



Programming – The perfect recipe for creating an Autonomous Robot

Our Robot programming is based on using Android Studio for Java. We used the FTC SDK 7.0 version to build our programs. In addition to this, we also learnt and included two different SDKs that greatly helped us build a top-class robot.

Object Recognition Using Tensorflow

Marvyn is equipped with a Webcam placed at a suitable height to help him evaluate the scene in front of him. We use this for object recognition using Tensorflow. The Tensorflow library allows the robot to recognise objects on the game field, such as freight, ducks, and field markers. We made a sample program that detected ducks and used the FTC Dashboard to analyze the camera feed to identify the bounding boxes of the ducks and estimate the position of the Warehouses based on that.

Once we were successfully detecting ducks, we then moved on to train Tensorflow to recognize our team shipping element. We were lucky to be able to participate in the FTC Beta program for Machine learning. After 3D printing our Team Shipping element, we uploaded videos of the shipping element taken from various angles to the FTC machine learning site. Using the built-in OpenCV program, we marked the TSE in all the frames of the videos we uploaded and used that to create a trained model of the TSE. We then used this in our program to identify the TSE. Through our training and careful choice of TSE design, we have been able to achieve >90% confidence level in recognizing our TSE.



The world through Marvyn's eyes

Trajectory Sequence for Autonomous Navigation

In addition to using the FTC SDK and its OpModes, we also imported and integrated the Roadrunner library provided by ACME robotics. Roadrunner allows to move the robot using Paths & Trajectories specified by robot position, heading, and velocity. This way of

navigation greatly improves robot speed, accuracy and most importantly simplifies autonomous navigation. With the use of roadrunner trajectories, we were able to accomplish all our Autonomous runs in under 10-12s.

Key Concepts that we Implemented

- Extensive PIDF tuning to help Roadrunner program understand the physical and technical constraints of the robot
- Used FTC Coordinate system to carefully locate the robot and plan trajectories
- Trajectory Sequences using Temporal Markers to execute the entire Autonomous trajectory in one go
- Debugged the trajectory positions to fit width of game field

Building Trajectories:

Through this **simple** Trajectory sequence we are able to achieve our **entire autonomous program** which includes navigating to the shipping hub, dropping the preloaded element, navigating to the duck carousel, delivering a duck and parking in the Storage unit. The total execution time is **12s** and the robot moves **at 45 inches per second** speed.

```
void buildTrajectories()
{
    mrvBlue2 = marvyn.mecanumDrive.trajectorySequenceBuilder(marvyn.blue_2_pose_estimate)
    //drive to hub
    .lineToLinearHeading(marvyn.blue_2_shipping_hub_pos)
    .addTemporalMarker()->{
        FreightDropOff(mrvWarehouseLevel);
    }
    //Release the Claw
    .addTemporalMarker()->{
        marvyn.The_Claw.setPosition(Claw_Open_Pos);
    }
    .waitSeconds(0.5)
    .addTemporalMarker()->{
        // retract Linac
        marvyn.setTargetPosition(Mrv_Robot.MrvMotors.LIN_AC, 0);
        marvyn.setPower(Mrv_Robot.MrvMotors.LIN_AC, 1);
        while (opModeIsActive() && marvyn.getCurrentPosition(Mrv_Robot.MrvMotors.LIN_AC) < 0) {
            idle();
        }
        marvyn.setPower(Mrv_Robot.MrvMotors.LIN_AC, 0);
    }
    //enter warehouse
    .lineToLinearHeading(marvyn.blue_warehouse_enter_pos)
    .lineToLinearHeading(marvyn.blue_warehouse_pos)
    .lineToLinearHeading(marvyn.blue_park_pos)
    .build()
;
return;
}
```

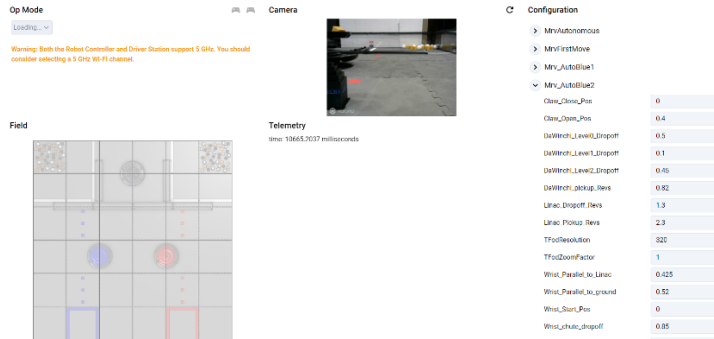
Autonomous Program for Blue 2 position

Tools used for Programming

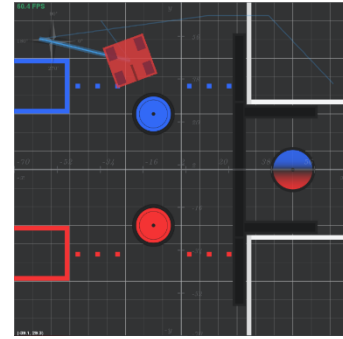
We primarily used 3 tools to help with our programming tasks

- **FTC Dashboard**
- **MeepMeep for Trajectory planning**
- **Github as a Source code repository**

AZTECHS (#18397)



FTC Dashboard for Testing and monitoring



MeepMeep

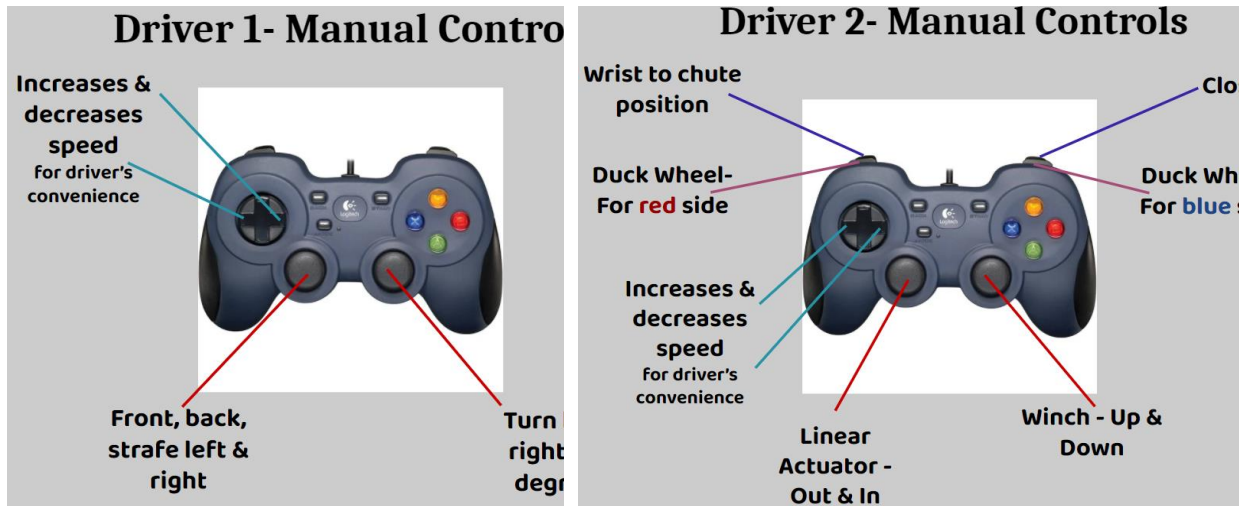
The FTC dashboard allowed us immensely in debugging our computer vision and trajectory programs when running the robot. In addition, we learnt a new trick using Configuration variables which would allow us to enter values in the dashboard and rerun the robot without having to recompile our code. This method increased our productivity greatly when tuning the winch and linear actuator positions for the autonomous programming.

MeepMeep was a handy little tool we picked up to help plan our trajectories. We utilized this tool to map the coordinates of the field and visualize the robot trajectory without having to directly run the robot each time. This was greatly beneficial as it allowed us to quickly estimate the needed coordinates. After MeepMeep proved out the trajectory, we later fine tuned it when running the robot with minor precision adjustments.



Github Source code control Commit History

Manual Control



Controls

- **Speed control**
 - The problem was having the robot too fast made it hard to control things when aligning pickups and drop offs
 - **Our solution was having a control that lets the driver adjust the speed of different motors/servos when necessary**
 - We used the d-pad because it is a button control. You can program it like a scale. (1-10 speed)
- **Driver controls & Winch/Linear Actuator**
 - These controls are essential for our runs to work.
 - **We needed these controls to be easy to use, since they are the most used controls**
 - We used the joystick because these movements are continuous. Meaning we need to move forward/extend/ pull down but also do the opposite (move back/ pull in/ push up)
- **Duck Wheels**
 - We used to have the duck wheels where you click once to start and click once to stop. But that would result in spending a lot on reaction time and waiting a couple seconds for the stop.
 - **We decided to make it where you press and hold to rotate, and release for it to stop.**
 - We used the bumpers because they are binary (on/off)
- **Wrist**
 - We needed the wrist to come back and drop freight into our chute. But also be used for picking up freight

Control Award Sponsored by Arm Inc. Submission Form

Team # 18397	Team Name: AZTECHS
--------------	--------------------

Autonomous Objectives:

1. Detect Team Shipping Element for warehouse level.
2. Drive to pre-determined safe position to drop freight, deliver duck and park in warehouse or storage unit
3. Raise winch, extend linear actuator, move wrist and open claw to drop freight
4. Reach the carousel and engage duck wheel to deliver duck
5. Park in storage unit or warehouse and prepare for manual operation

Sensors used:

1. **Webcam** for detecting Team Shipping element. We used the live feed from the webcam to allow Tensorflow to detect the shipping element and return the coordinates of the bounding box.
2. **Motor encoders:** In-built PID controllers on drive motors for RoadRunner PID tuning. Motor encoders to raise the winch precisely and extend the linear actuator.

Key Algorithms:

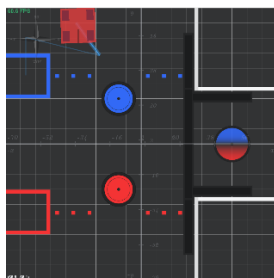
1. **Tensor flow** to read the barcode. Our webcam is positioned to see two of the barcode positions. If the bounding box of the detected TSE is on the left hand side of the picture then we consider this to be the middle position. If the bounding box is on the right side of the image, we consider this as the top position. If the camera cannot see a TSE, we consider this as the bottom position. We first programmed using the pre-trained duck model. Eventually, we used FTC Machine learning portal to train our TSE model and replaced the model in our software. See Autonomous section in Portfolio
2. **Road Runner Trajectory sequence:** Using Roadrunner, we precisely calculated the coordinates and heading of the robot. This allowed the robot to reach its destination precisely and perform actions seamlessly in a super quick and reliable manner.
3. **Vuforia for Position detection:** We attempted using Vuforia to read the navigation images in order to help determine our position. For our Roadrunner trajectories to work, we needed a very precise initial pose estimate that helps it recognize the field position. To do this, our idea was to read one of the navigation images and calculate our starting position based on the OpenGL transformation matrix returned by Vuforia. We were successfully able to get this working, however, we found out that our camera could either read the duck positions or navigation images (due to focus range) but not both at the same time. We attempted to mount another webcam, but this will require careful planning on hardware (ex: mount it on a servo etc.). We did not successfully integrate this for this go round, but this is the next challenge we would like to undertake.

Driver Controlled Enhancements:

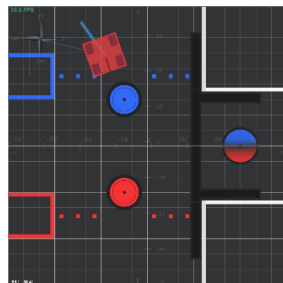
For manual control, we have the standard forward, backward, strafe, turn routines programmed on the two joysticks for the driver controller. We also programmed a speed control so the driver can change it as and when needed. Our drivers use this for buzzer beating parking at the end of delivering the duck wheels. On the operator controller, we programmed triggers to open and close the claw for pickup and the two joysticks to extend the linear actuator and raise/lower the winch to assist with pick up. All of our mechanisms can be speed adjusted using the dpad as and when needed. We also programmed a claw drop off position for the shared hub such that drivers can easily drop freight into the chute for the shared hub.

Autonomous Navigation – Trajectories

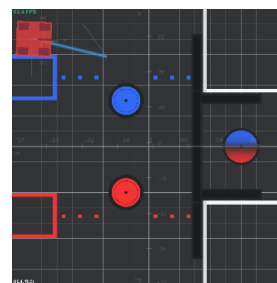
We have two different positions on either side of the field. **In position 1**, we start closer to the carousel, deliver freight, deliver the duck by spinning the carousel and park in the storage house so as to avoid robots that either can't move or park in the entry way of the warehouse. At each of these locations, we precisely specify the (x, y) coordinates in terms of the field coordinate system and the robot heading direction.



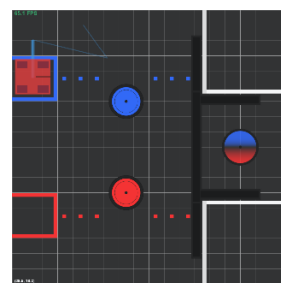
Starting position



Stop to deliver freight

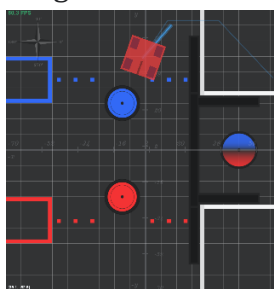


Spin Duck position

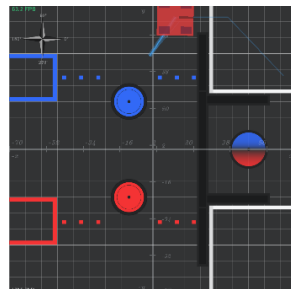


Park in Storage

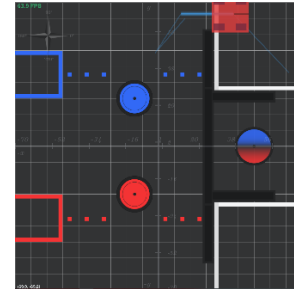
When Starting in **Position 2**, we start closer to the warehouse. We deliver freight, and then enter the warehouse. To enter the warehouse, we go through the channel. We first stop at the entrance of the warehouse to allow roadrunner to precisely position the robot to head into the warehouse such that we can avoid the tracks. Then we move forward bearing that heading into the warehouse. We also then move over to the far end of the warehouse facing the freight. This gives us two advantages, allows our partners to park in the warehouse from the other side and also positions the robot such that our drivers can begin delivering freight to the Shared hub.



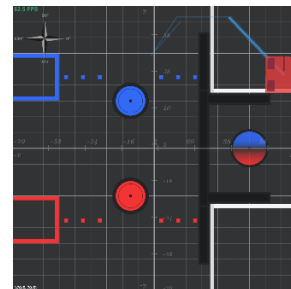
Stop to deliver freight



Align warehouse



Enter through channel



Park in the far corner

We achieve all our Autonomous functions in under 12s to give the other partner ample time to accomplish theirs

