ECE 445

SENIOR DESIGN LABORATORY

Design Document

Autonomous WiFi Mapping Car

Autonomous Car to Map WiFi Signal Strength in a Room

Team No. 2

JOSH POWERS
(jtp6@illinois.edu)

BEN MAYDAN
(bmaydan2@illinois.edu)

AVI WINICK
(awinick2@illinois.edu)

TA: Jason Jung
Professor: Arne Fliflet

October 13, 2025

Abstract

This document provides a detailed description of our project, **Autonomous WiFi Mapping Car**, and the designs currently being implemented and tested. In this document, we breakdown the physical and circuit design, the estimated cost and timeline, as well as the ethics and safety of our project.

Contents

1 Introduction

The following sections provide an overview of the problem we are focusing on, as well as our solution to said problem.

1.1 Problem

The proliferation of wireless local area networks is fundamental to the operation of modern homes, offices, and industrial environments, supporting everything from personal communication to a vast ecosystem of IoT devices. However, the performance of these networks is highly susceptible to the physical environment, with signal strength often varying dramatically due to architectural features, furniture, and interference. The process of identifying and mitigating areas of poor coverage, or "dead zones," is crucial for network optimization but typically relies on a manual survey [3]. This method involves an individual walking through a space with a signal-measuring device, which is a process that is not only time-consuming and laborious but also yields inconsistent, low-resolution data. This leaves the placement of routers and access points to guesswork rather than analysis [1].

This project directly addresses the inefficiencies and inaccuracies of manual WiFi mapping by developing an autonomous mobile robot capable of generating a signal strength map of an indoor environment. By integrating a LIDAR sensor with a SLAM algorithm, the system can build an accurate 2D representation of an area and then navigate it autonomously along a calculated path. As the robot traverses the space, it continuously records the WiFi Received Signal Strength Indicator with an ESP32 module and correlates it with its precise location data. The final dataset enables the creation of a detailed heat map, providing a clear visualization of the WiFi coverage. This automated approach eliminates the manual labor and inconsistency of traditional methods, offering a powerful tool for users to diagnose connectivity issues and effectively optimize wireless infrastructure [2].

1.2 Solution

Our solution is an RC car designed to systematically map the WiFi signal strength within an indoor environment and generate a visual heat map. The system is built on a custom RC car platform featuring omnidirectional wheels, which allow for movement in any direction. The car is equipped with a LIDAR sensor for spatial awareness, an ESP32 microcontroller to serve as the low-level hardware controller, and a Raspberry Pi single-board computer to act as the "brain" for SLAM and the path planning algorithm. All onboard electronics, including power distribution from a LiPo battery and signal routing, will be integrated via a custom designed PCB. The system's operation is divided into two distinct phases: a short initial manual-control phase to allow the Raspberry Pi to build a 2D map of the environment, followed by a very long methodical autonomous phase to collect WiFi signal data across the entire room [3].

The implementation hinges on a clear and robust communication hierarchy. During the initial mapping phase, a user will control the vehicle from a custom GUI on a host computer, sending command packets via Bluetooth to the ESP32 which will interface with a four-channel motor controller. Simultaneously, the ESP32 will continuously process the raw data stream from the LIDAR sensor and relay parsed distance and angle information to the Raspberry Pi over a UART connection. The Raspberry Pi will then use a well-established SLAM algorithm to construct the 2D map in real-time.

Once the user engages the autonomous mode via the GUI, the Raspberry Pi finalizes the map and utilizes a path-planning algorithm, such as calculating the most efficient trajectory to cover the entire known area. It then begins sending a sequence of simple directional commands to the ESP32. As the ESP32 executes these movement commands, its onboard WiFi module is tasked with continuously scanning for the network's Received Signal Strength Indicator simultaneously correlating it with a (x, x) coordinate given by SLAM. After the planned path is complete, the ESP32 transmits the collection of coordinate and RSSI data points back to the host computer, which then renders the final heat map visualization for the user.

1.3 Visual Aid

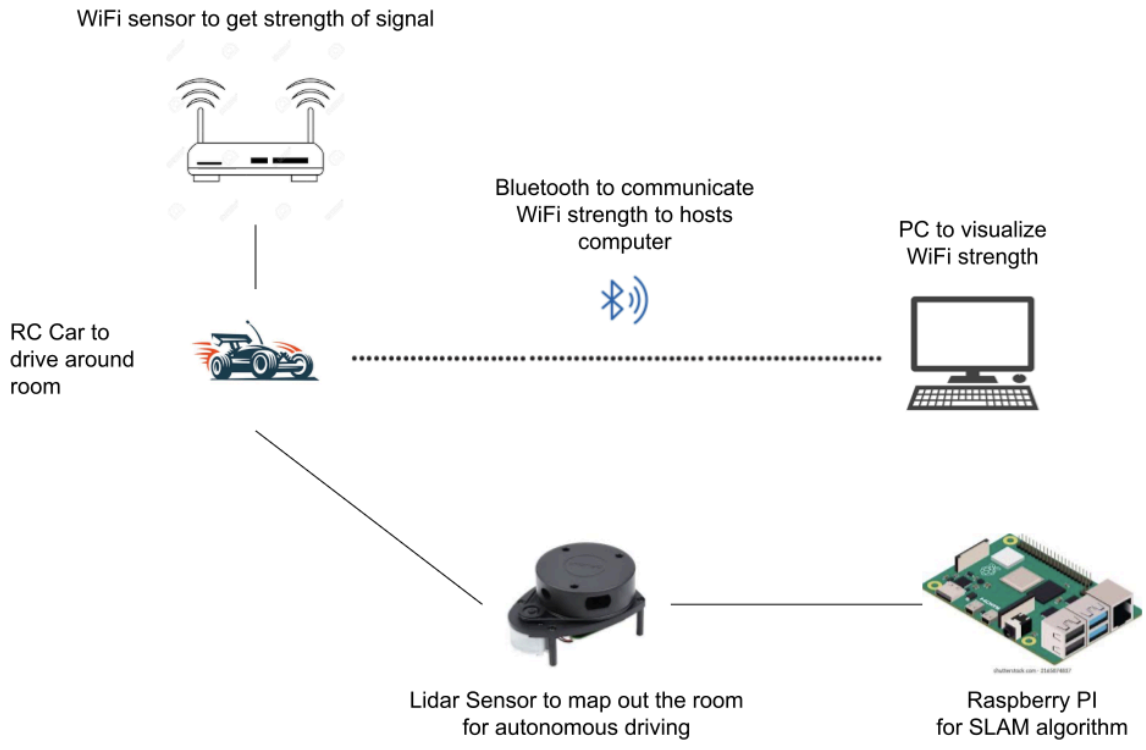


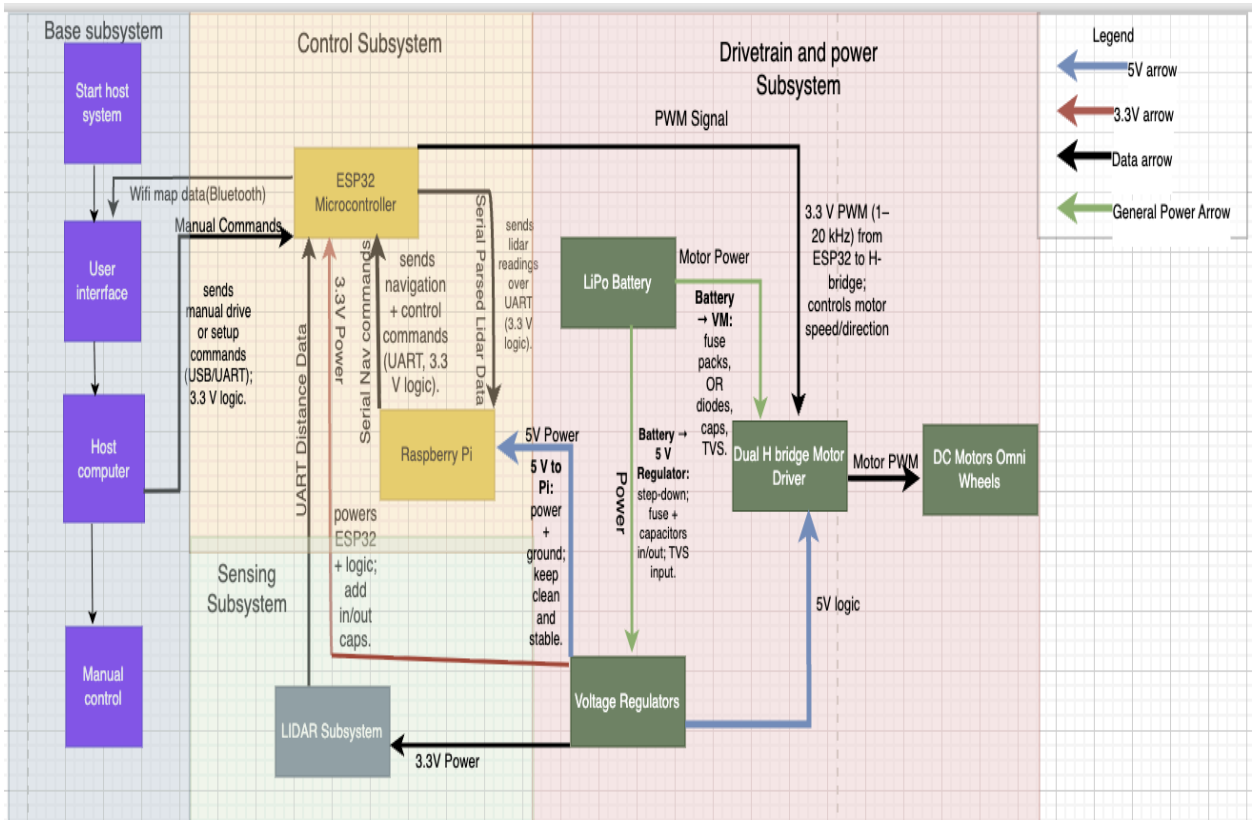
Diagram 1

1.4 High-Level Requirements

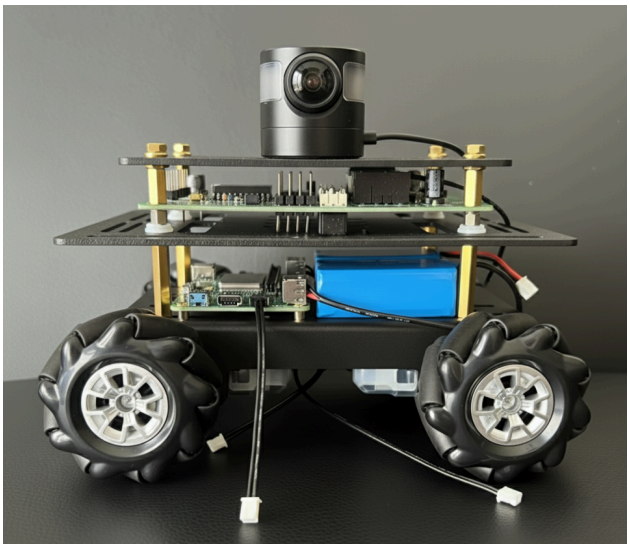
- A user must be able to manually drive the car from the host computer using Bluetooth, with the car responding to all four directional commands within 500 milliseconds latency from a distance of at most 5 meters.
- The car must drive a planned route of at least 5 meters and generate a 2D map that visually identifies the primary boundaries of the testing area as well as the obstacles around it subject to the physical constraints of the LIDAR sensor accuracy.
- The car must measure WiFi signal strength at a minimum of 5 distinct locations to generate a heat map that visualizes the varying signal intensities using at least 3 different colors, correctly showing a lower signal strength at locations farther from the WiFi router.

2 Design

2.1 Block Diagram



2.2 Physical Design



Gemini generated image of expected physical design (includes batteries, PCB, and lidar with manufactured third layer on top) [3].

2.3 Functional Overview and Requirements

2.3.1 Base Subsystem

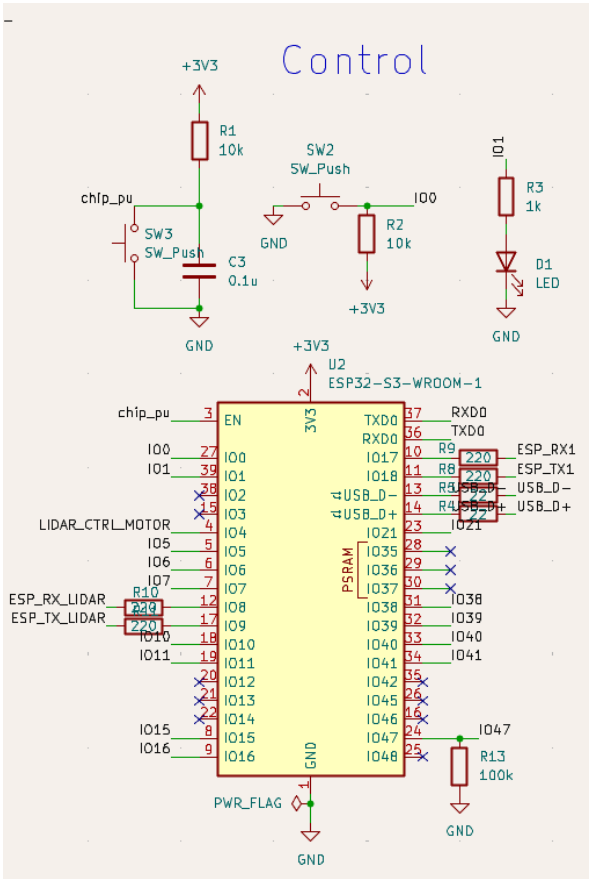
This subsystem hosts the user interface on the host computer and is responsible for starting the system, teleoperating the vehicle during the manual-mapping phase, and displaying the final Wi-Fi heat map. Functionally, it sends manual drive commands and a start signal from the GUI over Bluetooth to the ESP32 in the Control Subsystem. It also acts as the receiver for the Wi-Fi map data the ESP32 transmits back over Bluetooth once a run is complete. In short: Base ↔ Control is a Bluetooth control/telemetry link; Base does not directly touch sensors or motors, but it initiates operation and consumes the final dataset for visualization.

Requirements	Verifications
1. Must provide a graphical user interface (GUI) for manual control of the RC car.	- Launch the GUI on the host computer. - Press the on-screen controls for forward, backward, and lateral movements. - Confirm the RC car responds appropriately to each command.
2. Must be able to establish and maintain a stable Bluetooth connection with the ESP32	- From the GUI, initiate a Bluetooth pairing sequence with the car's ESP32. - Confirm that a stable connection is established and acknowledged in the GUI. - Maintain the connection without dropouts for a continuous 5-minute period while sending manual control commands.
3. Must be able to send a distinct command to initiate the autonomous mapping phase.	- After establishing a connection, click the "Begin Autonomous Mapping" button in the GUI. - Verify that the car stops responding to manual control inputs. - Confirm that the car begins to execute its autonomous path-planning algorithm.
4. Must be able to receive and parse the final (x, y, RSSI) data structure from the car.	- After the car completes its autonomous run, it will transmit the collected data. - Monitor the host computer to confirm the receipt of a data packet. - Verify that the received data structure can be correctly parsed into a list of coordinates and their corresponding signal strength values.
5. Must render the received data as a visual heat map of the mapped area	- Upon successful parsing of the (x, y, RSSI) data, observe the GUI. - Confirm that a 2D plot is generated.

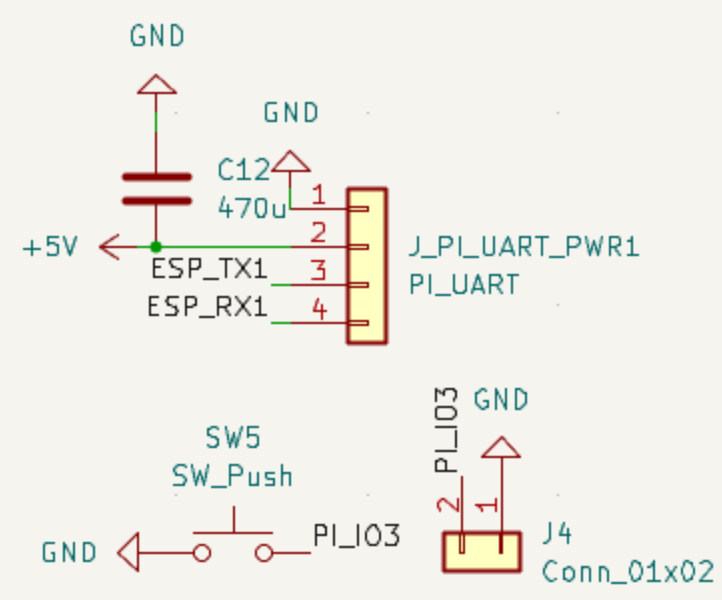
	- Verify that the plot is color-coded to represent WiFi signal strength at different locations, forming a complete heat map.

2.3.1 Control Subsystem

This subsystem contains the ESP32 microcontroller and the Raspberry Pi. The ESP32's functions are to (1) maintain the Bluetooth link with the Base Subsystem, receive manual commands/start signals, and send Wi-Fi map data; (2) generate PWM drive signals to the dual H-bridge motor driver in the Drivetrain & Power Subsystem; (3) parse incoming LIDAR distance/angle data (UART) from the Sensing Subsystem and forward parsed data to the Raspberry Pi; and (4) indicate when the final Wi-Fi map data is ready for the Base Subsystem. The Raspberry Pi runs navigation: it ingests serial LIDAR data from the ESP32, performs mapping/localization/path planning, and returns serial navigation commands to the ESP32 for execution. Power-wise, the ESP32 receives 3.3 V and the Raspberry Pi 5 V from the Drivetrain & Power Subsystem's regulators. Thus, Control is the hub: Bluetooth to Base, UART to Sensing, PWM to Drivetrain, and regulated power in from Drivetrain & Power [3].



ESP32 and IO pin connections



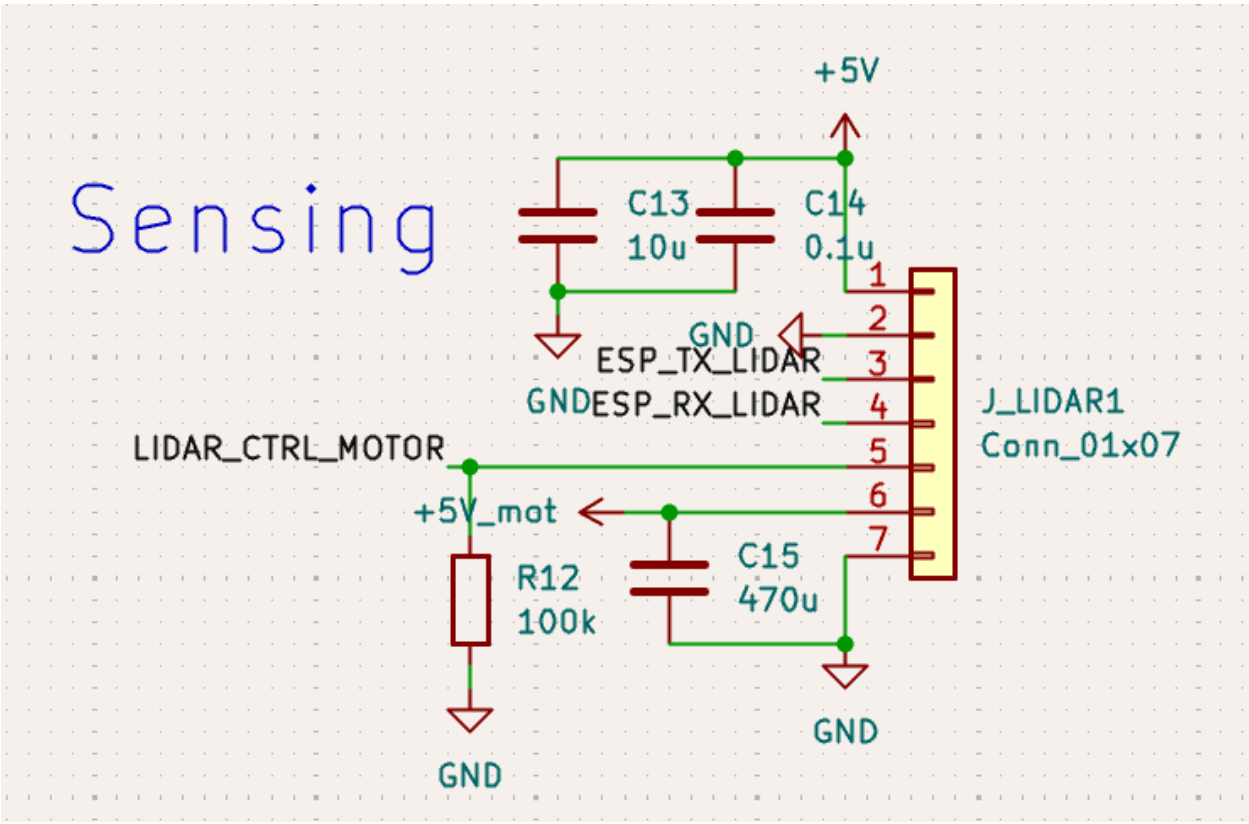
Raspberry Pi Header and Shutdown Button

Requirements	Verification
1. The ESP32 must establish a Bluetooth connection with the host computer to receive manual control commands and transmit collected WiFi map data.	- Write a test script on the host computer to send a specific command string (e.g., "FORWARD"). - Use a serial monitor connected to the ESP32 to confirm the exact string is received. - Hardcode a sample data array of (x, y, RSSI) tuples on the ESP32 and program it to transmit this data upon receiving a "SEND" command. - Verify that the host computer successfully receives the complete and unmodified data array.
2. The Raspberry Pi must process incoming LIDAR data and successfully execute a SLAM algorithm to generate an accurate 2D map of its environment.	- Place the vehicle in a room with known dimensions (e.g., 4m x 5m). - Manually drive the vehicle around the perimeter of the room. - Confirm that the generated map is topologically correct subject to the constraints of the LIDAR sensor (i.e. we cannot detect obstacles that absorb light rays).
3. The ESP32 must translate high-level directional commands into the appropriate PWM signals for the motor driver to achieve forward, backward, and lateral movements.	- Place the vehicle on a stand, allowing the wheels to spin freely. - Send a "forward" command from the host GUI and visually confirm that all four wheels spin in the correct direction for forward motion.

	- Repeat the test for "backward," "strafe left," and "strafe right" commands, verifying correct wheel behavior for each. - Probe the PWM output pins of the ESP32 with an oscilloscope to confirm that signals are present and their duty cycles change in response to different commands.
4. The subsystem must accurately sample the WiFi RSSI and associate each measurement with the vehicle's (x, y) coordinates provided by the SLAM algorithm.	- Manually position the vehicle at a known starting coordinate (e.g., (0,0)) in the mapped area. - Trigger a single data capture. Log the resulting data point (x, y, RSSI) to the serial monitor. - Verify that the logged x and y coordinates are within a reasonable tolerance (e.g., +/- 50cm) of the known position. - Verify that the logged RSSI is a valid integer (e.g., between -90 and -30 dBm).

2.3.1 Sensing Subsystem

This subsystem is the LIDAR sensor, powered at 5 V from the Drivetrain & Power Subsystem's regulators. When the RC car is started, the Control Subsystem initializes the LIDAR, after which the sensor continuously streams UART distance/angle data to the ESP32 within the Control Subsystem. The LIDAR has no direct connection to the Base Subsystem or the motor driver; its single logical link is UART → ESP32, enabling the Raspberry Pi (via the ESP32) to build the map and produce navigation commands.



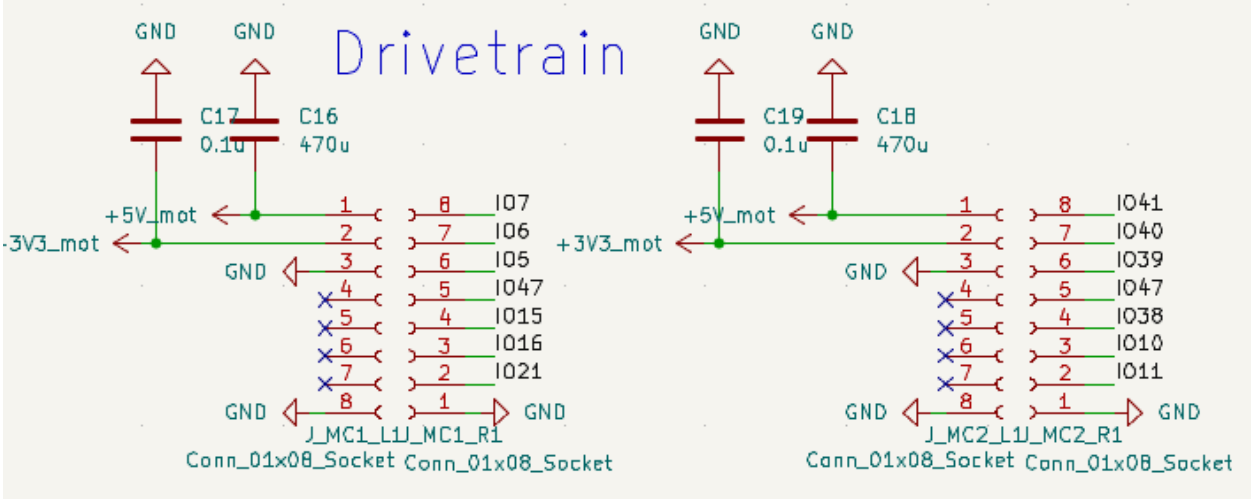
LIDAR Header

Requirements	Verification
1. The subsystem must be supplied with a stable 5.0V +/- 0.1V.	- Ensure the LIDAR is connected to the 5.0V output from the Drivetrain & Power Subsystem's voltage regulator. - Probe the LIDAR's VCC, VM, and GND pins with a multimeter. - Confirm the measured voltage is stable and within the 4.9V to 5.1V range during operation.
2. The LIDAR sensor must provide a continuous 360-degree, 2D scan of its environment.	- Power on the sensor and interface with it from a host computer. - Using visualization software, place a distinct object at known angles (e.g., 0°, 90°, 180°, 270°) relative to the sensor. - Confirm that the visualization displays the object at the correct angular positions and that the data stream covers the full 360-degree range without gaps.
3. The sensor's measurement range must be at least 8 meters with reasonable accuracy.	- In a long hallway, place a flat object (e.g., a large box) at a distance of 8 meters from the

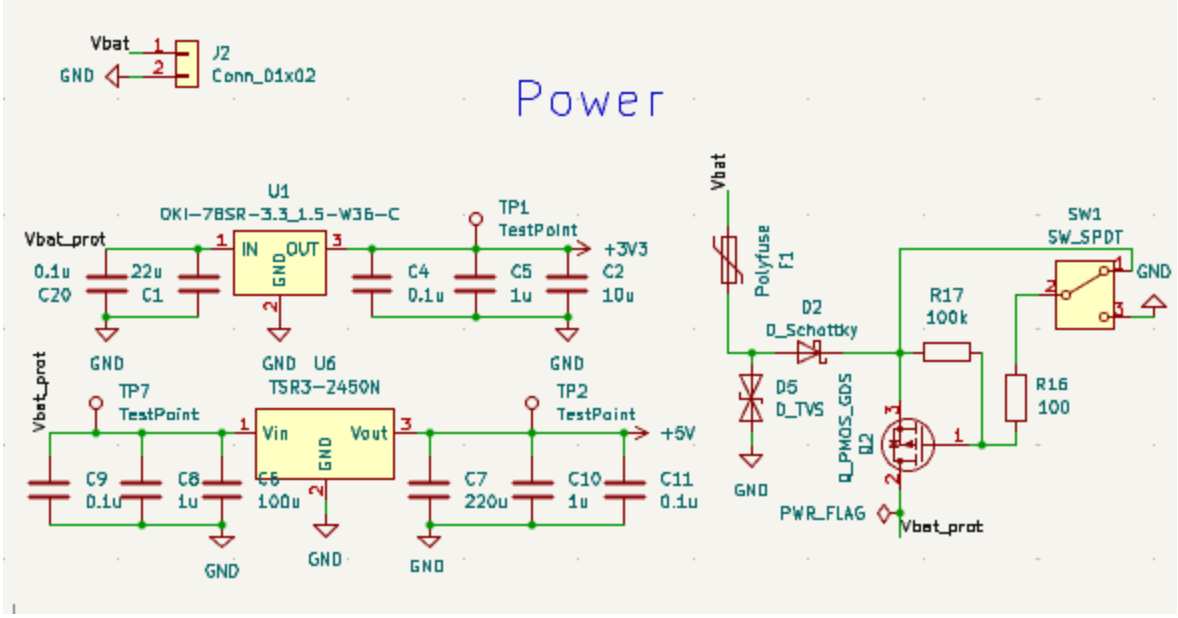
	sensor, measured with a tape measure. - Check the sensor's output data to confirm it registers the object's presence at the correct distance. - Verify that the reported distance is within the accuracy tolerance specified by the manufacturer's datasheet (e.g., ±5 cm).
4. The sensor must provide data at a rate sufficient for the SLAM algorithm to perform real-time tracking (minimum 2.5 Hz).	- Write a test script on the Raspberry Pi to read and timestamp incoming data packets from the LIDAR over the UART connection. - Calculate the frequency of complete 360-degree scans over a 10-second interval. - Confirm that the average scan rate is consistently at or above 2.5 Hz.

2.3.1 Drivetrain & Power Subsystem

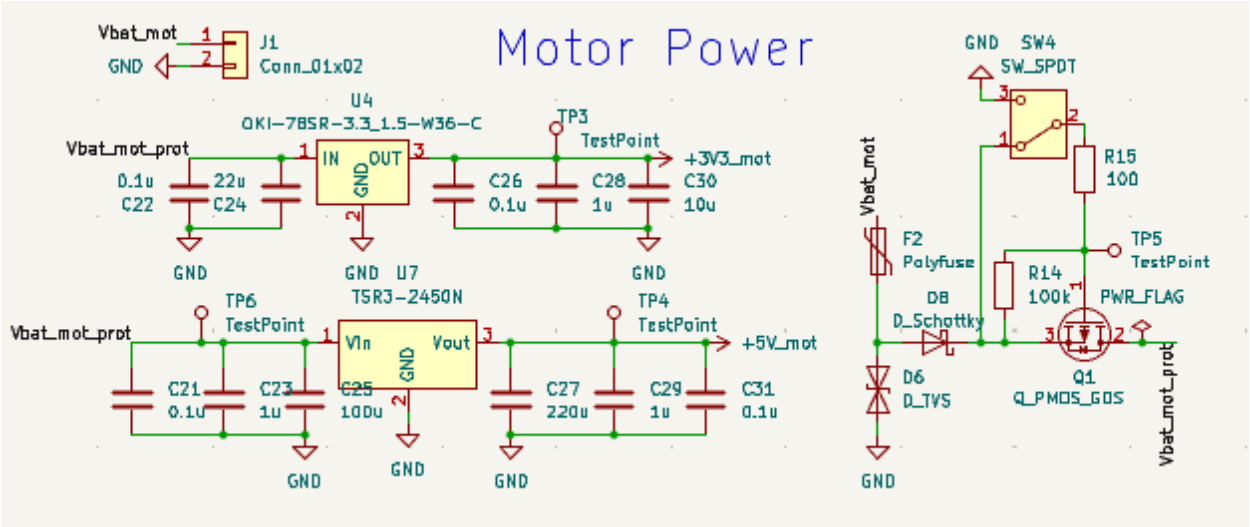
A LiPo battery feeds two paths: (1) high-current motor power directly to the dual H-bridge motor driver, and (2) the voltage regulators that generate the logic rails. The motor driver receives PWM inputs from the ESP32 (Control Subsystem) and outputs the corresponding drive currents to the DC motors on the omnidirectional wheels. The regulators supply 5 V logic to the motor driver, 5 V to the Raspberry Pi, and 3.3 V to both the ESP32 and the LIDAR sensor. The system “start” action enables these regulators so all downstream electronics power up in order. In summary, Drivetrain & Power provides the power rails for Control and Sensing, accepts PWM control from Control, and converts that into mechanical motion at the wheels.



Motor Driver Inputs



Battery Protection and Voltage Regulators



Separate Battery Protection and Voltage Regulators for Motor Power

Requirements	Verification
1. The LiPo battery must provide a nominal voltage between 7.4V and 11.1V (2S or 3S) and have sufficient capacity to power the entire system (Raspberry Pi 4, ESP32-S3, two TB6612FNG drivers, four TT motors, and LIDAR sensor) for at least 15 minutes of continuous operation.	- Fully charge the LiPo battery and connect it to the system. - Run the robot in a continuous operational mode, including driving the motors and running all processing units. - Time the duration from the start of the test until the battery's voltage drops to its safe cutoff level (e.g., ~3.3V per cell). - Confirm the operational time meets or exceeds 15 minutes.

2. The 5V voltage regulator must supply a stable 5V (±0.2V) to the Raspberry Pi 4 and the logic inputs of the two TB6612FNG motor controllers, even under motor load.	- With the system fully powered on, probe the 5V and GND test points corresponding to the regulator's output with a multimeter. - Confirm the measured voltage is in the 5.0V (±0.1V) range. - Connect an oscilloscope to the 5V rail to observe voltage stability and ripple while starting and stopping all four motors simultaneously.
3. The 3.3V voltage regulator must supply a stable 3.3V (±0.1V) to the ESP32-S3 microcontroller and the LIDAR sensor.	- With the system powered on, probe the 3.3V and GND test points corresponding to the regulator's output with a multimeter. - Confirm the measured voltage is in the 3.3V (±0.1V) range. - Connect an oscilloscope to the 3.3V rail to ensure there is no significant voltage drop or noise when other components are active.
4. The two TB6612FNG motor controllers must correctly interpret PWM signals from the ESP32-S3 and supply sufficient current (up to 1.2A continuous per channel) to drive four TT motors simultaneously.	- Write a test firmware for the ESP32-S3 to generate PWM signals of varying duty cycles (e.g., 25%, 50%, 100%). - Probe the output channels of the motor controllers with an oscilloscope to verify that the output matches the commanded PWM signal. - Connect an ammeter in series with one of the motors and command it to run at full speed. Confirm the current draw is below 1.2A. - Use an infrared thermometer to measure the temperature of the TB6612FNG ICs after 5 minutes of continuous motor operation to ensure they are not overheating.
5. The drivetrain, consisting of four TT motors and omnidirectional wheels, must be capable of executing forward, backward, lateral (strafe), and rotational movements in response to commands.	- Develop a simple test program that allows for sending specific movement commands to the robot. - Command the robot to move forward 1 meter and verify it moves in a straight line. - Command the robot to move backward 1 meter and verify it moves in a straight line. - Command the robot to strafe left for 1 meter and verify it moves laterally with minimal rotation.

2.4 Software Design

2.4.1 Bluetooth Communication

The ESP32 microcontroller's integrated Bluetooth Low Energy (BLE) capabilities are leveraged for wireless communication between the RC car and the host computer's GUI. BLE is selected for its power efficiency, which is critical for a battery-operated device, and its robust range, ensuring a stable connection within a typical indoor environment. The Arduino IDE's core ESP32 libraries, including BLEDevice.h and BLEServer.h, will provide the framework for this communication link. These libraries facilitate the setup of the ESP32 as a BLE server, allowing it to define services and characteristics that the host computer client can interact with.

To establish the communication protocol, the ESP32 is configured as a BLE server with a custom service. This service exposes two primary characteristics, each with a unique UUID, to handle the bidirectional flow of information:

1. A write characteristic is defined for the host computer to send commands to the RC car. This allows the user to control the car's operation from the GUI. The commands are sent as short strings and include manual driving instructions (e.g., "FWD", "ROT_L") and high-level state changes (e.g., "START_AUTO", "E_STOP").
2. A notify characteristic is defined for the RC car to transmit the collected Wi-Fi signal data back to the host computer. After the autonomous mapping is complete, the ESP32 compiles the list of (x, y, RSSI) data points into a single data structure, likely a JSON formatted string. It then sends this data payload to the subscribed host computer using notifications. This method is efficient for transmitting the complete dataset once it's ready for visualization.

The operational flow begins with the ESP32 initializing its BLE server and advertising its custom service. The user then initiates a connection from the GUI on the host computer. Once connected, the GUI can send commands via the write characteristic. During the initial manual phase, this allows for real-time teleoperation to map the room's boundaries. When the user initiates the autonomous phase, the ESP32 switches modes and begins its programmed path. Upon completion, it transmits the final heat map data back to the GUI via the notify characteristic, which then parses the data and renders the final visualization [3] [7].

2.4.2 Signal Control

The project's signal processing is managed by a two-tiered system, with the Raspberry Pi handling high-level computation and the ESP32 managing real-time hardware control. The Raspberry Pi processes complex LIDAR data to execute the SLAM algorithm and perform path planning, while the ESP32 interfaces directly with the motor driver, sensors, and the Bluetooth module. This division of labor ensures that time-sensitive tasks like motor control and data collection are not delayed by heavy computational loads.

Motor and Drivetrain Control: The ESP32 is responsible for translating both manual and autonomous commands into precise electrical signals for the drivetrain.

- **Manual Mode**: When a command like "STRAFE_R" is received over Bluetooth, the ESP32 generates four unique Pulse-Width Modulation (PWM) signals. These signals are sent to the dual H-bridge motor driver, which in turn powers the four omnidirectional wheel motors with the correct speed and direction to achieve the desired lateral movement.
- **Autonomous Mode**: The Raspberry Pi sends simple directional commands (e.g., "F", "B", "L") over a UART serial connection to the ESP32. The ESP32 receives these commands and generates the corresponding PWM signals in the same manner as in manual mode, executing the path calculated by the Raspberry Pi.

LIDAR and SLAM Data Flow: The system's spatial awareness is achieved through a continuous flow of data between the sensing and control subsystems.

- The LIDAR sensor continuously rotates and streams raw distance and angle data to the ESP32 via a UART connection.
- The ESP32 parses this data stream into a clean format and immediately relays it to the Raspberry Pi over a second, dedicated UART connection.
- The Raspberry Pi ingests this steady stream of spatial data, using it as the input for its SLAM algorithm to build the 2D map and continuously update the car's (x, y) coordinates in real-time.

Wi-Fi and Coordinate Synchronization: During the autonomous phase, the ESP32 collects Wi-Fi signal strength data and correlates it with location data from the Raspberry Pi. As the Raspberry Pi executes its path-planning algorithm, it continuously sends the car's current (x, y) coordinate, as determined by the SLAM algorithm, to the ESP32. Concurrently, the ESP32's Wi-Fi module samples the Received Signal Strength Indicator of the target network at a fixed interval. Each time an RSSI measurement is taken, the ESP32 pairs it with the most recently received (x, y) coordinate and stores the (x, y, RSSI) triplet in an array in its memory, eventually sending the full array over bluetooth back to the host computer to be visualized [3] [8].

2.5 Tolerance Analysis

The biggest obstacle we are facing is making sure each component gets enough current to function correctly, making sure that when all motors and chips are drawing their maximum, the system can handle it, and avoiding brownouts which is when instantaneous voltage drops below a tolerance during expected spikes like stalling or starting up. We will be using a 2-battery system, where the chips will be separated from the motors as a way to make sure that the chips can always get the current they need, regardless of motor draw.

The table below shows the components, path, and worst case current draw.

Component	Voltage	Expected current draws
ESP32	3.3V	.5 A
TB6612FNG logic (motor controllers) [9]	3.3V path 1	.05A
TB6612FNG logic (motor controllers) [9]	5V path 2	4A (This is the 8 amps total from the motor divided by 2 for two motor controllers)
Raspberry Pi[10]	5V path 1	3A
RPLidar	5V path 2	.5 A (From experimental data)
4x Motors	5V path 2	8A

Additionally from datasheet verification, we made sure each of the chips, capacitors, resistors, diodes, etc can all handle the voltage of the path it's on safely, as shown below on the table

Path	Parts	Max current allowed	Peak expected
Batteries -> V_bat	Connector->fuse-> Pmos	>10 A (Pmos)	9A
V_bat-> 5V path 1	5V buck-> Pi	3A	3A
V_bat-> 5V path 2	5V buck->motor controllers + Lidar	10A peak	8.5
5V->3V	.6A	.6A	.6
USB D+ and D-	USB-C->diodes-22 ohm resistor-> ESP32	n/a	n/a

Using worst case scenario for both current paths, our first one is

Path 1, Chips:

$I_{esp} + I_{Pi} + 2* I_{TB6612FNG} = I_{total1}$

Path 2, Motors:
$I_{LiDAR} + 4* I_{motor} = I_{total2}$

With the Esp using .5 amps, both motor controllers using a combined .1 amps, the Raspberry Pi needing 3 amps, LiDAR needing at most .5 amps, and total expected motor current being 8.5 amps, plugging those into the formula above:

Path 1 maximum current is 0.5A + 3A + 0.1A = 3.6 amps

Path 2 maximum current is 0.5A + 8A = 8.5 amps

The current path for the chips, combining the 3.3V components (0.6A) and the 5V Raspberry Pi (3A), must sustain a peak draw of 3.6A. The separate higher current path is the most current-hungry, needing to supply a peak of 8.5A for the car motors and LiDAR motor.

The next thing we need are batteries that can provide the current and voltage we need. Following the formula above, we are currently looking at Lipo batteries as the car can hold a maximum weight of 1.5 Kg including the LiDAR, PCB, batteries, and plates that came with the car.
We are leaning towards 2 sets of 2 cell LiPo with 8.4 V when fully charged, with a capacity of 5000mAh due to our weight constraints, with an ability to discharge >20 amps continuously, which gives us a lot of wiggle room.

Since the car isn't enclosed, we are expecting that heat won't be much of a problem, but will require testing later as we put the car together with the components.

3 Cost and Schedule

3.1 Cost Analysis

The following table goes through the estimated cost of both labor and parts for the Autonomous WiFi Mapping Car. We assume a yearly salary for new graduates from computer engineering to be $118,752 [4] and new graduates from electrical engineering to be $88,321 [5], as those are the average salaries posted by the college of engineering. This comes out to hourly rates of $42 and $57 per hour respectively. We estimate an average of 10 hours of work per week per member for 14 weeks. We also estimate the cost of using the machine shop to be $504 of labor for the semester [6]. Considering each team member and the machine shop, the total estimated cost of labor is $20,244. After our first PCB order, the parts listed in the table are what we have purchased for testing and creation of the car. We have totaled it out to be $290.82. When combining both labor and component costs, our total for production of the car is $20,534.82.

Labor					
Team Member	*$/hr*	*Hours Worked/week*	*Weeks Worked*	*Total Cost*	
Josh Powers	$42	10	14	$5,880	
Ben Maydan	$57	10	14	$7,980	
Avi Winick	$42	10	14	$5,880	
Machine Shop	$84	3	2	$504	
				Total Labor Costs:	$20,244
Parts					
Description	*Manufacturer*	*Part Number*	*Quantity*	*Unit Cost*	*Total Cost*
Chassis / Car Kit	LewanSoul	B093WDD9N5	1	$36.99	$36.99
Lidar	Slamtec	A1M8	1	$99	$99
Raspberry Pi 4 8GB	Raspberry Pi Foundation	Adafruit 4564	1	$82.50	$82.50
Dual Motor Drivers	Teyleten Robot	TB6612FNG	2	$8.99	$17.98
Tactile Pushbutton	Omron Electronics Inc	SW415-ND	3	$0.47	$1.41
Slide Switch	C&K	401-2016-1-ND	2	$1.14	$2.28
PMOS	Diodes Incorporated	31-DMP4010SK3-13CT-ND	2	$1.65	$3.30
Schottky Diode	Comchip Technology	641-1707-1-ND	2	$0.45	$0.90
TVS/Signal Diode	Nexperia USA Inc.	1727-3837-1-ND	2	$0.33	$0.66
TVS Diode	Littelfuse Inc.	SMBJ13ALFCT-ND	2	$0.30	$0.60
Fuse Holder	Littelfuse Inc.	F1889-ND	2	$2.50	$5.00
USB C Receptacle	GCT	2073-USB4085-GF-ACT-ND	1	$0.88	$0.88
ESP32-S3-WROOM-1	Espressif Systems	5407-ESP32-S3-WROOM-1-N8CT-ND	1	$5.49	$5.49
Murata DC-DC Module	Murata Power Solutions Inc.	811-2195-5-ND	2	$4.97	$9.94
TRACO DC-DC Module	Traco Power	1951-TSR3-2450N-ND	2	$8.45	$16.90
HDMI Cable	Amazon Basics	B00NH11KIK	1	$6.99	$6.99
				Total Part Costs:	$290.82
				Grand Total:	$20,534.82

3.2 Schedule

The following is our schedule of the work we have already completed and plan for further completion of the project in the coming weeks.

Week of	Task	Group Member(s)
9/15	Divide tasks	All
	Create Proposal	All
9/22	Proposal Review	All
	Meet with machine shop for chassis deisgn	All
	Find and order main subsytem components	All
	Code ESP32 to take commands	Ben
	GUI on host computer for commands over bluetooth	Ben
	Begin schematic design and power subsystem	Josh and Avi
9/29	PCB Review	All
	Finish schematic and pass ERC	Josh and Avi
	Set up footprints and find PCB components	Josh and Avi
	Begin LIDAR and SLAM software setup	Ben
10/6	Breadboard Demo 1	All
	Finalize PCB layout and routing	All
	First Round PCBway Order	All
	Order components for PCB	Josh and Avi
10/13	Design Document	All
	Code to send lidar data from ESP32 to Raspberry Pi	Ben
	Test and verify motor controllers to integrate into PCB	Josh and Avi
	Second Round PCBway Order, order components	All
10/20	Run SLAM on Raspberry Pi to take data and produce map	Ben
	Test motor current draw and battery selection	Josh and Avi
10/27	Breadboard Demo 2	All
	Update GUI to include start for autonomous driving	Ben
	Solder PCB and integrate into chassis	Josh and Avi
	Verify voltage regulation and signal integrity from components	Josh and Avi
11/3	Third Round PCBway Order	All
	Code the Raspberry Pi to be able to generate path	Ben
	Further component testing and PCB updates	Josh and Avi
11/10	Fourth Round PCBway Order (if necessary)	All
	Send location and instructions to ESP32 in autonomous phase	Ben
	Test autonomous navigation, SLAM accuracy, data transmission	All
11/17	Mock Demo	All
12/1	Final Demo	All
	Mock Presentation	All
12/8	Final Presentation	All
	Final paper	All

4 Ethics & Safety

4.1 Ethics

Our Commitment

Our team is committed to upholding the highest standards of ethical conduct and ensuring the safety of all individuals and property throughout the development and operation of this project. We will be guided by the principles outlined in the IEEE and ACM Codes of Ethics and will adhere to all relevant safety regulations and standards.
Ethical Considerations

This project, while academic in nature, involves the collection and processing of environmental data, which necessitates a careful ethical review.

Data Privacy and Security: The primary ethical concern is the project's data collection capability. The LIDAR sensor builds a detailed 2D map of a physical space, while the ESP32 logs wireless network information. In the wrong context, this could infringe on an individual's reasonable expectation of privacy. Intentional misuse could turn the device into a tool for surreptitious mapping of private spaces. This directly invokes the **ACM Code of Ethics 1.6 ("Respect privacy")** and the **IEEE Code of Ethics, Principle 1 (Integrity),** which includes the commitment to protect the privacy of others.
Mitigation Strategy: To prevent ethical breaches, this project will be conducted exclusively in controlled, non-sensitive university lab environments with the full consent of all persons present. The data collected (maps and RSSI values) will be used solely for academic purposes, will not be associated with any personal information, and will be securely deleted from all systems upon project completion. Data transfer from the car to the host computer will be handled responsibly to minimize the risk of interception [3].

Potential for Malicious Use: The hardware platform could be modified for malicious purposes. For example, the WiFi-enabled ESP32 could be reprogrammed from a passive signal-strength measurement tool into an active network snooping or attack tool. This possibility requires us to consider our responsibilities under **ACM Code of Ethics 1.2 ("Avoid harm")** and **IEEE Code of Ethics, Clause IX: “to avoid injuring others, their property, reputation, or employment by false or malicious action.”**
Mitigation Strategy: Our team will ensure that the developed software is strictly limited to the project's stated goals of navigation and RSSI mapping. We will not develop or distribute any software functionalities that could facilitate network intrusion or privacy violations. The project's documentation will explicitly state its intended academic purpose and highlight the ethical considerations of its hardware capabilities.

Intellectual Honesty: The project will leverage existing open-source software (e.g., SLAM on the Raspberry Pi). We will honor licenses and **properly credit upstream contributors**, aligning with **ACM Code of Ethics 1.5** (“Respect the work required to produce new ideas, inventions,

creative works, and computing artifacts") and the **IEEE Code of Ethics, Clause VII** ("…to credit properly the contributions of others").
Mitigation Strategy: Our team will diligently track the use of all third-party libraries and code. We will provide clear and accurate attribution for all open-source software used in our project reports, documentation, and presentations.

Safety Analysis

A thorough safety analysis has been conducted to identify and mitigate potential hazards associated with the project's hardware and operation.
Electrical Safety: The primary electrical hazard stems from the use of a Lithium Polymer (LiPo) battery to power the device. LiPo batteries can pose a significant fire risk if they are punctured, short-circuited, or improperly charged. The custom PCB, which handles power distribution and signal routing, must be designed to prevent short circuits.

- **Standards and Regulations:** We will ensure all electronic components are CE/UL certified where applicable. The wireless transmitters (ESP32, Raspberry Pi) are subject to FCC Part 15 regulations, and we will use pre-certified modules to ensure compliance.
- **Mitigation Strategy:** A dedicated LiPo battery charger with cell-balancing and overcharge protection will be used at all times. The battery will be physically secured within the RC car's chassis to protect it from impact. The custom PCB design will include short-circuit protection (fuses) and will be thoroughly reviewed for proper component spacing and trace routing before fabrication.

Mechanical Safety: The project is a mobile autonomous robot. Although small, it has the potential to collide with people or fragile objects, posing a minor physical hazard and a risk of property damage.

- **Mitigation Strategy:** The car's autonomous operational speed will be deliberately limited in software to a slow walking pace. All testing and operation will occur in a designated, controlled lab area, cleared of obstacles and unnecessary personnel. The system will include a clear and accessible emergency stop mechanism, likely implemented through the Bluetooth control interface on the host computer.

Laser Safety: The LIDAR sensor uses a laser to map its surroundings. Direct exposure to a high-power laser can cause severe eye damage.

- **Standards and Regulations:** Laser safety is governed by the international standard IEC 60825-1.
- **Mitigation Strategy:** We will ensure the selected LIDAR unit is a Class 1 laser product, which is eye-safe under all normal conditions of use. We will verify this classification on the component's official datasheet before integration and operation.

Campus Policy Compliance: As this is a university project, we are bound by the safety protocols of the University of Illinois at Urbana-Champaign and the Electrical and Computer Engineering department.

- **Mitigation Strategy:** All project work will be conducted in authorized lab spaces and in full compliance with university and departmental safety guidelines. We will consult with our faculty advisor and lab technicians to ensure all procedures, particularly those related to battery charging and autonomous vehicle testing, are approved and follow established

protocols.

4.2 Risk Analysis

While the autonomous Wi-Fi mapping car is a low-voltage system, a thorough risk analysis is essential to ensure safe development and operation.

Electrical Hazards
The primary electrical risk comes from the Lithium Polymer battery used to power the car. LiPo batteries can present a significant fire hazard if they are punctured, short-circuited, or charged improperly. This risk is mitigated by using a dedicated LiPo charger with overcharge protection and physically securing the battery within the car's chassis to prevent impact damage. Additionally, the custom-designed PCB will incorporate short-circuit protection, such as fuses, to prevent component damage and potential fire.

Mechanical Hazards
As a mobile autonomous robot, the car presents a potential mechanical hazard. It could collide with people or fragile objects, leading to minor injury or property damage. To manage this risk, the car's autonomous speed will be limited in the software to a slow walking pace. All testing will be conducted in controlled lab environments cleared of obstacles, and a remote emergency stop will be implemented through the host computer's GUI to halt the vehicle instantly if needed.

Laser Safety
The LIDAR sensor, which is critical for mapping, uses a laser to measure distances. Direct exposure to certain classes of lasers can cause serious eye damage. This hazard will be mitigated by ensuring the selected LIDAR unit is a Class 1 laser product. Class 1 lasers are considered eye-safe under all normal operating conditions, eliminating the risk of injury.

References

The authors acknowledge that Google's Gemini and OpenAI's ChatGPT were used for inspiration and to help flesh out initial ideas for this project.

[1] Haptic Networks. (n.d.). Common Mistakes In WiFi Network Design (And How To Avoid Them). Haptic Networks. Retrieved September 13, 2025, from https://haptic-networks.com/wifi/common-mistakes-in-wifi-network-design-and-how-to-avoid-them/

[2] Cisco Systems, “Understand Site Survey Guidelines for WLAN Deployment,” Cisco Support Documentation, updated Nov. 14, 2023. https://www.cisco.com/c/en/us/support/docs/wireless/5500-series-wireless-controllers/116057-site-survey-guidelines-wlan-00.html

[3] Google, Gemini. [Online]. Available: https://gemini.google.com. Accessed: Sept. 19, 2025.

[4] The Grainger College of Engineering. (n.d.). *Computer Engineering*. University of Illinois at Urbana-Champaign. Retrieved October 13, 2025.

[5] The Grainger College of Engineering. (n.d.). *Electrical Engineering*. University of Illinois at Urbana-Champaign. Retrieved October 13, 2025.

[6] University of Illinois at Urbana-Champaign. (n.d.). *Service Rates*. Retrieved October 13, 2025.

[7] Santos, Rui, and Sara Santos. “ESP32 Bluetooth Low Energy (BLE) on Arduino IDE.” Random Nerd Tutorials, 11 Aug. 2024, randomnerdtutorials.com/esp32-bluetooth-low-energy-ble-arduino-ide/. Accessed 12 Oct. 2025.

[8] Santos, Rui, and Sara Santos. “ESP32 UART Communication (Serial): Set Pins, Interfaces, Send and Receive Data (Arduino IDE).” Random Nerd Tutorials, 5 Aug. 2024, randomnerdtutorials.com/esp32-uart-communication-serial-arduino/. Accessed 12 Oct. 2025.

[9] Toshiba. (n.d.). TB6612FNG Dual Motor Driver Carrier Datasheet. SparkFun. Retrieved October 12, 2025, from https://cdn.sparkfun.com/datasheets/Robotics/TB6612FNG.pdf

[10] Pi 4 maximum power consumption. (n.d.). Raspberry Pi Stack Exchange. Retrieved October 12, 2025, from https://raspberrypi.stackexchange.com/questions/114239/pi-4-maximum-power-consumption