MEAM5100 Final Report - Team 24

Anthony Saldutti, Christopher Chong, Sachin Pullil

December 29, 2022

1 Introduction

1.1 Overall functionality

For the final project, our goal was to build an autonomous mobile robot that could follow a wall, track and grab a beacon, and localize itself and nearby objects. Based on the project specification instructions, we employed the ESP32 family of microprocessors for edge computing and HTC Vive for localization. We used direct current (DC) motors to drive the wheels and infrared (IR) phototransistors to detect the beacons.

In terms of our team's unique design choices, we chose to create a holonomic mobile base using mecanum wheels to attain velocity in any direction. We also added a Light Detection and Ranging (LIDAR) sensor, ultrasonic distance sensors, and an IR distance sensor to detect the wall, obstacles, and the beacon once grasped. We employed two servo motors that actuated two claws which could capture the beacon once it was in close range. Since there were multiple elements to the project and to facilitate parallel work, we decided to split functionality between three different ESP32 microprocessors - a low-level C3 for motor control, another low-level C3 for Vive localization, and a high-level S2 to perform logic calculations and interface with the user through WiFi. Finally, we powered the entire system with a Lithium Polymer (LiPo) battery and a power bank.

1.2 Detailed approach

1.2.1 Robot control

To control the robot, we used WiFi and HTML. The S2 microprocessor and our team's laptop was connected to the General Motors (GM) lab's router. The processor would then output a webpage designed in HTML, shown in figure 1 over the wifi. The page was displayed on a local IP address which we connected to from our browser. From there, we could command the robot to move with a certain velocity or perform a certain behavior.

Figure 1: HTML page that was used to control the robot from our team's computer.

The button click triggered internal logic on the high-level S 2 which decided what speeds to send to which of the four wheels and what time. It then communicated this information via an Inter-Integrated Circuit (I2C) serial communication protocol to the dedicated C3 driving the motors. The low-level C3 then generated pulse-width modulation (PWM) signals to drive the DC motors connected to the wheels.

1.2.2 Holonomic motion

We chose to equip our mobile robot with holonomic wheels to attain velocity in any direction in the XY plane. The type of wheel we chose was a mecanum wheel, which is a type of wheel with small rollers angled 45° to the wheel axis and therefore allows it to move with two degrees of freedom in the XY plane. The wheel we used is shown in figure 5.2.1.

Figure 2: A left mecanum wheel. Photo taken from Adafruit website [1]

To achieve holonomic motion using mecanum wheels, we placed four of them in an H configuration (similar to how wheels are arranged on a car) around our base and then supplied different velocities depending on the desired direction. A summary of this approach is shown in figure 3. A key point to note here is that mecanum wheels are not identical - the top left and bottom right wheels are the same, and the other two are the same. To make a holonomic robot, we ensured that the rollers all face inwards toward the center of the vehicle.

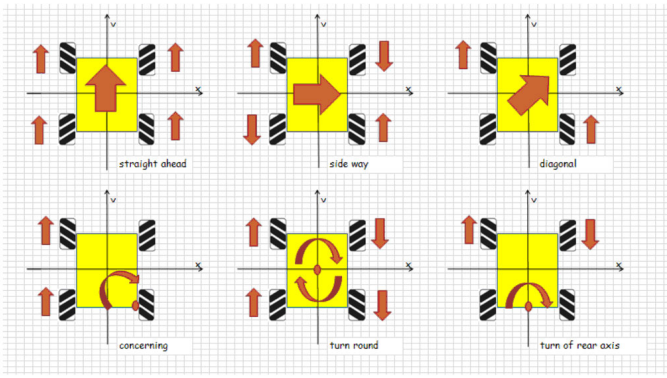


Figure 3: How to achieve different base velocities using different wheel velocities. Photo taken from RoboteQ website [2]

1.2.3 Collision avoidance and wall following

We initially envisioned using sensors on all four sides for collision avoidance and wall following (LiDAR Time of Flight in front and Ultrasonics on sides and back). However, due to excessive noise from the ultrasonics, and a conflict between the I2C connecting the S2 and C3 with the S2 and LiDAR, we ended up only using one ultrasonic to achieve wall following backwards.

1.2.4 IR beacon tracking

To allow the robot to track the beacons, we utilized two IR Sensors with their inputs amplified with reflective cones and their outputs amplified with op-amps. These outputs were fed into an Atmega32U4 which was programmed with frequency detection code that would send logic highs to the ESP32-S2 depending on which sensor could see the beacon, and would send a logic high or logic low depending on which frequency, 23Hz or 700Hz, was read.

The S2 would then send commands to the wheels in order for the robot to hone in on the beacon. This is where the functionality ends, but the original plan was to integrate this code with an IR distance sensor and two servos in the front in order to capture the beacon in acrylic arms once it was close enough. Then the robot would use Vive coordinates to navigate towards one box or the other depending on the frequency read by the Atmega32U4.

1.2.5 Vive localization

To localize the robot and to partially localize its environment, we used an HTC Vive as directed in class. We built the Vive circuit using photodiodes, as described in the Electrical Design section of this report, and employed the provided code for signal detection to calculate the relative position of the robot with respect to the Vive emitter. The police car and other robots in the field all transmit their respective locations so we planned to leverage this functionality to navigate to the police car and avoid collisions with other robots.

In addition, we also employed the Vive circuit to discern the heading of the vehicle. We aimed to compare the actual heading of the car with the desired heading and apply a proportional-integral-derivative (PID) controller to ensure the robot moved in the direction we wanted it to.

The Vive readings were read by a dedicated low-level C3 processor and transmitted to the staff communications C3 and our high-level S2 via ESP-NOW, a communication protocol unique to the ESP microprocessors.

1.3 Performance

1.3.1 Motor control

Since we have four wheels and need to independently control direction of each wheel, we understood that an S2 would not have enough general-purpose input/output (GPIO) pins to achieve this functionality and the others. Further, since motor control is logically isolated to the other parts of the robot and only required desired speed and direction, we chose to use a dedicated C3 for it.

Initially, we used ESPNOW to communicate with the C3 but we found out that the transmission was unreliable. Even after ensuring that we resent the message until we received a success code, the messages were still sent too slowly and only worked about 90% of the time. Hence, we switched to I2C communications and were able to achieve a success rate of 100%.

1.3.2 Robot motion

In general, we had almost no issues with holonomic motion. The problem was with the wheels rotating at different rates even though they were powered from the same battery. To resolve this, we tried to use optical encoders to detect the speed of each motor and try to match their speeds together. While encoders worked well for one motor, they failed when even two motors were powered. When that didn't work, we applied a different approach instead. We found out that the right side of the vehicle was always moving faster than the left side. To ensure they moved at the same speed, we programmed the C3 to send in 70% of power to the right side compared to the left. At nominal battery voltages, this resulted in a nearly perfect straight line drive.

1.3.3 Sensor noise

For wall following, we initally attempted to use a LIDAR sensor and three ultrasonic sensors so that we could detect the distance on all four sides of the vehicle. The idea was to use this information to accurately control the robot and make it follow the wall. However, we discovered that the ultrasonic sensors were noisy. We tried applying a moving average filter and a median filter, but neither of them could reliably reduce the noise from the sensors. Therefore, we decided to use just two sensors, the LIDAR and one ultrasonic, so as to reduce the number of noise sources.

The LIDAR sensor we used utilized I2C communication to talk to the S2 microprocessor. Note that we also had the S2 communicate with the motor driver C3 through I2C. This led to a conflict. We investigated integrating two I2C communications on the same microprocessor, and it seemed possible, but the LIDAR sensor used a different type of I2C (it uses the built-in Arduino libraries) while the S2-C3 I2C used the Espressif (manufacturers of ESPs) library. We could have tried to used UART with

the LiDAR, however we found an effective solution simply using one ultrasonic sensor to achieve wall following. To minimize the noise, we cut out false readings. We added an 'if' condition where only if the sensor detects the wall in 7 readings in a row, we affirm that there is a wall ahead and behave accordingly.

1.3.4 IR detector range and field of vision

The IR Detection range without the reflective cones was around two feet but increased to around four-five feet with them. The cones also narrowed the field of vision from around 40 degrees to around 15 degrees. These two attributes led to accurate mid-range beacon sensing abilities.

1.3.5 Vive transmission

We had no issues detecting the Vive location data and transmitting said data to the staff communications C3 via ESP-NOW. The issue was when we tried to send the data to our central S2. The S2 was connected to the WiFi so that we could control the robot via our computer. However, we discovered that we couldn't receive ESP-NOW communications on the S2 while connected to an external WiFi network. Hence, we tried a different approach - to detect the Vive signals and connect to the WiFi on the same C3. With this approach, the Vive signals were noisy to the point of complete randomness. We suspected that the Vive detection code was sensitive to interrupts and required near-perfect timing, since similar randomness was seen whenever interrupts were added to the code. In the end, we took the Vive data of the starting position and the police car at the start of the checkoff, calculated the required trajectory to get the robot to the car, and commanded the robot to do so.

2 Mechanical design

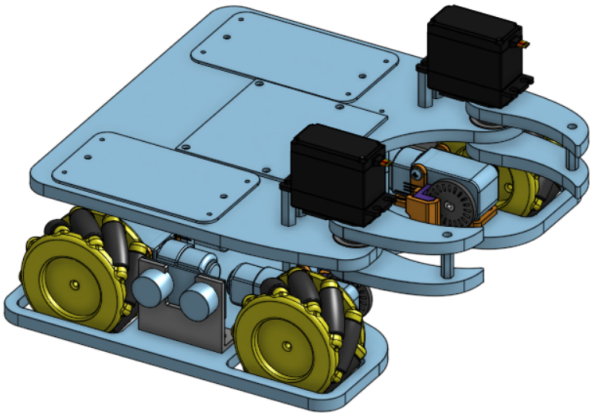


Figure 4: CAD model of the robot. Photo taken from Onshape

2.1 Frame design

The robot was designed to be contained within two levels. The bottom contained the wheels, the bumper, the ultrasonic sensors, the time of flight sensor, and the IR distance sensor. The top contained the servos with attached acrylic arms, the Vive circuits, the ESP32-S2, the Atmega32U4, and the IR detector circuit. The platforms were designed with holes in order to mount all of these different components, with the hole positions being determined after making CAD models of each item.

2.2 Grabber design

The grabbers were designed to contain a four-inch diameter cylinder when closed to account for the diameter of the final beacon design. Each grabber consisted of two levels separated by a standoff on one end and a machined shouldered aluminum dowel on the other. A bearing was pressed on to each dowel and each bearing was pressed into the top frame level to allow for smooth movements.

2.3 Wheels

We chose mecanum wheels as a compromise between stability, ease of integration, and mobility. Due to pin limitations, we linked the PWM signals of the two left wheels to each other and the two right wheels to each other. The wheels provided motion in the four cardinal directions as well as rotation in either direction. These wheels contain 45 degree rollers which enable the horizontal movement.

2.4 Reflective cones

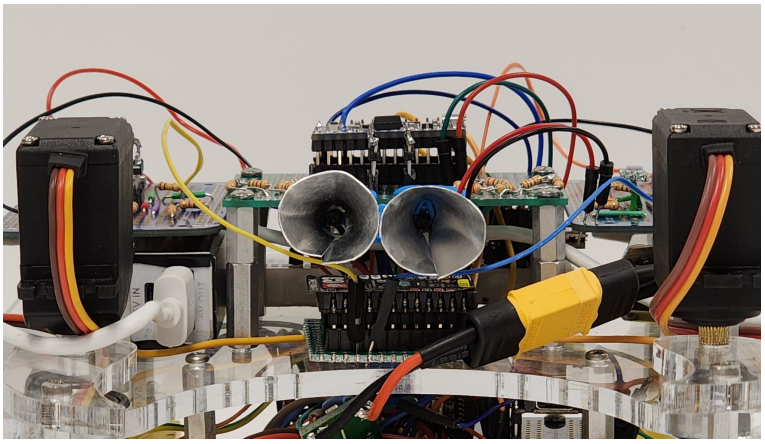


Figure 5: IR Photoresistors and Reflective Cones

At first, we put the cones on the IR detectors for the aesthetic, but we found that it increased the range that the photodetector could sense signals. Furthermore, we found that if the cones were made narrow, we could focus the detection of the signal to the angular width of the cone. We chose to make it somewhat narrow in order to ensure that the robot would only detect the cone if it were in front of the vehicle instead of being outside the width of the grabbers.

3 Electrical design

3.1 Overall circuit design

The robot contained four separate microprocessors, one ESP32-S2, two ESP32-C3's, and one Atmega32U4. Each of these systems was connected through the batteries on the robot and to circuits mounted on the base layers. The circuits on the robot include the motor driver circuit, the IR detector circuit, and the two Vive circuits, as well as sensor and servo connections.

3.2 Motor driver circuit

To drive the 4 motors, we used 2 H-bridges and 1 Hex Inverter. The circuit is shown in figure 10 in the Appendix. The hex inverter allowed us to save on pins because they inverted the signal from the C3 and thus allowed us to reverse the direction of the wheels. Only two PWM signals were used, one for the left side of the robot and one for the right side. This was done to minimize complexity and allow us to control the heading of the robot by varying left and right speeds. With four PWM signals, we could have achieved full holonomic motion.

3.3 IR detector circuit

The IR detector circuit utilizes IR phototransistors to determine the location and frequency of beacons on the field. This is accomplished through the use of two op-amp circuits integrated with the Atmega32U4 paired with code adapted from Lab 2.

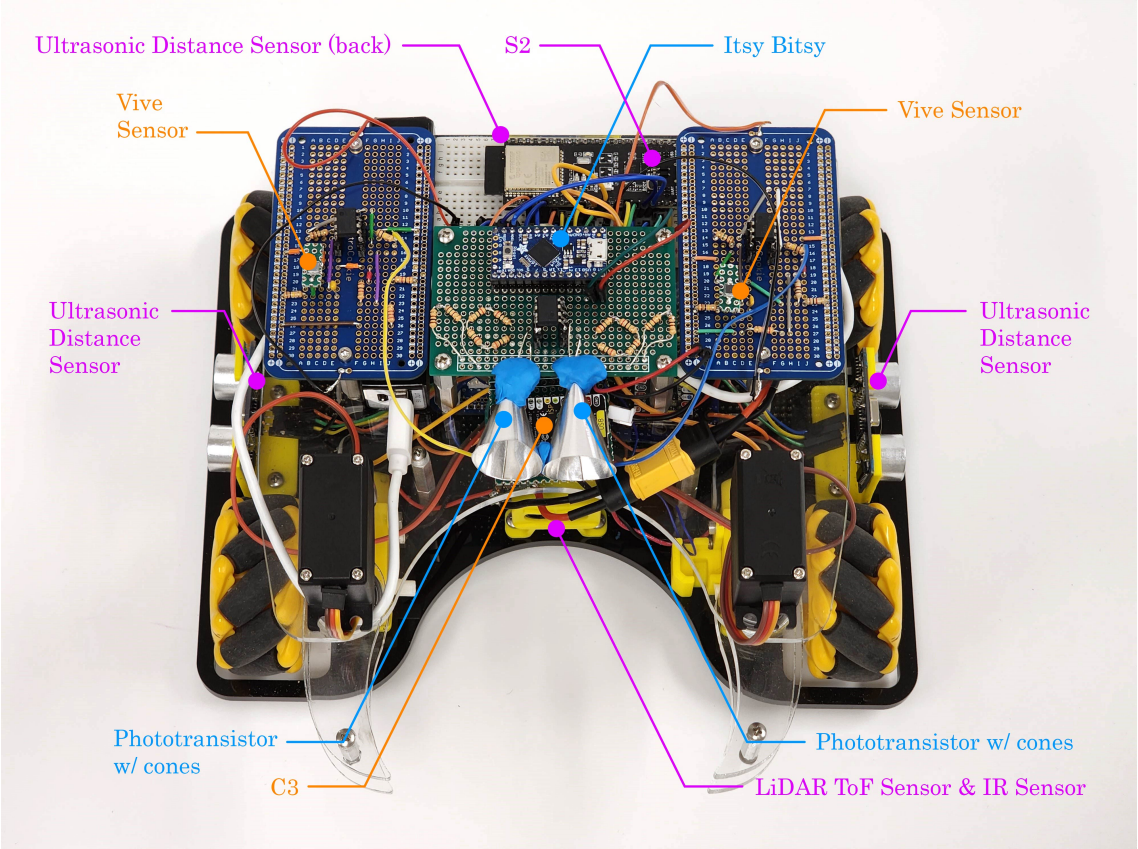


Figure 6: Sensing modalities and microcontrollers on our bot

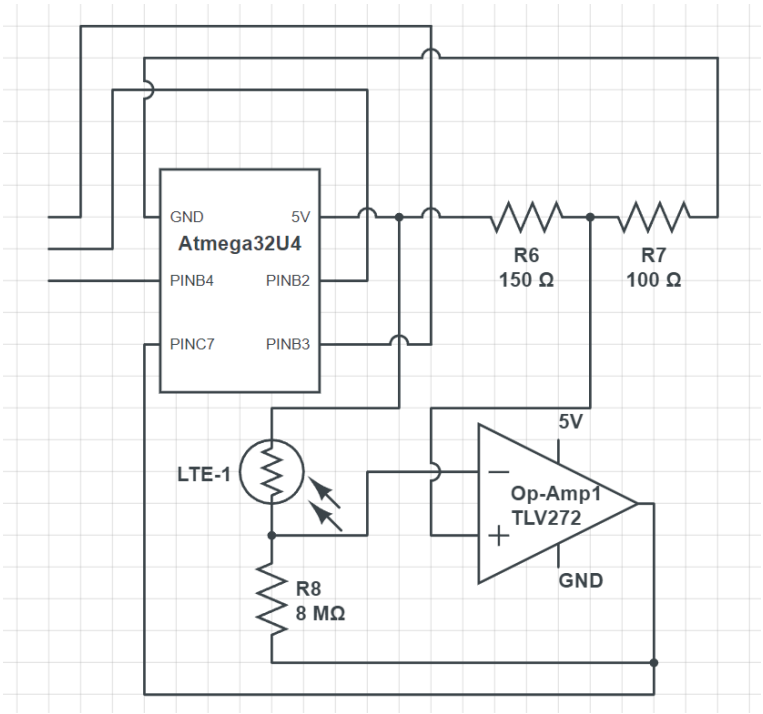


Figure 7: IR detectior circuit. Three outgoing pins connect to the S2

3.4 Vive circuit

For the Vive circuit, we simply used the circuit shown in lecture 22, depicted in figure 5.2.2. We used two of these circuits and the outputs were fed into the C3 dedicated for Vive data processing.

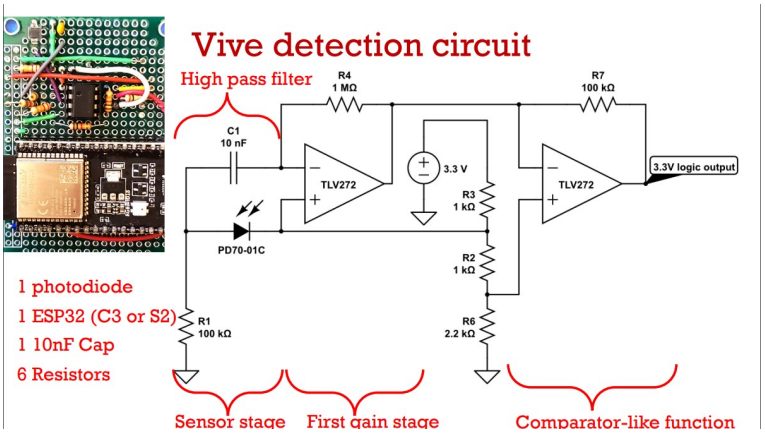


Figure 8: Vive circuit. Photo taken from slides of lecture 22 of MEAM 5100 Fall 2022.

4 Processor and code architecture

4.1 Microcontroller architecture

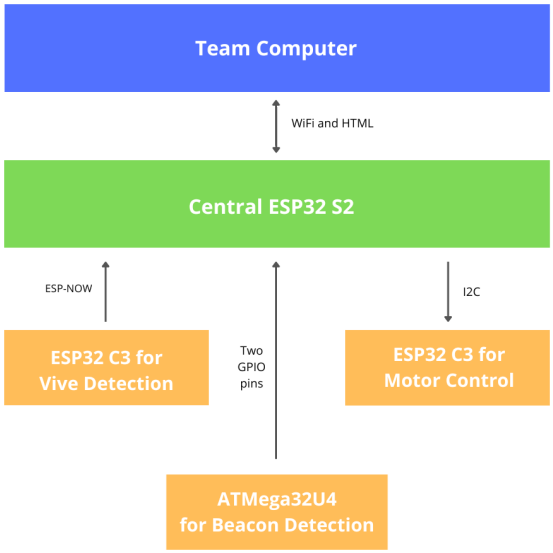


Figure 9: Block diagram describing the microcontroller unit architecture.

Our team decided to delegate tasks to multiple microprocessors to decrease the computational load on each, make space for more functionality due to the availability of greater GPIO pins, and to facilitate parallel progress on different tasks by dividing tasks up across teammates. We used four microcontrollers: a central S2 to perform wall following and to aggregate the other tasks, a dedicated C3 for Vive detection, another C3 for motor control, and an ATMega ItsyBitsy for Beacon Detection.

The C3 for motor control received direction and speed commands from the S2 through I2C. We found that this yielded the most robust system since this was the most important communication line. The C3 for Vive detection sent messages to the S2 via ESP-NOW. Even if certain messages failed to send,

the frequency of these messages meant that reliability wasn't critical. Finally, the ItsyBitsy simply used two GPIO pins to indicate whether the left or right IR detector could detect the IR signal. We intended to add a third GPIO pin to indicate whether the detected signal was 23Hz or 700Hz but since we didn't compete, we didn't add in that feature.

4.2 Software approach

4.2.1 Motor control

For the C3 that performed motor control, we used I2C to receive the desired direction of motion of the robot as a character variable; one character representing forward, backward, left, right, clockwise, anti-clockwise, and stop. We also received the speed of the left and right motors as two integer variables. This was to manually control any drift in a desired straight line motion by setting different speeds for each side of the robot.

Depending on the desired direction, we used a switch case to decide the directions of rotation of each motor. The desired speed was mapped to motor PWM through the LED control (LEDC) functions on the ESP.

4.2.2 Wall following

As explained in section 1.3.3, we used only one ultrasonic sensor for wall following. This sensor was equipped on the back of our vehicle because our mechanical design assumed that the LIDAR sensor would be in the front. To save us from redesigning the base, we chose to wall follow by moving our robot backwards. Hence, the back ultrasonic sensor became our front distance sensor.

First, the robot moved forward. Once the distance sensor detected the wall, we programmed the robot to stop, move back for half a second, move left for half a second, rotate counter-clockwise for half a second (this rotated the robot by 75° approx.), and continue moving forward. This ensured that the robot would move forward at an incline to the wall so as to always face it when moving forward.

4.2.3 Beacon tracking

For beacon tracking, we used the ItsyBitsy and its input capture functionality with the IR photodetector circuit. When the circuit detected a logic high from the circuit, meaning that an IR beam is detected, we used the timer to measure the time between logic highs in order to understand whether the signal was of 23Hz, 700Hz, or anything else. For signals of the first two frequencies, we sent out a signal to the S2 through a logic high on a GPIO pin.

In the main S2, we first move a little left from the starting corner and begin the beacon tracking loop. We move forward for a bit, rotate a half circle left, rotate a full circle right, and rotate a half circle left to center the robot. Then, we move forward a bit. This behavior is repeated until the ItsyBitsy detects a signal from the beacon. In which case, the interrupt stops the robot from rotating and we move forward until we hit the beacon.

We had code to detect hitting the beacon with an IR distance sensor and to close the claws in order to grab on to the beacon, but we chose not to proceed with the implementation of that since we ultimately did not participate in the competition.

4.2.4 HTC Vive

For the Vive, we used the example code but also instantiated another class so that two Vive signals could be detected in the pass of one loop. We found that the Vive circuit only really worked when placed standalone with a different microcontroller, as explain in section 4.3.2.

4.3 Performance issues

4.3.1 Sensor noise

We found that the ultrasonic sensor readings were noisy, hence we applied three different filters. First, we applied a differential filter (basically difference of any new value minus previous value) that ignored any new reading that was more than 20cm from the previous reading. This is because we sample the

sensor at more than 20Hz so it is impossible to move 20 cm within microseconds. Second, we applied a moving average filter to average out the readings from the high frequency noise. Third, we applied a simple sequential count filter. By this, we mean that only if the sensor detected the wall as closer than 20cm six times in a row, we would consider the wall to actually be present and therefore take action.

4.3.2 Vive timing

The main issue we faced with the HTC Vive is that the sensor readings could not be reliably read if it were transmitted over WiFi through HTML or through ESP-NOW. We suspect this has to do with how both of those methods apply interrupts which disturb the Vive sensing code's timer functionality. The way to get past this was to store the Vive location as a local variable and send that variable through ESP-NOW at a moderate frequency of about 10Hz. While the information was being sent, the Vive sensing would be disturbed so the serial monitor would print strange values, but since we send only one value at a time, the value that is read would be a correct value.

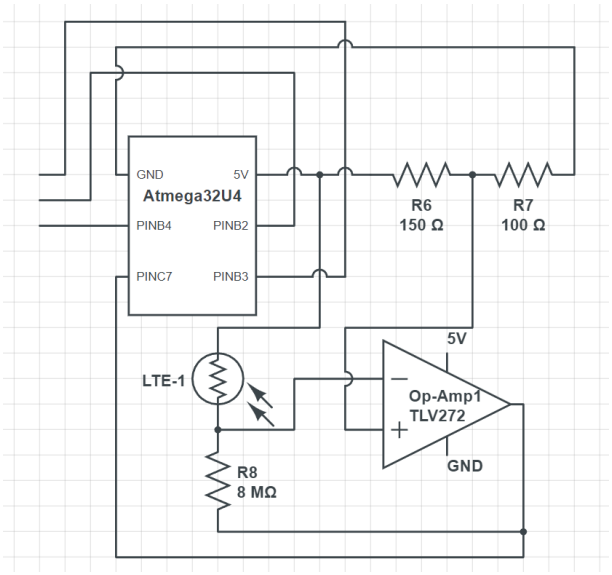
5 Appendix

5.1 Bill of materials

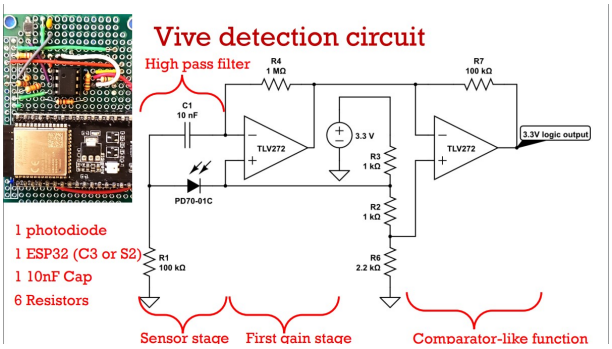
- 1x clear acrylic sheet
- 1x black acrylic sheet
- 2x machined aluminum dowels
- 2x mg996r servos
- 4x mecanum wheels
- 3x ultrasonic sensors
- 1x time of flight sensor
- 1x IR sensor
- 2x LTE-1 phototransistors
- 1x TLV272 op-amp
- 4x acrylic arms
- 2x Electrocookie boards
- 4x perf boards
- 2x metallic sensor cones
- 2x ESP32-C3
- 1x ESP32-S2
- 1x Atmega32U4
- 1x two cell 7.4V li-po battery
- 1x 5V USB battery

5.2 Circuit schematics

5.2.1 IR detector circuit



5.2.2 Vive circuit



5.2.3 Motor driver circuit

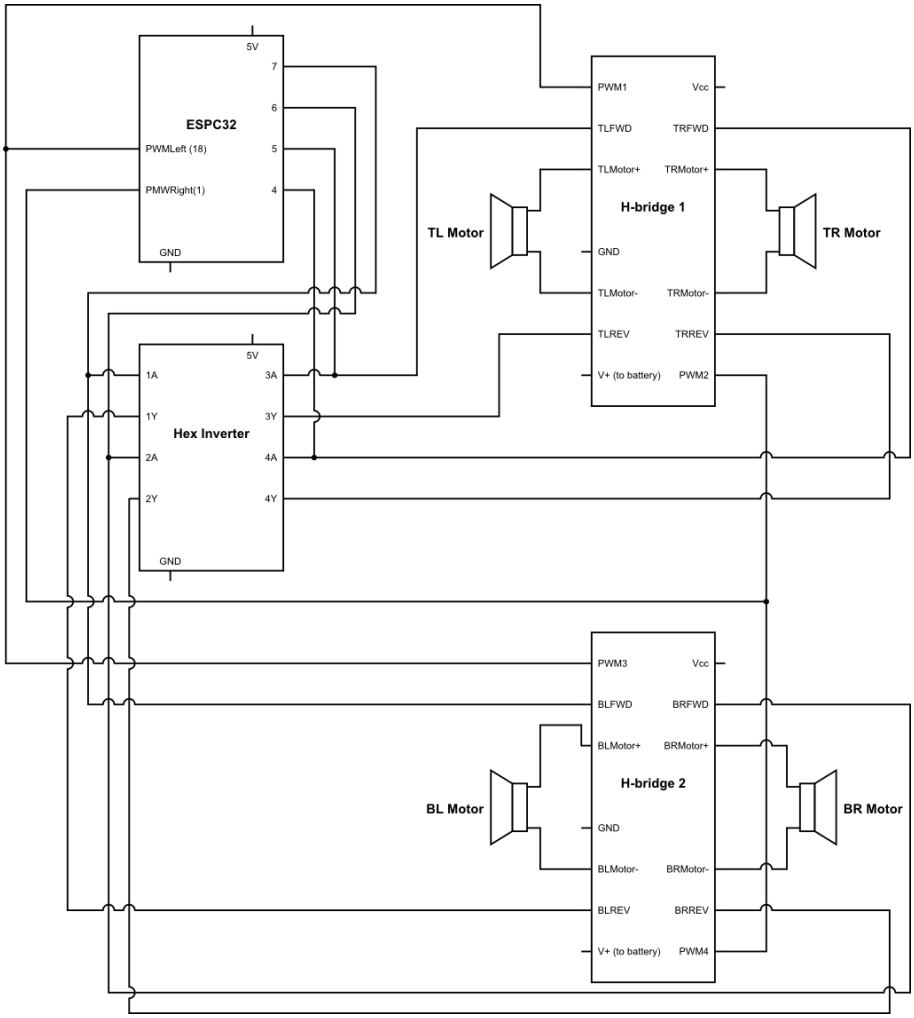
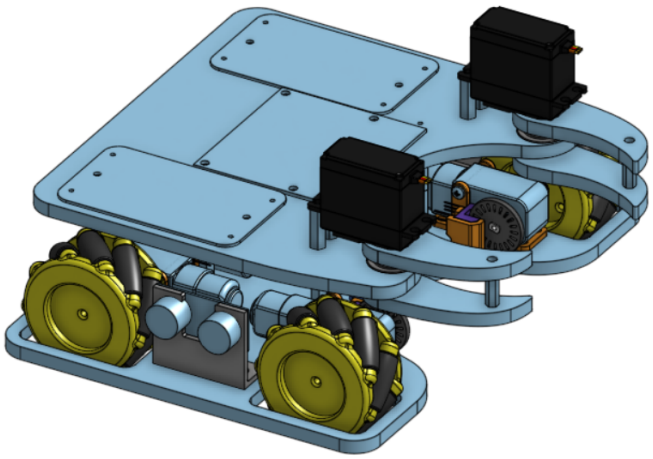


Figure 10: Circuit to control 4 motors using a hex inverter and 2 H-bridges

5.3 Render of robot in CAD



5.4 Videos

Video of wall following: (https://www.youtube.com/watch?v=3LLsQAhQDSM)
Video of beacon tracking: (https://youtu.be/fJMwKjhOudY)
Video of car pushing: (https://youtu.be/zk4yIJdQh60)

5.5 Datasheets

mg996r servos: https://www.electronicoscaldas.com/datasheet/MG996R$_Tower - Pro.pdf$

References

[1] Adafruit. Adafruit - left mecanum wheel - 48mm diameter. https://www.adafruit.com/product/4679, Unknown. Last accessed 18 December 2022.

[2] RoboteQ. Driving mecanum wheels omnidirectional robots. https://www.roboteq.com/applications/all-blogs/5-driving-mecanum-wheels-omnidirectional-robots, Unknown. Last accessed 18 December 2022.