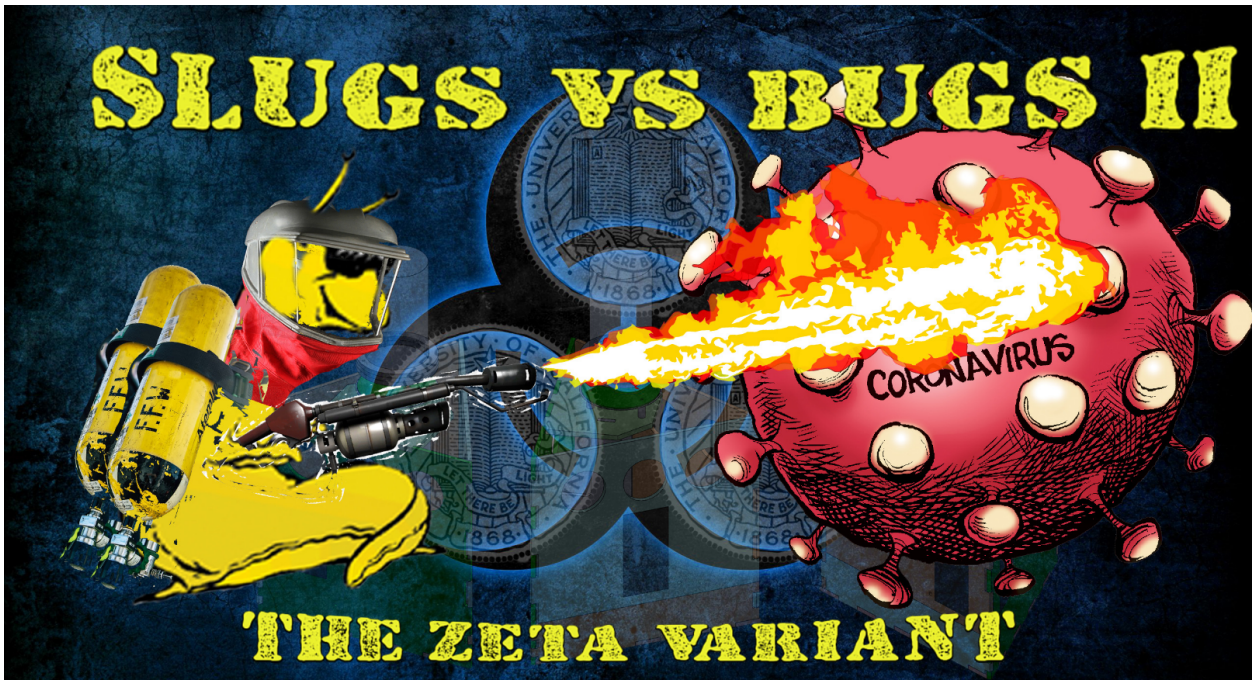
ECE 118 Mechatronics

Slugs vs. Bugs: The Zeta Variant



Tiffany-Ellen Vo, Atharva Ranadive, and Riley Altman

Introduction and Project Overview

The purpose of this final project is to apply all the material learned in past labs this quarter in order to build an autonomous robot. The topics for the past labs include hierarchical state machines, circuit filtering design, CAD, soldering, and motor control.

The theme of this year's final project is Slugs vs. Bugs: The Zeta Variant. Students will build an autonomous robot that can navigate the game field that is 11ft by 11ft and bound by black tape. The robot has to find vat towers using a 2KHz beacon atop the tower. The robot then has to find the correct face of the tower, which has a 25KHz track wire alongs it's perimeter, and shoot a ping pong ball in the correct hole of the vat tower. The correct hole is determined by the presence of black tape underneath the hole. Completing this task will result in positive points for the team. If the ping pong ball is shot on the incorrect face and/or hole, it will result in negative points for the team.

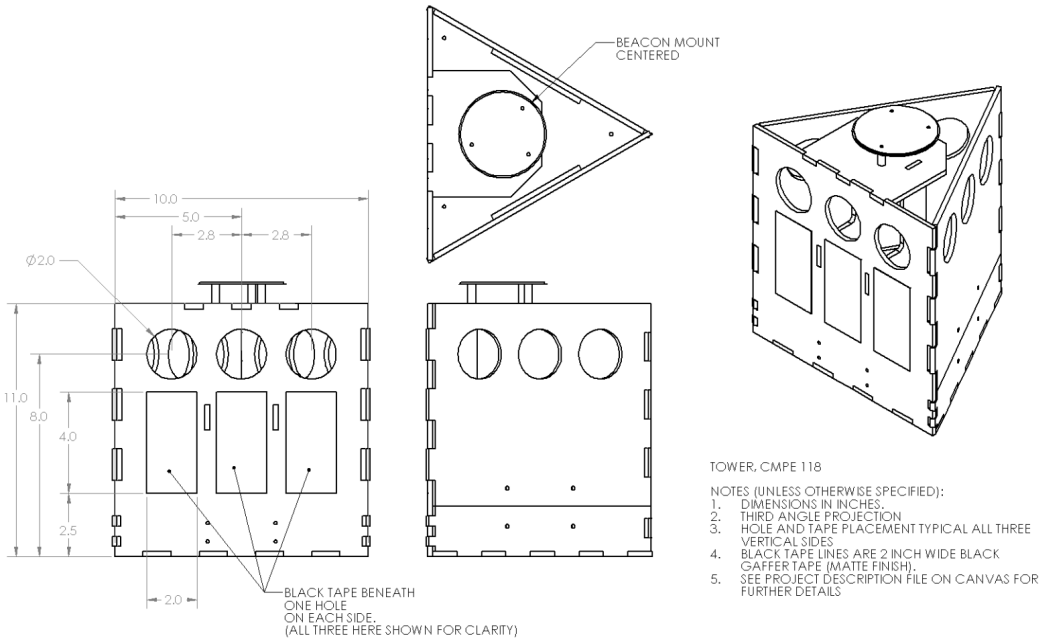


Figure: Vat Tower Model

Design Overview

To complete the task, the robot found the closest tape from its starting square and orientated itself to be directly centered on it. It then straddled the external tape, moving clockwise around the field. The robot was equipped with a beacon detector pointing to its right, thus pointing towards the center of the field. If a beacon, and thus a tower, was successfully detected, the robot would move directly towards the signal. It then adjusted itself until it was parallel to one side of the tower, and checked for the correct side using the track wire detector. If unsuccessful, the moved circumnavigated the tower until it found the side with the track wire. It then searched this side for the tape, orientated itself to be aligned with the hole and dispensed a ball. After successfully depositing a ball, it turned away from the tower and headed straight towards the tape again, and restarted the process. The aforementioned strategy only works in an ideal environment, thus several techniques were used to overcome the many edge cases. For example, while following the outside tape, if any obstacles are encountered, the robot wall hugs (follows the edge of the

obstacle itself) around the obstacle until it detects a tape and begins tape following. The track wire detector is also turned on during this time, in the case that the obstacle is a tower itself. Another example is after successful completion of a tower, the beacon detector is turned off for a certain duration to avoid detection of the same tower multiple times. The multitude of other edge case handling strategies are explained in their respective areas.

Software Components

The project was coded in embedded C using the MPLABX environment. The microcontroller used was the PIC32MX320F128H, created by Microchip Technologies. The microcontroller had a custom made (by the teaching staff) IO Shield, that allowed access to the GPIO pins. The behaviour of the robot was created using a Hierarchical State Machine (HSM), i.e. a state machine within which each state itself is a Finite State Machine (FSM).

Embedded System

To handle the event driven environment required for the HSM, the Events and Services Framework was used. This was provided by the teaching staff, and allowed for atomic event generation using student programmed functions. On a higher level, the framework behaved as a 'while' loop, calling the appropriate functions at the appropriate times, creating a pseudo-ROS system with very little overhead. The framework split the event generation into two different types, namely Event Checkers and Services. Event Checkers were executed every loop, and were used in cases where the status of certain pins required constant polling. One such example were the bumper switches used, where the status of the specific GPIO pins had to be checked at every opportunity. If an event had occurred, i.e. a detectable change as ascertained by the programmer, the event was posted to the queue of a service. The Services were only executed when an event was currently in their queue. This event could be generated by the Event Checkers, as previously described, or could be a timer expiring. This specific functionality was used to run services at certain frequencies, allowing for very consistent polling. One example of this was the beacon detector, where the value of the pin was only tested every 100ms, to avoid overloading the Analog to Digital Convertor system, and to increase noise resistance. The HSMs and FSMs themselves were coded as services, allowing events to induce changes in the state machines and preventing the execution of the entire state machine every cycle.

Services

Other than the HSM and FSMs, multiple services were used to generate the required events. For the Tape Sensors, Beacon Detector, and Track Wire Sensors, the services were necessary as these sensors were only polled periodically, to reduce the spamming of events and providing inherent debouncing. The Bumpers were slightly more complicated as they used a multiplexor, and had explicit debouncing, but required periodic sampling. The dispenser was implemented as a service, as it required holding the servo in different positions for various amounts of time.

Tape

The robot had 8 tape sensors, 6 of which were placed underneath the bot, while the other 2 were used as tower tape detectors, placed on the dispenser. Due to the nature of the perfboard implementation, one GPIO pin controls power to all tape sensors, while 8 separate ADC pins are used to check the output from each tape sensor. In the current implementation, all tape sensors are always on, but not all generate events constantly.

Every execution, each pin is polled and its value tested against the hysteresis. In this case, if the value is greater than 450, a TAPE_ON event is generated, or if it's under 200, a TAPE_OFF event is generated. For each tape sensor, if this event does not match the previously recorded value, the current event, with information about which tape it is, is posted to the HSM.

Beacon Detector

The Beacon Detector had two modes, Moving Sample and Stopped Sample, however both poll the beacon at 100ms. The hysteresis used here was 100 and 200. The Moving Sample works similar to the tape sensor, wrt to the hysteresis. It returns a BEACON_DETECTED if the ADC value is greater than 200. The Stopped Sample assumes that the robot is currently not moving, and takes 10 samples without returning an event. It then averages all of these samples, and the average is tested against the hysteresis. The event detected here is returned as BEACON_CONFIRMED or BEACON_UNCONFIRMED.

Track Wire Detector

Similarly to the previous sensors, the service posts a TANK_ON or TANK_OFF depending on the current value of the ADC. The hysteresis used here is 300 and 450. For both the Beacon Detector and Track Wire Detector can be turned off with a function that sets a static variable. Only if this variable is high, the timer that calls these services is reset. If the timer is never set, and never posts to the service, they never execute and never post their own events.

Bumpers

As the 4 directional bumpers were OR’d and then Mux’d together, it requires 2 input selector pins and one output pin. The other 2 tower bumpers are polled using separate GPIO pins. All bumpers have to be debounced first before they can generate events. This is done by checking the value of one of the 4 directional bumpers and the two tower bumpers every 10ms, and storing this value in an array of size 3. Only when all three values match is the event generated. A separate array is held for each of the directional bumpers, but due to the multiplexing, their sampling frequency is 40ms. In the case of an event, BUMPER_PRESSED or BUMPER_RELEASED is posted to the HSM.

Dispenser

The Dispenser service does not run periodically, but only when a function, namely DispenserStart() is called. This extends the servo piston arm, and starts a timer. When this timer

expires, the Dispenser Service is run, which resets the piston arm and returns a DISPENSING_FINISHED event to the HSM.

Motors and Encoders

Each of the motors required 3 inputs, for forward, reverse and enable, and had two outputs, A and B quadrature encoders. Forward and Reverse were set high depending on which direction was required, while the enable carried the PWM for the motor. This controlled the speed for each motor. Each of the four directions of travel were set using these 3 pins. The encoder is used to accurately measure how much each wheel is spinning. The encoder A and B pins are polled in an Event Detector as the highest possible testing frequency is required. As each encoder pin is pulled high and low, it traverses through the following state machine, where the "+" and "-" represent the count of each motor. When a certain amount of distance is required from the motors, this count is reset and an event is generated when the appropriate ticks have been counted.

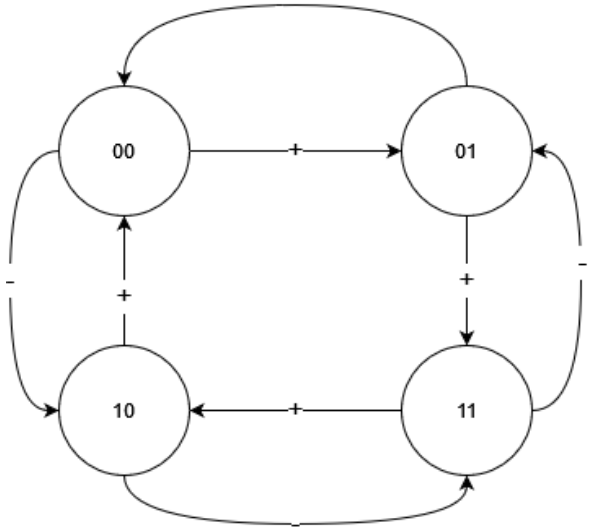


Figure: Quadrature Encoder State Machine

Hierarchical State Machine

The structure of the HSM creates layers of abstractions, allowing the top level machine to house the highest level logic. As such, the specifics of 'how' a task is to be completed is handled by the lower level FSMs. The HSM used here has 10 states, or 10 FSMs. As shown in the diagram, the initial state of the robot is always 'Orientate'. According to the rules of the task, the robot will always start in a corner of the field, and as such will be close to an outside edge. In this state, the robot finds this edge and aligns with it such that the front and rear tape sensors are on the tape. It then switches to the Edge Following SM, where the robot is constantly checking for beacons. Here, the front tape sensors are used to keep the robot on the tape, and take turns at the corners of the field. When the robot detects an object in its way, it switches to the Edge Obstacle SM, to travel around the object and get back on tape. If it detects a Track Wire, it is assumed that the obstacle was a tower, and the robot is currently on the correct side of said

tower. It then switches to the Deposit Ball SM, which is explained further. In Edge Following, if a beacon is detected, Travel to Beacon SM is run. If the beacon is confirmed, it starts moving towards the strongest beacon value, until it detects an object (assumed to be the tower) and switches to the Parallel with Tower SM. If the beacon is not confirmed, Edge Following SM is then run and the robot continues to search for towers. In Parallel with Tower, the robot makes the appropriate turns to perfectly align its right side, which houses the Track Wire detector, with a face of the tower, then exits to Find Correct Side CW SM. As seen in the diagram, Find Correct Side and Deposit Balls have two separate SM, one clockwise and another counter clockwise. The SM themselves are mirror images of one another, but have to be separated. As designated by the specifications, the robot can not have more than half of its body over the tape. When traversing the tower, if a tape is encountered and the robot can not reach the next side, it switches to the counter clockwise SM. Depending on which SM it is currently in, CW or CCW, it switches to the appropriate Deposit Load SM. This SM starts at one end of the correct side, testing the whole side until it finds the tape, indicating that the correct hole is also found. As the tape sensor is directly underneath the dispenser, the ball can then be directly deposited. The robot then switches to the Reorientate SM, which simply turns away from the tower and begins the Edge Following SM. Due to the nature of the ES Framework, switching directly to the Edge Following SM would force the robot to follow the tape once it is eventually encountered. Another benefit of this method is that Edge Following already has obstacle detection, thus not requiring another SM. As there is no maximum number of towers in the specifications, this loop persists indefinitely.\

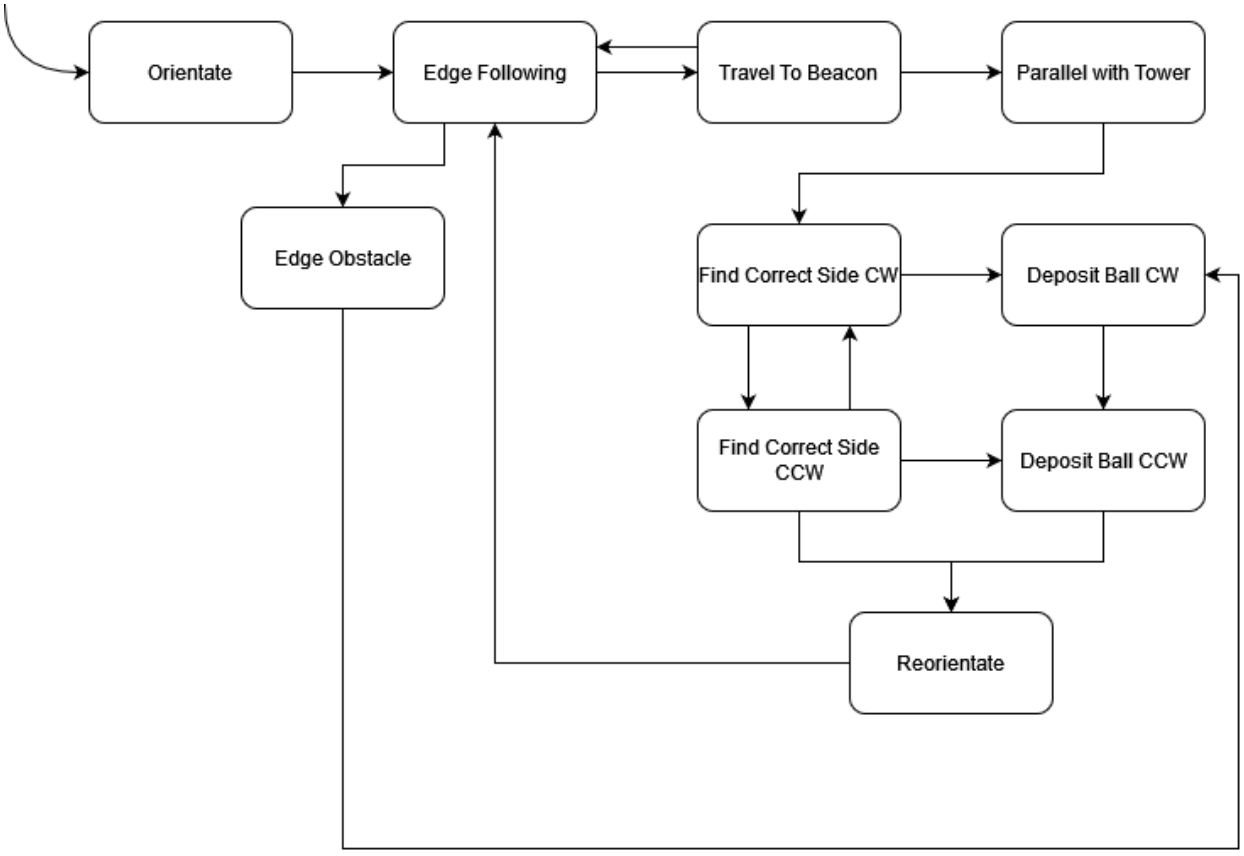


Figure: Top Level Hierarchical State Machine Diagram

Finite State Machines

Orientation

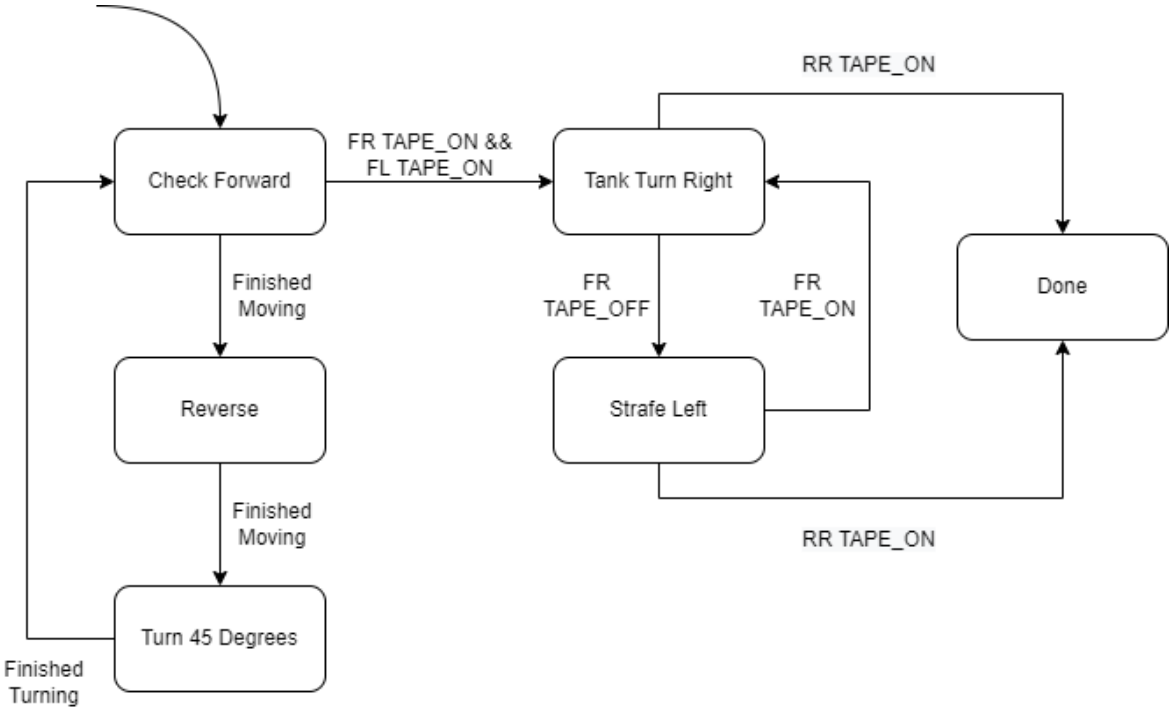


Figure: Orientation State Machine Diagram

The orientation SM locates and aligns the robot with the outer square tape. It does this by checking the forward direction until the two front tape sensors receive a TAPE_ON event. If no tape is detected after moving 6 inches forward, the robot returns to its starting position, turns 15 degrees, and moves forward again. Because the robot starts within a 11 by 11 inch square on the corner of the field, by the time the robot completes a full 360 degree rotation, a TAPE_ON event must be recieved. Once tape is found, the robot performs a series of strafe lefts and tank turns until the rear right tape sensor is over tape. From this state, the entire robot is aligned on the tape, and can begin its edge-following algorithm.

Edge Following

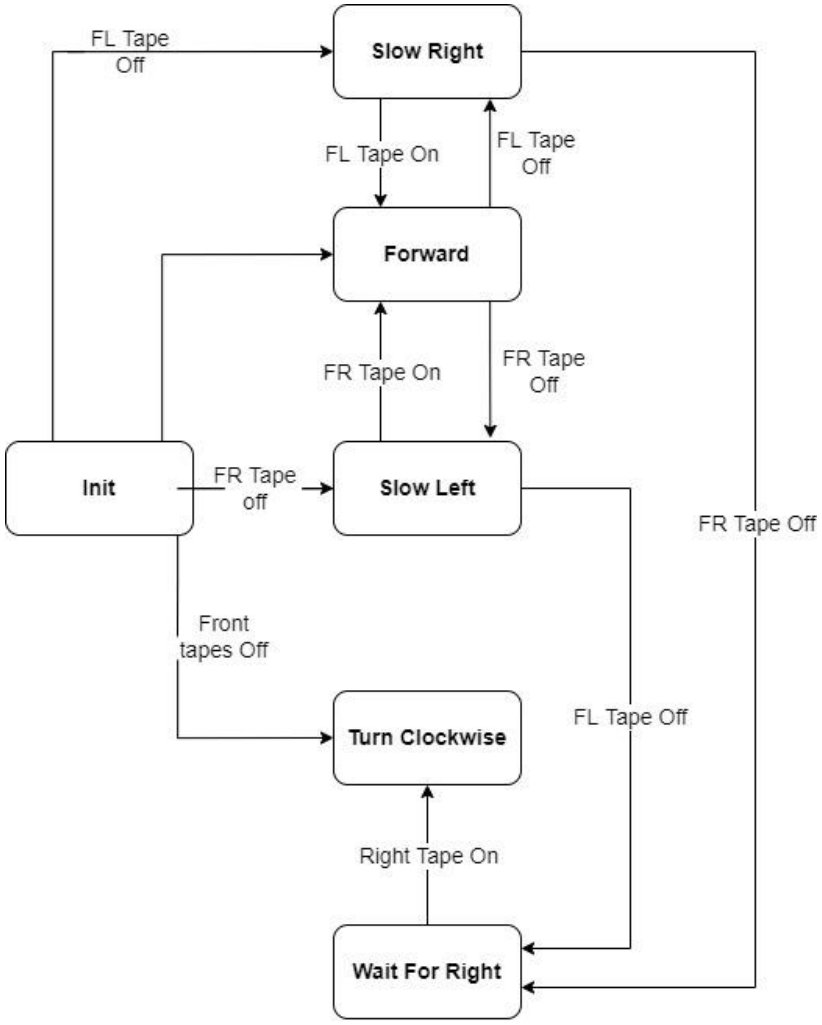


Figure: Edge Following State Machine Diagram

The edge following SM begins under the assumption that the robot is already aligned with the tape. It begins by always traveling forward until one of the two front tape sensors receives a TAPE_OFF event. If the front left sensor receives this event, the robot performs a right tank turn until the left tape sensor receives a TAPE_ON event. Similarly if the right tape sensor comes off the tape, a left tank turn is performed. These two cases ensure that the robot only moves forward while it is still aligned with the tape. When the robot reaches a corner, both tape sensors will receive a TAPE_OFF event within a short period of time. When this occurs, the robot will continue moving forward until either the right or left tape sensor receives an event. Depending on which tape sensor is triggered, right or left, the direction the robot is traveling in can be determined. A right tape event suggests that the robot is traveling clockwise, and a

left tape event suggests the robot is traveling counterclockwise around the outer tape. Because the beacon detector points to the right side of the robot, towers can only be detected while traveling clockwise around the outer tape. Therefore, if a left tape sensor is triggered, the robot must turn around and begin edge-following in the other direction. This is achieved by tank turning left until the front tape sensors receive TAPE_ON events. If the right tape sensor is triggered, the robot is already moving in the correct direction, and a tank turn right is performed until the front tape sensors are back on the tape. Once either turn is completed, the robot continues moving forward. In the forward state of this SM, if a beacon signal is detected, the robot stops and enters the Travel To Beacon SM. If in any state, a front bumper event is observed, this state machine exits to the Edge Obstacle state machine.

Travel to Beacon

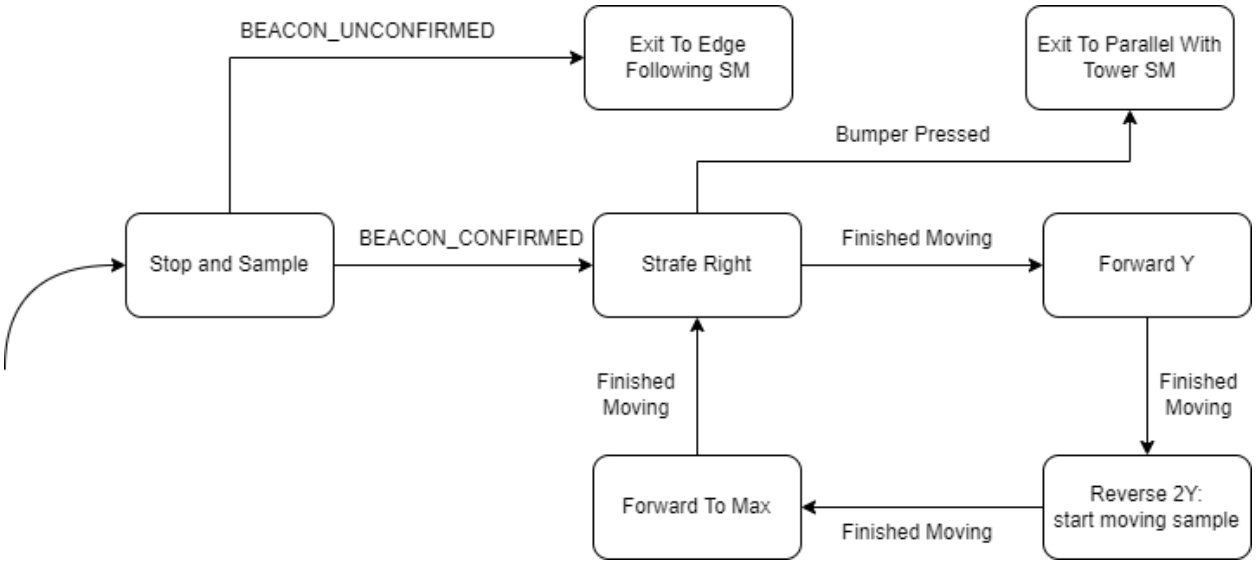


Figure: Travel to Beacon State Machine Diagram

To ensure that the signal received was in fact a tower, this SM first runs a stop and sample in which the robot takes ten samples over a second. If the average value is greater than the hysteresis bounds, the tower is confirmed. If the sample is not greater than the hysteresis bounds, this state machine exits back to edge following. Once a tower is confirmed, the robot strafes right towards the tower for approximately 6 inches. The robot then moves forwards and backwards while taking a moving sample. The moving sample stores beacon values and the encoder positions that they were taken at. Once the sample is completed, the maximum beacon value is found and the robot uses the encoder values stored at the max value to return to the position of the strongest signal. The robot then begins to strafe right again, repeating this algorithm until a bumper event is received. This algorithm ensures that the robot stays aligned with the tower when strafing towards it.

Parallel to Tower

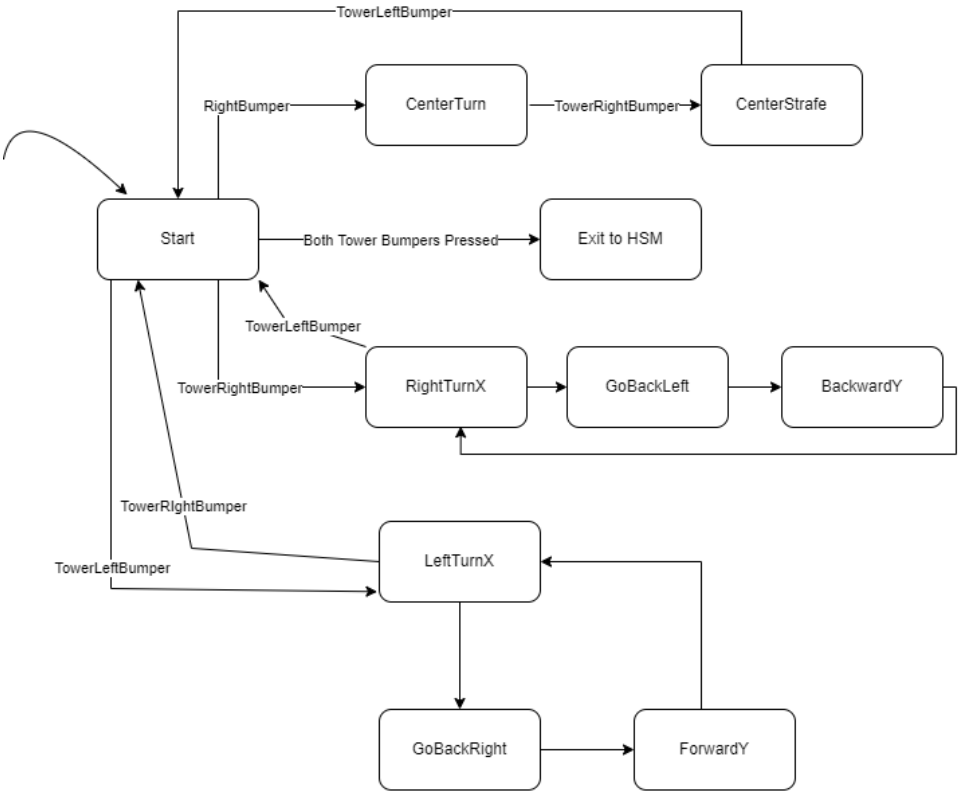


Figure: Parallel To Tower State Machine Diagram

This SM begins with a bumper event from a tower. One of the three bumpers located on the right side of the robot could trigger this state, so an initial polling of the previous bumper events is needed to determine which bumper was pressed. The purpose of this state is to align the robot parallel to the face of a tower. If the left bumper was hit, a strafe right is performed where only the rear wheel turns. This turns the robot around the front wheel's axis, keeping the left bumper pressed. Once the remaining right bumper is pressed, the robot stops turning as it is now parallel to the tower. Similarly if the right bumper is pressed, a strafe right is performed where only the front wheels turn. The last case, where the center collision bumper is pressed, suggests that the robot collided with the corner of the tower, missing the two bumpers positioned above. In this case, the robot first tank turns left until the right bumper is hit, and then strafes right towards the center of the tower. Once in the center the robot turns until the left bumper is hit, thus aligning with the face. Once aligned, this SM returns a DONE event to the HSM, resulting in a transition into the Find Correct Side CW state machine.

Find Correct Side CW \ CCW

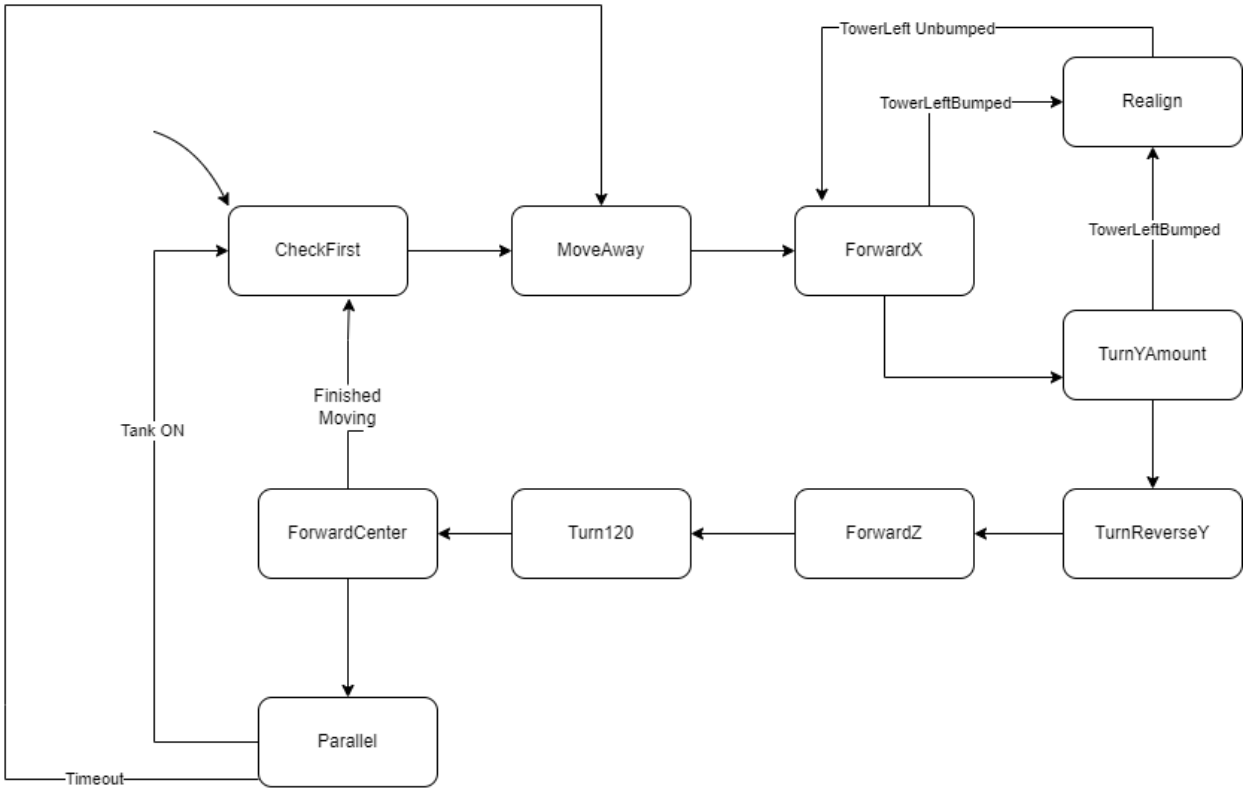


Figure: Find Correct Side Clockwise State Machine Diagram

Find Correct Side CW and CCW both travel around the tower, periodically checking the inductor reading to determine if the robot is currently on the correct face. To ensure that the robot does not become misaligned with the tower, the robot moves forward a small distance and turns into the tower until a bumper is pressed, either left or right depending on the direction it is circling the tower. The robot then turns in the opposite direction until that bumper is released, and moves forward a small amount. This constant bumper checking allows for the robot to remain aligned with the tower, and the ability to detect corners. If during the turn, a bumper event is not received, a corner is detected and the robot performs a 120 degree turn, moves forward, and stops in the center of the next face. If a tank circuit reading is not above the hysteresis bounds, the robot moves forward and repeats this process. When the correct face is detected, the corresponding deposit ball state machine is entered. If at any time during the search, a tape event or front bumper event is received, the robot has encountered an obstacle and continues its search in the opposite direction.

Deposit Ball CW/CCW

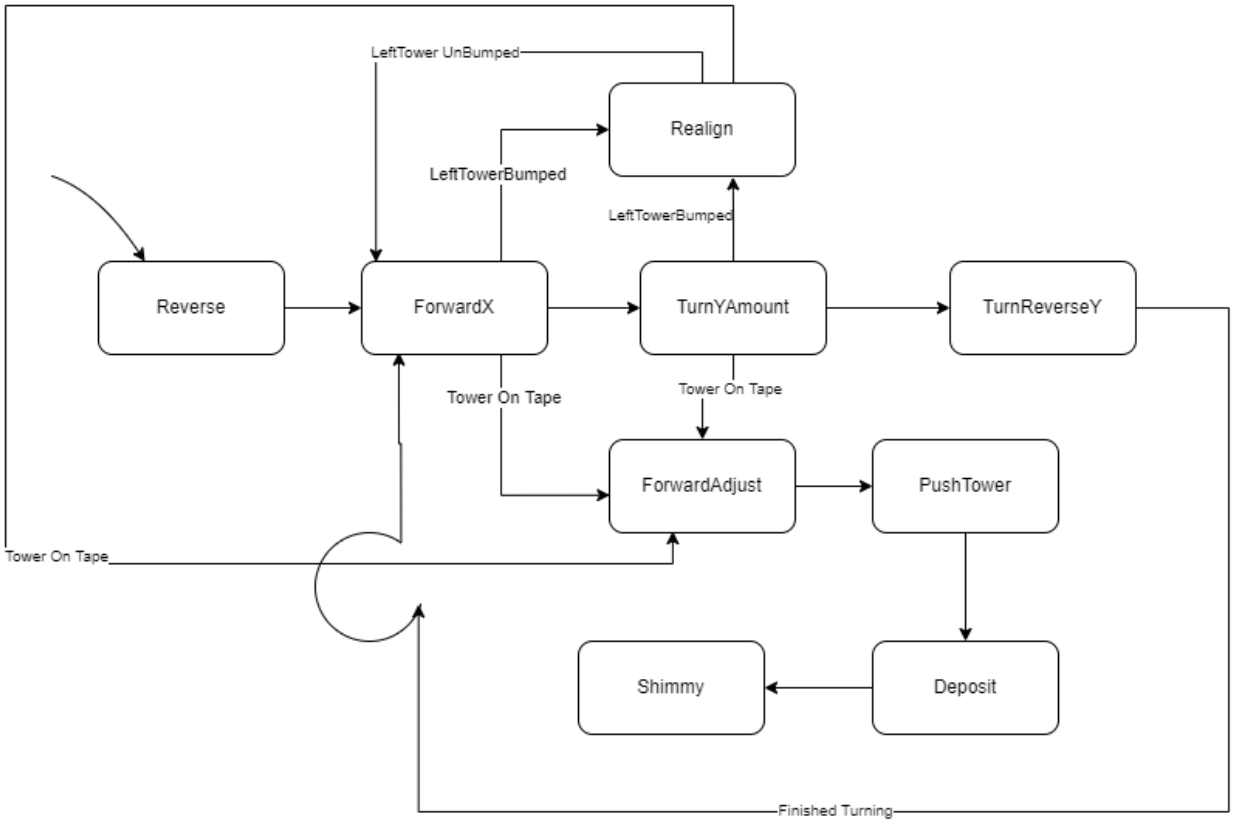


Figure: Deposit Ball State Machine Diagram

Deposit Ball CW is run when the correct side has been found. In the CW state machine, the robot reverses a short amount to line itself with an edge of the face on the tower. Similar to the Find Correct Side SM, it moves forward and turns towards the tower to avoid misalignment. In this checking phase, if both the tower tape sensors detect a TOWER_ON_TAPE event, the correct hole has been found. Here, the robot pushes itself towards the tower and gets as parallel as possible. Then the Dispenser service is called, and the robot stays stationary until a DISPENSER_FINISHED event is received. Due to the robot being slightly misaligned with the holes on certain occasions, the robot does a short “jig”. The robot moves forward and reverse for 250ms each, 5 times, to push the ball into the right hole. At this point, the ball should be in the right hole and the HSM switches to the Reorientate State Machine. If the HSM was previously in the CCW Find Correct Side, the CCW version of this SM runs. Here, the only difference is that the robot initially moves forward, instead of reverse in the CW, to find the edge of the tower. It then reverses to find the correct hole along the edge. The rest of the SM runs the same.

Reorientate

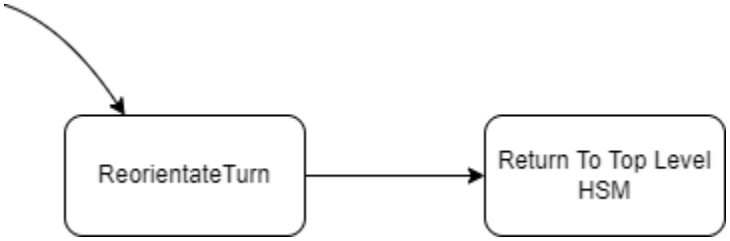


Figure: Reorientate State Machine Diagram

This SM is used purely to turn the robot away from the tower it has just completed. Upon entry to the SM, the encoder service is used to turn the robot 90º to the left. During this turn, if it encounters tape on both front tape sensors, it immediately switches to the Edge Following SM. It also switches to the Edge Following SM after completion of the turn. Using EdgeFollowing SM here, even when the robot is not on the tape has 2 uses. First, the initial state of the SM is 'Forward', where the robot goes forward until it receives a TAPE_OFF event. While moving, if it encounters a tape, it receives a TAPE_ON, then drives off the tape, receives a TAPE_OFF, and takes the appropriate action to start following the tape. The other advantage of using the EdgeFollowing SM, is the inherent obstacle detection through Edge Obstacle SM. This state is usually triggered by the front bumper, and the robot in this case should be moving forward.

Edge Obstacle

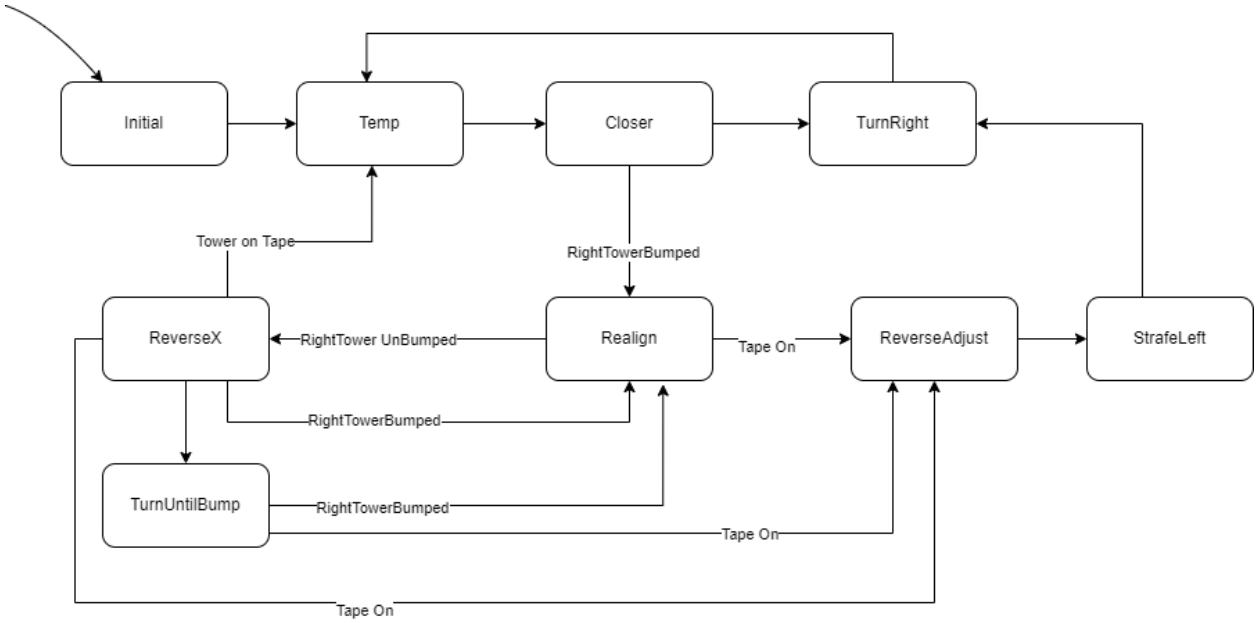


Figure: Edge Obstacle State Machine Diagram

This SM is used to circumnavigate any obstacle that is present on the tape when Edge following. When entering this SM, it is assumed that the front bumper is currently pressed, as such, the robot reverses a few inches then it tank turns to the left, to point its tower bumpers directly towards the obstacle. It then strafes to the right (towards the obstacle), until the right tower bumper makes contact. At this point, it uses a similar algorithm as before, but here, it moves forward, tank turns right until its bumper is

pressed, tank turns left until its bumper is released, then moves forward again. This process helps it get around any obstacle, no matter the shape and orientation. Due to the orientation of the robot itself, the rear tape sensors would encounter the tape first, as such, when both sensors return a TAPE_ON event, the robot strafes left for a couple inches, then the tank turns until the front tape sensors get a TAPE_ON event. At this point, Edge Following SM can be used to follow the tape again. Two edge cases have to be considered here, if the obstacle is a tower, and if the obstacle is not on tape originally. For the first case, the tank circuit is turned on during the Forward state. If the state receives a TANK_ON event, it switches directly to the Deposit Ball SM to complete the tower. For the other case, when the obstacle is in the middle of the field, the robot should simply leave the obstacle after a certain amount of time. Here, a timer is started when the SM starts, and if a TIMEOUT event is ever received, it immediately switches to the Edge Following SM, i.e it starts driving forward in a new direction.

Mechanical Components

As per requirements, the robot must fit within an 11”x11”x11” cube, and must be able to detect collisions with obstacles above a height of 3”. The robot was designed using solidworks 2019, and was laser cut out of 0.21” sheets of mdf.

Four Bemonoc 12V 130RPM motors were used in this project. Each of these motors have a hall effect encoder that outputs quadrature A and B. As the battery used on this project was a 9.9V, the motors never received 12V and thus outputted less than 130RPM. The motors were run using 2 H-Bridges (one for each side). The encoder itself required 5V which was generated using the 5V voltage regulator. Eight tape sensors were used, six are mounted on the bottom of the robot for field boundary detection and the remaining two are mounted on the dispenser, used for finding the correct hole of the vat tower. A total of ten bump switches were used in this design. Two switches were used per collision bumper located on each side of the robot. The remaining two switches were added to the right side of the robot to assist with aligning with towers and obstacle avoidance. One servo motor was used to actuate the piston dispensing mechanism.

CAD Design

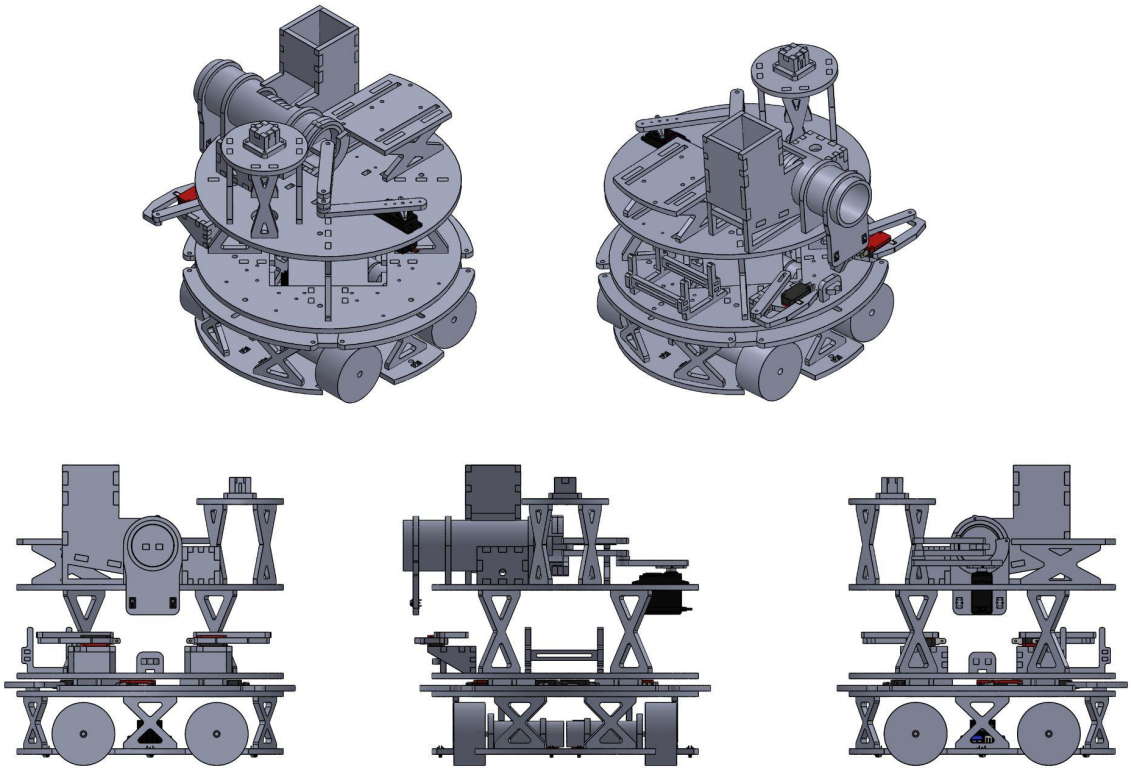


Figure: CAD Design Overview

Mecanum Drive

In order to navigate the field, locate towers, and deposit balls into the towers, precise position and orientation of the robot is a primary concern. Two wheel drive is simple to implement, however it is limited to motion in a single direction. For small adjustments in position of the robot, a robot using two wheel drive must make a series of turns to align its wheels with the desired direction. Mecanum drive, however, allows for motion in any direction without any change to the robot's orientation. Thus, this driving method is optimal in allowing the robot to make many small adjustments to its position and orientation when navigating the field and locating towers.

The following image depicts the directions the wheels must turn to achieve desired motion.

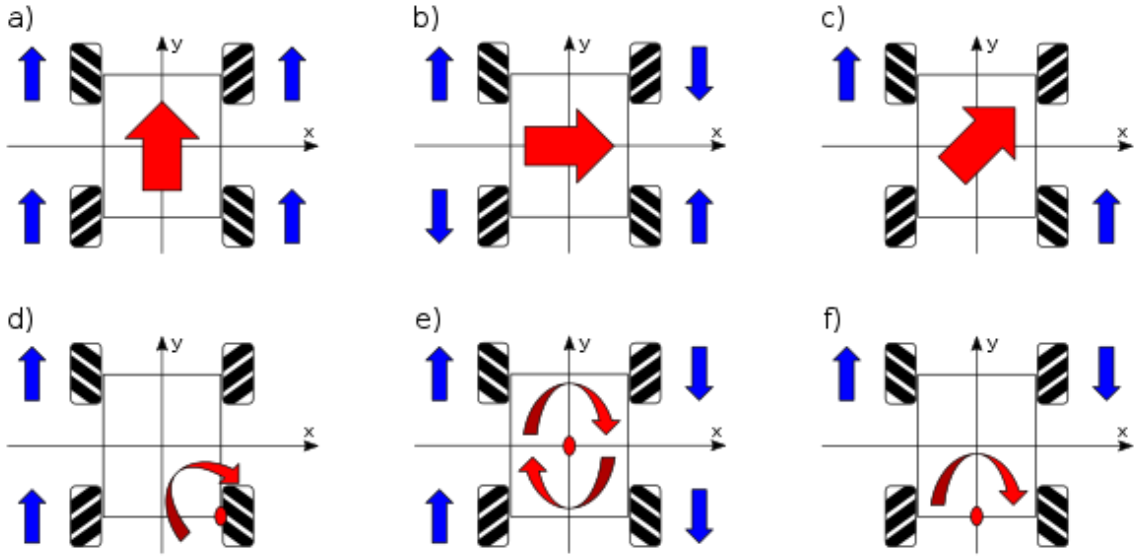


Figure: Driving Mecanum Wheels Diagram

Robot Body

The body of the robot is split into three different levels. The bottom level contains the motor mounts, the H-bridges needed to drive the four motors, and the six tape sensors. The layer above it contains the microcontroller, bumpers, and mounting holes for the circuitry. The top layer contains the ping pong ball dispensing mechanism and a small stand for the beacon detector. Between the layers, there are tab and slot supports that are squeezed together using all-thread screws. No adhesives were used to hold the layers together, which allowed for easy disassembly and modification.

The modular design of the different layers allows for incremental testing of each part of the robot. It allows for prototypes to be made and tested simultaneously. With the large focus on precise motion of the robot and its navigation around the field, the navigation algorithms needed to be tested and protyped early. This modular design allowed for this early prototyping of the mecanum drive and testing of navigation algorithms, while the dispensing mechanism was being designed.

Tape Sensors

Six tape sensors are placed on the bottom layer of the robot. The two tape sensors on the right and left side are used to detect tape corners and the four tape sensors along the center are used to follow tape once aligned with it.

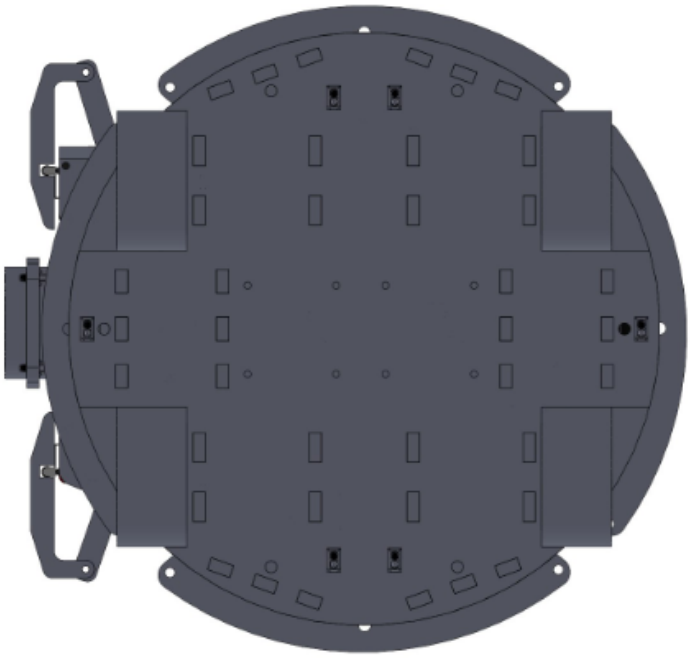


Figure: Bottom of Robot

Bumpers

The bumpers are used for obstacle detection and tower alignment. There are four bumpers located on each side of the robot that are used to detect collisions on the field. These collisions can come from a robot on the field or from running into a vat tower. The two bumpers located on the right side of the robot are designed to avoid obstacles and navigate around the tower. Because the bumpers are separated by approximately 8 inches, they can be used to detect corners of the vat towers and resolve obstacle collisions.

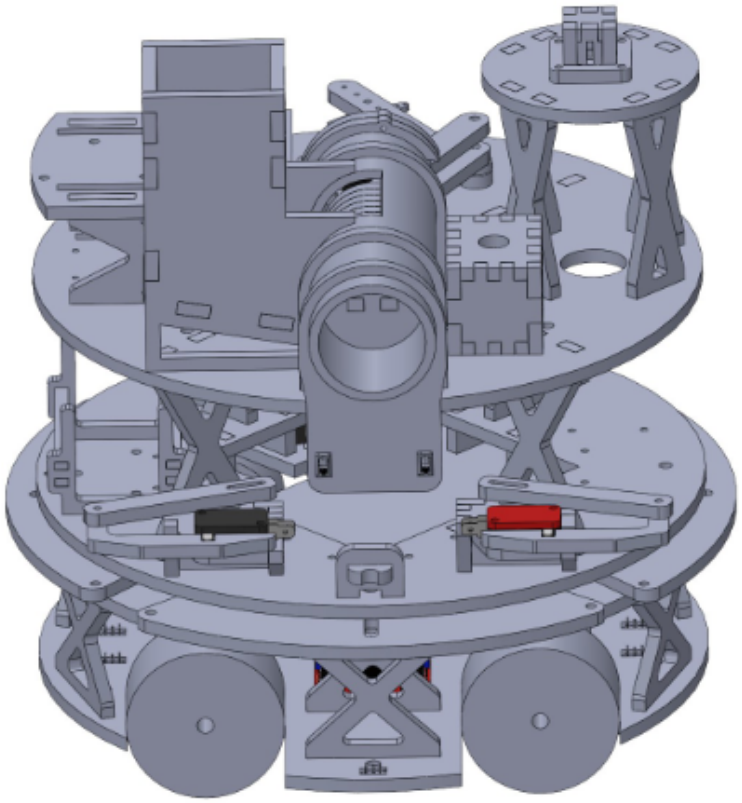


Figure: CAD Design of Tower Bumpers

Ball Dispenser

To dispense ping pong balls, a piston mechanism is used to push a ball through a pvc pipe and into a hole in the tower. The pvc pipe has a hole on one side that allows a ping pong ball to fall into it. When the piston is fully extended, the plunger blocks this hole, to keep the balls from falling into the pipe. To dispense a ball, a servo is used to pull the plunger past the hole to allow a single ball to fall through. The dispenser is then extended pushing the ball out of the pipe, while also blocking the hole. The diameter of the pipe was chosen to be close to the diameter of a ping pong ball so that when the hole isn't blocked, only a single ball can fall through. The two tape sensors mounted at the end of the pipe are used to determine the correct hole once the correct face of the tower is found. he

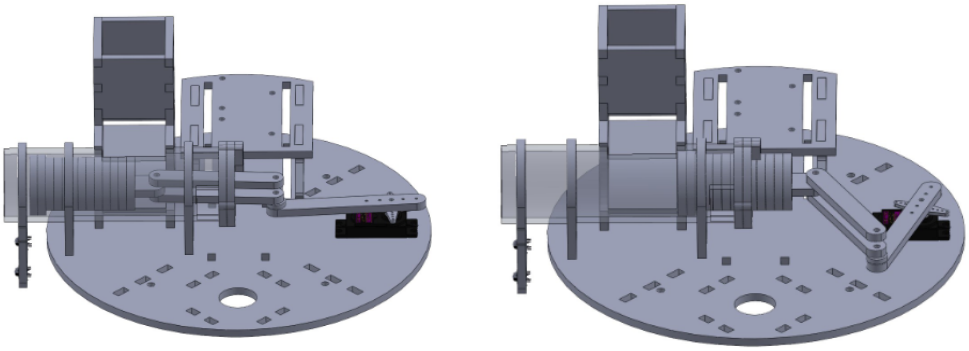


Figure: Ball Dispenser Piston Design

Beacon Detector

The beacon detector sits at a height of 11 inches, the height at which the beacons are mounted on the vat towers. The photo-diode sits within a cube casing that has a single thin slot on one side. The shield makes the beacon detector more resistant to noise, and the slot narrows the range at which a tower can be detected. The cube casing mounts to the platform with screw holes for easy removal and modification. The beacon detector points towards the right side of the robot to allow for tower detection while following the outer tape.

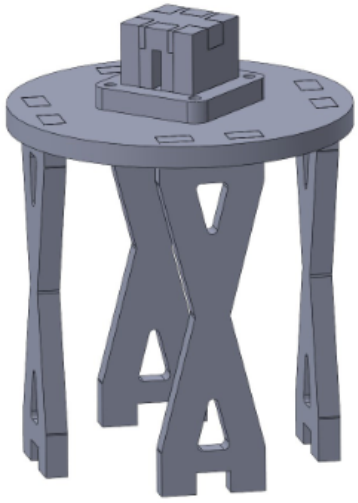


Figure: Beacon Detector Mount

Electrical Components

There are four main electrical components on the autonomous robot: the beacon detector, track wire, tape sensors, and bumper sensing. The beacon detector is used to locate the vat towers and the track wire is used to find the correct face of the vat tower. The tape sensors are used for the edge-following algorithm, as well as to stay within the bounds of the active field. The bumper sensing is used to detect obstacles, as well as be useful for getting parallel to the vat

tower and traversing around the vat tower. Each component was first done on a breadboard to test. Once that was done, pinouts were created. The pinouts were then soldered on small perfboards, to reduce the amount of space they take on the robot, as well as weight, and were tested incrementally to make for easy debugging.

Beacon Detector

An infrared emitter is used as a beacon, which sits on top of each vat tower. These beacons let the robot know there is an active vat tower near them that they can then traverse to. The emitter is driven by a microcontroller, and generates a 50% duty-cycle wave at a frequency of 2 KHz. The design for the beacon detectors were created a few labs ago. In order to accommodate the algorithm created to solve the overall project themed challenge, a new beacon detector design was implemented for the final project. The beacon detector was created where the range is less than 5 feet; the purpose of this is to be able to only detect beacons in a quarter of the field. Additionally, since an edge-following algorithm is used, as discussed above, having a beacon detector that has a shorter range enables us to only detect the closest beacon, which avoids any issues of detecting a vat tower at the other end of the field, which the robot would eventually get to with the edge-following algorithm.

The beacon detector consists of the infrared emitter, which is then fed through stages of filtering, with a 4th order band pass filter, for the noise due to other light interferences such as computer monitors, wall current, and the like. There is a split rail buffer to offset the filtering stages to have a virtual ground of 1.65 V, which is then removed with ac coupling in between the gain stages. Once the signal is large enough, the output is fed through a peak detector. In a previous lab, a comparator was used before the peak detector, however it was decided to remove it entirely from the circuit and instead do that in the software portion. This makes it really easy to have different hysteresis bounds when testing the robot in different rooms that have different lighting that would affect the reading from the beacon detector.

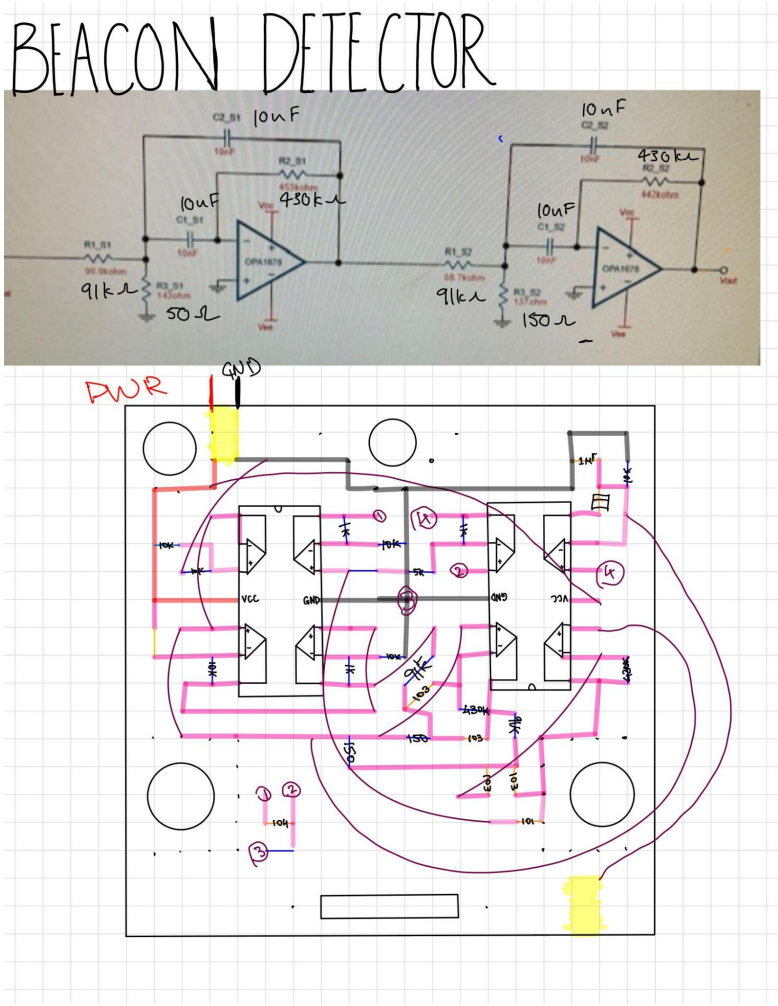


Figure: Beacon Detector Pinout and Filtering Circuit

Track Wire Detector

The track wire circuit is first introduced in Lab 1, and is used again in the final project to determine the correct face of the vat tower. The track wire carries a current which oscillates at a frequency of 24-26 kHz with a peak to peak amplitude of about 150 mA. This results in an oscillating magnetic field around the wire. When an inductor is in this oscillating field with the right orientation, the inductor will experience an oscillating EMF, which can then be detected as voltage. For the vat tower, the track wire is taped on the backside of one the faces. The inductor is strategically placed on the robot in order to detect the track wire.

The tank circuit that describes the behavior above is created, and then fed into two filtering stages in order to get a really clean signal that can detect 25kHz. A split rail buffer is also used to offset the filtering stages so that instead of going to true ground, it will go to 1.65V. Once the filtering is done, which already has an internal gain stage of 100, the output goes into some DC coupling to get rid of the bias from the split rail. Afterwards, another gain stage is used before going to the peak detector.

This is also demonstrated in the figure below, where the pink indicates where the solder will go and red and black determine power and ground respectively. Blue are resistors and orange represents capacitors, and the curved lines indicate wires that will be soldered to make connections that cannot be done with solder lines. The green symbol indicates the diode for the peak detector, as well as the orientation it should be in. The yellow blocks indicate the header connectors where the top left has two wires inputted for power and ground, and the bottom right one has one wire connected to the output.

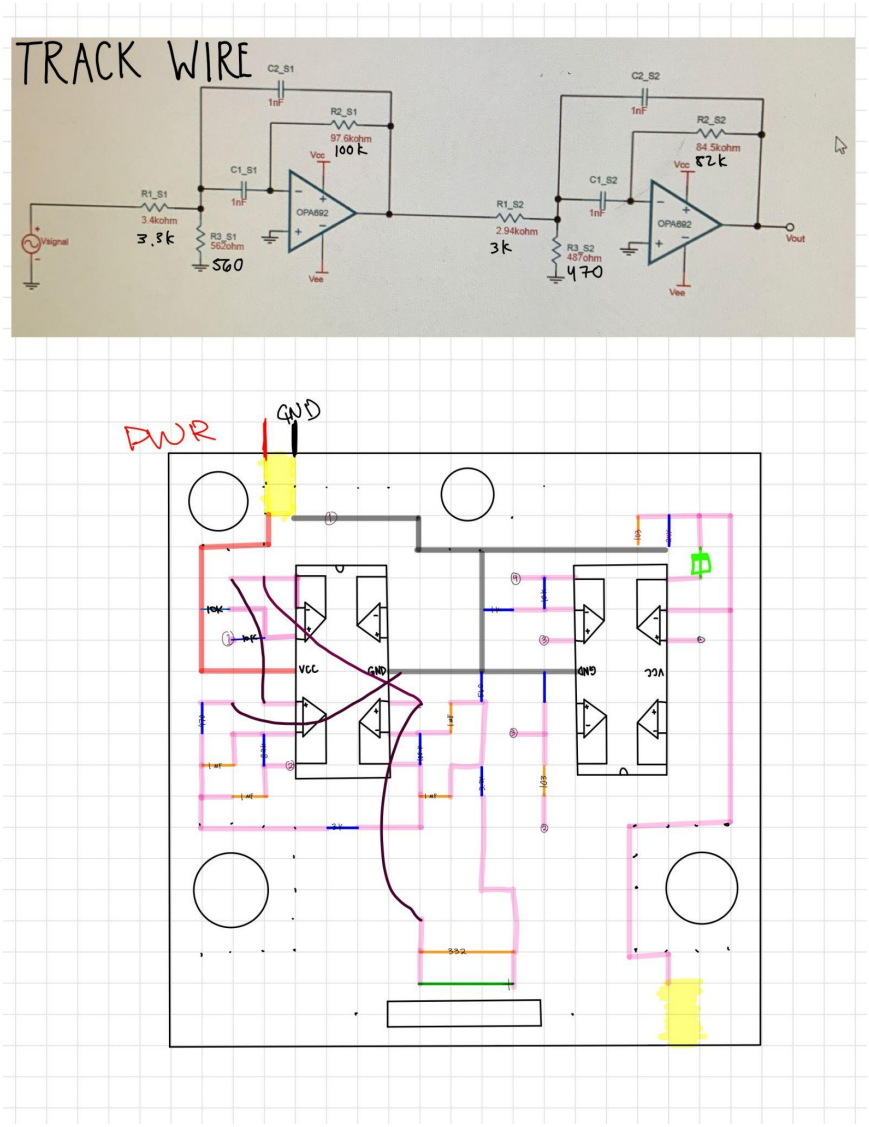


Figure: Track Wire Pinout and Filtering Circuit

Tape Sensors

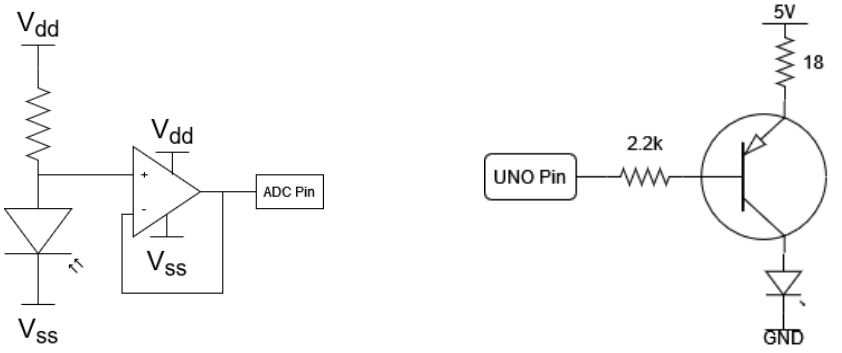
Infrared sensors were used for two reasons; to detect the black tape borders for the field, as well as the black tape on the vat towers that denote the correct hole to deposit a ball into. An

infrared sensor is an electronic device that measures and detects infrared radiation in its surrounding environment. Active infrared sensors were used for this robot; active infrared sensors both emit and detect infrared radiation. This is done with a light emitting diode, an LED, as well as a receiver. When an object comes close to the sensor, the infrared light from the LED reflects off of the object and is detected by the receiver.

There are 6 tape sensors on the bottom of the robot. Two in the front and back, and one on each side of the robot. The front and back tape sensors are largely used in the edge-following algorithm and the distance between the two tape sensors are within the width of the black tape. Once the black tape is not detected anymore, the robot will shift to the left or right, depending on which tape is not on black tape. One of the functions for the left and right tape is to be able to know when the robot is on a corner; if the robot has the front two tape sensors off black tape, while the right sensor and back tape sensors are on black tape, the robot would know it is on a corner and can turn until the front tapes are on black tape once again.

Each phototransistor on the tape sensor is set up as a voltage divider, with one 10k ohm resistor and the transistor. Due to this arrangement, when the transistor receives the more light, it lets more current through, decreasing its resistance and lowering the voltage drop across itself. This increases the voltage input to the op-amp buffer. Thus, as the amount of light that hits the transistor decreases, the greater the value on the ADC pin.

On the LED side of the tape sensor, a TIP122 is used as a power darlington to generate the required amount of current. As there were 8 tape sensors used, with each requiring 25mA of current, the total required current was 200mA. These LEDs were powered through a 5V rail, but with a voltage drop of 1.2V across the TIP, they received 3.8V. Thus the required resistor here is 19ohm, and the closest available resistor used was 18ohm. The UNO connected to the base of the TIP through a 2.2Kohm resistor to fully saturate the chip on a 3.3V rail. This allowed the software to switch the tape sensors on and off.



(a) (b)

Figure: Tape Sensors (a) Phototransistor Circuit (b) LED Circuit

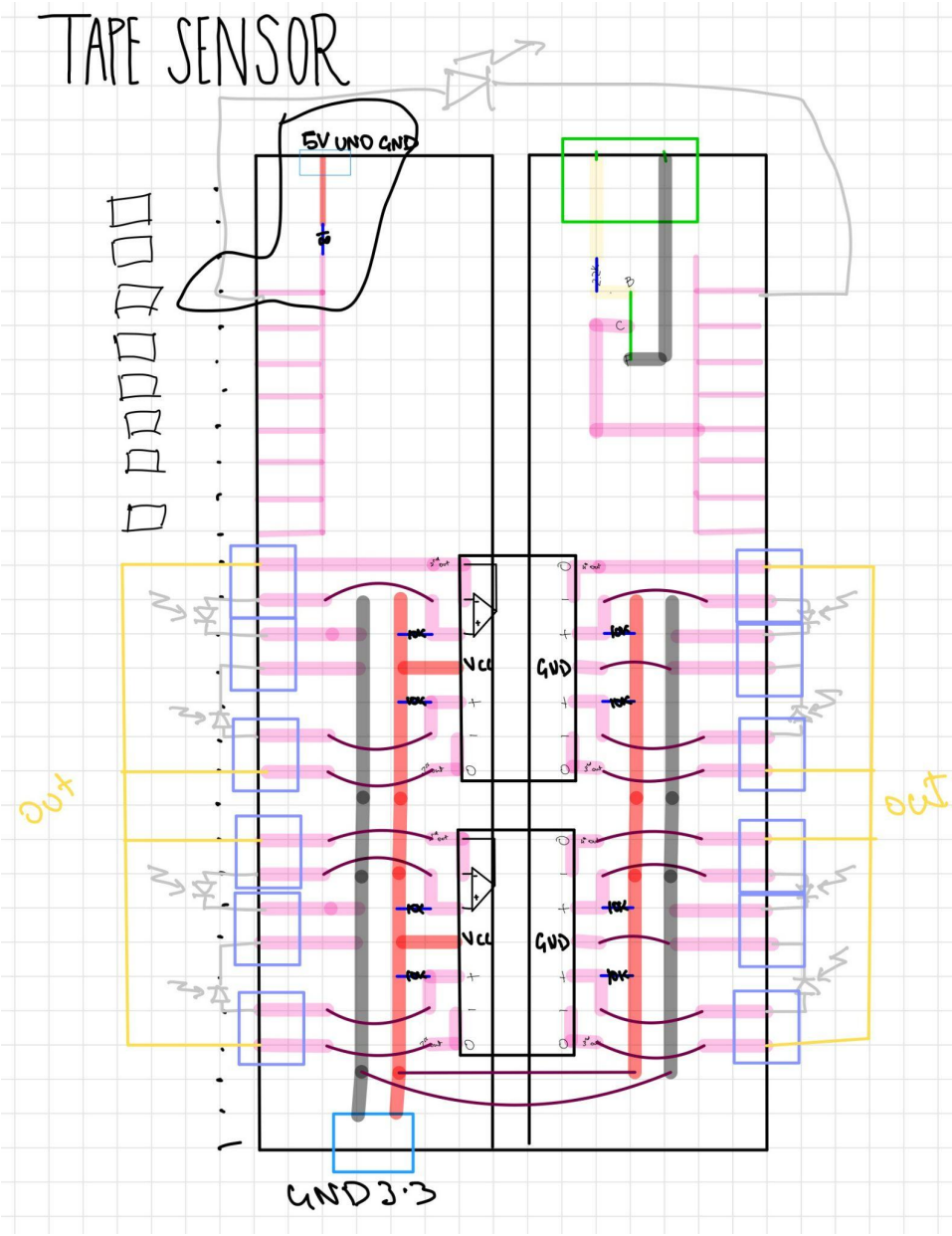


Figure: Tape Sensor Circuit

Bumper Sensing

Switches that have metal whiskers attached to them are used for the robot to recognize that it has bumped into something, whether that be another robot, a vat tower, or some other obstacle. To avoid these metal whiskers getting caught on another robot, bumpers were created in the mechanical design to be more effective. There are four bumpers around the robot that detect these scenarios. Additionally, there are two extra bumpers at the front of the robot, that help ensure that the robot is parallel with the vat tower, which is important for traversing around the triangular tower as well as aligning properly with the correct hole. These bumpers are also explained and shown in the mechanical design portion above.

For the four bumpers that are around the robot originally had one switch for each bumper. However, to add more stability and reduce false positives with being bumped, two switches were used for each bumper to make them more robust. To reduce the amount of pins going into the uno, every pair of switches per one bumper were fed into a logical OR chip, MC14071. The four outputs are then fed into a multiplexer, HEF4051BP. The pinout for this perfboard is shown below, where the first table on the left denotes which selectors should be high to get what output on the chip. An 8-to-1 multiplier was used, however only half of it was needed. As a result, one of the selectors is always set high, since the last four outputs of the 8-to-1 multiplexer were used. The second table denotes which bumpers correspond to the inputs for the logical OR chip, as well as the following output from the multiplexer.

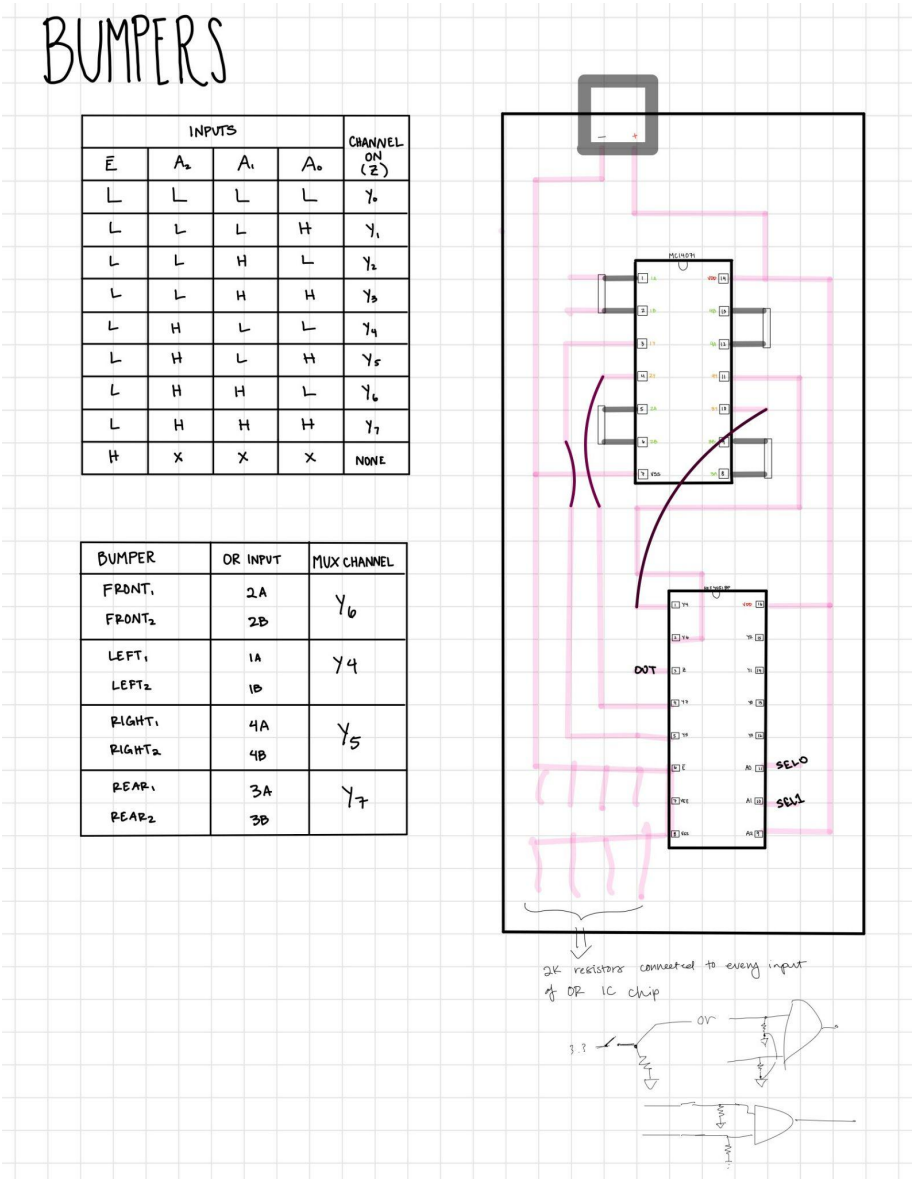
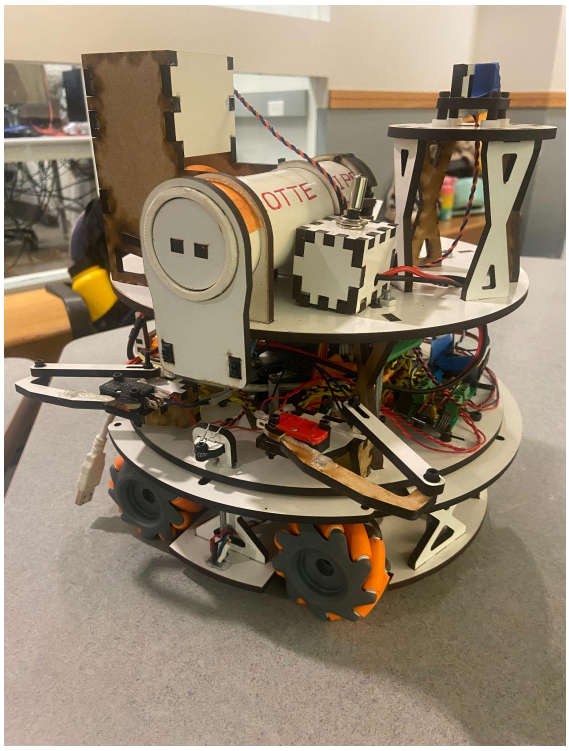


Figure: Bumpers Circuit and Output Table

Conclusion

This project accumulated many different topics and fields critical to robotics including circuit design and construction, embedded systems programming, and mechanical design using cad software. Aside from that however, this project heavily stressed the importance of working in a team. Designing, constructing, and programming a robot in 5 weeks comes with a large amount of work that is near impossible to complete alone. This project heavily stressed on the importance of team collaboration, and the ability to communicate ideas in a group setting.

The minimum specification required of the robot is to locate and deposit balls into two unique towers within two minutes. This must be completed twice out of three randomly generated fields to ensure that the design of the robot and its strategy in completing towers is robust. Out of the three randomly generated fields this design completed the first two fields in 45 seconds and 90 seconds respectively.



Video: https://www.dropbox.com/s/vkpwfoddni4522b/Video%20Dec%2001%2C%201%2056%2026%20PM.mov?dl=0

More Pictures: https://www.dropbox.com/sh/kfrermv8llddl4w/AAC4yeaBymuFC3XoTXxEthJIa?dl=0