MEAM 510 Final Project - Fall 2022

Lada Korotaeva, Haley Morgenstern, Shreyas Ramesh

Functionality

The overall approach for this project was to enable autonomous functionality for each part of the game using a simple web interface with commands such as “find and reach the police car”, “find and reach the trophy” (or fake), “move to (x,y)” and “follow the wall”, for testing manual movement functions and an emergency stop buttons were implemented as well. For simplicity, the robot featured two wheel drive with a caster in the front and was powered by a 7.4V Li-ion battery and controlled by an ESP32-S2 microcontroller.

IR Functionality

In order to find the IR beacons, two IR sensors were positioned on a servo motor at the front of the robot, the setup was powered and controlled by an ESP32C3 and ran continuously whenever the robot was on finding and tracking the signal of specified frequency (700 by default), it then communicated back to the central ESP-S2 by setting one of the three channels (right, left or forward) to high depending on the angle of the servo motor. When commanded to find the beacon the robot would read the signal, move in the corresponding direction then stop, read the signal again and adjust the direction of movement if necessary. The “go, stop, adjust” strategy was chosen over continuous movement due to the latter being difficult to control in a timely manner, as well as sensing noise confusing the signal. Overall, the IR sensing setup performed well and the robot was able to successfully find and reach the target every time.

YouTube Link: https://youtu.be/1bbZGvqWBLo

Wall Following

For wall following, a simple distance threshold or “bang-bang” control was implemented. An ultrasonic sensor on each side was used to determine the distance to the wall, the signal from each sensor was passed through a software filter to reduce noise. If a distance to the wall was less than a certain value, the robot would move away and if the distance to the wall was greater than another set value but smaller than the distance to the other wall, the robot would move towards the wall. In the front a time of flight sensor was used to determine whether the robot had reached a corner or was angled too much towards the wall. If the wall was sensed in the front, the robot would move back then turn right or left based on the readings from the ultrasonic sensors on the sides. Initially the robot would perform maneuvers too often and too quickly which decreased the overall speed of movement and caused issues with overcoming tape and seams on the game field. However, after tweaking the distance thresholds, movement speed and delays for each of the maneuvers, the wall following functionality performed well.

YouTube link: https://youtu.be/bY3g8CGZt5U

Vive Sensing

The position and orientation of the robot relative to other objects is obtained by means of a phototransistor that receives infrared signals from the Vive lighthouse transmitter. The pulses from the lighthouse and the time it takes from the sync pulse determines the X and Y coordinates of the receiver. Two such receivers could be used to determine the orientation of the robot. Since we could not get two sensors to work simultaneously, we used one sensor to obtain the position of the robot. These coordinate values are then broadcasted using ESPNOW.

YouTube link: https://youtu.be/pkYnq0KXads

Mechanical Design

Our robot used a differential wheel base. This consisted of two separately driven wheels with a ball caster supporting the front end of the car. The base was very stable due to the three points of contact, and easy to steer by controlling the speed of each wheel independently. The ball caster was the best choice of support for the front since it doesn't resist movement in any direction or while turning, compared to a traditional wheel caster. The wheels were picked to have a smaller diameter and better traction, prioritizing control of the car over speed. The motors were the standard motors given to us, since they worked well in the previous lab.

Although the differential base allowed good mobility, unfortunately, using only 2 motors did not provide sufficient torque for full game functionality. The robot had difficulty overcoming tape and other surface imperfections of the field, which we were able to fix by increasing the power (voltage) supplied to the motors. This also allowed us to slowly push the trophies, however the robot was still not nearly strong enough to push the police car, using 4 motors or motors with higher torque would most likely be necessary for that. Aside from that we have experienced issues with lack of traction, the robot was quite light and thus sometimes struggled with slipping, the majority of the weight was also forward of the wheels, which only worsened the situation. To counter that we shifted some components to the back of the robot (for example the vive circuit), as well as placed a small steel block in the rear to add weight and adjust the weight distribution at the same time, this was sufficient to avoid further issues with lack of traction.

The components of the robot were connected in three main tiers. Each tier consisted of a laser cut chassis with perfboards or breadboards on them with circuits corresponding to the sensors on that level. There were cutouts on each side of the tiers so that wires could connect to circuits on other levels while staying contained in the robot. Everything was connected and spaced apart with hex thread adapters. This allowed for a simple assembly and the ability to easily disassemble if necessary. The perfboards were also elevated above the laser cut piece with the hex adapters so that they were more stable, since there were connections on both sides. Strain relief for all wires was provided using tape or hot glue.

Electrical Design

Power Supply

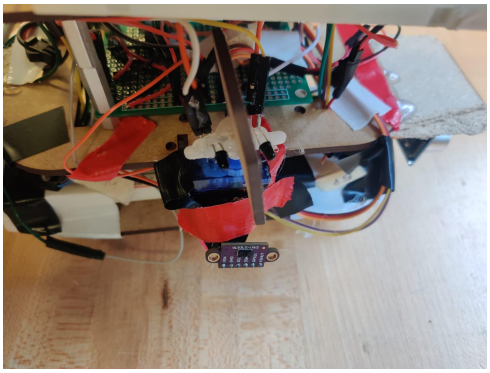
The entire robot was powered with a 7.4V Li-ion battery which plugged into a step-down circuit on a perfboard. The circuit had two pins at the full battery voltage that connected to the motor drivers, additionally it had an array of pins at 5V (stepped down from battery using an LM7805 voltage regulator) for powering the rest of the components such as microcontrollers, sensors, etc

Driving Motors

To control the motors two SN754410 H-bridges were used each receiving signals from the ESP32S2 and power from the battery and step-down circuit. Initially the motor driving circuit was implemented on a perfboard that housed the H-bridge and female pin receivers, pins were chosen instead of soldered connections to ease the assembly and wire routing process, however these connections proved unreliable, Combined with a faulty H-bridge and lack of time, it led to the final circuit being implemented on a breadboard for simplicity and ease of adjustment.

IR Functionality

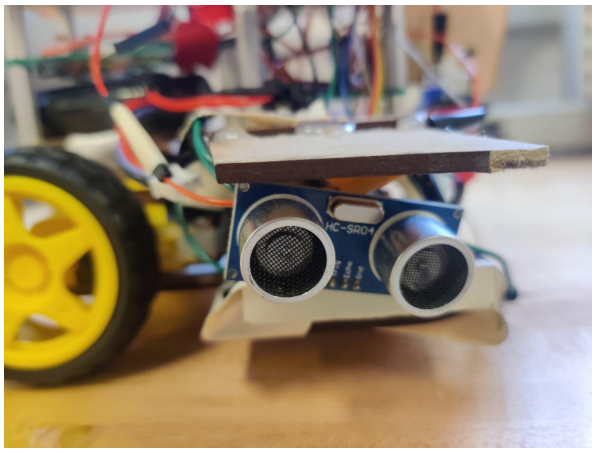
The IR tracker in front of the robot used two IR sensors. The circuit to amplify the signal from these sensors was implemented on a perfboard using 2 Op-Amps (embedded in TLV272), a 100nf capacitor and several resistors per sensor. After being amplified the signal was sent to an ESP32C3 which timed the period between changes to the signal and determined frequency accordingly, based on which of two sensors detected the frequency the microcontroller changed the PWM output signal to the servo motor. No significant noise issues were found with the IR sensors. The tracker in the front was able to align itself with the trophy extremely well as long as the trophy was more than about two inches away, if it was closer the signal the sensors got was very clear so the angle changed a little too much every time which resulted in slight back and forth movement, which however did not interfere with the overall performance.



Ultrasonic and Time of Flight Sensing

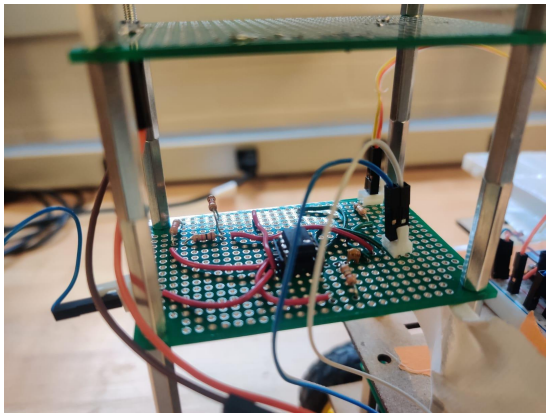
Ultrasonic and time of flight sensors connected directly to the ESP32S2 and 5V or 3.3V power and did not require additional circuitry. We have experienced issues with slight noise in both, more so in the time

of flight, however. For both sensors issues were fixed by using a software median filter, which helped filter out faulty readings, as well as increasing the time allowed for robot action in between readings, which allowed time to get a more stable sensor value and stopped the robot from switching motion modes too often. Another issue that was noticed with ultrasonic sensors is that when the robot got too close to the wall (almost or completely touching the wall), the sensors could not produce any valid readings and therefore when the robot saw the wall in the front it would assume that the side wall was on the opposite side from where it actually was (since it was unable to get readings from that wall) and would turn the wrong way. To counteract this, mechanical "stops" were added on each side of the robot above the ultrasonic sensors. The stops consisted of MDF panels protruding about ¼" past the sensor and preventing it from getting too close to the wall.



Vive Sensing

The Vive sensing module consists of a phototransistor (PD70-01C) that detects pulses from the Vive Transmitter (Vive Lighthouse). Its fast switching time and low capacitance makes it the ideal phototransistor for the job. The phototransistor is connected to a detection circuit consisting of two Op-Amps, one that acts as a signal amplifier and one acts as a comparator. The output from the signal is sent to the ESP32-S2 which processes the signal pulses (Time between the sync pulses and the X or Y pulses) into coordinates that determine the position of the robot.



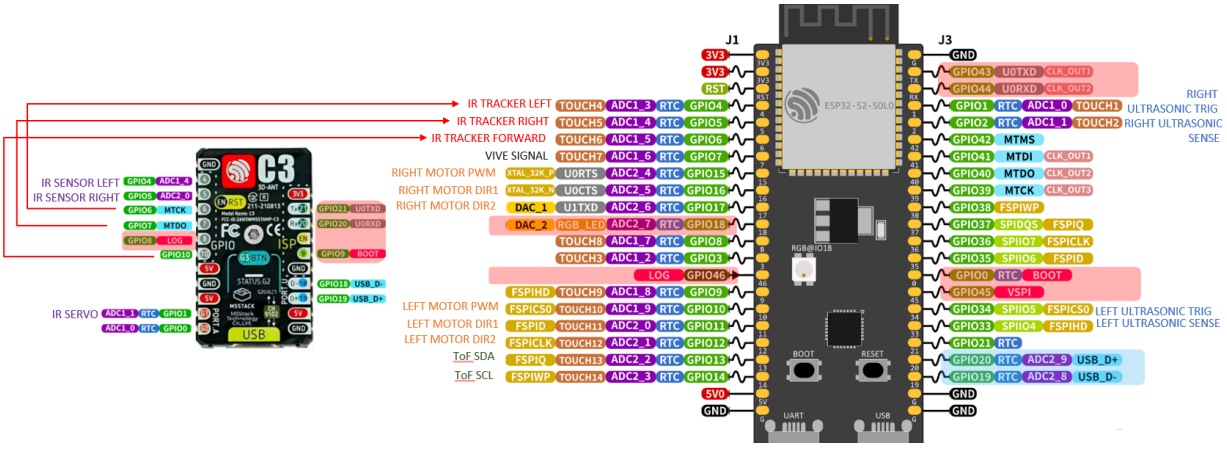
Some of the issues we faced with the vive circuit is that the initial circuit did not amplify the signal enough to be read consistently. Hence we used the new circuit that was provided which worked well.

Secondly, soldering the phototransistor has been quite hard to do, leading to a lot of loose connections and debugging that has been time consuming. Due to the time constraints and the scarcity of the vive sensors many of which were lost to soldering or faulty circuits, we were unable to implement 2 vive circuits, and thus the functionality that required us to know the orientation of the robot in space, such as moving to a specified (x,y) location on the field since the robot wouldn't know whether to move forward, backward, left or right.

We also observed that the signal was noisy. To solve this problem, we used a Median Filter of a window of 3 entries. This smoothened the signal output and led to much stable and accurate position estimates. Additionally, we experienced issues with stack smashing when sending the Vive signal to the staff ESP at times. The problem was traced back to faulty readings that exceeded the 4 digit allowance that existed in the package string for the output, thus overflowing the allocated space, to prevent this from happening we changed the code to allow only values below 10000 to be sent and otherwise send a zero value.

Processor architecture and code architecture

The architecture chosen was rather simple – one ESP32-S2 microcontroller processed all of the logic required for the robot to move autonomously and implement required behaviors, receiving signals from all sensors features and sending signals to control the motors, additionally an ESP32C3 was used to process the signals from and control the movement of the IR tracker in the front. A separate microcontroller was added for the IR tracker due to the IR signal frequency calculation being highly sensitive to timing and delays between reading the sensor signal, because ESP32-S2 had to process robot movement and other functionality in between reading the signal, it was unable to provide consistent frequency calculations.



The code consisted of a web interface file in html and java script, a couple supporting "library" files for the web interface and the vive sensing, and finally the main logic file for each microcontroller. The ESP32C3 microcontroller code housed a routine for reading a sensor value and updating the servo angle accordingly, and a loop to do so continuously for both sensors and send the HIGH/LOW signals to the ESP32S2 via the right, left and forward channels depending on the servo angle. The code for the primary microcontroller was split up into the following subroutines. Handlers for each of the web interface buttons updated the robot mode variable to one of 7 modes, sent a signal to the staff ESP, and a "requested" response to the server. Wifi subroutines dealt with ESPNOW communications and receiving police car location through UDP. Ultrasonic subroutines triggered and read from the ultrasonic sensors returning the

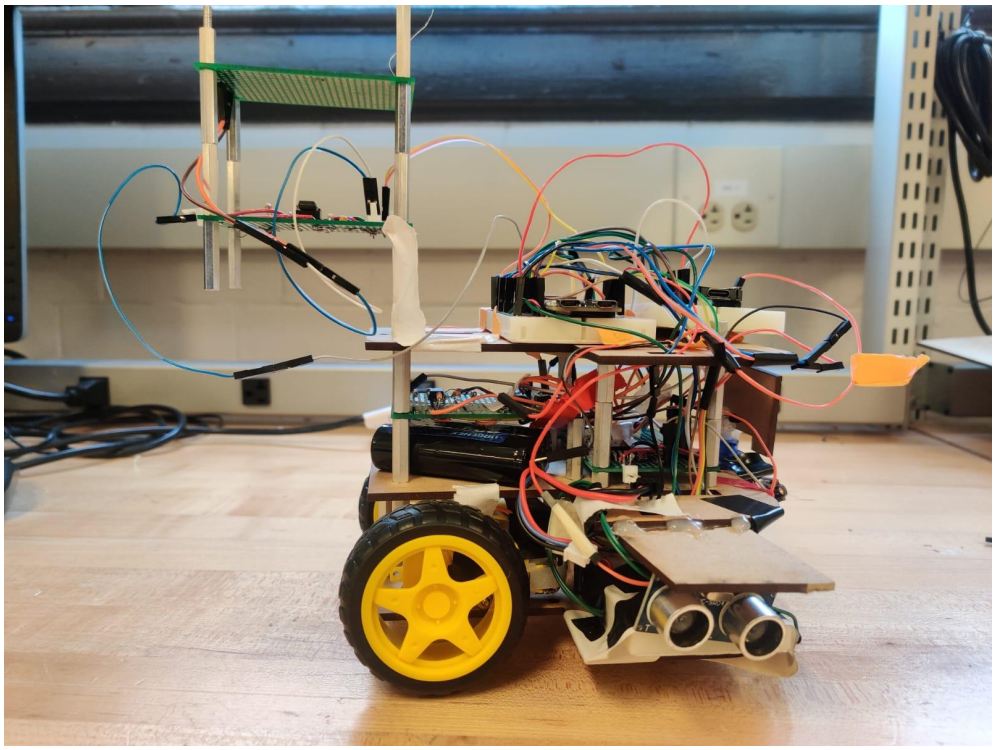
distance values, VIVE subroutines did the same thing for VIVE. Finally, motion functions control the motors depending on the desired direction: move back or forward, turn right or left in place or stop. The loop handled different modes, if the mode was set to 0 – simple motion – nothing was done in the loop, 1 – trophy chase – read in the values from the ESP32C3 and handled motion accordingly, similarly for 2 – fake chase, mode 3 was intended to find the police car, so it called the UDP subroutine and was supposed to implement movement based on the angle and values of the VIVE sensors, which ended up not being implemented in the final version (mode 5 – move to – was intended to work in a similar manner), mode 4 – wall following – implemented bang-bang control using ultrasonic subroutines and a time of flight library, the last mode 6 was a software emergency stop. The final function of the loop was to track the time and send (x,y) location to the staff every second.

Appendix

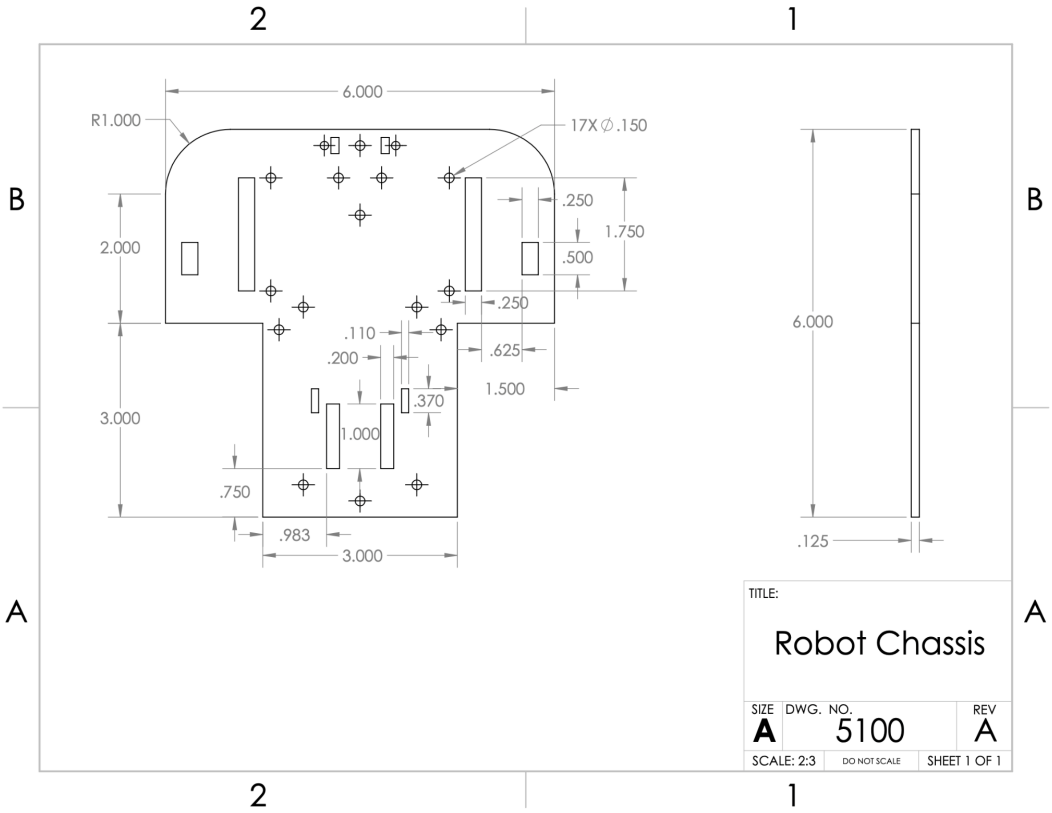
Bill of Materials

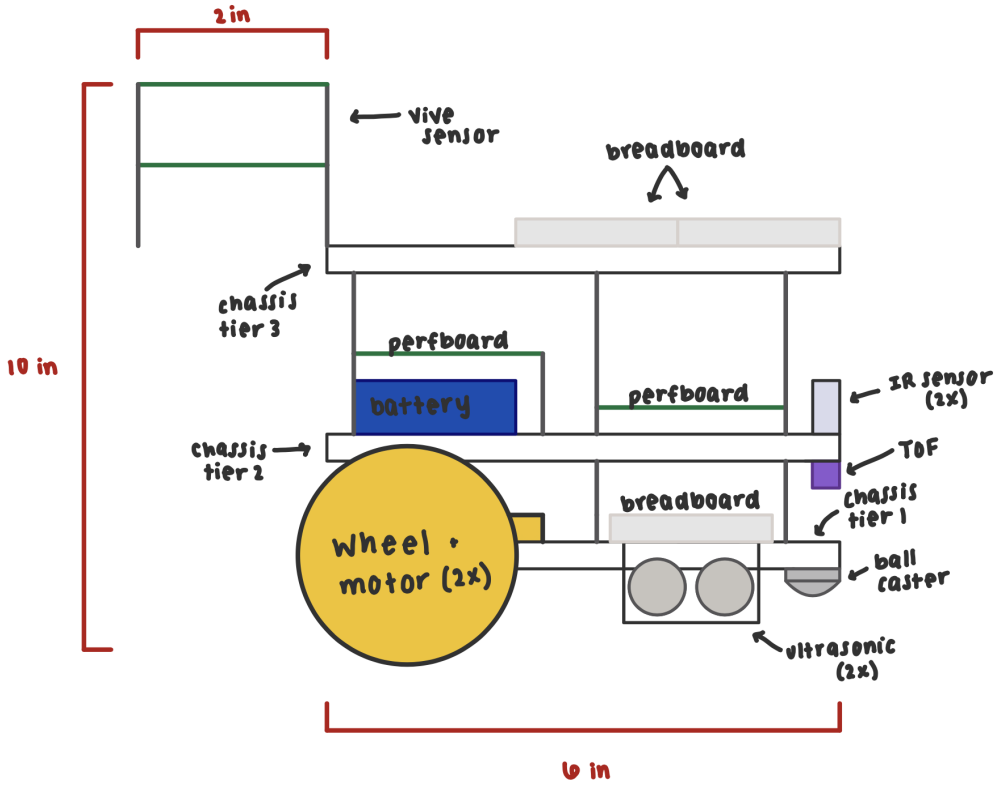
Item	Quantity	Estimated Cost (US$)	Source of Procurement
TT Motor	2	10	Amazon
Wheels	2	5	Amazon
Caster Wheel	1		Own
⅛ inch MDF Board	1		RPL Lab
7.4 V Li-ion Battery	1	25	Amazon
ToF Sensor	1	5	Amazon
Ultrasonic Sensor	2	9	Own
SN754410 H Bridge	1		Ministore
PD70-01C Phototransistor	2		Ministore
TLV272 Op-Amp	4		Ministore
ESP32-S2	1		Ministore
ESP32C3	1		Own
Miscellaneous Items			Ministore

Photo of the Robot



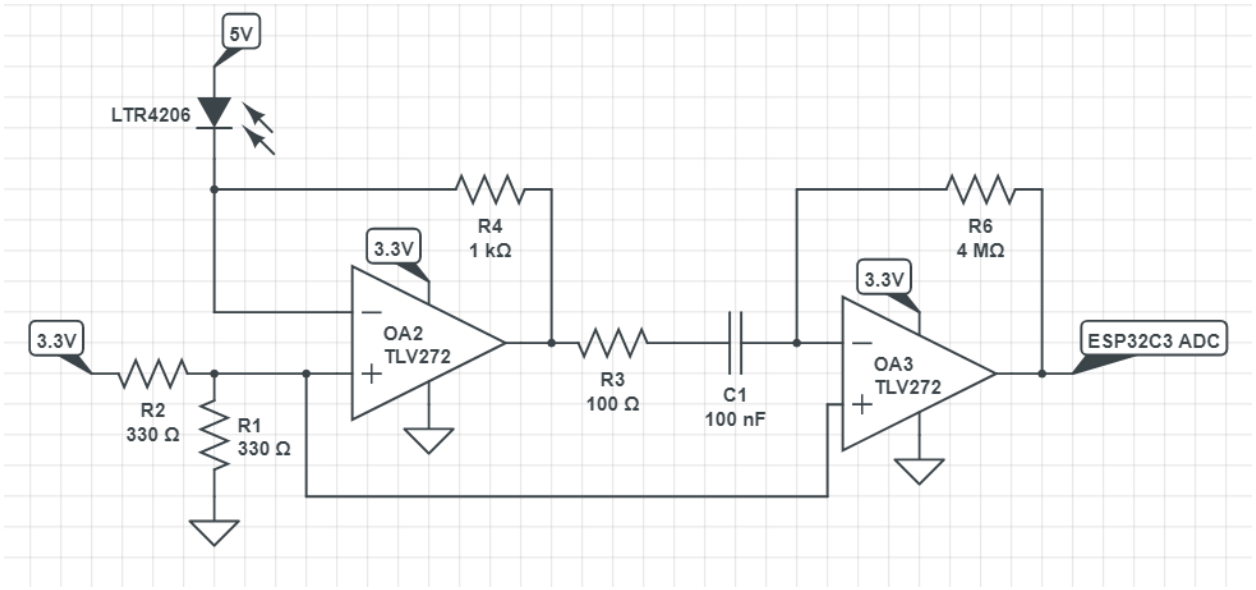
Mechanical Drawings



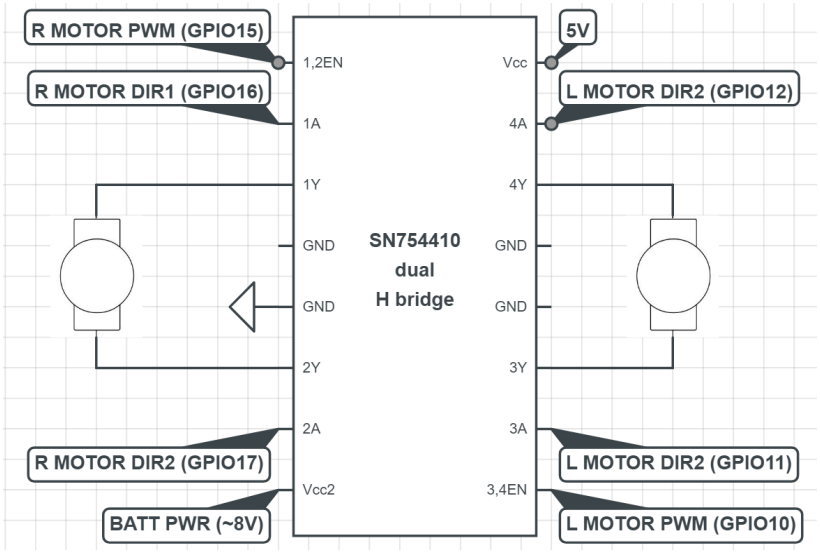


Electrical Schematics

IR sensor circuit (used twice):



Motor driving circuit:



Vive circuit: