Team Toast Report

Nikolaas Bender, Christian Sabile, Justin Wong

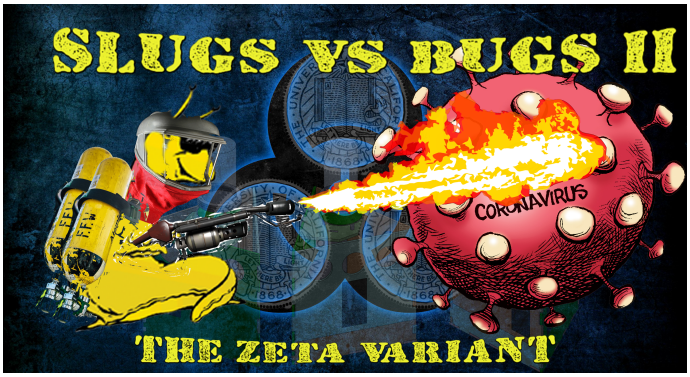
University of California, Santa Cruz

ECE118 - Professor Gabriel Elkaim

December 10th, 2021

INTRODUCTION

For the final project of ECE 118, we had to make a functioning robot that could accomplish certain tasks on a field within the span of five weeks. All of the previous labs have been building blocks to complete this task. From the circuit building to the use of the ES Framework, we had to use our experience in these labs and lessons learned from them to accomplish this quarter's game, "Slugs vs. Bugs II: The Zeta Variant."



1 GENERAL GAME INFORMATION

The general information of the game is as follows. The field is 84" x 84" or 7' x 7' with a 2" black tape perimeter within the bounds of the field and in the center of the field making a 24" x 24" "Keep-Out" square that defines our field as shown in Fig. 1. If more than half of the robot goes beyond the outer edge of the tape then you are disqualified. Each corner of the field has a 16" x 16" starting zone, and the robot starts in one of these starting zones in a random orientation. Fig. 1 shows the basic field.

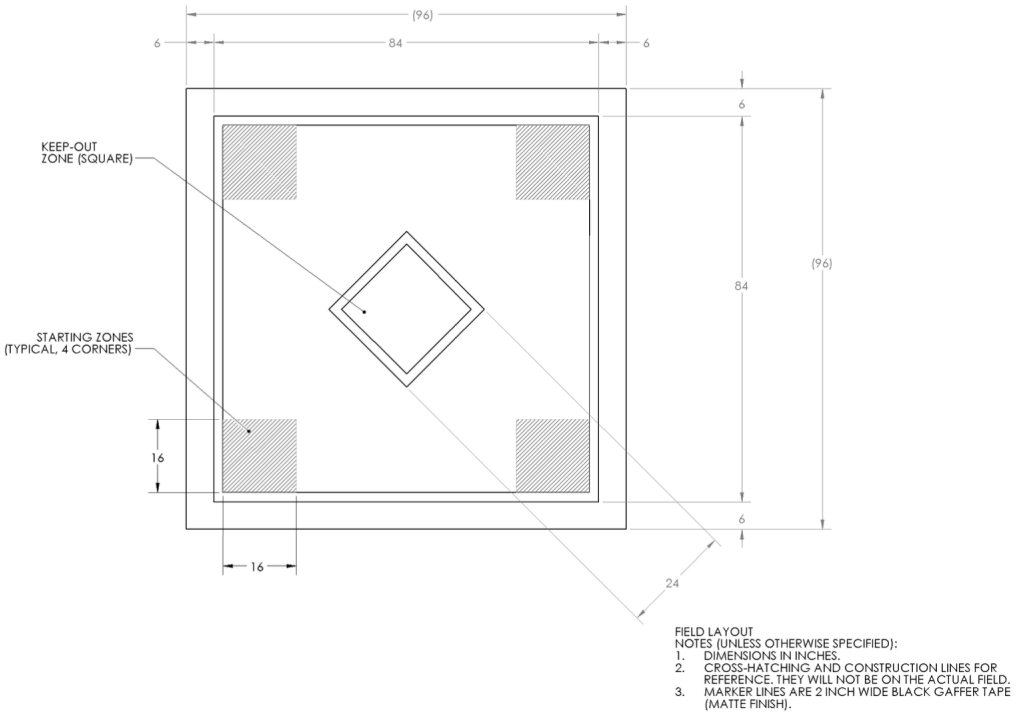


Fig. 1: Basic Top View of the Field

The field itself has three triangular towers called vats. Each vat is about 11.25" tall has a 2 kHz beacon resting on top. Each face of the vat has three holes where ping-pong balls can be deposited through. Below one of the holes is a strip of black tape which labels the correct hole for that particular face. However, inside the vat is a 24-26 kHz track wire which reveals the correct face of the vat. Fig. 2 shows the CAD drawing of the vat.

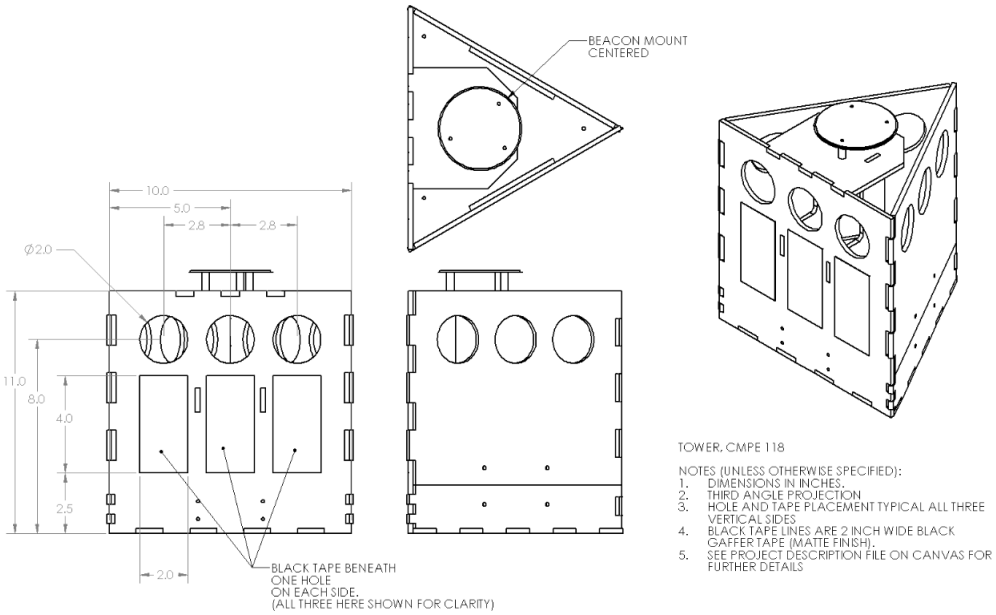


Fig. 2: Basic Dimensions of the Vat

In order to solve each vat correctly, the robot has to find the vat via the beacon, locate the correct face of the vat from the track wire, and finally locate the correct hole with the black tape. For minimum specifications, each robot must find two of the three vats and solve each one correctly two out of three matches performed on the field. Each match is two minutes long and for minimum specifications one team will be operating on the field while a weighted black box will be used to represent a dead robot. Each match has the field set in a random orientation which decides vat placement/orientation as well as where the team will start and where the dead robot will be placed. While the general information of the game is simple, the overall implementation are several hurdles to overcome.

2 Mechanical Design

Initially we had to come up with a few designs to get started for the overall shape and design of the robot. The preliminary design we went with was a box shaped robot. For the initial design we were thinking of making it a rectangular box that would house everything inside. We had initially thought of the box being an open bottom with cross bracing to mount the sensors. Just towards the back of the robot, we would have a continuous servo spinning a ship-like wheel that would hold and deliver ping-pong balls to a downwards ramp into the hole. The initial sensor package would use an IMU, two ultrasonic sensors, two track wire detector circuits, a beacon detector, and seven tape sensors meant for both detecting the tape on the ground and on the face of the vat. The overall design looked like a toaster, hence why our team name is Team Toast. The following figures show the initial ideas we stumbled upon during brain storming.

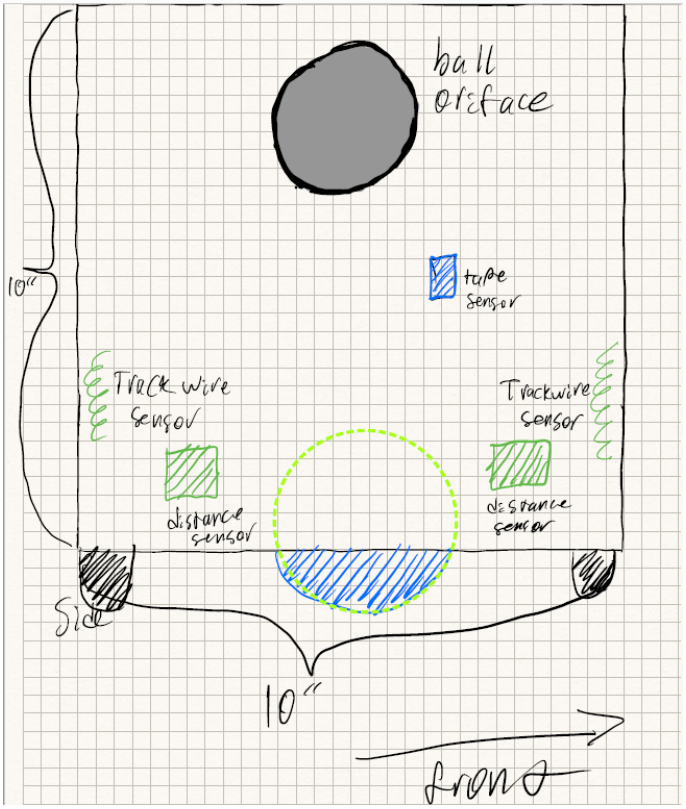


Fig. 3: Initial Side Design of the Robot

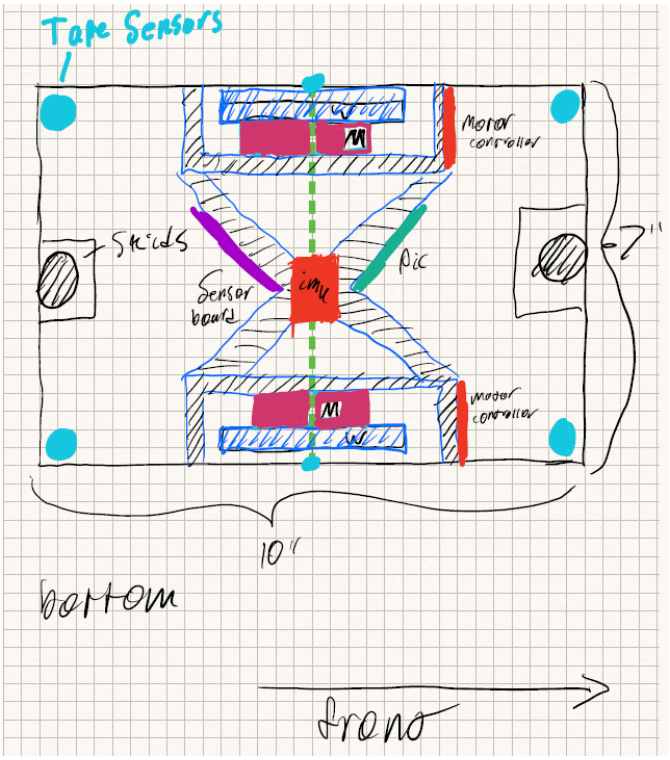


Fig. 4: Initial Bottom Design of the Robot

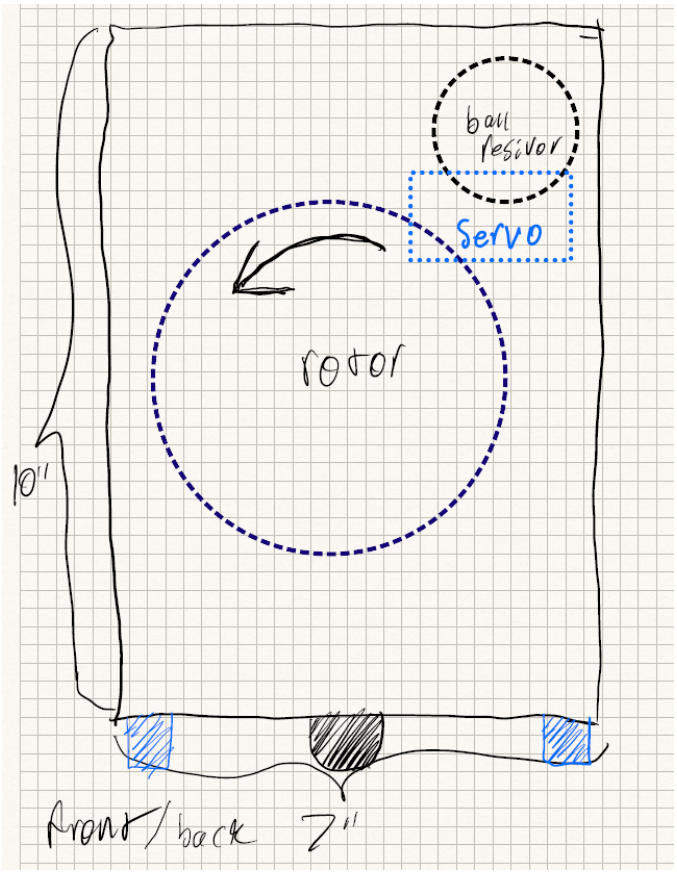


Fig. 5: Basic Dimensions of the Ball Delivery System

As time went on, we had changed from using ultrasonic sensors to start using three lidar sensors connected to a small micro-controller. The secondary micro-controller would then communicate to the main uno32 micro-controller over UART to send distance measurements which would help is getting our robot parallel to a face of a vat. The ball delivery is still the same however it rests on top of the robot and a ramp is deployed using another servo to aid in ball delivery. We ended up ditching the IMU and instead started using bumper switches for an easier implementation to determine where something hit the robot. The final build is shown in the following figures.

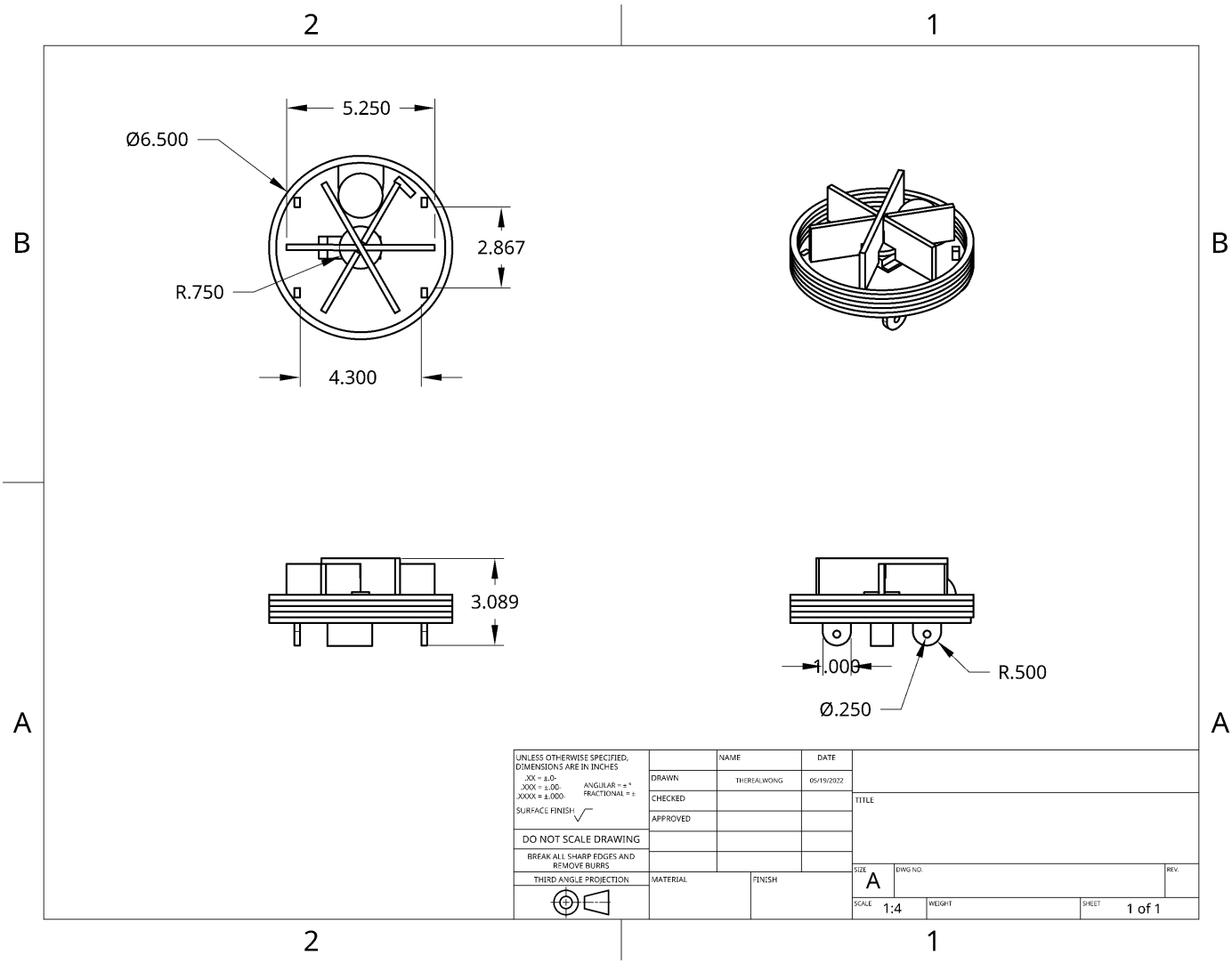


Fig. 6: Ball Loader CAD drawing

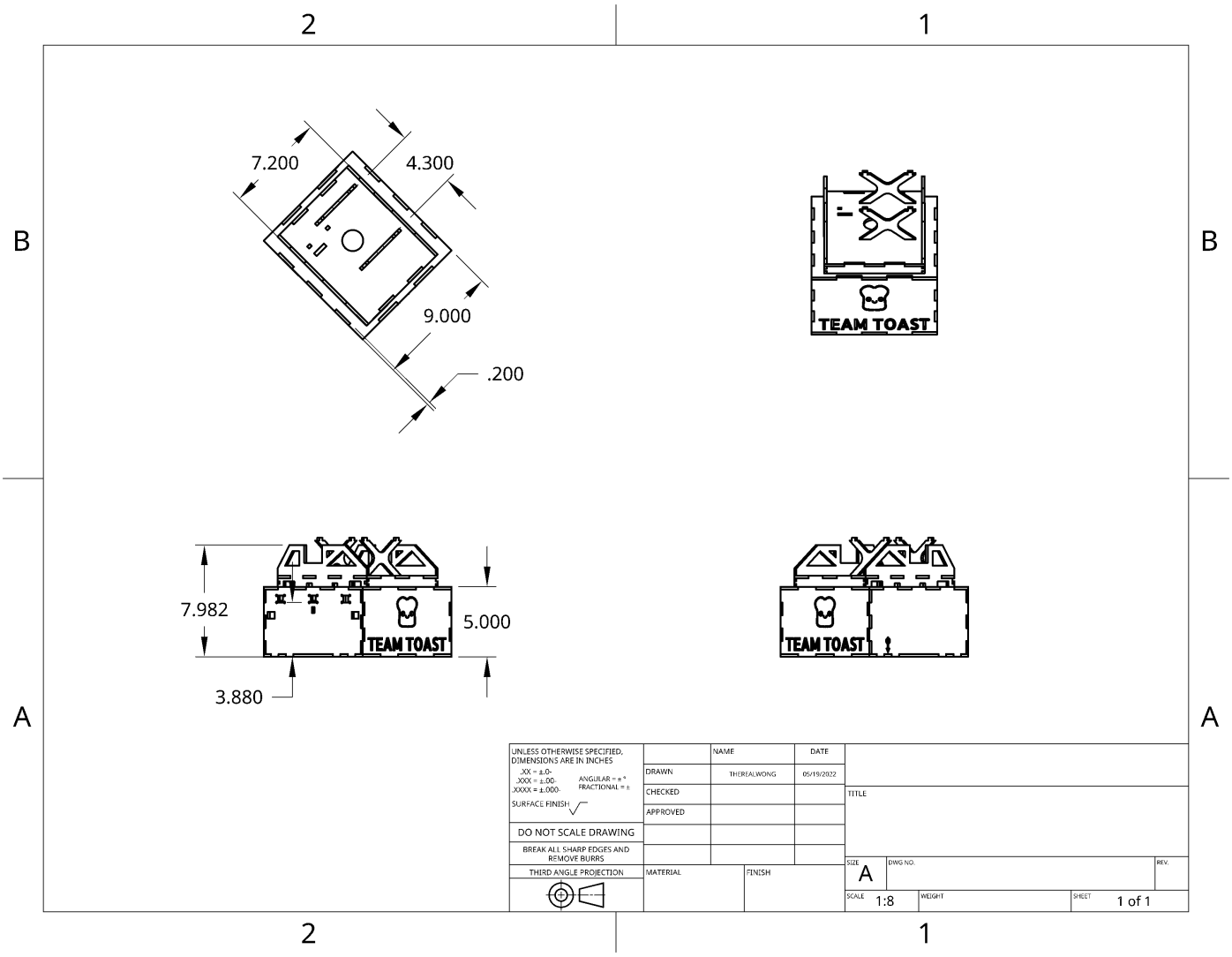


Fig. 7: Full Robot CAD Assembly

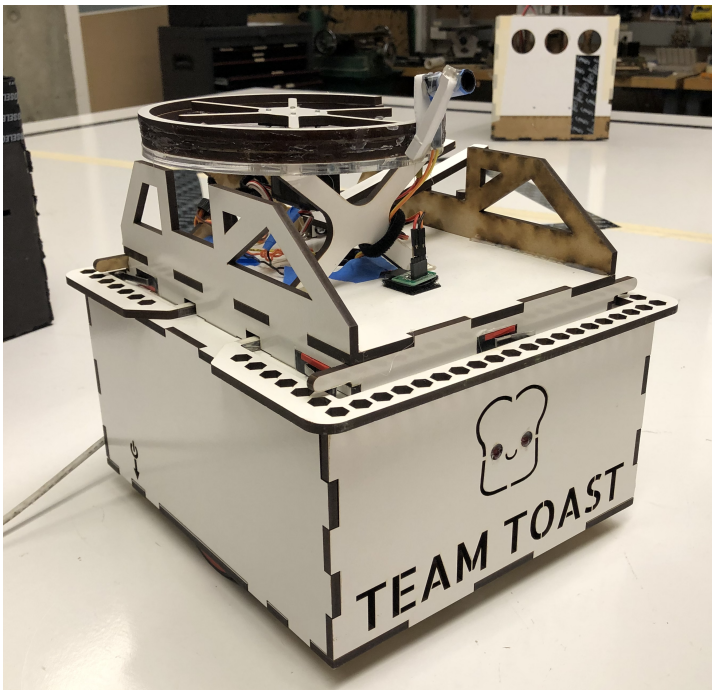


Fig. 8: Front of the Robot

Fig. 9: Back of the Robot

While most of the initial design of the robot had changed drastically, it still looked like a toaster so we decided to stick with our original name of team toast.

3 Electronics and Hardware Design

For the overall electronics we had decided to use the following for our sensor package:

- For the general field we had:
  - 4 tape sensors for the floor
  - 6 bumpers for impact detection
  - 1 hall effect sensor meant to deliver balls
  - 2 servos for the ball delivery system
  - 1 Teensy LC micro-controller
  - 2 Motors
- For the Vat we had used:
  - 3 lidar sensors
  - 2 track wire sensors for finding the correct face
  - 1 beacon detector
  - 1 tape sensor meant to detect where the correct hole is

The following figure shows a simplified diagram of the overall hardware being connected to the uc32.

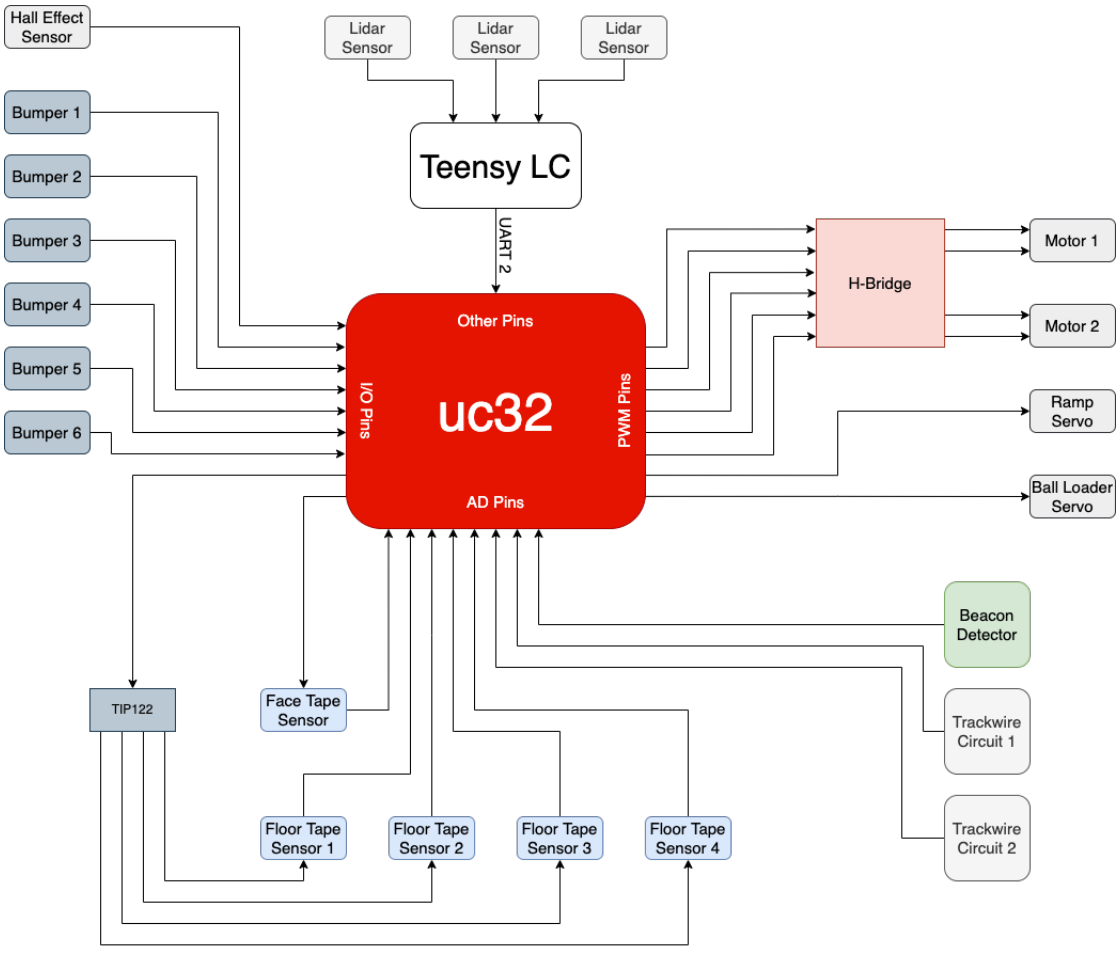


Fig. 10: Simplified Hardware Diagram

Nikolaas provided the track wires from his lab with a few alterations, which the schematic is shown in Fig. 9. Christian did the tape sensor circuits and provided the beacon detector, the schematics are shown in Fig. 10-14. For deploying balls into the vat as well as handling bumpers, Justin had to make a daughter board for all the appropriate connections from the uno32 which is shown in Fig. 15.

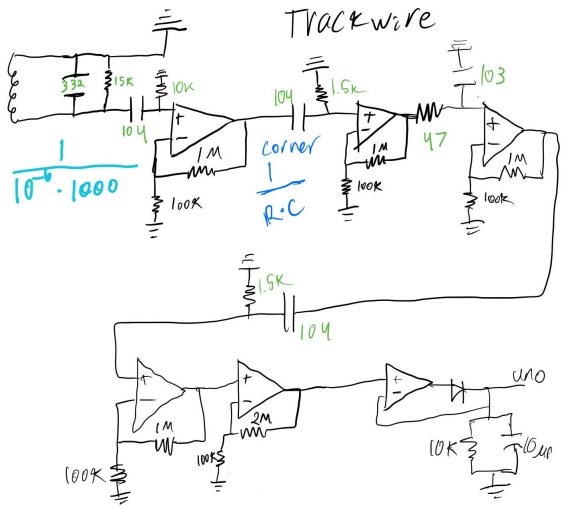


Fig. 11: Trackwire Circuit Schematic

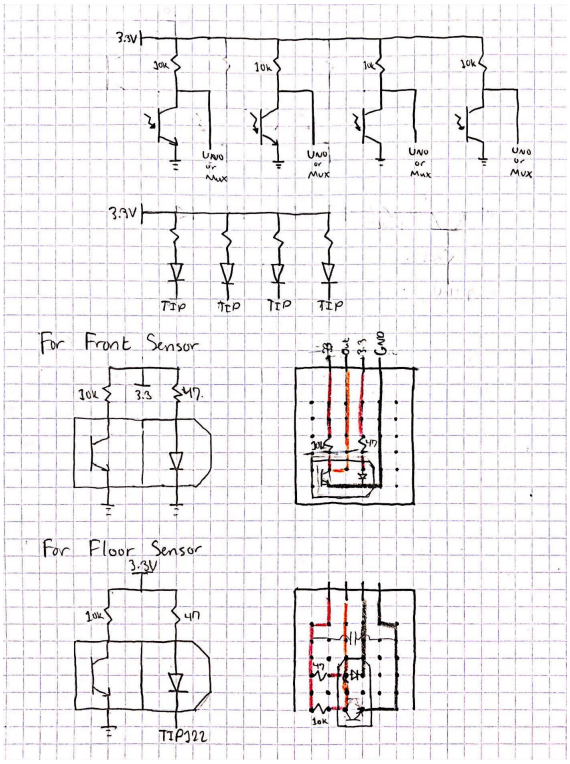


Fig. 12: Tape Sensor Circuit Schematics for Theoretical (Top) and for Individual Sensors for the Floor and Face (Bottom)

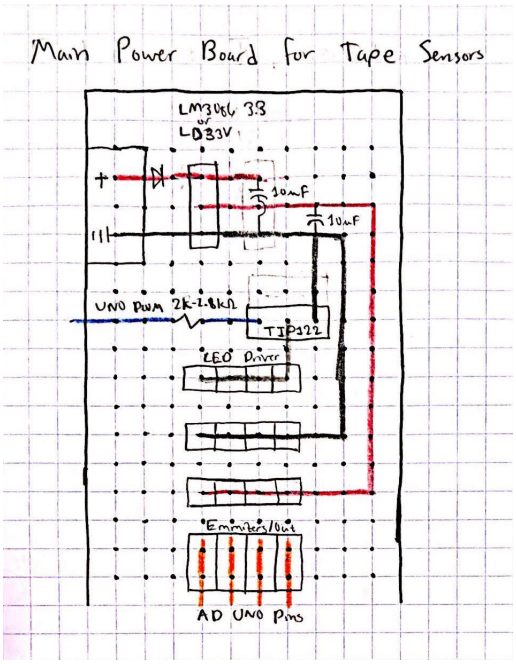


Fig. 13: Perfboard Diagram of Main Power Board For Floor Tape Sensors

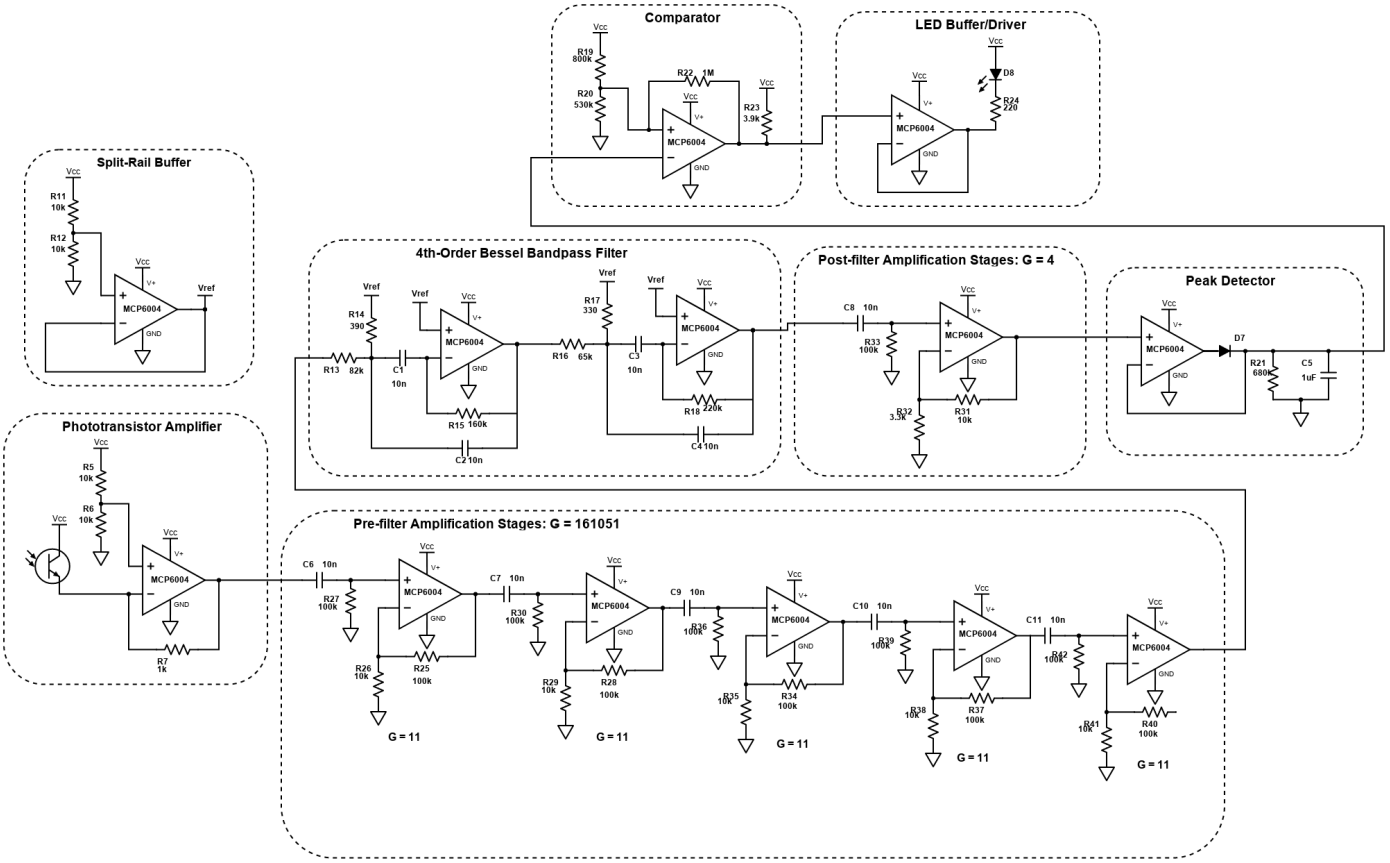


Fig. 14: Beacon Detector Schematic

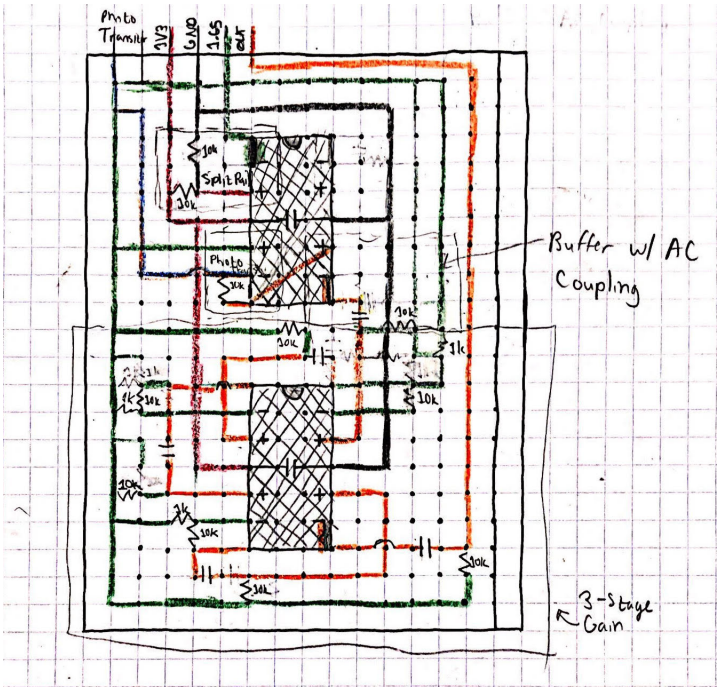


Fig. 15: Perfboard Diagram of the First Part of the Beacon Detector

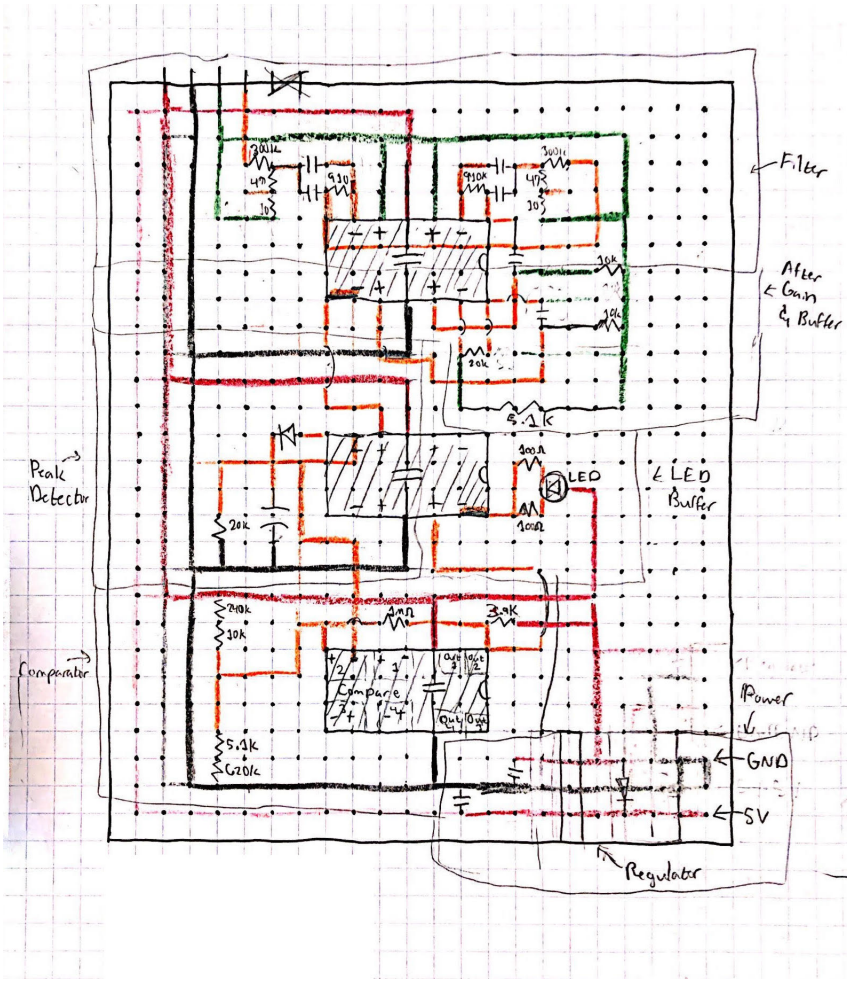


Fig. 16: Perfboard Diagram of the Second Part of the Beacon Detector

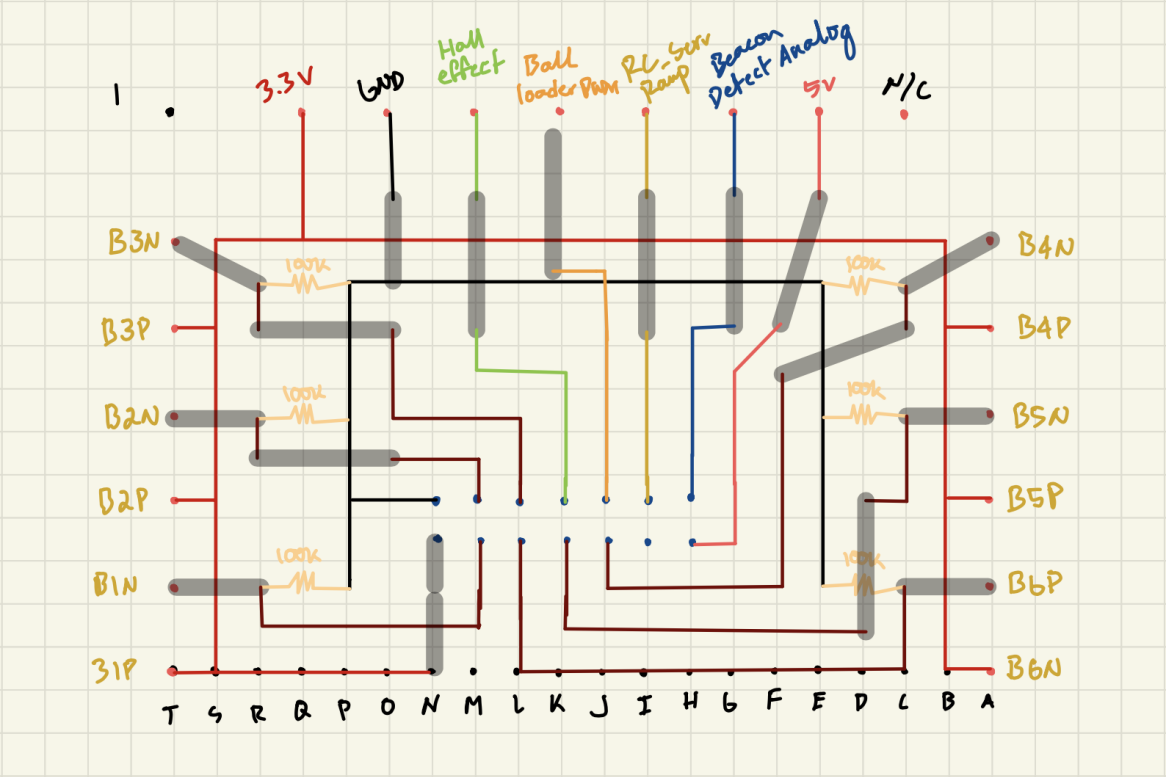


Fig. 17: Perfboard Diagram of the Daughter Board for our Bumpers and Ball Delivery System

As to why we chose two trackwire circuits to begin with was because the correct face of the vat has trackwire around the frame of the face. If we use two circuits, we could figure out what the correct face of the vat is by only visiting one face, cutting down on solving time. As for using four tape sensors for the floor, we weren't going to use a tape following method and instead avoided tape at all costs when finding a beacon. For the ball deployment we had decided to use 2 servos because it was robust and the minimum number necessary for our ball deployment design requirements. While a shooter would have been cool or a solenoid to push balls into the vat, we wanted to use a design that could be easier to implement, reliable on delivering balls, and would be robust to impact with the vat tower. Overall, the mechanical design works really well despite small hiccups and changes that were needed, most of the fine tuning for our robot was done solely in software.

4 Software Design

Software was the largest hurdle for us to overcome. The software started on whiteboards and went through many iterations before landing on our robot. The main software challenge we faced was one of theory vs practice. Our software solutions encountered issues upon implementation due to the relatively stochastic nature of reality. We added in watchdog timers and magic numbers to tune our software to the issues we faced. At the start of the project we used an amusing naming convention for our sub systems, naming them after types of bread. The system that dealt with UART communication was named $coissant_serial$. This naming convention was later dropped due to communication difficulties.

The first major hurdle we had to overcome was learning how to use UART to communicate data between the teensy micro controller and the UNO. This proved to be far more difficult than we originally thought.

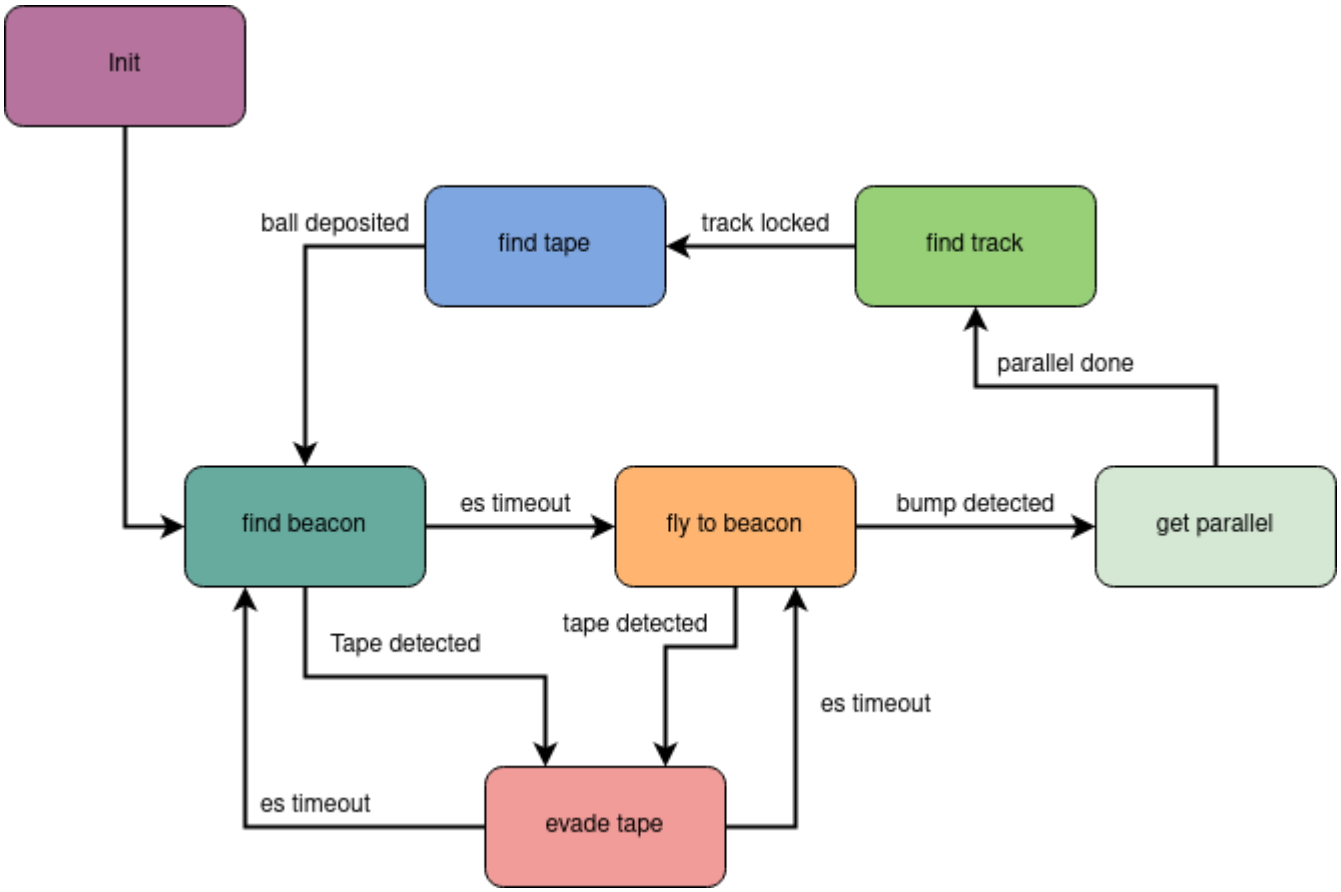
```
// clear the buffer and turn to es_events
ES_Event EatCroissants(void){
    ES_Event E;
    E.EventType = ES_NO_EVENT; // Default assume no event
    // package for the es framework
    if (IsReceiveFullC()){
        int idx = 0;
        int FULL_FLAG = 0;
        uint32_t dists = 0;
        idx = 0;
        while(idx < CROISSANT_QUEUESIZE-1){
            uint8_t val = ReceiveBuffer[idx];
            //     fore        mid         aft
            if((val == 0 || val == 1 || val == 2) && FULL_FLAG < 3){
                // correctly encode bytes for param
                shifter(val, ReceiveBuffer[idx + 1]);
                idx++;
                FULL_FLAG++;
            }
            idx++;
        }
        ReceiveTail = ReceiveHead;
        dists = TRUE; // Set distance packet
        E.EventType = DISTANCE_MEAS;
        E.EventParam = dists; // param is only 16 bits large
        if (UART_EN){
            PostToastHSM(E);
            printf("croissant publishing\r\n");
        }
        // THIS IS IMPORTANT AND OPENS THE BUFFER BACK UP FOR RECEIVING
        Receiving = TRUE;
        }

    return E;
}
```

The lesson we learned from this system was one of getting fresh data. Our final design for UART communication was based on hardware requests for data. The UNO pulls a pin high and the teensy reads that pin as high to enable writing its buffer to UART. What was convenient was that the teensy had several hardware serial ports that supported different communication protocols. The second UART on our teensy could run along side of the usb serial port for live monitoring of the teensy from a connected computer.

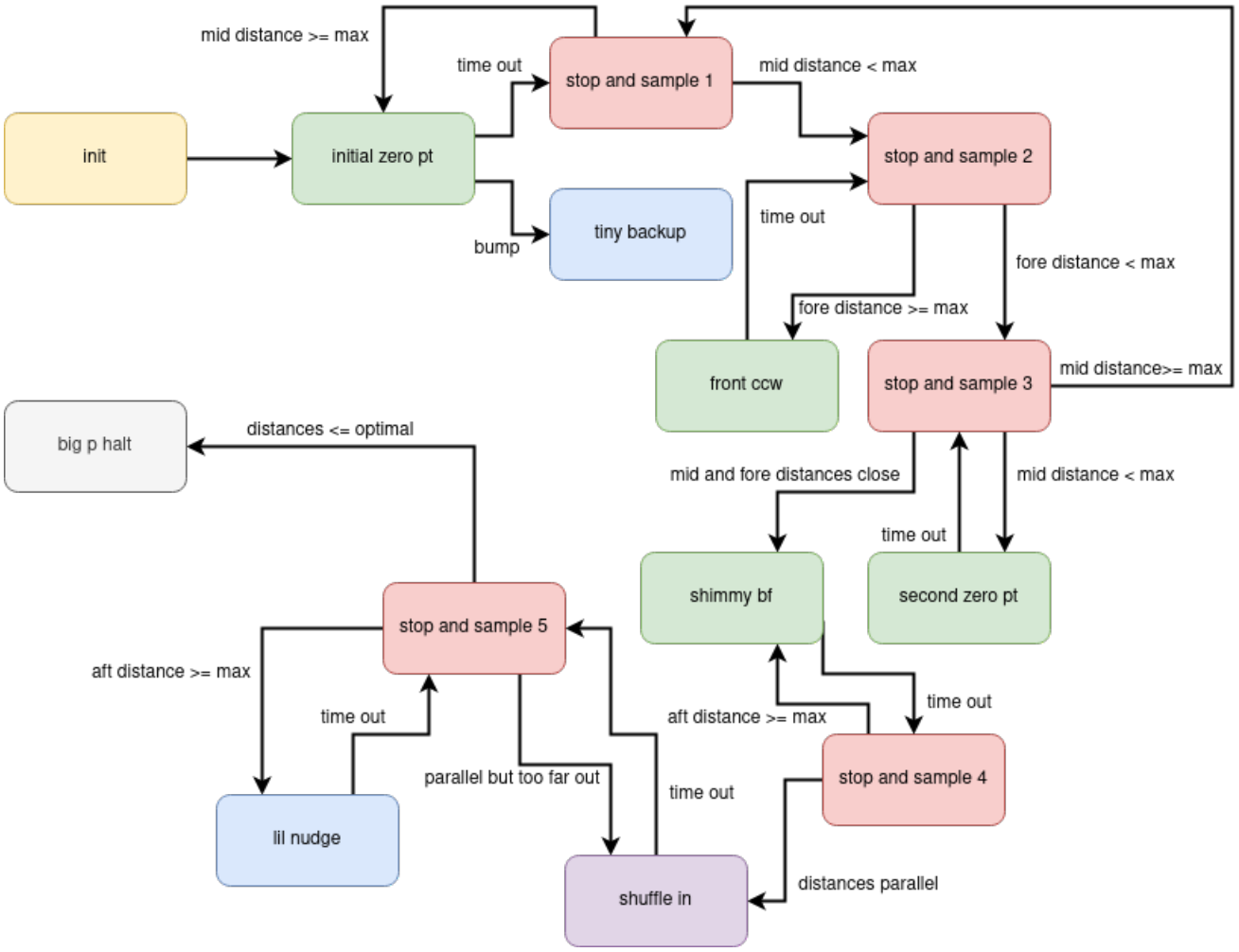
A similar technique was used for i2c on the teensy as well. i2c works on a bus and requires devices to be assigned different addresses on the bus. A difficulty for us was reprogramming the addresses of our identical lidar sensors on startup so that each sensor could be told apart. This required us to run extra jumpers to each lidar modules so that we could put each module into a programming mode. This setting programming mode pulls the XSHUT pin low. This shuts down the sensor and allows us to reprogram it with a new address. This happens to all of the lidars. These extra controls allow for other interesting properties such as synchronous polling and

asynchronous polling. This data is the collected into a two byte buffer where it is then sent out over UART for the UNO to use.

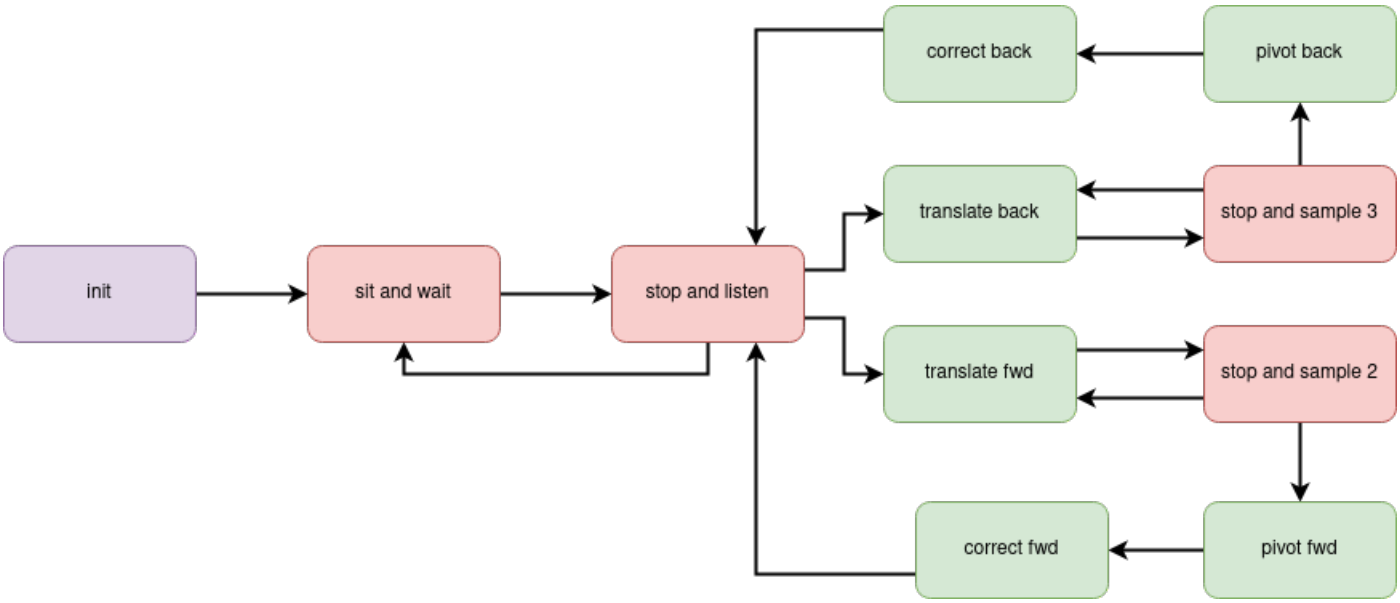
Our state machine has seen a number of revisions. our final design breaks up the stages of solving the vat towers into its primary components. The states are find, follow, get parallel, and find track wire. The names are fairly self explanatory but the most difficult of the states was get parallel. Due to the mechanical design of our robot we needed to backup and execute a sequence of maneuvers to shuffle the robot into the vat. We ran into algorithmic issues and priority issues during the development of this state as we had to decide if getting parallel or getting all of the sensors on a single face of the vat tower was more important.



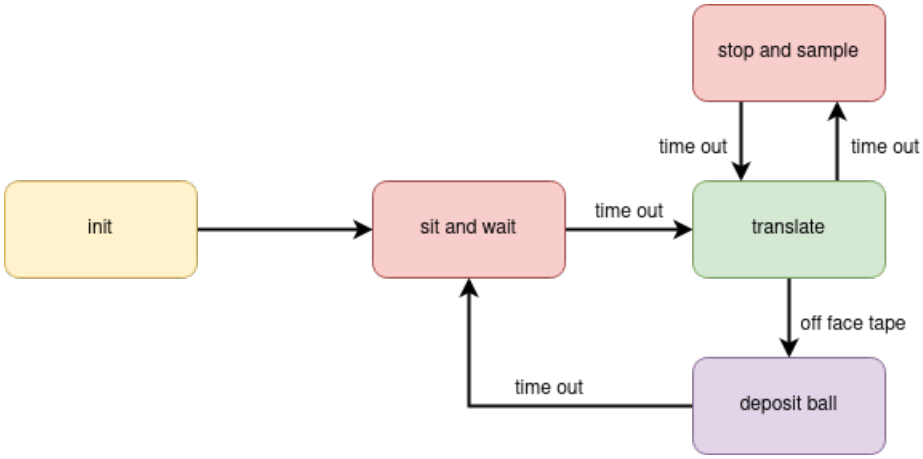
Get parallel went through six or seven revisions before we landed on the version we are currently using. This get parallel system still has some issues but works for most cases and allows us to move on to other pieces of our software. This was one of the bigger lessons we had to learn was that its a better use of our time to get something working than to get something perfect. We ended up spending so much time trying to make our get parallel state perfect and capable of solving every case. This ended up wasting so much of our time and setting us back on the rest of the software development. One of our notable iteration was dubbed “BRUCE” because like the hulk it was strong and smart trapped in an incompetent body. This iteration taught us that more dedicated sample states was vastly superior to one sample state with a number of locks to control the flow of logic. This logical flow control was critical for our success with so many and such advanced sensors.



Listening to track wires is something we decided to add more sensors to solve. We used a sensor at the leading edge and trailing edge of our robot. The reason we went for this design was so that we could immediately tell where the track wire is by only visiting one face initially. This idea was great on paper but in practice has presented some issues. The main issue is that the track wires were set more randomly on the walls than we originally anticipated. This lead to dead spots and sensors that were not sensitive enough to pickup the weaker than anticipated signals. To solve this we used our traversal state machine to search every face for a track wire quickly.



Find tape is the most simple of our state machines but shows a number of our lessons learned from all previous state machines. All that this state machine is tasked with is finding the tape that indicates the hole in which to insert a ball. We used the stop and sample state to slow down our translation and synchronise our lidar sensors to the motion of the robot. This constant switching of states seems clunky but allows us to better split up and encapsulate the relevant logic.



Conclusion

This project was a massive learning experience for us. We pushed the limits of what we knew about state machines and wiring to the limits. The real knowledge gain was in project management though. We all learned about time management and team work to integrate and build systems faster. We learned when to cut an effort off and we gained intuition into what works and what doesn't before a solution is even attempted. As an example we spent a week trying to build this beautiful ball deployment system that lifted our ball dispensing system high above the robot. It wasted so much of our time and we now have the knowledge to look for the most simple design while rapidly building anything. These simplifications are not short cuts, they are efficiencies that we can leverage to have the maximum impact in the shortest period of time.