MEAM510 Final Report: Design of Mechatronics Systems

Group 34: Daudi Zein, Odekhoa Okaisabor, and Vir Goyal

Functionality

Group 34's game strategy and match tactics for "Grand Theft Autonomous 2022C"

Our approach allows for maximizing total score and out maneuvering opponents:

Game Strategy
Priority Object Retrieval

Early Game: Quickly identify and secure the Trophy and Fake object from the opponent's side. These items are consistent point sources and easier to target at the beginning when the field is less chaotic.
Middle Game: Assess the situation with the Police Car. If the opponent is focused elsewhere, secure it. If it's heavily guarded, consider using it as a distraction.
Exploiting the Double Points Area

Timing: Place objects in the 2X scoring area judiciously. Early in the game, this can be risky due to potential counter-moves by the opponent. It might be more effective in the mid-to-late game when the opponent's movement patterns are clearer.
Securing Points: Whenever safely possible, move scoring objects into the 2X area, but don't let this focus compromise the security of already scored points.
Communication Multiplier

Consistency is Key: Ensure constant, error-free communication to leverage the multiplier. Design your game plan to include regular check-ins or updates, balancing this need with bandwidth limitations.

The design involving a caster wheel combined with two DC motors offers several strategic and tactical advantages over other potential designs in the context of "Grand Theft Autonomous 2022C. Two DC motors provide both speed and independent control of each drive wheel, allowing for differential steering. This means the robot can perform sharp turns and quick directional changes, essential for navigating the compact field efficiently and reacting swiftly to dynamic game situations. The caster wheel acts as a stabilizing point, preventing the robot from tipping over during fast turns or sudden stops, especially when the robot is carrying objects like the Trophy or Fake objects.

Functional Performance

PID CONTROL FOR TWO DC MOTORS
DC motors, especially in applications like robotics or conveyor systems, often suffer from steady-state errors due to constant load variations or friction. The integral component of the PI controller is excellent at eliminating these errors over time, ensuring that the motor maintains its target speed more precisely.

While a full PID controller provides more nuanced control, it also adds complexity, particularly in tuning and stability. In many motor control applications, the derivative component is not necessary, as the system might not have fast oscillating dynamics. Therefore, a PI controller strikes a good balance between control precision and simplicity.

Our code implements PI (Proportional-Integral) control for two DC motors with encoders. The core of the PI control lies in how our code calculates the control signal for each motor based on the current speed and the desired target speed. Let's break down the key aspects:

1. **Reading the Encoder Inputs**:
The encoders attached to the motors provide feedback about the rotational position of each motor. The readEncoder and readEncoder2 functions are called on each encoder pulse (using interrupts). These functions update pos_i and pos_i2, which are the position counts of Motor 1 and Motor 2 respectively.

2. **Calculating Motor Speed (Velocity)**:
In the loop() function, the code calculates the velocity of each motor. It does this by measuring the change in encoder counts (pos - posPrev and pos2 - posPrev2) over a time interval (deltaT and deltaT2). The velocities (velocity1 and velocity2) are then filtered using a low-pass filter to reduce noise.

3. **Error Calculation**:
For each motor, the error is calculated as the difference between the target velocity (vt and vt2) and the actual filtered velocity (v1Filt and v2Filt). This error represents how far the current speed is from the desired speed.

4. **Computing the Control Signal**:
The control signal for each motor is calculated using the PI control formula:

Proportional Term (kp * e): This term is directly proportional to the error. If the motor is far from the target speed, the proportional term is large, causing a significant response.

Integral Term (ki * eintegral): This term sums up the past errors (integral of error over time). It addresses the accumulated offset that a purely proportional controller cannot eliminate. The eintegral and eintegral2 variables accumulate the error over time for each motor.

5. **Setting Motor Speed and Direction**:
Based on the control signal, our code sets the direction and speed of the motors. The direction is determined by the sign of the control signal (positive or negative), and the speed is set by the magnitude of the control signal. If the control signal is larger than the maximum PWM value (255), it's capped at 255.

6. **Actuating Motors**:
Finally, the setMotor function uses the analogWrite function to set the PWM values for the motors, which controls their speed, and the digitalWrite function to set the direction of the motors.

The following video shows the stability of our PI control on our DC motor:
Video link: https://youtu.be/Dm9f_TBSjN4?si=2jfWSGUYd5ASmK6O

The tuning began with a low kp value, gradually increasing it while observing the system's response. A kp value of 3.0 showed improvements in response time and precision in reaching the target speed, balancing these against any signs of overshoot or instability. After setting kp, ki was introduced and adjusted. Starting with a small value, we increased ki until 0.002; the system consistently corrected any lingering offsets without causing instability or slow oscillation.

Autonomous navigate to trophy/fake

The section details functionality of our autonomous robot ability to identify, navigate towards, and interact with an object (either a trophy or a fake) by tracking beacon signals. The robot uses different beacon frequencies (550Hz for the trophy and 23Hz for the fake) to discern between the objects and perform the task.

Beacon Detection

The detectFrequency() function is integral for identifying the beacon's signal frequency. This detection is crucial for the robot to determine its navigation path. The function utilizes the pulseIn() method on the BEACON_PIN to measure the duration of a HIGH (or ON) pulse. Based on the length of this pulse, the robot infers the frequency of the beacon signal.

Frequency Detection: Two primary frequencies are considered:
550Hz: Detected when the pulse duration is between 400 and 3000 microseconds.
23Hz: Detected when the pulse duration exceeds 8000 microseconds.
Any other pulse duration leads to the conclusion that no beacon signal is detected.

Beacon Navigation

The headToBeacon() function governs the robot's movements in response to the detected beacon frequency. If the frequency is either 550Hz or 23Hz, the robot moves forward.
In the absence of these frequencies, the robot executes a left turn, briefly stops, and then assesses the beacon frequency again.

Frequency Re-evaluation: Post-movement, the beacon frequency is reassessed to update the robot's navigation strategy.

The circuit we used for our beacon detection is shown in Appendix.

Testing of our beacon circuit video: https://youtu.be/RoGbqQ5VsEM?si=qIuaPlcXQaghkcIB

Autonomous detection of trophy/fake video with second robot video:
https://youtu.be/yR4LILgSSY8?si=4jyVOIrJIgMs5cY8

The robot successfully detected different beacon frequencies (550Hz and 23Hz), and upon detecting the correct frequency, the robot effectively moved towards the target object. The robot demonstrated the ability to adjust its course based on the beacon signal. The robot was able to detect the frequency more than 15 inches from trophy or fake. What was improved after testing to the final competition was the navigating of the robot towards either the trophy or the fake.

Autonomous move police 10" inch pushed from original spot

Our robot is designed with a Vive tracker. Its primary functions include establishing WiFi connectivity, handling UDP communication for data transmission, and executing movement commands to track and interact with a designated "police car," also equipped with a Vive tracker.

WiFi and UDP Communication:
We established a WiFi connection using specified SSID and password. Then utilize UDP for sending and receiving positional data. The target and robot's IP addresses are predefined for network communication.

1. **Tracking the Police Car with Vive Technology**:
Vive Tracker Integration: The robot employs a Vive tracker (Vive510 class) to determine its own position (vive_x, vive_y) and the position of the police car (police_x, police_y).
UDP Communication for Position Updates: The robot receives the police car's coordinates via UDP packets. The handleUDPServer() function reads these packets and updates the police car's position variables.
Real-Time Position Monitoring: The viveSend() function regularly updates the robot's position. This is crucial for maintaining an accurate and current understanding of the robot's relative location to the police car.

2. **Moving Towards the Police Car**:
Distance Calculation: The cal_norm() function calculates the Euclidean distance between the robot and the police car. This function is pivotal in guiding the robot's movement strategy.
Directional Alignment: The align_robot() function adjusts the robot's orientation to ensure it is facing the police car. It involves straightforward movement and turning maneuvers to align the robot correctly.
Approach Mechanism: The robot uses the calculated distance and its directional alignment to move closer to the police car. The move_to_pol() function encapsulates this logic, repeatedly calling align_robot() and making forward movements until the robot is within a predetermined proximity to the police car.

3. **Pushing the Police Car**:
Final Approach: Once within a certain distance threshold, the robot prepares to push the police car.
Push Execution: The robot employs a sustained forward movement to exert physical force on the police car. This is achieved through the motor control functions, primarily forward(), to maintain a steady and controlled push.
Stop Mechanism: After completing the push or upon reaching another predefined condition (like a time limit or further distance threshold), the robot ceases its forward movement, through the stopcar() function.

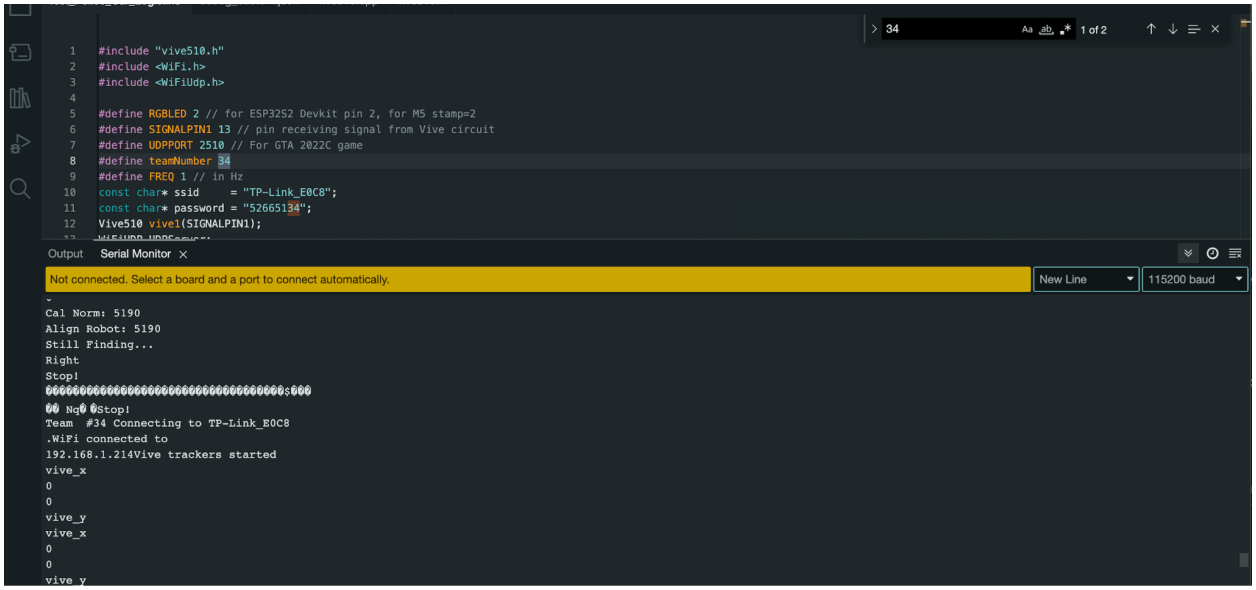


Figure 1: Serial Monitor of Vive System

Autonomous move police 10" inch pushed from original spot with test car video:
https://youtube.com/shorts/CrKyV_5toe4?si=5fA6AbJKa1Sqj5dW

Approach for all functionality

The group tackled the movement of the robot first which includes a chassis, a microcontroller, a motor driver, batteries, two motors, wheels, and caster balls. The approach is to make a two wheel drive car by placing a motor at the midpoint of both sides along with caster balls on each corner of our rectangular chassis. The group chose the SN754410 h driver in conjunction with two 9V AAA batteries to drive the motors. The plan was to minimize wiring and GPIO usage which is why the group decided on moving with just two motors. The microcontroller of choice is the ESP32 WROOM because it met our needed amount of GPIOs and supported the use of analogWrite to send PWM signals for our motors. We abandoned the use of LEDC because it was less efficient than analog and digitalWrite. We tested each movement that we needed: moving forward, rotating right and left, veering right and left, and moving backwards. Once we tuned our movement code, we saved those functions and applied it to each task.

For our wall following task, we chose to follow the track's wall by turning clockwise. We placed a VL53L0X sensor on the front of the car and on its left. We want it to keep a certain distance from the left wall and rotate when a corner is detected. When the left sensor detects that the robot is too close to the left wall, the robot will veer right briefly. When the robot is too far from the left wall, it will veer left. When the robot senses that there is a wall in front of it, the robot will stop and rotate right. As long as the robot is within the set distance of a front and left wall, the robot will keep moving forward. This way the robot keeps a certain distance to the wall, following it as it moves.

The IR sensor task was done with on IR sensor that we made. The robot will rotate to the left until the IR sensors pick up a valid frequency. Only the IR sensors see a signal will the robot move forward towards the trophy. When no desired signal is detected, the robot will scan the environment by rotating.

The VIVE and Police car task was completed with a VIVE detecting circuit that receives the robot location and code that receives the police car's location and commands the robot to pursue the police car. The robot will first match the x axis of the police's location by moving forward. Depending on the sign of the difference between the police's and car's y axis, the robot will turn left or right and then ram the police car.

All information received and sent to the car is broadcasted as a message via ESP-Now.

Performance what worked/what didn't and why

We succeeded in all the tasks. Our time of flight sensors were stable and produced minimal noise when measuring distance. The wall following task then relied on tuning the parameters of the car's movement and measuring the distances that were crucial to execute movements. We ran multiple tests until we saw consistent successes in finishing a lap. We conservatively measured the distance that the robot should stay in so that the robot does not get too close to anything. The IR sensors worked as intended. It was robust enough to detect signals far away simply by measuring time between highs and lows of the incoming signal. We only used one IR sensor and made the robot rotate around until it found the IR sensor and only then approach it. Using one VIVE sensor worked in our favor but it took tuning to get the stable values of both the robot and police. We chose to ignore data when the location outputted zero which helped in making the movement decision of the robot. What we learned was that making the robot move slow was important. It is better to move step by step and stop before doing a different motion.

Mechanical Design

First Robot

Actual Mechanical Performance

The actual mechanical performance met our expectations. We wanted to be able to rotate about the center, veer left and right, and move forwards and backwards. Steering the robot with two motors aligned at the midpoint of each side with the support of two casters made this possible. We thought the casters were going to be an issue but placing the small roller balls on each corner helped the car's movement. We also used wheels that had deep grooves to increase grip on the surface. The motors in conjunction with the drivers were strong enough to move everything at our desired pace.

Iteration of Mechanical Design

The group tried to use four motors with four wheels in conjunction with mecanum wheels for our basis of movement. The plan was to be able to translate the car in all directions as well as rotate. We thought that this would make tasks like finding the police car easier. However, we noticed that driving four motors caused a lot of noise that interfered with other signals (be it a received signal from our IR sensors or the other outputs of our motor driver) and required a lot of current which we could not provide with our 9V AAA batteries. We connected all the components ground together and tested with a power supply but our drivers still produced messy signals. Additionally, with four motors wiring became a huge issue. As much as we tried to keep things organized, the amount of wires involved made it an issue when it comes to debugging and building. Perhaps with a better motor driver and LiPO batteries the four wheel drive idea would have come to life. We found out that our microcontroller had ports that did not work. Using four wheels cost a lot of GPIOs to be used and at the end we saw that we did not need that many to achieve needed movements.

Second Robot

Intended Mechanical Performance

The robot was designed to have a standard four-wheel configuration (two caster and differential wheels), which provides stable support and straightforward movement. The size and width of the wheels are balanced to support the weight of the robot while also allowing for maneuverability. Using differential steering allowed the robot to pivot around its center, which is effective for precise movements and tight turns.

The motors need to be powerful enough to move the robot swiftly to the objects and exert the necessary force to push them. That is why our team decided to change to 12V DC motors.

The robot's chassis is designed to keep its center of gravity low, which is crucial for stability when pushing objects. The body structure is robust, likely to withstand the stresses of collision when interacting with objects.

Actual Mechanical Performance Observations

When the robot pushed objects, it remained well balanced without tipping over, which reflected a well-designed weight distribution. The robot was able to steer accurately towards objects and was able to align itself correctly to push objects. There was minimal slippage or skidding, which indicated a well-constructed PID control; that the wheel treads and motor power are well-matched to the robot's tasks and operating environment.

Iteration of Mechanical Design

Trial 1: Implementing a fixed-axle steering system for simplicity.

Failure 1: The robot struggled with maneuverability, unable to make sharp turns or pivot in place.

Lesson 1: A differential steering system or a more complex axle articulation was necessary for the required agility.

Trial 2: Using a single caster wheel as opposed to two.

Failure 2: There was less stability, especially when the robot is performing tasks like pushing objects. When the load is not centrally distributed, the robot would tip or wobble. The robot was also prone to veering off course.

Lesson 2: Using two caster wheels for our robot's base provided support at four points, forming a quadrilateral shape of support which provides a stable platform when the robot is stationary or moving.

Electrical Design

Actual Performance (motor drivers, sensors, noise etc.)

Starting with the wall following, our time of flight sensors are stable. It measured distance as expected. What we needed to do was account for minimal noise. It came down to tuning the distance that dictates the movement of the car. We conservatively measured where we should keep our car and from experiments we succeeded in mitigating crashes and following the wall well.

Our IR sensors met our expectations. It produced clean square 0 and 5V signals from far away without getting noise from other light sources. Though the car had to take time to sweep the area to find the trophy, in the end as long as the robot sees the trophy, it will approach the trophy quickly.

The vive and police car detection worked as intended as well. Our vive circuit is very stable and only jumps between values of around 10 to 40 as long as the robot does not move quickly. The police car detection was the same. What we had to tweak was following the timing of when the signals are being received and filtering out the zeros.

What failed? (and thus learned from)

We tried soldering all our components on a perf board to reduce space consumption and inconsistent connections that come from a breadboard. However, the results were that our perfboarded circuits performed worse. We checked our connections visually and using a DMM but the signals we were expecting were not being shown. We opted to use smaller breadboards instead. The group learned about the practicality of breadboards and that perf boards are not always the best answer. Perf boards depend on the individual's manufacturing skills and the quality of the solder.

Another thing we tried was making two vive circuits. Unfortunately, the ministore ran out of op amps quickly so we had to opt out with just one vive circuit. This experience just taught us to be more prepared and to take opportunities early.

Electrical schematics of our circuits have been included in the appendix.

Processor architecture and code architecture

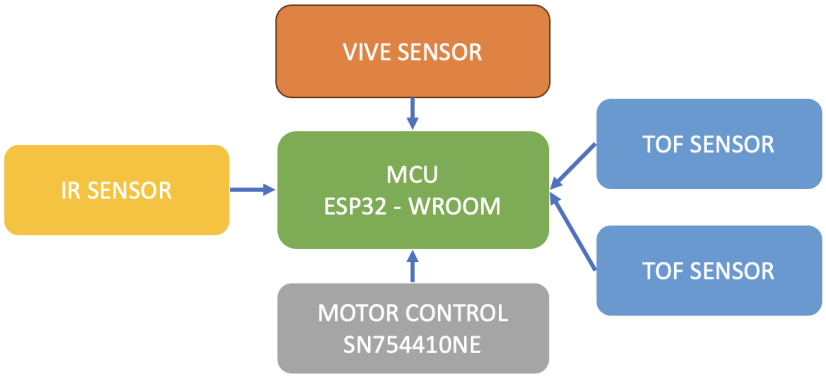


Figure 2: Block diagram of how MCU's are logically connected

Code Architecture/Software Approach

For the car's movement, we used digitalWrite and analogWrite in different functions to send signals. The manual movement using HTML assigned different numbers on each movement button. When a button is clicked, the microcontroller receives which button number is pressed and therefore sends the movement command assigned to that number.

The wall following task was done by continuously measuring the front and left time of flight sensors. When the front sensor reads a distance less than 40cm, the car has to rotate right slowly. When the left sensor reads a distance less than 25cm, the car should veer to the right. When the left sensor reads a distance more than 40cm, the car should veer to the left. When the car is within 25-40cm of the left wall and the front wall, it should keep going straight. This method ensures that the car tries to hug the wall as much as possible and turns right when it sees corners. Considering the play field's wall wasn't perfectly straight, a specific following distance was chosen.

The IR sensors measured frequency by measuring how long a signal is on (measuring period). We calculate the frequency, compare it to a conservative range of frequencies, and when it is within that range it will be assigned to the actual frequency. The robot will sweep the area to try and find the trophy, only when the robot reads one of the frequencies will it move towards it. Both IR and TOF sensors were incorporated using pre-existing libraries for efficiency and reliability in the software operation.

Our team incorporated a PID controller after establishing the ability to both monitor and regulate the speed of each motor. This controller's purpose was to align the motor speeds, as determined by the encoder function, with the user-defined target speed. The goal was to counterbalance environmental noise, achieving movement in line with user expectations.

Subsequently, our team focused on integrating wireless functionalities. This encompassed the transmission and reception of Vive data via the UDP server, along with sending instructions to the TA using ESPnow. A notable challenge was configuring the ESP in AP_STA mode without setting up the station mode, as doing so adversely affected the functioning of ESPNow.

The police car detection involved initial Vive data from the UDP broadcast, followed by the robot's own Vive readings. With only one Vive sensor, the robot's approach involved random movement to determine proximity to the police car, adjusting direction based on this feedback. When proximity fell below a certain threshold, indicating likely contact, the robot would exert full force to push the police car until manually stopped.

Our team's coding approach was chosen for its simplicity and efficiency in the main loop. Initially, our team validated the code's functionality in the main loop. Once confirmed, we transformed the code into a callable functional block. After successful testing as a functional block, it was integrated into the main code, allowing for streamlined operation and easier debugging.

What failed? (and thus learned from)

TOF (Time of Flight) Sensor Setup

In the initial stages, our group managed to operate a single TOF sensor through an ESP32C3's 12C port, utilizing the VL53LOX software. This setup was anticipated to facilitate distance measurement from 30mm to 6000mm, with an alert for distances beyond this spectrum. However, issues arose with accuracy diminishing past 600mm.

Additionally, the observation was made that employing just one TOF sensor necessitated the robot to trace a sinusoidal path for precise measurements, and variations in distance frequently led to decreased accuracy.

Moving forward, our group experimented with deploying two TOF sensors concurrently for measuring distances. Upon obtaining a second sensor, we opted to adapt the existing TOF sensor library and modify the example code. This strategy proved effective, allowing the simultaneous use of two TOF sensors for distance assessment.

WIFI Setup

Significant effort was also dedicated to integrating UDP with ESPNow for the Wi-Fi setup. Our team knew that most successful implementations of UDP with ESPNow were achieved using AP_STA mode. Despite incorporating code from lectures, consistent failures occurred in transmitting messages via ESPNow to the receiver ESP. Assistance was sought from other teams, but no specific errors were identified. The solution emerged when our team methodically examined the code, realizing that setting up the Wi-Fi in station mode interfered with ESPNow's functionality. Ultimately, our team managed to implement both UDP and ESPNow effectively, but only using the AP component of the mode.

Retrospective (non-graded)

The time invested in key projects and assignments for this course was as follows:

Automated System Design/Integration: 3 days - This project involved designing and implementing an automated control system. Developing the algorithm and testing its efficiency.
Sensor Integration Lab: 2 days - In this aspect we focused on integrating multiple sensors to achieve accurate data collection.
Robotics Challenge: 4 days - We designed a robotic solution to a given problem; primarily in programming and system testing.

The total time spent was approximately 9 days. This course has been essential in teaching us the intricacies of system design and the importance of precise sensor integration. It provided a deep dive into robotics, automation, and real-world application of these concepts.

The most rewarding aspects of the course were the Grand Theft Autonomous Challenge. The project allowed us to apply theoretical knowledge in practical scenarios, enhancing our understanding and problem-solving skills. The Sensor Integration Lab, though challenging due to its technical demands, was an invaluable learning experience that pushed us to improve our technical abilities.

In terms of improvement, the course could benefit from more structured time management guidelines and additional resources for complex labs.

Appendix

First Autonomous Robot BOM

Sr. No.	Part Name	Qty	Cost
1	Caster Wheel	4	$6.00
2	Mobile Base	2	Laser Cut
3	ESP32 WROOM	1	$9.99
4	12V DC Motor (GA25-370)	2	$33.86
5	SN754410NE Motor Driver	2	Mini-Store
6	LiPo Battery	1	$15.99
7	Power Bank	1	$16.19
8	TOF sensor (VL53L0X)	2	$30.00
9	PD70-01C/TR7	1	Mini-Store
10	LTR-4206	2	Mini-Store
Total			$112.03

Beacon Circuit

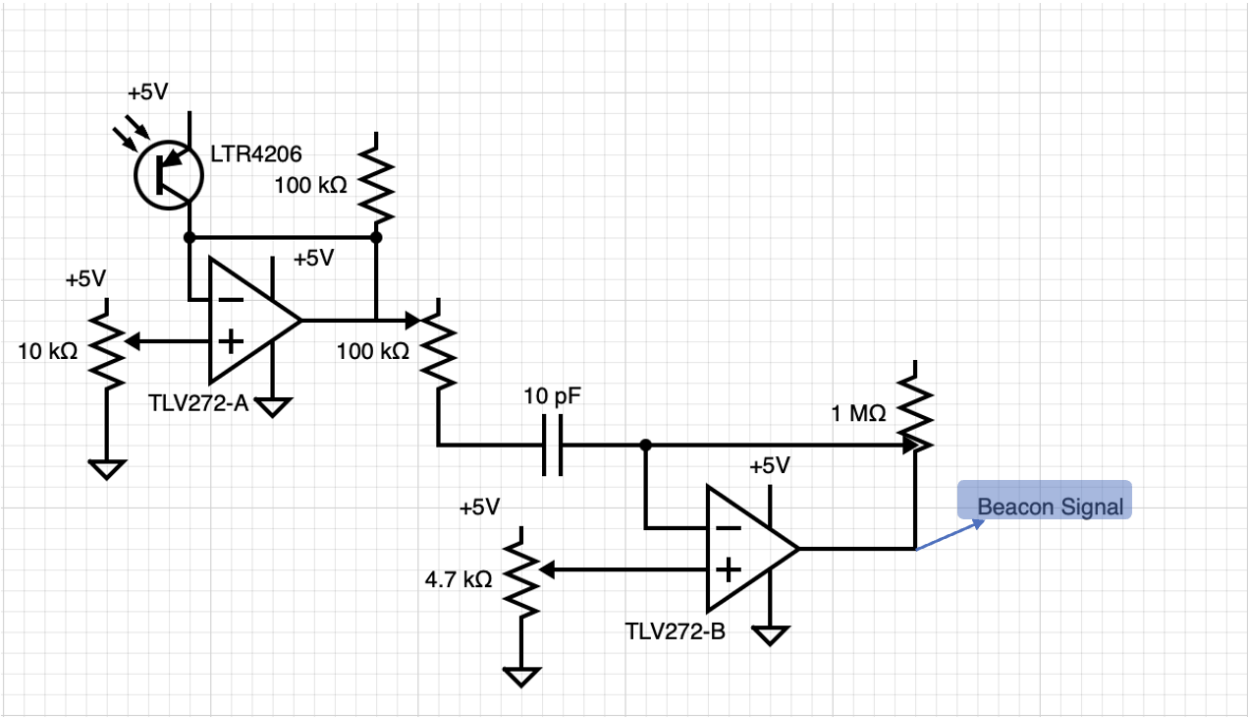


Figure 3: Beacon Circuit for Robot

Vive Circuit

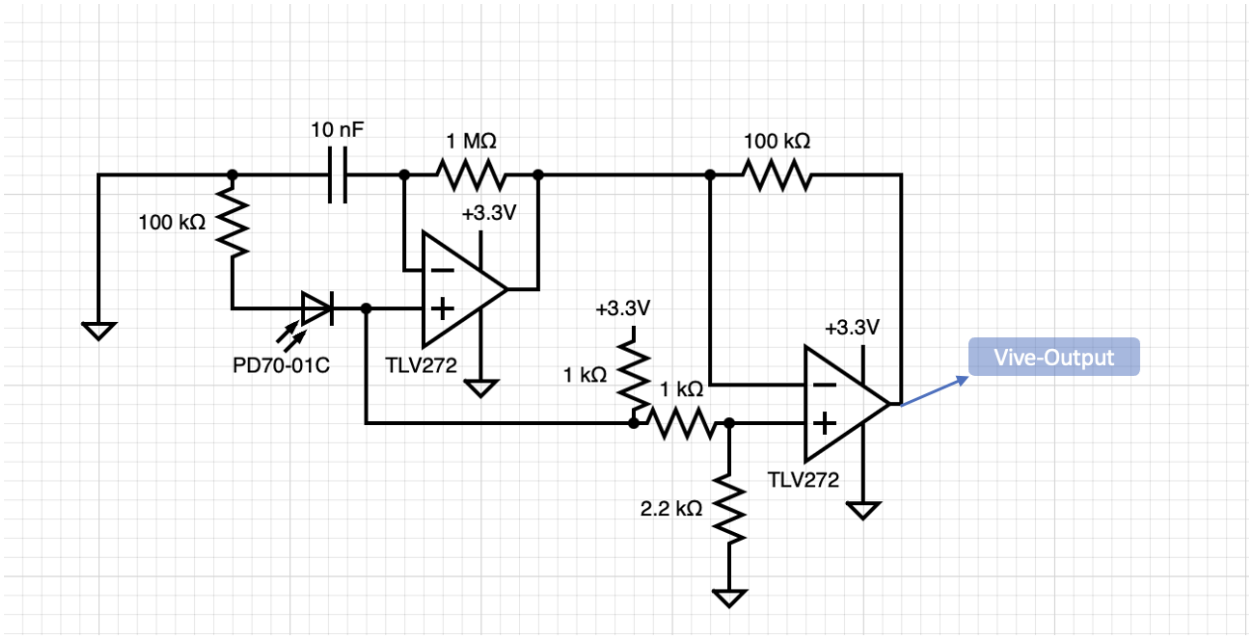


Figure 4: Vive Circuit

Time Of Fly Sensors

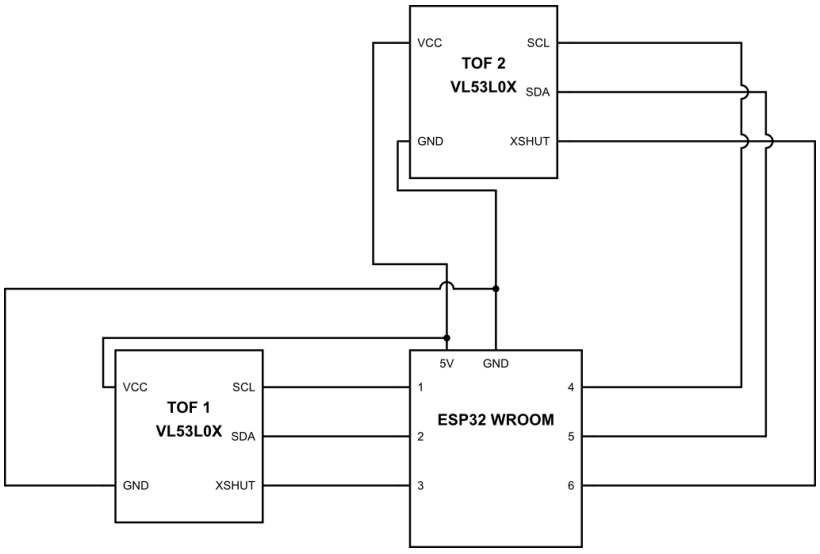


Figure 5: TOF Circuit

Motor Driver

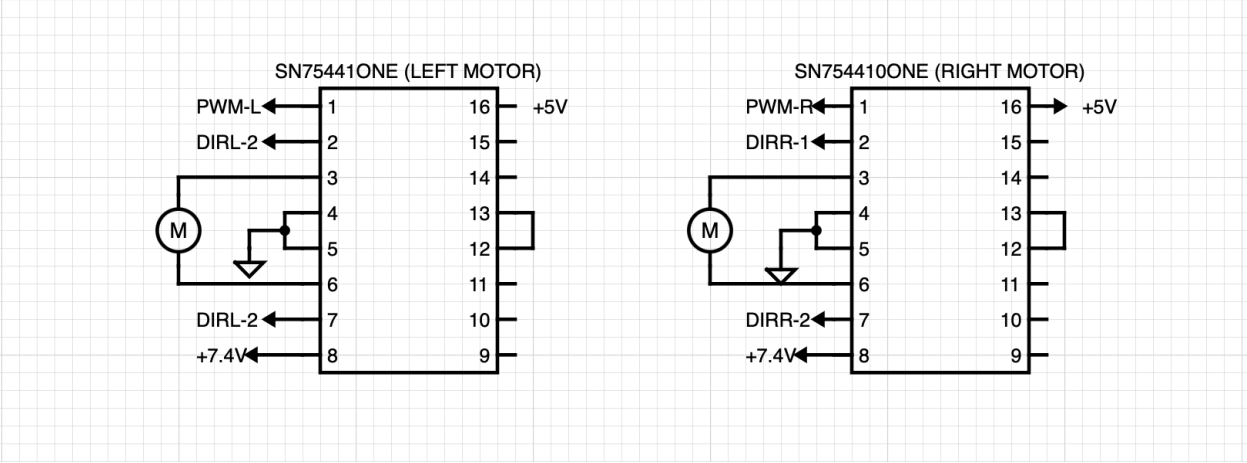


Figure 6: Motor Driver Circuit

Complete Circuit

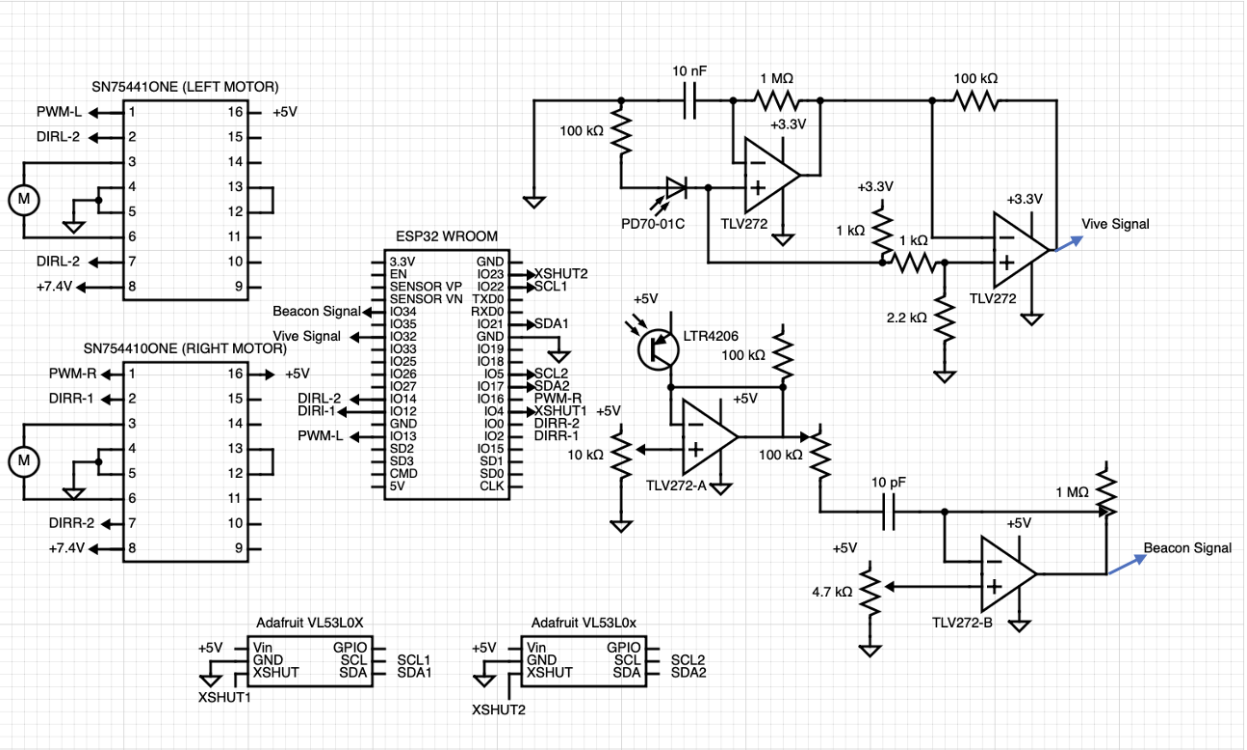
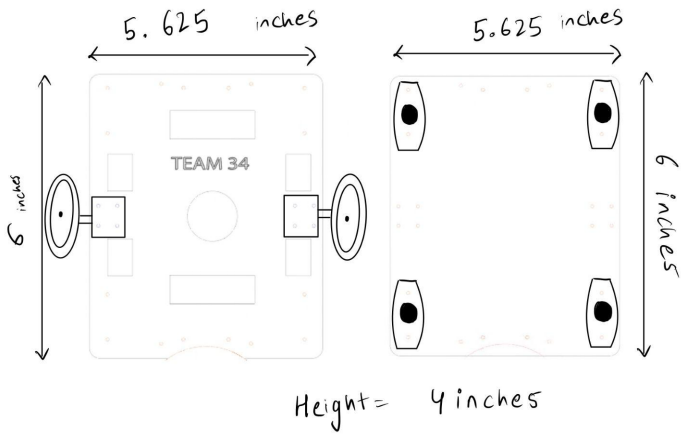
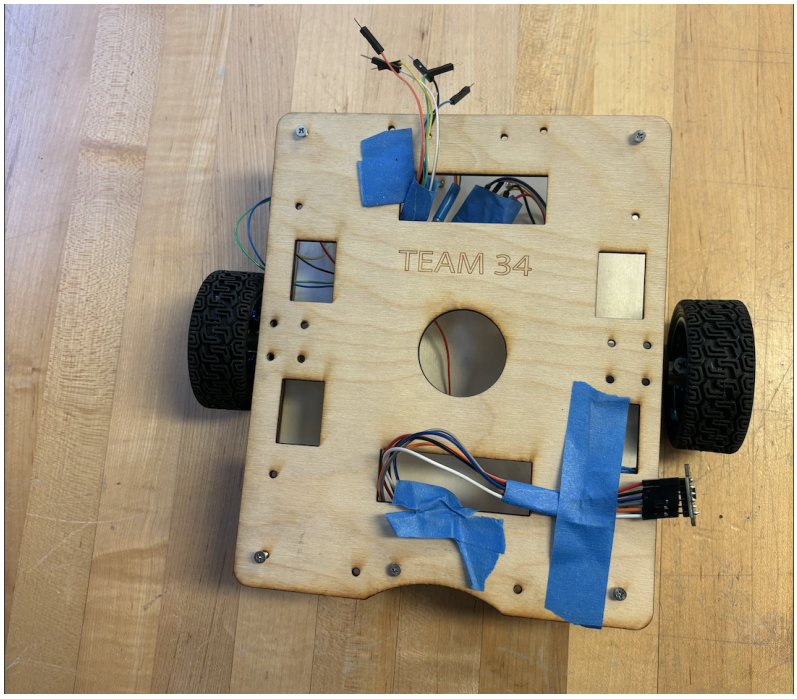
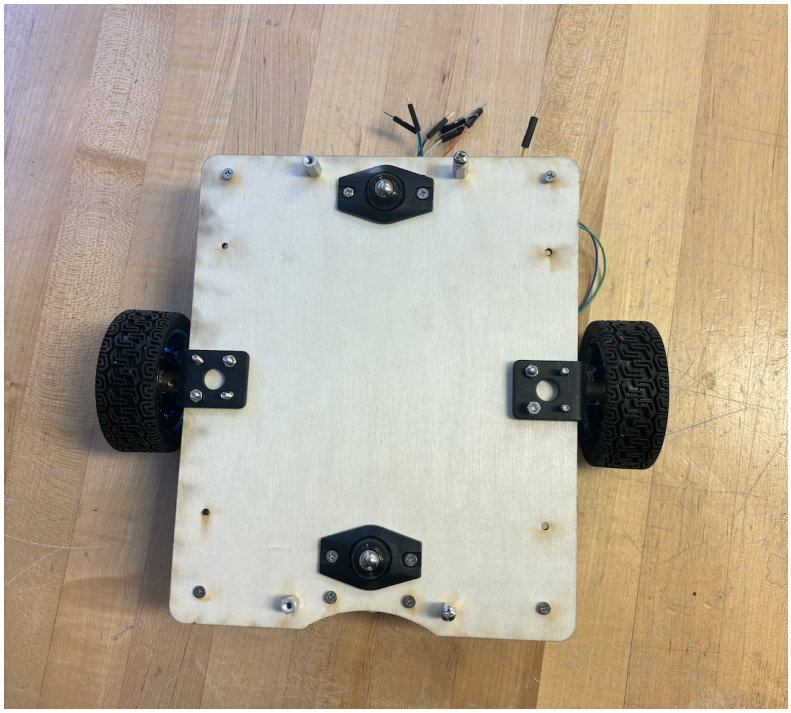


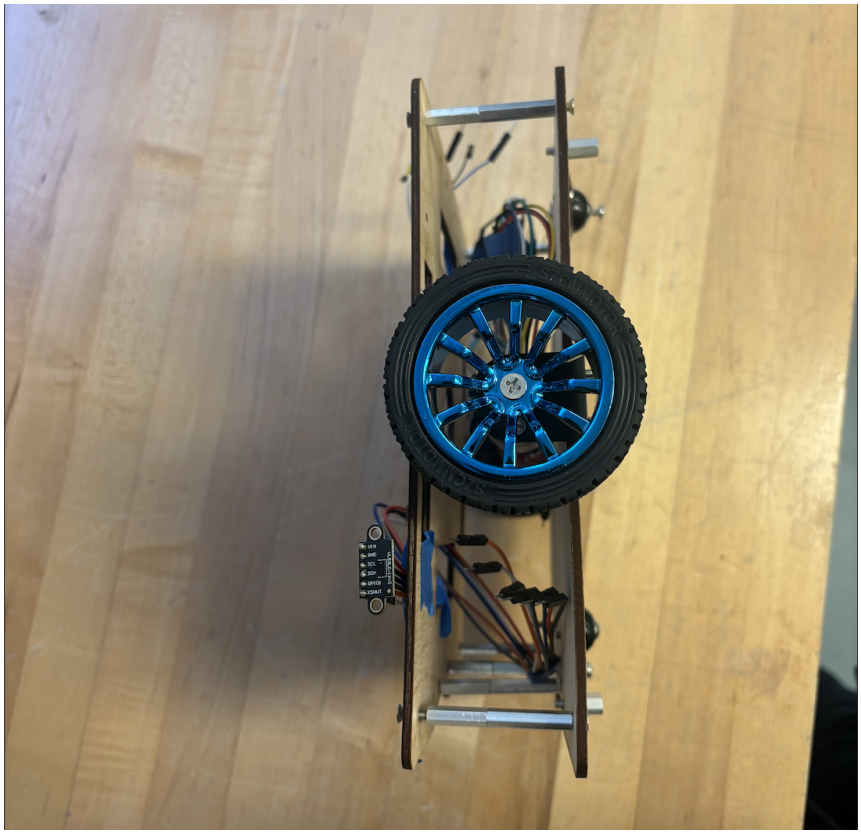
Figure 7: Complete Autonomous Robot Circuit

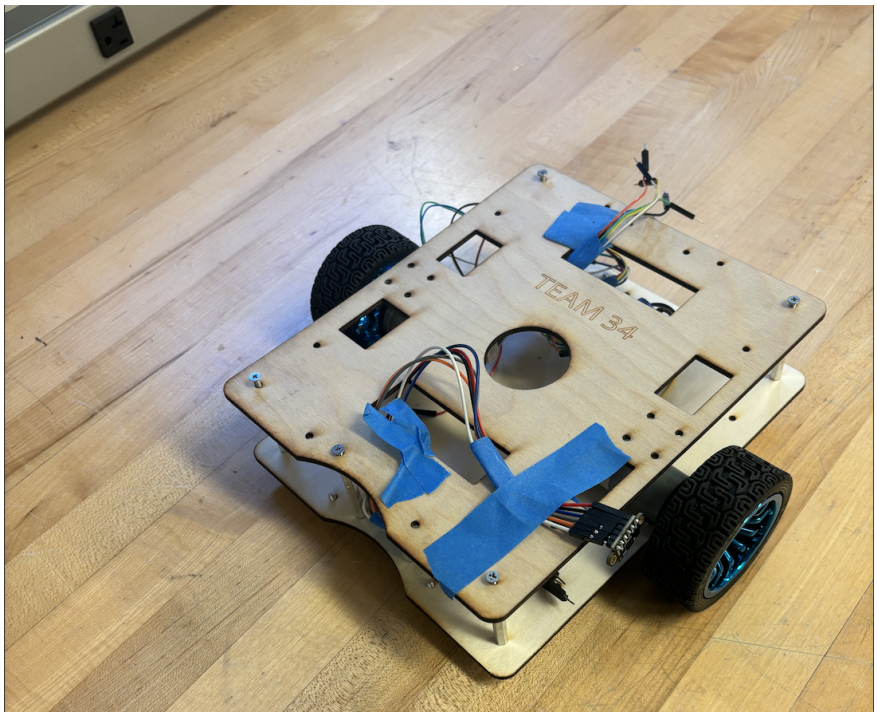
First Autonomous Robot Mechanical Drawing and measurements




TEAM 34







First Autonomous Robot Datasheets

1) [ESP32 WROOM Datasheet]

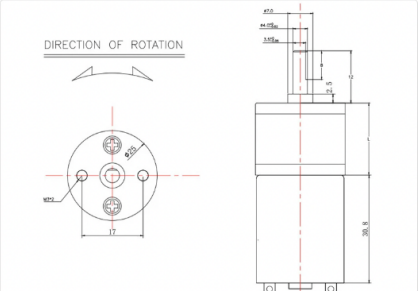
2) 12V DC Motor (GA25-370):

Specifications

- Operating voltage: between 6 V and 18 V
- Nominal voltage: 12 V
- Free-run speed at 12 V: 188 RPM
- Free-run current at 12 V: 50 mA
- Stall current at 12V: 1500 mA
- Stall torque at 12V: 5 kg.cm
- Reductor size: 19 mm
- Weight: 84 g

For other variations of this motor see the JGA25-370 gearmotor selector.

Voltage		No load		Rated Load				Stall		(L)Length
Range	Rated	Speed	Current	Speed	Current	Torque	Power	Torque	Current	
		r/min	mA	r/min	mA	kg.cm	W	kg.cm	mA	mm
3-9V	6V	1363	80	954	380	0.1	1.05	0.35	1300	18
3-9V	6V	646	80	454	380	0.22	1.05	0.76	1300	17.5
3-9V	6V	281	80	196	380	0.5	1.05	1.7	1300	19
3-9V	6V	171	80	120	380	0.8	1.05	2.8	1300	21
3-9V	6V	133	80	93	380	1	1.05	3.6	1300	21
3-9V	6V	77	80	54	380	1.8	1.05	6.2	1300	23
3-9V	6V	58	80	40	380	2.4	1.05	8.2	1300	23
3-9V	6V	35	80	25	380	4	1.05	13.6	1300	25
3-9V	6V	26	80	18	380	5.2	1.05	18	1300	25
3-9V	6V	22	80	16	380	8.7	1.05	20	1300	27
3-9V	6V	12	80	8.5	380	11.5	1.05	20	1300	27
6-18V	12V	1931	100	1370	330	0.14	2	0.5	1500	18
6-18V	12V	915	100	640	330	0.3	2	1.06	1500	17.5
6-18V	12V	400	100	280	330	0.67	2	2.35	1500	19
6-18V	12V	250	100	175	330	1.1	2	3.86	1500	21
6-18V	12V	188	100	130	330	1.43	2	5	1500	21
6-18V	12V	108	100	76	330	2.5	2	8.6	1500	23
6-18V	12V	82	100	62	330	3.3	2	11.4	1500	23
6-18V	12V	50	100	35	330	5.4	2	18.8	1500	25
6-18V	12V	37	100	26	330	7.2	2	20	1500	25
6-18V	12V	22	100	16	330	12	2	20	1500	27
6-18V	12V	17	100	12	330	16	2	20	1500	27
12-30V	24V	863	10	604	60	0.12	0.7	0.41	210	18
12-30V	24V	400	10	280	60	0.27	0.7	0.9	210	17.5
12-30V	24V	178	10	125	60	0.6	0.7	2	210	19
12-30V	24V	110	10	80	60	1	0.7	3.26	210	21
12-30V	24V	85	10	60	60	1.26	0.7	4.2	210	21
12-30V	24V	49	10	35	60	2.2	0.7	7.3	210	23
12-30V	24V	36	10	25	60	2.8	0.7	9.6	210	23
12-30V	24V	22	10	15.5	60	4.8	0.7	16	210	25
12-30V	24V	14	10	12	60	6.3	0.7	20	210	25
12-30V	24V	10	10	7	60	10.6	0.7	20	210	27
12-30V	24V	7.5	10	5	60	14	0.7	20	210	27



3) [TOF sensor (VL53L0X) Datasheet]

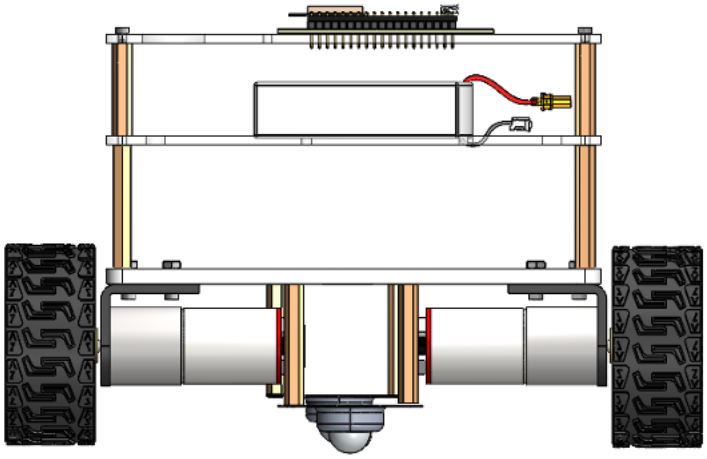
Video links of functionality

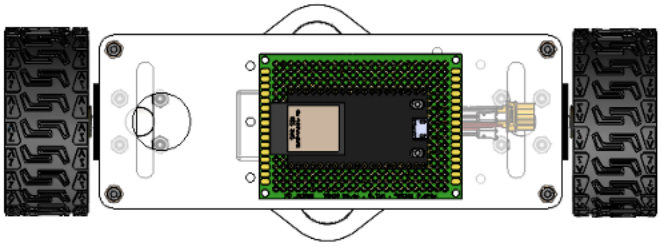
VIVE AND POLICE CIRCUIT:
https://youtu.be/gAvrapdnEAo

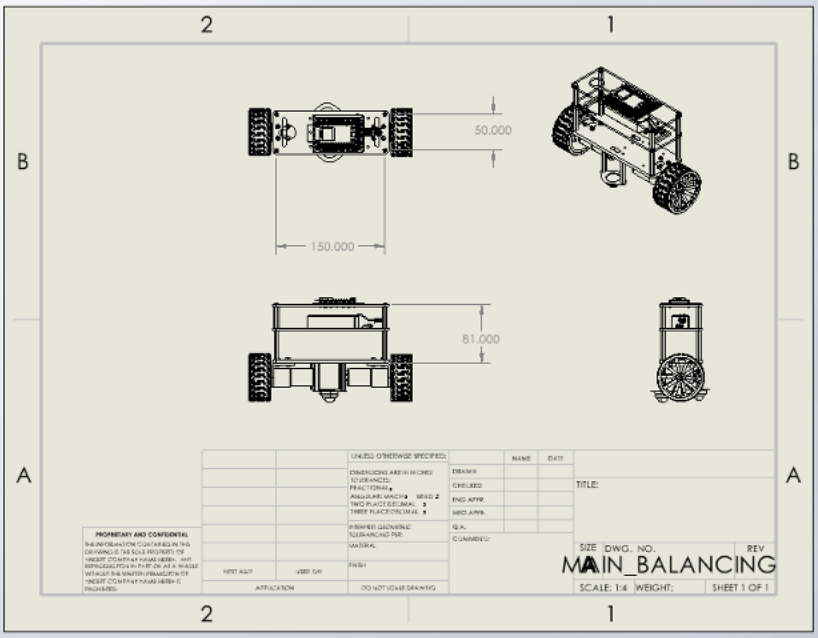
WALL FOLLOWING:

https://youtu.be/EMXnKcgKYlY

Second Autonomous Robot CAD and measurements






2
1
B
A
50.000
150.000
81.000
MAIN_BALANCING
SCALE: 1:4
WEIGHT:
SHEET 1 OF 1

BOM (Second Robot)

Sr. No.	Part Name	Qty	Cost
1	Caster Wheel	2	$2.20
2	Mobile Base	2	Laser Cut
3	ESP32 WROOM	1	$9.99
4	12V DC Motor	2	$33.86
5	SN754410NE Motor Driver	2	Mini-Store
6	LiPo Battery	1	$15.99
7	Power Bank	1	$16.19
Total			$78.23

Datasheet Used (Second Robot)

1). ESP32 WROOM:
https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf
2). 12V DC Motor:
https://www.amazon.com/GA25-370-Encoder-Stainless-Electric-Reduction/dp/B08LD2JKTY?th=1
3). Power Bank:
https://www.amazon.com/Anker-Ultra-Compact-High-Speed-VoltageBoost-Technology/dp/B07QXV6N1B/ref=sr_1_4?crid=3777WG3537YZT&keywords=power%2Bbank&qid=1703145529&sprefix=power%2B%2Caps%2C85&sr=8-4&th=1
4). LiPo Battery:
https://www.amazon.com/OVONIC-5200mAh-Connector-Airplane-Helicopter/dp/B07JJ4Q65Z/ref=sr_1_1_sspa?crid=10VV9GRY1QS9W&keywords=LiPo+battery&qid=1703144744&sprefix=lipo+battery%2Caps%2C88&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1