Final Project - Slug-O-Lympics: L'Escargo vs. Slug

ECE118: Intro to Mechatronics - Spring 2024

Aidan Doshier - adoshier@ucsc.edu - 1907346
Christian Hernandez - chruhern@ucsc.edu 1830035
Derrick Lai - dlai3@ucsc.edu 1865921

Last Checkoff Date: 6/6/24
Public Demo: 6/7/24
Due Date: 6/14/24
Instructor: Prof. Elkaim

Introduction:

The objective of this project is to design, construct, and program a small autonomous vehicle capable of efficiently navigating a standardized field, locating and capturing targets, and managing ball discharge. The primary goal is to minimize the number of balls left on our field by transferring them to the opponent's field through a designated slot, ultimately aiming to "clean up" our field for maximum points. This project will be executed in teams of three over a five-week period, during which we will iteratively design, implement, and test our bot to ensure reliable performance.

The field of play is a large 4'x8' white surface marked with 2" black tape and divided by a low wall featuring a slot with a one-way door for ball transfer. The boundary tape ensures that the robot remains within the play area, with more than half of the robot moving out of bounds resulting in disqualification. The field includes two towers equipped with 2kHz IR beacons, which randomly eject 25mm chrome balls, and weather stripping bumpers to contain the balls within the field. Additionally, an immovable obstacle (a "dead bot") will be placed randomly on the field, which the robot must navigate around.

The autonomous robot must be able to detect and react to collisions, resolving them within five seconds to avoid disqualification. It must also effectively identify and track the black tape boundaries to stay within the field of play. The match's success will be measured by the number of balls transferred to the opponent's field minus the number of balls remaining on our field, with points awarded for each ball in the opponent's field.

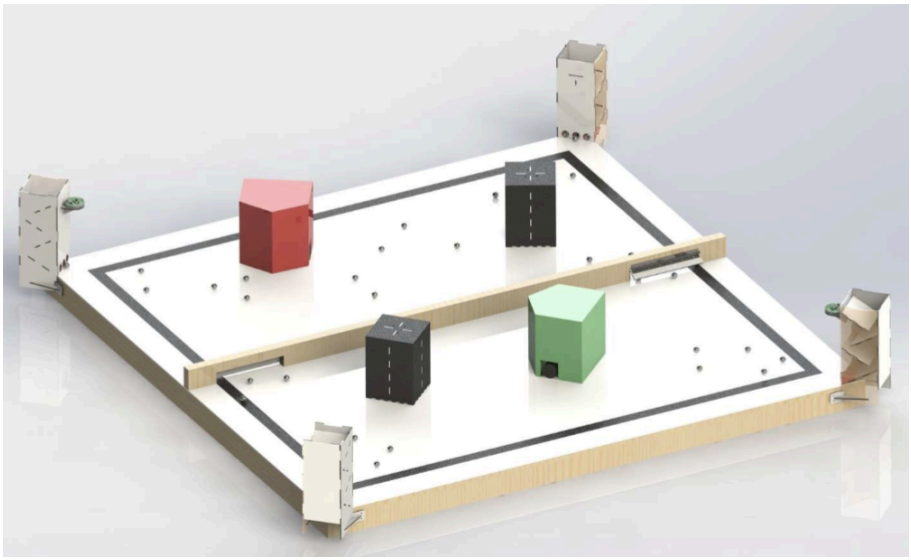


Figure: Render of the Playing Field

Minimum Specification Checkoff:

The goal of the bot was to collect 75% of balls dispensed from two towers (30/40) on the corners of the arena. It also had to dispense at least 2 balls through the trapdoor located on the middle wall. It could not go over the side of the field, or be stuck in the same spot for 5 seconds. and this all to all be done in 2 minutes.

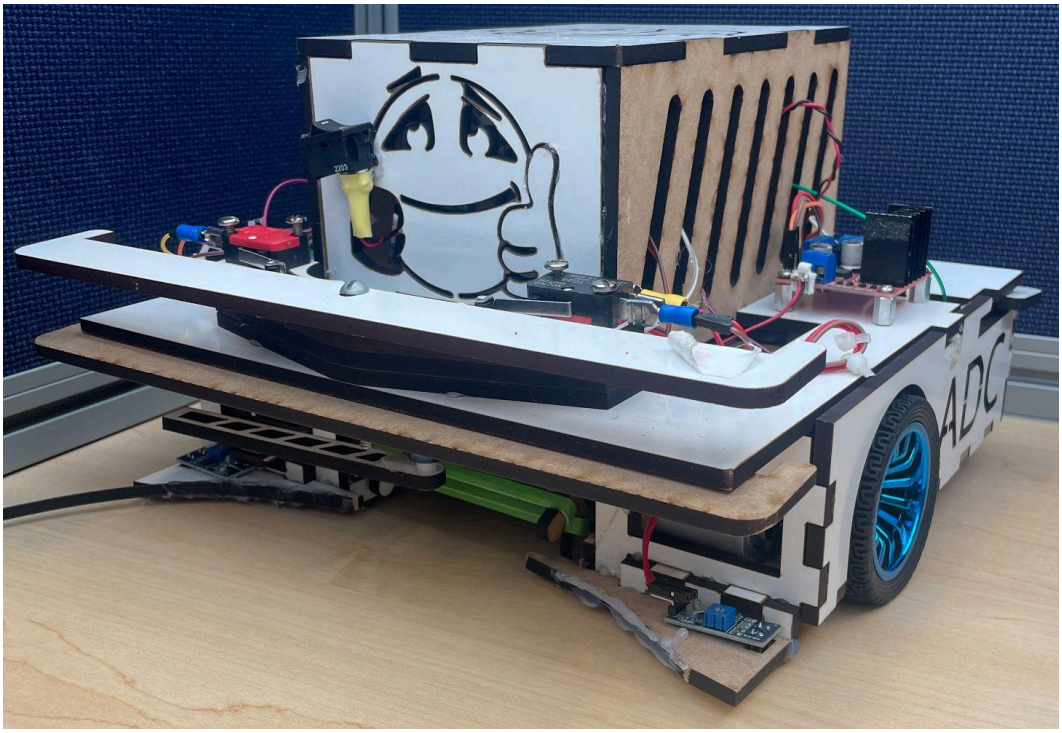
Project Main Roles:

Mechanical - Aidan Doshier
Electrical - Christian Hernandez
Software - Derrick Lai

Google Drive Link:

This Google Drive link houses all of our laser-cut iterations, parts list, and notes we took during the duration of the project.
https://drive.google.com/drive/folders/1slztBeDY4MpGfwusoZ0dTytQMhItKrVF?usp=drive_link

1. Mechanical Process:



Final CAD link:
https://cad.onshape.com/documents/735e3b587acb04230e2b2483/w/d589b06cebef545b47532829/e/fcbca52122b3b65e681ae66d?renderMode=0&uiState=664e7edb7f934073750cf737

Overview:

Our final robot consists of a square design with two wheels on the left and right, and castors in the back to keep it steady. The main frame of the bot is about 11" wide, 11" long, 3" tall. The bot has an open front with a spinner in the middle in between two front walls with the purpose of intaking the balls into the bot, up a ramp, and into the storage areas. The balls are helped into the spinner by two wedges on the front along with zip ties to help guide the balls. The spinner can be reversed so when the storage is full, the remaining balls will be on the ramp and shot out through the door. This door is opened by a servo arm on the roof towards the front of the bot. On top of the bot we have the top floor which has the bump switches for the bumpers on the bottom. It also holds the H-bridges, UNO, and battery in a cage. There are tape sensors on all four corners of the robot to detect the edges of the field.

Process:

To start the mechanical process for the final project, I first found a design of a ping pong ball collector online and modeled our very first prototype off of that. After reading through the lab manual and becoming more familiar with the minimum requirements and arena, I made some alterations to the design.

The design consisted of a square robot with two wheels on the sides towards the front of the bot. It would have an open middle front wall with a spinner to force the balls into the main storage inside of the bot. There were wedges on the front to guide the balls to the spinner and also open the door to the opponent's side of the arena. It has a second floor where the UNO stack, all of our circuits, H-bridges, and the battery would be housed. It had a ping sensor in the middle of the top floor to detect the balls and know where to go. We would also reverse the direction of the spinner to spit back out the balls to the opponent's side.
(first design and revamped design pictures)

Then I started using Onshape to CAD our design. From then on, I kept a notebook of things I did each day and tasks I needed to do. (See Mechanical_Design_Notes.pdf)
Some main problems I realized with the original design were:

- Parts could break off when colliding with something, such as the back castor wheel and door openers/funnels.
- We planned on using steppers, but DC motors are better for torque.
- The bot could be more structurally sound with the open inside.

One main update I did early on was making an overhanging floor on the top to protect the wedges to funnel the balls into the spinner, that way they didn't break off in the event of a

collision. I had to change the width and length of the bot multiple times to be able to fit the wheels, wedges, and bumpers. I chose to put the wheels inside the bot as well to save space on width. We didn't have a problem with height since our bot was compact.

We then opted for a spinner design with rubberbands stretched to get the balls inside the bot. The rubberbands gripped the balls when they came in and were even able to send them inside fast enough to go up a ramp. (Hmm.. this might be useful later)

One problem I encountered was the wedges I originally made to funnel in the balls and open the doors were only good at funneling the balls. They were just short of 2" and the door was on the other side of a 2 by 4, so they would only go into the wood but not touch the door. A solution I had for this was to mount two servos upside down on the top floor with long pieces of wood that would turn outwards when we wanted to open the door. Otherwise, they would be kept rotated under the top floor and away from everything so as not to break.

The next roadblock I had was implementing bumpers into the design. With the current height of the bot (about 3"), Putting the bumpers on the top would allow them to miss the middle wall of the arena completely. So I was able to put the front bumper under the top floor but the back bumper needed to cut into the back and side walls of the bot, which was a pain to CAD, once it was done though I figured out the dimensions of the holes for the bump switches and bumper itself and mounted them on the top floor of the bot.

I only needed to make a cage for the UNO on the top. I used the method of the Lab 3 hub with all of the motors on it. This was just a piece of MDF held up by very long screws and nuts on either side so it didn't fall.

Since my original idea of the momentum from stopping at the door would push the balls forward and into the spinner was going to be inconsistent, I made a ramp for the balls to go up inside the bot by cutting out slots in the inside walls that block the balls from the electronics. This was a pretty easy implementation and the spinner was able to launch them up. We used a ramp because if the spinner was always on, then the balls could never escape and when we wanted to reverse the motor, they would easily be shot out. I then changed the two servos to one that serves the same purpose.

At this point, we had built our first full bot and were starting to wire it together but we noticed we needed a few more holes for wires to fit through and components to fit. The main change was making the bot more structurally sound by making the two bottom floors into one since we implemented the ramp. I also tab-in slotted the side walls to the bottom floors which connected everything nicely. After making the space for the wires and screws I printed out what we believed to be our final bot.

A large problem we found with the ramp was that the spinner was not strong enough to hold all of the balls in without getting clogged. The solution I had was that we could open up the bot behind-the-wheel motors where the balls could be stored. Once those were full, the rest of the balls could be kept up with the spinner since it wouldn't be nearly as much as before. The only problem with this is that we couldn't dispense balls until our storage was full, but that wouldn't

be an issue if we could collect enough balls. I also added a second bumper on the top front to differentiate a wall collision from a bot collision.

We decided not to use the beacon detector because we didn't really see a purpose to use it. Using the bumpers, timers, and tape sensors were all the detections we needed to align ourselves correctly and complete the project.

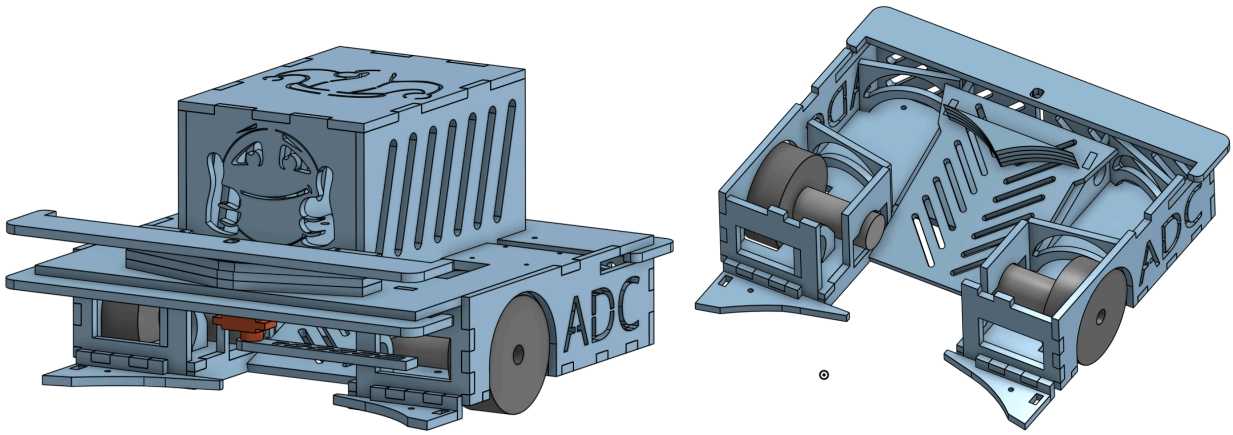
During this process we had to reassemble the bot multiple times. We built it, then tried to wire it up and found problems. Then once we solved the problems, we wired it up and found problems while testing the state machine. This continued many times but finally on June 5th, we had a working bot that only needed to be coded correctly. Overall, the most painful part of the bot were getting the bumpers to work with the design of our part.

Parts:

DC motor with wheels x2 - Used for the main driving of the bot.
Spinner DC motor - Used for spinning the spinner.
Spinner - Made with a threaded rod, nuts, lock nut, and bearing.
Castor wheels x2 - Used on the back bottom of the bot to help with driving across the ground smoothly.
Servo - Used for the door opener.
IR sensor x4 - Used for detecting the tape on the edge of the arena.

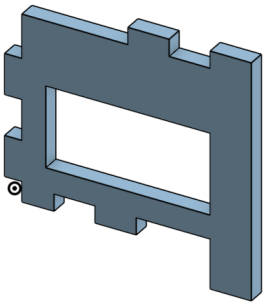
Main CAD assembly and parts with explanations:

Main bot full view and inside view:

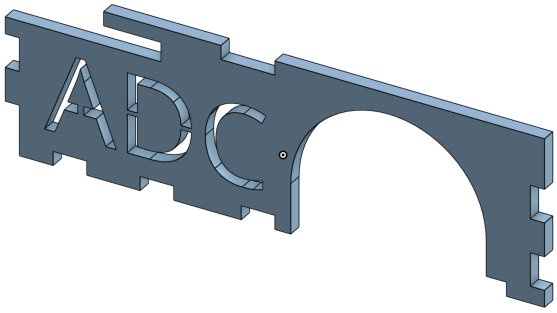


Walls:

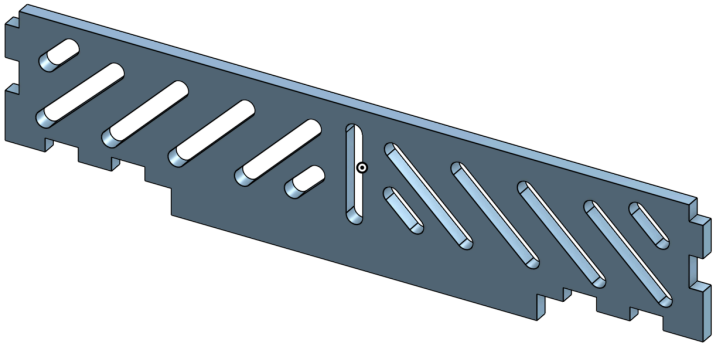
Bottom Front Wall x2 - Front wall of the bot with tab-in slots for the wedges and a square hole in the center for wires to be connected.



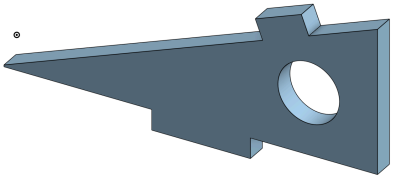
Bottom Side Wall x2 - Side wall of the bot with slots for the wheels, and tab-in slots for the top floors, front wall, and back wall. Also has a cutout for the bumper to move in.



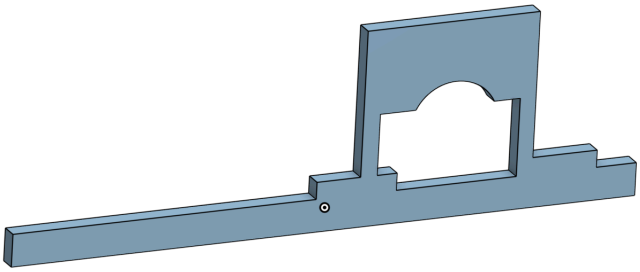
Bottom Back Wall - Back of the bot with holes for airflow and tab-in slots for the bottom floor and side walls.



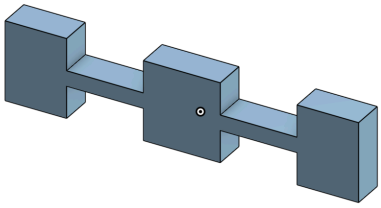
Ramp Holder - A small triangle in the back part of the bot to hold on the ramp up, slotted on.



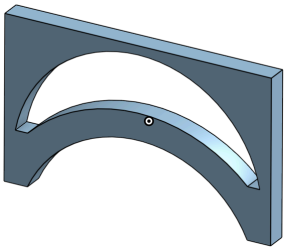
Corral Wall - Wall in the middle of the bot with a hole with the wheel motor, wires in the back, and tab-in slots to the bottom floor.



Wedge Wall - A small wall to connect the front wall to the wedges and make the wedges closer to the floor.

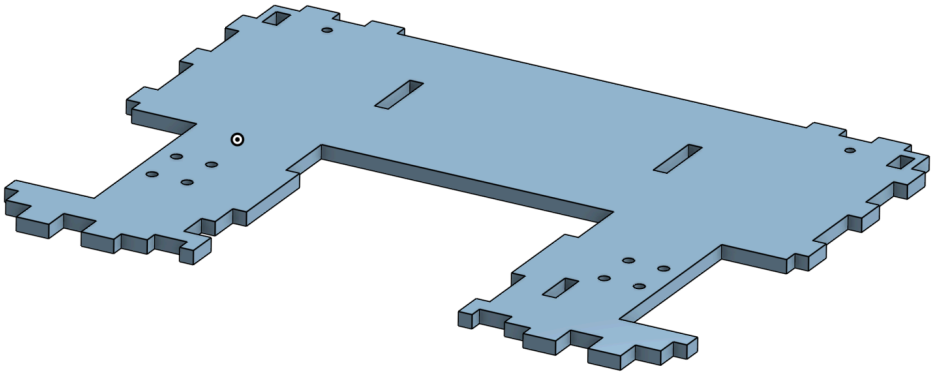


Ball Wall x4 - Wall to outline the ball storage and protect electrical and mechanical parts.

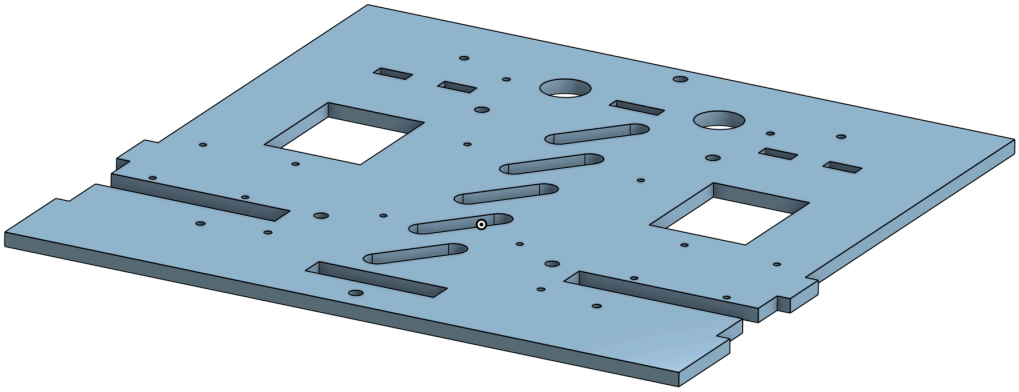


Floors:

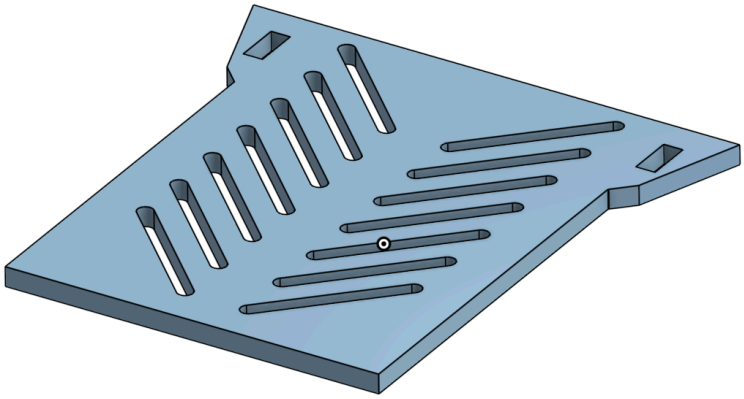
Bottom Floor - The main floor of the bot with slots for the tape sensors in the back corners, space for the wheel to fit, and a slot for the spinner motor holder for use on the left floor. Also has tab-in slots for the front walls, corral walls, and back wall. Also has holes for the H-bridges, spinner mount, spinner motor mount, and wheel motor mounts.



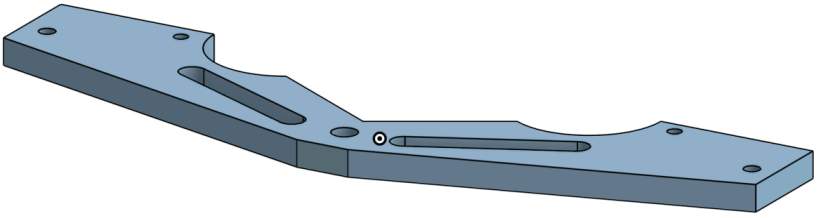
Top Floor - The main top of the bot that can be easily slotted on and off through tab in slots towards the front and back. It has spaces for wires to get out for the UNO on the sides and a hole in the front as well. It has screw holes for the bump switches, bumpers, beacon floor, and UNO. This is the part that hangs over the wedges and spinner so they can't get hit.



Ramp - A basic piece of MDF with small pieces going out to slot into the corral walls.

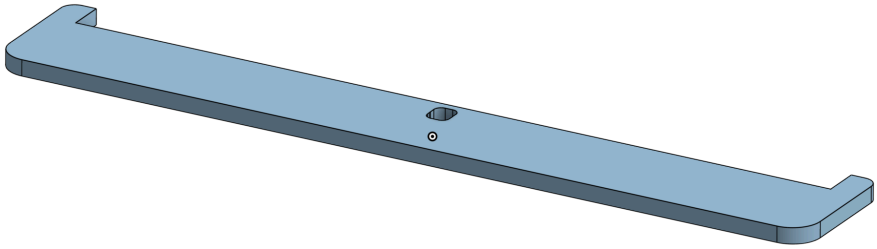


Bumper Floor x4 (stacked) - A piece to raise the second front bumper up above the wall.

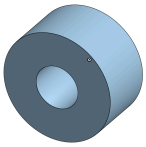


Moving Parts:

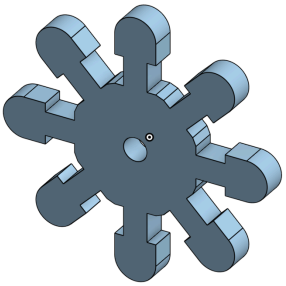
Bumper x3 - A bumper-shaped piece that has a hole to be screwed into the top floor. Can also be hit from the sides.



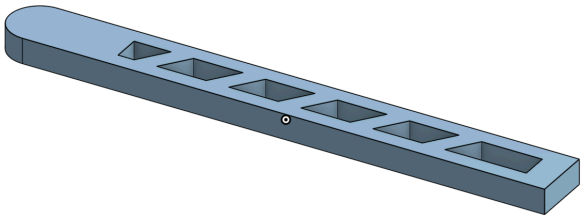
Rubberband Gear (Motor) x2 - A very small circular piece with a center hole to fit onto the motor shaft.



Rubberband Gear (Rod) x2 - An 8-pronged piece with a larger center hole to fit onto a threaded rod instead of the motor shaft. Also has small cutouts on the arms for rubber bands.

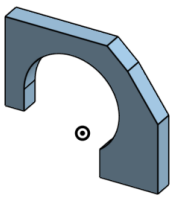


Opener - Long piece with holes cut in for weight that does the opening of the door.

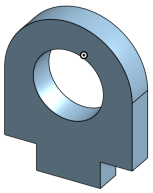


Miscellaneous:

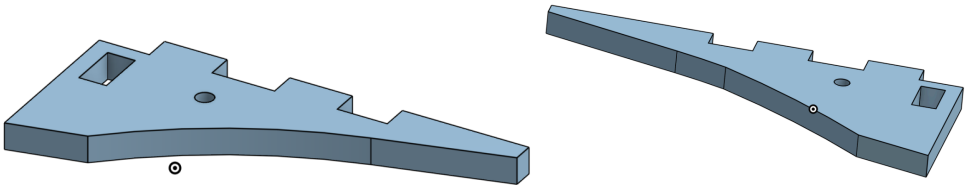
Motor Holder - A small piece that holds the propeller motor in place on the bottom floor. It also fits on the front wall for more stability.



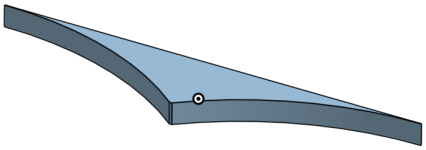
Spinner Holder - A small piece that has a bearing inside which allows the threaded rod to fit spin with the gears on it.



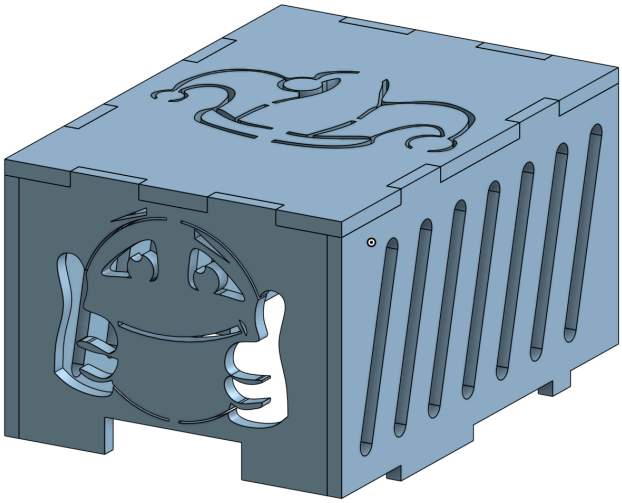
Wedges - Piece that has a small hole for a tape sensor. The left one is wider than the right as it has to cover a few gears glued together. Slotted into the wedge walls.



Guider x4 (stacked)- A pinball-machine-esque piece that allowed the balls to be guided into the two sides of the bot for storage.



UNO Cage - Box consisting of a roof and 4 walls to house the UNO, battery, and keep the wires inside nicely.



2. Electrical Process:

The beginning of the electrical design for ADC began with understanding how we wanted our bot to behave during the two minute checkoff round; as a result, we would then know what sensors, motors, and actuators would best fit those needs. Initially, our team wanted all the bells and whistles on the bot: beacon detector, track wire sensor, and a ping sensor for dictating the

bots behavior; however, after the first week, we took a step back and realized that we were not capable of perfecting each of those sensors in hardware or software.

For example, we realized that the beacon detector trade-off complicated the bots behavior. Our initial beacon detector design optimized the range at which we were able to detect the beacon; however, during testing we found that the opponent beacon was also picked up as well. This could have been handled with additional states in the software state machine, but we chose to opt out of using the sensor for more simplicity in software. A version of our design is seen below.

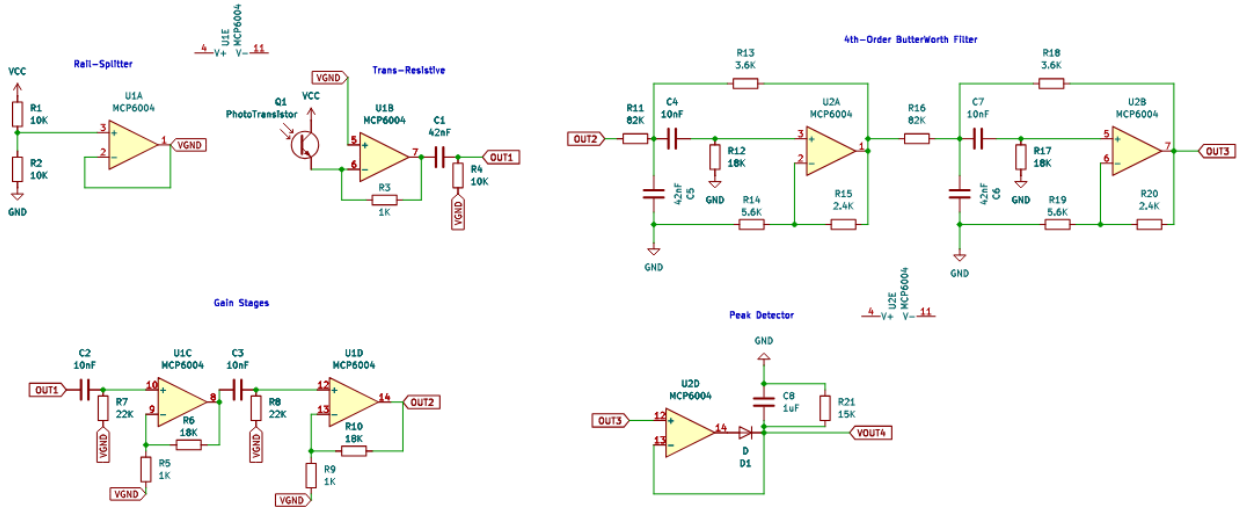


Figure 22: Beacon Detector Design

At the end of the second week, we finalized our electrical design to fit how we were planning to program the bots behavior. A top level view of ADC's electrical system is seen in the figures below.

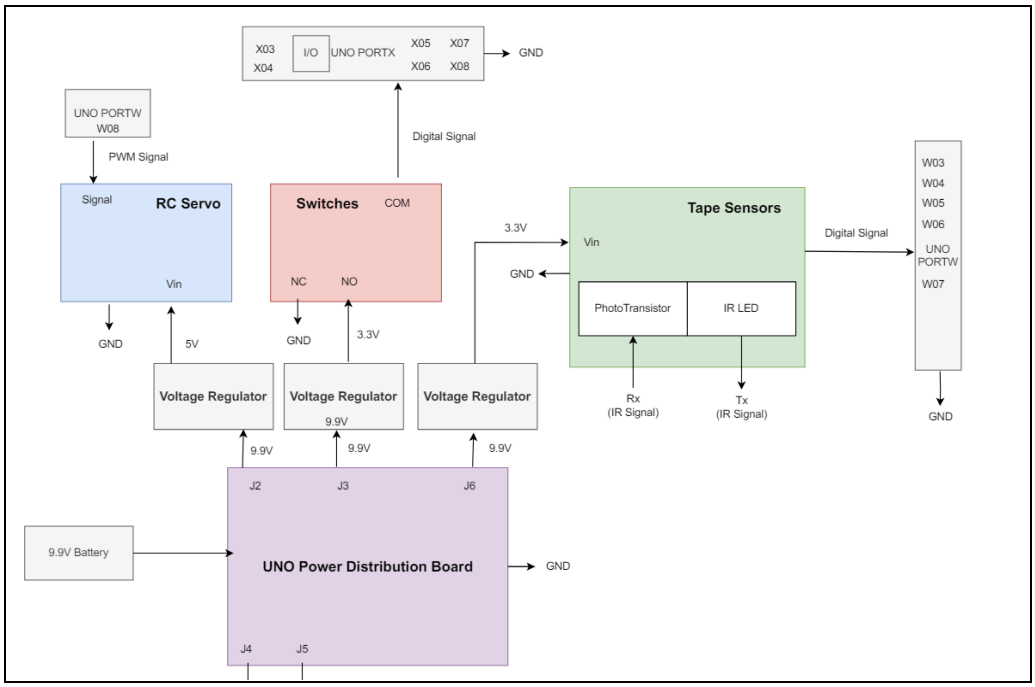


Figure 23: Top Level View of Electrical System

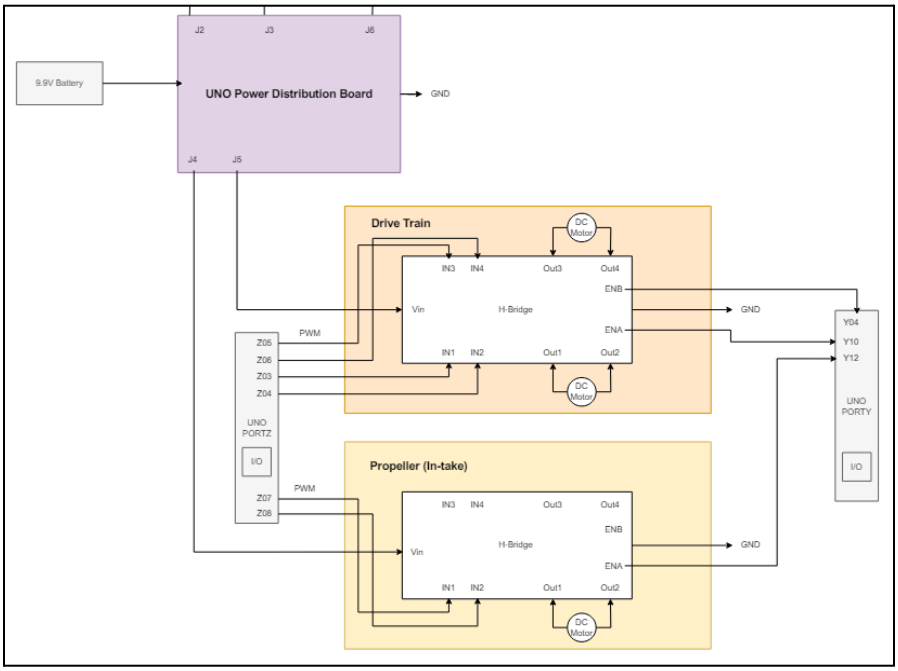


Figure 24: Top Level View of Electrical System

To begin integrating the entire electrical system illustrated in the figures above, the electrical team first took into account the steps needed to get the bot to minspec: the first step was to get the drivetrain working, the second step was to get the bot to make smart decisions, and the third step was to collect and dispense balls.

Drivetrain

Figure 25: Gear Motor, Wheel, and Mount Kit

The mechanical team took charge of selecting the proper drive motors and wheels for our bot. We settled on a kit from amazon that included the parts seen in figure 25. The drive motors came packaged with encoders, but due to the complicated nature of setting up the encoder driver, we decided to not use them.

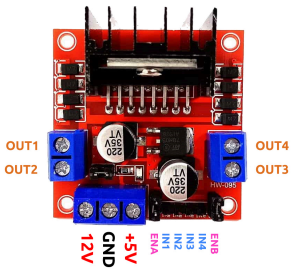


Figure 26: L298N Motor Driver

The data sheet has the motors rated for a maximum of 12V. We wanted the bot to move forwards and have it reverse as well, so we chose to drive the motors using the L298N Motor Driver. The driver is powered using the Uno Power Distribution Board which outputs the 9.9V coming from the battery. Assuming the driver has a voltage drop of 2V, the motors are then being driven with around 8V which never exceeds the maximum voltage rating.

These drive motors and the L298N driver were enough to get our bot to perform basic movements. The drive train block in figure 24 illustrates how we used one H-Bridge to control our two drive motors. Figure 26 provides a close up look of the module and the exact ports that needed to be connected. The EN pins allowed us to control the speed of the drive motors using a PWM signal, and the IN pins let us control the directionality of the motors.

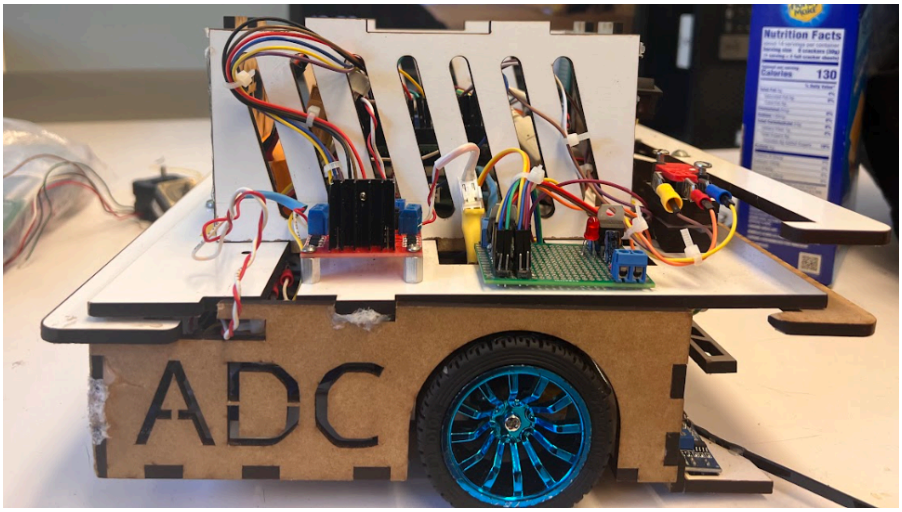


Figure 27: Side view of H-Bridge used for Drive Motors

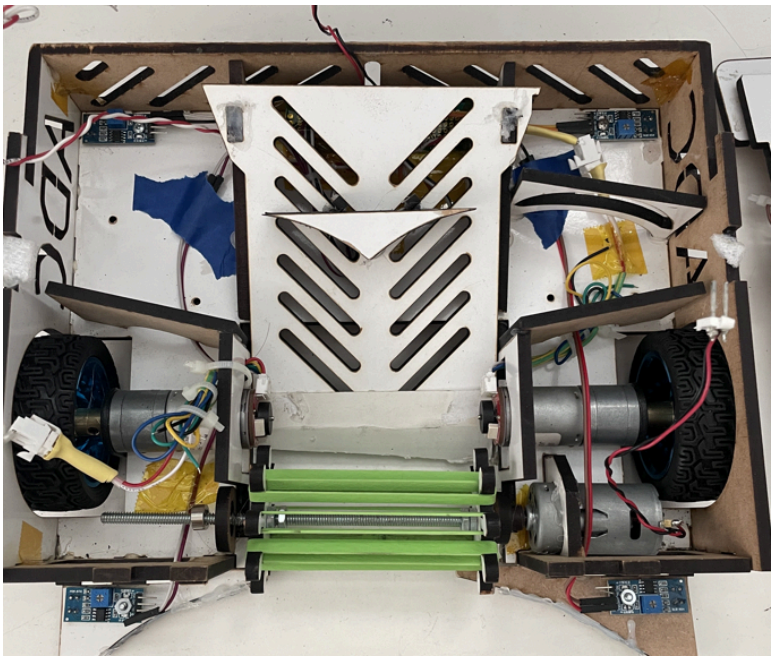


Figure 28: Top view of Drive Motor Placement

The side view of the bot in figure 27 shows how we placed and wired the H-Bridge. It sits on the top floor of the bot to keep the bottom floor empty; additionally, it made wiring the Uno32 much neater because there were no wires coming from the bottom floor. The only exception is made for the drive motor wires which can be seen in figure 28. To keep the bottom floor of the bot neat, we routed the left drive motor wiring under the ramp which made slotting the wiring up to the H-Bridge an easy task. The right drive motor sat directly beneath the module, so we simply routed its wires to the top floor. With the drivetrain powered and wired up, the software team was able to get the bot to perform basic movements.

Sensors

Our next goal was to get the bot to execute smart decisions on the field which meant setting up the sensors. As stated previously, our team made the decision to leave out the beacon detector, track wire, and ping sensor for the sake of simplicity, and because we had a plan for implementing software that would robustly perform each of their functions. The only sensors needed for our bot were bump switches and tape sensors.

Figure 29: Bump Switches and Tape Sensors

The bump switches were simple to configure. Figure 23 illustrates how they were powered and wired to the Uno32. The bumper switch sends 3.3V to the COM pin on the switch which then gets sent into the Uno32 as a digital signal. To power the switch with 3.3V, a voltage regulator was needed to drop the 9.9V coming out of the power distribution board to 3.3V. The schematic for the distribution board is provided in figure 30. Two capacitors are placed at the input and output terminals to reduce noise in the circuit, and a resistor-led circuit is placed at the output to indicate the regulator is switched on. The soldered perfboard is seen in figure 30.

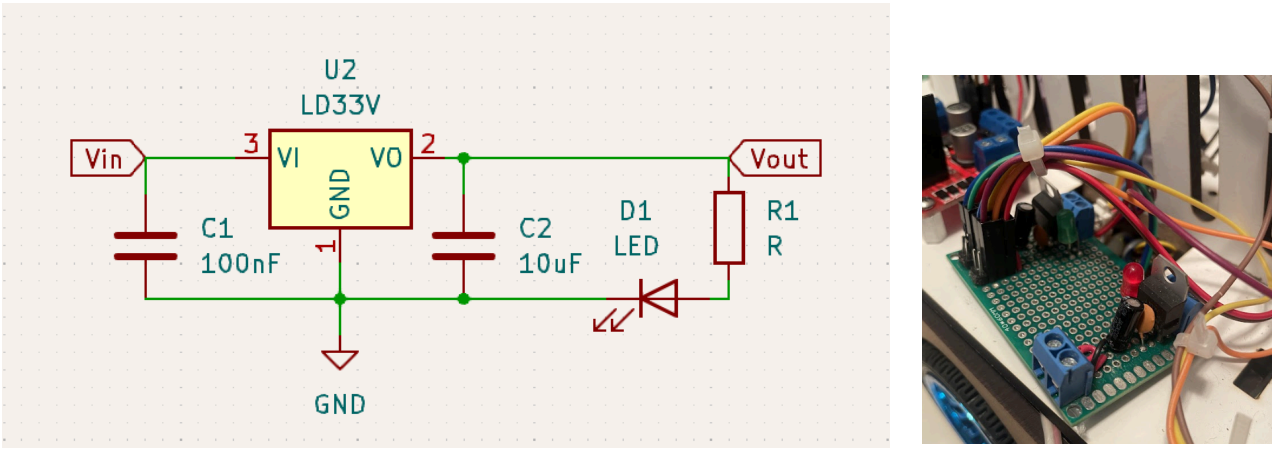


Figure 30: 3.3V Power distribution Board Schematic and Perfboard

With the bumper switches powered and connected to the Uno32, we moved on to wiring the tape sensors. Similar to the bump switches, configuring the tape sensors was a simple process since we used ones bought off amazon which meant there was nothing that needed to be soldered or designed. The tape sensor pictured in figure 29 is the model we used, and it came packaged with a potentiometer and four pins to control the sensor. We ran a couple of experiments to determine whether we should use the analog or digital output of the sensor, and we came to the conclusion

that the digital output would work best considering we were short on analog pins. The hardware hysteresis using the potentiometer also worked really well under most light conditions. To power the tape sensors with 3.3V, we used the same schematic in figure 30, but we did solder an additional voltage regulator to keep all the tape sensor wiring on the bottom floor of the bot.

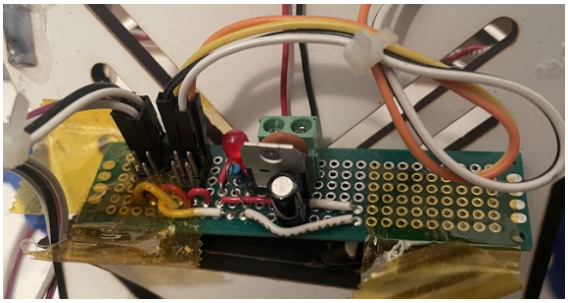


Figure 31: Power Distribution Board for Tape Sensors.

The perfboard in figure 31 is the 3.3V distribution board used to power the sensors, and the view in figure 28, reveals how the four tape sensors were wired on the bot. Two tape sensors were used at the back and two were placed at the front of the bot. With all the sensors powered and connected to the Uno32, the bot was beginning to detect wall hits, obstacle hits, and tape events while maneuvering the field.

Intake

The final step to setting up the electrical system was to power and connect the intake and outtake mechanism. For the intake mechanism, our team agreed on using a DC motor to control the spinner described in the mechanical section above. We purchased a 1200 RPM motor from Amazon to ensure the spinner had enough power to collect and dispense the balls on the field. The motor is rated for a maximum of 12V. To ensure we never exceeded the limit, we simply powered it using the 9.9V coming out of the Uno Power Distribution Board.

Figure 32: 1200 RPM DC Motor used for Intake and Outtake

Similar to how we controlled the drive motors, we also used an H-Bridge to control the spinner motor. The H-Bridge allowed us to control the speed and direction of the spinner motor which was essential to our intake and outtake mechanisms. During collection we wanted the spinner to rotate forwards and during dispensal, we wanted the spinner to reverse its direction.

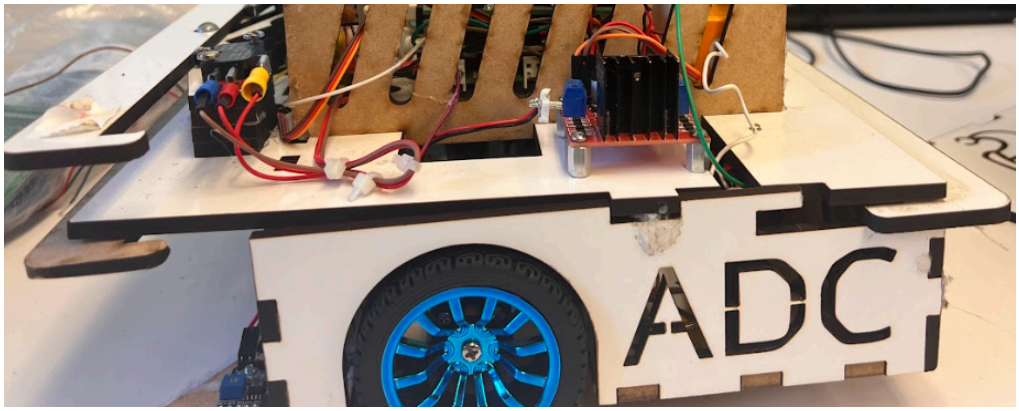
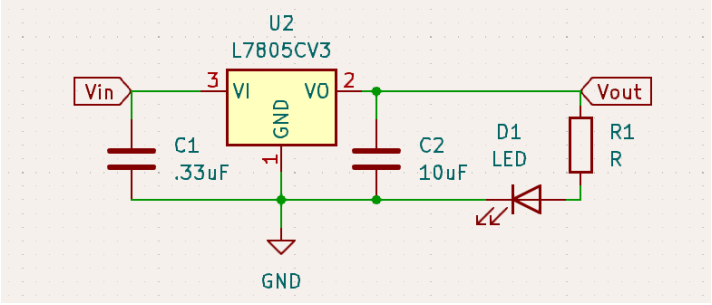


Figure 33: Side View of H-Bridge used for Spinner Motor

The exact pins and ports used to connect the H-Bridge to the Uno32 are described in figure 24. With the spinner powered and wired to the Uno32, we were able to collect and dispense balls using PWM signals that were sent to the H-Bridge. The final task was getting our bot to open the trap door during its dispensing state.

Figure 34: Micro Servo

We agreed on using a 9g micro-servo to open the trap door due to its light-weight and simplicity. The servo requires more than 3.3V, so we had to design a 5V regulator that would reliably supply the servo with power. The design of the regulator is identical to the 3.3V regulator apart from the IC and capacitors used. The L7805CV3 regulator was used and a .33uF and 10uF capacitors were used at the input and output ports. An additional resistor-led circuit was used as well to indicate the regulator was switched on. With the servo powered, all that was left was connecting the servo control wire to the Uno32 which is detailed in figure 23.



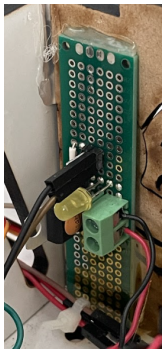


Figure 35: 5V Regulator Schematic and Perfboard

With the intake and outtake mechanism wired, the bot was fitted with enough hardware to complete the minspec tasks. The figure below provides a look into the brain of the bot. All the wiring is housed in a box to protect the Uno32, wires, and battery from coming into contact with any external objects during the minspec tasks. We kept the wiring as neat as possible, and we believe that contributed a lot to our success during the tournament and checkoff because we were confident none of the wiring could come apart or possibly create a short circuit. The electrical system was an extremely fun task to complete.

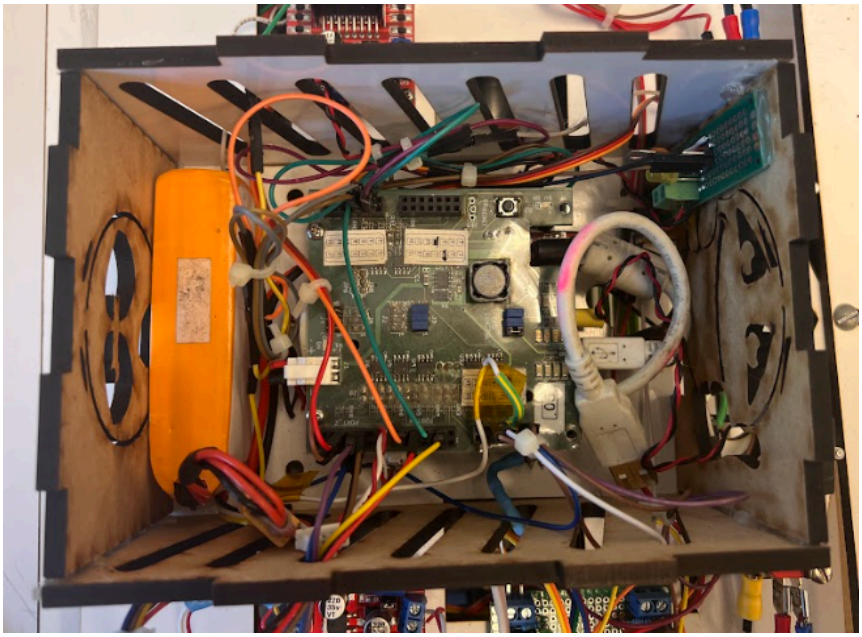


Figure 36: Top View of Uno32 Wiring

3. Software Process:

After completion of the mechanical and electrical components, the next and last essential step to a functional autonomous robot to implement a drive base for the robot, then subsequently followed by the creation of a state machine to allow the robot to traverse the field intelligently. The codebase will be written based upon the provided ES_Framework, a framework built to allow for ‘events’, actions raised to be handled accordingly based on the needed behavior, to be seamlessly polled, fired, and listened upon.

Drivebase:

The first essential part of the software process is to develop the necessary modules to allow the robot to move by controlling the motors and for its sensors to be read. This is done through a module named ‘Robot.h/c’.

To allow for bidirectional movement, a PWM signal is sent through the enable pin of an L298 H-Bridge to determine the speed of the motors, with digital bits sent through the INA and INB pins to determine direction.

The sensors used have outputs that can be divided into two categories, a digital output, or an analog-digital output. A digital output is either a high or low, defined as a 1 or 0 respectively. The analog output is a digital value based on the ADC reading, which converts an analog value into a 10-bit digital value. Multiple getter functions are written that return either the digital or ADC value from the port pin connected to the sensor.

Event Checking:

The drive base only sets the motors and reads the sensor values. For the sensor readings, this is not sufficient enough to determine how and when the robot should react based on the reading. As such, a method must be developed to detect changes in the sensor reading to ensure the robot can react to sufficient change.

This method can be implemented using the ES_Framework. The framework allows the sensor readings to be periodically polled and compared with previous readings. The sensor readings are compared with a threshold. If it is above or below a low or high threshold reading and is different from the previously recorded event posting, this will indicate a sufficient change and an event will be posted.

In the current configuration, the sensors are polled every 20 milliseconds using SimpleService, a service provided by the ES_Framework. This polling of events will allow for events to be posted. The posted event will be used within the state machine to determine how the robot will behave.

State Machine Design:

After the drive base and the event polling and posting have been implemented, the setup is finished to allow for the design of a state machine. The overall goal of the state machine is to robustly traverse around the field to collect a minimum of 30 chrome balls. The field itself also contains an obstacle that the robot must move around. In addition, there is also a trapdoor present that the robot must navigate to dispense at least two balls through it.

Before the state machine was finalized, there were multiple design ideas. Initially, a very complicated state machine was designed where it was meant to cover every area possible from the field. However, this idea presented an issue. This initial design assumed that the motors would run at the same speed at all times to allow for the robot to move straight. However, it was quickly discovered that this was not the case. The motors themselves have multiple factors affecting their speeds. One of the biggest factors was the current from the battery. As time goes on, the voltage in the battery drains, which slowly decreases the provided current from the

battery. As current is used to drive the motors, this results in slower varying speeds. Because of the inability of the robot to move straight consistently, it would make snaking around the field very difficult.

A test was performed by dropping some chrome balls in the field to see where it ended up. From this, it was observed that most of the chrome balls accumulated mostly near the wall and some were in the tape, but very few if any balls ended up in the center of the field. From this observation, a more simplistic design was used.

HSM:

In the final design involves two hierachicical states. One to find the wall, and the other to continuously follow the wall to collect and dispense the balls.

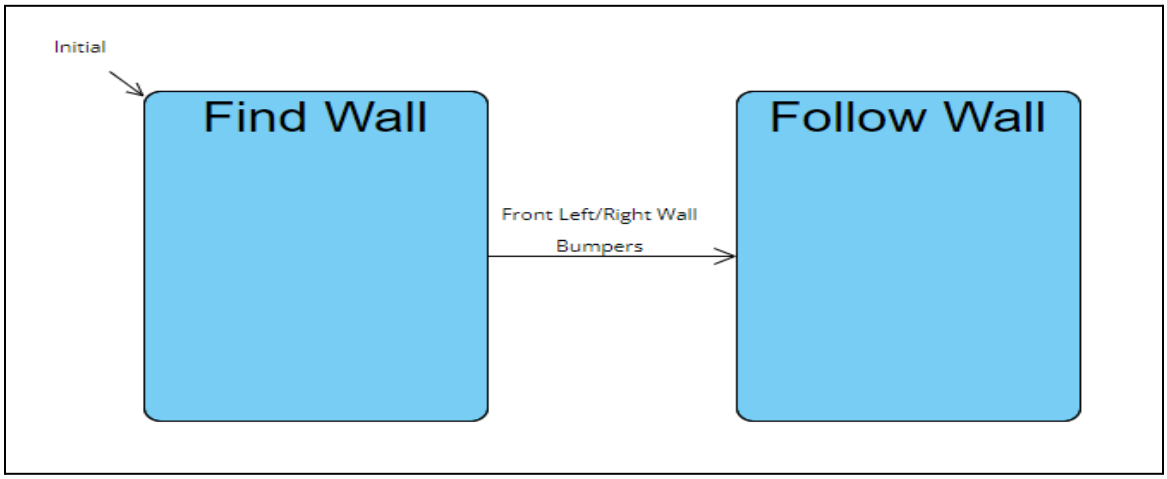


Figure 37: Top Level HSM

Sub-HSM - Wall Search:

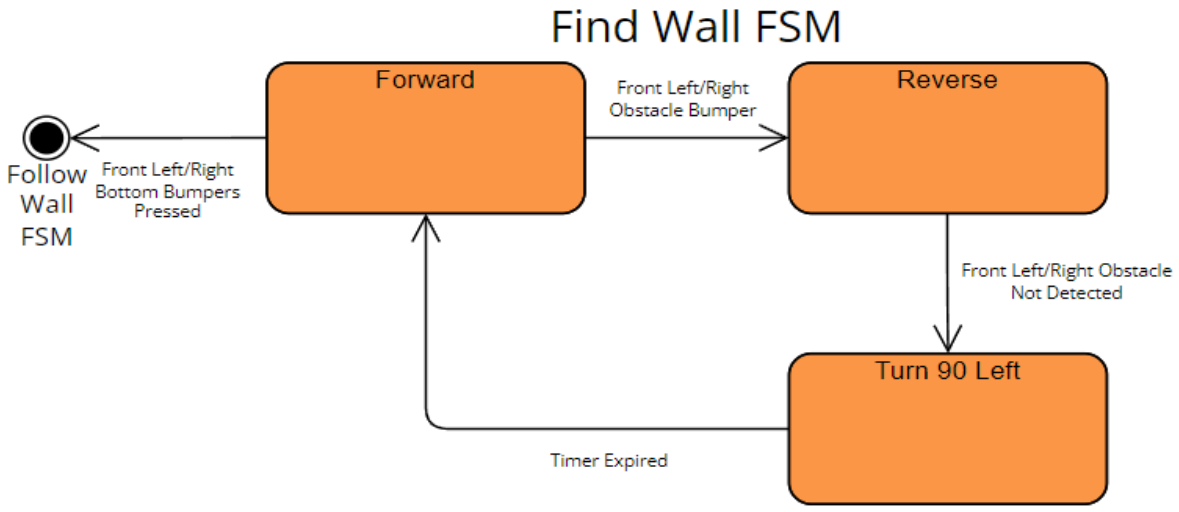
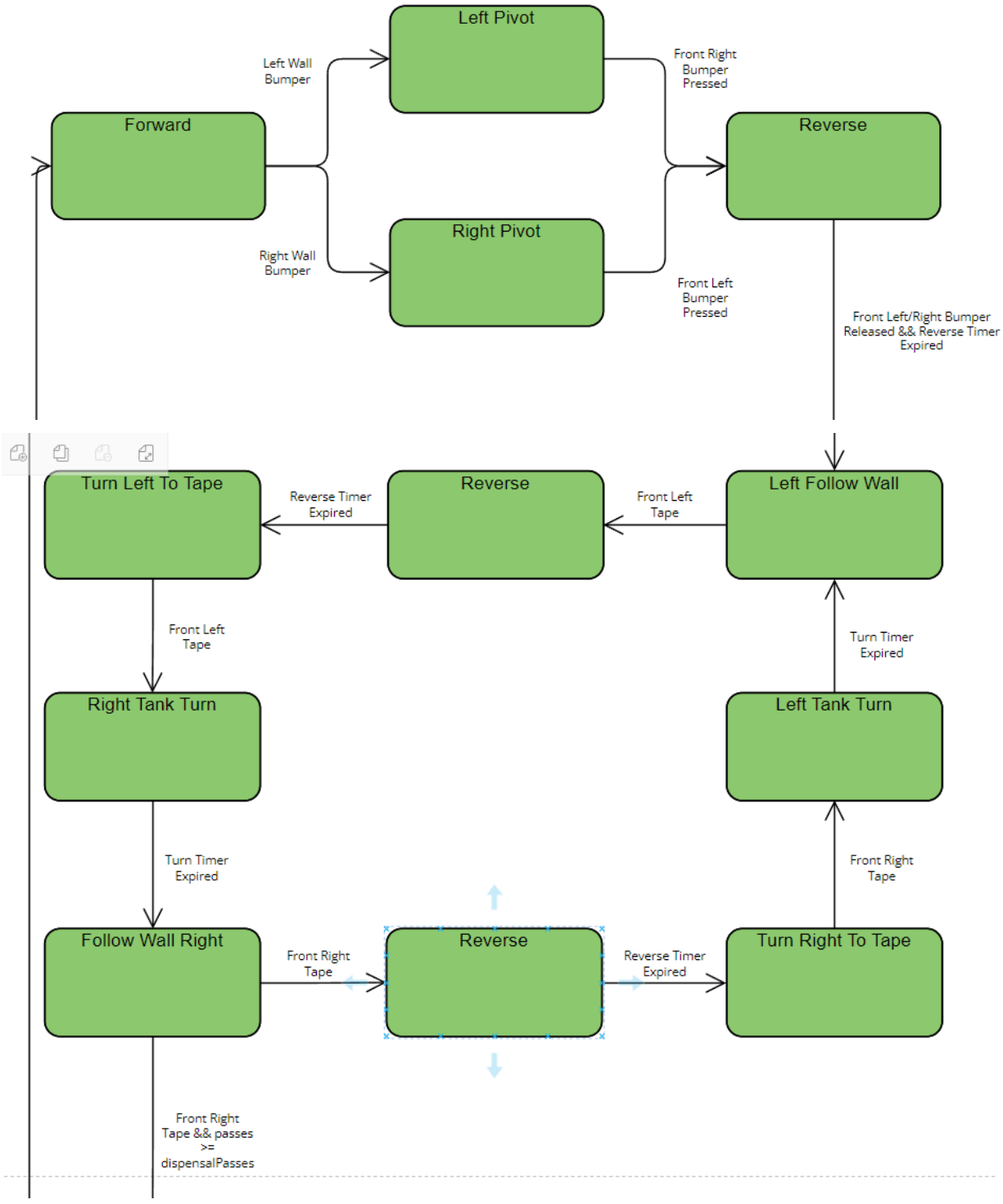


Figure 38: FSM of the FindWall substrate

The wall-finding state machine is fairly simple. Its main goal is to keep moving and turning until it finds a wall, which is indicated by the front bottom bumpers being pressed and the absence of the front top bumper events. The overall actions to find the wall are moving forward. If the top front bumpers (obstacle bumpers) are detected, then the robot will reverse until the bumpers are released, then perform a left tank turn of 90 degrees, then it will keep repeating this process until the wall is found. The 90-degree rotation is controlled by a timer. The accuracy of 90 degrees is not too important, thus using a timer will not cause too many issues. When a wall is found, it will transition to the wall following the state.

Sub-HSM - Wall Follow:

Follow Wall FSM



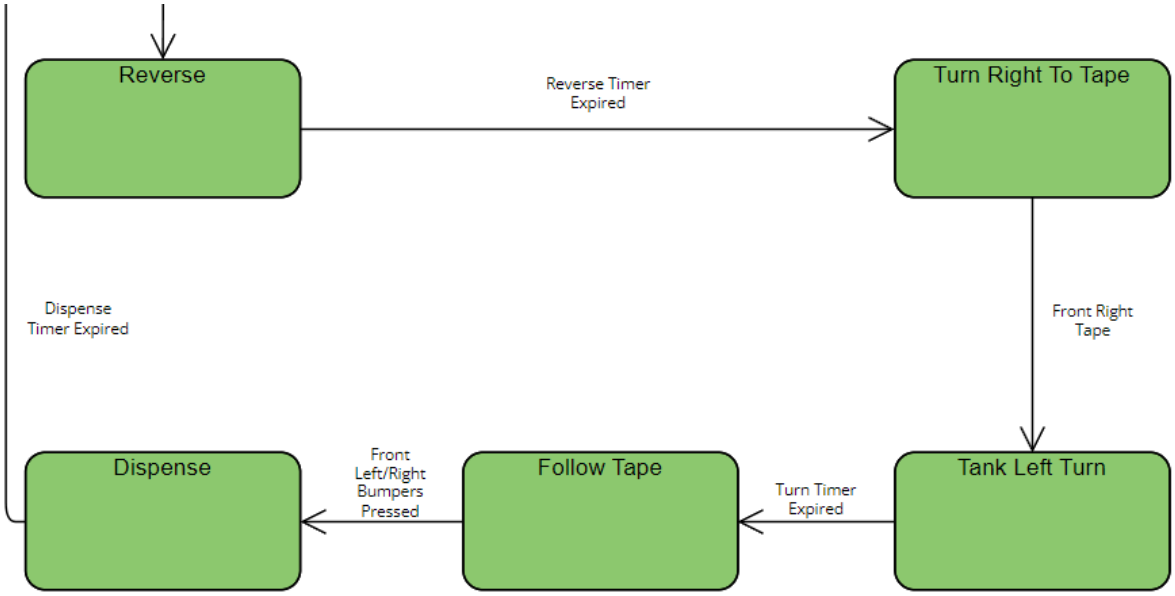


Figure 38: FSM of the FollowWall substrate

The wall-following state machine is slightly more complicated than the wall-finding state machine. However, the process is fairly straightforward. The main idea is that it will first follow the wall to the left, then perform a three-point turn before following the wall to the right. After the robot has performed a full traversal (both left and right walls follow) enough times to reach a preset limit, it will begin dispensing the balls.

Referring to the picture of the field, the rightmost side contains the trapdoor, hence after finishing a full right side traversal, the trapdoor can be reached by performing a turn and following the tape.

From the state machine, the states responsible for performing a turn to traverse the opposite direction may seem convoluted, as it seems like performing a tank turn of 180 will allow the robot to face the other way. However, this does not work because since the robot is following the wall, the wall itself will present itself as an obstacle. Because of the stated reasons, to perform a turn, the robot must back out, then veer to a direction opposite of the wall, before performing a tank turn to continue following the wall in the opposite direction.

The final design of the state machine came from multiple iterations based off of previous designs. Using what was learned from the mistakes from previous iterations, such as when to use sensors or timers to control turns, the final design allows for the wall following and dispensal to be smooth and consistent.

Conclusion:

Overall, there were a lot of essential skills that we learned upon min spec checkoff. On June 6th, we got our bot to pass min spec. In the competition on June 7th, we got 2nd place!

From our experiences of building the robot, we have learned how much each part (the mechanical, electrical, and software) is interdependent on each other.