Introduction to Mechatronics

Sluggy Potter and the Goblet of Slime

ECE 118 Final Project

Kyle Nip Sean Layno Daniel Fiallo

06/10/2022

Table of Contents

Introduction **3**

Procedure **3**

Mechanical 3

Electrical 9

Software 14

Conclusion **21**

Introduction

This project represents a comprehensive application of the concepts and techniques explored in the *ECE 118 Introduction to Mechatronics* course, culminating in a challenging yet stimulating task. Our primary objective is to engineer an autonomous robot capable of adeptly navigating a specified field. The robot's mission involves seeking out towers that emit a 2 kHz signal, discerning the correct side of these towers broadcasting a 25 kHz signal, and subsequently scoring on a target demarcated by 2-inch wide black tape. To achieve this intricate task, our approach is segmented into three distinct yet interdependent domains: mechanical, software, and electrical engineering.

1. *Mechanical Design:* This facet focuses on the structural development of the robot. It entails crafting a robust and functional frame that not only supports the robot's physical integrity but also strategically accommodates all necessary electrical components. Considerations for space optimization and component integration are pivotal in this phase.

2. *Software Development:* At the heart of our robot's functionality is the software that governs its operations. This encompasses the programming required for controlling motors, infrared (IR) sensors, servos, and bumper mechanisms. Furthermore, it includes the development of an intelligent state machine algorithm, which forms the behavioral backbone of the robot, enabling it to interact with its environment in a meaningful and goal-oriented manner.

3. *Electrical Engineering:* This component is the lifeblood of our project, involving the meticulous assembly and integration of all electrical elements. From power regulators to the specialized beacon detector, each electrical part is crafted by hand, ensuring a tailored fit for our robot's unique requirements. The precision in this segment is crucial, as it underpins the effective and efficient functioning of the robot.

Overall, this project not only challenges us to apply our accumulated knowledge but also to synergistically combine different aspects of mechatronics, thereby fostering a deeper understanding and appreciation of this multifaceted field.

Procedure

Mechanical

The initiation of our project involved an in-depth collaborative discussion, where we pooled together the most promising aspects of our individually conceptualized robots. This brainstorming session aimed at synthesizing three core ideas into a cohesive design for our final robot. To facilitate this, we hand-drew sketches of each proposed design, providing a visual basis for our deliberations.

During these discussions, we conducted a critical analysis of each design, identifying potential shortcomings and brainstorming viable solutions to address these issues. This phase was crucial in refining our ideas and ensuring that the final design was not only innovative but also practical and feasible.

After reaching a consensus on the primary design that best encapsulated our collective vision and addressed the identified challenges, we proceeded to the modeling phase. Utilizing SolidWorks, a sophisticated 3D CAD software, we transformed our conceptual design into a detailed and accurate digital model. This process involved meticulous attention to detail to ensure that every component of the robot was accurately represented.

The culmination of this phase is illustrated in Figure 1, showcasing our completed prototype. This visual representation is a testament to our collaborative effort and the meticulous planning that went into the design phase of our project.

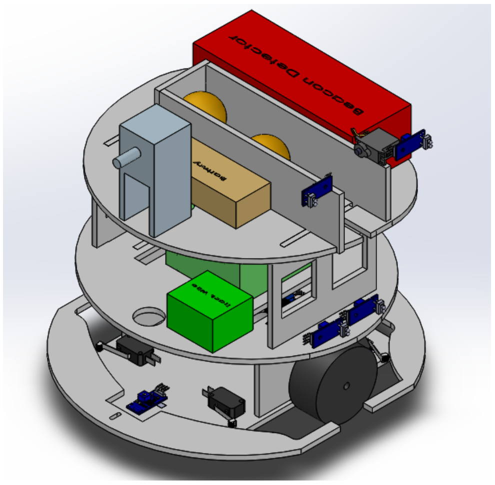

Figure 1: SolidWorks Placeholder Design

We decided to go with an elliptical-shaped robot, since we thought it would be the most ideal in dealing with the bumpers. Our first prototype of the robot involved using only a front bumper and three tape sensors. After further testing and various edge cases, we decided to use five tape sensors instead and added a back bumper to help line up our robot to the wall of the towers. Figure 1.1 displays our robot without any of the electrical components.

We added extruded cuts to our design for better cable-management and to make it more organized. There are circle extruded cuts located in each layer for easy access between each other and square extruded cuts on the side panels for cables that need to be put from the side. We verified on SolidWorks that there were no interferences with any of the parts and moved onto laser-cutting our design using MDF. There were no issues when we tried assembling all three different layers together, thus we began with the construction of the bottom layer of our robot.

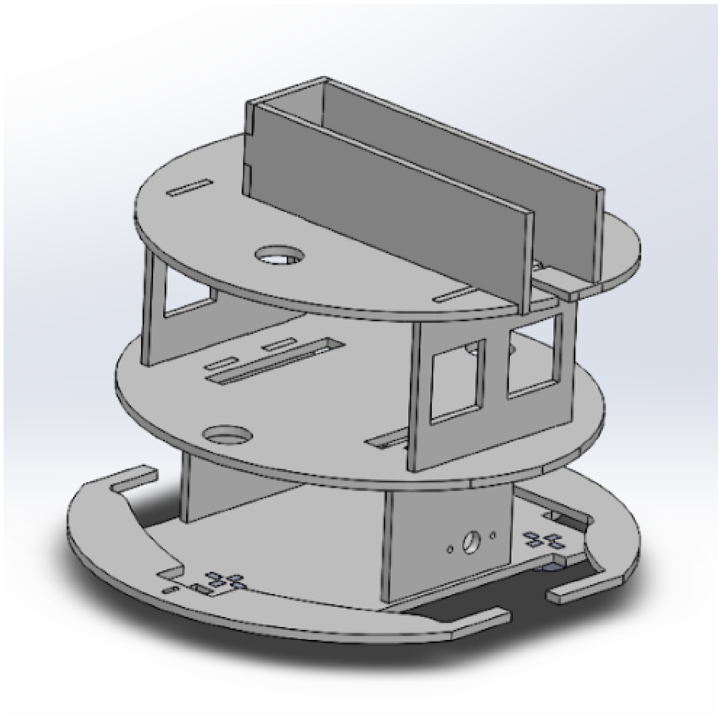

Figure 1.1: Empty Frame of Final SolidWorks prototype

The foundational layer of our robot incorporates a pair of DC motors, an H-Bridge, four strategically placed bumper sensors, and an infrared (IR) sensor. The design of the bumpers meticulously considers impacts from various angles, maximizing protective coverage. To counteract the torque-induced movement and ensure a stable and secure drive, the motors are firmly bolted to the side chassis. Our initial motor mount design featured dual side wall fittings; however, subsequent revisions revealed that a single-sided bolting mechanism sufficed. The DC motors' shafts, being somewhat short, necessitated custom extensions. These were expertly crafted using a 3D printer, designed to mesh seamlessly with the hexagonal shape of the wheels. Upon assembly, we observed that the wheels were elevated due to the excessive height of the skegs. We filed the skegs to achieve an even, functional height. Once balanced, we integrated the bumper sensors and the H-Bridge, conducting thorough tests to identify and rectify any functional discrepancies.

The intermediate layer of our robot houses the principal electrical components. This includes the Uno Stack and I/O interfaces, along with two power distribution boards: one to reduce the voltage from 9V to 5V, and another to further step it down to 3.3V. Additionally, it accommodates the track wire circuit and two lateral IR sensors, integral for accurate scoring. Designed for ease of access to both upper and lower electrical elements, this layer necessitated careful spatial planning to accommodate all requisite components. Each circuit was rigorously tested for functionality before being securely mounted on the platform with stand-offs.

The uppermost layer of the robot features our innovative shooting mechanism. It consists of a servo that methodically releases ping pong balls, complemented by an initial slope crafted from formacore to leverage gravitational force. A secondary slope is strategically placed atop the first to regulate the ball release sequence. The circuit board for our beacon detector is securely fastened to the medium-density fibreboard (MDF) platform, minimizing any potential displacement. Two additional IR sensors flank the shooting mechanism, playing a crucial role in aligning the robot with the tower sides. Furthermore, we have meticulously designed a compact stand and housing for our phototransistor, ensuring it remains focused on the beacon and resistant to signal interference from adjacent towers. The beacon housing, constructed from aluminum, is engineered to minimize external signal interference, featuring a singular aperture for precise signal detection.

Electrical

One of the main objectives of this project was to be able to design an analog filter able to only pick up a 2.0 kHz signal emitted from a beacon. The first step was to ensure all circuit boards had the appropriate voltage rating, thus we designed two power distribution boards, one for 5 V and another for 3.3 V that would supply IR sensors, trackwire detector, beacon detector, H-Bridge, bump sensors and servo.

Power Distribution Boards Schematics (5V and 3.3V):

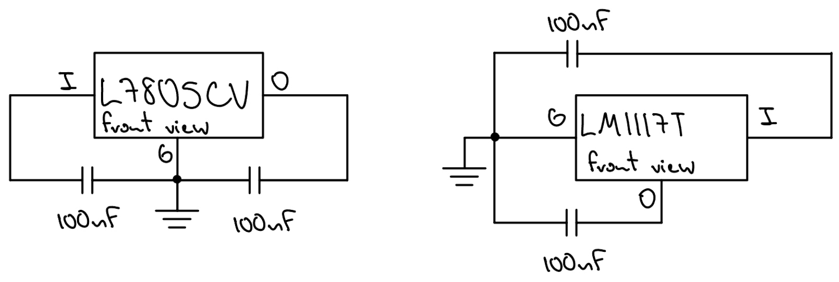

Figure 2: 3.3 V and 5 V rails.

Before implementing the circuit on the proto-board we first implemented it on a breadboard of the same size to ensure functionality. We found that with our filter, 2nd order Butterworth Bandpass from TI instrument designer, needed to increase the gain to about x242 to get the input to the filter to be maxed at 3.3 V when the 2.0 kHz beacon was at a distance of 6 ft. This meant that the input to the filter would be maxed at 3.3 V when the 1.5 kHz and/or 2.5 kHz at a range of 1 ft. During our testing the output of the filter for the incorrect frequencies (1.5 kHz and/or 2.5 kHz) were at a maximum peak of ~2.5 V with the peak-to-peak value of ~2.10 V and the maximum peak of the correct frequency (2.0 kHz) were 3.3 V with peak-to-peak value of ~3.3 V.

This seemed to be adequate filtering, however, the output of the peak detector was capped at ~3 V instead of 3.3 V, leaving only ~0.3 V difference. This still seemed to be an adequate difference, however, after calculating and implementing the resistor values for hysteresis bounds of 2.9 V (upper) and 2.4 V (lower), the Comparator circuit would simply switch between high and low at the threshold of 2.4 V. After much altering and studying the comparator, it was determined that the LM339 IC would only allow a hysteresis bound of maximum (Vcc - 1) V meaning while running the chip with 3.3 V the upper limit is 2.4 V. Due to this,we configured the LM339 to run on 5 V while the rest of the circuit runs on 3.3 V. This means that we included two voltage regulators on the circuit. The first outputting 5 V for the LM339 and the second outputting 3.3 V to the rest of the circuit.

After soldering the filter on our perfboard we found a significant amount of noise in the signal throughout the circuit as well as significant attenuation of the 2 kHz signal from 6 ft away. To solve this we kept our passive highpass filter with a cutoff frequency of ~1.9 kHz, 10 nF capacitor and 3% 10 kΩ resistor, but changed out our Butterworth filter to a 2nd order Chebyshev Bandpass filter with the same topology and capacitors but different resistor values. We also included a 10 μF capacitor across the 3.3 V power rail and reference ground to minimize noise. This decreased the noise in the transresistive op-amp and phototransistor which perpetuated through the 4 gain stages and gave the filter a strange waveform. The output of the comparator running on 5 V power was 5 V when the signal was not detected, we first tried a voltage divider to only input 3.3 volts into the LED/buffer stage but since the LED's buffer runs on 3.3 V the output of the buffer should be capped at 3.3 V instead of 5 V. We found that the finished product does not detect the 1.5 kHz/2.5 kHz beacons at all, and can detect the 2.0 kHz beacon from over 7ft away.

Beacon Detector Schematic

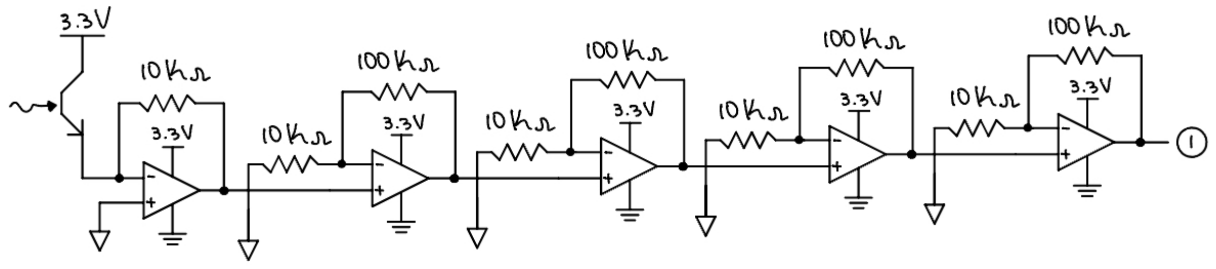

Figure 3: Phototransistor and transresistive op amp with 4 stages of gain.

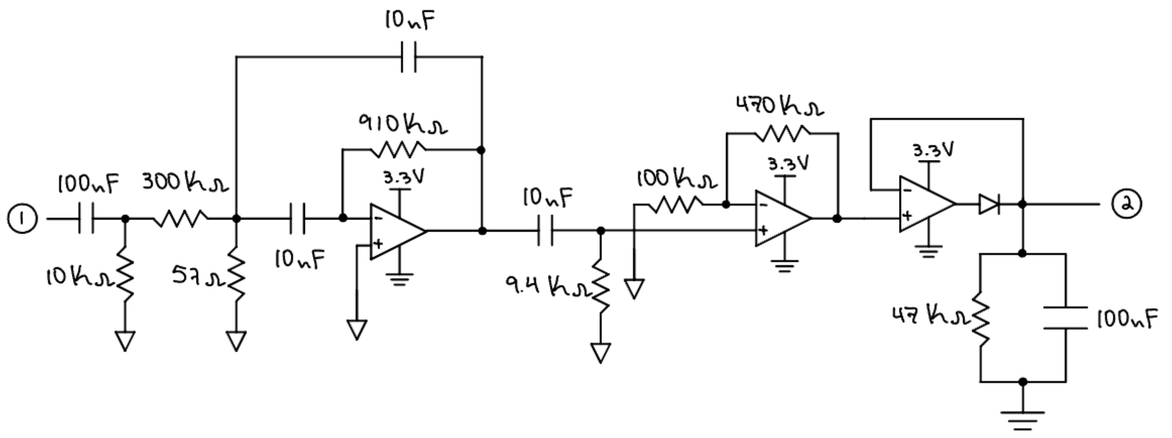

Figure 4: Active filter with AC Coupling before and after filter, one stage gain and peak detector.

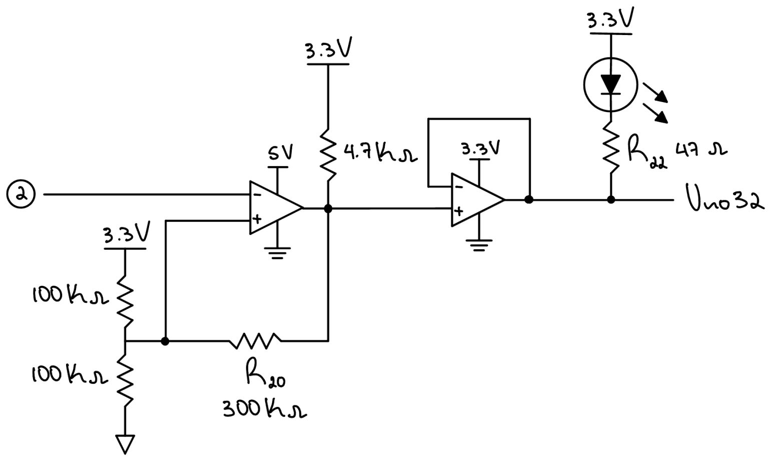

Figure 5: Comparator running at 5.0 V, buffer and LED.

Another main objective of this project was to implement a track wire detector that would detect a live wire with a 25 kHz signal. This was another important aspect because the live wire would mark the correct side of the tower at which to score. An electromagnetic field is generated when a current passes through a wire, inducing a voltage reading in the solenoid when in proximity. Pairing this inductor with a capacitor creates a LC circuit, 10 mH inductor and 3.3 nF capacitor, also known as a tank circuit. In order to increase the signal we put it through op-amps gain stages and created a virtual ground reference at 1.65 V in order to bias the output waveform at that voltage. We then used a peak detector with an RC circuit that would charge on the positive periods of the signal and discharge during the negative periods and fed that signal through a comparator which evaluated when the inductor was near the wire. For easier debugging we placed an LED at the end of the circuit to make sure it worked properly without having to read its value in software.

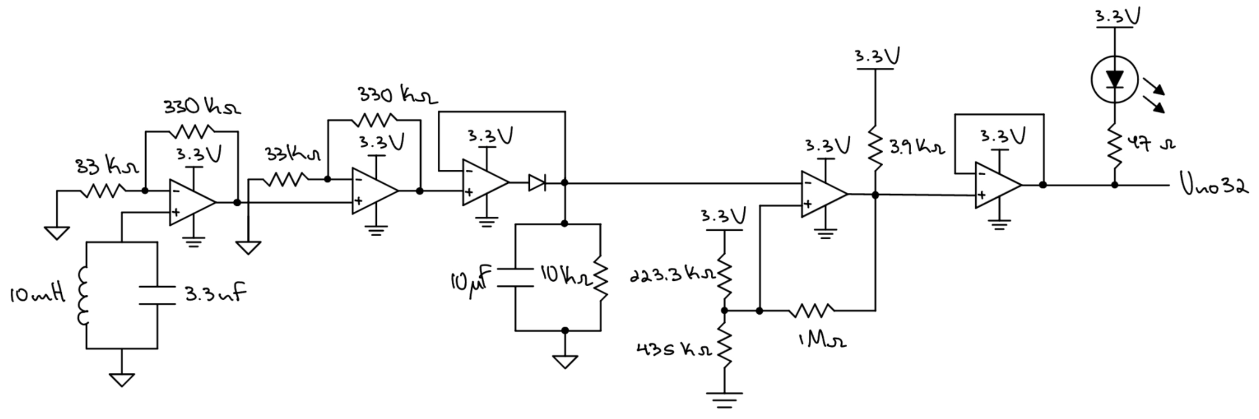

Figure 6: Track Wire Implementation

The last sensor we implemented was an IR sensor that would serve as a proximity sensor using the TCRT5000 IR sensor. We used a potentiometer for easier adjustment of when the sensor should trigger and put an LED to be able to physically tell when it triggers.

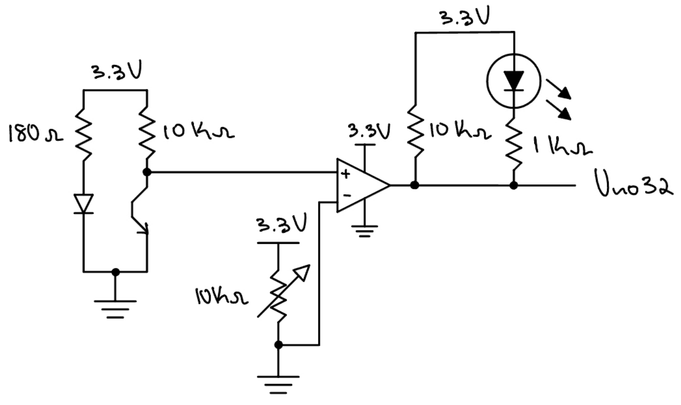

Figure 7: IR Proximity Sensor

Software

In terms of software development, our original concept of its behavior was quite simple. We imagined that it would have three top-level states: sensing, tape-handling and bumping. Sensing would handle finding the beacon by spinning the robot and heading straight until bumped. A bump event would transition the state machine from sensing to the bumping state-machine regardless of anything else. Despite our confidence in the beacon detector, we would set a timer in the event the signal was not strong enough to follow the tape for a certain time and transition back to sensing. We also imagined that the robot would simply line up with tape sensors and bumpers on the side and instantly understand if the object is a dead bot or a tower. After scoring, our robot would not bump and continue sensing for a tower by making a perfect circle around the tower. A beacon detection would transition back towards the sensing event. As simple as the concept sounds, the implementation was quite difficult. The major problems revolved around: lining up the sensor, tuning the behavior of turns and preparing for the dead bot. Our shooting mechanism required a high degree of precision in terms of both angle and distance, something that couldn’t possibly and consistently be achieved with our current design. In the following sections, the navigation system will be discussed first.

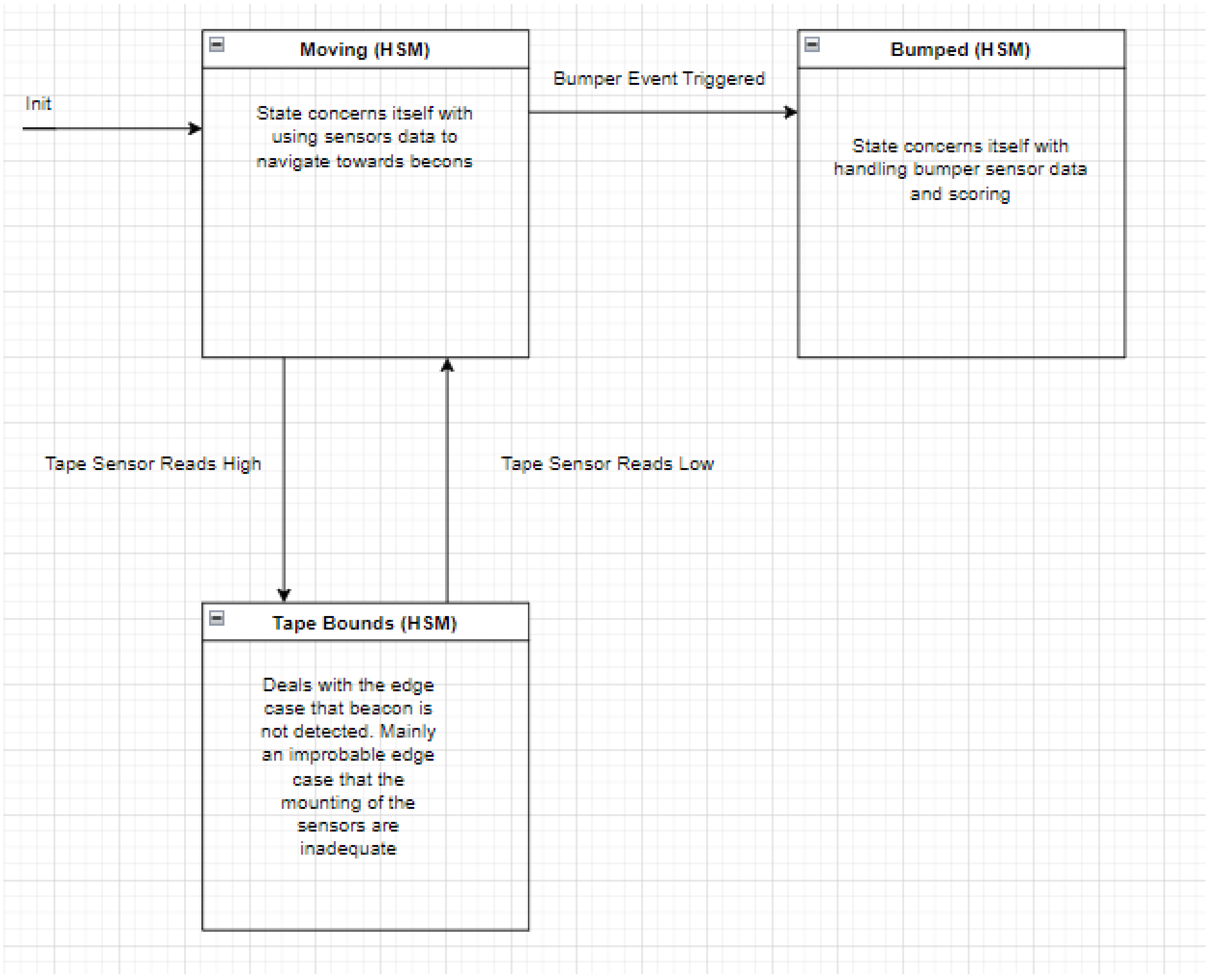
Moving (HSM)
State concerns itself with using sensors data to navigate towards becons
Bumped (HSM)
State concerns itself with handling bumper sensor data and scoring
Init
Bumper Event Triggered
Tape Sensor Reads High
Tape Sensor Reads Low
Tape Bounds (HSM)
Deals with the edge case that beacon is not detected. Mainly an improbable edge case that the mounting of the sensors are inadequate

Figure 8: Top Level Hierarchy of Robot Behavior

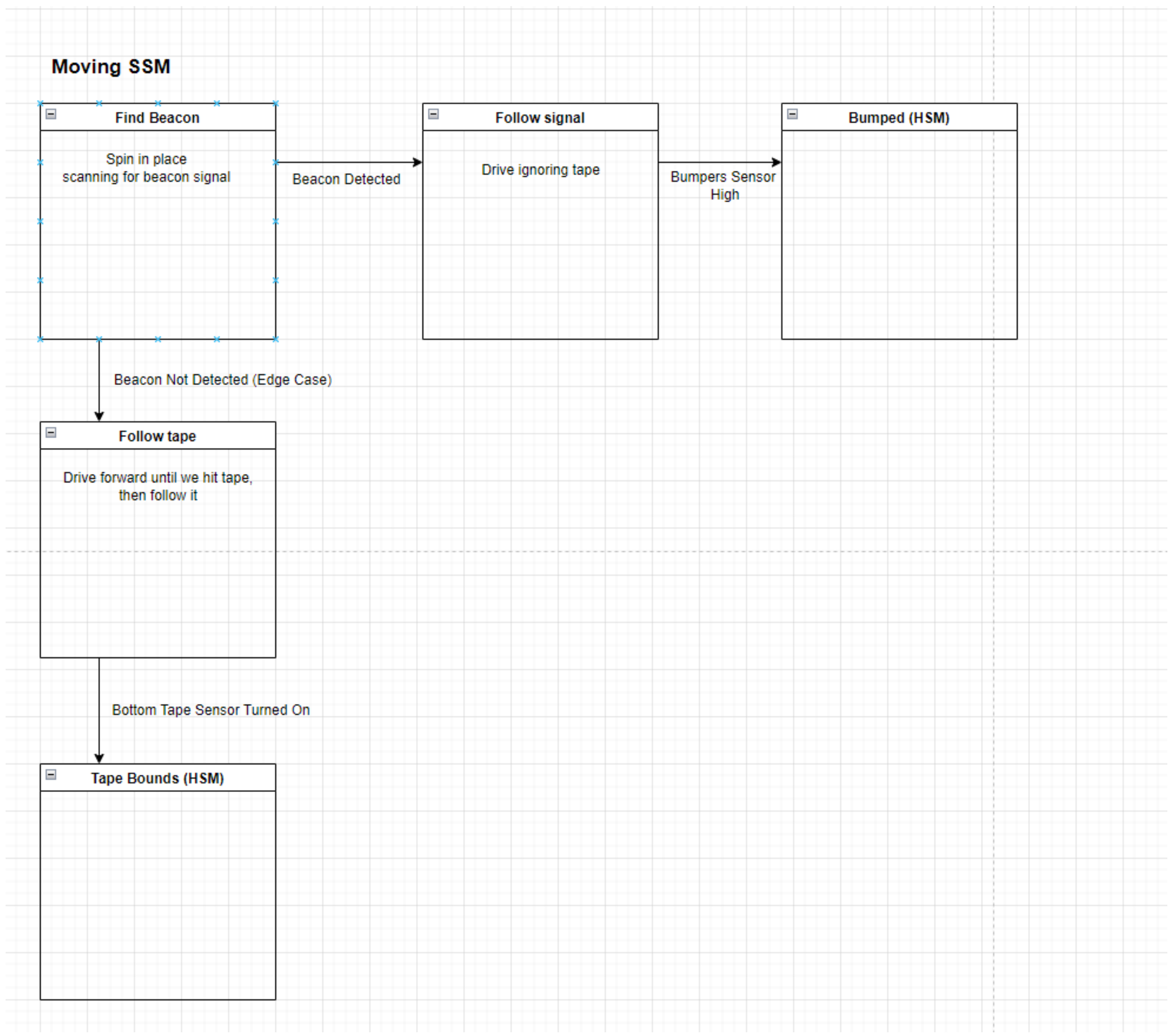

Figure 9: Moving SSM

The state machine depicted outlines the behavior of a robot during its operation. Initially, the robot is in the "Find Beacon" state, where it spins in place to scan for a beacon signal. Upon detecting the beacon, it transitions to the "Follow signal" state, driving while ignoring any tape sensors. If the beacon is not detected, which is considered an edge case, it moves to the "Follow tape" state, driving forward until it hits tape, then following it. If the robot's bumper sensor is triggered during the "Follow signal" state, it enters the "Bumped" hierarchical state machine (HSM) for further handling of the event. Additionally, when the robot is in the "Follow tape" state and the bottom tape sensor is turned on, it transitions to the "Tape Bounds" HSM for more specialized behavior related to tape navigation. This flow suggests a robot that is capable of both seeking out specific signals, such as a beacon for navigation, and following preset paths or boundaries, such as tape on the ground, while being responsive to physical interactions with its environment through its bumpers.

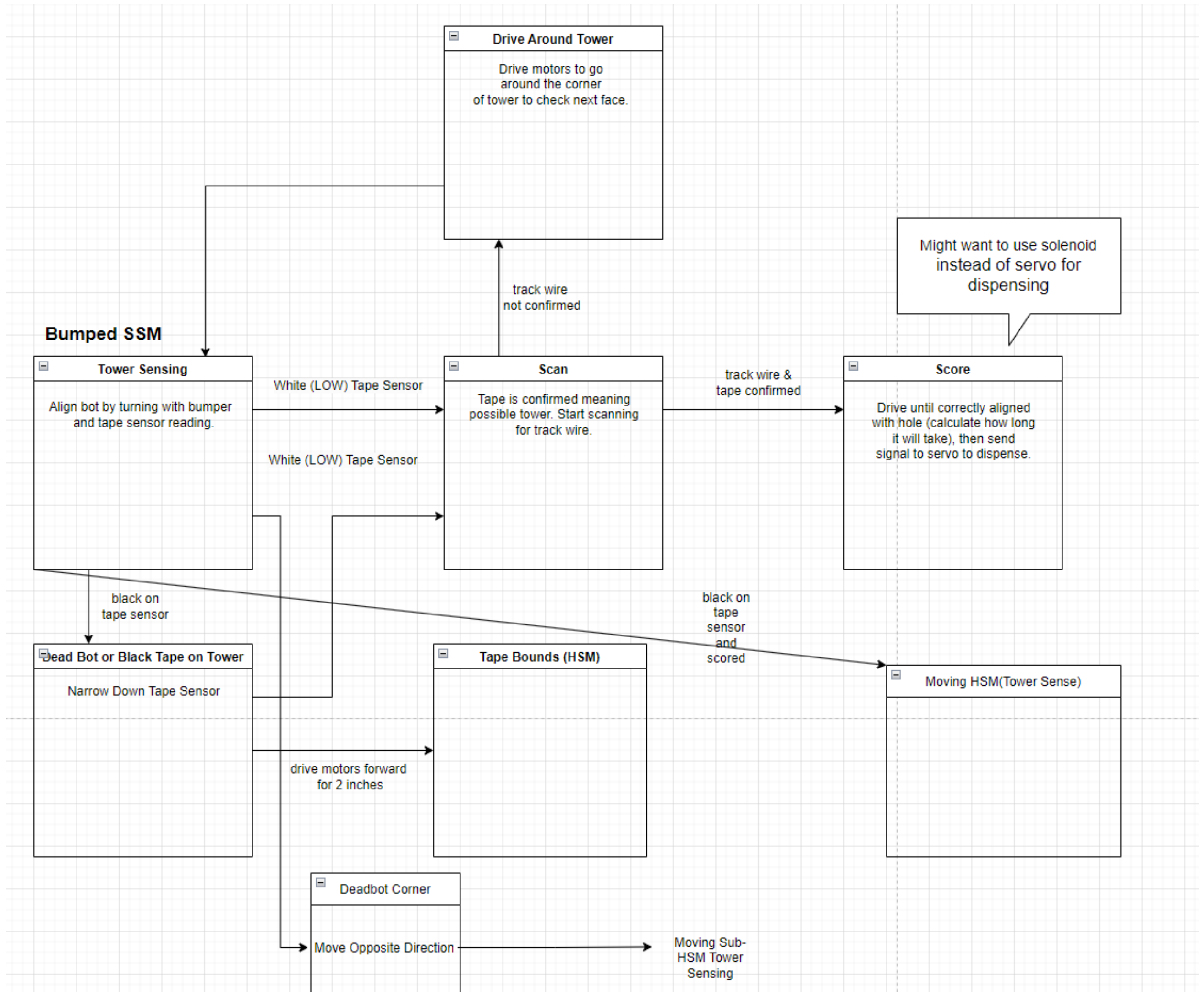
Drive Around Tower
Drive motors to go around the corner of tower to check next face.
Might want to use solenoid instead of servo for dispensing
track wire not confirmed
Bumped SSM
Tower Sensing
Align bot by turning with bumper and tape sensor reading.
White (LOW) Tape Sensor
White (LOW) Tape Sensor
Scan
Tape is confirmed meaning possible tower. Start scanning for track wire.
track wire & tape confirmed
Score
Drive until correctly aligned with hole (calculate how long it will take), then send signal to servo to dispense.
black on tape sensor
black on tape sensor and scored
Dead Bot or Black Tape on Tower
Narrow Down Tape Sensor
Tape Bounds (HSM)
Moving HSM(Tower Sense)
drive motors forward for 2 inches
Deadbot Corner
Move Opposite Direction
Moving Sub-HSM Tower Sensing

Figure 9: Bumped Lower Level SSM

As shown in Figure 9, when the robot aligns with a tower, it uses bumper feedback and tape sensor readings. If a white signal (indicating low) is detected by the tape sensor, the robot will continue to scan for a track wire to confirm the presence of a tower. If the tape confirms the presence of a tower or a track wire is detected, the robot enters the "Score" state, where it drives until it aligns with a scoring hole, then sends a signal to a servo (or potentially a solenoid, as suggested) to dispense an object.

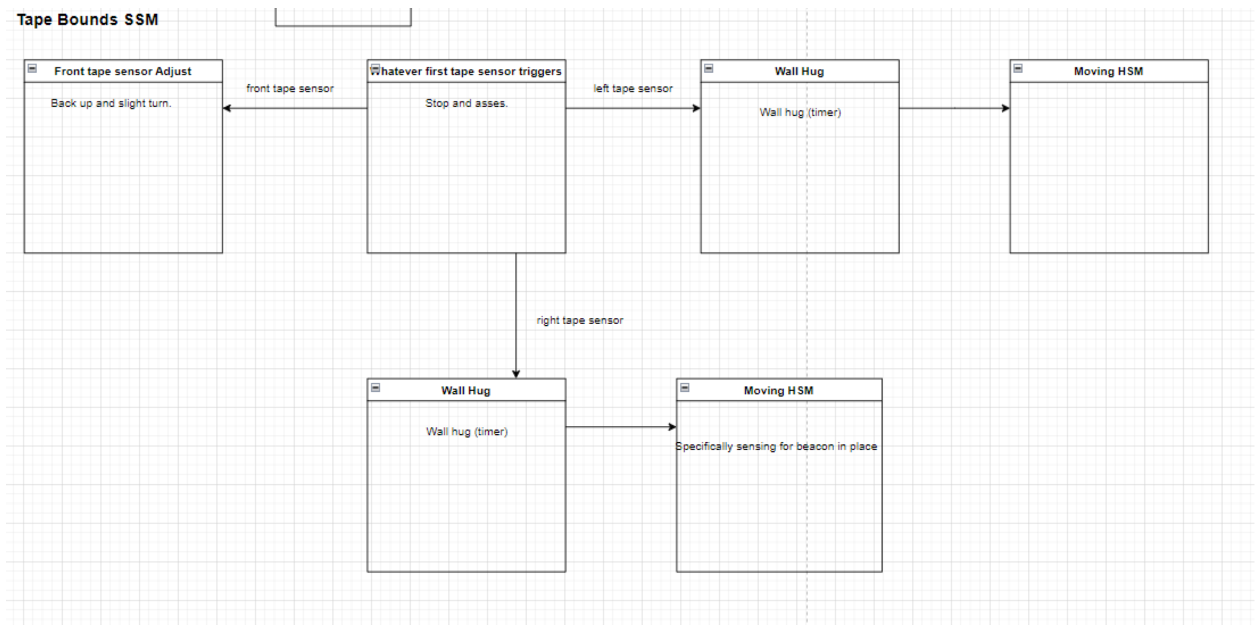

Figure 10: Tape Bounds SSM

If the tape sensor reads black, indicating either a dead bot or black tape on the tower, the robot will narrow down the tape sensor reading and drive forward two inches. If it hits a dead-end, known as a "Deadbot Corner," the robot will move in the opposite direction.

This behavior is part of a larger hierarchical state machine that also includes "Drive Around Tower" for navigating tower corners and "Tape Bounds HSM" for managing interactions with tape boundaries. The "Moving HSM" indicates a sub-state machine dedicated to tower sensing, suggesting a layered approach to navigation and interaction with the environment. In the upcoming section, problems and troubleshooting will be discussed.

Initially we had a design such that our wheels reached the maximal length limit according to the project specifications. This in turn generated quite a bit of issues when the wheel got stuck or caught and did not detect a bumper event. Even in the ideal scenario, with just bumpers we came to a harsh conclusion: it was impossible to make our robot parallel with the tower. By relying on timers and time between bump events, we could never quite account for every single angle the robot could hit. Hence, it caused our first redesign of our system: pushing the wheels/motors deeper into the frame such that it would never get caught. Then another question arose, how would we line up our tape sensors on the side and look for deadbots? To knock out two birds with one stone, it was then concluded that a tape-sensor at the top could resolve each issue. Depending on the height of the robot, the dead bot could not be sensed at all, yet in contrast the tower would be read from the tape-sensor. So, if that tape-sensor within a tuned time-frame did not pick up an event, it could declare the object as a dead bot, and move on to navigating towards a tower. In regards to staying close to the tower, we initially thought that using the analog reading of the tape-sensor could act as a pseudo-distance sensor. Unfortunately, this yielded inconsistent results even when affected by the hour of day, and proved to be too troublesome to change the threshold. A second tape sensor, at the same height, yet also on the opposite side of the wheel proved to be the best option. In doing so, we could have a state such that if one tape sensor was not reading the tower, then it would turn the right wheel towards the tower. Until both were sensed, it would not even bother to shoot. With no timers, the tape sensors at the top could slightly jerk the wheels millimeter by millimeter at the same rate the ES_Framework was processing events.

A significantly easier problem than lining up, yet also painstakingly time consuming: our turns and bumpers needed to have specific ticks in order to function properly. If a turn took too long, the dead bot timer we set for the bot would begin looking for another tower and not score. Hence, all we needed to do was play around with the timers until it worked. Some other issues include speeds. Alignment could never be perfect with the robot, and lowering the dominant wheel's speed (the wheel causing angular offset), we could have a relatively straight drive.

At times our event checkers proved to need to be different from the previous state in order to generate events. This in turn caused quite a few transitions that depended on these events to never occur. I.e if a tape-sensor is always black from sensing open space, there is no way to differentiate seeing black if our tape-sensors are immediately perked up against it. To resolve this, we could use a mini-version of our event-checkers to manually check the Analog to Digital reading at the top and force post an artificial event in that case. This allowed us a lot of leeway considering the nature of the digital output of tape-sensors to generate events and transition smoothly.

```
case Align: //stuck on this state
    if (AD_ReadADPin(AD_PORTV3) > 500) {
        SetRight_MotorSpeed(900); // aligns to side of wall
        SetLeft_MotorSpeed(0);
    } else {
        ES_Event ReturnEvent;
        ReturnEvent.EventType = CLOSED_SPACE;
        PostTopLevelHSM(ReturnEvent);
    }
```

Code Snippet: Force Event Post Example

We also could manually generate an event with no checker in-case there was no way for our system to perceive it. We tried static counters that might allow us to change behavior if something is scored, but instead we found it more efficient to transition from top-level by simply forcing an event with no other generation to it.

Our last manipulation would be that of BOARD.c, a given source file. This file had a battery threshold that ruined our bot if it was slightly lower than the maximum charge. Hence, we changed the thresholds to not care about battery level.

Conclusion

Our decision to utilize a three-layer configuration for our robot was driven by the need for a streamlined approach to shield our sensors from potential interference caused by the DC motors. During our navigation around towers, we encountered a challenge with the wheel becoming lodged at its corner. Despite repeated software adjustments proving ineffective, a redesign of the bottom layer ultimately proved to be a more effective solution. This redesign, though initially time-consuming, significantly enhanced our navigation and alignment capabilities.

The integration of wheels posed a unique challenge. The hexagonal configuration of the purchased wheels, coupled with the short shafts of our motors, necessitated the development of custom extension mounts. These mounts successfully addressed both the hexagonal wheel design and the required shaft length. However, during testing, we identified material limitations with both MDF and acrylic. The adoption of 3D printing technology resolved these issues effectively.

Alignment was a significant obstacle. The initial reliance on bumpers for wall alignment was inadequate, leading us to incorporate two additional IR tape sensors for tower detection.

This modification not only resolved the alignment issue but also enhanced our system's capability to handle edge cases, such as encountering inactive robots. The optimal placement of IR sensors for both tower and tape alignment was achieved through extensive trial and error. The tape detection issue was resolved by positioning the sensors precisely 2 inches apart and at an optimal distance from the tape. Similarly, the tower detection sensors were fine-tuned through trial and error to identify the correct settings for open space detection.

We initially faced challenges with the phototransistor in our beacon detector, which erroneously detected 2 kHz signals from other towers, even when not directly aligned with them. To address this, we experimented with various forms of shielding, including a foamcore box, additional layers of foamcore and cardboard, and ultimately, aluminum. The final design, incorporating aluminum, resolved the issue but reduced the sensor's range to approximately 4 feet. A subsequent modification, involving the construction of a small aluminum cylinder placed at the phototransistor's entrance, successfully restored the original range of the beacon detector. Another major challenge involved recurring fuse failures in the Uno Stack, traced back to the physical contact between the metal leads of the bypass capacitor on the DC motors and the motor housing. Isolating the capacitor with electrical tape and repositioning it away from the motors effectively resolved this issue, eliminating the problem of the robot unexpectedly powering off.