FINAL PROJECT REPORT SUBMISSION

MEAM 5100: Design of Mechatronics Systems

Submitted by: Aditya Jayant Ganapathiraju, Vaibhav Wanere, Andy Xiao

1 Description and Functionality of the Autonomous Bot

1.1 Performance

The evaluation for performance was done over 4 stages, 3 individual stages for Wall Following, Beacon Tracking, and Vive police car pushing, followed by a final evaluation for overall integrated performance.

Wall Following went very smoothly, even though it got stuck a few times, it reversed and continued to wall follow in one seamless motion. The use of IR-Retroreflective sensors greatly helped us in making the bot robust, quick, and easy to work with. Initially, it was a bit glitchy but it was due to the faulty microprocessor, once we switched to the ESP32-S2 Wrover, it went perfectly. You can watch our bot do wall following in this YouTube link `https://youtu.be/2iST1yge0s0`

Beacon tracking was very efficient, given our use of potentiometers instead of resistors in the circuit. This helped us reduce noise and improve detection, which made it easier for us to complete this task. It successfully detected both the 23Hz and 550Hz frequencies with ease. You can watch our bot do Beacon Tracking in this YouTube link `https://youtu.be/QMlous3lJWo`

Vive Police car push was a bit challenging but we completed it well. Firstly we detected the police car using Vive, then made sure we were getting the Vive location using our circuit. Then we wrote the logic to drive our bot towards the police car and once detected, push it 10 inches. You can watch our bot do the Vive police Car push in this YouTube link `https://youtube.com/shorts/f-kes_biBf8?feature=share`

The complete 360 of our bot outlining the modules and parts is given in this YouTube link for reference `https://youtube.com/shorts/zj8QHb2Ooqs?feature=share`

All in all, our bot was robust, quick, and simple (like the legendary MiG-21). We saw people complicate their bots too much, with ultrasonic sensors and a lot of redundant features, hence we made sure our bot did its intended task splendidly well.

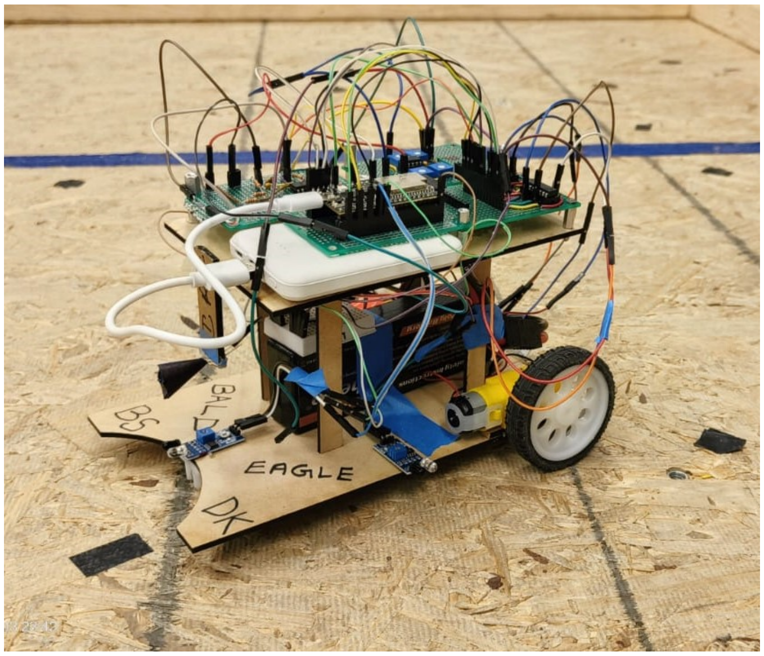

Figure 1: Team-31 Autonomous Bot

1.2 Mechanical Design

Our original chassis design had four 12V motors and wheels, intended to provide more torque and pushing power. However, as we encountered sustained purchasing delays, we decided to revert to our previous design used in lab 4: a two-wheel drive system with two TT motors, as well as one caster wheel at the front for stability. In our chassis test, we found that TT motors provided just enough torque and power for pushing the police car.

The driving performance of our vehicle is excellent. With the batteries and heavier components all in the lower space, the center of gravity is fairly low, and as a result, the drive is smooth, and turning is precise. However, the metal ball-bearing caster wheel does tend to cause issues where the ground surface is rough; it gets stuck from time to time, but with a little help it frees itself with ease.

Our material of choice is 1/8 MDF. Our vehicle consists of two distinct layers. The first layer serves as the chassis, which includes the motors, wheels, batteries, and IR sensors. The second layer is intended to house all of our circuits, including the ESP32 NodeMCU 32S control unit. The two layers were connected using four "pillars", which were also made from 1/8" MDF, and hot glued at both ends.

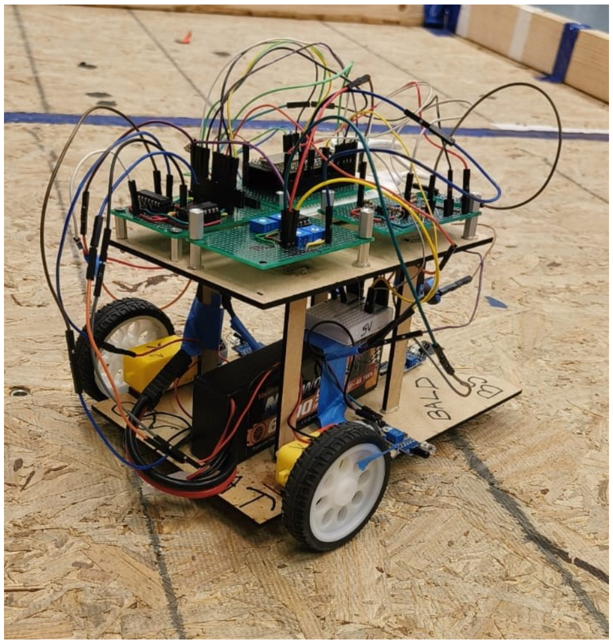

Figure 2: Top/Bottom Tier view showing components in various levels

The mobile base we used exemplifies a thoughtful engineering approach, designed to be both agile and sturdy. It employs a two-level structure, with each level dedicated to a specific functional aspect of the base's operation.

The lower level forms the base's chassis, where mobility and power distribution are centralized. It is equipped with a pair of large, white-spoked wheels on one side and one small caster wheel, it is visible at the front to provide stability and facilitate smooth directional movement. The presence of both sets of wheels ensures that the base can navigate various terrains and execute sharp turns with precision.

This level also houses the power supply components. A power bank, secured firmly onto the platform, is the primary energy source for the ESP32 microcontroller, for remote control using wifi or data transmission. Additionally, compartments designed to hold 9V batteries make up a separate power system intended for the motors used during racing. This dual-power setup is indicative of a design that prioritizes both energy efficiency and high performance.

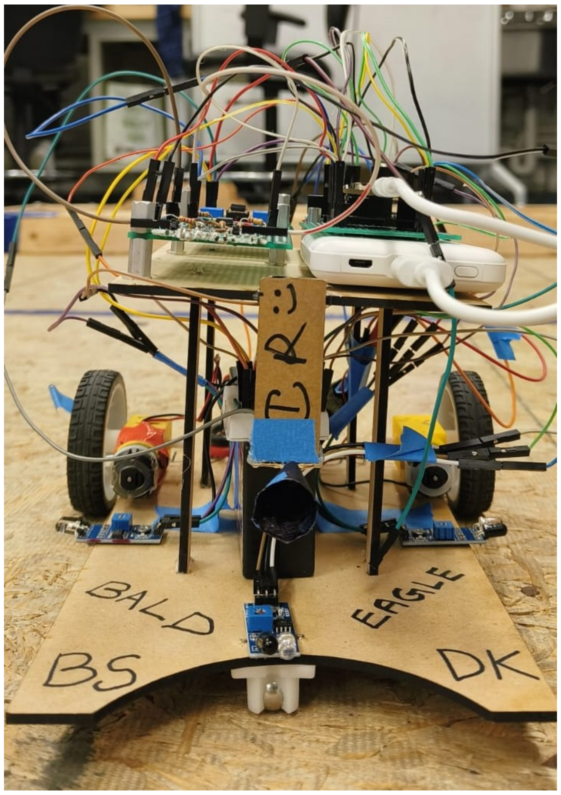

Figure 3: Front View of the bot

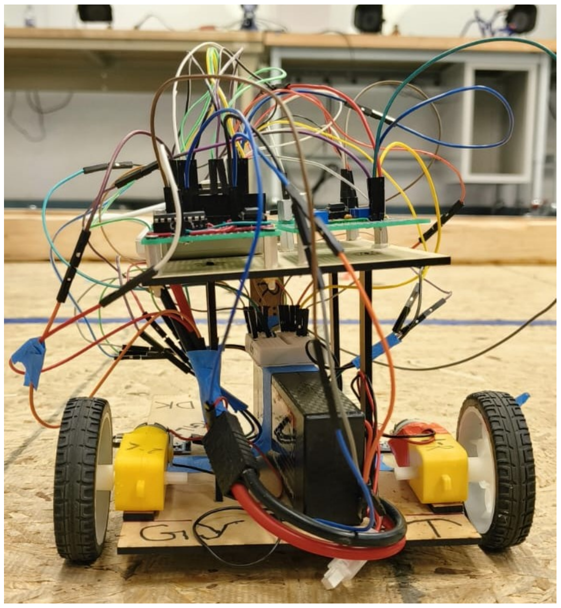

Figure 4: Back View of the bot

The upper tier, supported by a sturdy wooden framework, functions as the command center. A breadboard, meticulously wired with an array of color-coded jumpers, stands prominently, it's a sophisticated electronic design beneath its chaotic appearance. This breadboard is integral to the system, linking the ESP32 with various sensors and output devices. The strategic wiring suggests a well-planned circuit that maximizes the base's response time and reliability.

As the project transitions from the prototyping phase to a more permanent solution, the temporary setup on the breadboard is replaced by a perforated board where all components are permanently soldered into place. This transition marks the project's evolution from a concept to a finalized design, ensuring enhanced stability and reduced risk of connection failures. The perforated board, with its fixed connections, serves as a testament to the project's progression and indicates a readiness for real-world application. The soldered board embodies the final iteration of the mobile base's electronic system, promising resilience and consistent performance during operation.

The overall design reflects an educational project's prototype phase, where functionality and adaptability are paramount. The mobile base's construction demonstrates a keen awareness of mechanical dynamics and electronic integration, promising not only to fulfill its intended task but also to provide a platform for further experimentation and learning.

1.3 Electrical Design

High-Level View of Electricals:

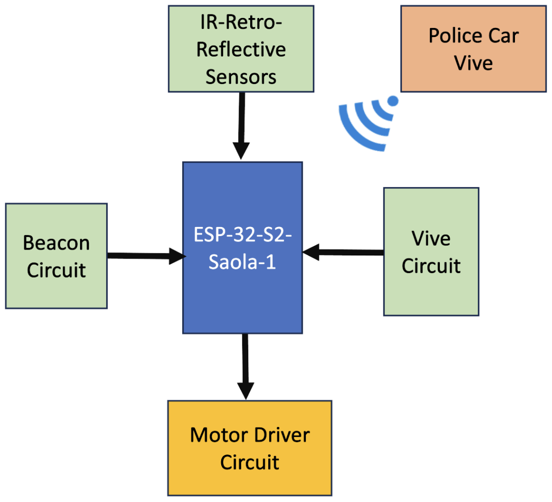

Figure 5: MCU and Peripherels

Circuits on the Robot:

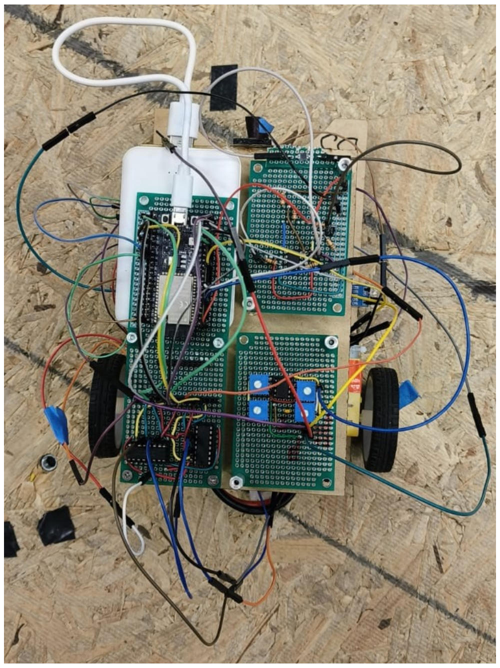

Figure 6: Compelte Soldered Electrical Circuitry of the Bot (Microprocessor, Beacon tracking circuit, Vive circuit and motor driving circuit)

We are one of the only two teams that utilized IR-retroreflective sensors to sense proximity. In hindsight, this was an excellent decision but with one slight caveat. Its advantages: are easily tunable by turning the potentiometer, precise and reliable sensing of obstacles, and ease of use. Each IR sensor only uses three connections: VCC, GND, and signal output. This means that in the grand scheme of things, jump wire connections are minimized thus making debugging and organization far more easier. Adding on to the ease of use, since the output is a simple logic HIGH or LOW, writing the wall following logic in Arduino was fairly straightforward. One slight caveat is that during testing, we found that the beacon's IR was interfering with our readings, but this did not hinder our actual performance since they could be disabled in software when not in use. Overall, choosing to use IR sensors proved a huge success.

For motor drivers, we soldered two separate SN754410 H-bridges to a perf board with male-female pins for jump wire connections. We elected to use two instead of one H-bridge in consideration of heat dissipation. We also used an IR photodiode for beacon tracking. We found the diode to be extremely sensitive to environmental light. Therefore, we have made a small light cone for the IR diode and painted it black, in an attempt to block visible light and focus light just for the diode. This proved effective, and after some fine-tuning of the amplifier circuit, we were successful in tracking the 23Hz and 550Hz beacons from a distance.

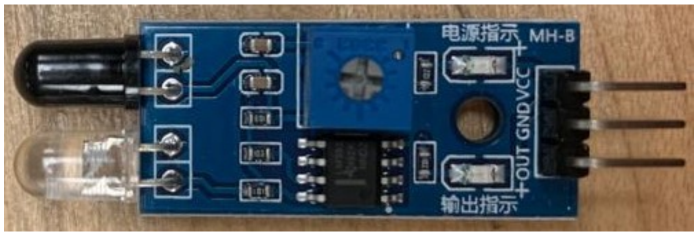

Figure 7: The efficient IR-retroreflective sensor used.

Following figure shows the Vive Coordinates as measured on the field:

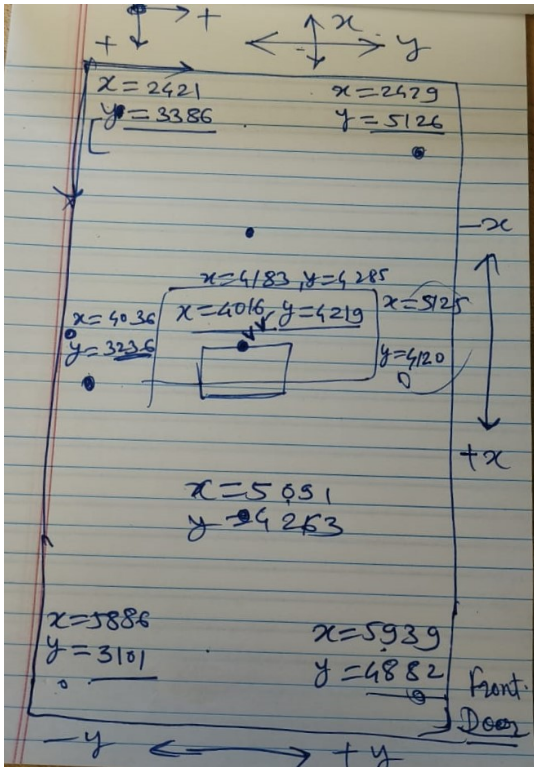

Figure 8: Vive Localization Coordinates Map used for understanding and code logic.

The vive circuit proved relatively more difficult to solder for us, but we did successfully solder a working vive circuit afterward. In our experience, the vive circuit appeared to be the second most susceptible to environmental noise (in addition to beacon tracking photodiode), as the reading may fluctuate when the vehicle is not in motion. Our team attempted to strengthen the solder points and connections on the vive circuit in order to improve noise, and the end result was deemed acceptable by our standards.

Vaibhav also had a stroke of brilliance for the IR-beacon tracking circuit where instead of resistors, he used potentiometers as the resistance can be adjusted to cope with noise or to improve tracking of the IR beacon. This helped us optimize the resistances on the track, to improve performance in real-time.

One more impressive thing is that all the circuits are completely **Soldered**. No breadboard was used, which again contributed to the robustness and efficiency of the bot.

All circuits are connected to one single MCU in our case, namely the ESP32-S2-Saola. This particular board proved reliable and provided an ample amount of GPIO pins, saving us from the hassle of communicating between two smaller C3 boards. We also fixed a tiny breadboard on top of the LiPo battery as a common 5V power supply and ground bus. All 5V power and ground connections are connected to this tiny breadboard, simplifying the debugging process and improving overall wire organization.

1.4 Software Design and Approach

Our software approach is three-fold, firstly to accomplish wall-following, beacon detection, and then police car push using Vive. The software logic is described below.

Wall Following: This is designed for a wall-following robot, employing the integration of motor control and infrared (IR) sensors. It meticulously configures motor parameters for two motors, using Pulse Width Modulation (PWM) for precise speed and direction control. The inclusion of three IR sensors, assigned to specific pins, allows the robot to detect obstacles in its path. The setup function, initializes motor and sensor systems, setting the groundwork for responsive navigation.

The core functionality lies in the loop function, where the robot continuously reads sensor inputs to navigate. The script smartly handles different obstacle scenarios: it turns left when both front and right sensors detect obstacles or keep moving forward otherwise. For prolonged detection of an obstacle on the right, it initiates a left turn after a set duration, showcasing adaptive behavior. Motor control is achieved through well-defined functions like `drive_forward, drive_left`, etc., which manipulate the PWM duty cycle and motor direction pins. It can even smartly know if it's stuck, if jammed in a place for more than 5 seconds, it reverses and continues the process of wall-following. We initially tried normal steering but switched to neutral steering as it was quick and more efficient.

This approach demonstrates a blend of responsive sensor-based decision-making with efficient motor control, enabling the robot to navigate through environments with varying obstacle configurations smoothly and safely. The code is structured to ensure continuous movement, avoiding situations where the robot might get stuck, highlighting its practicality in real-world applications. This code is important to get right as our navigation for beacon tracking and Vive Police car requires motor-driving and obstacle-detection logic.

Beacon Detection: This outlines a beacon tracking system for a robotic device, combining motor control and sensor integration with Arduino programming. It establishes a sophisticated motor control setup for two motors using PWM, ensuring precise speed and direction control. The definition of IR sensor pins and a beacon pin indicates the use of sensors for environmental interaction.

The setup function initializes serial communication, configures motor and sensor pins, and sets up PWM channels for motor control. This foundation is crucial for the subsequent beacon tracking functionality. The script also includes a comprehensive set of motor control functions (`drive_forward, drive_left`, etc.), allowing for versatile movement based on sensor inputs and program logic.

Central to the script is the `detectFrequency` function, which measures the frequency of the signal from the beacon sensor. This function allows the robot to identify specific frequencies and react accordingly. The `headToBeacon` function dictates the robot's response to the detected beacon frequencies, either moving towards the beacon or altering its path if the beacon is not detected. In the main loop, the robot continuously checks for beacon signals. The loop is designed to adapt the robot's behavior in real time based on the frequency detected, demonstrating a focus on target-oriented navigation. When a specific beacon frequency is detected, the robot moves forward; otherwise, it executes a left turn (using neutral steer) and stops, indicating a search for the beacon.

This code tells a responsive and adaptive approach to robotic navigation, where the robot is not only aware of its immediate surroundings through IR sensors but also guided by a more distant target represented by the beacon. The integration of beacon tracking with obstacle detection and motor control reflects a sophisticated approach to autonomous navigation, suitable for environments where the robot must locate and move toward a specific point or object.

VIVE Police Car Push: Firstly we got the police car location using the `Robot Vive Location` script. This is a sophisticated integration of the HTC Vive's tracking capabilities with a robotic system, focusing on transmitting positional data over a network. It uses the Vive510 library to interface with the Vive tracking system, receiving coordinates through a specific signal pin. The script sets up WiFi communication, indicating its intent to transmit data, possibly for remote processing or monitoring.

The `Pushing Police Car` function in the script is a focused robotic application designed for navigating towards and interacting with a target, represented by the coordinates of a "police car". This target's coordinates are received through network communication, possibly as part of a larger, interactive system or game. The script is well-equipped with motor control and sensor technology to achieve this goal.

- Motor Control and Configuration: The script sets up two motors using PWM for precise control over their speed and direction. The constants `LEDC_CHANNEL_LEFT` and `LEDC_CHANNEL_RIGHT` are designated for the left and right motors, respectively, ensuring individual control of each motor. This setup allows the robot to execute complex maneuvers such as moving forward, backward, and turning left or right.
- Sensor Integration: The robot utilizes three IR sensors, connected to defined pins, to detect obstacles. These sensors play a crucial role in navigating the robot by providing real-time data about the immediate surroundings, enabling the robot to avoid collisions while moving towards its target.
- Networking and Data Reception: The script uses WiFi communication to receive the coordinates of the police car. The handleUDPServer function is responsible for handling incoming UDP packets, extracting the X and Y coordinates of the police car, and storing them in global variables. This networked approach suggests that the robot's actions are part of a larger system, possibly involving multiple robots or interaction with a virtual environment.
- Targeted Movement Logic: Once the robot receives the coordinates of the police car, it uses its motor functions and tracking data to navigate toward these coordinates. The script includes logic to compare the robot's current position with the target's position, adjusting its movement accordingly. For instance, if the robot is to the left of the police car, it will execute a series of movements, such as turning right and moving forward, to align itself with the target.
- Adaptive Navigation: The robot's movement strategy is adaptive, changing based on its relative position to the target. This adaptiveness is crucial for real-world applications where the robot must deal with dynamic environments and continuously adjust its path.

In summary, the "Pushing Police Car" function demonstrates a comprehensive and sophisticated approach to robotic navigation. It integrates precise motor control, sensor-based obstacle detection, and networked communication to perform a specific task within a potentially interactive and dynamic environment.

Complete Code submitted to Gradescope. Please refer to the files submitted for the wall following, beacon detection, and Vive police car push.

1.5 Retrospective

The time spent in hours/days for the final project is:
Wall Following: 2-days
Beacon Detection: 1-day
Vive Police Push: 3-days

The total is 6 days for this lab. We feel time management and optimization are required for this course, and this course also taught us a great all-around background in mechatronics, electronics, and integration. The best part was the Waldo HW, Lab-4, and the final project. Lab 2 was the toughest, it took a lot of time and effort to get it done. The only feedback we have for this course is time optimization and lab occupancy issues as the lab is always crowded and the equipment is also dated.

2 Appendix

2.1 Bill of materials

Sr. No.	Part Name	Qty	Cost
1	Mobile Base	1	Laser Cut
2	Yellow DC Motors	2	$ 8
3	Big Battery 7.4V 4500mah	1	$ From Lab 4
4	Miady Power Bank	1	$ 9.39
5	Wheels	2	$ 9.07
7	Caster Wheel	1	$ 2
8	Sensors IR Retro-reflective	4	$15
9	ESP32-S2	1	$ 15
10	SN754410NE Motor Driver	2	Mini-Store

Table 1: Bill of Materials

2.2 Circuit Schematics

Schematics of all electronic circuits are down below:

2.2.1 Beacon

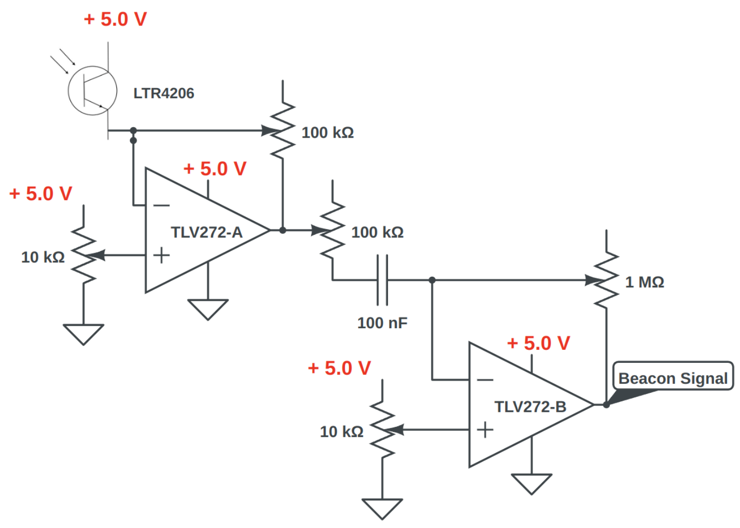

Figure 9: Circuit for the Robot

2.2.2 Vive

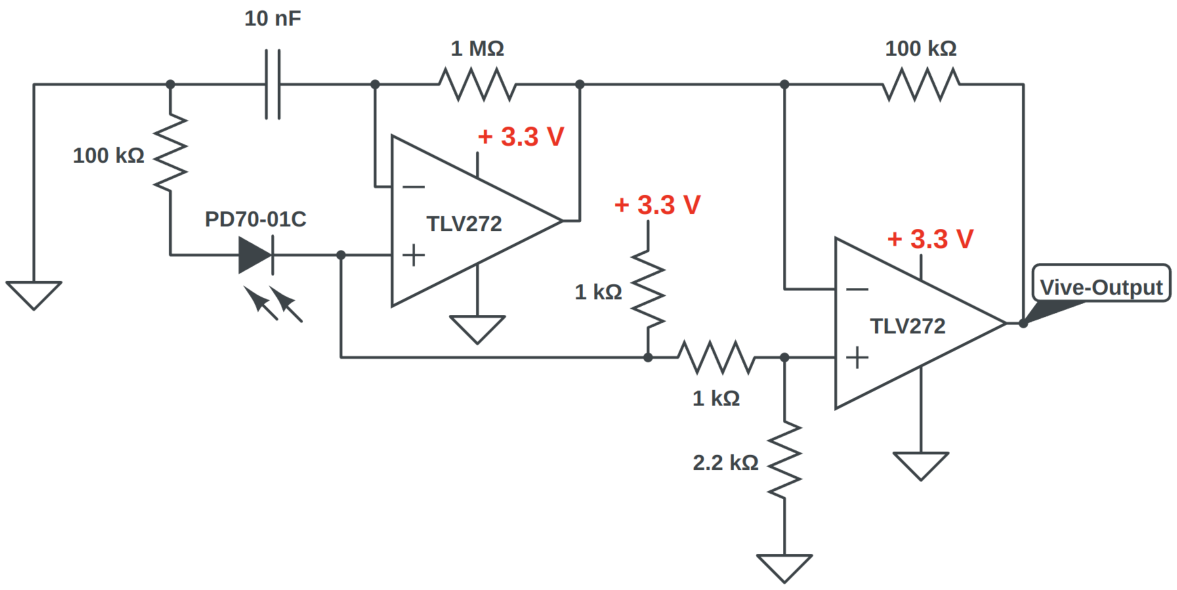

Figure 10: Circuit for the Robot

2.2.3 Motor Driver

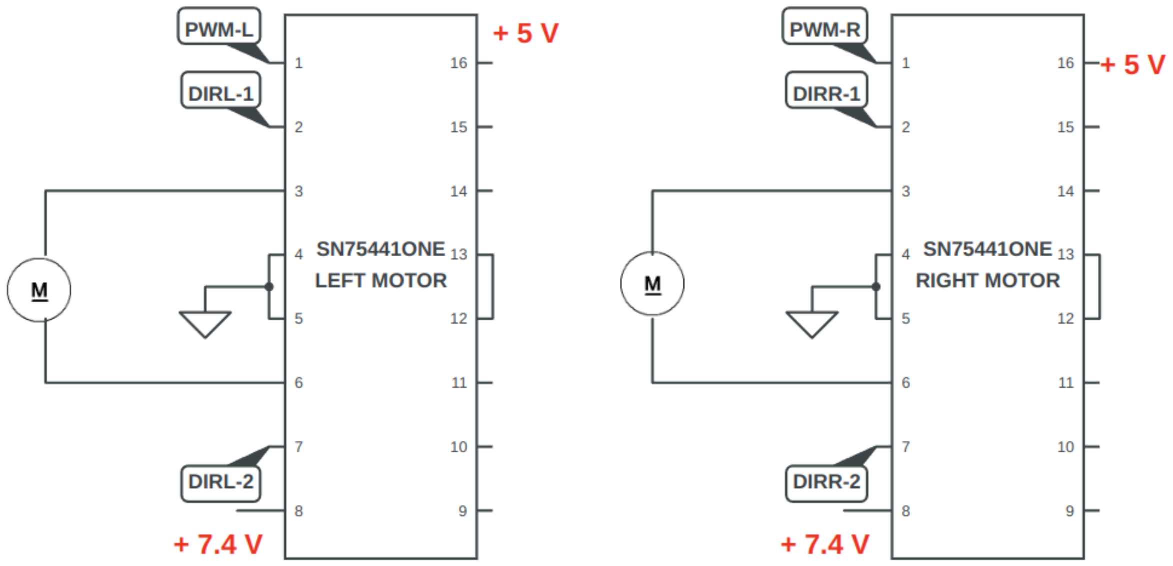

Figure 11: Circuit for the Robot

2.2.4 Complete Circuit

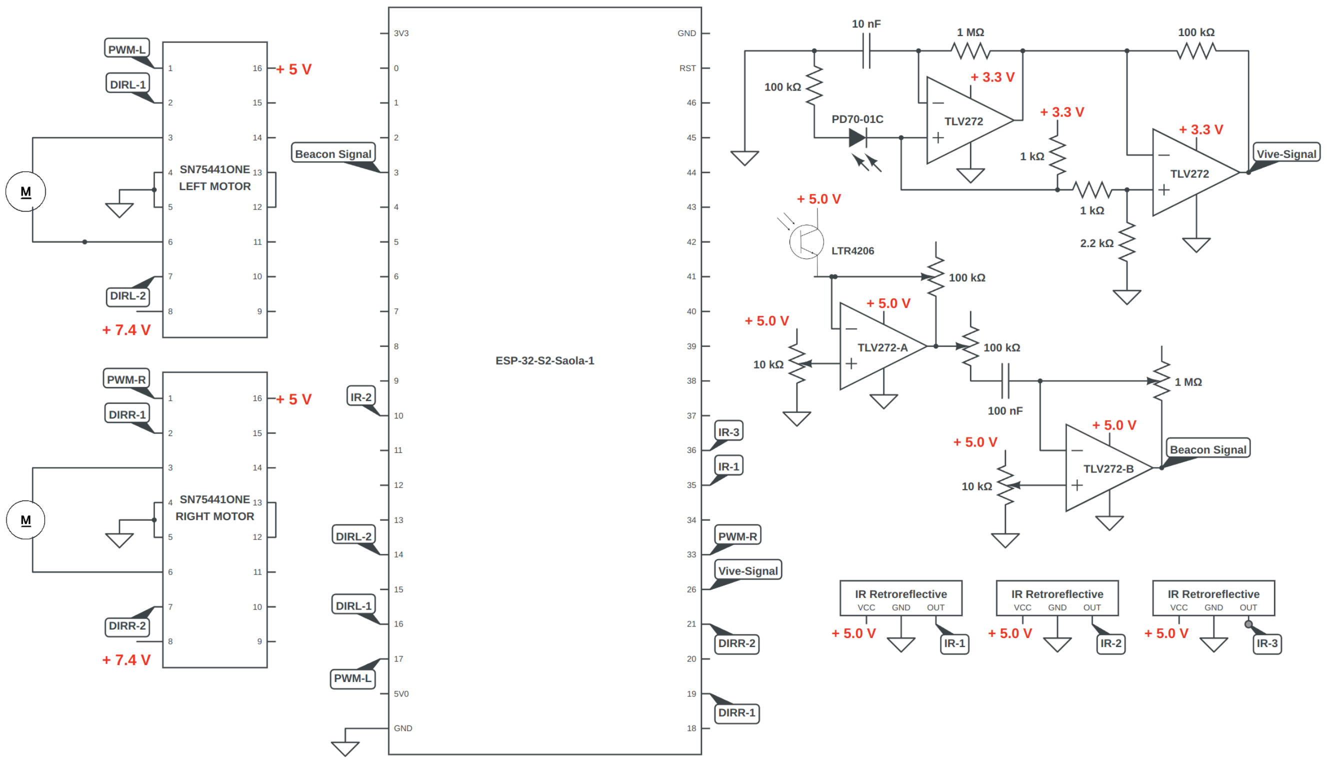

Figure 12: Circuit for the Robot

2.3 Complete Image and Rendering of the bot

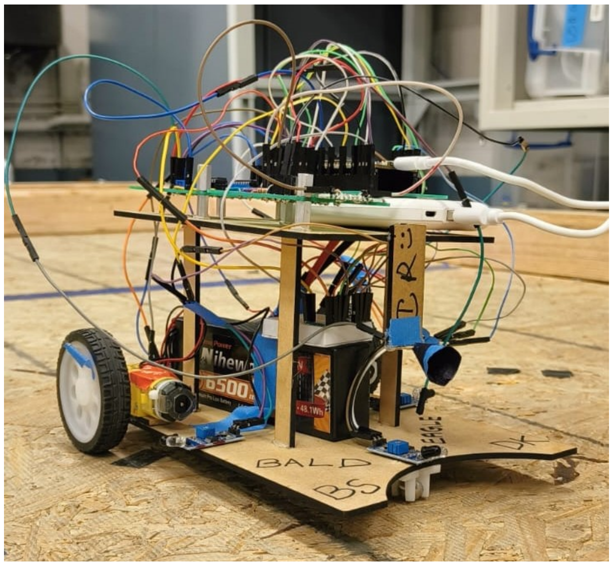

Figure 13: Final Autonomous Bot for our team.

2.4 Data-sheets of externally sourced components

We only sourced the IR-retroreflective sensors externally from Amazon, a guide for them is given in this link IR Retroreflective sensor datasheet.

The link for purchasing and the data specifications sheet is here https://www.amazon.com/DAOKI-Infrared-Obstacle-Av dp/B00XAGSWR4/ref=sr_1_3?keywords=ir+retroreflective+sensor+for+robot&sr=8-3

2.5 Videos of Functionality of the Bot

All links to videos of functionalities are given below:

You can watch our bot do wall following in this YouTube link https://youtu.be/2iST1yge0s0

You can watch our bot do Beacon Tracking in this YouTube link https://youtu.be/QMlous3lJWo

You can watch our bot do the Vive police Car push in this YouTube link https://youtube.com/shorts/f-kes_biBf8?feature=share

The complete 360 of our bot outlining the modules and parts is given in this YouTube link for reference https://youtube.com/shorts/zj8QHb2Ooqs?feature=share

3 References

My team members were Vaibhav and Andy. It was my pleasure working with them, great folks! My thanks to the TAs who helped me. Thanks for the leniency and support, especially with the motor procurement issues.