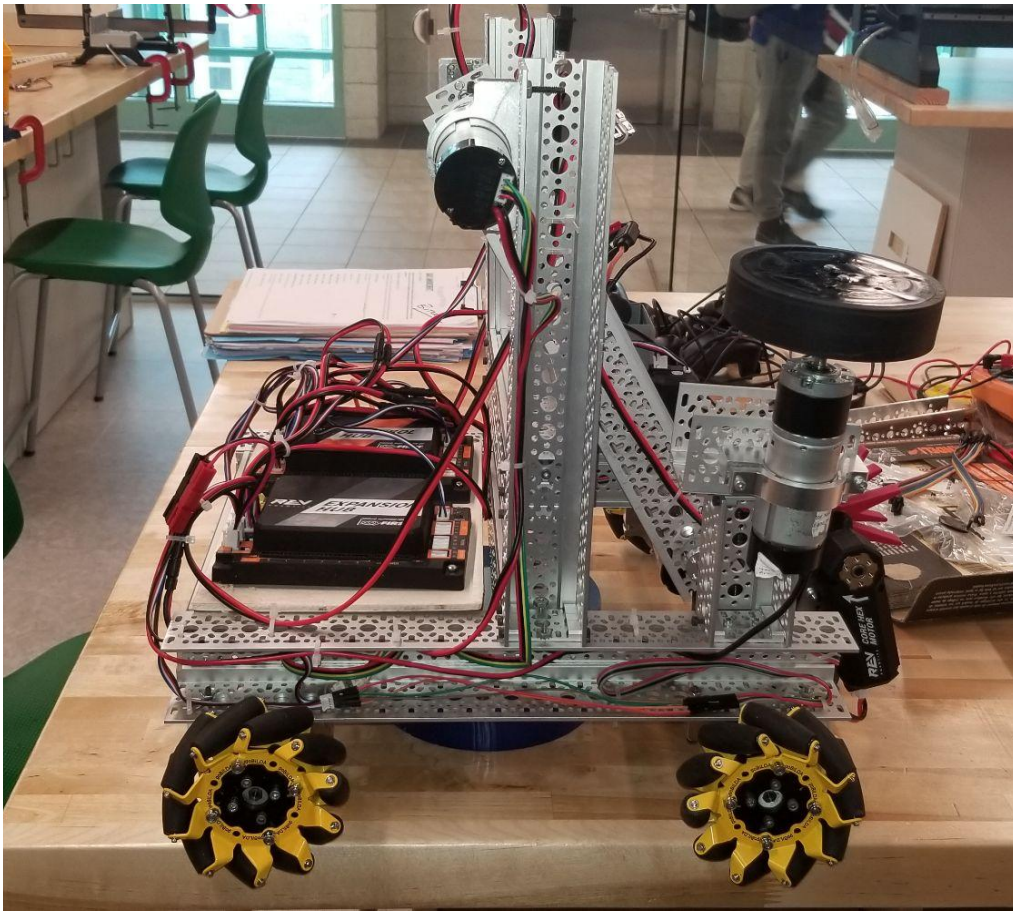
FTC Team 755 “Delbotics” Engineering Portfolio

Delbarton School

Jack Finning, Edward Malhan, William Du, Cortland Forstner, Michael Xu, Jack Wade, Raymond Xu, and Christian Barravecchio



Team Introduction

This year, our core team is comprised of Jack Finning, Edward Malhan, William Du, Cortland Forstner, Michael Xu, Jack Wade, Raymond Xu, and Christian Barravecchio. Jack Finning has participated in robotics competitions for two years. Still, everyone joined the robotics team for the first time this year. Though the group is primarily new, we all share a common love for engineering and bring unique skills to the table. We're excited to take this opportunity to share our robot for the 2021 First Tech Challenge with you!

Strategy

As always, our robotics season started with a meeting where we watched the introduction videos to the challenge. We familiarized ourselves with the rules and point distributions and began to form a strategy. We determined that speed, reliability, a targeted and straightforward design, and minimizing user error are the key characteristics of a winning robot. The core idea of our design was to have a robot that had an arm with an intake system on the end, which would allow us to pick up freight and move the arm over as our robot moved towards a shipping hub, thereby saving time. During the autonomous period, we prioritized going for the duck wheel. During the driver control period, we decided that going for the shared shipping hub with the goal of tilting it was the best point-scoring strategy. In the endgame, we decided that we would either go for the duck wheel or continue placing scoring elements in the shared shipping hub depending on what the other team on our alliance plans on doing.

Chassis Design

Our original chassis design incorporates two parallel beams connected with two lateral beams placed between the first two pieces. We mounted our motors to the bottom of our chassis to give us extra clearance when going over the barriers.

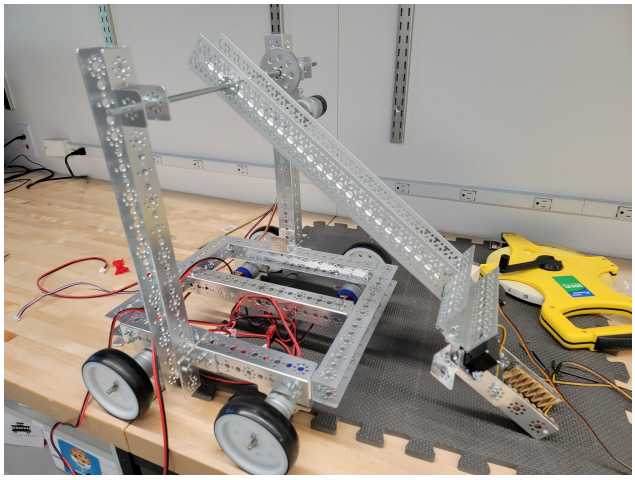
In our original design we placed the lateral beams at the very front and back of the robot, but this did not leave enough room for our intake system, so we performed robot surgery and moved the front horizontal bar farther back, making our robot more compact.

After our first competition, and though we had a very good point/match ratio, we decided to completely rebuild our robot and built a new chassis. We felt that there were enough good modifications that we could make to justify the complete rebuild. These included a stiffer chassis (just as car manufacturers do), newer and better wheels, a hex arm axle system, a new pivotable intake system, a dual arm lifting arm and numerous software modifications. We used the same layout as our first design, but built it using stronger, higher quality parts so the robot would be sturdier.

Intake System

Claw mechanism

We early on thought of attaching two servos to the end of our arm to grab the freight. This claw mechanism can be seen in the image below:



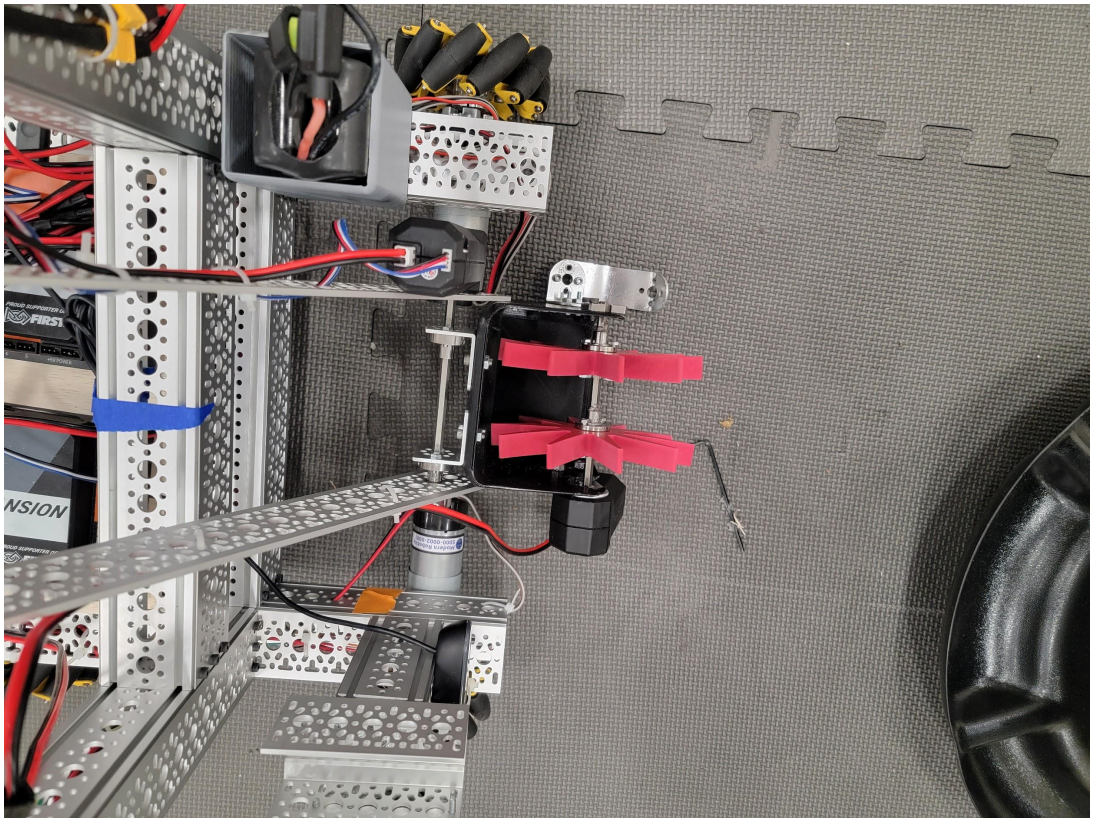
Originally we had two pieces of metal attached to each servo, but because metal wouldn't grip the freight well enough, we decided to add rubber strips to the inside of each metal plate. However, when we tested it, we found that it was difficult to synchronize both servos, which made grabbing freight difficult.

Entrapption Stars

Instead of the servo arms coming from the side, we decided to completely overhaul our intake system and opt for the use of entrapption stars. We 3D printed a box and mounted a motor to it to spin the entrapption stars and take in freight.

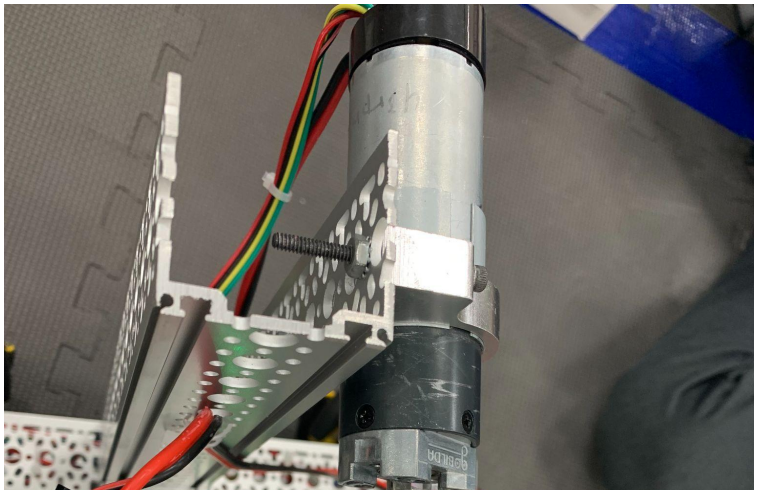
This system was considerably easier to use than the servos and was much more reliable. The problem with our original intake design was that it would not take in the Wiffle balls easily and they would get stuck in the back of our intake system. In our first competition, we managed to temporarily fix this problem by placing hot glue on the bottom in a way that prevents the spheres from going too far in. (On our new robot we redesigned the intake system so it could more easily pick up freight and let the entrapption stars slide, which meant picking up spheres was no longer an issue).

We went through multiple iterations of 3D design for the intake system to accommodate for the different sizes of freight and entrapption stars. The 3D printing was mostly done by team member Michael Xu. Iterations included sizing, mounting, and structure changes.



Arm

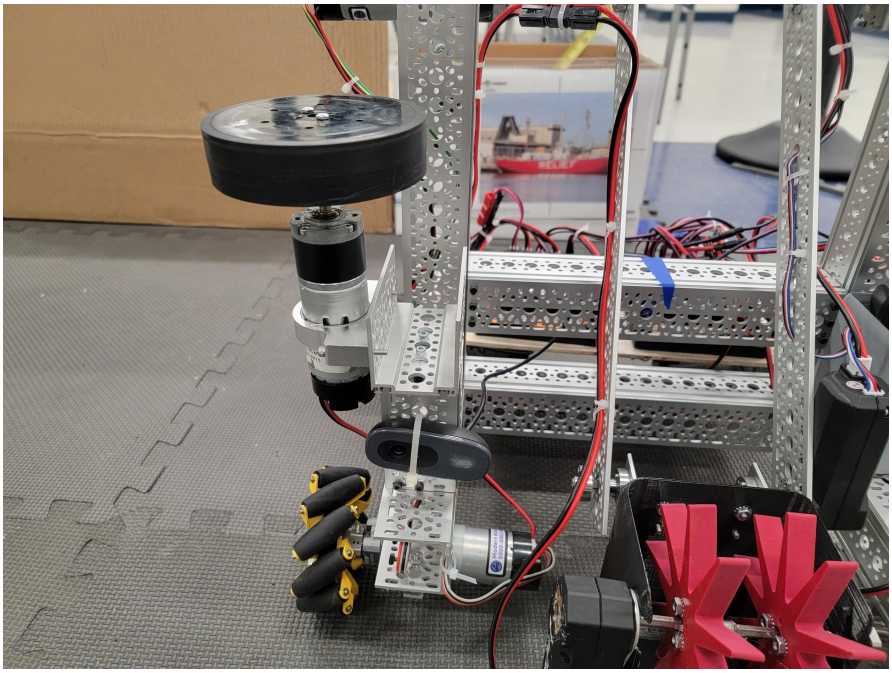
When choosing to design an arm mechanism to pick up freight, we agreed the best strategy was to go only for the shared shipping hub. In order to save time when traveling to the warehouse from the shared shipping hub, the arm travels over the top of the robot. Therefore we do not have to rotate the robot but simply move in a straight line. Originally we used a D-shaft motor with a 60:1 gear ratio. To attach this motor to the arm axle we connected two D-shaft hubs with screws. With this design we quickly discovered from testing that the motor produced too much torque for the D-bore hubs to handle. The metal began to deform around the axle which causes the arm to slip and not rotate with the motor shaft. The D-bore hub's set screws came loose as well. With this major problem we had to make a change to hex parts. We hauled in a new 139:1 gear ratio motor. The arm needed a massive increase in torque to handle the weight of the arm. The majority of the weight on the arm is located at the end of the arm which increases



torque even further. We got hex hubs and a new hex axle. The benefit of hex parts is that they have a lower tendency to slip due to the shaping of the metal bores and axles. In hindsight, it was a great decision to switch to hex parts. We have not experienced any slippage since the change.

Duck Wheel

To spin the duck carousel we vertically mounted a motor to a corner of our robot and attached a wheel to it. During the round, the robot backs into the corner with the carousel and presses the wheel against it, spinning the carousel when the motor spins the wheel.



Software

Raymond's opinion

Software was fun to work on. I think by far we spent the most time on getting the arm to work properly. While the solution seems simple in hindsight (attaching a different button on the gamepad to each position we needed to go to), it actually took quite a bit of trial and error to reach the conclusion.

Originally (back when we were still using a claw) we naïvely tried using a joystick to control the speed of the arm, in a similar way to how a joystick controls the speed of the wheels. Unfortunately, this led to the arm slamming down on the other side due to gravity being an

inconvenient challenge in the competition arena. To counter this, we decided to add a button which rotates the motor in the opposite direction with low power to act as a brake. Unfortunately this also meant that depending upon the rotational position of the arm it might either not be enough power or too much power! Balancing between forward and braking by hand essentially made the robot a giant haptic actuator, and after some deep thought we decided that that may be funny, but that we preferred having a more stable robot for the competition. While we could probably have made the robot more stable given more time, we found the next idea (our current system) much more promising and switched to using that.

We eventually reached an input system not very different from our current one, where we preset positions that the arm was supposed to go to and bound a button on the controller to each (though the analog trigger was still used for minor adjustments). To do this, I wrote a quick program that did nothing but print out the encoder readings for all the motors. We noted these readings down and, in the driver code, told the motors to go to these positions whenever the associated button on the controller was pressed. This method allowed for the driver to press a button and not have to focus on positioning the arm to the right place manually.

By this time, we switched from the claw to the entraption star based intake system. One issue we noticed was that it was hard to both pick up the freight and unload it to the correct place on the shipping hub due to inconvenient angles. We thus added a servomotor to control the angling of the intake system, with different angles for each function-position the arm was pre-programmed to go to (for example, it was angled downwards when intaking from the floor). Because of the aforementioned button-to-position system, this actually didn't add any complexity to controls from the human perspective.

In terms of autonomous code, we had several ideas that we wanted to pursue. I thought of dynamically calculating paths via A* by making every tile a node and creating a densely connected graph by connecting all nodes that have a "line of sight" between them (a straight line path not obstructed by a robot or game element). However, this method relied on having extremely accurate positional data and knowledge of the current field state. We ended up not using this idea because we had a huge amount of trouble getting accurate positional information, and we ran out of time doing so, forcing us to default back to a much more rudimentary approach.

Another idea we had borrowed from the utility code for the arm that read out encoder positions. We manually drove the robot around and dumped encoder positions every 10 milliseconds from all the motors and we then replayed that data afterwards. Unfortunately, we had some last minute issues with data replay (namely, the power levels sometimes got misapplied causing the robot to jerk around). I really wish we pursued this idea earlier, as, had we more time, I feel like this would have been a decent option.

Will's opinion

Software was a place where we got to be innovative. With relatively new technology such as machine learning and mapping perfected and utilized by other teams, we sought ways to improve our own robot's software by using innovative new code.

We started off simple, with the most basic functionality of a robot, such as wheel control, intake and arm movement, as well as hard-coding the autonomous movement. This stage allowed us to identify and fix basic mechanical and coding problems; for example, when the robot's wheels turned in a certain way, they would fall off and disassemble. In addition, the weight on the arm prevented it from transitioning smoothly over the robot, and there were some minor problems with the intake at first as well. Nevertheless, all of these were solved as we moved on to the final prototype.

Special software was needed for a few new features on the robot built to give us a competitive edge. For example, a pivot point for the intake system allowed it to cover more angles and unload the cargo at more convenient angles. We added sideways and diagonal code for the mecanum wheel movement, and adjusted the speed at which the duck spinner spun so as to spin the duck as fast as possible without making it fall out of the arena.

As the competitions get more and more competitive, we tried to come up with more innovative ways to code autonomous. Since we struggled to train Tensorflow to recognize our team shipping elements, we thought of another identification method, which was to recognize the tape color. If one of the pieces of tape was missing, that would mean that the team shipping element was there. Additionally, we experimented with the idea of creating preprogrammed paths by saving the positions as we manually controlled our robot. With more time, we believe this would allow our robot to take the most efficient routes possible. Lastly, we thought of creating code that could avoid obstacles by making it stop automatically before it hit them. For example, there may be a radius around the team shipping hub that automatically stops the robot when touched. Unfortunately, this requires positional mapping which we did not have time to do. However, we did have different ideas on how to accurately do this. One was to detect the distance to nearby images, but this proved inaccurate. Another idea was to add distance sensors to the robot and detect its distance to the edge. While all of this proved promising, we would need another couple of weeks to get it into applicable form.

Coding the robot was pretty fun, non-intensive work that felt rhythmic once you got into it. While technical issues and ClassNotFoundExceptions made it take a bit longer than expected, overall it was satisfying to see code translated into real world application and movement.