WIRELESS REMOTE MOTOR CONTROLLER

By

Aaron Chen
Kyungha Kim
Lee Boon Sheng

Final Report for ECE 445, Senior Design, Fall 2023
TA: Jason Zhang

04 December 2023
Project No. 20

Abstract

Our project introduces a Wireless Remote Motor Controller designed for adaptable and user-friendly motor control, with a focus on achieving an adjustable speed range from 0 to 100%. The key innovation is its wireless functionality, eliminating the need for wired connections, enhancing convenience, and mitigating tripping hazards. The user-friendly interface provides basic functions like start, stop, accelerate, and decelerate. The dual motor control design expands versatility for efficient robotic platforms or wireless carts. Incorporating closed-loop speed control ensures consistent performance, and current limiting control prevents overloading, ensuring long-term durability and user safety. Throughout the creation of this motor controller, we learned and created some of our own techniques to overcome obstacles face. Some topics that will be covered include different techniques in H-bridges circuit designs, dealing with high voltages, PCB design and the difficulties that come with the ESP-32.

Contents

1. Introduction

In today's rapidly advancing technological landscape, motors play a pivotal role in the functioning of countless devices and systems. Motors are the driving force behind robots that automate tasks in factories, drones that survey remote areas, and small wireless carts that navigate through crowded environments. The science or engineering problem addressed in the report is the need for efficient and convenient motor control in various applications, including robotics, automation, and remote-controlled vehicles. The existing solutions are often complex and lack user-friendliness, making them less accessible to a broader range of users. Therefore, the challenge lies in developing a wireless remote motor controller that is simple, user-friendly, and suitable for diverse applications, ranging from industrial robotics to everyday remote-controlled toys. To meet these diverse needs for simplicity and user-friendliness, a wireless remote motor controller is required. Our project aims to address this problem by proposing the development of a wireless remote motor controller that prioritizes ease of use, responsiveness, and versatility to meet the diverse needs of different applications.

In the upcoming chapter, we delve into the comprehensive design procedure and intricate details encompassing the five distinct subsystems that collectively constitute our wireless remote motor controller. These subsystems include the motor controller subsystem, power subsystem, current sensing subsystem, remote Android phone app subsystem, and remote Arduino IDE subsystem. Furthermore, a detailed examination of the costs associated with the components, particularly the Surface Mount Device (SMD) components on our PCB board, and the labor costs will be presented. As we conclude our report in the final chapter, we reflect on the accomplishments achieved and the obstacles encountered throughout the project. Additionally, we explore future avenues for enhancing the performance and adaptability of our wireless remote controller.

1.1 Visual Aid

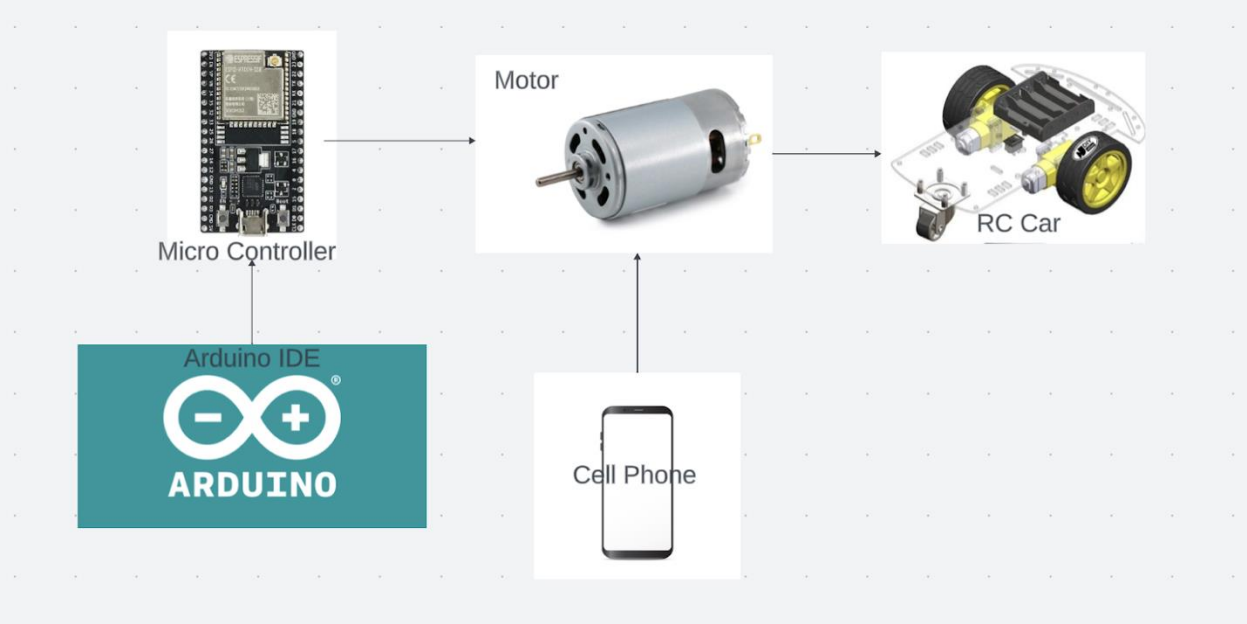

Figure 1: Visual Aid

1.2 Block Diagram

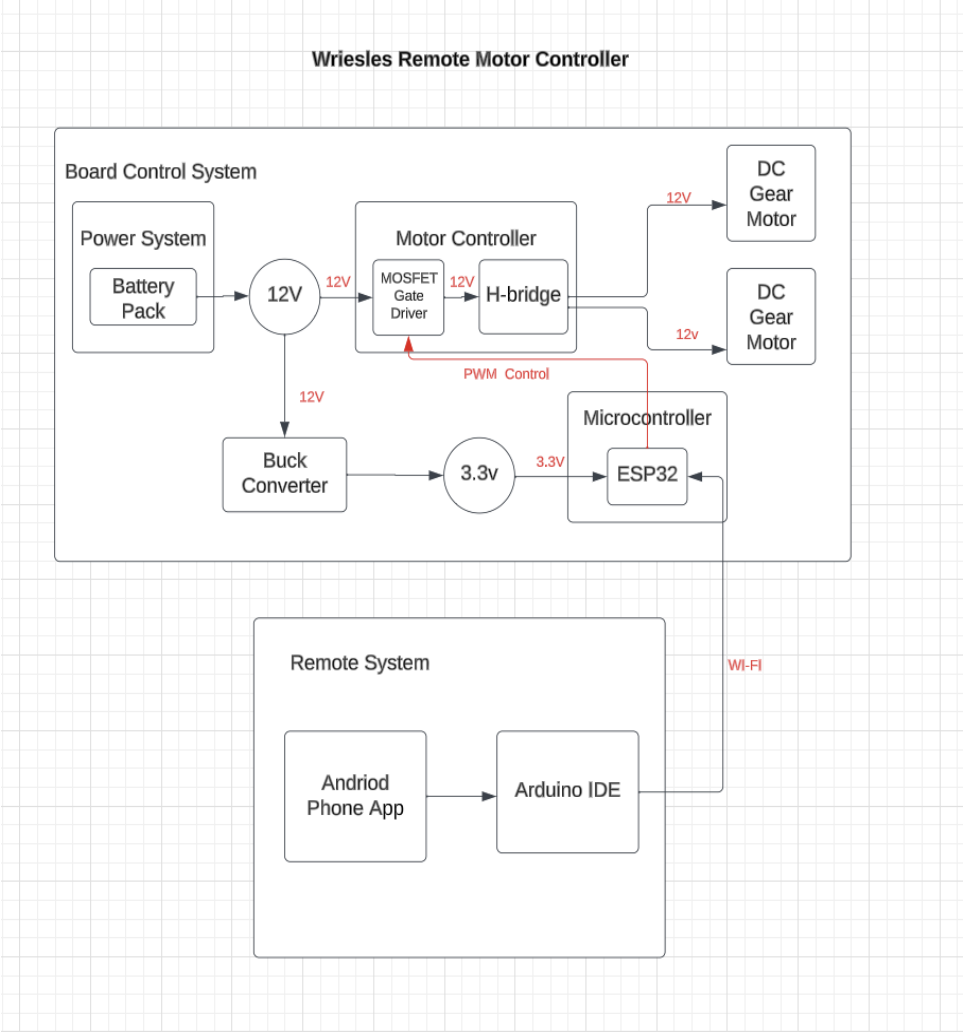

Figure 2: Block Diagram

2 Design

2.1 Design Procedure

During the initial stages of our project brainstorming, we explored various designs for the wireless motor controller. Following feedback from our Teaching Assistant (TA) and the Head TA during the design review, we made the strategic decision to construct a motor controller supporting a voltage range of 12-24V DC and capable of handling a maximum current of up to 10A. To fulfill this requirement, we opted to develop our H-bridge circuit, serving as the core element of our wireless remote motor controller project. The project is divided into five distinct subsystems: the motor controller subsystem, power subsystem, current sensing subsystem, remote Android phone app subsystem, and remote Arduino IDE subsystem.

Before proceeding to create the schematic using KiCad, we conducted simulations on LTspice to validate the functionality of our H-bridge circuit. Incorporating safety features into the H-bridge circuits to prevent short circuits was also part of our pre-schematic creation steps. The subsequent procedure involved assembling all the necessary components on the KiCad schematic. This included assigning appropriate symbols and footprints for each component, followed by moving on to the PCB layout.

In the PCB layout step, the primary focus was on ensuring the absence of air wires in the circuit and correct routing of all components. Notably, power line traces were designed with wider dimensions compared to other traces. After passing the design rule check, we advanced to the manufacturing phase, utilizing PCBWay to produce our PCB board. The final assembly was accomplished using a PCB oven to solder all the components onto the board.

Following the assembly, we initiated testing with the DC power supply in the lab, initially focusing on the power subsystem. Once functionality was confirmed, we proceeded to test the gate driver's ability to accept PWM signals from the ESP32, assessing output voltages of the HO and LO terminals with an oscilloscope. Lastly, we conducted tests to ensure the accuracy of the current sensor in measuring the current flowing into the motors.

2.2 Design Details

2.2.1 Motor Controller Subsystem

The **Motor Controller Subsystem** will include the hardware and software necessary to control the motor's speed, direction, and braking. The heart of this subsystem is the H Bridge includes four MOSFETS, two gate drivers for each side of the bridge with the associated bootstrap capacitors, and an Arduino Uno used to create the PWM signals that are fed into the gate drivers. Figure 3 below shows the basic circuit of an H Bridge circuit. When Q1 and Q4 are on, the left lead of the motor will be connected to the power supply (battery pack for RC Car demo) and current will start flowing in the forward direction and the DC gear motor shaft will start spinning. Conversely, when Q2 and Q3 are turned on, the current will flow in the opposite direction and the dc geared motor shaft will start spinning backwards. The top-end of the bridge will be connected to a power supply and the bottom-end of the circuit is grounded. Figure 4 shows the schematic of the H-bridge circuit while figure 5 shows the PCB layout of the H-bridge circuit.

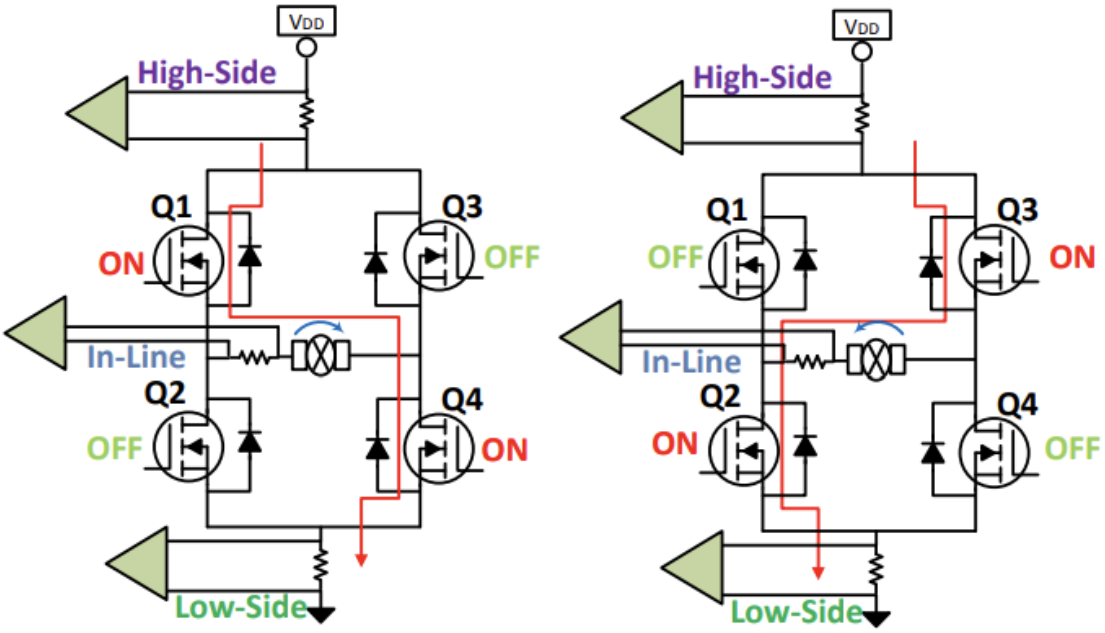

Figure 3: *H-Bridge Circuit*

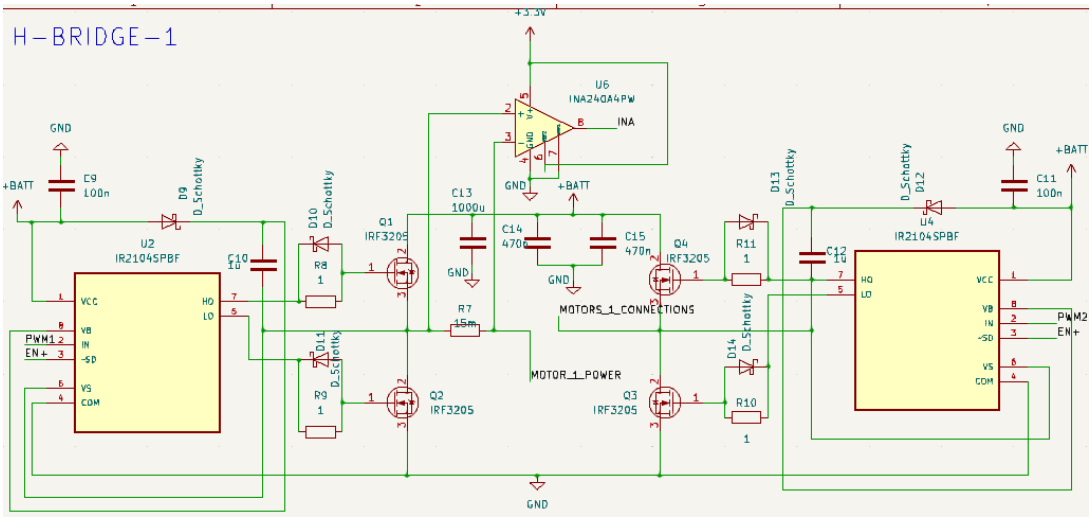

Figure 4: Schematic for H-Bridge Circuit

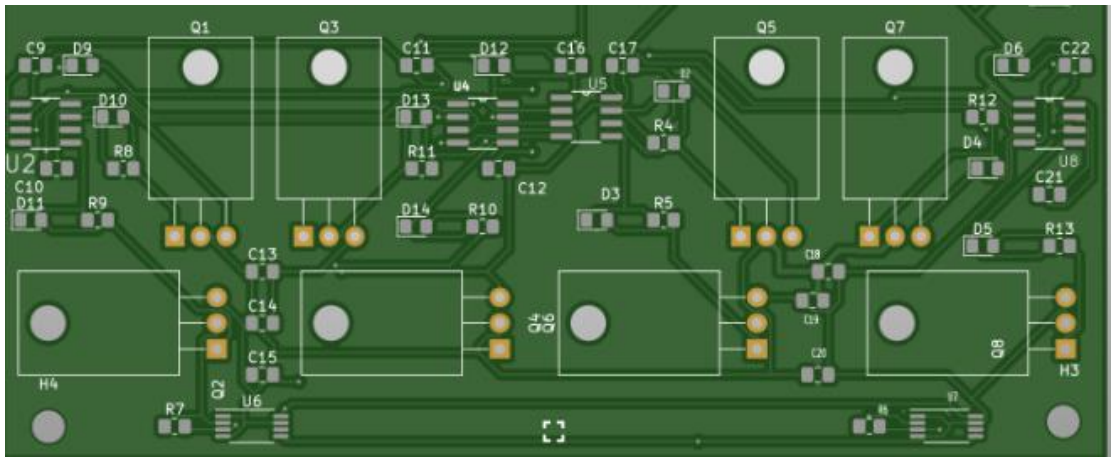

Figure 5: PCB Layout for H-Bridge Circuit

2.2.1.1 MOFSETs Selections

In our H-Bridge design, we will be using four N-Channel MOSFETS to act as voltage-controlled. For the MOSFET selection, it is important to consider the Drain-to-Source resistance of the device when the device is operating in the active region. For this reason, we will be selecting the IRF3205 N-Channel MOSFETs because the average series resistance is 8 milliohms. Therefore, the maximum power dissipation across this device would be:

$$(0.8\Omega * 10A) * 10A = 80W$$

It is safe to assume that the IRF3250 will successfully operate within the calculated current specification because the highest possible power dissipation across the device is 200 Watts according to the datasheet shown below.

Electrical Characteristics @ T_J = 25°C (unless otherwise specified)

	Parameter	Min.	Typ.	Max.	Units	Conditions
$V_{(BR)DSS}$	Drain-to-Source Breakdown Voltage	55	—	—	V	V_{GS} = 0V, I_D = 250µA
$\Delta V_{(BR)DSS}/\Delta T_J$	Breakdown Voltage Temp. Coefficient	—	0.057	—	V/°C	Reference to 25°C, I_D = 1mA
$R_{DS(on)}$	Static Drain-to-Source On-Resistance	—	—	8.0	mΩ	V_{GS} = 10V, I_D = 62A ④
$V_{GS(th)}$	Gate Threshold Voltage	2.0	—	4.0	V	V_{DS} = V_{GS}, I_D = 250µA
g_{fs}	Forward Transconductance	44	—	—	S	V_{DS} = 25V, I_D = 62A④
I_{DSS}	Drain-to-Source Leakage Current	—	—	25	µA	V_{DS} = 55V, V_{GS} = 0V
		—	—	250		V_{DS} = 44V, V_{GS} = 0V, T_J = 150°C
I_{GSS}	Gate-to-Source Forward Leakage	—	—	100	nA	V_{GS} = 20V
	Gate-to-Source Reverse Leakage	—	—	-100		V_{GS} = -20V
Q_g	Total Gate Charge	—	—	146	nC	I_D = 62A
Q_{gs}	Gate-to-Source Charge	—	—	35		V_{DS} = 44V
Q_{gd}	Gate-to-Drain ("Miller") Charge	—	—	54		V_{GS} = 10V, See Fig. 6 and 13
$t_{d(on)}$	Turn-On Delay Time	—	14	—	ns	V_{DD} = 28V
t_r	Rise Time	—	101	—		I_D = 62A
$t_{d(off)}$	Turn-Off Delay Time	—	50	—		R_G = 4.5Ω
t_f	Fall Time	—	65	—		V_{GS} = 10V, See Fig. 10 ④
L_D	Internal Drain Inductance	—	4.5	—	nH	Between lead, 6mm (0.25in.) from package and center of die contact
L_S	Internal Source Inductance	—	7.5	—		
C_{iss}	Input Capacitance	—	3247	—	pF	V_{GS} = 0V
C_{oss}	Output Capacitance	—	781	—		V_{DS} = 25V
C_{rss}	Reverse Transfer Capacitance	—	211	—		f = 1.0MHz, See Fig. 5
E_{AS}	Single Pulse Avalanche Energy②	—	1050⑥	264⑦	mJ	I_{AS} = 62A, L = 138µH

Source-Drain Ratings and Characteristics

	Parameter	Min.	Typ.	Max.	Units	Conditions
I_S	Continuous Source Current (Body Diode)	—	—	110	A	MOSFET symbol showing the integral reverse p-n junction diode.
I_{SM}	Pulsed Source Current (Body Diode)①	—	—	390		
V_{SD}	Diode Forward Voltage	—	—	1.3	V	T_J = 25°C, I_S = 62A, V_{GS} = 0V ④
t_{rr}	Reverse Recovery Time	—	69	104	ns	T_J = 25°C, I_F = 62A
Q_{rr}	Reverse Recovery Charge	—	143	215	nC	di/dt = 100A/µs ④
t_{on}	Forward Turn-On Time	Intrinsic turn-on time is negligible (turn-on is dominated by L_S+L_D)				

2.2.1.2 Gate Driver Selections

Besides, we will have to consider the Gate-to-Source Threshold voltage required by the device to switch the MOSFET on. Referencing the datasheet for the IRF3250 N-Channel MOSFET, the Gate-to-Source Threshold Voltage (represented as VGS(th)) is on 4 Volts when 250 microamps are flowing at drain. Because the HIGH signal from the ESP32 is 3.3V, we will have to use a gate driver to create a high enough charge to activate the high side MOSFETs in an H-Bridge. Let's say our source voltage is 12V from the battery and VGS(th) is 4 volts, the voltage applied to the gate of the high side driver must be:

$$(4\text{V} + 12\text{V}) = 16V$$

To activate the High Side drivers, we will have to apply 16V to the gate. If a gate driver is used in the design of an H-Bridge, then the IC itself has a built-in charge pump that can be used to amplify a charge that will in turn trigger the high side MOSFET. This internal charge pump is combined with a bootstrap capacitor that supplies the required charge needed to activate the

high side drivers. Note that this value is also lower than the maximum VGS of the RF3250 N-Channel MOSFET which is 20V which can keep the MOSFETS from operating at a safe range. The gate driver we are using is IR2104SPBF and the high side configuration can operates from 10 to 600V. This gate driver can also take in a maximum voltage supply of 25V and is compatible with the 3.3V input logic which is similar to the ESP32. The figure below shows the datasheets for the IR2104SPBF gate driver. The diagram below shows the full H-Bridge control schematic.

IR2104(S)&(PbF)

International
IOR Rectifier

Absolute Maximum Ratings

Absolute maximum ratings indicate sustained limits beyond which damage to the device may occur. All voltage parameters are absolute voltages referenced to COM. The thermal resistance and power dissipation ratings are measured under board mounted and still air conditions.

Symbol	Definition		Min.	Max.	Units
V_B	High side floating absolute voltage		-0.3	625	V
V_S	High side floating supply offset voltage		V_B - 25	V_B + 0.3	
V_{HO}	High side floating output voltage		V_S - 0.3	V_B + 0.3	
V_{CC}	Low side and logic fixed supply voltage		-0.3	25	
V_{LO}	Low side output voltage		-0.3	V_{CC} + 0.3	
V_{IN}	Logic input voltage (IN & $\overline{SD}$)		-0.3	V_{CC} + 0.3	
dV_s/dt	Allowable offset supply voltage transient		—	50	V/ns
P_D	Package power dissipation @ $T_A \leq +25°C$	(8 lead PDIP)	—	1.0	W
		(8 lead SOIC)	—	0.625	
$RthJA$	Thermal resistance, junction to ambient	(8 lead PDIP)	—	125	°C/W
		(8 lead SOIC)	—	200	
T_J	Junction temperature		—	150	°C
T_S	Storage temperature		-55	150	
T_L	Lead temperature (soldering, 10 seconds)		—	300	

2.2.2 Power Subsystem

The power system will be responsible to efficiently manage the energy source, provide stable voltage levels for various components, and incorporate safeguards to protect sensitive electronics. In our design it will comprise of a 12V DC battery input, a 3.3V solder jumper coupled with a buck converter for the ESP32-S3 module, and a battery voltage sensor with Schottky diodes to safeguard the system's stability and safety. The voltage sensor will output data to the ESP 32, which will allow us to monitor the voltage through the buck converter to ensure its stability. At the core of the power system lies the 12-24V DC battery. This will serve as the primary energy source for the entire controller. The choice of a 12-24V DC battery is deliberate, as it can be easily balanced between providing sufficient power for motor operation and being a common and readily available voltage source. This voltage level aligns with the requirements of many motors, making it an ideal choice for a wide range of applications. Figure 6 shows the schematic of the Power Subsystem circuit while figure 7 shows the PCB Layout of the Power Subsystem.

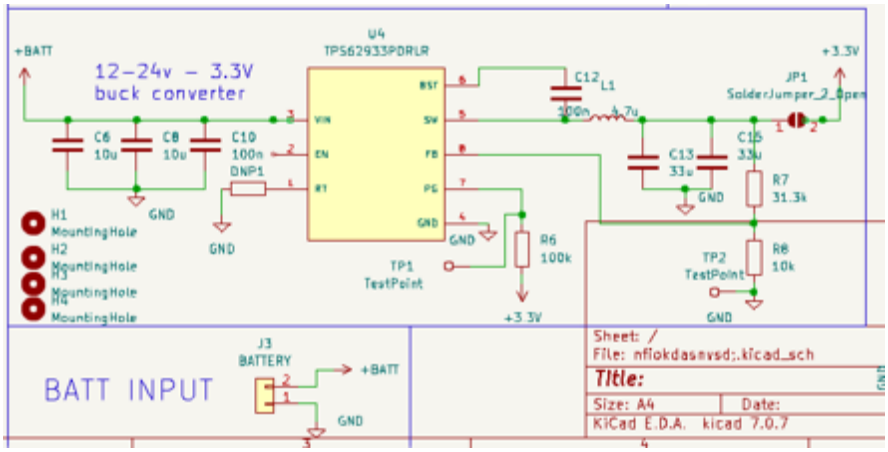

Figure 6: Schematic of the Power Subsystem

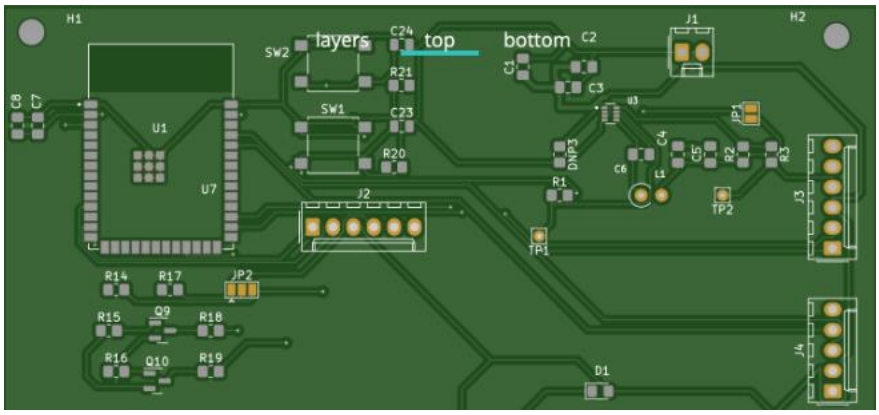

Figure 7: PCB Layout of the Power Subsystem

2.2.2.1 Powering the ESP-32

One of the key components in the wireless remote motor controller is the ESP32-S3 module, responsible for wireless communication, control logic, and user interface. However, the ESP32-S3 module typically operates at 3.3V, which poses a challenge when powered by a 12-24V source. To bridge this voltage gap, the power system incorporates a 3.3V solder jumper and a buck converter. The 3.3V solder jumper plays a crucial role in voltage regulation. It enables the selection of the appropriate voltage level for the ESP32-S3 module, allowing for flexibility in the power system design. By connecting the solder jumper, the voltage is adjusted to match the module's requirements which will be verified by using the test points we incorporated into our schematic. The buck converter, an essential part of the power system, efficiently steps down the voltage from the 12-24V source to the required 3.3V level. This conversion process ensures that the ESP32-S3 module receives a stable and precisely regulated power supply. The 3.3V can also be supplied to the EN channel of the gate drivers. Buck converters are known for their efficiency, making them an excellent choice for conserving battery power while providing a clean and consistent voltage source.

2.2.2.2 Safeguarding the System

A critical aspect of the power system is monitoring the battery's voltage to prevent overvoltage or undervoltage conditions that could damage sensitive components. To accomplish this, the system incorporates a battery voltage sensor. This sensor continuously measures the battery's input voltage, providing real-time information about the power source's health. To safeguard the ESP32-S3 module and other electronics, two Schottky diodes are employed. These diodes are strategically placed to clamp the voltage at the output of the divider circuit. Schottky Diodes are known for their unidirectional path for current and voltage. The purpose of this clamping is twofold: to protect the ESP32-S3 module's analog-to-digital converter (ADC) and to ensure that the voltage does not exceed 3.3V or drop below ground potential. The Schottky diodes are chosen for their low forward voltage drop and fast switching characteristics. This makes them effective in limiting the voltage and preventing any unwanted spikes or deviations that could adversely affect the ADC or other components.

In summary, the power system will start with a 12V DC battery input, which provides the primary power source for the system. A 3.3V solder jumper and a buck converter ensure that the ESP32-S3 module receives the correct voltage level for operation. To protect sensitive components, a battery voltage sensor and Schottky diodes are employed, ensuring the stability and safety of the power supply. This meticulous attention to the power system's design guarantees the reliability and longevity of the wireless remote motor controller, enabling it to excel in a wide range of applications while ensuring the safety of its users and components.

2.2.3 Current Sensing Subsystem

As mentioned before, the core of our motor controller circuit is an H-bridge. Because we are dealing with a maximum current rating of 10A, current sensing is used to monitor, manage, and control the load currents leading to improvement in safety, and reliability of our motor controller circuit. From the H-bridge circuit shown below, if Q1, and Q2 are both on or Q3, and Q4 are both on, it will cause a short circuit from battery to ground. In the figure, we can see that there are 3 different locations, High-Side, In-Line, and Low-Side, to measure current in an H-bridge. For our project, we use In-Line current Measurement for current sensing in an H-Bridge to direct motor current measurement and low-bandwidth amplifier. Accurate current measurement with an H-bridge is important to control motor torque. The PWM output often experiences overshoot and undershoot during transition from low to high and high to low transitions. Thus, it is important to have a current sense amplifier, which can endure these conditions while maintaining a fast response time and survive in harsh requirements of an inductive system. Figure 8 shows the schematic of the Current Sensing circuit while figure 9 shows the PCB Layout of the Current Sensing circuit.

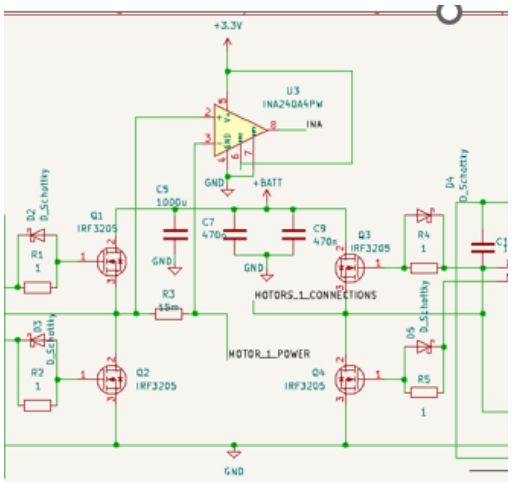

Figure 8: Schematic of the Current Sensing circuit

Figure 9: PCB Layout of the Current Sensing circuit.

2.2.3.1 Current Sensor Selections

We are using an INA240 current sense amplifier with enhanced PWM rejection, which ranges from -4V to 80V. It is designed to reject or filter out unwanted signals or noise related to PWM (Pulse Width Modulation), which helps us make accurate measurements in systems that use PWM signals. It provides a high level of suppression for common-mode transients (ΔV/Δt), which is important for real-time measurements of load current in in-line measurement positions. In other words, it can effectively filter out abrupt changes in voltage that occur in systems using PWM signals. Besides, INA240 is suitable for use in H-bridge as it can be used in various positions within the H-bridge: High-Side, In-Line, and Low-Side. The following figures show the functional block of the INA240 device.

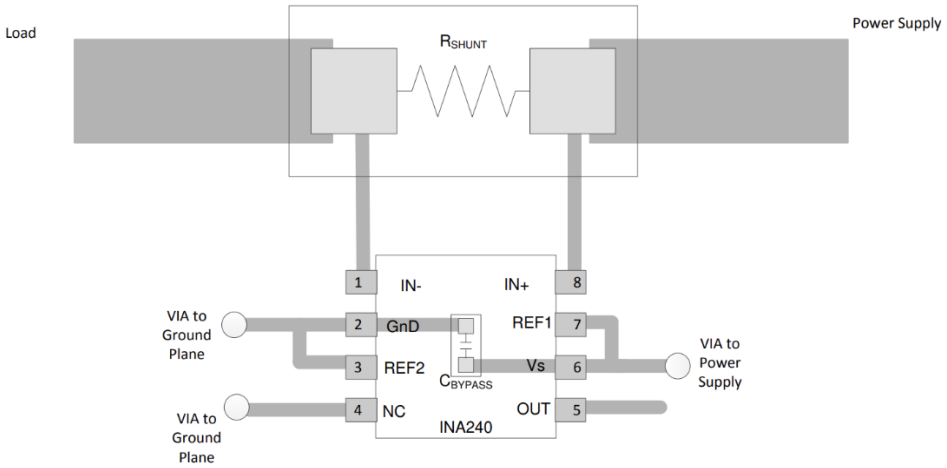

Figure 10: This shows the functional block of the INA240 device.

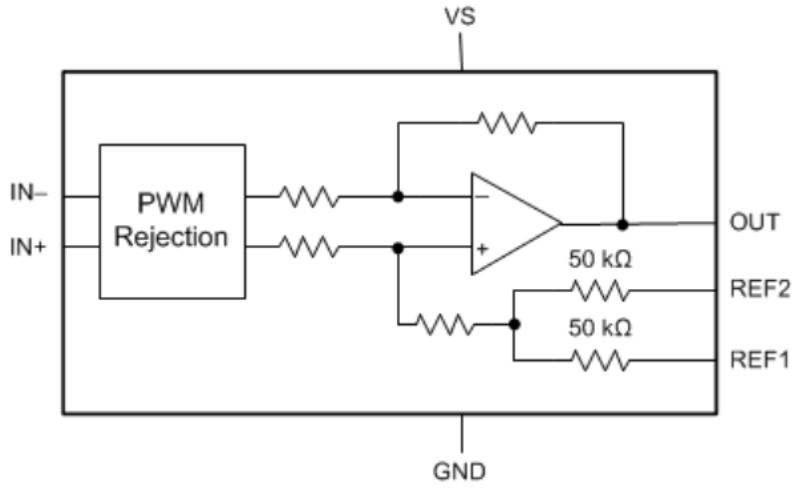

Figure 11: Functional block of the INA240 device

2.2.4 Remote Website Subsystem

The website provides wireless control for the DC motor, allowing users to send a signal that commands the behavior of the motor. These commands are delivered to the Motor Controller Subsystem and determines whether the DC geared motor to accelerate, decelerate, move forward, move backward, or stop. The H-bridge circuit on the board system drives the DC gear motor based on these commands.

Figure 12 is a screenshot of the website we implemented. Our implemented website features two control bars: the top bar manages the first motor, while the bottom bar controls the second motor. Each bar offers functionalities like speeding up, slowing down, rotating clockwise, and counterclockwise. When activated, the 'Start' button transforms into a 'Stop' button while the motor is running. The status bar displays real-time information indicating whether the motor is running or stopped. It also provides details about the rotational direction and the current speed percentage of the motor.

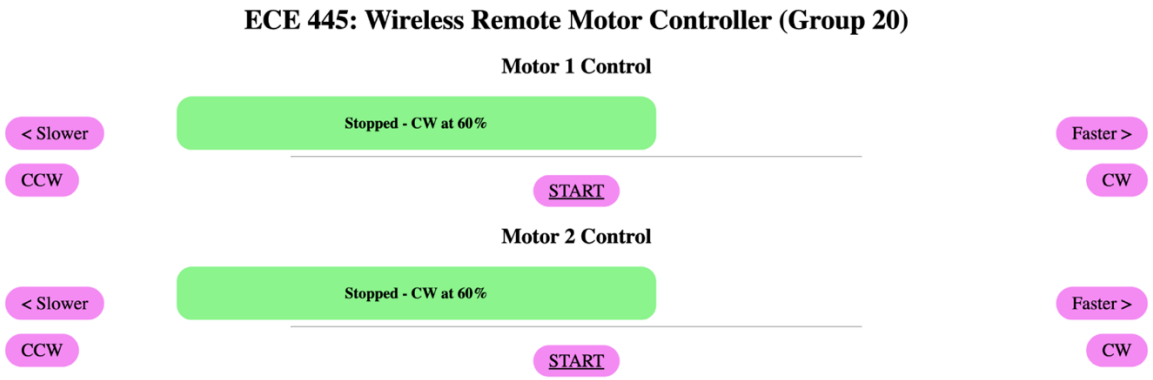

Figure 12: Screenshot of the Website

2.2.5 Remote Arduino Subsystem

Arduino IDE programs the ESP32 microcontroller. The code defines how the DC geared motors should respond to inputs received from the website. In short, the Arduino IDE enables ESP32 to control and fine-tune the motor controller based on user inputs. The pseudocode for Arduino IDE is in Appendix B.

2.2.5.1 Speed Control

We use PWM to control the speed of motors in remote control systems. By adjusting the duty cycle of the PWM signal sent to the motor controller, we can control the average power delivered to the motor. A PWM signal is a square wave with a fixed frequency and a variable duty cycle. By expressing the duty cycle in percentage, we can represent the fraction of time during one cycle. Figure 13 represents a varying duty cycle. The duty cycle represents the percentage of time during each cycle that the PWM signal is in the "on" or high state. A higher duty cycle means the motor receives power for a greater portion of each cycle, resulting in higher speed. Conversely, a lower duty cycle results in lower speed. In other words, we can decide the time it takes to go from one rising edge to the next. This allows you to vary the motor speed efficiently. This algorithm can be implemented in the Arduino IDE, and the speed could be adjusted based on user input. The percentage displayed on the website's status bar corresponds to a mapped range of duty cycles (0-255).

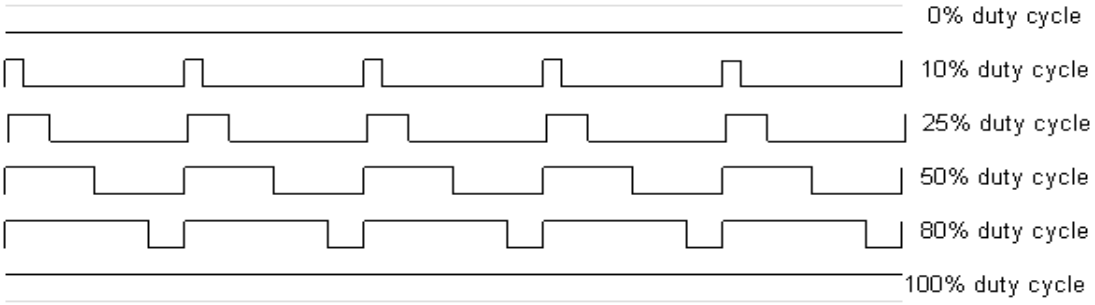

Figure 13: Graph of the Duty Cycle

2.2.5.2 Rotational Direction Control

We could also determine the direction of rotation for the connected motor by changing the state of the input pins (setting them high or low). Setting one input high and the other low (e.g., IN1 high, IN2 low) will cause the motor to rotate in one direction (clockwise). Reversing the configuration, setting the previously low input high and the previously high input low (e.g., IN2 high, IN1 low), will cause the motor to rotate in the opposite direction (counterclockwise). By manipulating the state of these input pins in the code, it configures the motor driver to create these different combinations, thus determining the direction of rotation.

3. Design Verification

Design verification is crucial for our wireless remote motor controller project because it ensures our final product meets its desired functional requirements as outlined in the project's initial stages. During the verification process, it helps identify errors in the early development process and confirms the functionality of all low-level requirements. Successful completion of these tests would affirm the overall compliance, while any failures would trigger a fallback to our conventional verification processes for troubleshooting. Appendix A includes a table providing a comprehensive overview of the requirements, steps taken during verification, and the corresponding results. Further insights into the verification process specific to each subsystem are explained in sections 3.1 to 3.5.

3.1 Motor Controller Subsystem

At the heart of this subsystem lies an H-bridge comprising 4 MOSFETs, 2 gate drivers, and one current sensor. We conducted simulations on the H-bridge using LTspice for testing, illustrated in figure 14. The graph representing the H-bridge exhibits a square wave, affirming the accuracy of the overall H-bridge design, as depicted in figure 15.

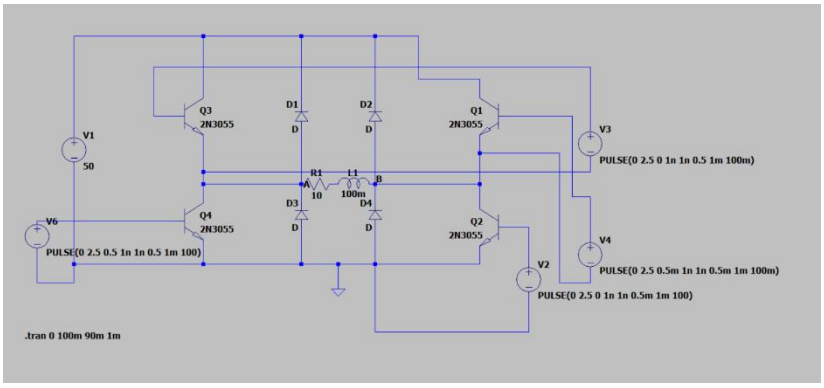

Figure 14: Schematic Used for Testing H-bridge

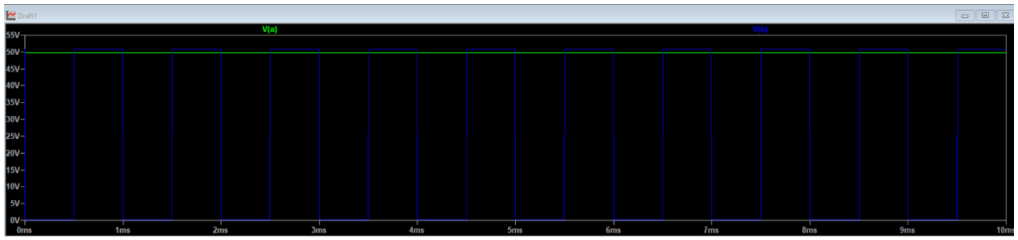

Figure 15: Graph for H-bridge

Upon the completion of assembling all H-bridge components on the PCB, we applied a 12V and 0.5A power supply to the designated terminal using a DC power supply from the laboratory. This step was taken to conduct a preliminary test for potential short circuits. Once it was confirmed that there were no short circuits, we utilized a multimeter to probe the VCC terminal on the gate driver IC, ensuring it received the designated 12V DC supply. Subsequently, we probed the HO and LO channels of the gate driver ICs to verify the presence of the 3.3V signal,

indicative of the gate driver successfully receiving the PWM signal from our ESP32 microcontroller.

3.2 Power Subsystem

In the power subsystem, our objective is to take in a 12-24V DC power supply and step it down to 3.3V. This lower voltage is then supplied to the VCC of the ESP32 microcontroller, current sensors, and the EN terminal of the gate driver ICs. To bridge this voltage gap, the power system incorporates both a 3.3V solder jumper and a buck converter. After assembling all the power subsystem components on the PCB, we applied a 12V and 0.1A power supply to the designated terminal using a DC power supply from the laboratory. This step served to conduct a preliminary test for potential short circuits and to evaluate the functionality of the buck converter. The presence of the solder jumper is crucial as it prevents potential damage to the ESP-32 in case the buck converter malfunctions by redirecting high voltage. Once it is confirmed that there are no short circuits, the solder jumper will be connected, and the desired 3.3V voltage will be verified using the test points incorporated into our PCB layout.

3.3 Current Sensing Subsystem

As previously mentioned, the core of our motor controller circuit is an H-bridge. Given the maximum current rating of 10A, the inclusion of current sensing is crucial for monitoring, managing, and controlling load currents, significantly enhancing the safety and reliability of our motor controller circuit. Before assessing the functionality of our current sensor, we verified the presence of the 3.3V signal in the VCC channels of both current sensor ICs. In our H-bridge circuit, a 15mΩ shunt resistor is positioned between the power supply and motor. Using a multimeter, we will measure the current across the shunt resistor for comparison with the current sensor reading. Once the accuracy of the current sensor is confirmed, we will connect the INA channel of the current sensor to the GPIO channel of the ESP32 for data retrieval.

3.4 Remote Website Subsystem

In ensuring smooth communication between the website and the Motor Controller Subsystem, our focus was on the ESP32's accurate interpretation of user inputs from the website. We meticulously checked and confirmed that the esp32 correctly recognized and interpreted various user commands—adjusting speed, changing directions, and initiating start and stop commands. To validate this, we conducted tests where we printed the user input from the website when buttons were pressed, ensuring the ESP32's understanding of these commands, crucial for the motor's behavior as per the programmed Arduino IDE code.

3.5 Remote Arduino Subsystem

Our attention shifted to the Remote Arduino Subsystem to ascertain the correct programming of the ESP32 microcontroller using the Arduino IDE. Prior to programming, we meticulously checked the software's connection to the ESP32, ensuring the correct pin configuration through the pinMode() functions from the Arduino IDE. Our testing methodology was centered on confirming the ESP32's accurate interpretation of user input from the website. We initially validated the ESP32's reception of user commands by cross-referencing data transmitted to it. Subsequent tests focused on observing the motor's response to these commands, ensuring their precise execution. Additionally, we confirmed the correct mapping of speed percentages

to the range of duty cycles (0-255) and paid close attention to consistent motor responses for both direction and speed control. This scrutiny ensured the system's accuracy and reliability in translating user commands into motor actions.

4. Costs

4.1 Parts

Table 4.1 Parts Costs Bought

Number of Items	Part Number and Description	Manufacturer	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
1	1x ESP32-S3-WROOM-1-N4R8	Espressif Systems	3.35	N/A	3.35
2	Motor with encoder	Bemonoc Store	14.88	N/A	29.76
1	Buck Converter (TPS563300DRLR)	Texas Instrument	0.68	N/A	0.68
2	Current sensor (INA240A4PWR)	Texas Instrument	3.23	N/A	3.23
8	MOSFET(IRF3205PBF-ND)	Infineon Technologies	1.44	N/A	11.52
1	Battery Pack	CBB store	19.99	N/A	11.99
5	PCB	PCBway	5.00	5 for 5	5.00
1	1x 470uH Inductor (RT0805BRD0731K2L)	Murata Power Solutions	0.29	N/A	0.29
6	0.1u Capacitor (11R474C)	Kyocera AVX	0.27	N/A	1.62
1	31.2k Resistor (RT0805BRD0731K2L)	Yageo	0.33	N/A	.33
6	0.47u Capacitor (GCM21BR71H474KA55L)	Murata Electronics	0.20	N/A	1.20
2	15m Ohm resistor (RL0805FR-070R015L)	Yageo	0.59	N/A	1.18
8	1 Ohm Resistor (RMCF0805JT1R00)	Stackpole Electronics	0.09	N/A	0.72
4	Gate Drivers (IR2104PBF)	Infineon Technologies	2.67	N/A	10.68
Total					**$92.46**

4.2 Labor

Overall, the total parts cost will be around $92.46, but with shipping and an additional 6.25% sales tax the total will come around $118. Over the course of the semester-long Wireless Remote Motor Controller project, our team meticulously navigated through distinct phases, each demanding specialized expertise. The initial weeks were dedicated to comprehensive research and planning, where the Project Manager led for 20 hours alongside the Electrical Engineer (EE) and Software Developer contributing 30 and 20 hours, respectively. Subsequently, the design phase unfolded, with the EE investing 40 hours in circuit design, the Mechanical Engineer allocating 15 hours for the enclosure, and the Software Developer dedicating 30 hours to interface design and wireless protocols. As the semester progressed, the team transitioned into prototype development, with the EE and Software Developer investing 50 and 40 hours, respectively, in building and testing the electronic circuit and programming the controller functionality. Testing, iteration, documentation, and final presentation phases were meticulously executed, with the entire team synergizing efforts. Throughout the semester, the Project Manager maintained a cumulative 50 hours for coordination and oversight. In total, the project demanded approximately 400 hours of collaborative effort, resulting in the development of a sophisticated Wireless Remote Motor Controller. With that said we will anticipate a compensation of $40 per hour for each team member working, resulting in $16,000 per individual. When multiplied by the number of team members, the total labor cost amounts to $48,000 which does not include overtime. The total spending for this project comes out to be around $48,092.46.

5. Conclusion

5.1 Accomplishments

The project boasts significant achievements across its key components. Firstly, the power subsystem demonstrates its capability by effectively delivering 12V to the power line and 3.3V to both the ESP32 and current sensors, ensuring smooth functionality. Secondly, meeting all three high-level requirements using the ESP32 dev board and L298N H-bridge module signifies a major milestone. Circuit works as intended as we replaced the L298N H-bridge with the H-bridge circuit we designed on the PCB board. One of our standout accomplishments is creating a website app that allows wireless control over the motor's direction and speed. It works well within a range of 10-15 meters which is one of our high-level requirements, showing that our system is strong and fully operational. Moreover, it continues to work reliably even beyond 15 meters, showcasing its extended operational capacity.

5.2 Uncertainties

Several uncertainties have arisen during the project, demanding attention for resolution. Initially, the critical need to replace the malfunctioning gate driver on our PCB board is evident. Moreover, a concern has surfaced regarding improperly sized diodes on the PCB, necessitating a redesignation of the circuit board. Due to the incorrect diode size that was ordered and did not fit the pad, we resorted to manually connecting the diodes using two wires. Moreover, challenges have surfaced regarding the ESP32's programming on the PCB board. In our KiCad schematic, we have two sheets of schematics organized hierarchically, with a root sheet and sub-sheet. The ESP32 problem is primarily stemming from labeling that wasn't set to global variables in the sub-sheet and thus a lack of an established connection on the PCB board.

5.3 Ethical considerations

In the development of the Wireless Remote Motor Controller project, we are committed to upholding the highest ethical and safety standards as outlined in the IEEE and ACM Code of Ethics. Specifically, we will prioritize safety by ensuring the device complies with ethical design and sustainable development practices. Safety is a paramount concern throughout the development and operation of the Wireless Remote Motor Controller project. It extends to various aspects, including power control, voltage regulation, soldering practices, and the proper use of equipment. Here, we emphasize the safety measures and considerations associated with these critical project components: To prevent overheating and protect the motor and other components, the power system must implement current limiting mechanisms. This ensures that the motor operates within safe limits, reducing the risk of damage or accidents. Maintaining a stable voltage supply, as achieved through the buck converter, is essential for the safety of the entire system. Fluctuations in voltage can lead to erratic motor behavior and pose risks to users and equipment. We will continuously monitor the voltage levels as a safety measure. The battery voltage sensor helps in this regard, allowing the system to take corrective actions if voltage levels fall outside safe operating limits. When working with electrical equipment, including the buck converter and voltage sensor, it is crucial to follow electrical safety practices, such as isolating power sources when making connections. Incorporating these safety measures and considerations into the

Wireless Remote Motor Controller project not only ensures the safety of the development process but also contributes to the overall safety of the product. Prioritizing safety at each stage of the project's lifecycle, from design and assembly to testing and operation, demonstrates a commitment to delivering a reliable and secure product. should have incorporated voltage input protection mechanisms to safeguard against voltage spikes, surges, or reverse polarity, reducing the risk of damage to the controller and connected motors. This requirement enhances the controller's durability and reliability.

5.4 Future work

Moving forward, several key areas require attention for future improvements. Firstly, replacing the correct gate drivers stands as a priority to ensure the smooth operation of the system. Optimizing the layout of PCB components by separating the digital and analog traces can reduce the chances of digital noise affecting sensitive analog signals. This approach can improve the efficiency and performance of the circuit because the digital and analog signals will be operating at different voltage levels. Additionally, incorporating more test points into the design will facilitate easier troubleshooting and validation in the future. It's also advisable to order surplus key components as backups, ensuring continuity in case of unforeseen failures. This is because most of our SMD components are the size of 0805 and they can easily be lost. Implementing decoupling capacitors in between the space of MOFSETs will contribute to better noise reduction and stability within the system. Placing MOFSETs at a90-degree angle can improve thermal dissipation which can improve the overall performance. Furthermore, expanding the functionality of the webpage by integrating features such as real-time RPM and current readings will enhance the user interface and provide valuable live data for monitoring purposes. These planned enhancements will significantly contribute to the system's reliability and performance in future iterations.

References

[1] Microchip Technology Inc. "AN905 - Stepper Motor Control Using the PIC16F684." Microchip Technology Inc., http://ww1.microchip.com/downloads/en/appnotes/00905b.pdf. Accessed 27 September 2023.

[2] Yavuz, Hasan. "" Ozderya, https://hasanyavuz.ozderya.net/?p=437. Accessed 27 September 2023.

[3] Modular Circuits. "H-Bridges – the Basics" Modular Circuits, https://www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridges-the-basics/. Accessed 27 September 2023.

[4] Michigan State University. "Application Note Regarding H Bridge Design and Operation" Michigan State University, https://www.egr.msu.edu/classes/ece480/capstone/fall14/group07/PDFs/Application%20Note%20Regarding%20H%20Bridge%20Design%20and%20Operation.pdf. Accessed 27 September 2023.

[5] Texas Instruments. "*Current Sensing in an H-Bridge*" Texas Instruments, https://www.ti.com/lit/an/sboa174d/sboa174d.pdf?ts=1685998152220&ref_url=https%253A%252F%252Fwww.google.com%252F. Accessed 27 September 2023.

[6] No specific author. "Micropython on ESP8266 and ESP32 - PWM LED Fading." EngineersGarage, https://www.engineersgarage.com/micropython-esp8266-esp32-pwm-led-fading/#:~:text=While%20the%20base%20clock%20APB_CLK,cycle%20resolution%20of%201%20bit.

Appendix A Requirement and Verification Table

Table A.1 System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. The **Motor Controller Subsystem** will include the hardware and software necessary to control the motor's speed, direction, and braking. a. The motor controller will be able to withstand 12 V b. The IRF3205 MOSFETs can take in ± 20V from the gate driver to be fully switched on. The IR2104SPBF gate driver has to be able to take in maximum supply voltage of 25V and 3.3V logic input.	1. When a button on the website the car will move the correct direction a. We will verify this by using a voltage source of 12V. We have placed test points in our schematic where we can monitor if there are any irregularities in the function of our system at 12V. b. We will ensure the MOSFETs work in a safe operation zone by choosing a high-speed power MOSFET driver , IR2104SPBF, that has a floating channel that can operate from 10 to 600V to drive the high-side of the N-channel MOSFETs. c. We will test this by connecting the battery input voltage(12V) to the VCC channel of the gate driver and the ESP32 input channel (any GPIO) to the EN channel of the gate driver(3.3V).	Y
2. The **Power Subsystem** will efficiently manage the energy source, provide stable voltage levels for various components, and incorporate safeguards to protect sensitive electronics. a. The power system will be able to limit the input voltage of the ESP 32 to only 3.3 Volts.	2. We will test each VCC input of each component to make sure it is turned on. a. Before connecting the source to the ESP 32 we will check the output of the buck converter at the solder jumper using the multimeter to verify the	Y

	output is a steady 3.3Volts.	
3. The **Current Sensor** will be to achieve current measurements in the H-bridge and feedback the signal to the ESP32 from the INA channel. b. The INA240 current sensor will be able to operate from a supply voltage ranging from 2.7 to 5.5V.	3. We will be using a multimeter to measure the current flowing across the shunt resistor (15m ohm) in between the power supply and motor connection to get the accurate measurement of the current to compare with the reading of the current sensor. a. To verify, we will make sure the current sensor is connected to the 3.3V source similar to ESP32 as well as the gate driver mentioned above.	N
4. The **Remote Website Subsystem** manages DC motor behavior through Motor Controller Subsystem. a. The ESP32 must interpret user input status.	4. Ensure that the Android phone app is in the default unpressed state. Record the data transmitted from the app to the ESP32. We will also Record the behavior of the DC geared motor in response to each command and confirm that it aligns with the expected actions (acceleration, deceleration, forward, backward, or stop). a. Test the wireless connection between the Android Phone App and the Motor Controller Subsystem by sending simple test commands (start or stop motors) from the app to the Motor Controller subsystem. Confirm connectivity by checking if the Motor Controller Subsystem correctly responds to the commands by executing the requested actions.	Y

5. The **Remote Arduino IDE subsystem** programs the ESP32 to dictate DC geared motor responses to website inputs. b. Arduino IDE programs ESP32 for motor direction via the website commands. c. Arduino IDE programs ESP32 for speed control via the website commands.	5. Upload a sample program to the ESP32 via the Arduino IDE, specifically designed for handling direction-changing and speed-changing commands. a. Test varied commands from the website and confirm consistent motor response. b. Verify PWM signal reduces speed by adjusting duty cycle.	Y

Appendix B Pseudo Code for Arduino IDE

```
// Define motor pins
Motor1: ENA, IN1, IN2
Motor2: IN3, IN4, ENB

// Setup function
setup():
    // Initialize pin modes for motors
    pinMode(ENA, OUTPUT);
    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    pinMode(ENB, OUTPUT);
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);

    // Initialize PWM channels for motor speed control
    //...

    // Initialize WiFi connection and server
    //...

// Function to handle user commands and update webpage
Webpage():
    // Construct HTML page with motor status and controls
    //...

// Main loop
loop():
    // Handle client requests
    //...

    // Control motor rotation based on direction, speed, and stop states
    //...

// Function to handle changes in motor speed
MotorSpeed():
    // Update motor speeds based on user commands
    //...

// Function to handle changes in motor direction
MotorDirection():
    // Update motor directions based on user commands
    //...

// Function to handle motor braking
MotorBrake():
```

```
    // Control motor stopping based on user commands
    //...

// Function to apply brake to motors
brake(int motor):
    // Stop the specified motor
    //...

// Function to rotate motors based on direction and speed
rotate(int motor, int value, int dir):
    // Rotate the specified motor in the specified direction and speed
    //...

// Start the system
setup();
// Continuously run the system
while (true):
    loop();
```