ECE 118 Final Project Report

ECE 118: Mechatronics

Robert Box
Harrison Callahan
Nikko Echevarria

June 10, 2022

Contents

1 Introduction

Our team made an octagon shaped robot with three levels that enable modulation and physical separation between parts of our robot. It moves using four omni wheels in X-drive to mechanically solve the problem of being flush with a tower we are circulating. Ping pong balls are gravity fed and using a RC servo we move two nails that allow shooting one ball at a time. To find the right hole we designed our robot to have two tape sensors and two track-wire sensors. Having two of each sensor prevents faulty readings in once sensor to mess up our robot, making ball shooting very consistent. The beacon detector can find a tower across the arena, allowing us to ignore tape and always drive towards a tower. With filtering and physical shielding we do not have to worry about detecting incorrect frequencies.

2 Robot Layout

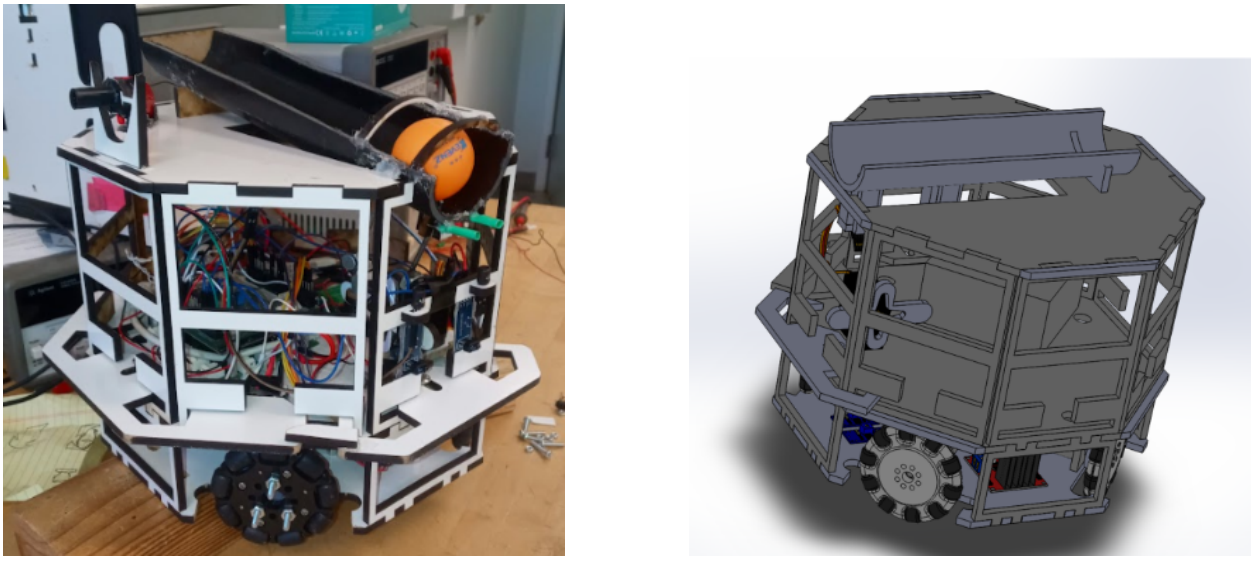

Figure 1: Finished design of robot in real life and simulation

2.1 First Floor

The first floor contains four dc motors with wheels, two x-drives, and two tape sensors. The motors create the letter X with their layout and control four omni-wheels that allow movement in any direction. The purpose of omni wheels was to drive into a tower we are rotating to become flush for easier shooting. Both X-drives are in between two motors so that the wire length between the X-drives and motors is as short as possible. Two tape sensors were placed in the center of the robot to detect tape on the ground. This modification was

made after min spec to be able to rotate towers near the edge of the arena. Previously, the tape sensors were at the edges, making the robot unable to circle towers near tape.

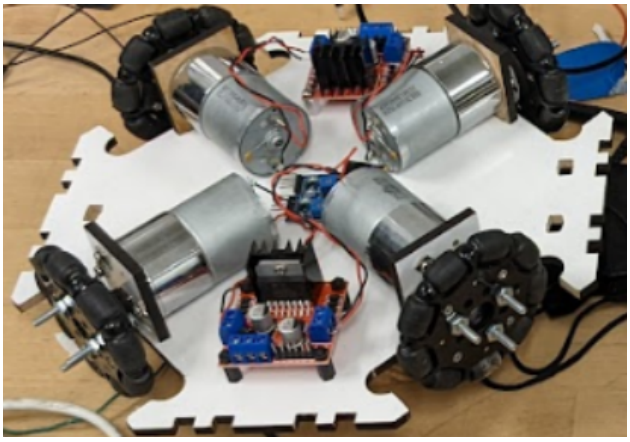

Figure 2: First floor containing X-drive and tape sensors

2.2 Second Floor

The second floor has the uno, battery, shooting servo, tape and track-wire sensors, bumpers, and voltage regulators. We put circuits on a different level from the motors to reduce noise getting into our circuits. Uno controls all the peripherals on the robot. There are holes in all the floors for easy wire management. Two track-wire sensors are on one side of the robot that is flush with the towers. The sensors make sure we are next to the correct side and we have two to prevent having a false reading. There are also two tape detectors to find tape on the side of the tower. We have a front bumper to know if we drive into a tower and a side bumper that keeps the robot's side close to the tower when circling.

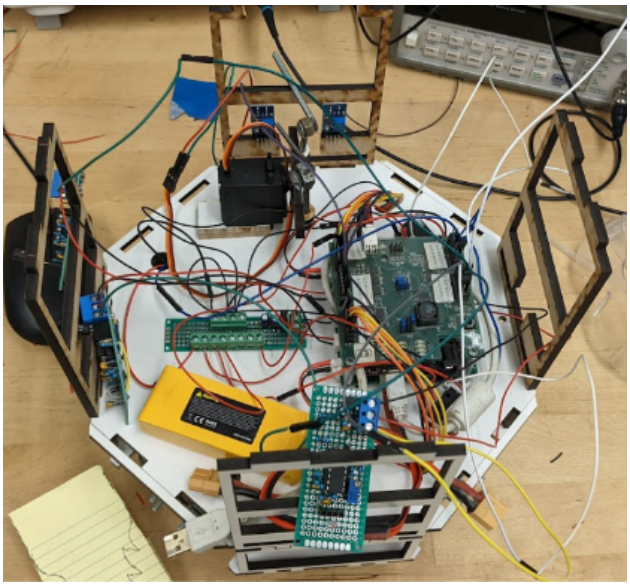

Figure 3: Second floor containing microcontroller and sensors

2.3 Third Floor

The third floor has a half pipe to hold ping pong balls, beacon sensor, and the on switch. The pipe can hold up to six ping pong balls and shoots one ball at a time using a servo on the second floor that moves nails up and down to shoot the balls. The on switch is connected to the uno, bypassing the built in switch and making it much easier to turn the robot on and off. The beacon sensor is in a shield made out of a banana plug and electrical tape. It aims at beacons that are eleven inches in the air so we put the beacon sensor on the highest floor.

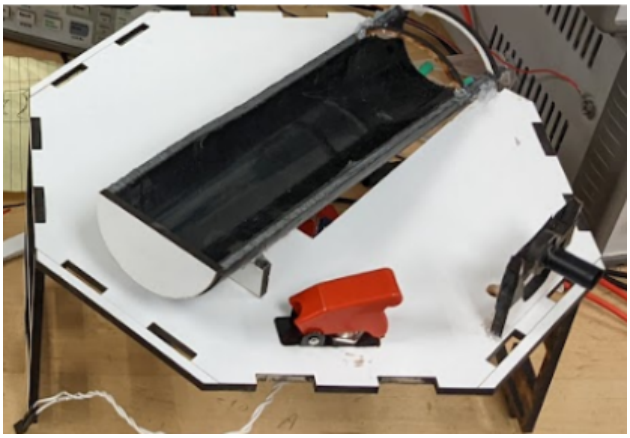

Figure 4: Third floor containing ping pong storage and power button

3 Track-wire Sensor

All of the circuits were prototyped and soldered within the first week. The first of those being made was the track-wire sensor. Our robot uses two track wire sensors which was a design choice to help with the fact that on the corners of the tower while shooting or turning we would be able to read the track wire even though it is on the other side. Having two made it so even when turning around the tower, the only time both are ever active is when you are flush with the correct side of the tower. We all had basic sensors from previous labs, but two people from other groups (Leo and Ben) had the idea to try to make a slightly better one that only uses one chip and has a strong filter to deal with noise, which is something we probably have a lot of because we use four motors. They finished the prototype on a breadboard and made the basic soldering layout for the perfboard which I then joined to help them learn to solder and actually make the perfboard. The end result was this track wire detector:

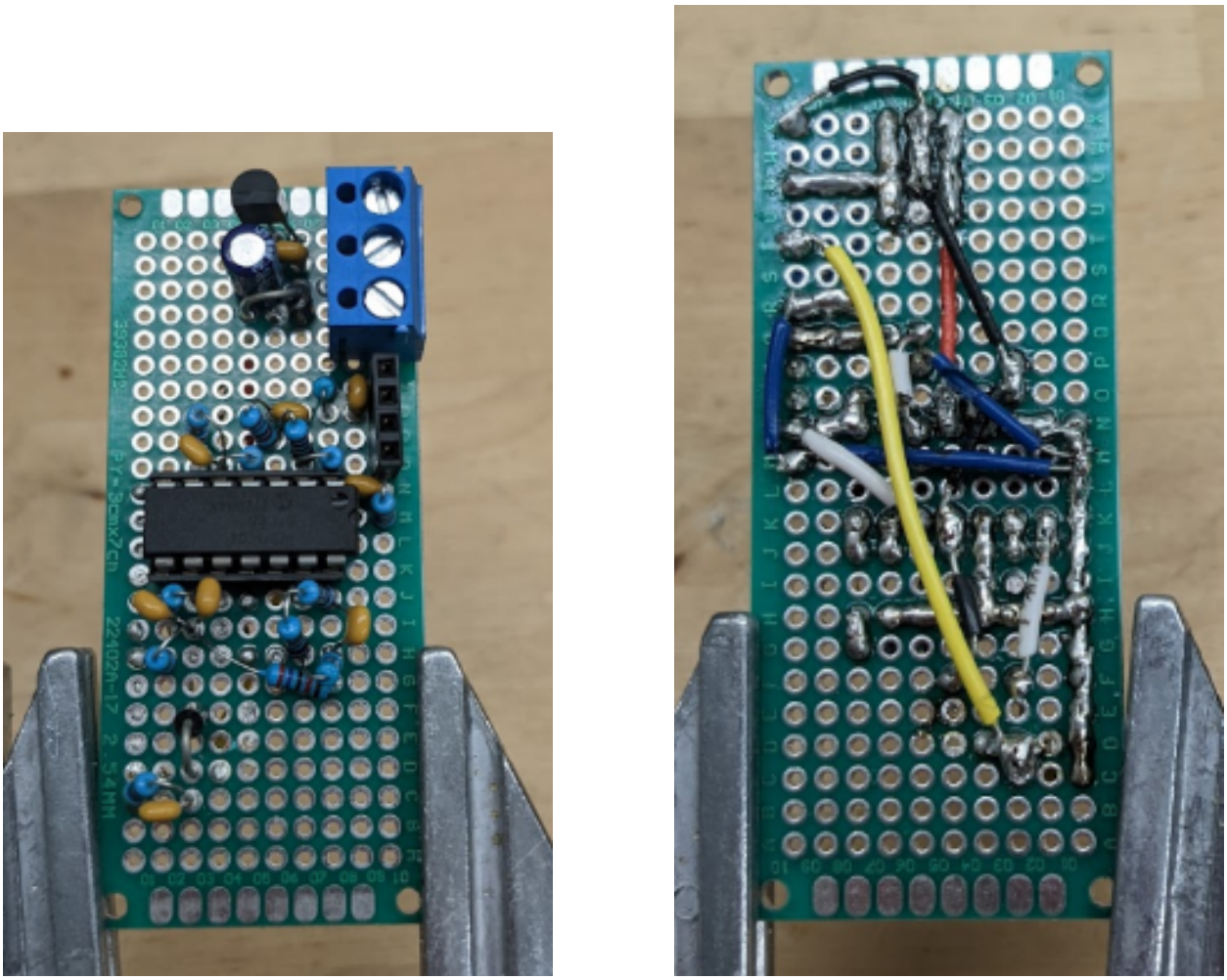

Figure 5: Initial circuit of our track-wire detector

Now this track wire worked just fine, but because of the layout of it making one was actually quite difficult and impractical taking basically the entire day to make. Also the back looks ugly, and looking pretty is the most important thing for a circuit. Since we needed to track wire circuits, this time I sat down with Ben and we worked on a much more practical design that we could make much faster. The drawing of the final track wire design looks like this:

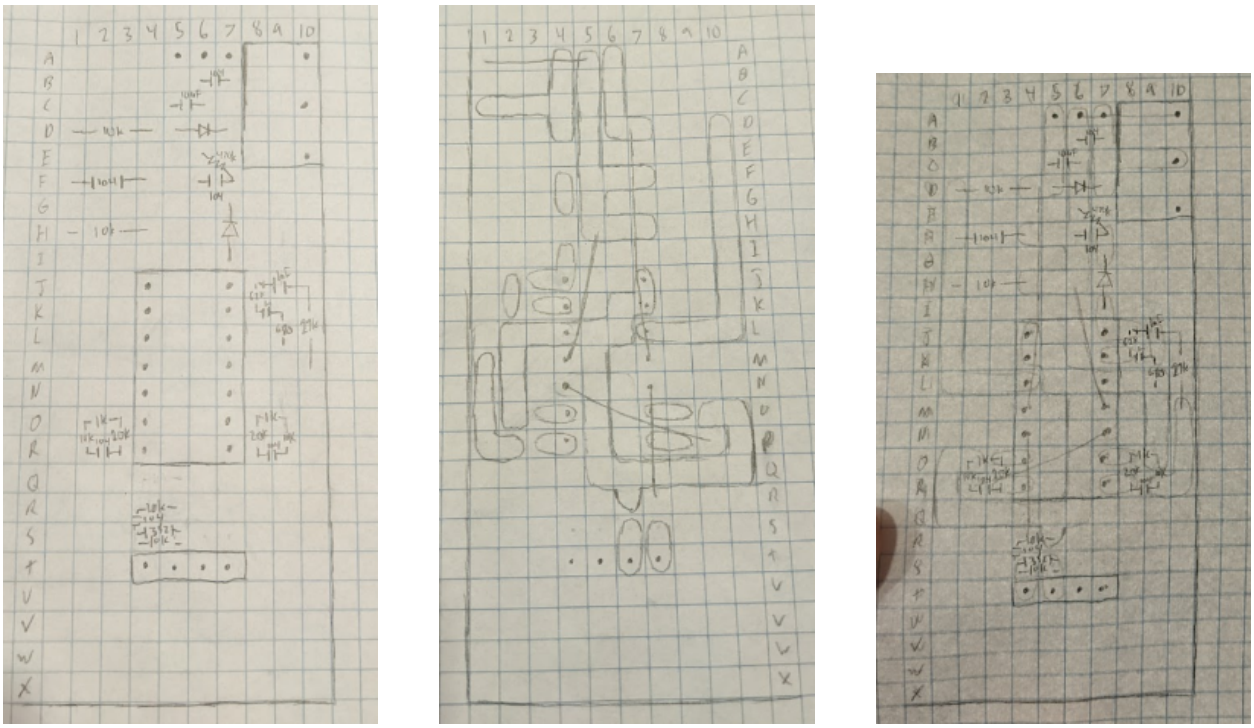

Figure 6: Finalized design of our track-wire detector

I like to make all of my perfboard layouts on paper so that I can draw the front and back of the perfboard, then hold it up to the light and have it be see-through. This helps me see connections much more clearly and is hella cool no matter what other people say. This second design was much easier to make, being done in only a few hours, but most importantly, looks way prettier, which as we have established, is the most important feature of a circuit.

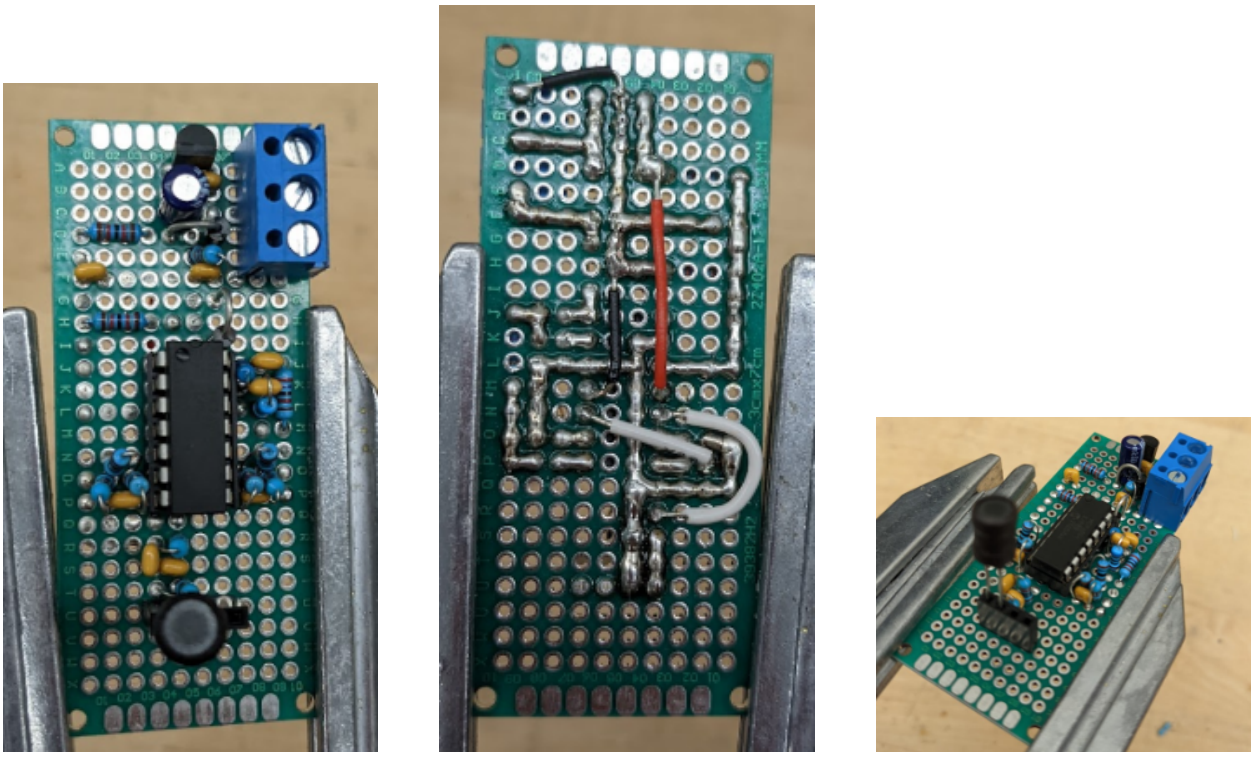

Figure 7: Finalized circuit of our track-wire detector

4 Beacon Detector

The beacon detector is also a one chip design. A big chunk of the original breadboard design was done by Scott, who I then joined on his quest to make the one chip. Mostly just helping by fixing some bugs in the circuit and then once it all worked documenting the whole thing. The Circuit for the one chip wonder looks like this:

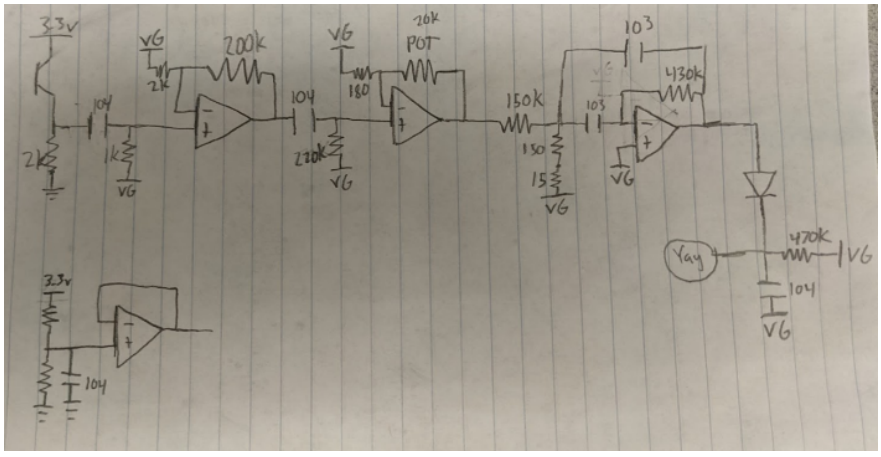

Figure 8: Wiring sketch of beacon detector

It uses 2 gain stages, one of them being a potentiometer to help adjust based on the beacon's strength, then a very strong band-pass filter with about 1.8 gain in the filter followed by a peak detector. Also the last op-amp slot was used to make a virtual ground, because the circuit was hard to get working without it. I then took the circuit drawing and converted it into a perfboard layout, heavily inspired by the layout designed earlier by me and Ben, just changed slightly to accommodate the differences from the track wire and beacon circuit.

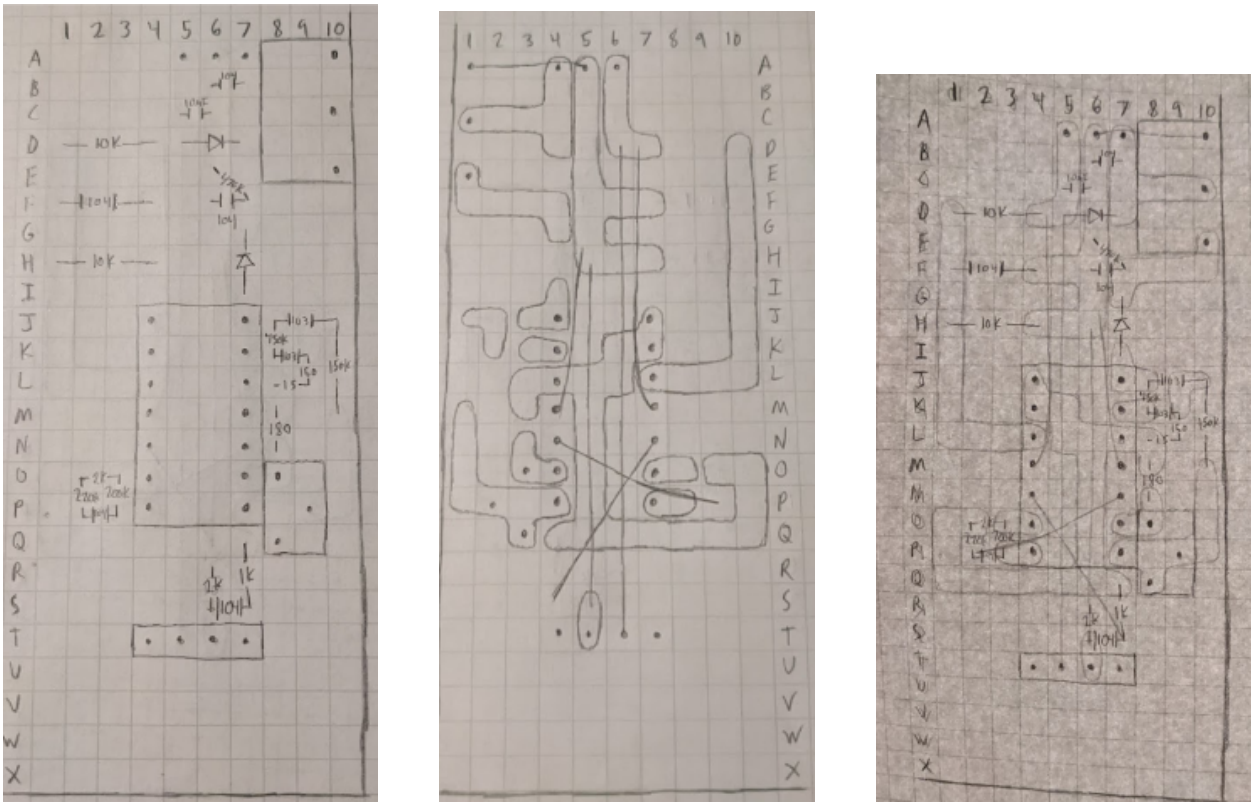

Figure 9: Finalized design of Beacon detector

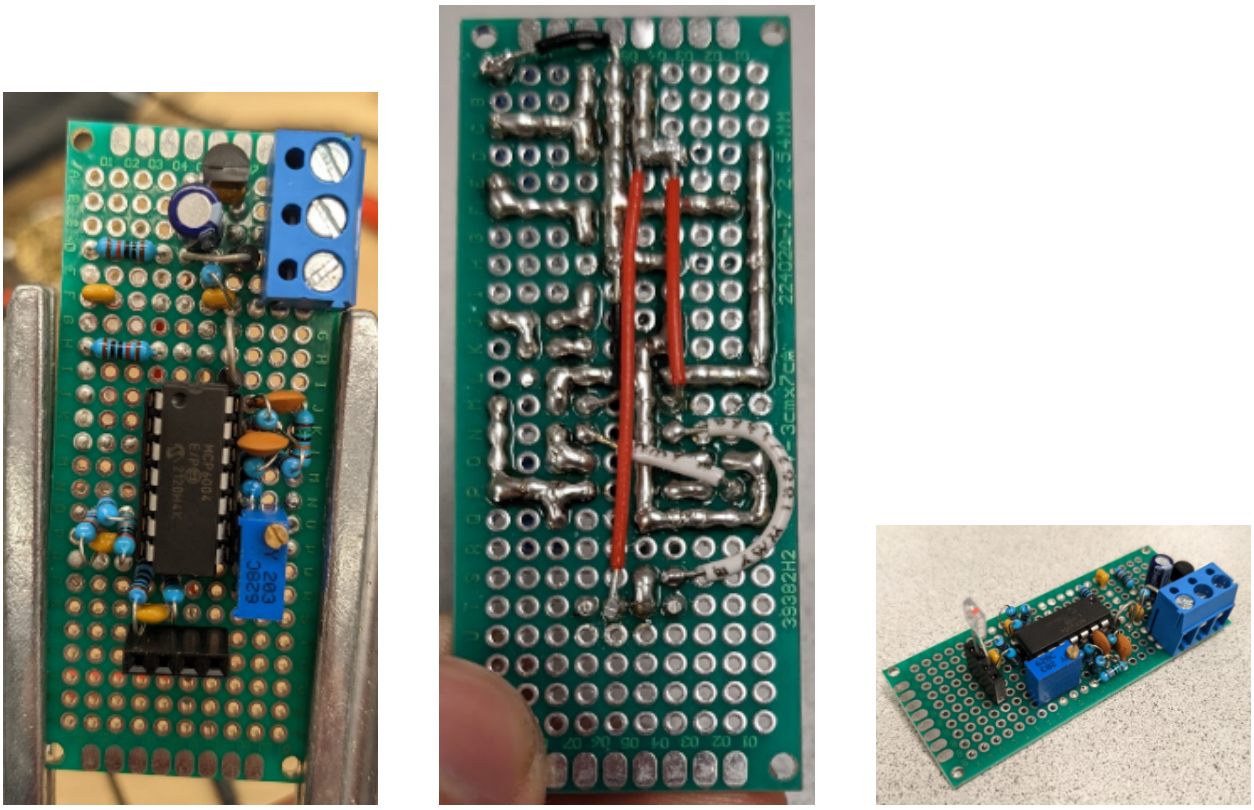

Figure 10: Finalized circuit of Beacon detector

You may notice on the back the wiring at the bottom is backwards compared to the drawing, this is because I switched the polarity because I decided I wanted the photoresistor to face the other direction, but this ended up not mattering because we connected it with jumpers anyway. Having all the circuits be really small one chip circuits was very nice because it allowed us to not really worry about where to save space for them because they basically fit anywhere on the robot. Also they are all analog outputs, which is very nice because we can easily change the bounds in code, but also later we used the analog beacon to actually tell how far away towers were. The potentiometer was very nice for this, allowing us to tune it to a point where the difference between towers was noticeable, and that allowed us to write a script that locates the closest towers to make our robot much faster.

5 Tape Sensor

The tape sensors were bought on amazon. There was a really cheap 10 pack of tape sensors on amazon that worked amazingly well under any light conditions and had a potentiometer to tune their range as well as the choice of an analog or digital output. All we did was tune the sensors and then use the digital output to post tape events. Originally we had 3 tape sensors on the bottom and two on the sides, but honestly the bottom tape sensors were really bad and caused more problems than they helped with. We had them front and back, but after we got min-spec check off, we took apart the robot and switched it to be just a middle

tape sensor on the bottom, this was much much better for handling all the edge cases that were tricky with front and back tape sensors. So really if we cleaned up the robot, the total sensor count could be 3 tape sensors, 1 beacon sensor, 2 track wire sensors, and 2 bumpers.

6 X-drive

We went with an x-drive design for our robot. This allowed us to save a lot of time coding and use fancier driving techniques to avoid needing to think of solutions to problems like flushing up against the tower and circling cleanly. The first version of our robot was actually just an x-drive with no sensors at all. Once we made the entire code library that controls the motors we never had to touch it again and just used that library when developing the state machine. The x-drive is very nice because with complete omnidirectional movement we barely had to use any timers at all for navigating, which feels great because timers are cringe (although lots of timers exist to if we get stuck or something messes up to get us out of that situation ;-;). Almost all of our turns and pivots go until a sensor reading occurs, and because of the angles the x-drive gives us we can be sure we will always make it to that sensor reading (unless another robot bumps us or our batter dies). 4 wheels is also great because we never had to deal with making skids and never had to think about the balance of our robot, or potentially losing traction on the wheels, the 4 wheels always have ground contact so we never encountered any of those problems.

7 Shooting

Our initial idea for shooting was to have a RC servo rotate a gear that would rotate two other gears which would each have a nail attached to it. The nails would move up and down due to the gears moving and they would block ping pong balls from falling off a slanted pipe on the top of the robot.

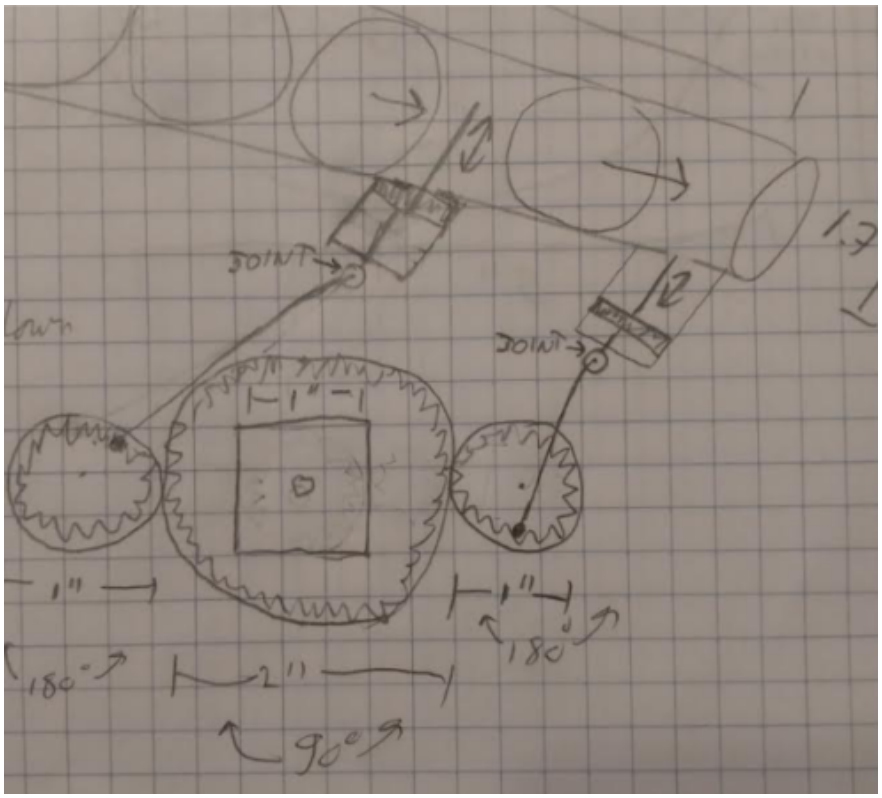

Figure 11: Initial sketch of our shooting mechanism

When sharing our robot sketch during the in-class presentation, we were suggested to only use one gear instead of three and achieve the same effect. By implementing the recommendation into our design it changed to only one gear with two nails attached on each side of the gear. Part of the design was for the nails to have a free moving joint somewhere in the middle to deal with the nails moving left and right due to being moved by a servo in a circular motion. We only wanted movement on the y-axis. So we limited the range of motion and welded a free moving joint as well. To make the joint we welded a nail to a nut, put another nail through the nut and welded a third nail to that nail. We cut the nails to the length we needed and glued one end to the RC servo adapter with the other end being pushed into the shooting pipe.

Figure 12: Initial sketch of our shooting mechanism

A problem arose of the ping pong balls jumping over the nails designed to stop them. Our initial solution was to build a roof on the pipe tight enough for the balls being unable to go through when the nails were blocking it. However the balls were getting pinched to the roof, lifting the whole pipe up and deforming the foamcore roof. Over time the roof would become loose and stop working. Therefore, we changed our design to MDF and made it impossible to pinch balls to the roof by making two thin arches offset from the nail holes.

Figure 13: Initial and final shooting cover

8 State Machine

To find the first tower, we wanted our bot to do a tank turn until it detected a beacon signal. When hitting tape it would spin in the other direction to only search for the beacon in the arena. If no beacon is found, a timeout will transition to moving forward to try to find a signal at a different location.

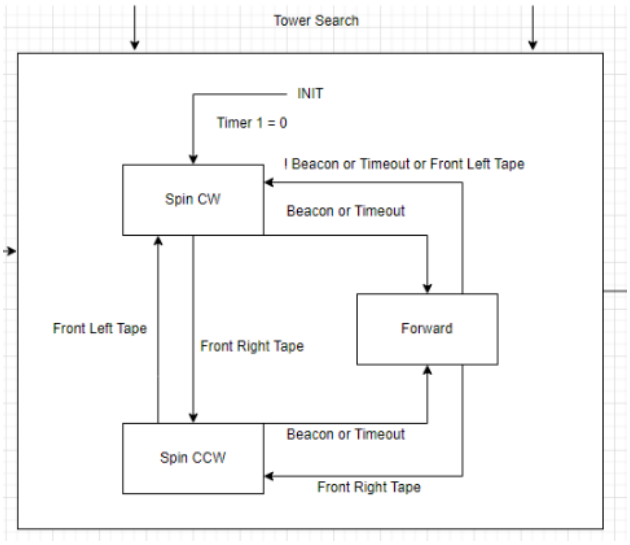

Figure 14: Proposed tower search

We ignored tape when doing the tank spin because we can't go out of bounds and the benefits were minimal. A beacon could be sensed from anywhere in the arena so a timeout was not necessary. If a signal is lost it spins CW for a bit and times out, then spins CCW. While moving forward it has a timeout to move backwards for some time in case we are stuck on a tower without being bumped. If bumped we go to the Dead Bot state

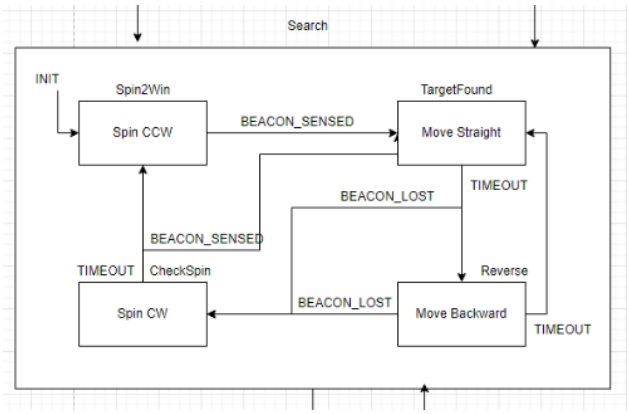

Figure 15: Actual tower search

8.1 Dead Bot

We assumed every object hit was a tower, if we did hit the dead bot, we would be stuck until we timeout after a long time.

We assumed everything hit with a bumper was a dead bot. While driving around a dead bot we would be searching for a beacon to drive to. If while we are driving around the dead bot and we read white on our tape sensor, we switch states to Tower Shark state.

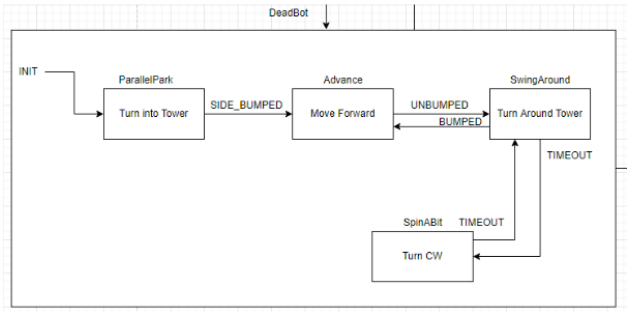

Figure 16: Actual dead bot state

8.2 Tower Shark

8.2.1 Proposed

After a bumper trigger we wanted to rotate 90 degrees to be flush with the tower using our omni wheels. Once the side bumper is hit we drive forward until it's un-bumped and we rotate around the edge of the tower. We leave the state when we find track-wire or front tape.

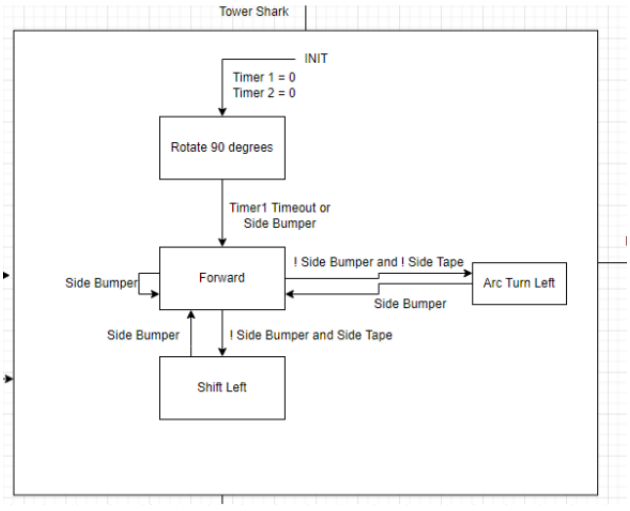

Figure 17: Proposed tower shark state

8.2.2 Actual

We combined shooting and circumnavigating because we are confident the object is a tower. Using the side bumper we drive around the tower and have two timeouts in case we get stuck. Once side tape is found, a series of states to shoot the ball and wiggle to make sure the ball falls in.

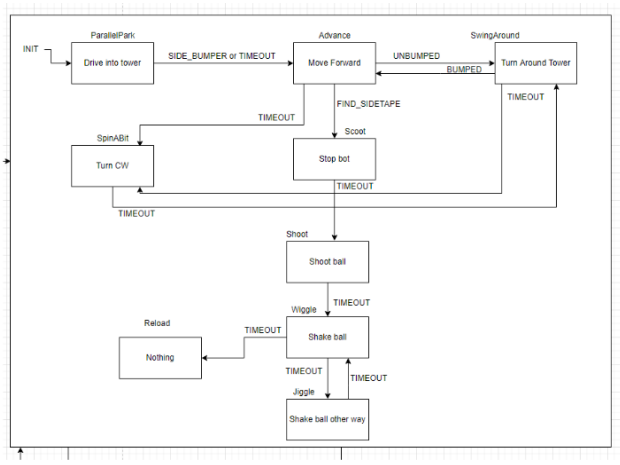

Figure 18: Actual tower shark state

8.3 Reverse Tower Shark

8.3.1 Proposed

Same as Tower Shark but the robot goes in reverse. This occurs when tape is detected while tower sharking and prevents the robot from going out of bounds.

8.3.2 Actual

Same as Tower Shark but in reverse. When tape is detected starts sharking in reverse to not go out of bounds. If tape is detected after entering reverse tower shark it is ignored to prevent getting stuck on a tower next to the cross on the field.

8.4 Second Tower

8.4.1 Proposed

After depositing a ball we wanted to rotate around the same tower until we detected a beacon of another tower.

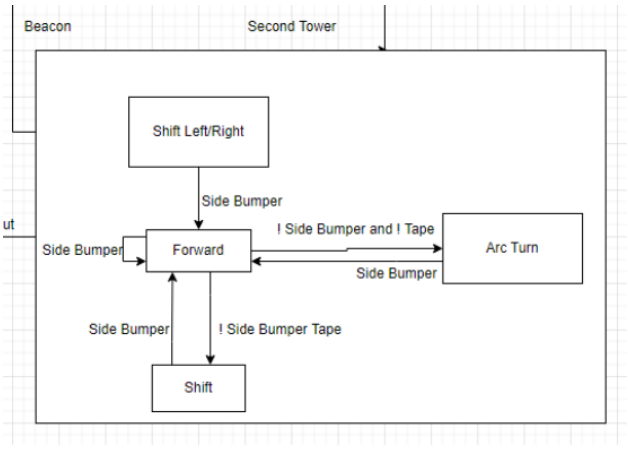

Figure 19: Proposed state logic to find second tower

8.4.2 Actual

After depositing we rotate around the tower similar to Tower Shark. If we hate tape on the ground we rotate in reverse. We rotate until a beacon signal is detected. There are multiple timeouts so we don't get stuck.

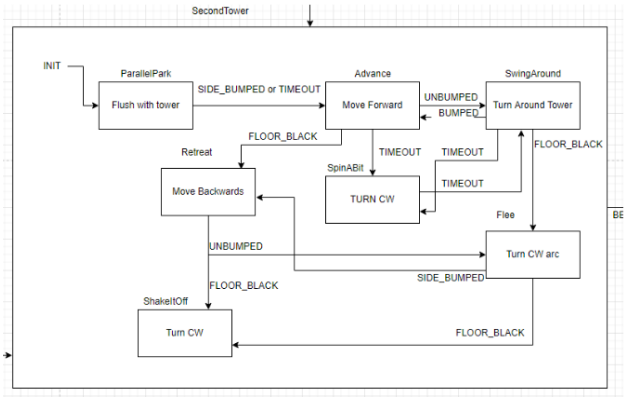

Figure 20: Actual state logic to find second tower

8.5 Main Differences

The biggest change to achieve min spec was driving around a tower in reverse to prevent going out of bounds, always assuming we hit a dead bot, and timeouts for getting stuck. The main ideas did not change, they became more complex after modifying for edge cases.

9 Solidworks

9.0.1 Proposed

When coming up with ideas for the overall structure of our bot, we were pretty set on an octagonal shape due to its not completely square shape to corner the towers easier but also

not too circular so that it can stay flush along the side of the towers better. Most of our designs involved your standard two-wheeled drive trains with a depositing ramp for the ping pong balls. We also considered a two-floor or three-floor design, with circuit and motor spacing in mind. There had to be minimal clutter in the robot so it was easier to work with.

9.0.2 Actual

For our actual design, we went with the octagonal shape of the chassis but instead of a two wheeled drive train, we decided to go with an x-drivetrain using omni wheels. Which in theory, would help us with maneuverability around towers and lining the bot parallel with the tower more efficiently. The three-floored design won out as it seemed more efficient for spacing out our wiring and circuits. From here we went about getting the parts in order for the bot and modeling them in SolidWorks.

9.1 Base Floor

For the base, there were slots that held the motor mounts, the wall supports and where all the components for movement were needed. The overall design was simplistic, with no extra cuts or slots for other components besides motors, H-bridges and tape sensors.

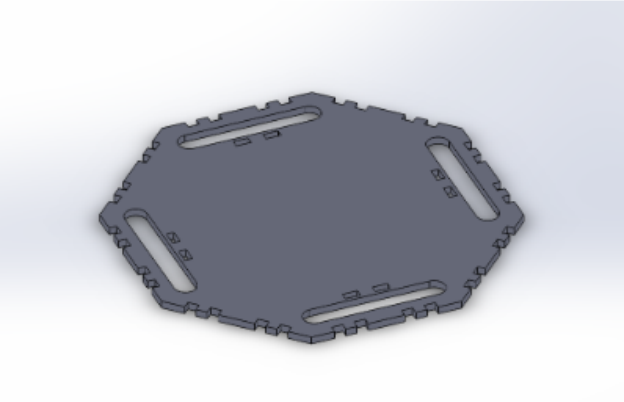

Figure 21: First level frame in Solidworks

9.2 Second Floor

For the second floor, the design was also in the same octagonal shape with slots for the wall supports and holes so that wiring can be fed from the motors and H-Bridges on the base floor to the UNO and other circuits on the second floor.

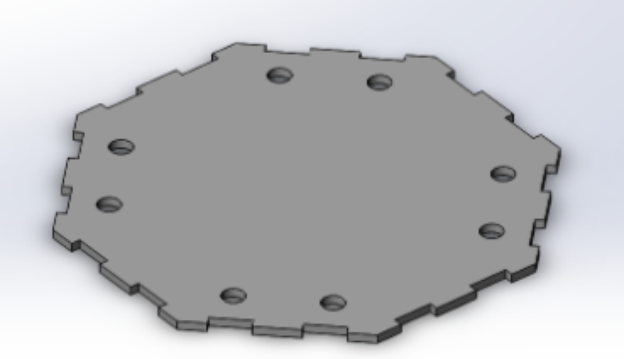

Figure 22: Second level frame in Solidworks

9.3 Top Floor

For the top floor, a rectangular gap was left so that the pipe that was to hold and dispense the ping pong balls could be placed.Slots for the wall supports are also placed here along with the shape of the floor itself that the whole body follows.

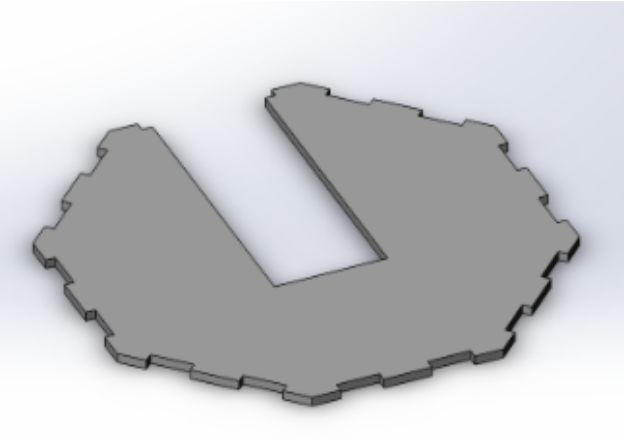

Figure 23: Top level frame in Solidworks

9.4 Lower Wall

The design for the lower wall was primarily due to lessening the weight of the bot due to it being made out of MDF and being able to access the components of the bit once it has been fully built. It features tabs so that it can slot into the floors of the bot but also a slot so that the upper walls connect to it as well.

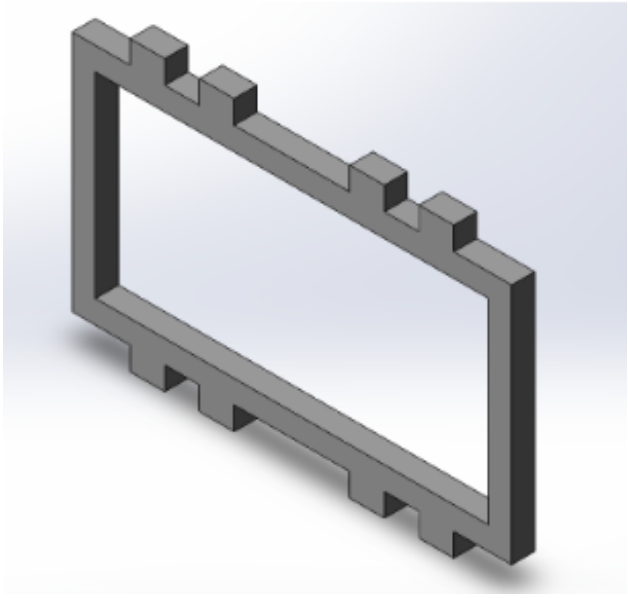

Figure 24: Wall frame connecting first and second floor

9.5 Upper Wall

The upper wall follows a similar design as the lower wall but instead being taller to account for the height of the UNO board and clearance for pipe and servo dispensing mechanism. It also features slots on the half for bumper placement and support.

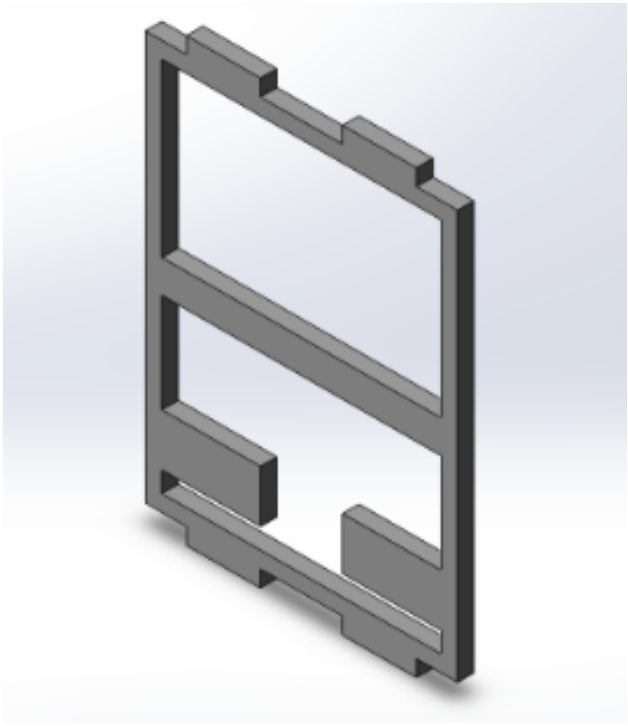

Figure 25: Wall frame connecting second and top floor

9.6 Front Bumper

The front bumper was very tricky to design, as it needed to stick out enough to provide decent object detection as well as large enough so that it could pick up any front detection around the front of the bot. The design went through many iterations before finally being the design you see here. This design consists of the same overall shape of the bot but just slightly wider so that it wraps around and also has three pegs that slot into the upper walls so it is supported. A ovular hole is placed into the center peg so that a screw could be inserted to allow it to be pressed and unpressed without falling off.

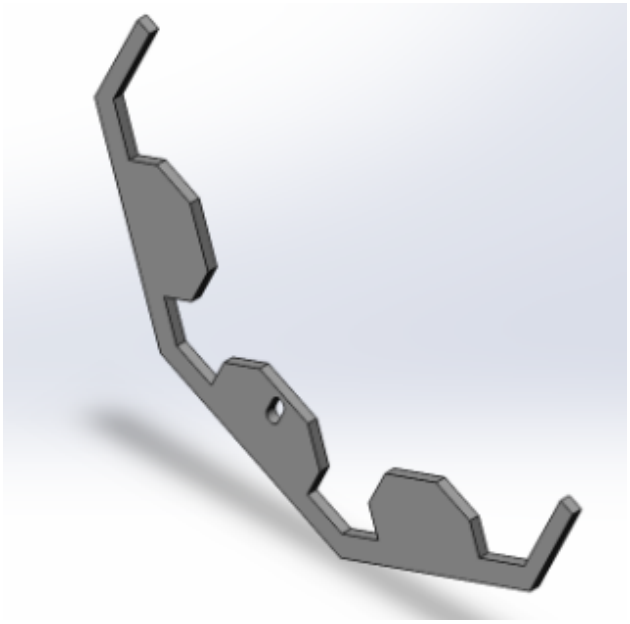

Figure 26: Bumper design to detect front left and right hits

9.7 Side Bumper

The side bumper was just as tricky as the front and went through about the same number of redesings to be what it is now. Being only a side bumper, it is a lot smaller than the front but still has the center peg with hole for a screw. Instead of the two side pegs, they are a lot small and thinner so that they can provide support. This also slots into the upper wall where it helps with side detection needed for aligning the bot with the tower.

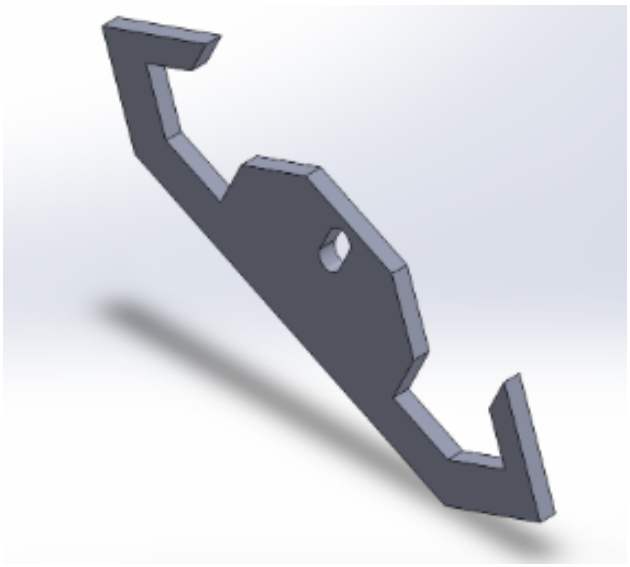

Figure 27: Bumper design to detect being flush with a tower

9.8 Motor Mounts

The motor mounts are very simplistic in design, only having two small pegs to slot into the base floor as well as holes for the motor to be screwed into. The square shape is purely a design choice, with no clear benefit to the function of the motors.

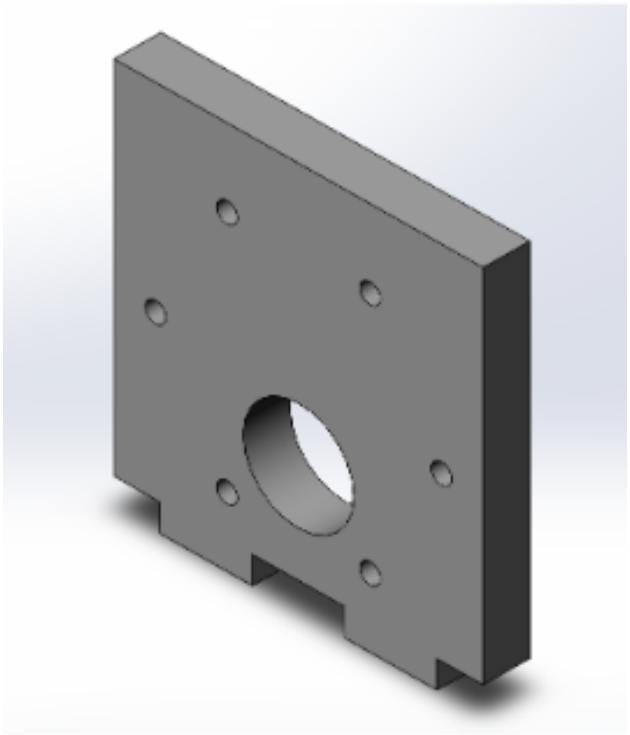

Figure 28: Motor mount

10 Conclusion

Somewhere in between the first and second week we had finished all the sensors and the initial design for the chassis so we were able to put everything in and start organizing the layout more. At this point the chassis also had a working drive so it could move, but then we needed to make this move platform detect the world around it. The sensors had already been tested to work with the code so once they were mounted it was all good, then we just needed to get shooting working. Shooting was a constant battle, but a few weeks in it was fairly consistent getting the ball in the hole every time, so beyond that was just flushing out the state machine and repairing the bot every time something broke until it was worthy and passed min spec. During min spec it was held together with lots of dreams and also some electrical tape, but immediately after passing min spec we took the entire thing apart and reorganized it to make the entire thing much more robust and easy to work with. We did some experimenting with a lot of things to try to get beer check off and ended up succeeding in that on the last day, so then we were done.