Hexagon Robot

Spring 2024

ECE 361

Shane Wood and Logan Martin

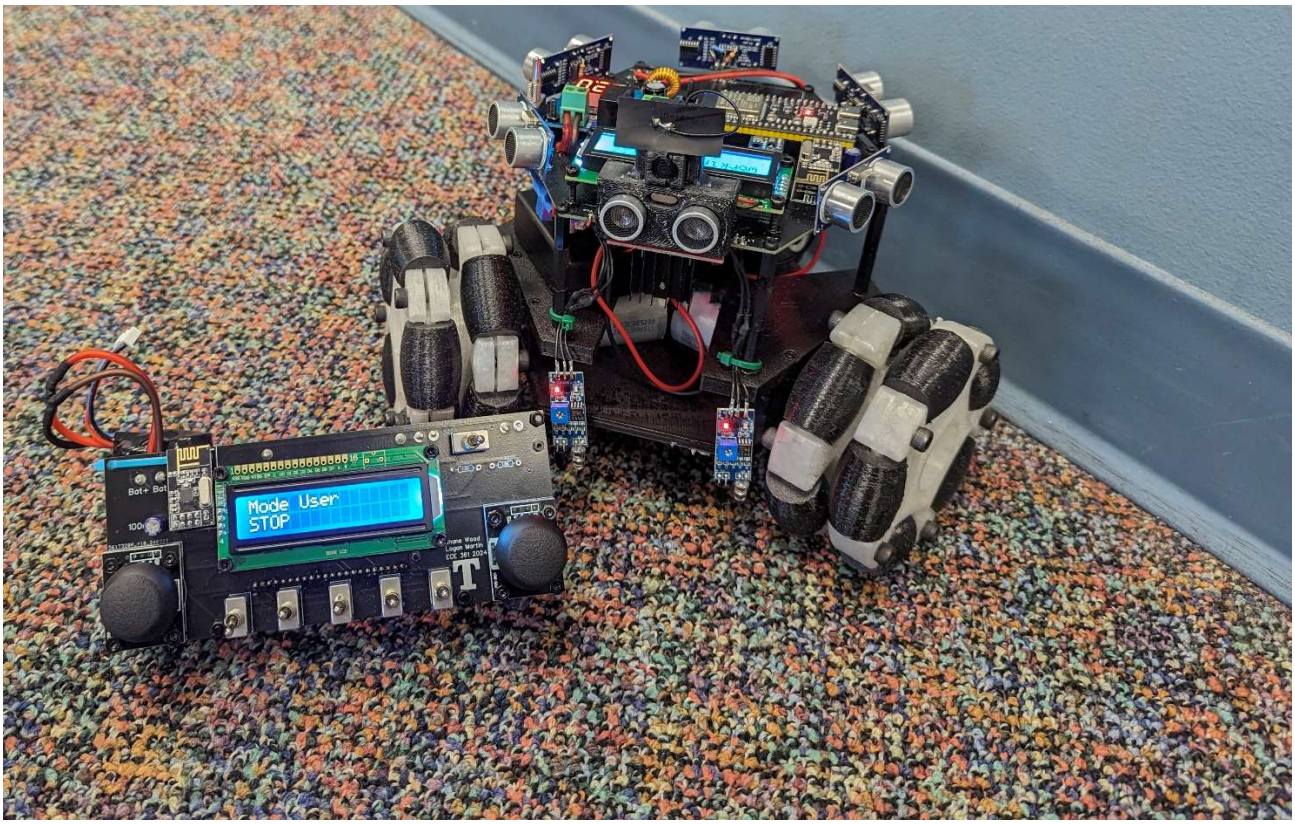


Table of Contents

Preface:

This is a robotics platform that uses three omni wheels for locomotion. The robot has three main modes: user control, wall following and line following. The robot has a PCB which makes wiring easier and features a custom controller.

Bill of Materials

Electronics

Microcontrollers

- 2 ESP32-S3 DevkitC – Microcontroller in the robot and controller
- 1 ESP32-S3 Seed Studio Xiao Sense - Microcontroller dedicated for the camera web server.

Motors

- 3 12V 120 RPM Geared DC Motors – Three drive motors for the robot

Motor Drivers

- 3 L298N motor drivers – PWM and direction control for the motors

Controller Inputs

- 7 SPDT switches -Power and data input switches
- 2 Joysticks – Data input joysticks for the controller

Sensors

- 6 HC-SR04 - Distance sensors
- 1 BMP180 - Temperature sensor
- 2 IR Infrared Obstacle Avoidance Sensor Module – Line following sensors

Power

- 12 V battery – Battery in the robot.
- 7.7V battery – Battery in the controller.
- 2 Buck Converters – Converting the battery voltage to TTL level (5 V).

Other

- 2 NRF2401 – Radio modules used between the controller and robot.
- 2 I2C Character Displays – Displays controller information and robot status.

Hardware

- M3 Standoffs, assorted sizes, and quantities – Holds the PCB above the body and keeps the controller together.
- M3 Heat set inserts – Attaches the PCB to the body and the wheels to the motor shafts.
- M3 bolts and nuts, assorted sizes, and quantities – Holds everything together.
- M2 bolts and nuts, assorted sizes, and quantities – Holds components with poor tolerances together.
- M5 bolts, assorted sizes, and quantities – used to hold the wheels together.
- 2.54 Male Headers – provides connection points.
- 2.54 Female Headers – provides connection points.

Printed Pieces

- 1 Top plate - PLA
- 1 Bottom plate - PLA
- 3 motor mounts – PLA
- 3 Wheel hubs – PLA
- 3 Wheel outer plates - PLA
- 3 Wheel inner plates - PLA
- 30 Wheel rollers - TPU
- 30 Wheel shafts - PLA
- 3 Wheel inserts - PLA

Circuit Boards (PCBs)

The PCBs were manufactured by JLCPCB. The boards have two layers.

- 1 Robot Board
- 2 Controller Board – only one is used, the other is to cover the electronics.

Software Tools Used

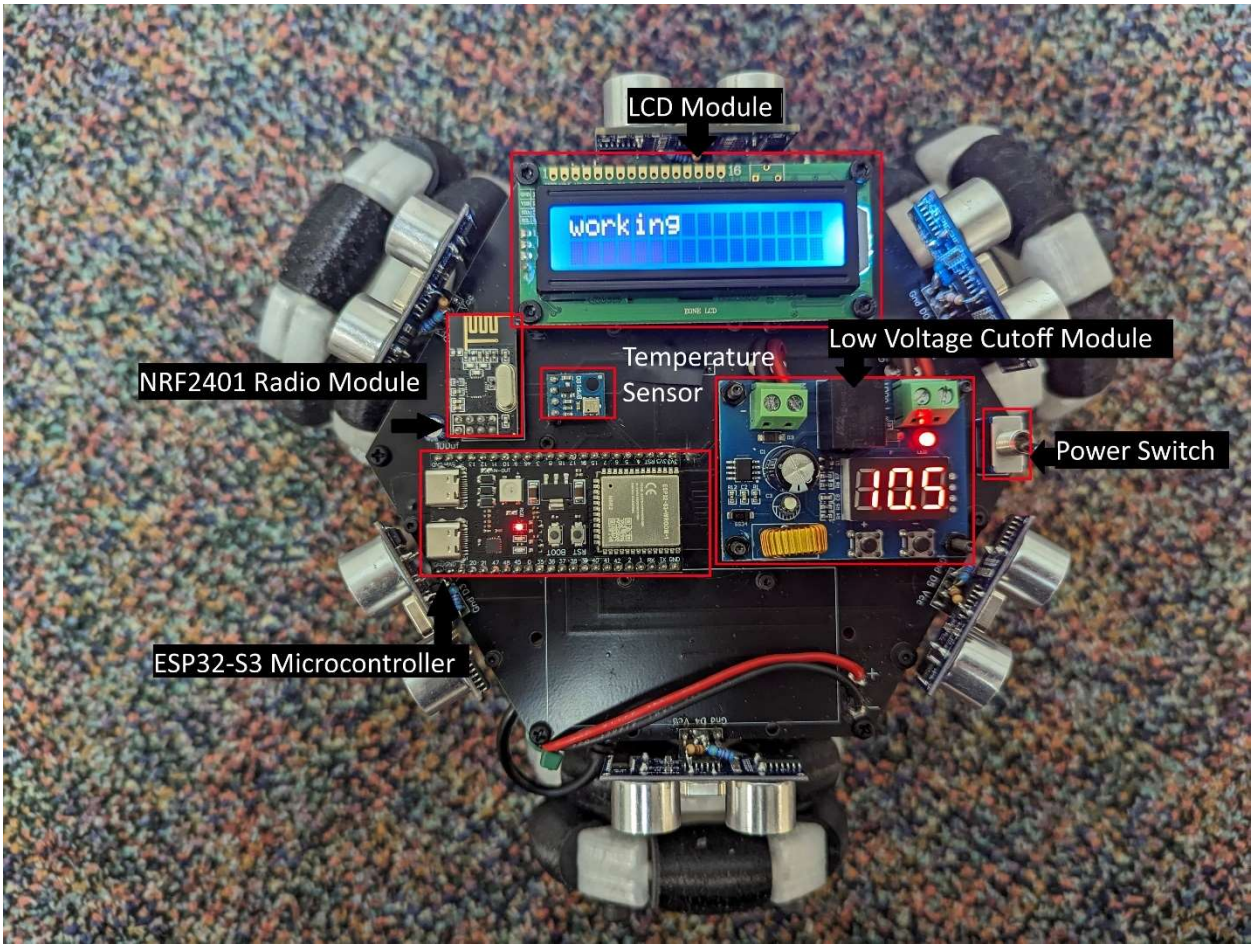
Blender – Used to create the assembly of parts to test measurements compared to the physical parts.

EasyEDA – Used to design schematics and PCBs.

Freecad - Used to design the body using CAD.

VS Code with PlatformIO – Used to program microcontrollers.

The Design



Size and Shape

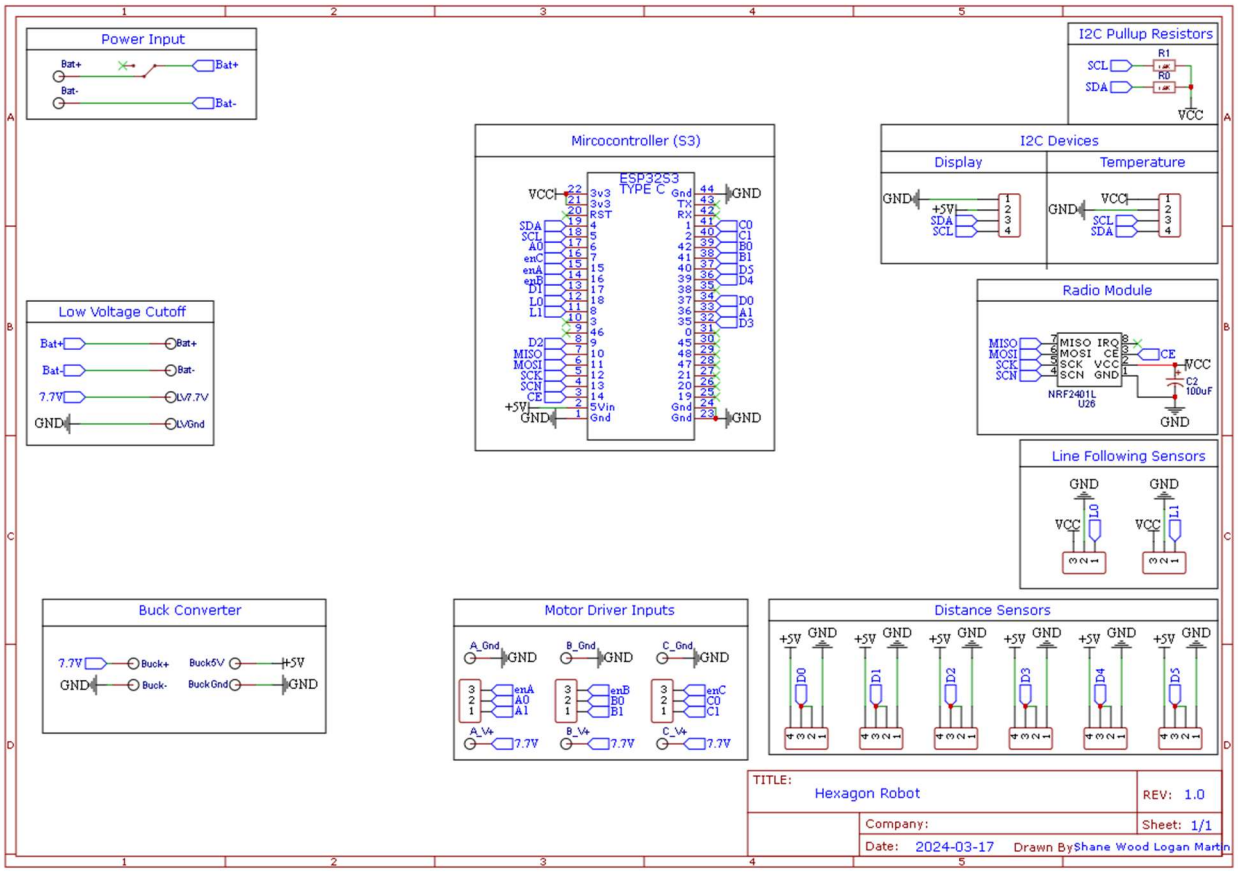
To break the typical mold for omni wheel robots, which are usually a triangle or circle, this design features a unique hexagonal shape, allowing for sensors to be in locations close to a circle, without being circular.

Wheels

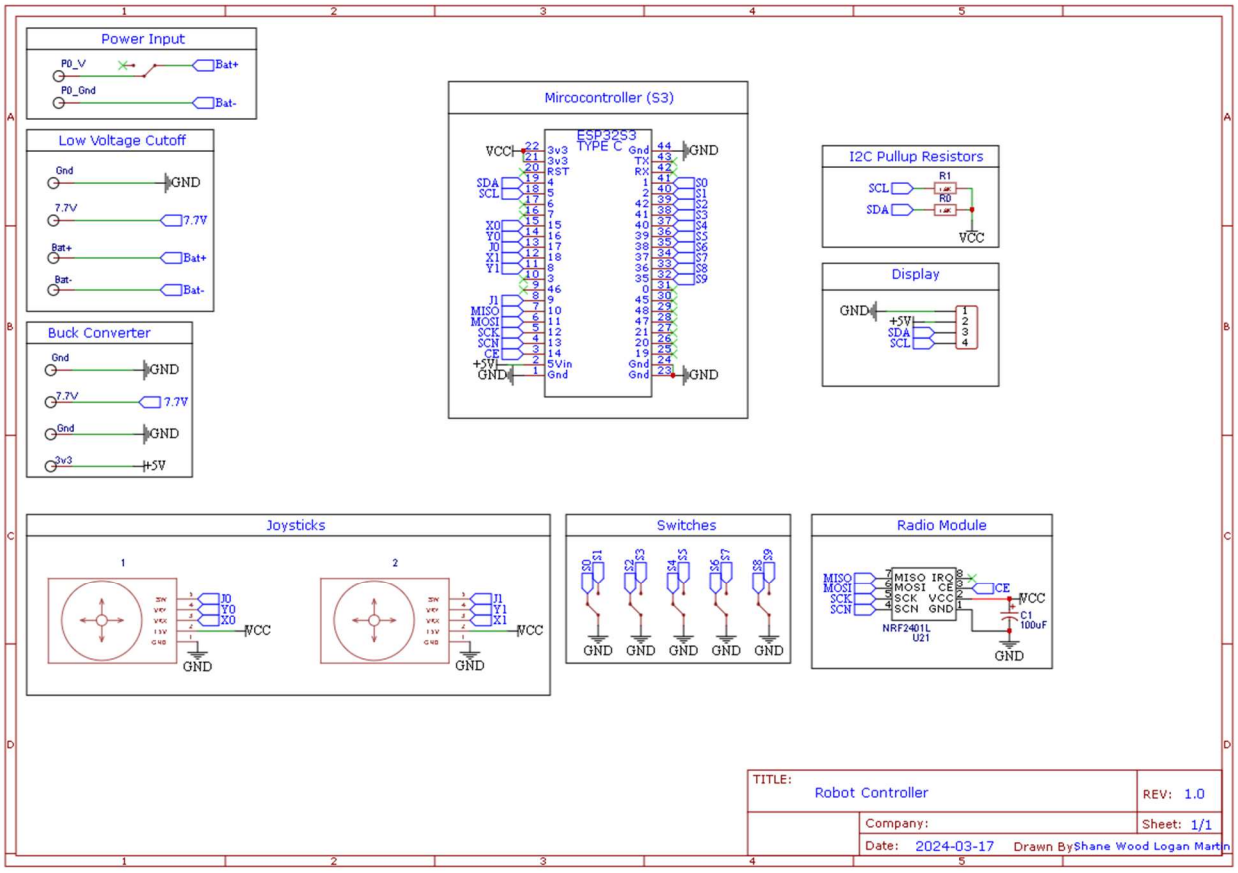
The wheels posed a challenge for this design as the original wheels were too small given the size of the rest of the robot. The old wheels were then replaced with larger ones, which caused other issues. One of these issues was a major lack of traction as the robot was not capable of driving on smooth surfaces after the wheel swap. To rectify this problem the rollers on the wheels were switched out for 98 shore hardness TPU to get the wheels more grip and flexibility. The motor shafts on the wheels would also come loose over time, requiring the need for wheel inserts with no tolerance which made it difficult to press fit the wheels on the motors, therefore a M3 bolt was installed on the wheel providing pressure on the motor shaft.

Schematics

Robot Schematic

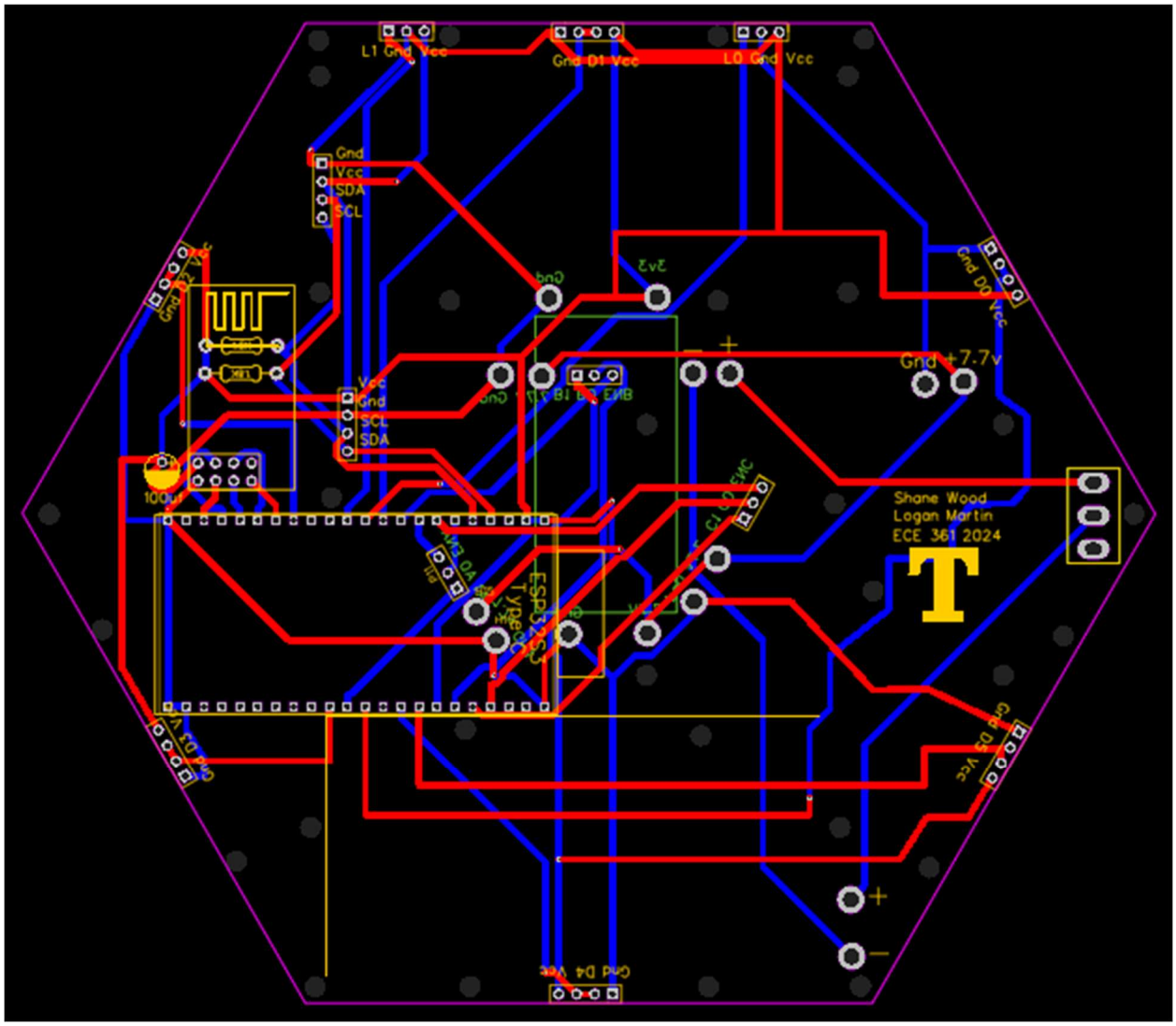


Controller Schematic

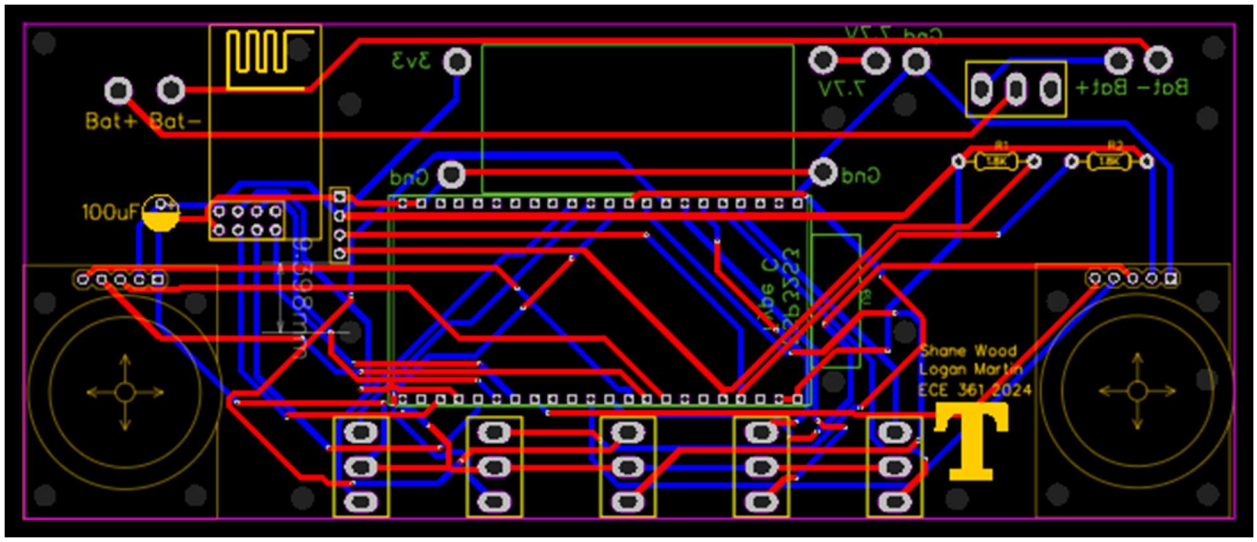


PCBs

Robot PCB



Controller PCB



Power

Power for the robot is managed by a 12V lithium-ion battery power pack capable of delivering three amps. This allows for the motors to run at their specified voltage leading to them spinning at their expected RPM. Immediately following the power pack is a switch, which allows for easy access to kill power to the robot. Following this there is a low voltage cutoff module, this prevents oversharing the battery which ensures the lifespan of the battery to be longer. From this point the motor drivers are powered and so is a buck converter. This buck converter drops the 12 V into 5 V for the distance sensors, microcontroller, and display. The 3.3 V line is fairly unloaded, therefore the built in 3.3 V voltage regulator on the microcontroller is used to generate this voltage for the rest of the devices.

Microcontroller

The microcontroller used in this project is an ESP32-S3. This is a mid-range controller with an excess of GPIO pins and has a maximum clock speed of 240 Mhz. The microcontroller handles processing of wireless signals, measuring distances, reading line sensor inputs, managing the I2C bus, creating PWM signals and all other functions of the robot.

Wireless

The wireless module used in this project is NRF2401. This communicates with the ESP32-S3 using SPI (serial peripheral interface).

Camera

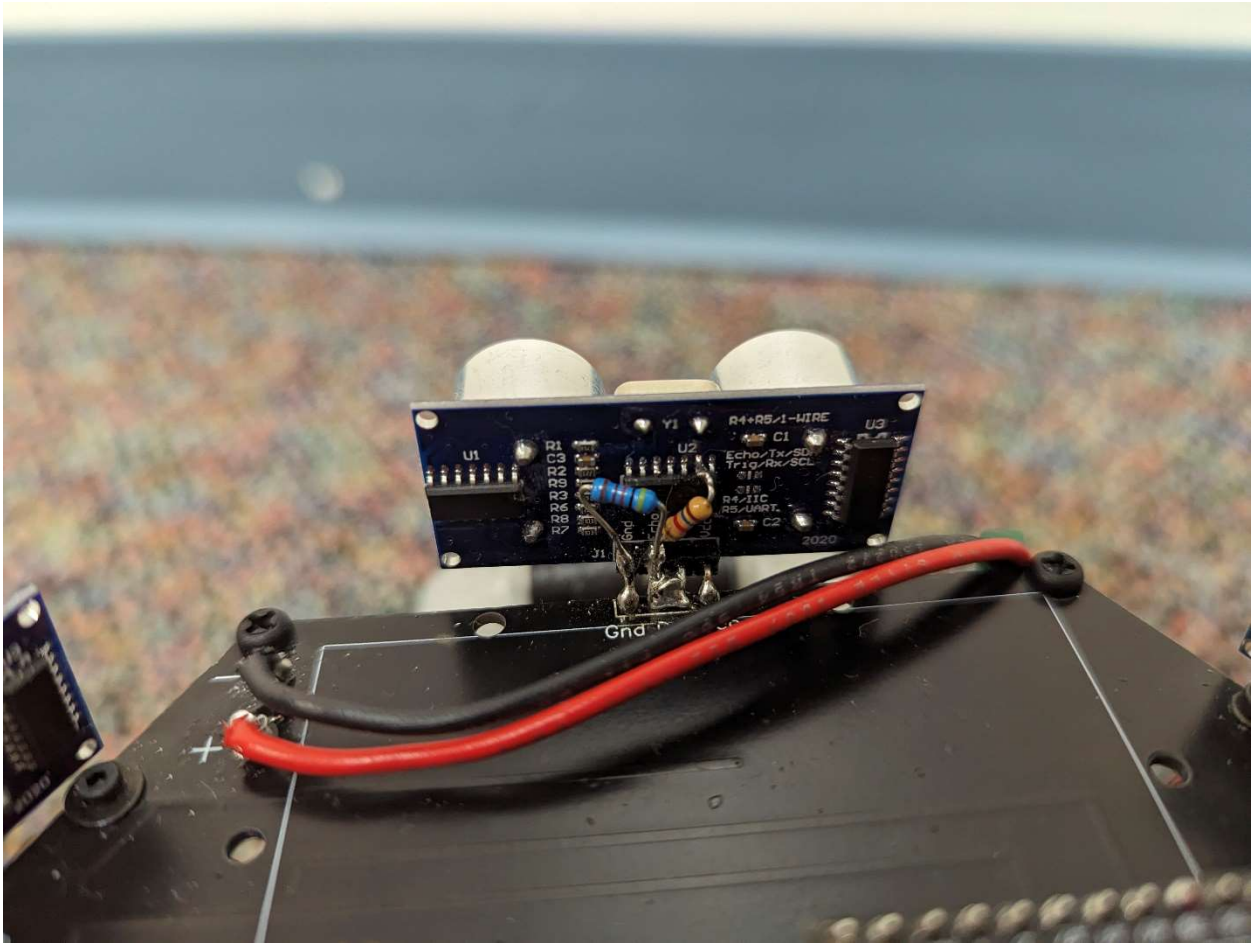
The robot also features a WIFI camera so that the user can control the robot without having to see it. This was done by using example code for the camera module. It creates a WIFI access point that once connected and navigating to the gateway the camera server can be started.

Sensors

3 types of sensors exist on this robot.

Distance Sensors (HC-SR04)

This robot features 6 distance sensors that when used together, allow it to follow walls. These sensors are 5 V components, meaning to use them with the 3.3 V microcontroller, the voltage needed to be dropped. To do this a voltage divider was added on each distance sensor by cutting the echo trace on the PCB and adding the two necessary resistors from the echo pin on the surface mount part to the pin on the module. Only this trace needed to be cut, as this is the trace sent as an input to the microcontroller, 5 V. To reduce the amount of GPIO pins required for these sensors trigger and Echo can be tied together. This can be done as once the microcontroller outputs a pulse it can change the pin into input mode, allowing it to receive the pulse.



IR reflectivity

These are the sensors used for line following, they work by sending out IR light and seeing if the receiver picks it up. This works as different surfaces have different values for reflectivity. When the IR light hits the black line, the light is absorbed, allowing the module to detect the line. Note that this process works best when using a black line because black will absorb the maximum IR light.

BMP085

This device allows for the current temperature to be measured. This is crucial for the accurate function of the HC-SR04 as the speed of sound changes based on the temperature. This device allows for the distances measured to be more accurate rather than just using a placeholder value.

I2C

Two devices are connected to the I2C bus in this robot.

16x2 Character display

This display shows the current mode that the robot is in or other useful debugging information, such as the measured distances or the current state of the line following sensors.

BMP085

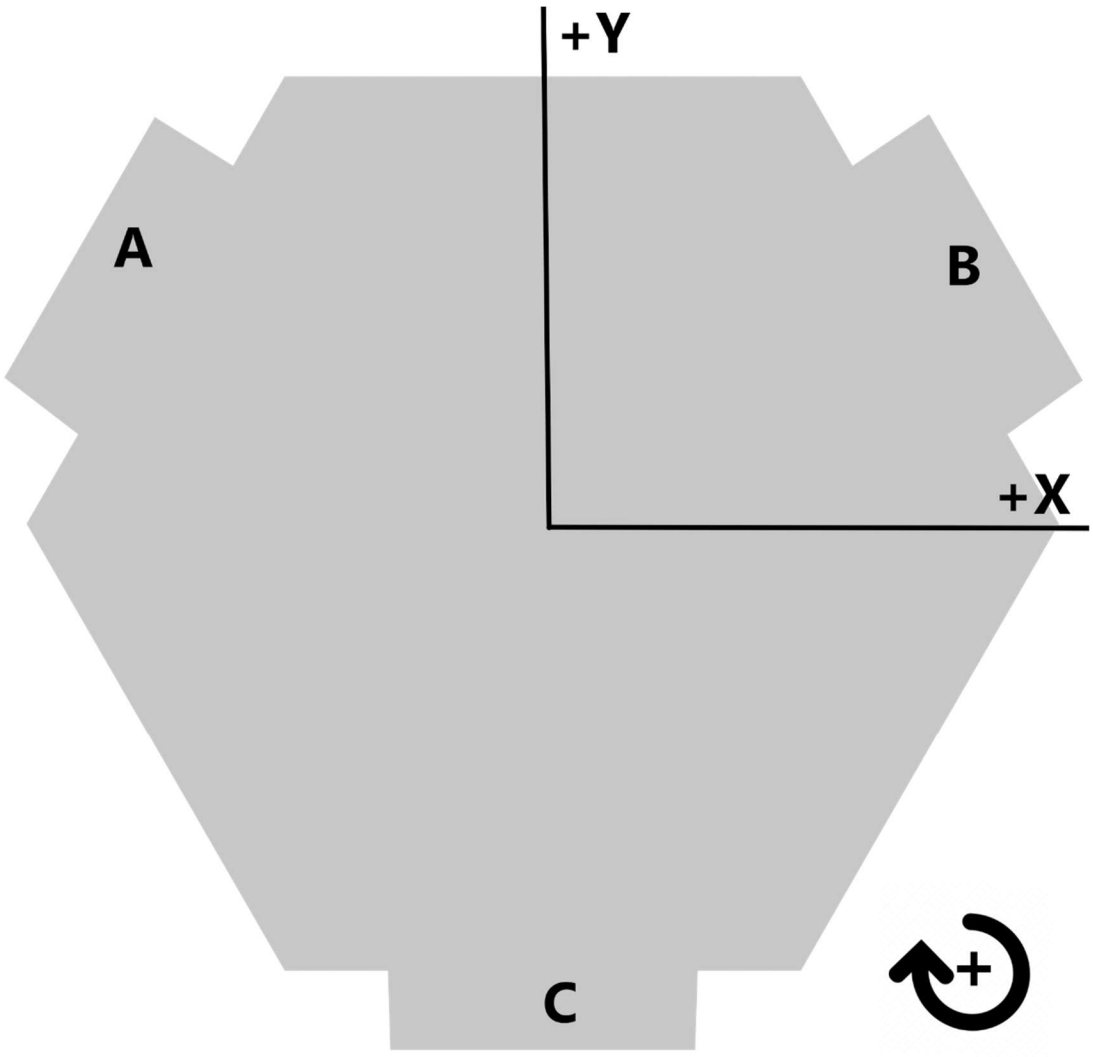
As mentioned in Sensors, this device measures the current temperature.

The Inverse Kinematics

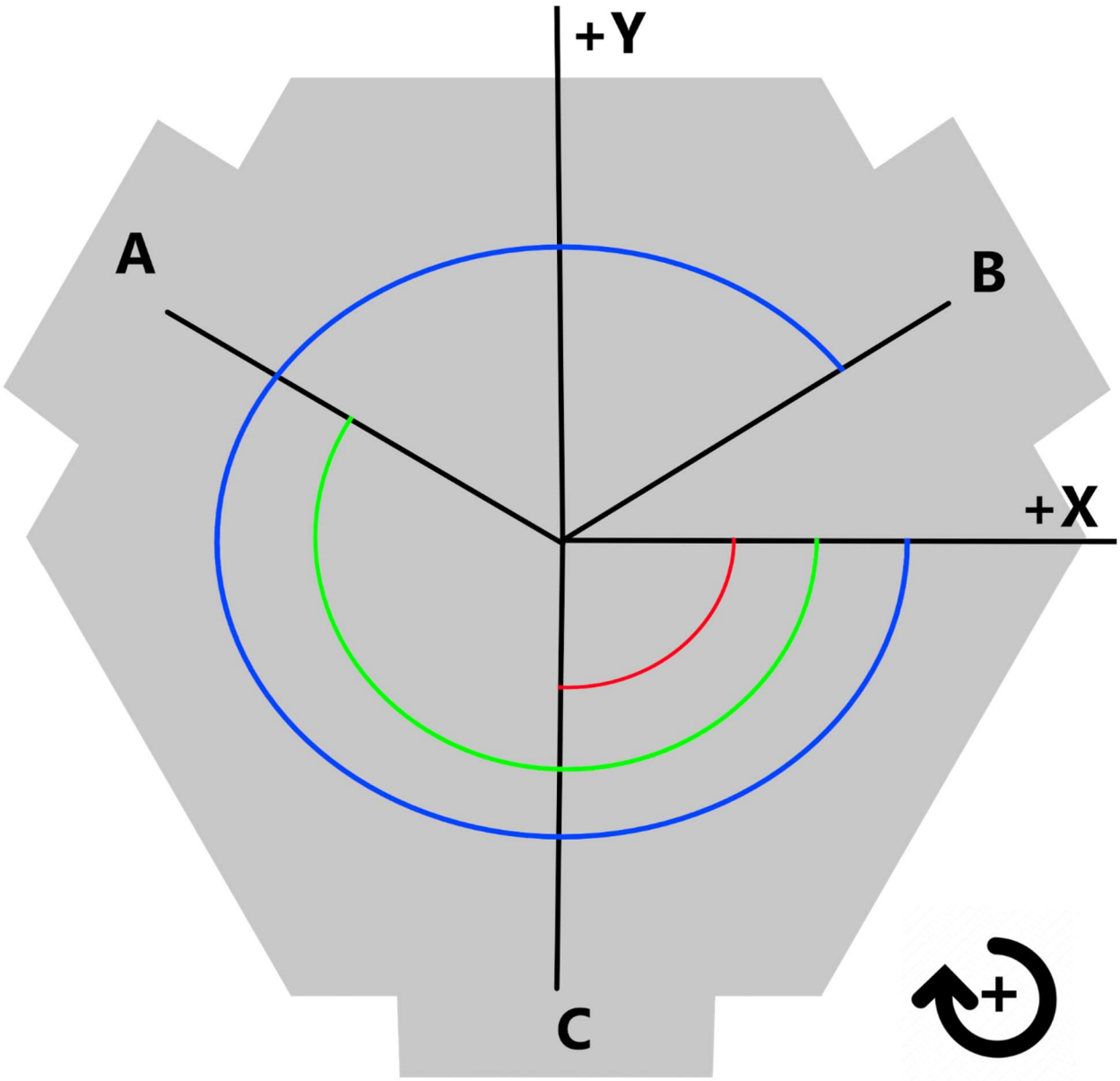
Moving

To ensure that the robot can move in any direction in a xy grid without turning the robot features an inverse kinematic model. This model takes the desired magnitude and direction of the robot and converts it into motor speeds and direction. It also handles spinning in place.

This was developed by setting a positive reference axis for the motors in 2D space.



Once this was done each motor was given an angle from the +X axis, which is treated as 0 degrees.



In this robot the angle offsets are:

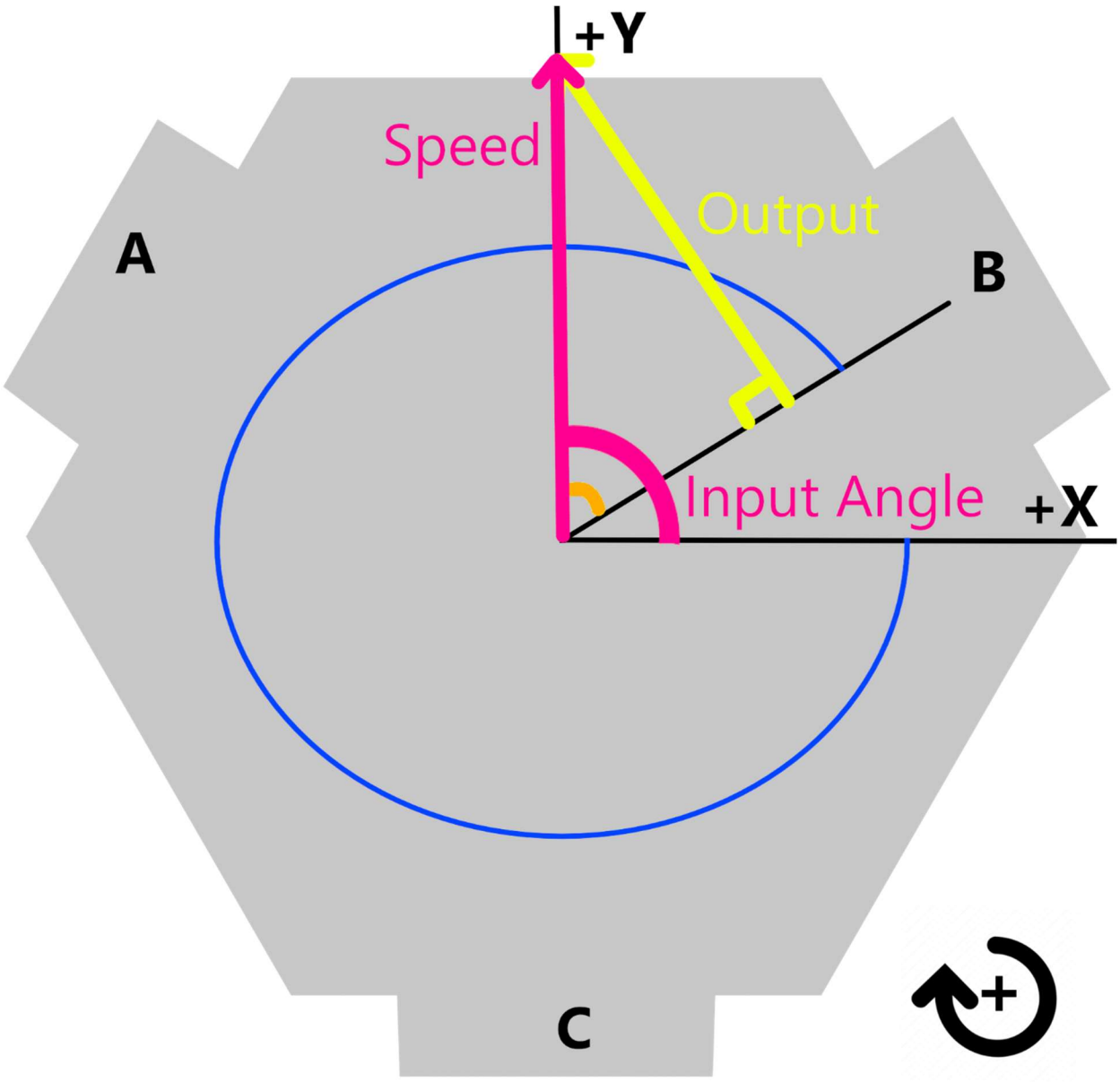
Motor A: 210 degrees

Motor B: 330 degrees

Motor C: 90 degrees

Now that each motor has its own offset, given any speed and input signal a triangle can be drawn between the motor and the desired direction.

Shown for example is the math for motor B when the robot is moving straight forward.



The pink is the input magnitude and direction, blue is motor b's offset and orange is the calculated angle.

The calculated angle is found using this formula:

Angle = offset – input direction.

This finds the angle +180 deg of its desired value, however due to properties of sine it will act as the same angle as the desired.

Simply by using sine and the angle the motor speed can be found by this formula:

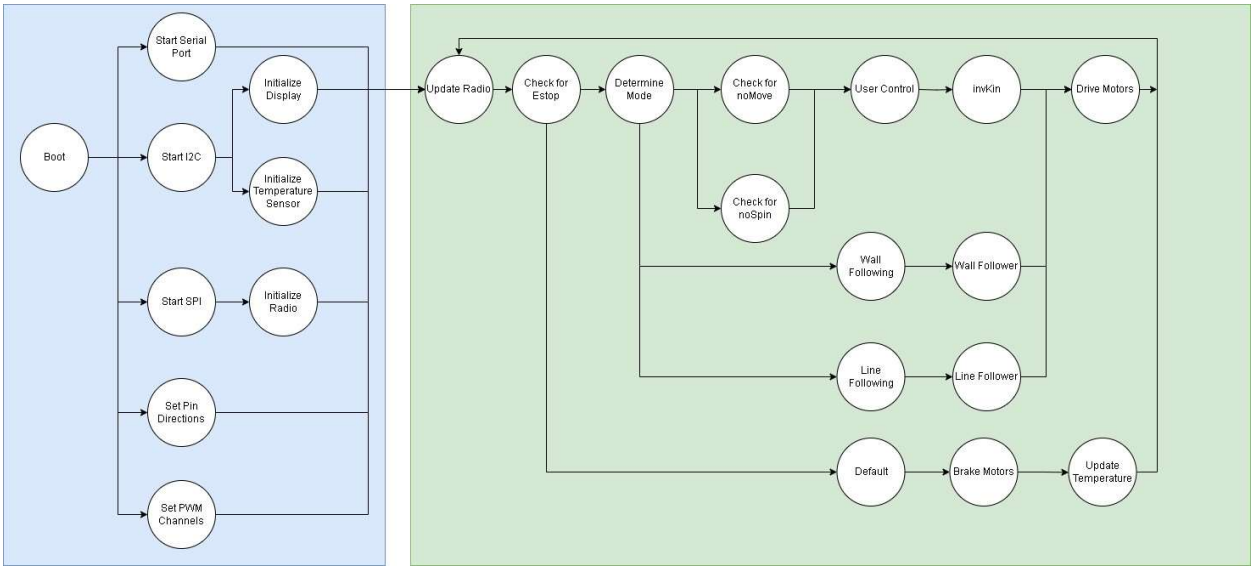
Motor Speed = Input Speed $\cdot \sin(\text{calculated angle})$

This same equation is applied to all three motors with the only change being the motor offset.

Spinning

The implementation for spin is basic due to this being a three wheeled robot. To spin in place all motors just need to spin the same direction. So, when a value for spin is received, all motor speeds are changed by the spin value allowing for the motor to spin in either direction.

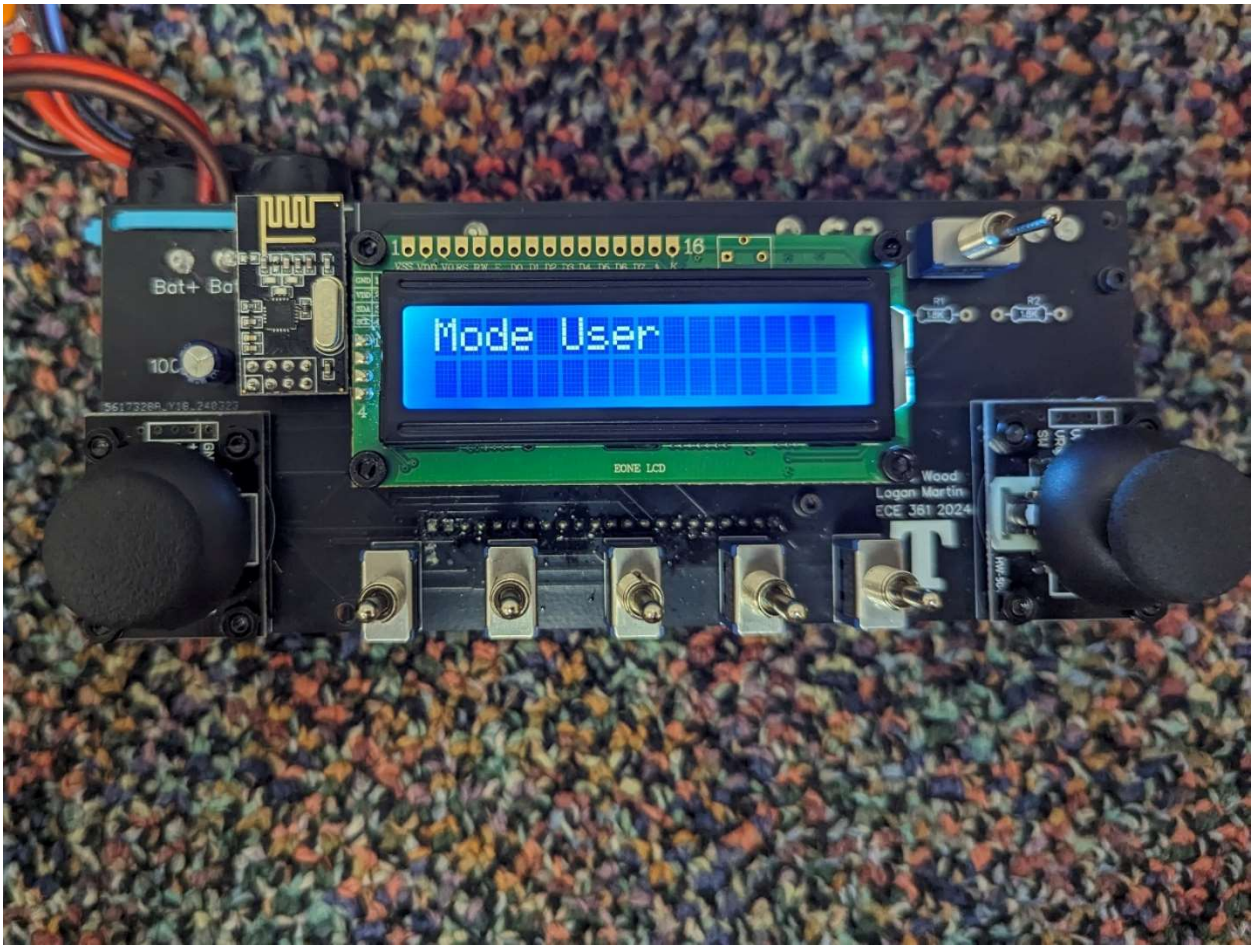
The Modes



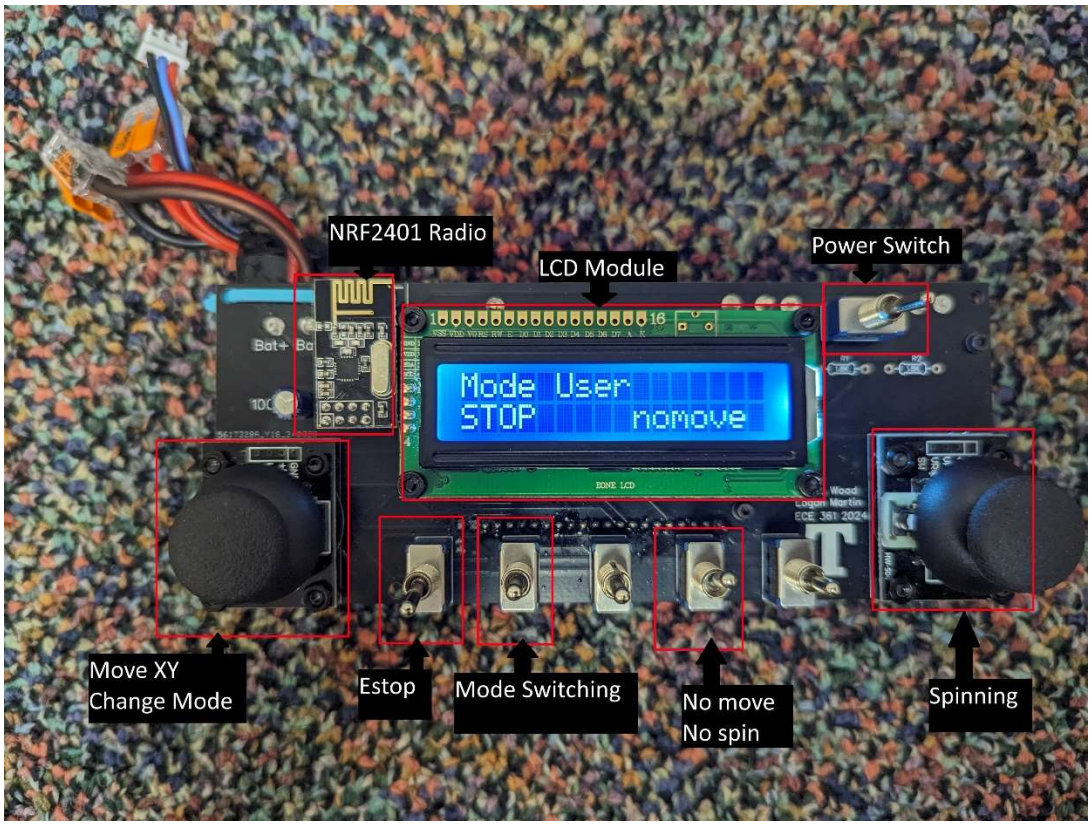
User Control

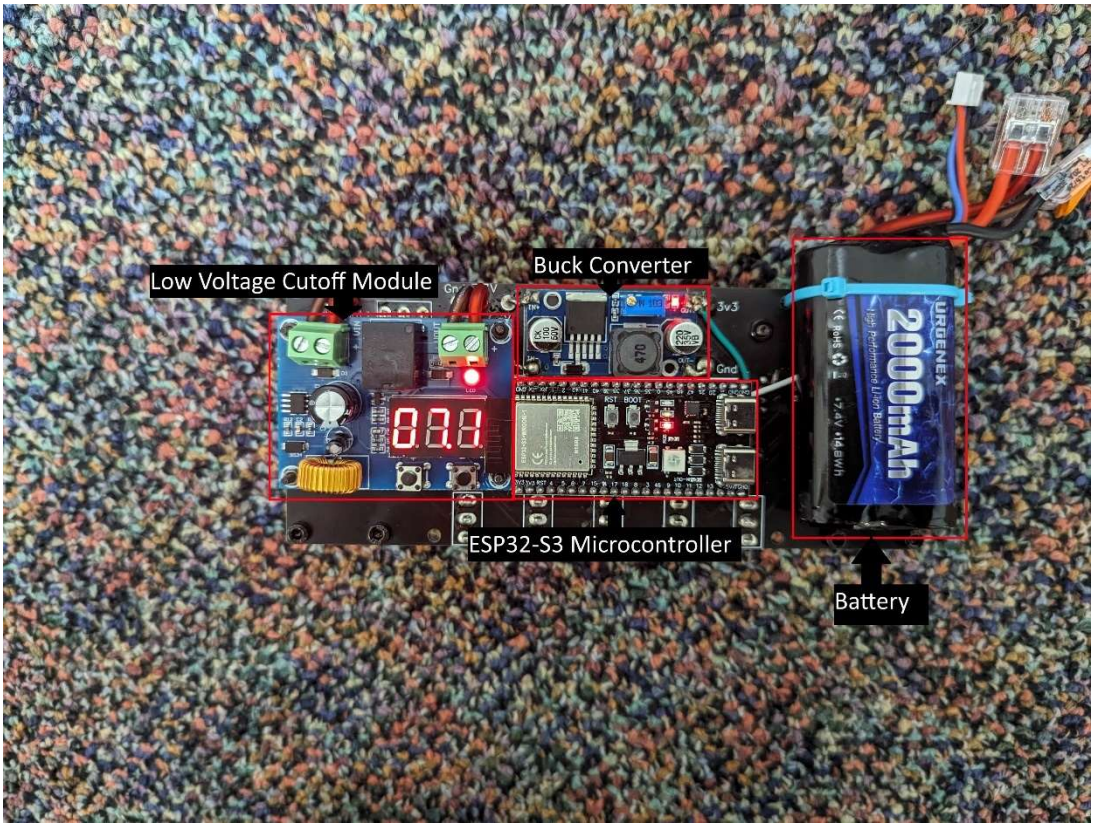
This robot features a user control mode where a person can drive it around using the inverse kinematic model. This is done using a controller that was also built as part of this project.

The Controller



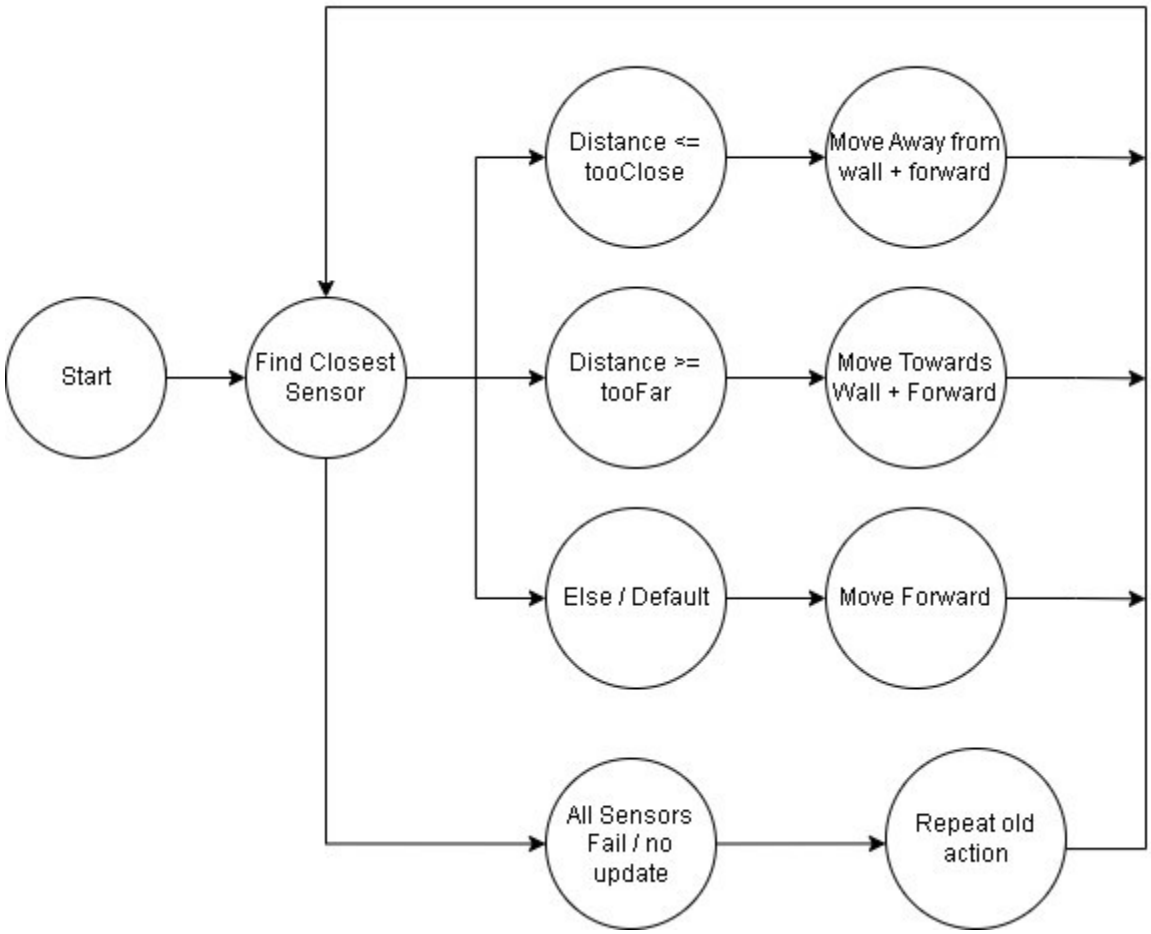
The controller has similar hardware to that of the robot itself. It is powered by an ESP32-S3 and uses an NRF2401 to communicate with the robot. It also has a display to show what mode the robot is currently in as well as showing any special settings, such as Estop and the disabling or spin or moving in user control mode. The functions of the joysticks and switches are shown below:


NRF2401 Radio
LCD Module
Power Switch
Mode User
STOP nomove
Move XY
Change Mode
Estop
Mode Switching
No move
No spin
Spinning


Low Voltage Cutoff Module
Buck Converter
3v3
Gnd
ESP32-S3 Microcontroller
2000mAh
Battery

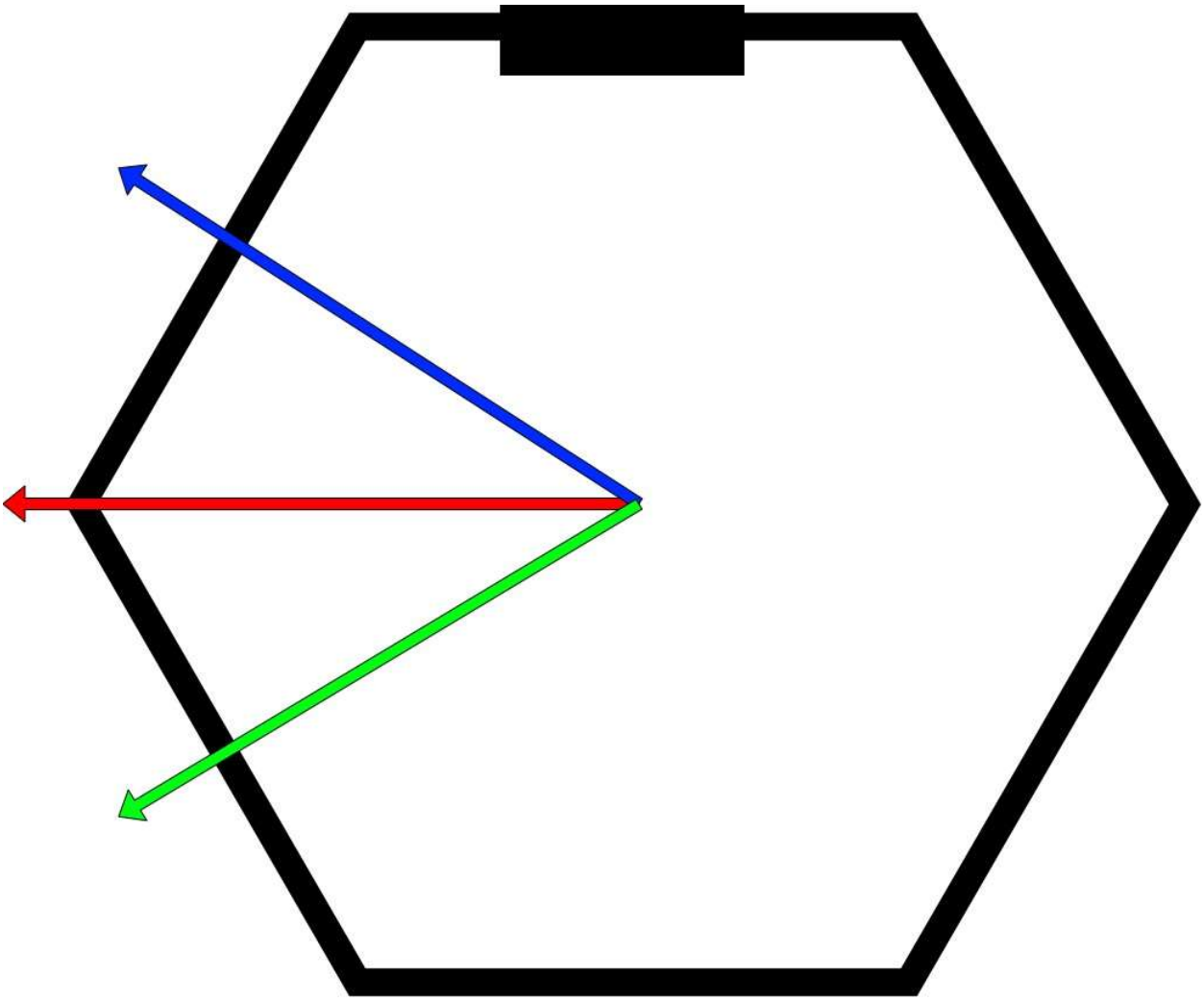
Wall Following

Wall Following uses the HC-SR04 ultrasonic distance sensors that are on each side of the robot to navigate around walls without running into them.



Due to the sensors not working at an angle more logic is required for it to function as intended. However, in testing this system seems mostly consistent, assuming the battery is near full charge. The motor speed slightly changes as the voltage drops, creating unpredictable behavior at lower voltages. One main issue is the robot can be stuck in between areas if it is equal distance between the 2 walls. This is further described in the design flaws section.

The angles that wall following makes the bot move per sensor were derived in the following manner.

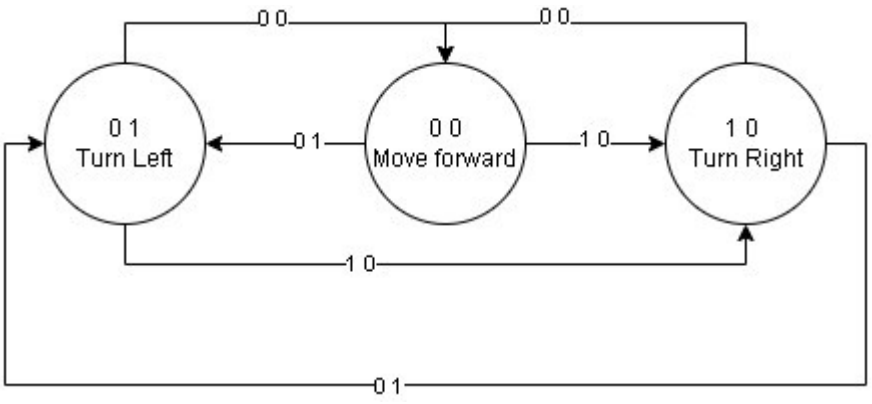


- The blue arrow is +60 degrees from the targeted sensor, this is used when the robot needs to move closer to the wall.
- The red arrow is +90 degrees from the targeted sensor, this is used when the robot is in between the too close and too far values. At red it is driving parallel with the wall.
- The green arrow is +120 degrees from the targeted sensor, this is used when the robot is too close to the wall, and it needs to move away.

All these angles are derived for all six sensors with the same offsets.

Line Following

Line following functions by existing in three different states, this can be shown as a finite state machine.



The two inputs are the left and right IR detectors. This creates a system where if the robot starts to cross the line it will turn into the line allowing the line to stay in-between the two sensors.

The states are the readings of the IR detectors, if the input remains 0 0 the robot with continue to move forward and likewise for the other two states.

The Code

This robot is built upon the Arduino framework giving easy access to thousands of libraries.

Libraries

- The nRF24L01 and RF24.h libraries work together to enable the NRF2401 to be accessed at a higher level. This allows for a structure to be sent in between the robot and the controller.
- The math library is used due to it having trigonometry functions, which are needed for the inverse kinematic model.
- The Adafruit_BMP085 library is used to communicate with the temperature sensor.
- The LiquidCrystal_I2C library is used to write to the display.
- Esp_camera library is used to set up the camera.

Functions

- getData() the function that updates the payload structure instance, only updates if the radio has new data available.
- InvKin() the function that modifies the three speed values for the motors given the speed, direction, and spin.
- Decrad() converts degrees to radians.
- LineFollowing() the function that handles line following mode, returns a MoveValues structure so that the robot can drive and spin.

Structures

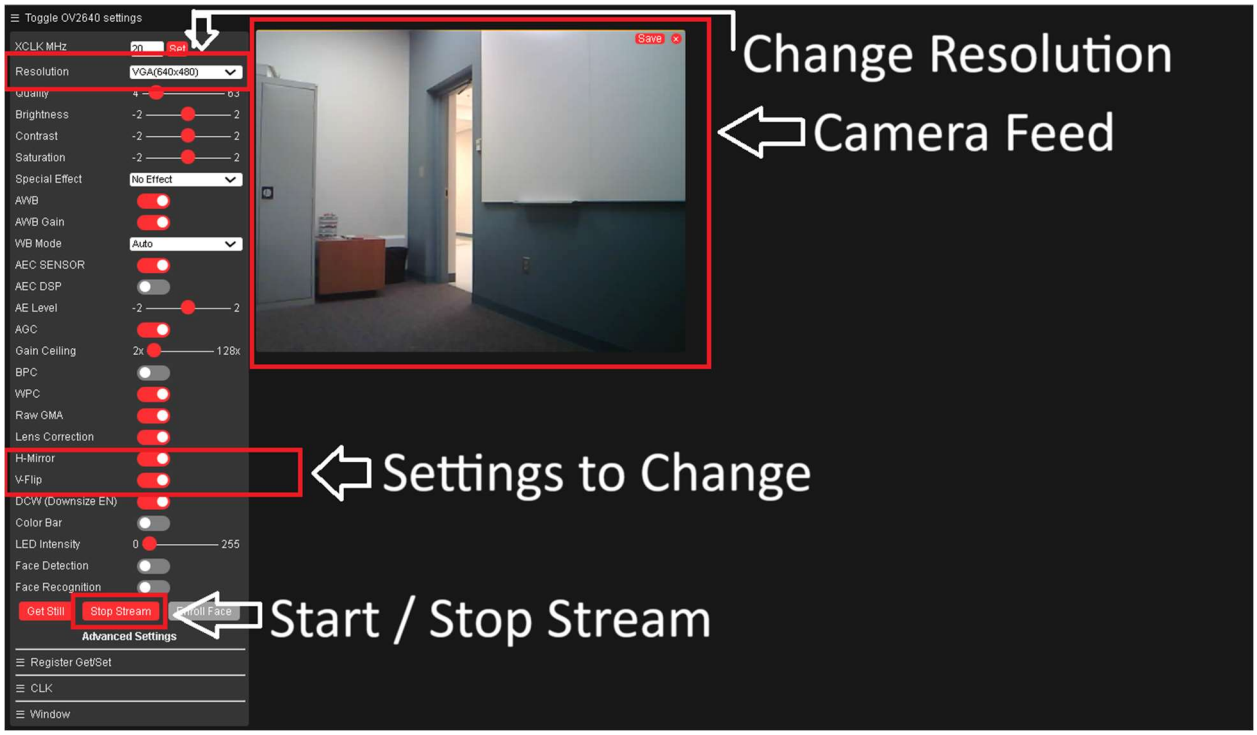
- PayloadStruct handles the variables that are transmitted using the NRD2401, including mode, speed, and direction as well as additional values.
- MoveValues handles the magnitude, speed, and spin values in between the non-user control modes on the robot, allows a function to return all the needed values for the inverse kinematics at once.

Classes

- Motor class handles the control of the motors, this includes speed and direction control by only passing a single ~+- 255 speed value. It also implements braking in cases when the motor should not be moving.
- DistanceSensor class handles the control of a single distance sensor, this class is basic and only gets the measured distance and allows for the updating of the speed of sound.
- Distances class handles the array of distance sensors. It stores the last measured values in array and allows for all six sensors to be updated with a single function. It also writes the six distances to the display, allowing the user to ensure the sensors work as expected. Temperature updating and wall following are also handled.

Accessing the Camera Webserver

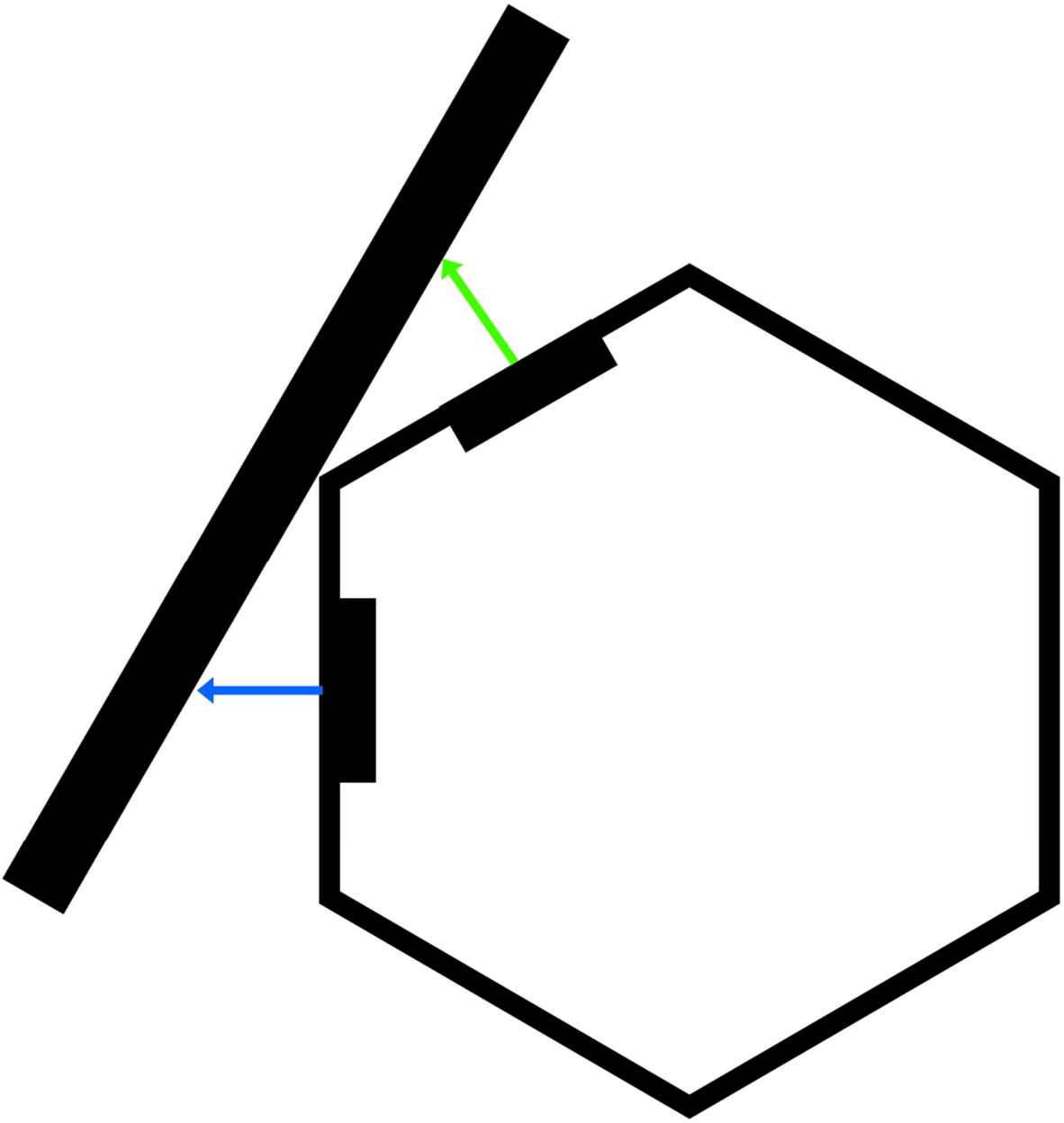
When the robot is being powered the ESP32-S3 Sense board also powers up. When powered the power creates a WIFI access point called “GilberttheSecond” with the password being “ShaneandLogan”. Once connected, one can navigate to the gateway address, http://192.168.4.1/. Once here many options exist for the camera. The only settings recommended to change are the resolution to HVGA (480 x 320), H-mirror and V-flip. Then the start stream button can be pressed, allowing the video to be streamed over 2.4 Ghz.



Potential Improvements and Design Flaws

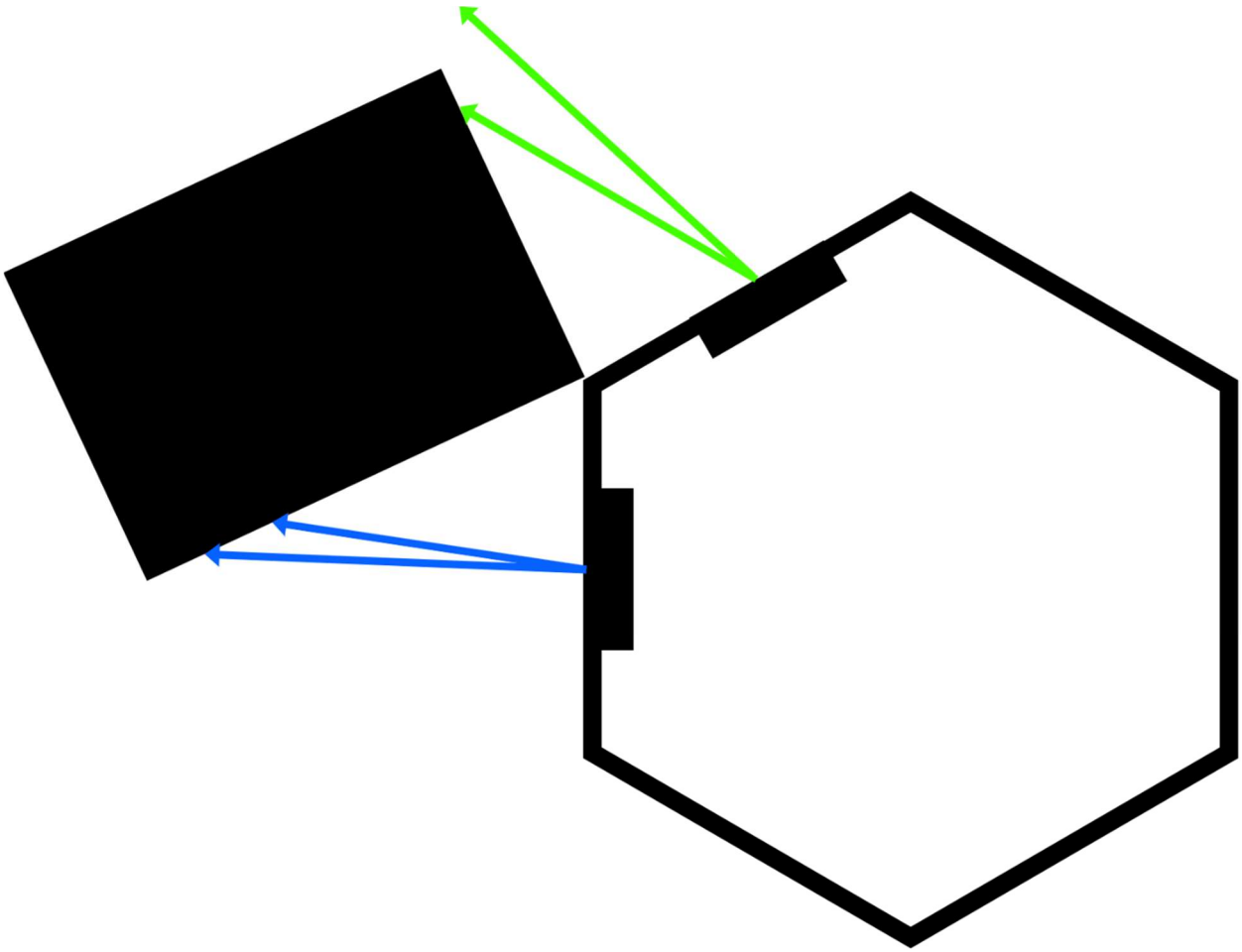
Distance Sensors

The main flaw with the distance sensors chosen involves how they function. By sending out a sound pulse and receiving it back. This works well when the wall is parallel to the sensor, however in cases where it is not, which occurs often in wall following, the values are occasionally never received. This made wall following far more complex and, despite trying to catch all possibilities something the robot will fail to follow a wall or fail to detect, even though it is right next to the wall. The 30 degrees of detection that the HC-SR04's have is not enough for the rest of the robot. Switching to a spinning lidar module would resolve this due to the spinning nature of the modules as it ensures that the sent beam will always hit something, assuming something is in range, as it can line up will the wall. The robot can get temporarily get stuck in the positions as shown.



In this case the robot was following a wall and in some way got too close to the wall, this should not normally occur as it should remain the fixed distance away from the wall, but it can occur sometimes. The two measured values return the same, similar, or alternating values, causing the robot to constantly switch between sensors and not make progress This error has been mainly dealt with given the present code.

The second way the robot gets stuck is much harder to deal with. This is mainly because it is also the same time when the robot is switching what sensor. Similarly, two sensors also start reading the same value in this case.



This is in part due to the thrity degree field of view of the HC-SR04 sensors. If the robot gets stuck here it will not know which sensor is closer preventing it from escaping this corner.

PCB issues / GPIO Issues

On the original PCB the HC-SR04's and the display were both intended to be powered by 3.3 V. However, upon attempting to power these from 3.3 V neither worked. The display would turn on, but the contrast was incorrect leading to it being unreadable. The HC-SR04's would also not work correctly on 3.3 V. Upon switching these to 5 V the HC-SR04's had to have a voltage divider installed. Had this been caught prior to ordering the PCBS less wires would have had to been run outside of the PCB. Later during the project some GPIO pins stopped working, and those traces also had to be run to different pins on the ESP32-S3. The reason for this is unknown, although it can be assumed that something went wrong with the distance sensor, and it ended up sending 5 V to a 3.3 V pin, damaging it.

Wheel / Motor issues

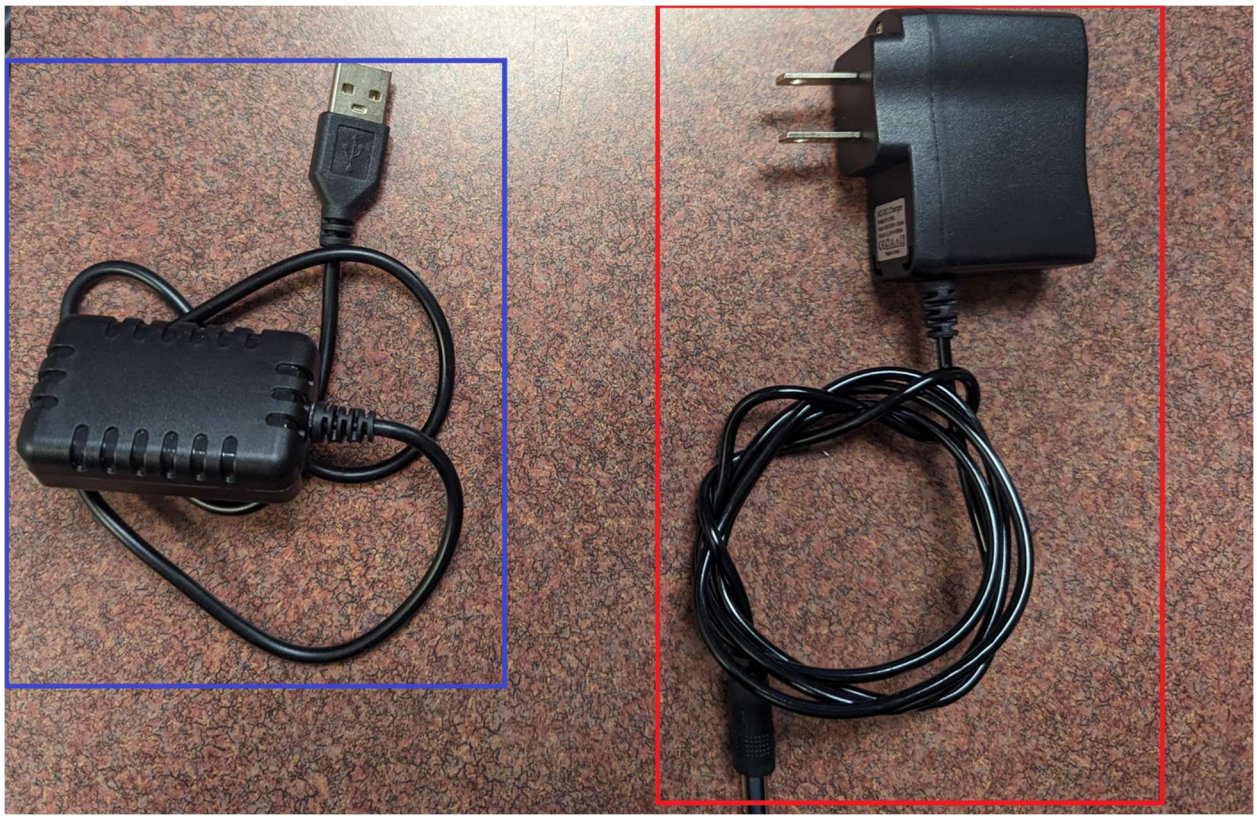
As described earlier, the wheel's limited traction causes issues when moving in some ways. This occurs when all three wheels are active as some, despite being given an identical PWM signal, will still spin at a lower speed. This is a major issue with the motors as despite being the same motor they do not start spinning at the same frequency. This can be slightly corrected in code by using scalers on the motor's speed which was done in this case.

Using the Robot

This section details how one would go about using the bot.

Charging the Batteries

The Chargers



The charger in the blue box is for the controller, The charger in the red box is for the robot.

Controller

A USB charger charges the controller battery. It does not need to be unplugged from the controller to charge, although it is recommended that the controller is off during charging. The battery is connected to the charger through the exposed JST plug near Bat+ and Bat – on the top left corner of the PCB. Once charging the light on the charger will become solid green and then can be unplugged.

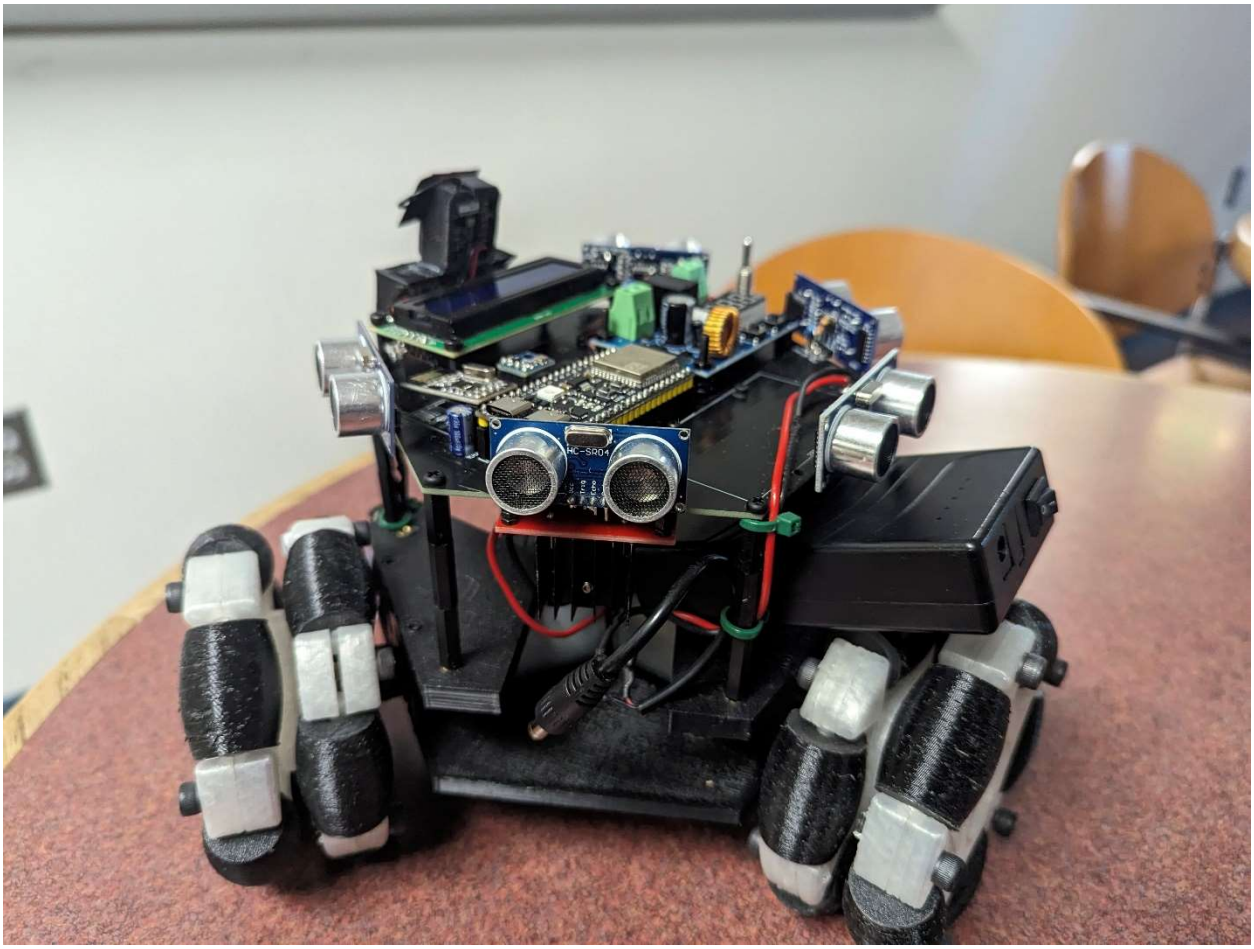
Robot

The robot battery is charged by a 12 V DC supply. This battery does need to be turned off using the switch on the battery itself. When off the lights on the battery will no longer be turned on. This battery must be unplugged as the battery charges and is drained through the same barrel connector. Once charged the LED on the charger will go from red to green.

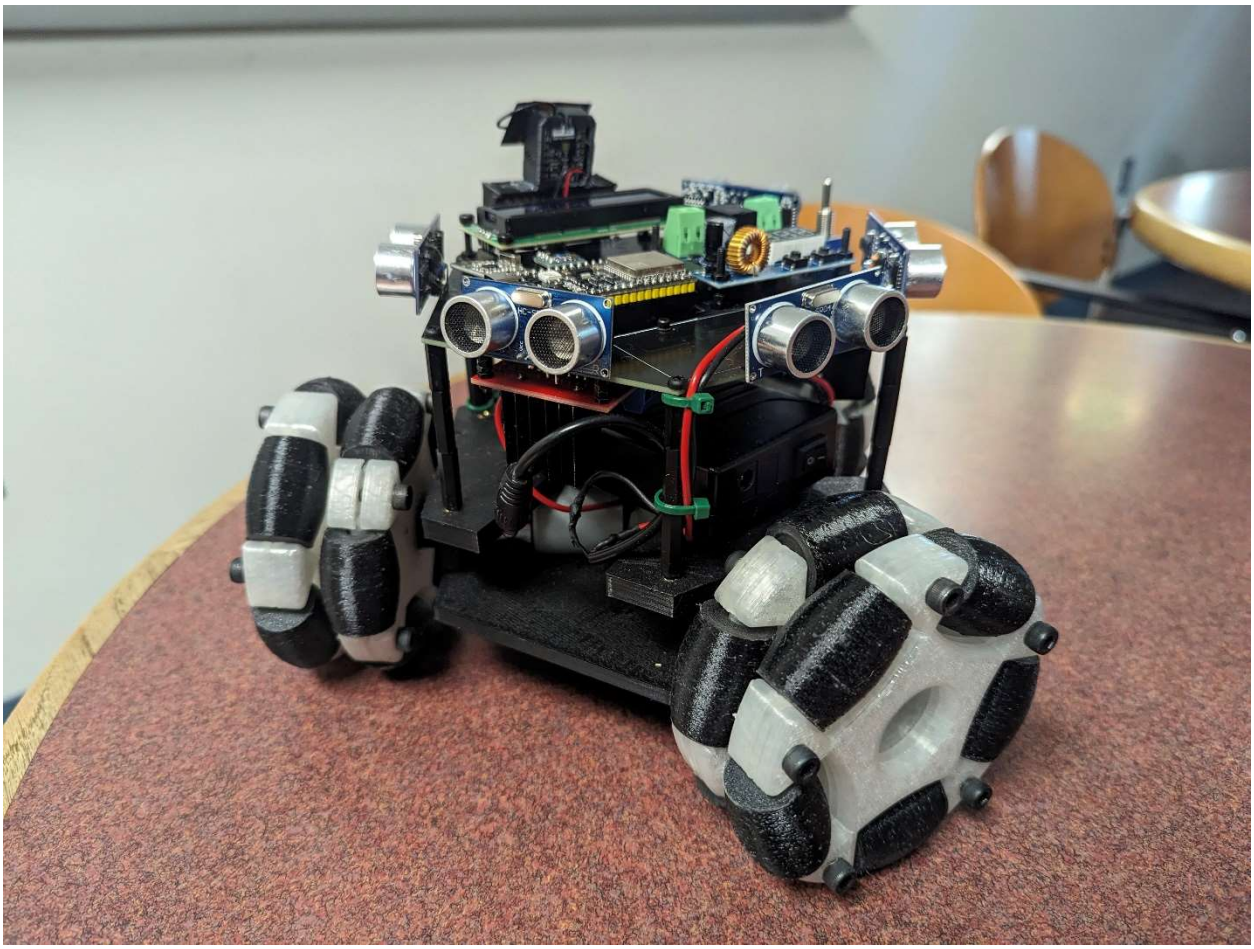
Inserting the Batteries

The controller battery is fixed in place and should not be removed.

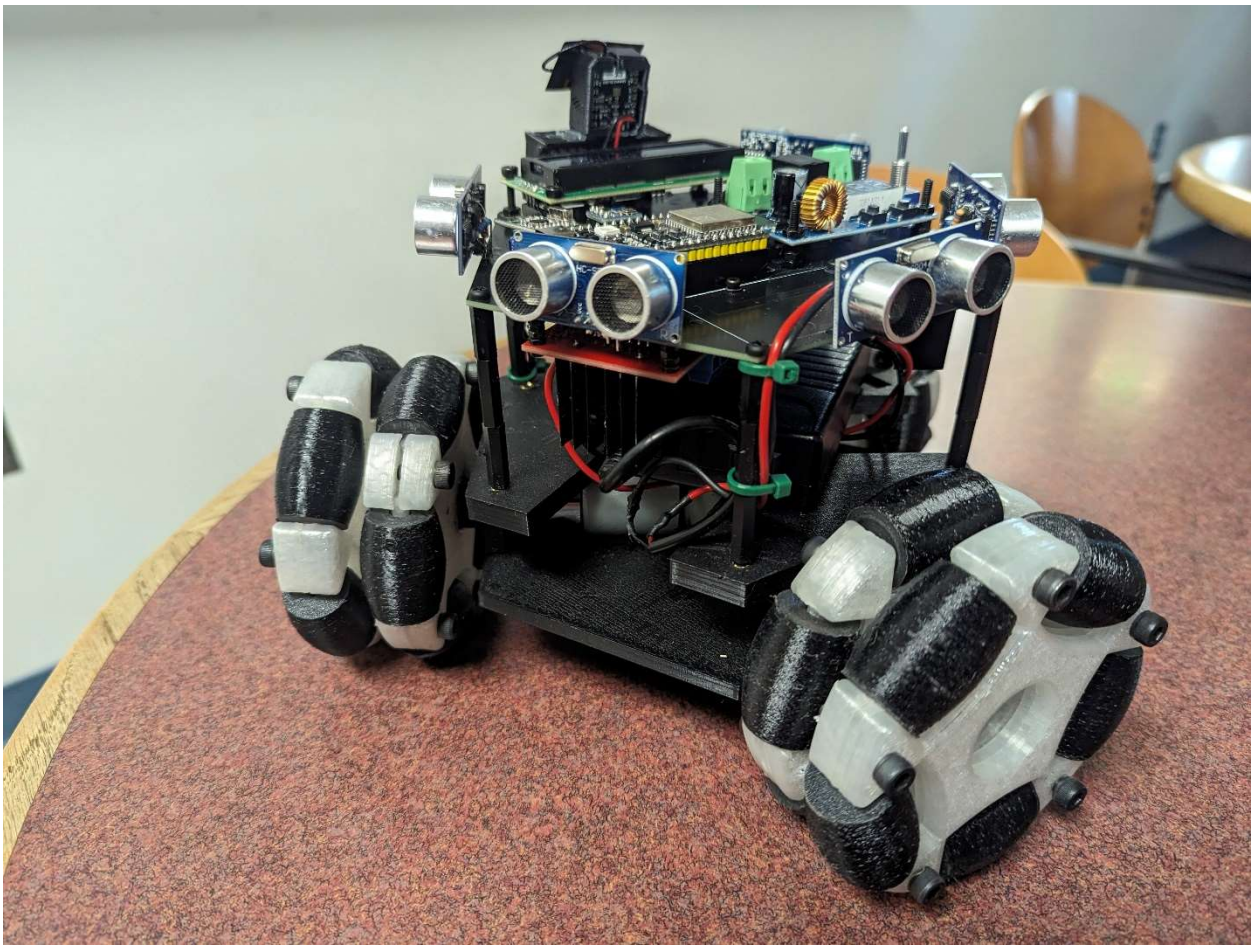
The robot's battery can be inserted as shown in the following instructions.



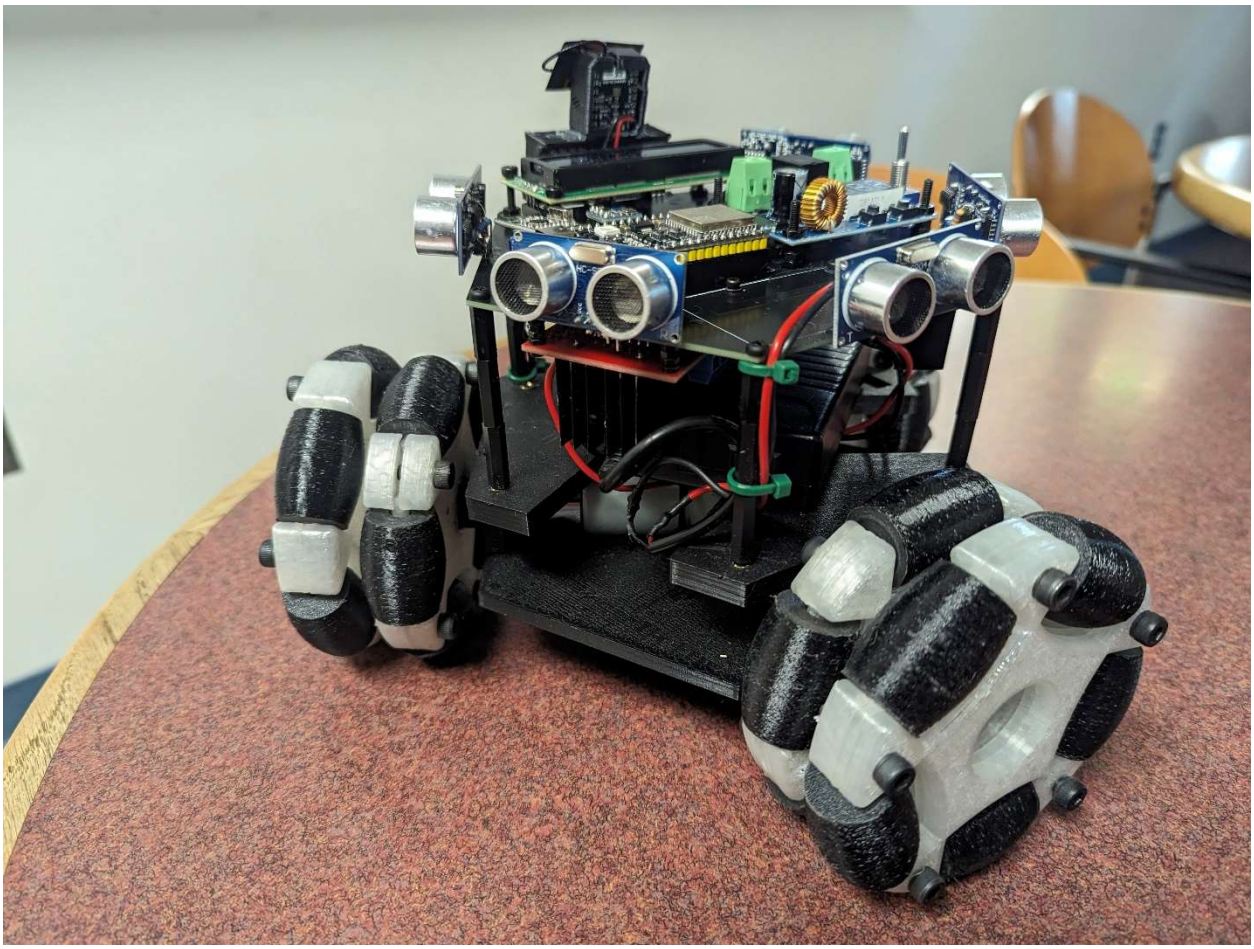
1. Insert the battery above motor C's wheel at an angle with the switch and charging port facing outwards.



2. Fully insert the battery so that it is flush with the top 3D printed piece.



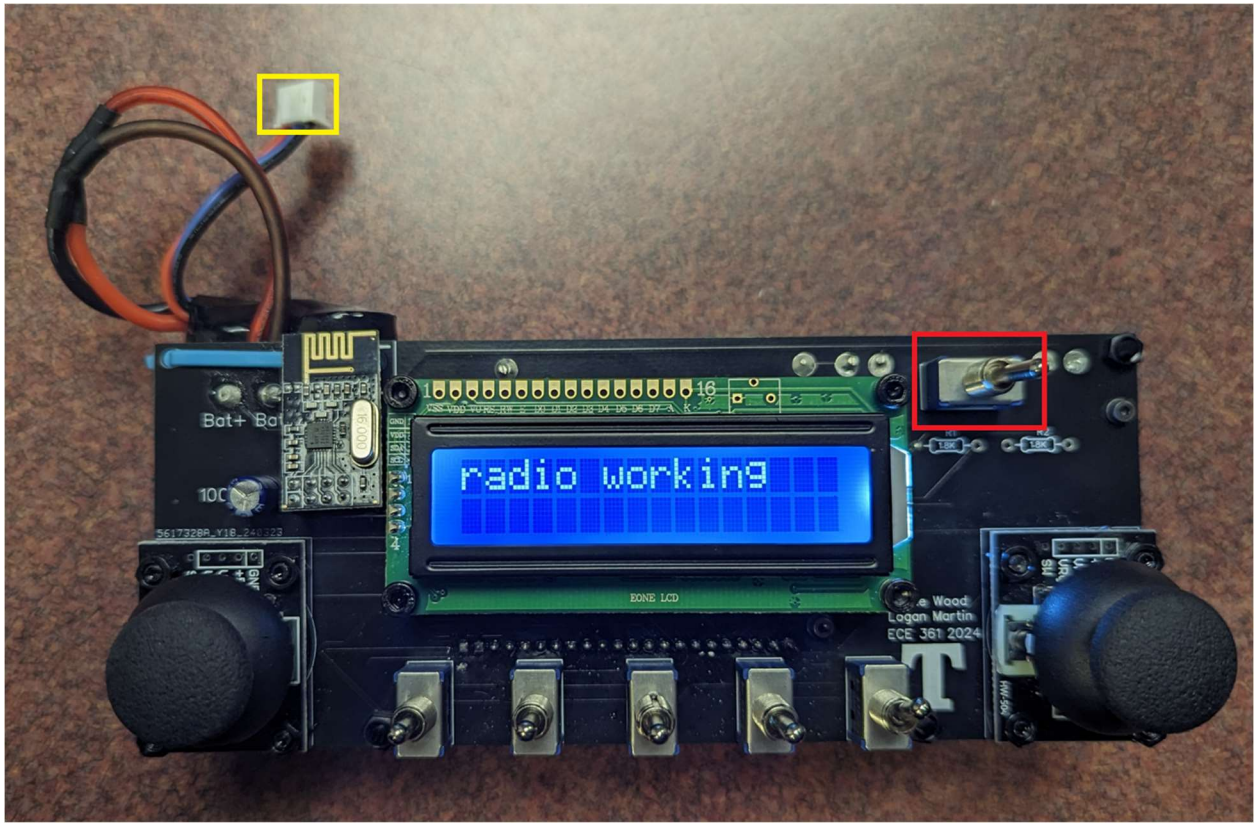
3. Rotate the battery 60 degrees counterclockwise so that the edge of the battery is parallel to distance sensor D3.



4. Insert the barrel jack into the battery and flip the switch on the battery to the on position.

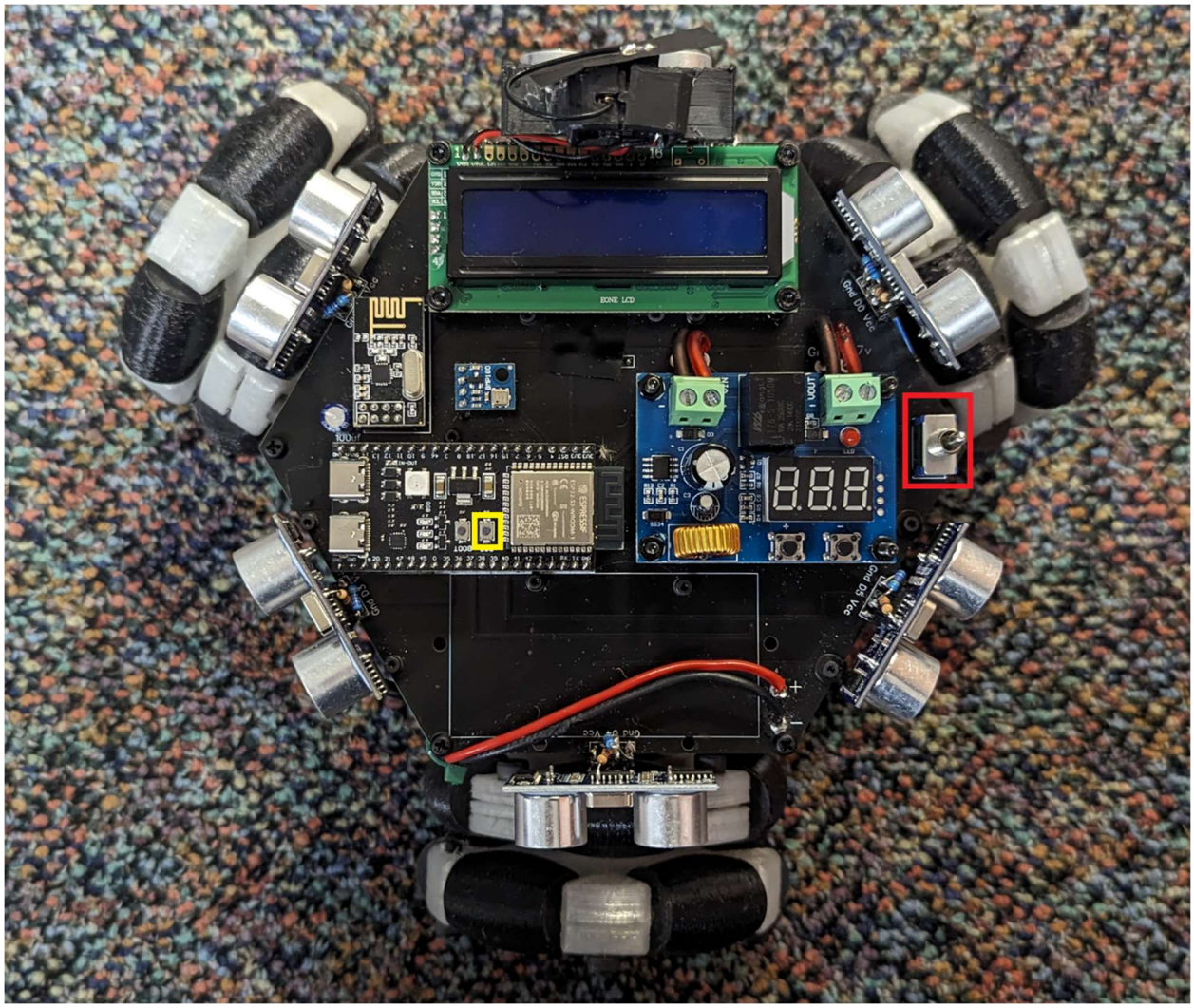
Turning on the Devices

The Controller



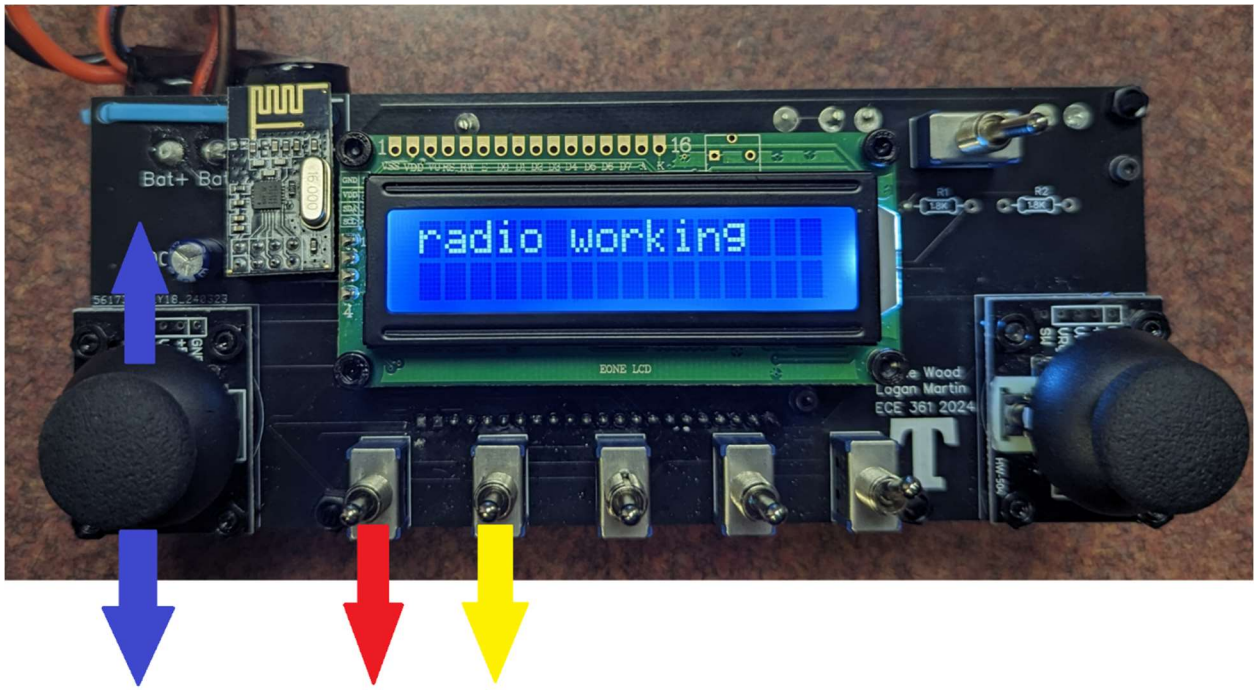
The switch in the red box is the power switch, the connector in the yellow box is the JST connector described above. Once the switch is pushed to the right the controller will turn on assuming the battery is charged. At this point, assuming everything is working the display should read “radio working”. At this point, any switch mapped to a function can now be pressed see the dashboard view.

The Robot



The switch in the red box is the power switch, to turn it on the switch needs to be in the down position. Due to some delays in the microcontroller being powered the reset button sometimes needs to be pushed, this is indicated by the text only displaying white rectangles on the first row, this is the button with the yellow box around it on the microcontroller. Like the controller, the screen should also display “working”. When the robot is switched to an autonomous mode it will update to show the distances from the HC-SR04’s or the readings from the IR sensors. Once switched back to user control the screen will stop updating until it is switched back to one of these modes.

Switching Modes



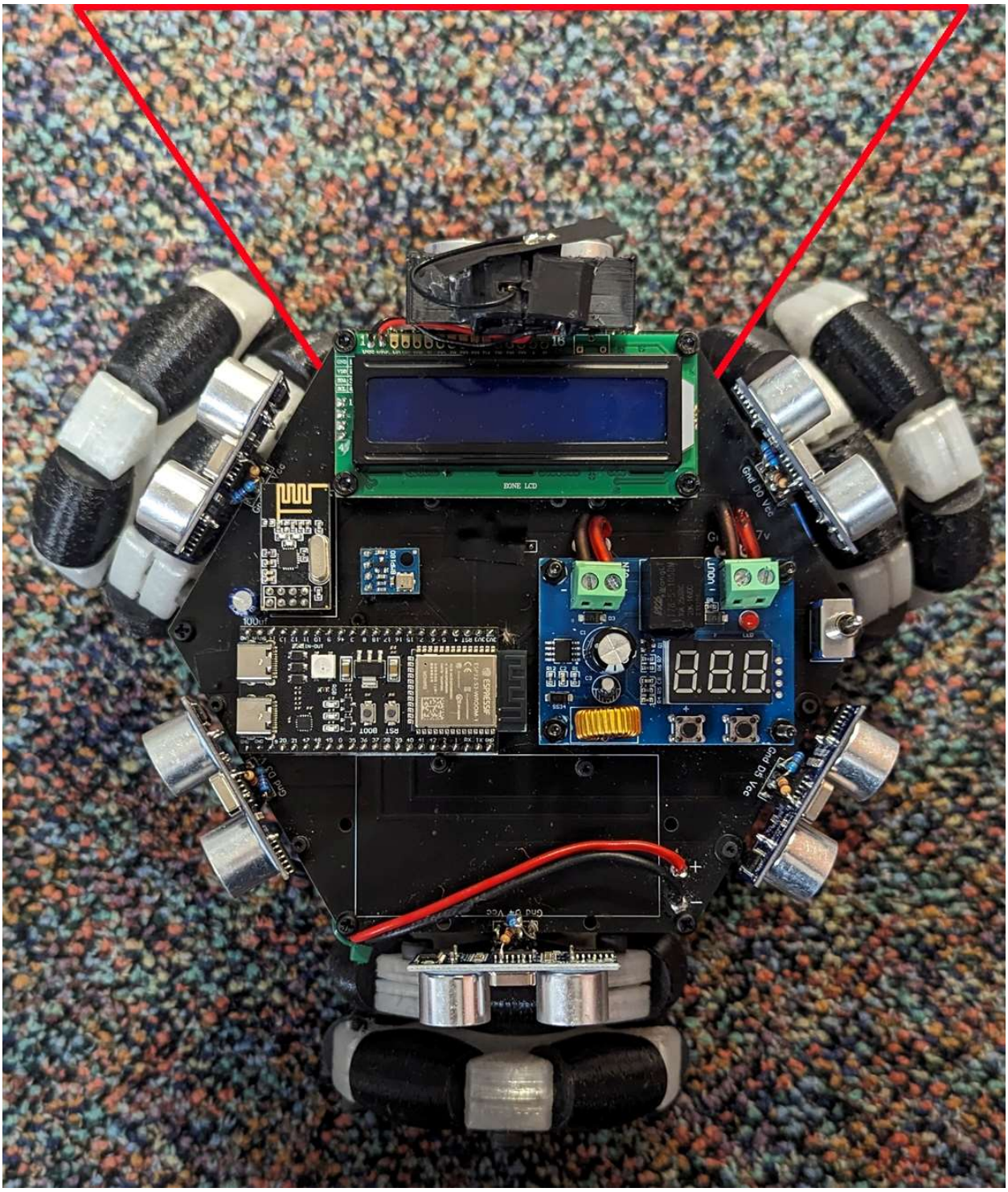
To switch modes on the controller the second (yellow) switch needs to be put into the down selection. It is also recommended that the first (red) switch is also put into this position so that values are no longer sent to the robot. Once this is done the left joystick can cycle through the available modes when moved vertically (blue arrows). To confirm a mode, the second switch needs to be put back to the neutral stage. The first switch can now be put back to neutral as well since the mode is now confirmed, allowing the robot to function once more.

Setting up Autonomous Modes

Wall Following

To follow a wall, the robot needs to be close to a wall, this can be done by any mode, such as the bot following a line to a wall. Once close to a wall, and in the correct mode, the robot will start following it. Note that the robot only follows walls counterclockwise, meaning that if turned around, it will continue to drive the same way despite physically facing the other direction.

Line Following



To follow a line, the robot needs a line on the ground, any surface with contrasting reflectivity will work, but it is currently tuned to follow black lines on white paper. This can be adjusted with the two potentiometers on the IR modules. The line needs to be approximately 13 mm (about 0.51 in) and angles >+-50 degrees from the robot's +90-degree mark as shown, are less

likely to work. A working line is attached to the end of this document and is in the project's GitHub.

GitHub Links:

https://github.com/Shane-Wood-TL/Hexagon_Robot_ECE361/tree/main

Code Direct Links

Robot

Main file: https://github.com/Shane-Wood-TL/Hexagon_Robot_ECE361/blob/main/Code/Robot/HexagonRobot/src/main.cpp

Constants file: https://github.com/Shane-Wood-TL/Hexagon_Robot_ECE361/blob/main/Code/Robot/HexagonRobot/include/constants.h

Controller

Main file: https://github.com/Shane-Wood-TL/Hexagon_Robot_ECE361/blob/main/Code/Controller/HexagonRobotController/src/main.cpp

Menu file: https://github.com/Shane-Wood-TL/Hexagon_Robot_ECE361/blob/main/Code/Controller/HexagonRobotController/src/menus.cpp

Constant file: https://github.com/Shane-Wood-TL/Hexagon_Robot_ECE361/blob/main/Code/Controller/HexagonRobotController/include/constants.h