AutoServe

By

Ethan Jiang

Johan Martinez

Nikhil Vishnoi

Final Report for ECE 445, Senior Design, Spring 2026

TA: Po-Jen

06 May 2026

Project No. 84

Abstract

This project presents the design of a compact, 3-wheel rear-drive autonomous delivery robot intended for indoor navigation in multi-floor environments. The overall system measures approximately 1 ft^2, and weighs under 5 lbs. The robot connects to a PC that runs a Python-based GUI via WiFi, allowing user customization, map/path-planning and commands to be sent through wireless monitoring from the host. Additionally, the robot is equipped with a z-axis pressure sensor that enables multi-floor level detection, which the GUI uses to send different path commands corresponding to each building floor and allows elevator-based floor transitions to behave appropriately. Lastly, to address safety and ethical concerns we incorporated a front-mounted time-of-flight sensor that checks for a clear path before allowing for the motors to run, preventing the risk of collisions with both objects and humans. The final built system demonstrates a low-cost, adaptable autonomous delivery robot that combines several synergized features with wireless control and sensing, with test results showing proof that our concept can be scalable for further development in the future.

Contents

1. Introduction

1.1 Purpose/Problem

In hospitality-related environments such as hotels, guests or residents often request small amenities such as snacks, toiletries, chargers, and more to be delivered to their rooms. Fulfilling these repetitive requests requires manual labor that is time-consuming, inefficient, and tedious. Small deliveries are especially inconvenient during peak and off-hours of operation, where staff shortages cause increased labor costs. Reliance on manual delivery reduces overall service efficiency and customer satisfaction.

While automated service robots exist, current commercial solutions are extremely expensive and often impractical for smaller deployments or retrofitting existing buildings. According to the International Federation of Robotics, "With 102,900 units (+14%) sold in 2024, more than every other professional service robot was built for the application class transportation and logistics," highlighting the rapid growth of demand for mobile robots designed for supplementing service objectives. While similar industries such as restaurants have begun to implement more widespread technology with service robots, hospitality facilities have not integrated changes yet due to a lack of solutions that are useful and justifiable in cost. There is a need for an affordable, flexible indoor delivery system capable of autonomously transporting small items within multi floor buildings while operating within existing building infrastructure constraints. These robots can also have several positive societal implications such as reducing worker fatigue, tackling workforce staffing shortages, and minimizing response times for operational efficiency.

1.2 Solution

This project addresses the proposed solution for an adaptable autonomous delivery system. The navigational robot itself is capable of transporting small loads between dispatch and user-defined destinations in a multi-floor building using a precomputed shortest path. The host computer connects and monitors the robot in real time wirelessly through a WiFi connection, and a Python graphical user interface allows intuitive operation with numerous customizable features. To mimic realistic scenarios such as delivering an item from a hotel lobby desk or snack bar to a room upstairs, the robot autonomously navigates hallways with encoders tracking movement and various onboard sensors to detect floor-level or impeding obstacles. The implementation of forward-facing proximity sensing ensures safe operation in dynamic environments. Elevator actuation is assumed to be externally triggered by the building staff as is currently most common in real service situations, while the robot will autonomously handle entering, riding, and exiting the elevator at the correct floor with sensor detection (explained in Design section).

The navigational platform is designed with our custom PCB that connects all of the sensor subsystems with an ESP32 microcontroller communicating and transferring data to be interpreted by our host PC's GUI. Two brushless motors operated in rear-wheel drive orientation with built-in encoders move the robot while accurately assessing its position. The overall system focuses on reliability, cost-effectiveness, and flexibility, demonstrating a practical approach to automating routine delivery tasks in various service environments.

1.3 Visual Aid

Figure 1: Conceptual diagram explaining high-level interaction between robot system and destination.

1.3 High Level Requirements

1. **Delivery:** Robot should be connected to computer via wifi sending back information about current task and taking commands to start deliveries.
2. **Maneuverability:** Robot is able to detect and maneuver around simple obstacles within 15 centimeters (ex. person blocking a straight path).
3. **Reliability**: The robot is able to track position accurately and arrive at a location and return within 10 minutes, having a battery life that can last the duration of delivery.

2 Design

2.1 Physical Design

The picture below models a general sketch of what our physical navigational robot looks like. Key differences between our original proposal design and our final physical design include the removal of 1 caster wheel and reversed forward orientation. These changes resulted from discoveries following in-depth testing of the physical movement that the two caster wheels were uneven in height, and that dragging them with wheels in the front caused angled drifting and a plethora of weight/balance related issues. The two driver wheels are 6 inches in diameter with a wheelbase length of 13 inches, connected to Hall encoder DC motors (B07GNGQ24C) attached to brackets underneath the lower platform base. Our circuit board and other electrical components such as the battery are all housed on the lower base platform with the ToF sensor mounted on a servo motor oriented to face directly forward. We utilized the ECEB Machine Shop to construct the physical structure and platforms. The dimensions of both platforms are similar, approximately 13x15 inches with the upper carrier raised about 6 inches above.

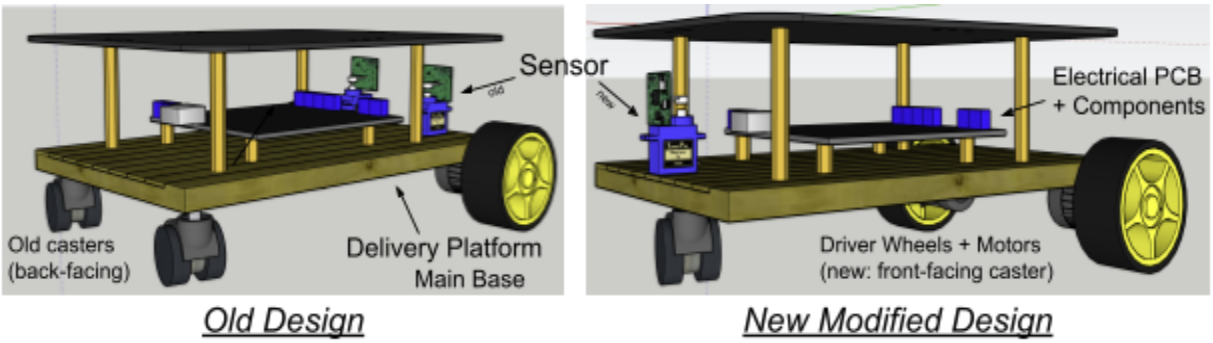

Figure 2: Contrasting old and new 3D models of our general physical navigational robot design.

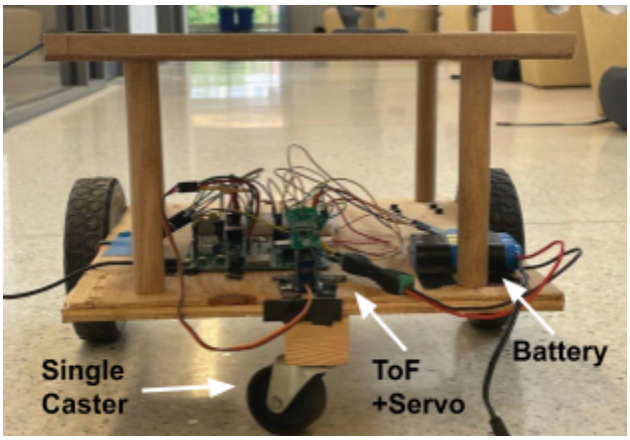

Figure 3: Image depicting our final achieved navigational robot design.

2.2 Block Diagram

Our proposed block diagram and final block diagram is shown in Figures 4 and 5 respectively. The main differences between the two designs are as follows: replaced Buck converter with LDO in power subsystem because it did not need high current output to power on a raspberry pi; removed the IR

sensor because breadboard testing found it was unable to meet our requirements for sensing; removed the IMU because when soldered into the PCB it did not boot up on the I2C line. Additionally we integrated a Servo to sweep the toF breakout between different angles to get more than 1D data for the robot.

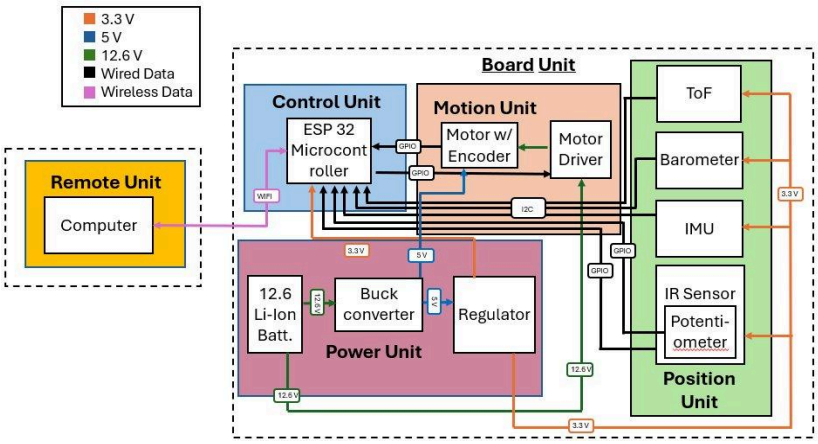

Figure 4. Initial Block Diagram

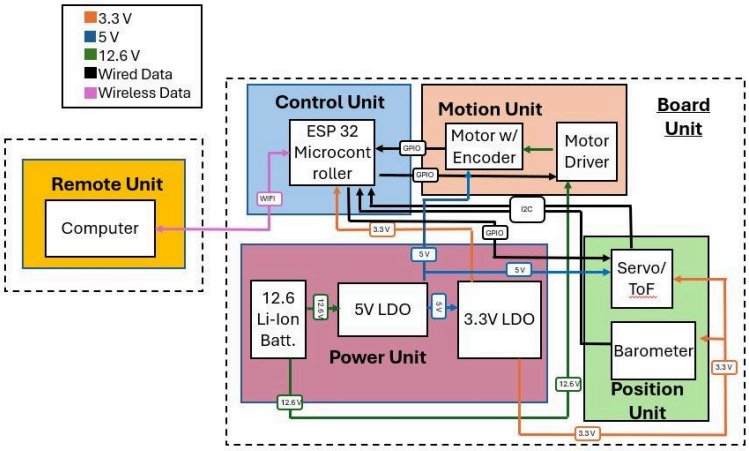

Figure 5. Final Block Diagram

2.3 Motion Subsystem

The motion subsystem of our robot handles all of our robot's movement: moving forward, backward, and turning left and right. Our full-motion system includes two brushless motors, one caster wheel, two encoders, and two motor drivers.

2.3.1 Driver

We used two LMD18245 motor drivers for our brushless DC motors. These drivers were chosen because they were readily available. Figure 7 shows a schematic of the motor driver circuit. We tested this

configuration on the breadboard and were able to drive our motors. The resistor on the CS_OUT pin needed to be tweaked during testing to the value that allowed high enough current flow. The RC pin is used to set the Monostable value enabling the DIR pin to set the current flow between OUT1 and OUT2. Figure 6 shows the different operating modes of this motor driver based on the digital logic pins.

SWITCH CONTROL LOGIC TRUTH TABLE(1)

BRAKE	DIRECTION	MONO	Active Switches
H	X	X	Source 1, Source 2
L	H	L	Source 2
L	H	H	Source 2, Sink 1
L	L	L	Source 1
L	L	H	Source 1, Sink 2

Figure 6. Motor Driver Direction

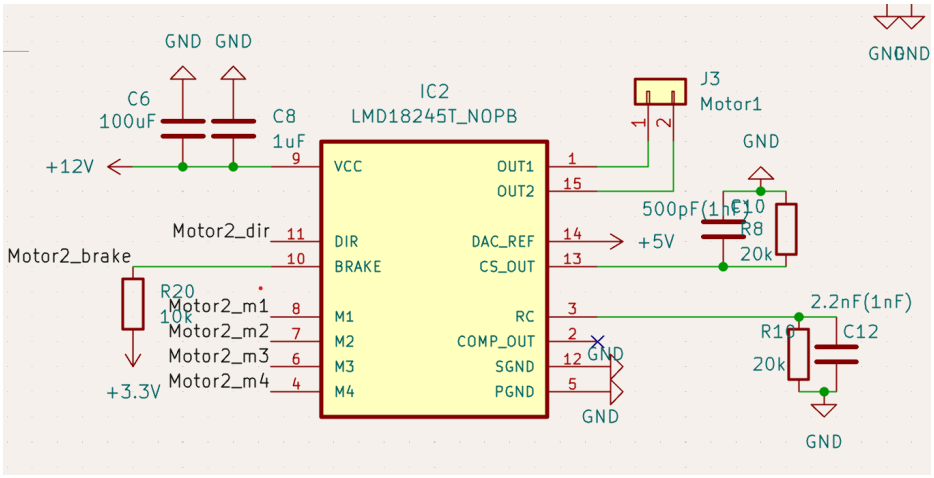

Figure 7. Motor Driver Circuit

The m1-m4 pins on the driver enabled some speed control using a 4-bit built-in DAC, had a direction pin, and had a brake pin to immediately stop the motor. The speed of the motor was controlled using a 4-bit DAC and a sensing resistor using the following:

$$I_{motor} = \frac{(V_{DAC\,REF} \times \frac{D}{16})}{250{*}10^{-6} \times R_s}\,[A] \qquad (2.1)$$

where I_{motor} is the current going to the motor, D is the decimal equivalent of the 4-bit DAC number M_4-M_1, where M_4 is the most significant bit, and R_s is the sensing current. The driver, therefore, allows us

to have 16 different speeds. From our test, we soon found out that the same DAC values gave us similar but not identical motor speeds. As a result, our robot was not capable of moving forward. A PWM onto the BRAKE pin was implemented to limit the power going into the motors to get them to move at the same speed. From our tests, when D = 4 and $R_s = 1k\Omega$, motor 1 had 36.6 mA going through it and motor 2 had 38.1 mA. If we want the motors to move at the same speed, the average power going to them had to be the same

$$P_{avg} = P \times D_c [W] \quad (2.2)$$

P_{avg} is the average power; P is the DC power, if no PWM were present; and D_c is the duty cycle of our PWM. Because the ratio of currents is 1.04, the ratio of motor duty cycles must be 1.04^2 because DC power is proportional to DC current squared. This method gave us decently straight motor movement.

2.3.2 Encoder

The encoders were included in the purchased motors and outputted a recordable signal enabling us to track how far the motors have traveled. Every rising edge signal from the A pin of an encoder resulted in a step of 0.667 degrees in physical motor rotation, or a movement of 0.035 inches in translational motion. The ESP32 would keep track of the motor movement and would stop the motor after 15 encoder ticks, or 0.52 inches, to detect if anything was in front of it using the IR sensor so that it could move without bumping into anything. This resulted in a continuous pulse-like movement.

2.4 Sensing Subsystem

When trying to detect objects, we tried implementing two forms of sensing devices: a custom-built IR sensor and a time-of-flight (ToF) sensor. Our custom IR sensor did not function as intended, but it is worth noting how we tried implementing it. Because the IR sensor failed, we resorted to only using the ToF sensor to detect an object's distance. This subsystem is mainly of I2C devices, Servo plugins and a Sonar module plugin (did not end up being used since toF worked).

2.4.1 Pressure Sensor

A BMP 585 pressure sensor was connected to our MCU in I2C mode by setting the corresponding pins on the IC.

2.4.2 Time-of-Flight (ToF) Sensor

For detection we controlled a VL53L3CXV0DH with the I2C protocol. It calculates the distance by multiplying the time it takes for an emitted pulse of infrared light to come back to the IR sensor with the speed of light and dividing it by two. The emitter is facing the z normal of this chip which required us to create a breakout board for the chip so we can mount it where needed. The breakout board (Fig 8) is wired to the ESP32 on the main PCB. The breakout is mounted on a Servo driven by the ESP for scanning.

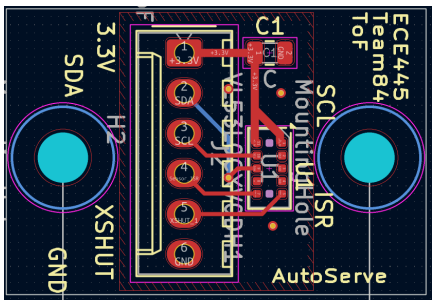

Figure 8. toF breakout board

2.5 Power Subsystem

The power subsystem is very simple and important, providing the system with reliable power that will not damage components. In the routing part of the PCB the traces were wider (20millis) for the 5 Volt and 12 Volt signals allowing higher current flow. The Servo 5 Volt is separate from the 5 Volt for the rest of the system since they may draw a too high current which could stop our ESP from running.

2.6 Control Subsystem

The MCU we chose to use is the ESP32-S3-WROOM–1 because it had wifi capabilities, ease of programmability and there were dev kits readily available in the lab room. Figure 9 shows the schematic of the ESP32 and the components added for programming. On the left the IO pins are wired to net names corresponding to the nets shown for the other subsystems. The I2C lines have two pull up resistors valued at 5kOhm to stabilize the signal between all devices.

The wiring on the right was grabbed from the ECE 445 class wiki and involved components for many different programming modes. The first mode is through the native USB on the IC chip itself by connecting straight to the USB_B_Micro plug. To be in this mode GPIO0 and CHIP_PU pins need to only be set up with the push button, pull up and debouncer capacitor. The second mode is through the USB-UART programmer and requires the DTR and RTS signals to connect to the transistors to set Chip_PU and GPIO0. When we were first trying to program the ESP we were unable to connect with either the USB or with the UART programmer. We also had a JTAG plug in shown on the right that we purchased to try to program the ESP. In the end discovered the issue with the ESP32 being the GPIO46 strapping pin held at 3.3 Volts because that sets the ESP to block flashing only sending data over.

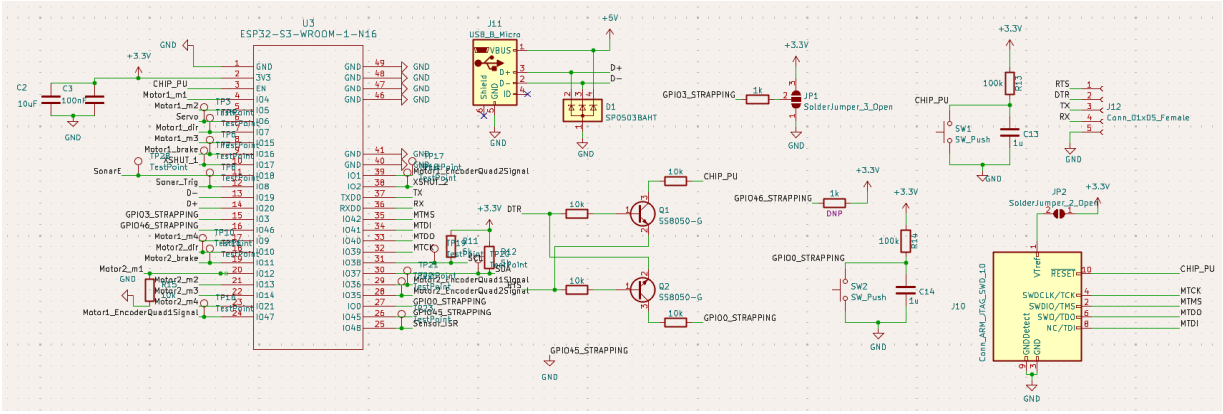

Figure 9. Control Subsystem Schematic

2.7 Software Design

2.7.1 ESP32 and Devices

The software on the ESP32 is programmed using freeRTOS. The threads are broken up into: a Wifi Softapp, telnet rx/tx, shell, motor task, tof task, elevation task and a servo task. The Wifi is set up using the provided ESP wifi software and the telnet creates a connection on 192.168.4.1:23. The main job of the telnet is to give the shell an interface that it can send and receive characters from using a UART based command structure. Each task thread gave each device their own thread to run allowing devices to share processing power. For early testing the shell gave an interface we could use with putty (Fig 10)

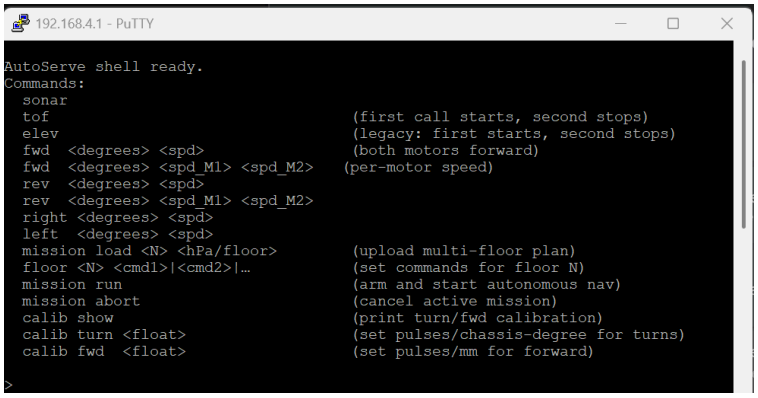

Figure 10. Putty interface for early testing

The devices are split between different control protocols. Direct IO pins are used to control and read from the Motor Drivers, Motor Encoders and Sonar. A I2C bus is used to communicate with the toF (0x29) and BMP585 (0x46). As previously mentioned the Motor Driver's brake pin is set up using a PWM. We utilized the ESP32's dedicated motor PWM which offers us the ability to change the duty cycle in a isr safe method. This is important because the Motor Encoder pins when armed signal a isr to count rotation pulses. Inside of the encoder isr once the motor has hit it's targeted angle it stops right away which results in the cleanest stop motion. Since the robot was sliding the encoder's were not able to

measure it and we were unable to add a proper corrective algorithm to compensate. As for the toF and elevation the software mainly served as an API wrapper allowing us to call simpler commands to get data from our sensors. The Servo task just swept between different pwm angles at a fixed rate to create a constant rotation.

2.7.2 GUI & A*

The system is operated with a Python-based Graphical User Interface (GUI), which handles all of the mission configurations, path planning and live monitoring with an intuitive visual layout. The GUI communicates with the robot over the same wireless telnet WiFi protocol, transmitting sequences of instructions for navigation. The host computer handles generation of paths which reduces computational demand on the embedded circuit microcontroller. The program allows users to define different layouts specific to each floor of a building, including adjusting dimensions, obstacle placements, and even the height threshold for new floor detection. For each map, tile distances can be rescaled, movement speed can be adjusted, and starting orientation can be toggled. We incorporated as many customizable modifications as possible in order to support flexibility when deploying across new environments.

The path-planning algorithm uses an A* search, which computes the most efficient route from the starting tile to the end tile whose location is indicated by the user. The algorithm evaluates neighboring nodes through a cost function (2.3) that considers a heuristic cost to destination estimate, h(n), and a cumulative cost sum, g(n).

$$f(n) = g(n) + h(n) \tag{2.3}$$

The algorithm avoids the tiled obstacles in the map and generates a path which is converted into directional motion commands for the robot in the same format as the terminal commands. Once the mission is executed, the robot validates the starting height and location, executes motions and turns sequentially, then polls for the height threshold all while the GUI monitors the status of each process in real time. Safety measures include popups for invalid dimensions/paths, and a remote “abort mission” button on the live monitor which can immediately stop the robot.

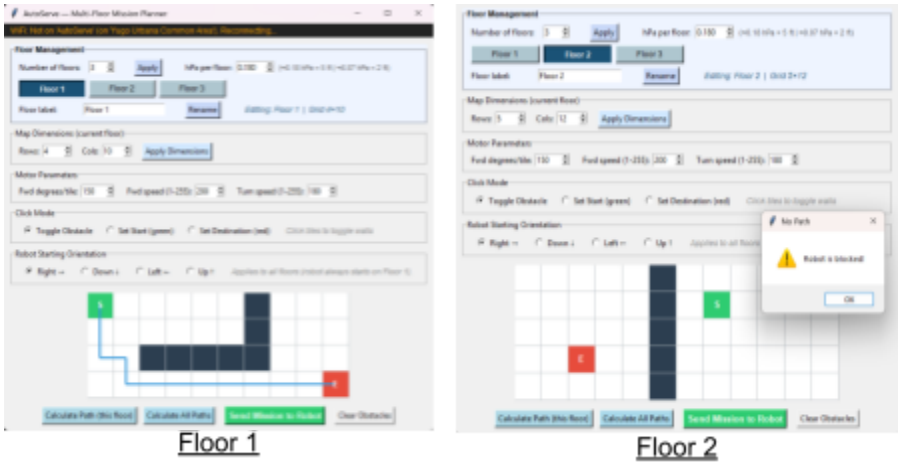

Figure 11: Side-by-side view of two floor maps running A paths on the same program launch.*

3. Design Verification

To verify our design a lot of qualitative experiments were run testing different variables since a big challenge we faced was our wheels being uneven resulting in our quantitative data being inconsistent since the sensors could not measure sliding. The main PCB Power and Control subsystem were verified after probing the put together PCB to verify correct voltage levels and ability to program ESP32 and drive the proper IO pins to control each device. This was verified using testpoints placed on all crucial logical signals. Figure X shows an image of our put together PCB (the breadboard contains a voltage divider to protect the esp32 from the 5V encoder output data).

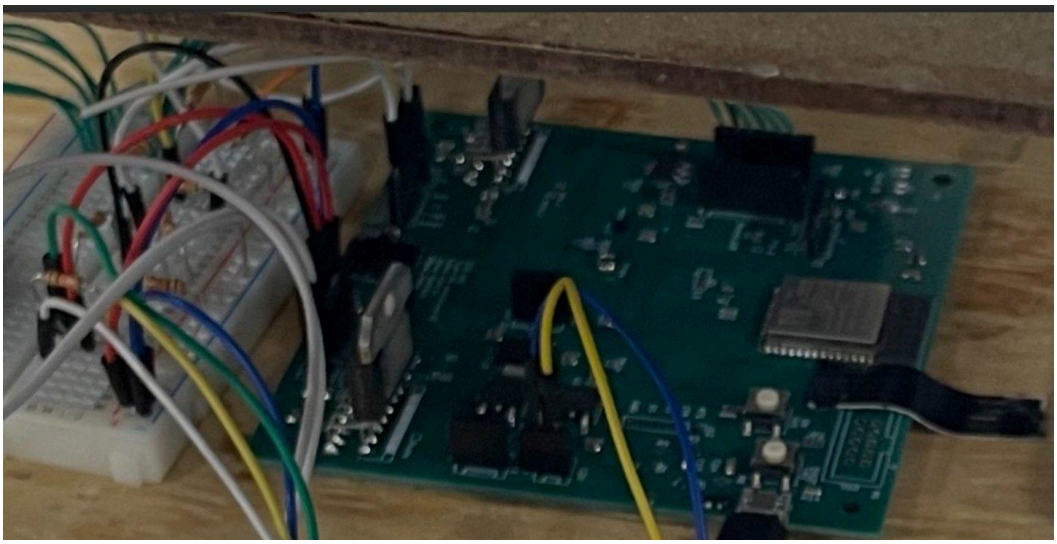

Figure 12: Image of Assembled PCB on the Platform.

3.1 Motion Subsystem

When driving the robot in the ground initial tests involved moving by small angle steps and checking the encoder readings to verify the motion. Figure X shows the tracking of the motor encoder pulses when moving the motors to target 36 degrees. It is broken down into 10 degrees steps and can see how there is a consistent small error reading in M2 which correlates to the right wheel. Additionally, it can be seen that the time for the next steps are smaller than the first step indicating that there are mechanical issues in the force for starting up the robot that also caused the robot problems.

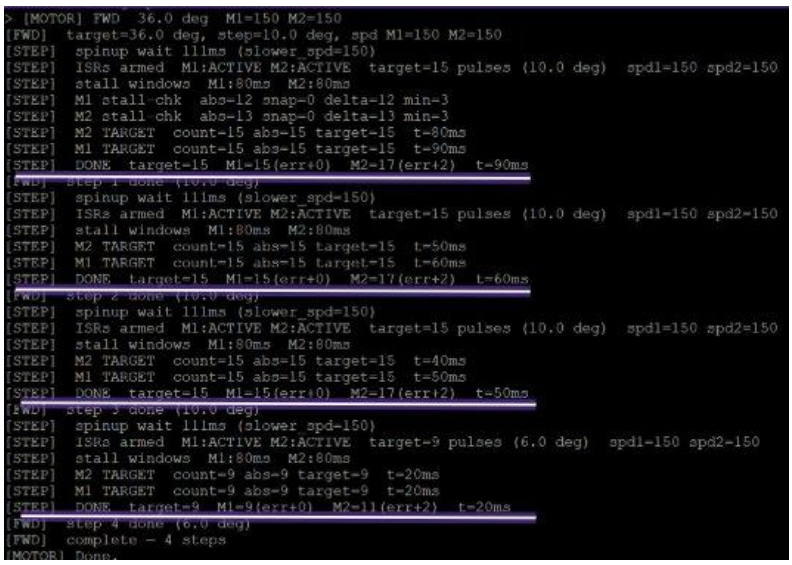

Figure 13. Motor Encoder Verification

To attempt to fix the drifting we searched different speed combinations of the motors that resulted in the motor going straight. We found that the speed of the right one (150) and the left one (163) resulted in straight motion. Additionally, we found when we did not pulse the motion and instead moved it constantly, we would go straight, indicating that our main issue was the slipping caused by poor mechanical design regarding the wheel system.

Test	Direction	Motor 1 L	Motor 2 R	deg
1	fwd.	100	100 Faster	360
4	fwd.	150 faster	50	360
5	fwd.	150 faster	75	360
6	fwd.	200 faster	125	360
7	fwd.	163 faster	150	60
8	fwd.	163	150 faster	60
9	fwd.	163 faster	150	60

Figure 14. Qualitative trial runs when testing our motor speeds

3.2 Sensing Subsystem

To test our sensors, we conducted various trials to verify their capability to accurately measure surroundings.

3.2.1 toF sensor

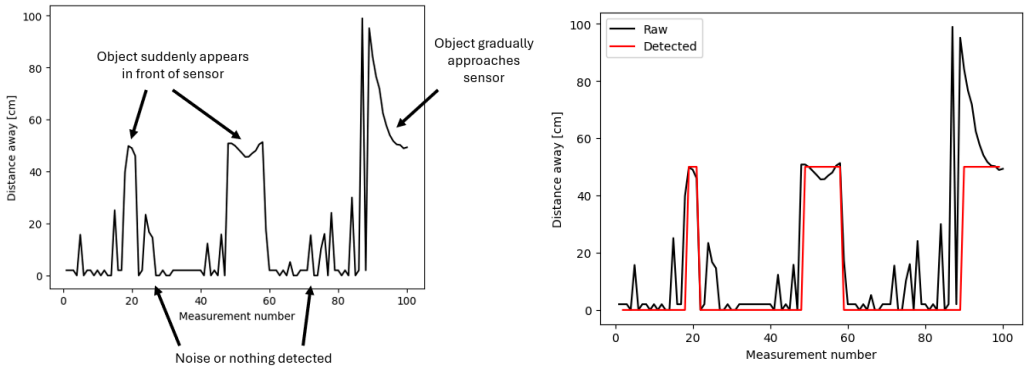

Figure 15. Raw ToF Data.

After a certain distance away from the ToF sensor, the ToF data could be regarded as valid. This meant that our data had to be filtered using a minimum threshold where data could be trusted. Additionally, in order for an object to be considered detected, it should be measured above a threshold consistently for at least three consecutive measurements. An example of how that would look is shown in Figure X, where the red line demonstrates what our robot should consider a detected object. Our robot was capable of polling up to 30 ToF distances per second. This meant that our robot would detect an object and stop 100 ms after it reached a dangerous limit.

Table 1 ToF data

	Actual Distance (cm)	Average Measured Distance (cm)	Variance (cm^2)
Measurement 1	(Open air)	80.777	10.03
Measurement 2	11	10.8	0.132

3.2.2 Pressure Sensor

Figure Z shows the pressure readings from the BMP 585 being correlated to height levels. This data is used for the robot to communicate to the controller PC when the floor level changes and to start sending the next floor commands to it.

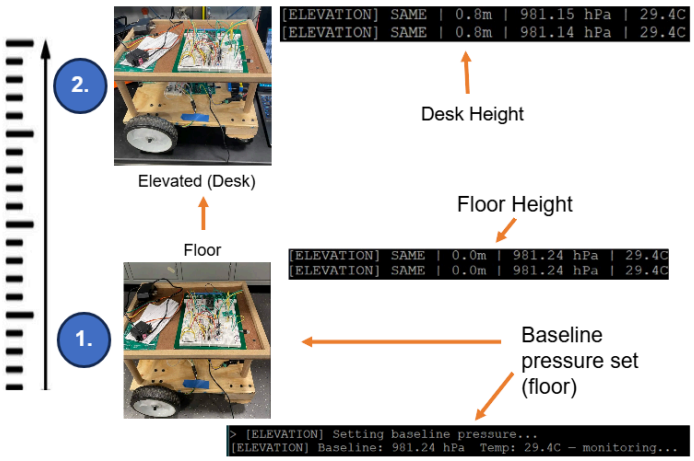

Figure 16. BMP height data

3.3 Software Verification

3.3.1 Wireless Communication (PuTTY/Telnet/Terminal)

To verify the reliability of our command transmission via the telnet operated WiFi-connection, we implemented terminal based commands that could be operated independently of the GUI program. Commands for movement of a certain degree/speed as well as initiation of proximity and height sensors measured responsiveness and allowed immediate testing of correct operation parameters. While in range, commands were transmitted and executed correctly in 100% of trials with no observed packet loss and <2 seconds of delay.

3.3.2 GUI

We tested our GUI for correctness and reliability across all of the customizable parameters, which included tile size, floor height thresholds, speed and orientation. We verified that for 100% of 50 trials, the correct speeds, distance, and height thresholds were set by observing the terminal output text. For 10 trials, we changed the distance and height parameters, ran a command to have the robot run forwards 360 degrees, then lifted the robot to the threshold height and verified that the robot ran the correct distance and motion activated again once passing the new height set for 100% of 10 trials.

3.3.3 Path Planning (A*) Verification

We verified the correctness of our A* path-planning algorithm by conducting over 10 tests with different controlled layout scenarios, including several edge cases and controls for no valid path or just a straight line with no impeding obstacles. Some layout and path examples are shown below in Figure 17. When compared to expected optimal paths, the A* algorithm evaluated maps and generated paths which avoided all obstacles with **100%** accuracy.

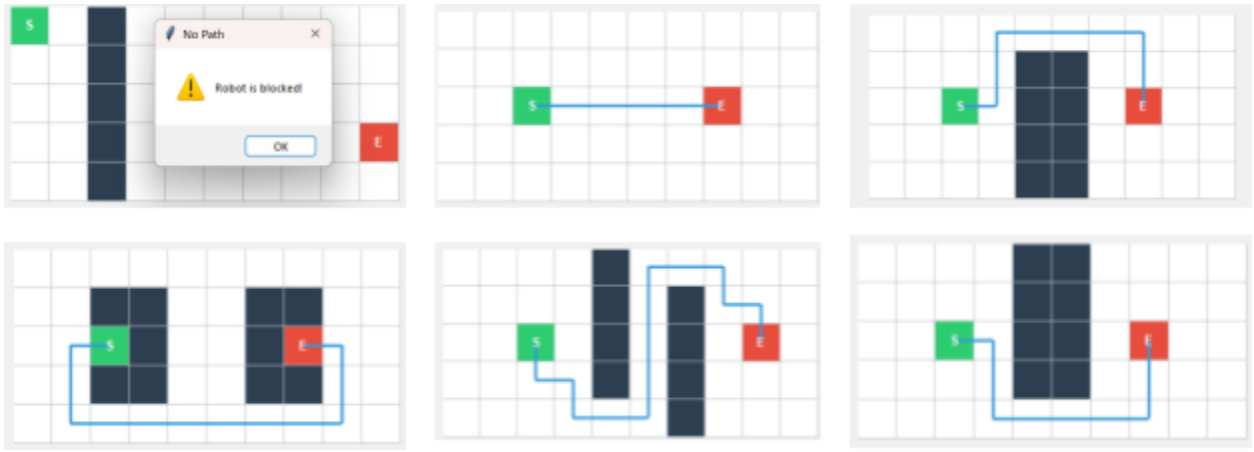

Figure 17: Select examples depicting various correct A calculated paths from our GUI.*

4. Costs

4.1 Parts

Table 2 Parts Costs

Description	Manufacturer	Part #	Cost Per Unit	Qty.
12V Hall Encoder DC Geared Motor	Bemonoc	B07GNDG2NC	$20.99	2
ESP32-S3 VROOM Microcontroller	Espressif Systems	ESP32-S3-WROOM-1-N16R2	$5.92	4
ESP32-S3 Dev Kit	Espressif Systems	ESP32-S3-DEVKITC-1-N8R8	$15.00	1
Motor Driver	Texas Instruments	LMD18245	34.33	2 (no longer manufactured)
3.3 V Regulator	Texas Instruments	LM3940IT/LM350T	$1.71	1
Pressure Sensor Devkit	Bosch Sensortec	BMP585	$14.95	1
Pressure Sensor	Bosch Sensortec	BMP585	$4.32	1
ToF Sensor Devkit	Silicon Labs	VL53L3CXV0DH/1	$9.95	1
ToF Sensor	Silicon Labs	VL53L3CXV0DH/1	$3.93	1
IMU	STMicroelectronics	LSM6DSOTR	$3.51	1
Battery	CITYORK	XZ02	$21.99	1
Simple Circuit Components	-	Resistors Capacitors BJT	$10	1
			Total Cost:	**$219.68**

4.2 Labor

For labor, each lab member will be paid at a rate of **$40/hr**, so after factoring in the multiplier and total hours each member is expected to put in to complete the project, we arrive at a total of $40/hr * 2.5 * 100 hours = **$10000 per member**. The total labor cost of our 3 member team is **$30,000**.

We are also commissioning the ECE Machine Shop for labor in building the physical robotic structure and foundation. The estimated quoted time of labor from the machine shop is 3 working days of labor, and using a quoted $56.12/hr salary we get **$1262.70** for the estimated total machine shop labor cost.

Summed together with Table 2 we get a total project cost of **$31,482.38**.

5. Conclusion

5.1 Accomplishments

Our final design accomplishes a successful demonstration and implementation of a low-cost, flexible and autonomous indoor delivery robot capable of operating on several floors. While our final product has unpolished movement calibration, we successfully integrated almost all of our proposed subsystems, and improved on our original software proposal with more features. The overall design and verification demonstrates valid proof that building a cheaper commercial indoor delivery system with smaller footprint and in-depth customizability is useful and definitely achievable.

From our subsystems, we successfully integrated our Time-of-Flight sensor with a separate PCB to detect impending obstacles with an acceptable accuracy range, also combining its operation to stop motion from frequent polling. We also integrated our pressure sensor to convert raw data to height difference readings accurately, using it to run new command sequences with height thresholds to demonstrate the possibility of autonomous elevator operation. We also connected and controlled our robot through a WiFi-based connection, realistic to operating in modern buildings. Our software incorporates several features that make operating the robot intuitive and also highly flexible. The path calculation works with 100% accuracy, and our live monitor is useful for safe operation. The main aspect that requires future improvements are related to the mechanical design and component choices that caused our robot to be unable to run straight reliably.

5.2 Uncertainties

Some of our uncertainties after achieving the progress we made with our robot revolve around the physical limitations of our design. Throughout the implementation and debugging process, we struggled the most with movement operating directly as intended without drifting or straying at an angle, but we discovered through our tests that the root of this problem originated from the balance issues with our caster wheels not touching the ground at the same height. Some uncertainties remain in the limitations with our scalability in future deployments as well. We are unsure of the performance in unideal environments with spotty wireless connectivity or with extreme complexity in layouts that are difficult to integrate precisely with a tiled map.

5.3 Ethical considerations

As responsible engineers, we recognize that ethical considerations associated with developing autonomous robotic systems that interact with people in public spaces are important to address and keep in mind when making design decisions for our project. Safety is a primary concern that is addressed through features such as obstacle detection, where we implemented emergency stop mechanisms and fail-safe behaviors to minimize risk of injury or property damage, adhering to the second code in the IEEE Code of Ethics. This also coincides with the University of Illinois policies with university facility use, as we plan to demo our completed and functioning project in the ECE building on campus and utilize various university resources in order to achieve this.

Privacy considerations are also an important consideration in ethics, also highlighted in the IEEE Code of Ethics. Any sensing data used for navigation will avoid unnecessary collection or storage of personally

identifiable data. If any vision or environment related sensing is employed, it will be limited to navigation purposes only and not used for surveillance of any sort.

Potential misuse must also be considered to avoid any possible ethical breaches by users. Autonomous delivery robots could be repurposed in ways that create safety hazards or privacy risks. To mitigate this, the system will include clearly defined documentation clarifying intended use for service needs, incorporate speed limits, and limited authentication for who has permission to access controls. We will also adhere to standard lab safety documents [15] and training when dealing with potentially volatile electronic components such as a battery.

Societally, autonomous service robots may also have potential economic impacts, including increased efficiency in hospitality settings, but also possible workforce displacement, which can be negatively perceived in terms of ethics. Our project focuses on augmenting and aiding human service roles rather than replacing them completely, especially since many facets of proper operation still require human oversight or control, demonstrating how technology can assist people with repetitive logistical tasks.

5.4 Future work

Some feedback we received involved redesigning the robot to fix the balance issues affecting the reliability of the motion subsystem and also more definitive qualitative statistics for our robot's operating standards. In the future, we can implement several improvements beyond the scope of our current project. Some upgrades we have in mind include advanced localization techniques and enhanced human-to-robot interaction, such as voice interfaces or mobile app integration. There is always room to improve or expand on autonomous mapping algorithms as well as efficient rerouting calculations. As this product will support logistical tasks involving humans, interactions can be later improved by developing more intuitive user interfaces for control or software/mobile app connections that allow remote access to our robot. We would also love to implement additional sensor functions and machine learning methods, which we believe could improve navigation reliability in crowded or unfamiliar environments. On top of those features, some general performance enhancements include further expanding battery efficiency, payload/weight capacity, and long-term reliability testing, which would all be necessary in the case of any plans for potential commercial deployment. Ultimately, further developments and advancements could enable broader applications in more environments, including hotels, hospitals, office buildings, and campus facilities.

References

[1] *I. Matijevics, "Infrared Sensors Microcontroller Interface System for Mobile Robots," 2007 5th International Symposium on Intelligent Systems and Informatics, Subotica, Serbia & Montenegro, 2007, pp. 177-181, doi: 10.1109/SISY.2007.4342647..*

[2] NI Multisim Live, "12V to 5V DC Converter," Accessed: Feb. 13, 2026. Available: www.multisim.com/content/ExAz7JQzFgNMpcjwbcCgGo/12v-to-5v-dc-converter/.

[3] Texas Instruments, LM3940IMPX-3.3 Low Dropout Voltage Regulator Datasheet, available: https://www.ti.com/lit/ds/symlink/lm3940.pdf.

[4] Espressif Systems, *ESP32-WROOM-32E Datasheet*, available: https://documentation.espressif.com/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf

[5] Texas Instruments, LMD18245 Motor Driver Datasheet, available: https://www.ti.com/lit/gpn/lmd18245.

[6] STMicroelectronics, *LIS2DH12TR Accelerometer Datasheet*, available: https://www.st.com/content/ccc/resource/technical/document/datasheet/12/c0/5c/36/b9/58/46/f2/DM00091513.pdf/files/DM00091513.pdf/jcr:content/translations/en.DM00091513.pdf

[7] Elonics.org, "IR Proximity Sensor / Obstacle Detector Circuit Using LM358 OPAMP," Accessed: Feb. 13, 2026. Available: https://elonics.org/infrared-ir-proximity-obstacle-sensor-using-lm-358/

[8] ETF Trends, "Coming Labor Crisis: America's Robot Revolution," Available: https://www.etftrends.com/disruptive-technology-content-hub/coming-labor-crisis-americas-robot-revolution/

[9] Hoover Institution, "Robotics: Build, Train, Electronic Workforce," Available: https://www.hoover.org/research/robotics-build-train-electronic-workforce

[10] International Federation of Robotics, "Service Robots See Global Growth Boom," Available: https://ifr.org/ifr-press-releases/news/service-robots-see-global-growth-boom

[11] STMicroelectronics. LSM6DSO IMU Datasheet. STMicroelectronics, https://www.st.com/resource/en/datasheet/lsm6dso.pdf.

[12] Bosch Sensortec. BMP585 Barometric Gauge Datasheet. DigiKey, https://www.digikey.com/en/products/detail/bosch-sensortec/BMP585/19522589.

[13] Texas Instruments. TL494 PWM Controller Datasheet. DigiKey (if applicable) / Texas Instruments, https://www.ti.com/lit/ds/symlink/tl494.pdf.

[14] Silicon Labs. SI1151 ToF Detector Datasheet. DigiKey, https://www.digikey.com/en/products/detail/silicon-labs/SI1151-AB00-GM/9842913.

[15] University of Illinois Urbana-Champaign. Laboratory Safety Guide. https://drs.illinois.edu/site-documents/LaboratorySafetyGuide.pdf.

[16] “A* Search Algorithms.” GeeksforGeeks, https://www.geeksforgeeks.org/dsa/a-search-algorithm/.

Appendix A Requirement and Verification Table

Table X Power System Requirements and Verifications

Requirement	Verification	Y/N
12 Volt line supplies at least 2 Amps to Motors	Motors capable to draw enough current to move robot on ground	Y
Able to supply at least 800 mA stable 3.3 Volts to Control Subsystem - Steps down 12 to 5 Volts safely using a LDO	ESP32 constant power line and capable to turn on its wifi under battery power - Probe points for different voltage levels in circuit are correct	Y

Table X Control System Requirements and Verifications

Requirement	Verification	Y/N
When Control Subsystem given list of motion commands (move 1 foot x, 2 feet y, 1 feet x,...) it is able to measure how far it is moving and accurately follow each command with a margin of +- 6 inches	GUI can send paths to robot that it will then execute - Control subsystem is able to quickly within 500 ms give a stop and start command with speed and runtime to the motion subsystem - Control subsystem can read xy motion through motor encoder outputs accurate to +- 6 inches - GUI can run A* to find the optimal path to send to the robot	Y
When Control Subsystem given information to go to next floor it uses the elevator with human/operator assistance	Subsystem enters elevator it must poll the barometric height sensor until it reaches the level it wants	Y

Table X Motion System Requirements and Verifications

Requirement	Verification	Y/N
When Motion subsystem given command to move <x,y> coordinates at a given speed and time it individually controls the motor speeds to achieve desired motion within a speed margin of 0.5 ft/s and a angle margin of +- 5 degrees	Competent Motor Control - System is able to control both motors and have them rotate forward at the same speed using encoder data as feedback to achieve a maximum motion of 3ft/s - System can accurately do turning motions accurate to +- 5 degrees by rotating the motors at different premeasured speeds.	Y

Motion subsystem is capable to drive the motors at stall current with a minimum of 1 Amp delivery to each motor	Motors do not stall - Motor driver is able to transfer enough current at the motors 1 Amp to each - Output from motor drivers does not randomly drop from unstable voltage supply through use of bypass capacitors	Y

Table X Sensing System Requirements and Verifications

Requirement	Verification	Y/N
Collision sensors can detect objects within 1 foot where the robot might want to move	toF able to stop robot when moving by reading an object within a proximity threshold.	Y
barometric gauge used to get y within a margin of +- 5 inches	Can detect small increases in height from ground to a table	Y