ECE118 Final Project - Slug World Cup

Anuj Pongurlekar. email: ajpongur.ucsc.edu
Eric Vetha. email: evetha@ucsc.edu
Jerek Cheung. email: jepcheun@ucsc.edu
- Mentors -
Dino Raphael. email: dfraphae@ucsc.edu
Dr. Tae Huh. email: thuh@ucsc.edu

1.1 DESCRIPTION/OBJECTIVE

The purpose of this project was to combine all the new concepts and skills we've learned through the labs this quarter and put them into one mighty robot. The goal of the robot was to autonomously navigate through a playing field and score ping pong balls into a goal. Since the robot needed to drive autonomously and react to different scenarios, we had to use the knowledge we learned from the first lab coding the Roach. The robot also needed to locate where the goal and goalie were when trying to score. This was done using skills from the second and third labs as both the goal and goalie had their respective beacons, the former emitting a 2.0 kHz and the latter a 1.5 kHz frequency.

The robot was also built using mainly MDF, which we laser cut and assembled using slot and slot construction, similar to what we did in Lab 2. Lastly, we incorporated the devices we learned to use in Lab 3, using an Uno 32 board to drive motors through device chips, such as H-Bridges, reading values through ADC pins, and driving RC motors. Although we had practiced all these skills before, putting it all together into this robot and having everything work to the highest level was much more difficult than expected. We would run into new roadblocks every day, and solutions wouldn't always be simple to find.

Overall, this was an extremely rigorous project that challenged us greatly, but it ended up being super fun to work on, and it was super rewarding to see the final product working as we expected.

2.1 STAGE 1: BRAINSTORMING

When the project and groups were first formed, we all had dozens of different ideas on how to tackle various challenges and concepts for the robot. While brainstorming, we collectively embraced a shared mindset: no idea was deemed too far-fetched or absurd. Although some concepts may have seemed unconventional, we fostered an environment where all suggestions were valued and considered. We simply listed everything that came to mind and tried to be as creative as possible. While some designs exhibited resemblances, there were also several unexpected and distinct approaches that pleasantly surprised the rest of the team.

We initially considered a few notable ideas, including:

- Mecanum drive
- Tension-Crossbow launcher
- Layer-cake tiered robot design

These ideas exemplified the team's willingness to explore unconventional approaches and think beyond traditional design boundaries. The mecanum drive wheel setup offered enhanced maneuverability, the tension-based catapult system showcased innovative projectile launching, and the layered robot design demonstrated a novel approach to component organization.

We spent some time considering an initial design, which we presented at the beginning of the project. Initially, we liked the maneuverability of the mecanum wheels and narrowed down our approach to using a layered chassis with a 4-wheel mecanum drive and a flywheel launcher. However, after hearing the criticism provided during the presentation, we decided to simplify the design significantly. Mecanum wheels presented an unnecessary complication, with arguably little benefit, and thus, we dropped them. Instead, we opted for a simple roach-like design, with a single bottom layer for motors, sensors, and their accompanying chips. Because of this, our chassis had to undergo minimal modification from the early stages. This allowed us to be flexible with various implementations of sensors and launcher components. After this, our design largely emphasized the simplicity and flexibility of components. As this was the first robot ever made by all members of our team, this seemed to be the best option, allowing us a degree of safety in simplicity but still enabling experimentation.

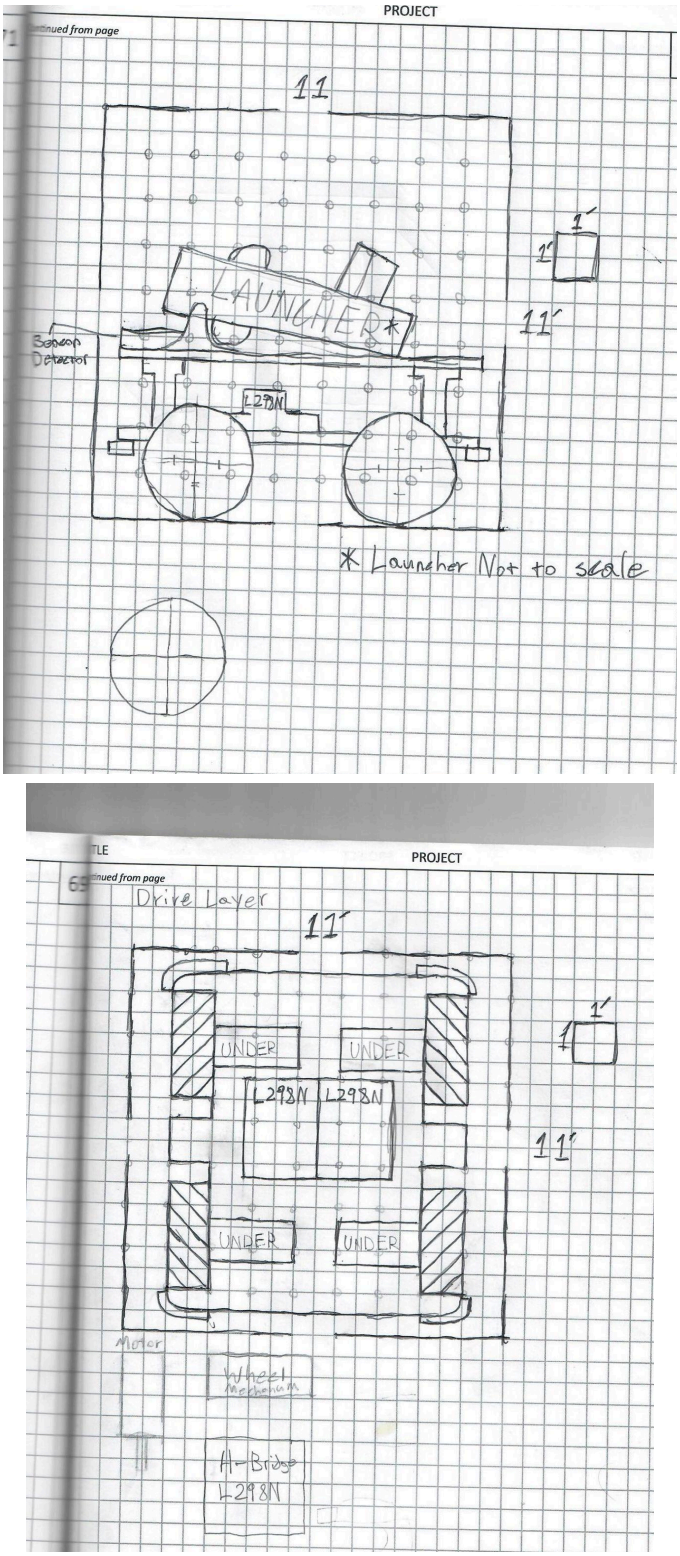

Figure 1: First Sketches for Robot Layout

From there, it would work best for the competition. We spent days thinking about this because, before we actually started building anything, we wanted to have a good idea of what our robot would look like. This meant we

had to spend time figuring the state machine out as well, as that affected how our robot should be structured, and vice versa. Ideas such as choosing to process our comparator through code turned out to be good when checking with tutors and the professor, but other ideas such as the mecanum wheels didn't receive the same reaction. We were advised to avoid them, as many groups who tried them for their robot in the past faced many difficulties, and in the end, it added more problems than it solved.

We knew that the goal would have a 2.0 kHz beacon on top, and the goalie would have a 1.5 kHz beacon that kept moving left and right. The beacon detectors we had from our previous labs could detect the 2.0 kHz signal from six feet away, but the robot would start in the reload zone, about eight feet away from the goal. We thought about many different ways for our robot to locate itself and the goal. Some ideas we had were locating the corner of the field or using ping sensors to figure out where the robot was on the field. These proved too noisy and inconsistent and would require an unnecessary amount of complexity in our state machine. The option we opted for instead was to create a more robust beacon detector so we could simply detect the goal from the reload zone, telling the robot which direction to go. We decided to also try using ping sensors as the more the robot knew about its surroundings, the better it could navigate the field.

Our next task was to figure out, once the robot located the beacon and drove to the proper zone, how to shoot the ping pong balls into the goal. Brainstorming different shooter ideas was fun, as we had a multitude of different concepts we had come up with for the midterm. As for actually shooting the balls, our best ideas were either a tension-based system using rubber bands or strings, or a flywheel that would spin and send ping pong balls flying out rapidly. We decided that the flywheel would work much more consistently, and we weren't sure if the tension method would always have enough power. However, for a flywheel system, we also needed a method of loading the ping pong balls one by one. After lots of discussion, we decided to use a tension system here that would use an RC Servo and either rubber bands or springs to push the ping pong balls into the flywheel.

3.1 MECHANICAL: ORIGINAL CHASSIS AND LAUNCHER DESIGNS

One of the first tasks we embarked upon was designing the CAD for our chassis and the initial launcher prototype. The chassis had to fit within a box measuring 11" by 11" by 11". However, since various modules needed to be accommodated on the platforms, we aimed to maximize available space. We opted for a two-layered chassis with spacers in between and proceeded to determine the placement of different components. Recognizing that the bumpers were to be positioned on the bottom platform, we utilized SolidWorks for CAD placement. To prevent the robot from getting stuck on walls or corners, we ensured that all edges and bumpers were rounded. Drawing inspiration from the roaches employed in the first lab, we designed two bumpers—one for the front and one for the back—surrounding four distinct bump sensors, enabling the robot to detect bumps from all directions.

Initially, our plan involved a four-wheel-drive system with four separate motors. However, before attempting to implement this design, we reconsidered. Having four different wheels seemed unnecessary, occupying more space without providing tangible benefits. We deliberated over whether to use skids or ball bearings that could roll at the bottom when the robot tilted forward or backward. Considering the motors for our drivetrain, supplied by BELS, were not particularly robust, we opted for ball bearings to minimize friction. We created slots for four ball bearings beneath the bottom platform. However, upon exploring our ball bearing options online and setting everything up on SolidWorks, we realized that the ball bearings would extend below the bottom of our actual wheels—a significant issue. Faced with difficulty in finding smaller ball bearings, we decided to mount them slightly above the lower platform to address this problem.

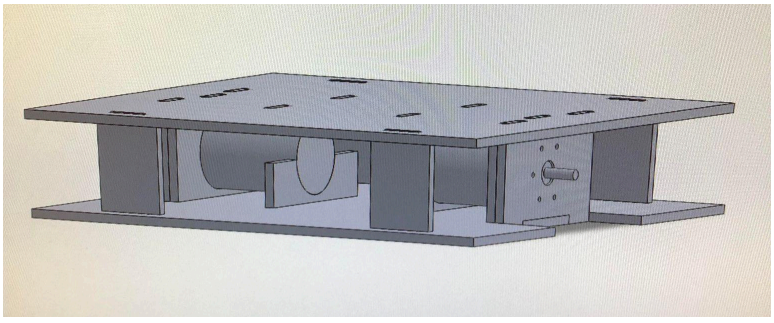

Figure 2: Original Chassis Design without bumpers

With all the bumpers and ball bearings spread out across the lower platform, there wasn't much space left. We knew we also had to place an H-bridge somewhere on the lower platform, which we positioned in the center right between our motors. There was no longer space for the Uno on the lower platform, so we made the decision to place it on the upper platform and create holes for the wires to feed down below. We anticipated that this might be troublesome later on when we wanted to change wiring or if certain wires got disconnected, but that was an issue we'd have to deal with no matter what. To account for this, we purposely created spacers so the top layer could easily be placed and removed.

We then shifted our attention to the launcher. The first part of the launcher was a tension system using an RC Servo to wind up, then release and push the ping pong balls into the flywheel. We created many sketches of how it would look, planned out the dimensions, and then decided to make our first prototype. We bought some PVC pipe that would fit perfectly for the ping pong balls, with just the right amount of space for them to move. We went through many different prototypes of this mechanism. We were definitely making progress, and it was starting to work, but after a couple of days of working on it, we started to question if it was even worth it. It began to feel like over-engineering when a much simpler solution was to have a downward slope, using the RC Servo to stop and let balls go through. We also realized the PVC pipe was unnecessary as it was harder to cut and much more difficult to mount.

Figure 3: Tension based Launcher Prototype

We gave our original ideas a shot but decided it was much smarter to make a shift and completely redesign our launcher rather than wasting more time. Within a day, we were able to laser-cut and start prototyping our new launcher design. It was far simpler, and we were sure it would work well. This process, which consisted of about the first two weeks, presented us with many realizations, but we made the smart decision of not clinging onto our original ideas too hard and making the correct adjustments. While it was definitely annoying at the moment, scrapping everything we had been working on for the past couple of days, it was the right decision and saved us much trouble later on.

3.2 ELECTRICAL: BEACON DETECTORS, BUMP SENSORS, AND TAPE SENSORS

A critical component of our early electrical prototype was the development of an effective beacon detector that could operate across the full length of the field. We opted not to utilize the beacon detectors created in Lab 2,

choosing instead to construct two new beacon detectors at frequencies of 2kHz and 1.5kHz. This decision was influenced by our intention not to use an LM339N comparator chip. Instead, we planned to feed the output of the peak detector into an UNO ADC pin and implement hysteresis digitally. This approach offered two major advantages. Firstly, it allowed us to quickly adjust the hysteresis bounds without modifying the circuit. Secondly, it enabled us to employ various averaging functions to filter out noise. However, this required us to expedite the development of the beacon detector to conduct thorough testing.

We decided to employ a full-size rail perf board for the project. The design for the beacon detectors closely resembled the one used in Lab 2, with some modifications to the filter. I chose to implement a 4th-order Chebyshev filter for both the 2kHz and 1.5kHz detectors, mirroring each other. Utilizing the TI filter design tool, I incorporated a 5-gain in the filter for each, providing additional amplification to the signal post-filter while still effectively attenuating undesired signals. The beacon detector then performed post-filter amplification of 100 for the 2kHz frequency and 400 for the 1.5kHz frequency. Finally, we equipped the beacon with its own voltage regulator to minimize any potential fluctuations that might arise if it were supplied power from a distribution chip.

Our initial prototype on the breadboard demonstrated a range of over 8 feet when tested with a beacon. Building on this success, we made the assumption that the 1.5kHz detector would perform similarly at that range and proceeded with its implementation.

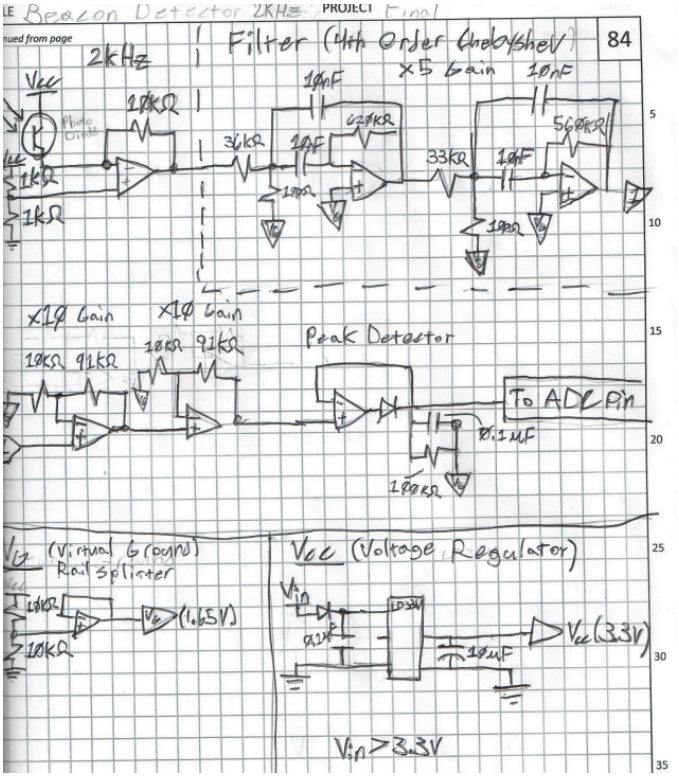

Figure 4: 2kHz Filter Circuit on Paper

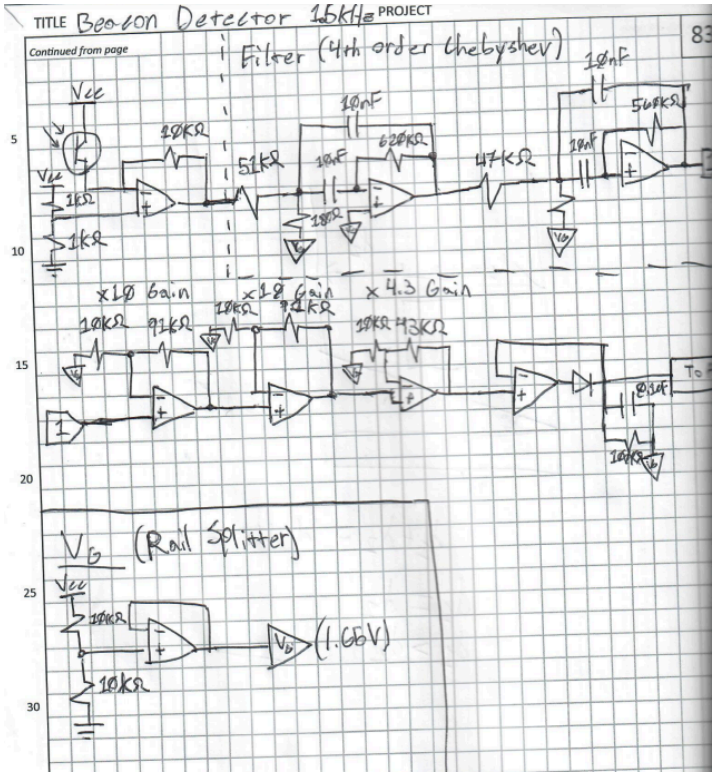

Figure 5: 1.5 kHz Detector Circuit on Paper

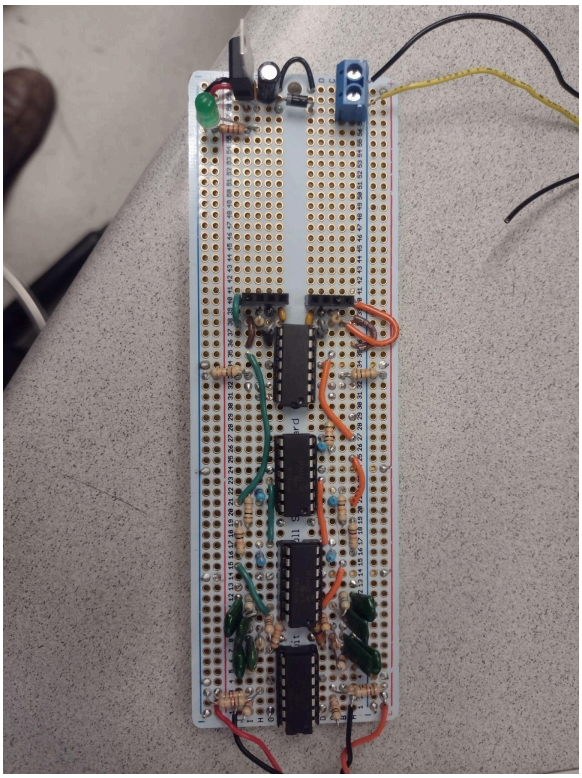
Figure 6: Dual beacon detector board. Left Side: 2kHz, Right Side: 1.5kHz

The implementation of the beacons was successful, with both the 2kHz and 1.5kHz operating with little issue. Using digital hysteresis, we also implemented a moving window average on the readings in software, limiting noise significantly.

After the beacon detector was built, the next step was to power the systems we knew would be on the bot. The first step in this process was a simple cable that would connect a battery to the UNO power distribution board. However, there was some uncertainty about which sensors we would be using in the final implementation. Still, we definitely knew that we would use both bump sensors and tape sensors. We decided to use four of each of these sensors. We then acquired the sensors and tested them independently with lab equipment.

The first consideration in implementation was powering these sensors. The bump sensors effectively acted as mechanically activated switches with a default pin and an active pin, and thus could be simply run to a 3.3v rail and ground. The default pin was soldered to a ground wire, and the active pin was soldered to 3.3v, with a lengthy output wire to connect to the UNO. Next were the tape sensors, for which we decided to use the pre-built circuits offered at BELS. These had a voltage input of 5 volts, a ground wire, and an output wire. These were male pins on the chip, so we used female-to-female jumper wires to connect them to power and ground and female-to-male wires to connect to the UNO.

To power each of these sensors, we first decided to make a power distribution board using a 3.3 and 5-volt power regulator. This would be the lower power distribution board on the bot, which would run power to these sensors. It would have four female slots for 5V power-ground pairs and seven female slots for 3.3V.

From there, the bump sensors were connected to the 3.3V slots, and the tape sensors into the 5V slots. The only issue that arose from this setup was that the wires to the bump sensors initially were weakly soldered, which led to the wires breaking, requiring a couple to be re-soldered.

3.3 SOFTWARE: WRITING THE LIBRARIES

The initial stages of software development primarily involved conducting tests on the various hardware components and writing libraries for them. Our first priority was to create PWM functions to effectively control the driving motors. Subsequently, we focused on the development of bumper and tape sensors. However, we encountered difficulties during the testing phase when we realized that these sensors were being powered externally rather than through the UNO32. This caused irregular results due to the absence of a shared ground connection for the pins. Additionally, we had to address the issue of inverting the output of the tape sensors in order to accurately sense tape. Once we successfully resolved these challenges and ensured that the functions for the bumpers and tape sensors were functioning as intended, we proceeded to implement their respective event checkers and services using the provided templates. Finally, we turned our attention to developing functions for the servo and flywheel motor.

We made the decision to utilize a digital comparator for our 2 KHz and 1.5 KHz beacon detectors, anticipating that it would offer greater flexibility for adjustments during operation. The validation of our choice came through testing these detectors, as well as their associated event checkers and services. We found that modifying the threshold was a straightforward process, and to our surprise, the beacon detectors were able to detect signals from a significantly greater distance than we initially expected. Furthermore, implementing a circular buffer to buffer the readings yielded even cleaner results, completely eliminating any noise interference.

After successfully organizing all the necessary basic libraries, we faced a challenge when we didn't have a robot available for testing. To overcome this obstacle, we decided to experiment with a ping sensor as an alternative. However, implementing the Interrupt Service Routines (ISRs) for the ping sensors proved to be a time-consuming task. Once we managed to get the ping sensors functioning, we discovered that they were highly susceptible to noise and proved ineffective for practical field applications. Notably, aiming the ping sensor towards any corner resulted in completely inaccurate readings. Furthermore, we observed that ping sensors were not well-suited for integration within an event-service framework, limiting their overall effectiveness.

4.1 MECHANICAL: MIDWAY CHECKPOINT

The competition date was getting closer now and at this point, it was our goal to have a close-to-final robot, mechanically and electrically, as soon as possible so we could start grinding out the state machine debugging. We had gone through many renditions of the chassis, but it was finally almost done.

After starting to add all the electrical components, we realized we needed far more space between layers, so one of the changes we made was increasing the length of the spacers. This gave us much more room for the wires, and the robot looked far less cramped. Another major change we made to our chassis was the shape. At first, our platforms were almost in the shape of squares, meaning our bumpers had to be that shape too. When we placed the robot on the competition field, though, and ran some tests, we realized there was barely any space for the robot to move around and spin in the reload zone. Due to the sharp corners of the bumpers, the robot would often get stuck,

and there was nothing we could do about it since there was little to no room to react to bumps efficiently. To fix this, we did a quick redesign of the platforms and bumpers and rounded them out more. The first fix still wasn't enough, but we then rounded the bumpers out even more in the parts they were getting stuck, and after testing, the robot no longer got stuck.

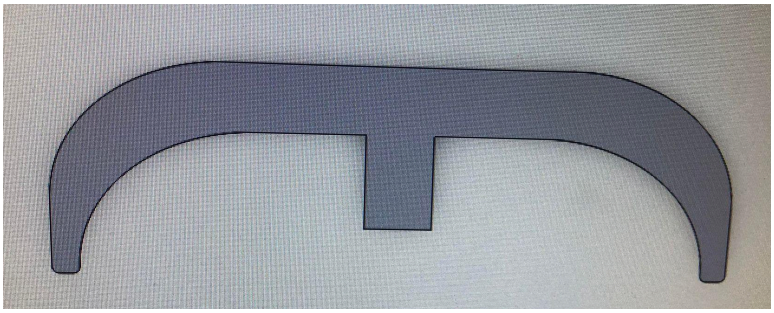

Figure 7: Rounded Out Bumpers

Progress on the launcher was going well but we kept running into small issues that kept it from performing the way we wanted it to. The original flywheel we had gotten had a soft rubber tire that fit into place around the wheel. However, while this fitment seemed fine and secure normally, when we began to run the flywheel motor at above 50%, the tire would start to slip off. We tried different solutions to keep it on, experimenting with all sorts of glues, but the issue persisted.

We then ordered a different flywheel which took a couple of days to come. However, that flywheel then presented its own problems as it had a large diameter meaning it wouldn't spin as fast. It could maybe shoot ping pong balls far enough to score from the one-point zone, but our ambitions were far higher than that.

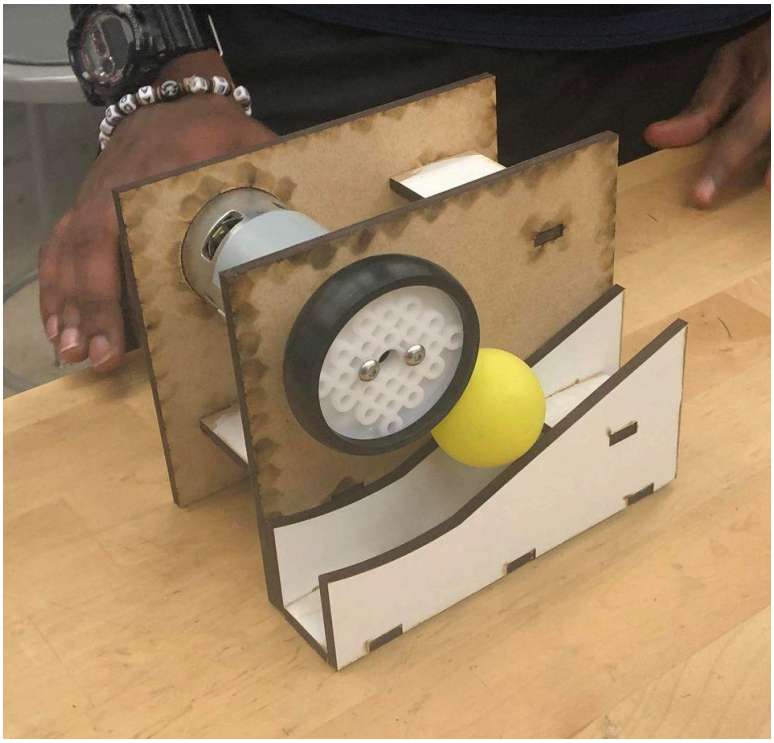

Figure 8: MDF Launcher Prototype

Finally, as we drastically wanted to complete the mechanical aspects of the robot and move onto the software, we started testing with other group's flywheels to see which would fit our design best. We decided on a small rubber flywheel with lots of give, meaning it would work for all sized ping pong balls with ease. Once it arrived, we recalculated the measurements for how high the flywheel should sit and after some testing with a couple different heights tested, we had our final launcher and chassis.

4.2 ELECTRICAL: MOTOR INSTALLATION, 5V REGULATOR, DETECTOR HOUSING

At this stage in the project, having a somewhat more functional chassis, we were able to properly attach the motors and H-Bridge into the base of the bot, and could have movement from the bot. The H-Bridge was simply powered directly from the UNO power distribution board, with PWM data pins coming from the top-mounted UNO.

The addition of movement to our prototype required a bit more control over the systems of the bot, so we decided to implement a power switch onto the existing battery cord on the power side. This allowed us to quickly turn on and off the robot.

While we were still considering the use of ping sensors, we decided to make a board in order to accommodate the use of the ping sensors and any RC servos we may be using. This secondary power distribution board had only 5 volt pairs. This would have 6 female slots in order to accommodate the upper estimate of powering 2 RC servos and 4 ping sensors. Only one of these slots would be used to power an RC servo after we scrapped the ping sensors.

Lastly, to increase the accuracy and further reduce the noise of the Beacon detector, we had taken to using Foamcore and electrical tape in order to house the detector's Photo-transistors. We used a wider triangular housing for the 2 kHz, for a broad range of detection, with a top-side bias. This housing suited the 2 kHz, as we wanted a broad search area to find the general direction of the beacon. We used a much smaller black banana plug case as the house for the 1.5 kHz transistor, in order to limit the detection area to a very specific direction. We decided on this housing for the 1.5 kHz transistor, so that we would only detect the goalie when it is directly blocking our launcher.

4.3 SOFTWARE- INITIAL STATE MACHINE

Having decided to discontinue the use of the ping sensor, we proceeded to focus on the development of the state machine. We carefully devised a comprehensive plan, outlining the intended states for the state machine. The visual representation of the planned states can be observed in the graphs below.

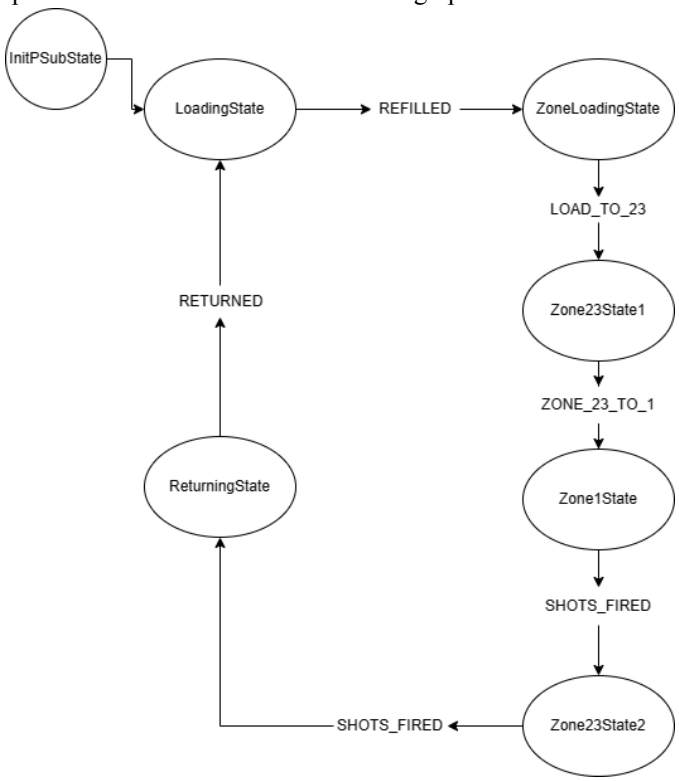

Figure 9: Top State Diagram

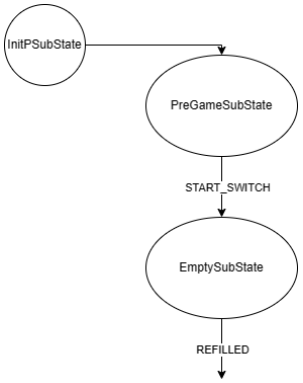

Figure 10: Loading Hierarchical State Machine

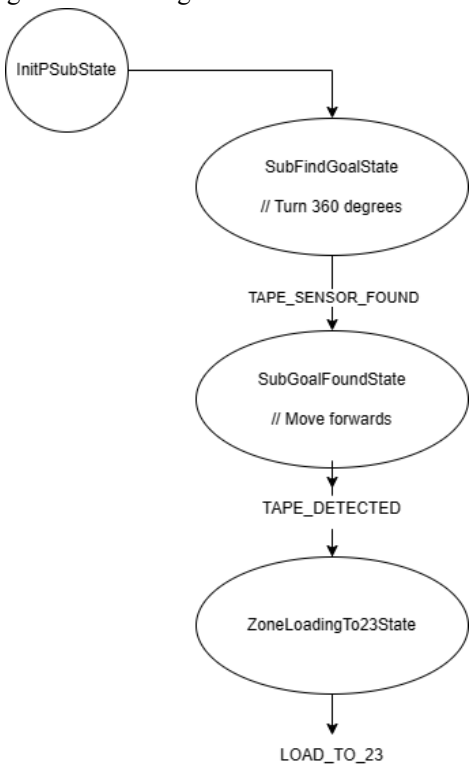

Figure 11: Zone Loading Hierarchical State Machine

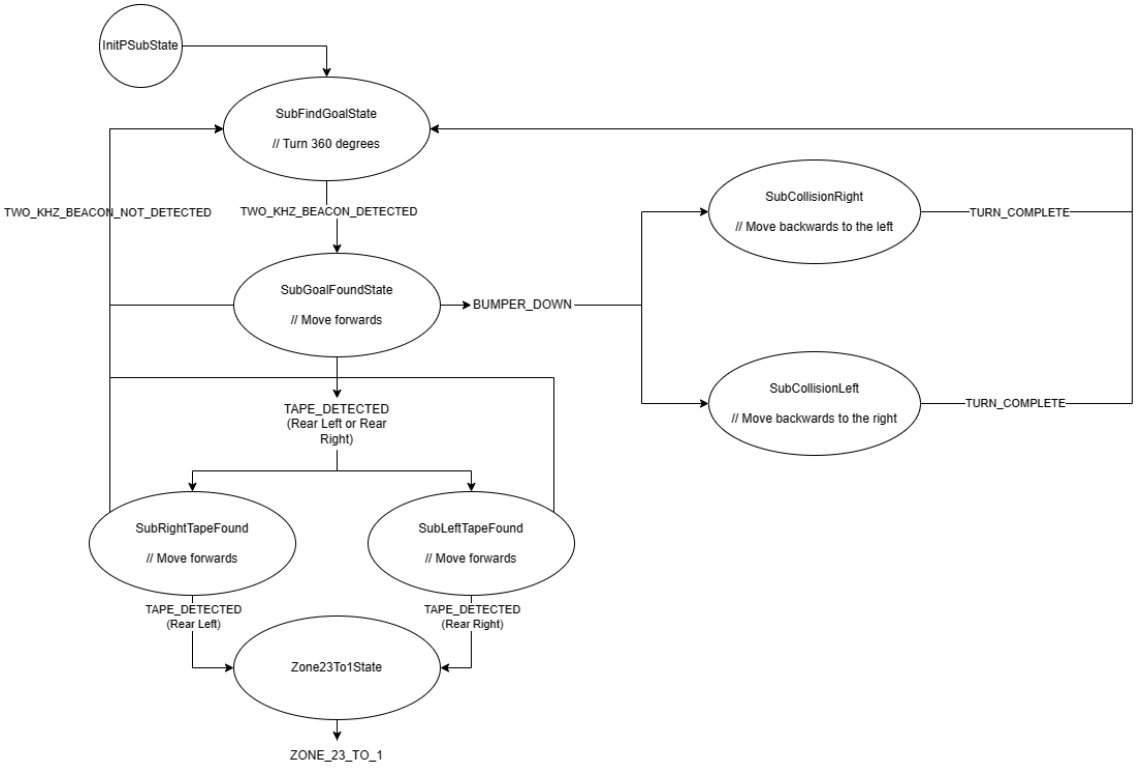

Figure 12: Zone 23 State 1 Hierarchical State Machine

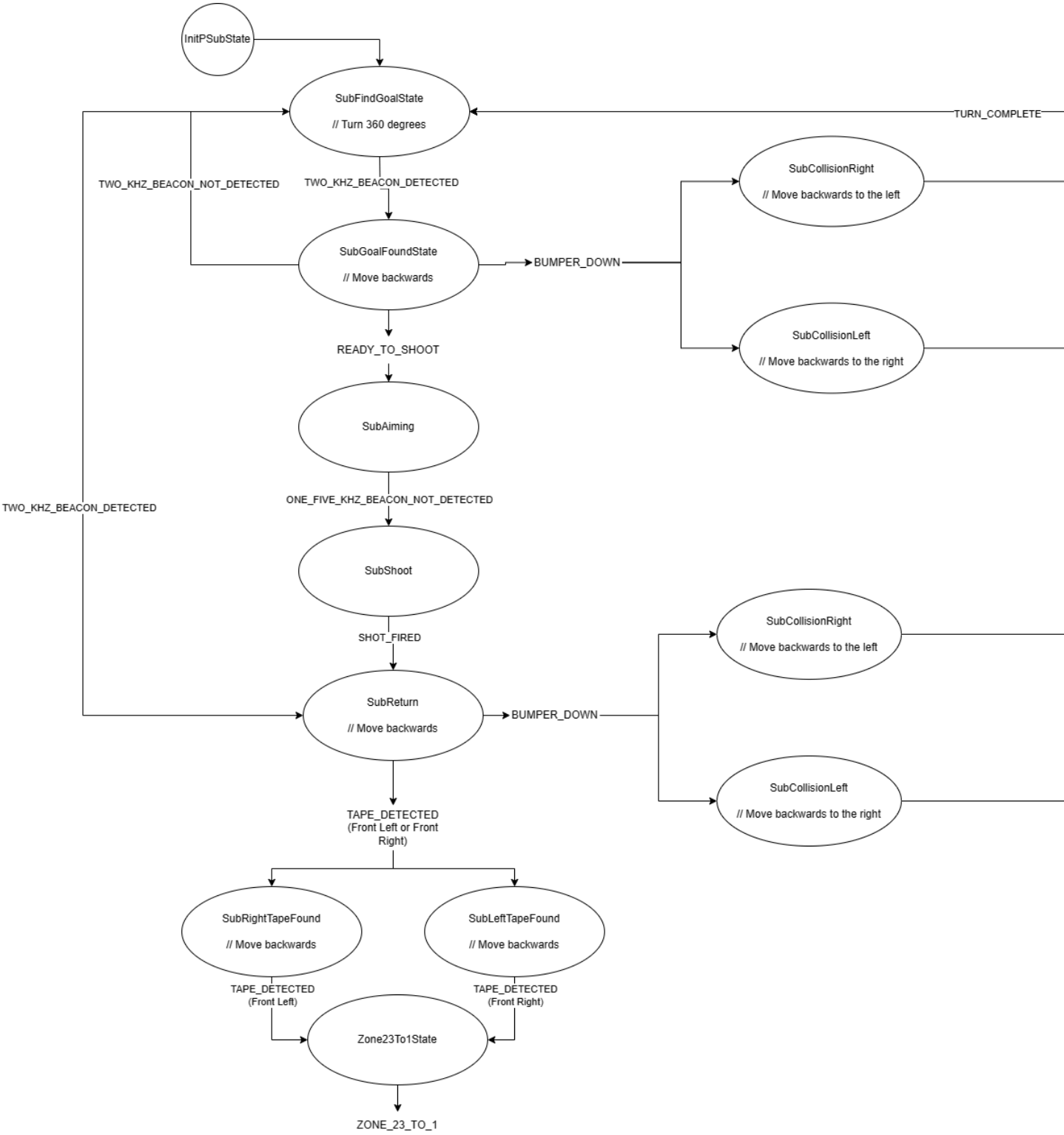

Figure 13: Zone 23 State 2 Hierarchical State Machine

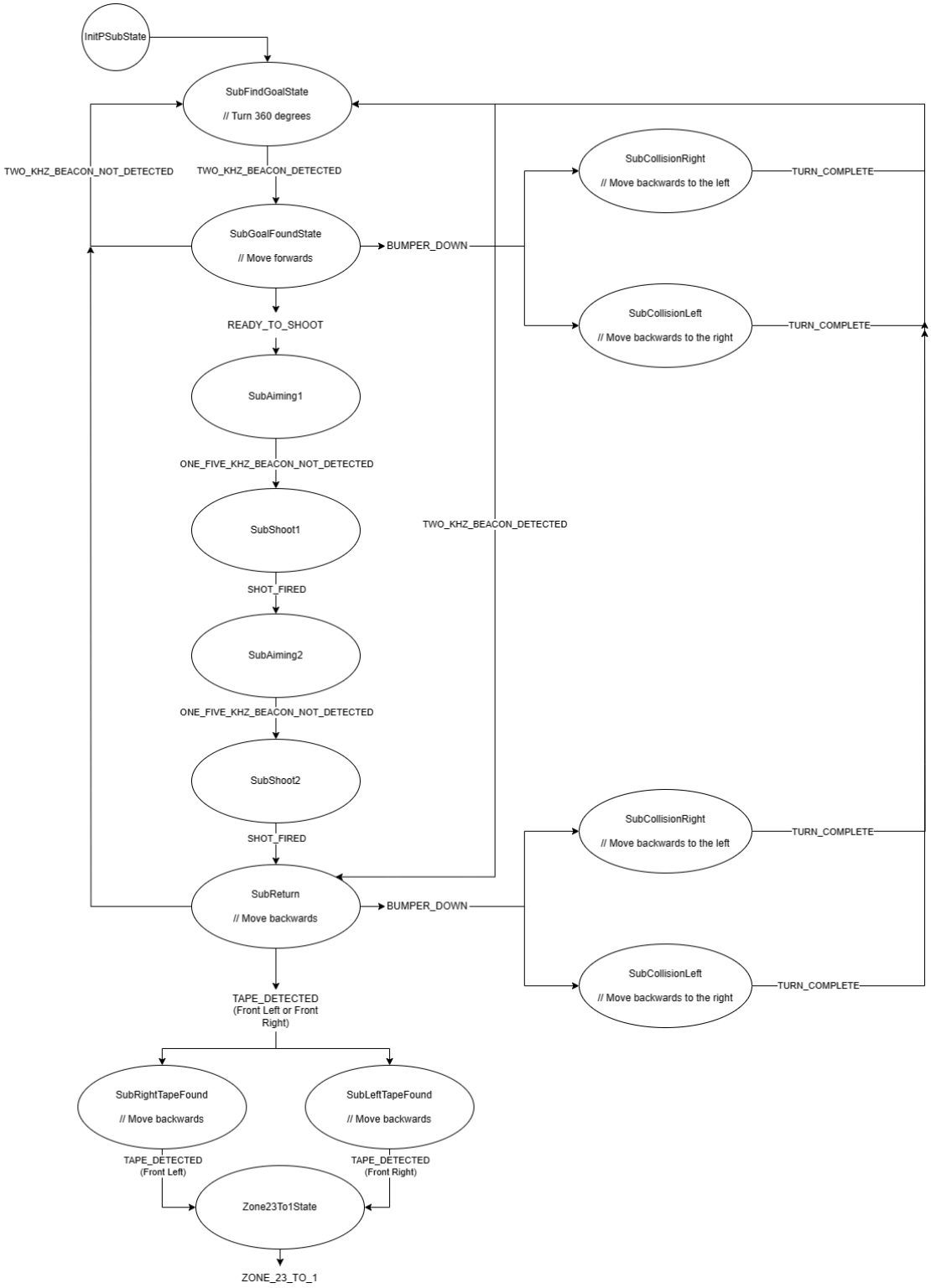

Figure 14: Zone 1 State Hierarchical State Machine

Figure 15: Returning Hierarchical State Machine

5.1 MECHANICAL: FINAL FIXES

At this point, we were all in on state machine testing and wanted to change the mechanical design as little as possible. Major mechanical changes would only delay the testing process. One of these minor changes was adding a guard roof for the bumpers. When the robot hit bumps at high speed or the occasional odd collision with another robot, the bumpers would sometimes get jammed. To fix this, we added small pieces of heat shrink that essentially kept the bumpers from moving up and down. This fix was rather makeshift and fast, but it did the job and worked well.

Another small issue we were having was due to the robot shaking through testing and matches, bolts would often loosen and sometimes even fall out. We didn't want to have to keep tightening screws, so a quick, easy solution was to use hot glue and keep the bolts in place. Aside from that, our robot was mechanically finalized, and all our focus was on the software.

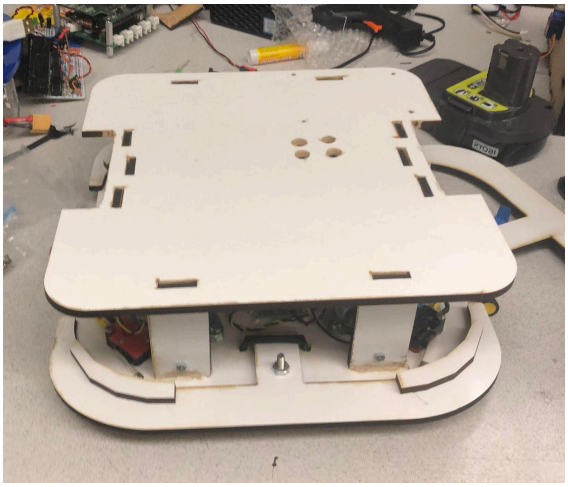

Figure 16: Final Robot Chassis

5.2 ELECTRICAL: POWER SWITCH AND BEACON HEAD ADJUSTMENTS

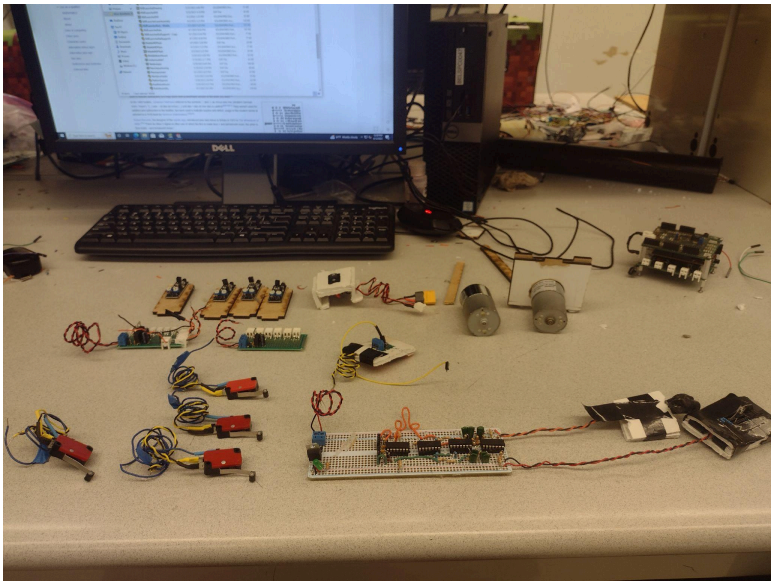

Figure 17: Final set of electrical components

During this stage of the project, we found it necessary to remake the housing for the 2 kHz Beacon detector, as the current configuration had some biases that would cause strange behavior in our state machine.

In order to address this, I decided to make a new board and housing explicitly for the 2 kHz. This board consisted of 2 rails, with a series of plugs on each side. One rail was connected to the power rail on the Beacon detector, and the other to the input of the phototransistive op-amp. The plugs on these rails allowed us to adjust the position of the phototransistor, allowing us to expand or shrink the detection area of the phototransistor.

The top of the housing was loosely taped onto the lower piece containing the board to be easily removable. This setup worked well, eliminating some bias, but there were occasional issues with the rails shorting in on the screw pins. However, this could be avoided if the wires are properly adjusted and if they are properly insulated.

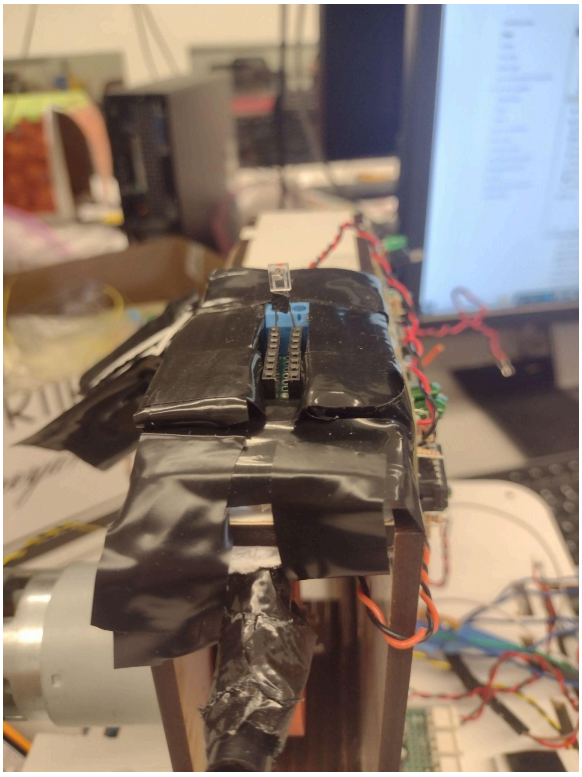

Figure 18: Adjustable phototransistor rails with top cover removed.

Also, in order to meet the minimum specifications, we needed to know on which side of the field we started. Initially, when considering the use of ping sensors, we could determine the distance from one side to the other after locating the beacon. However, with that option off the table, we opted for the brute force method of adding a switch that we would flip at the beginning to indicate our starting side. The switch was relatively simple, with inputs plugged into a 3.3V distributor in the motor base, then outputting to the UNO via a jumper cable.

We experienced some electrical issues regarding the power switch. In some cases, a fully charged battery would not turn on the UNO unless the pins were being measured with a multimeter when the switch was flipped. Treating this as an intermittent issue, we eventually speculated that it worked with the multimeter due to the added capacitance. Consequently, we added a bypass capacitor to the power switch to permanently introduce similar capacitance. After this implementation, it worked without error. This was the last modification made to the bot before the competition.

5.3 SOFTWARE: STATE MACHINE TESTING

Testing the state machine proved to be a challenging endeavor due to the inconsistent behavior of the robot's motors. Regardless of receiving the same commands, the robot's navigation outcomes were never consistent. Consequently, significant modifications were required to meet the minimum specifications.

The first crucial adjustment involved addressing the issue of the robot's inability to drive straight. To tackle this problem, we implemented a linear PID controller to ensure that the robot consistently followed the beacon, even in situations where it lost sight of it momentarily.

To prevent the robot from crossing the middle line, which would disrupt the functioning of the state machine, we introduced a tilt towards either the left or right side. The direction of the tilt was determined based on the robot's initial loading zone. This adjustment effectively ensured that the robot remained on one side of the field. From an observer's perspective, this approach appeared similar to a wall-following algorithm. However, in reality, the robot simply tilted towards one side and corrected its orientation whenever it lost sight of the 2 KHz beacon.

Another significant improvement was ensuring that the robot consistently started from the same position for the sake of maintaining consistency. Without this measure, the final position of the robot would vary with each run. To address this, we programmed the robot to locate the closest wall and align itself along that wall and the loading

zone tape before beginning its operation. This approach guaranteed that the robot consistently started from a standardized position, allowing for reliable and repeatable performance.

Furthermore, due to the distinct state machines required for different sides of the field, we implemented two separate hierarchical state machines, one for the left side and another for the right side. These state machines encompassed the necessary functionalities such as robot orientation, movement towards the goal, and transitioning from one side of the field to the other.

In instances where a collision occurred, the corresponding state machines were activated. When a collision was detected, the robot would promptly return to its loading zone, proceed to the opposite side of the field, and then adhere to the state machines designated for that side. Upon reaching the opposing loading zone, the robot would realign itself with the correct loading zone before resuming its navigation towards the goal. This process was repeated as the robot traversed back and forth between the two sides of the field, ensuring the appropriate execution of the desired algorithms based on its location.

The finalized state machines that successfully meet the minimum specifications can be found below. The actual files of these state machines, and the rest of our code, can be found in our shared repository (https://github.com/ericdvet/mech-final-project). These state machines represent the culmination of our efforts, incorporating all the necessary functionalities and behaviors required to accomplish the designated tasks. Through careful iteration and refinement, we have ensured that these state machines align with the specified minimum requirements, enabling our robot to navigate the field efficiently and complete the assigned objective.

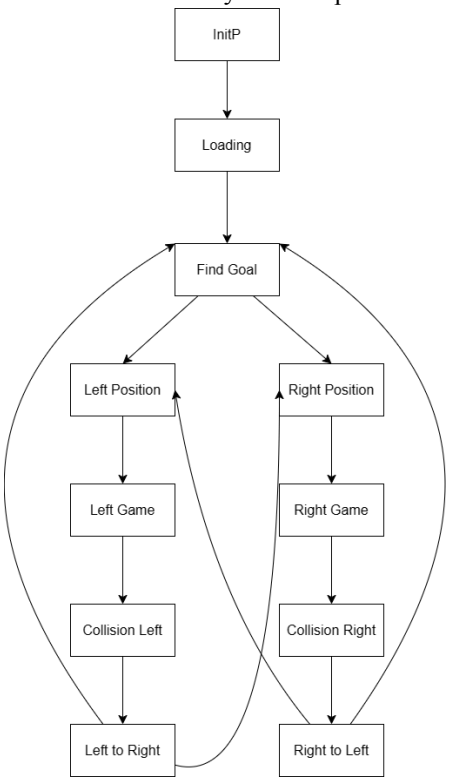

Figure 19: Top State Machine

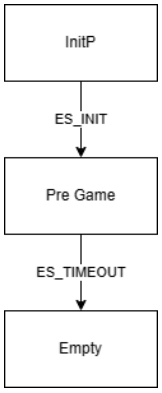

Figure 20: Loading Hierarchical State Machine

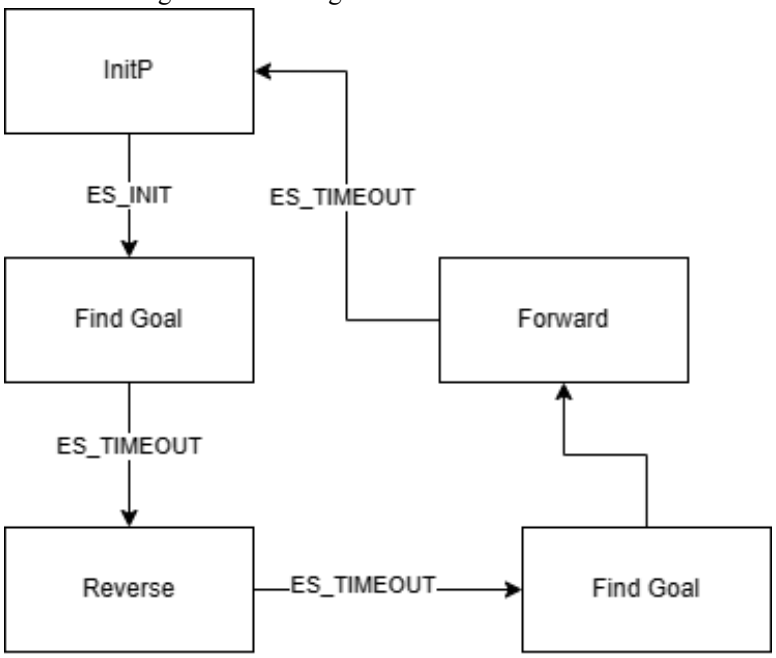

Figure 21: Find Goal Hierarchical State Machine

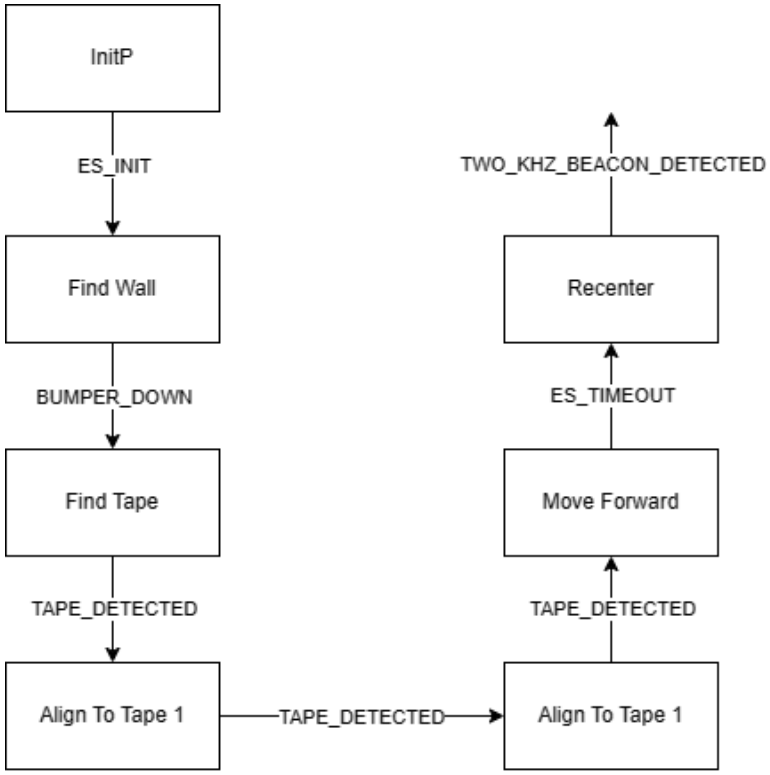

Figure 22: Right and Left Position Hierarchical State Machine

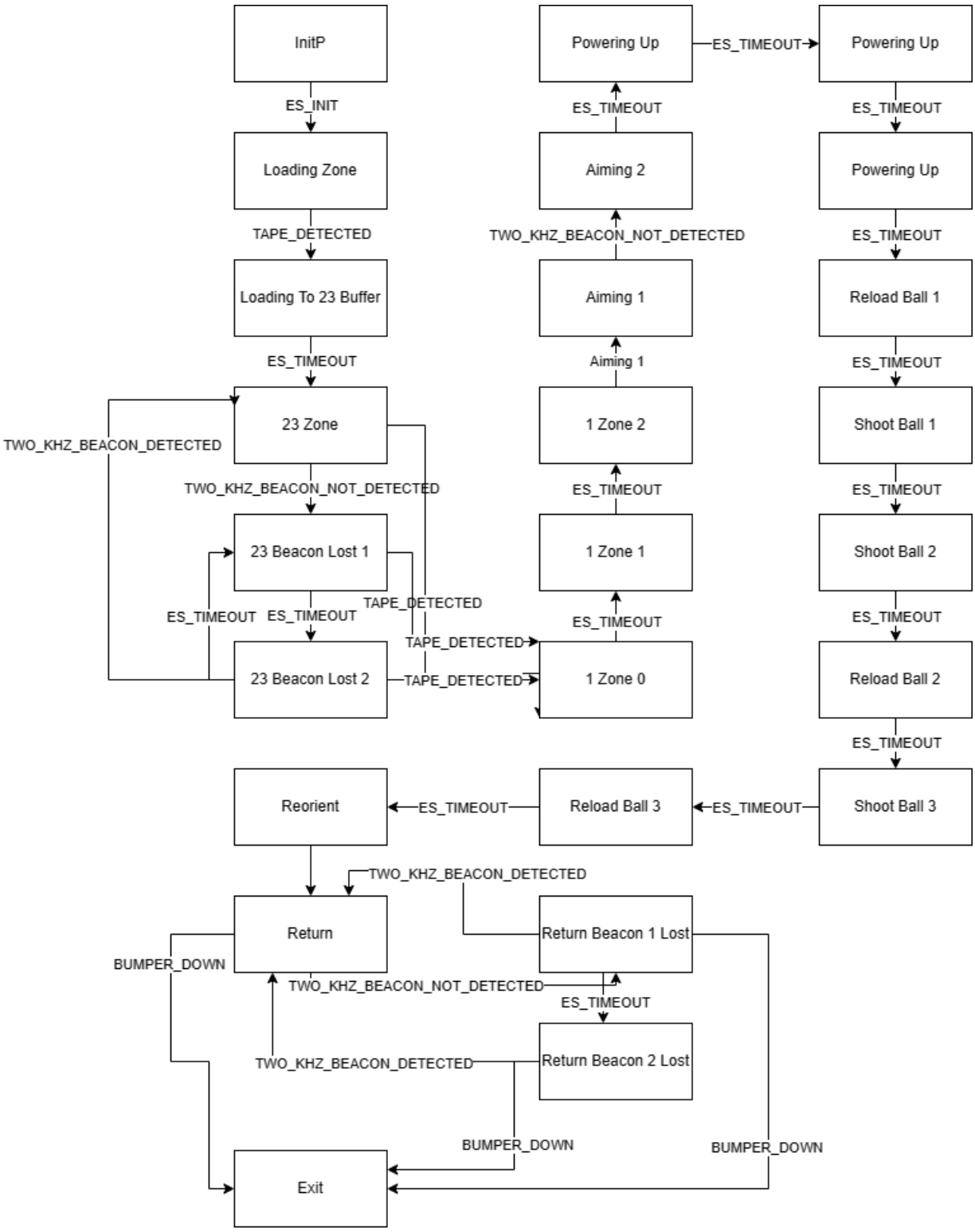

Figure 23: Left and Right Game Hierarchical State Machine

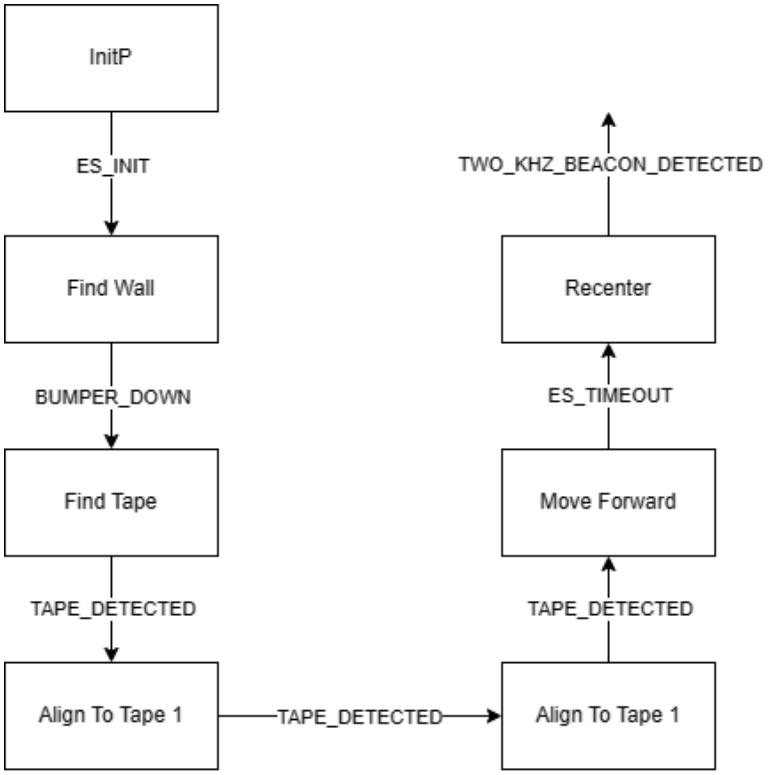

Figure 24: Left to Right and Right to Left Figure Hierarchical State Machine

6.1 CONCLUSION

Overall, this project required all of us to be at our best for five weeks straight. We all experienced many sleepless nights and moments when we desperately needed the support of our teammates. This robot was certainly not built overnight, and we were all very thankful for not procrastinating throughout the weeks. We weren't the fastest team to the finish line, but we always managed to produce solid work slowly and steadily, so it wasn't a week filled with countless all-nighters. Another reason for our success was our method of splitting the work. Each category—mechanical, electrical, and software—had a lead and a backup, ensuring that there were always two sets of eyes overseeing the process. It's often easy for someone to make small mistakes or forget a key detail, and always having a second pair of eyes saved us several times.

Even though things didn't work perfectly as expected every time, we didn't let ourselves get discouraged and embraced the engineering process. Every roadblock or failed prototype was viewed as a learning opportunity and simply part of the process. In the end, the robot looked nothing like we first envisioned it, but the core properties remained the same. If we were to do this project again, there would be a couple of things we'd do differently. First, we'd make sure to allocate much more time to the final stage, state machine testing. While this might mean rushing the other sections a bit more, the groups that had a drivable robot the earliest were the ones that had the most success in the end.

We realized in the end that the best robots weren't the ones with the best mechanical features but the best state machines. Another thing we'd do differently would be to try more creative designs or use different types of sensors. As mentioned, we experimented with ping sensors but eventually decided against them as it was taking too long. However, there's a variety of different sensors and mechanical designs that could have worked really well and would make our robot much more unique. For this competition, we were simply trying to get our robot done and working, but if we were to do this again, we'd definitely try to take more risks and experiment with more unique solutions for our robot.

Without a doubt, though, this competition was one of the most difficult and fun projects of our lives, and if we were ever super bored and depressed, we'd love to do it again.

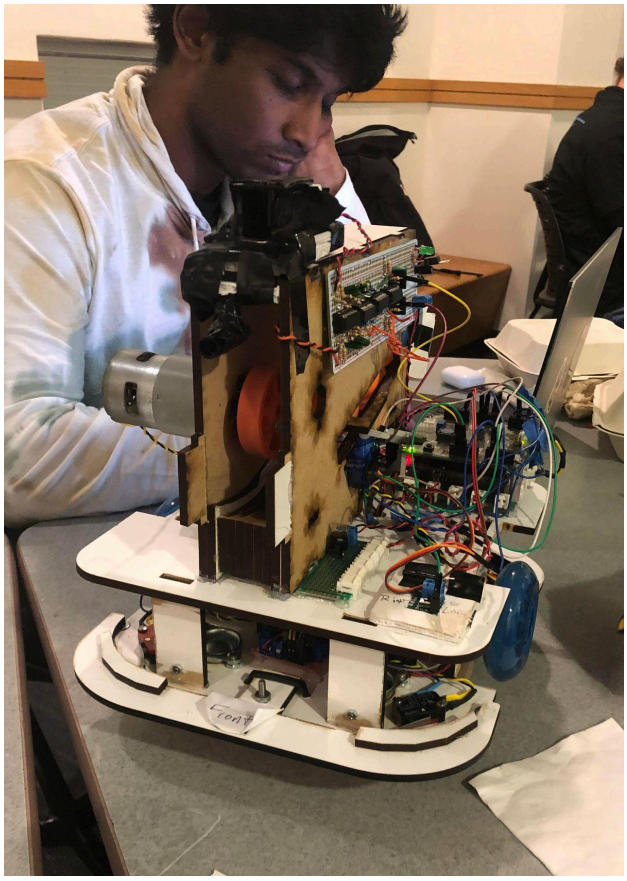

Figure 25: Final Robot and Slumbering Eric