Wireless Remote Motor Controller

ECE 445 Design Document - Fall 2023

Professor: Arne Fliflet

TA: Jason Zhang

Group # 20

Aaron Chen, Kyungha Kim, Leeboon Sheng

Contents

1. Introduction

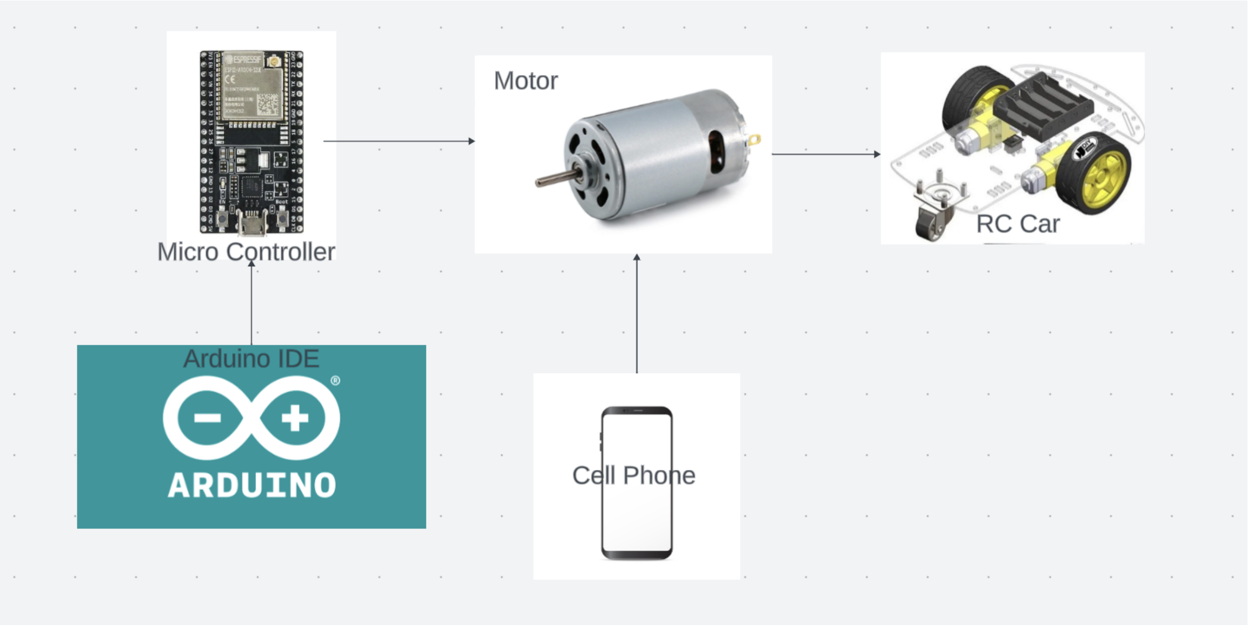
1.1 Problem

The need for efficient and convenient motor control is prevalent in various applications, such as robotics, automation, and remote-controlled vehicles. Existing solutions often lack simplicity and ease of use, making them less accessible to a broader range of users. Therefore, there is a demand for a wireless remote motor controller that is simple, user-friendly, and suitable for a variety of applications, including robotics and small wireless carts.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. As these applications become more commonplace, the need for efficient and user-friendly motor control solutions becomes increasingly critical. One of the primary challenges in motor control is the complexity of traditional systems. Many existing motor control solutions require users to navigate through intricate interfaces, memorize numerous button combinations, or undergo extensive training to operate effectively. This complexity not only limits the accessibility of motor control but also introduces the risk of errors and accidents, especially in high-stress environments. The demand for simplicity and ease of use in motor control is particularly evident in the field of robotics. As robotics technologies continue to advance and become more accessible to a broader audience, there is a growing need for intuitive motor control interfaces. The rise of remote-controlled vehicles and drones has created a need for wireless remote motor controllers that can operate seamlessly and reliably. These applications often involve real-time decision-making, precision control, and the need to adapt to changing environments. A simple and responsive motor controller can be the difference between a successful mission and a failed one. The concept of simplicity in motor control extends beyond robotics and automation. It also encompasses applications in everyday life, such as remote-controlled toys and gadgets. Users, especially children and casual hobbyists, should be able to enjoy the benefits of motorized devices without the frustration of navigating complex control interfaces. To meet these diverse needs for simplicity and user-friendliness, a wireless remote motor controller is required. Such a controller should prioritize ease of use, responsiveness, and versatility to cater to a wide range of applications.

1.2 Solution

Our project envisions the creation of a Wireless Remote Motor Controller, addressing the pressing need for a user-friendly, versatile, and adaptable motor control solution. With a primary goal of delivering an adjustable speed range of 0 to 100%, this controller aims to redefine motor control capabilities for various applications, from industrial automation to hobbyist robotics. The core of our controller's design is its wireless functionality. It will be compatible with wifi, allowing users to operate motors from a distance without the hassle of wired connections. This wireless control not only enhances convenience but also reduces the risk of tripping hazards and enables the use of motors in diverse and dynamic settings. Simplicity is paramount in our design philosophy. The controller will feature a user-friendly interface with essential functions like start, stop, accelerate, and decelerate. These functions will be accessible wirelessly through an app that can be downloaded directly through your phone, ensuring that users of all skill levels can operate motors effectively. Recognizing the demand for more complex applications, we will explore an alternative design that enables the control of a pair of motors. This dual motor control capability opens up exciting possibilities for steering mechanisms, making it ideal for building highly efficient robotic platforms or small wireless carts. To enhance precision and performance, our controller will implement closed-loop speed control. This feedback mechanism ensures that the motor operates at the desired speed regardless of external factors, contributing to consistent and reliable performance. Our controller will incorporate current limiting control to prevent overloading, protecting both the motor and the user from potential hazards. This feature contributes to the long-term durability and reliability of the motor control system.

1.3 Visual Aid



Visual Aid

1.4 High-level requirements

1) Wireless Control
    a) The system requires wireless control of a DC motor using Wi-Fi and allows remote control within a 5-10 meter range in open spaces.
2) Voltage and Current Control:
    a) The motor controller needs to support a specified voltage range, preferably 12-24V DC and handle a maximum of 10A current.
3) Speed Control
    a) Users should be able to control the motor's speed from 0 to 100%

2. Design

2.1 Physical Design

This description outlines the key components and features of the controller's physical design, including two motors, wheels, chassis, a ball caster, and other essential elements.
The motors are responsible for driving the wheels and, by extension, the entire system. They are placed on the chassis to provide the necessary power and control to move the device. We selected a brushed motor as controlling the speed of a brushed motor is relatively straightforward. By adjusting the voltage applied to the motor, you can control its speed easily without complex electronic circuitry. In addition to that brushed motors are inherently reversible. If you reverse the polarity of the voltage applied to the motor, it will change direction.

Image of the specific motor we plan to use

The wheels are essential components that enable the controller to move smoothly and efficiently. The design incorporates two wheels, each directly attached to one of the motors. The choice of wheels will depend on the intended application and terrain. We decided on these wheels as they were readily available and we were not too concerned about the type environment of where the car would operate.

Image of the wheels

The chassis serves as the structural frame of the controller, holding all the key components together. It is designed to accommodate the battery packs, PCB components, and other necessary electronics. The chassis is constructed from materials that balance strength, weight, and durability. It should be rigid enough to support the components and withstand any mechanical stress while remaining lightweight for optimal mobility. The chassis is specifically designed to house the battery packs, which provide the necessary power to the motors and other electronic components. Careful consideration is given to the placement and secure fastening of the battery packs to prevent unintended movement or damage.Within the chassis, the printed circuit board (PCB) components are strategically placed and securely mounted. These components include the motor control circuitry, the wireless communication module, voltage regulation components, and safety features. The layout of the PCB is designed for efficient heat dissipation and to minimize electromagnetic interference. The physical design may also include an enclosure or cover to protect the internal components from environmental factors, dust, and potential impacts. Accessibility features, such as removable panels or hatches, are incorporated to facilitate maintenance and component replacement.

To enhance stability and maneuverability, the physical design incorporates a ball caster. The ball caster is positioned at the front of the chassis and provides a point of contact with the ground that helps distribute weight evenly and allows for smooth directional changes. It ensures that the controller can pivot and navigate efficiently, especially when turning or changing direction.

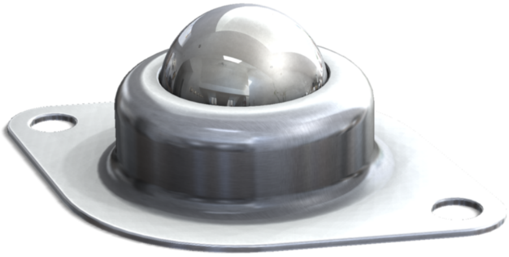
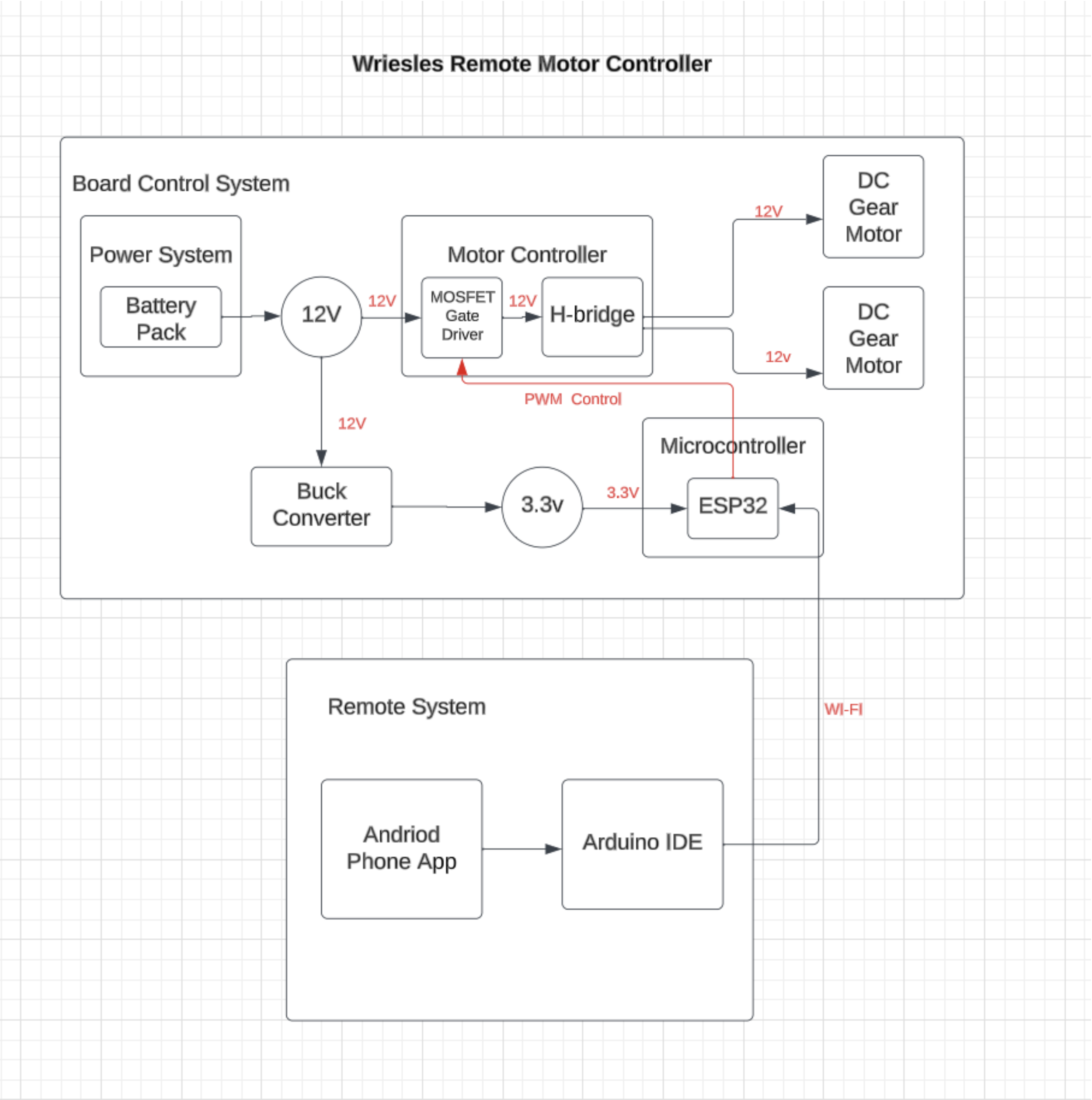


Image of the ball caster wheel

2.2 Block Diagram



Block Diagram

2.3 Functional Overview & Block Diagram Requirements

2.3.1 Board Control Subsystem

The Board Control Subsystem is responsible for receiving transmissions from the remote, taking data from the sensing subsystem, and commanding the electronic speed controllers. Based on these inputs, the board control subsystem will command the electronic speed controllers to power the motors accordingly. The Board Control Subsystem determines whether the DC geared motor will accelerate or decelerate and move in either forward or backward direction with a press of a button. The button's response time will be tested to be less than 1 sec in order to ensure a fast response time. In addition to being able to operate the movement of the motor, the motor controller will also be able to collect real time data from the motor and display the revolutions per minute (RPM) of the motor.

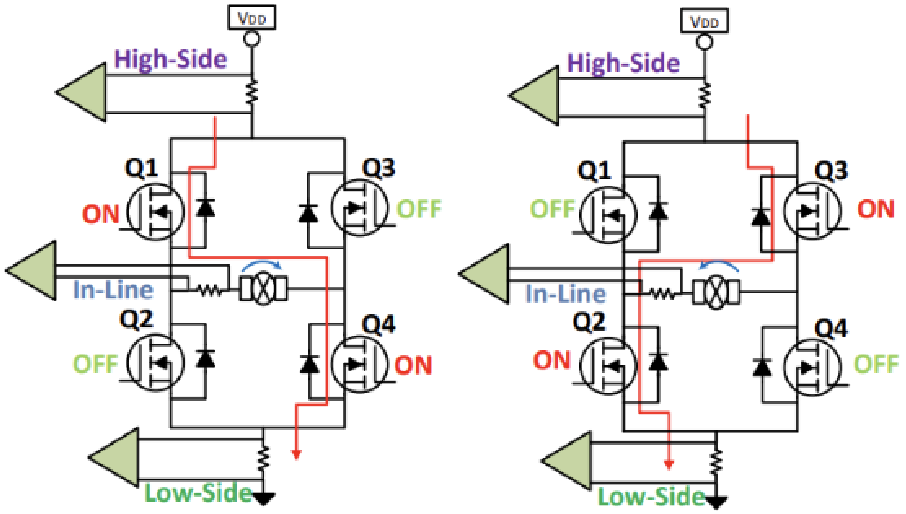
The microcontroller employed for this project is the ESP32S that interfaces with the motor controller. 3.3V DC supply is designed for the ESP32S microcontroller and it is used to feed PWM signals to the motor driver. PWM is achieved by varying the pulses applied to the enable pin of the H-bridge to control the applied voltage of the motor. The board system will then wirelessly communicate with the remote system via an Android Phone App designed using Arduino IDE. The DC gear motor will be driven by the H-bridge circuit. For more information on the software design of the Board Control Subsystem, please refer to Section 2.5.1.

Table 1 Board Control Subsystem requirement and verification chart

Requirement	Verification
The motor controller will be able to be controlled wirelessly.	● NO wires will be needed to connect the motor controller from the wireless remote.
The board control system will be able to receive and output data up to 12 meters in open space	● We will measure out exactly 12 meters from our wireless controller and see if the motor controller and RC car will be able to operate within that distance. ● We will also go beyond 12m to obtain the absolute max distance before the car disconnects.

2.3.2 Motor Controller Subsystem

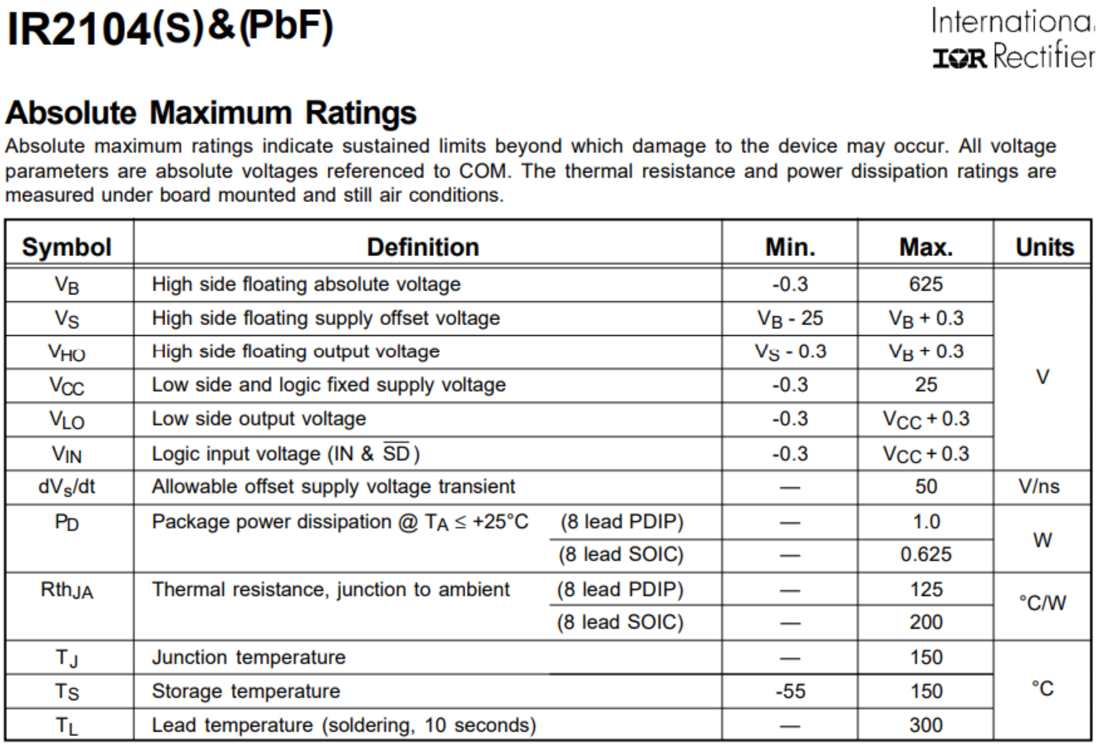
The **Motor Controller Subsystem** will include the hardware and software necessary to control the motor's speed, direction, and braking. The heart of this subsystem is the H Bridge includes four MOSFETS, two gate drivers for each side of the bridge with the associated bootstrap capacitors, and an Arduino Uno used to create the PWM signals that are fed into the gate drivers. The figure 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.



H-Bridge Circuit

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:
(.8 Ohms * 10 Amps) * 10 Amps = 80 Watts. IIt 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.

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 Volts + 12 Volts = 16Volts.** In order 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 safety 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 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.



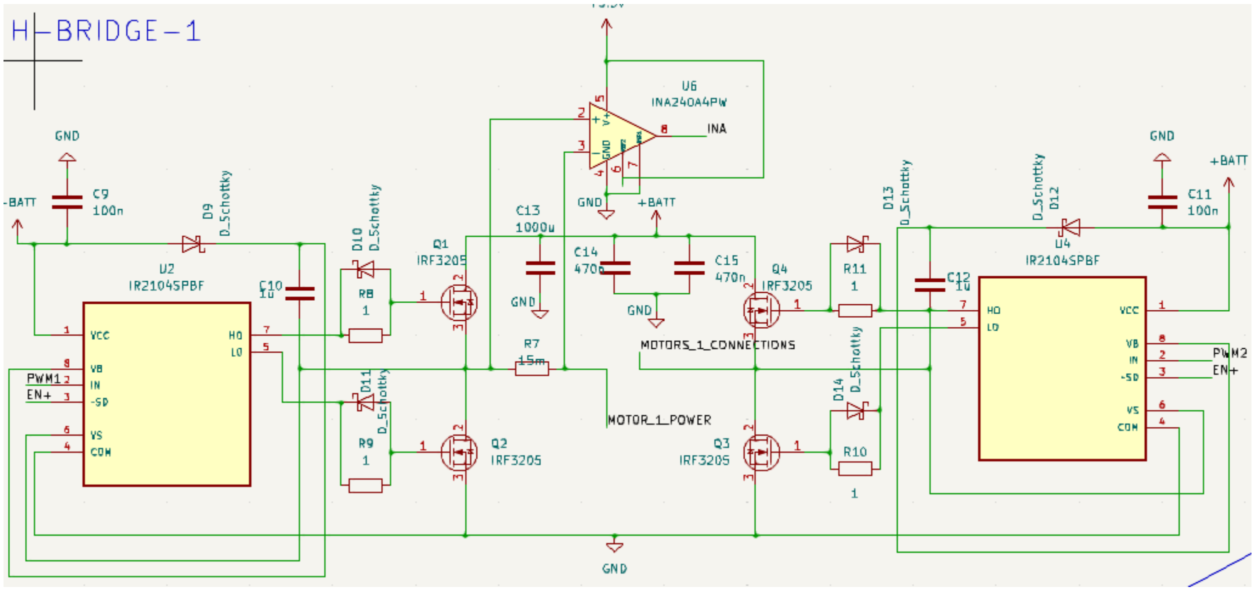
IR2104(S)&(PbF)

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	
Rth_{JA}	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	



Full H-Bridge Control Schematic

Table 2 Motor Controller Subsystem requirement and verification chart

Requirement	**Verification**
The motor controller will be able to withstand 12 V	● 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 ● To be safe we have also picked components like MOFSETs, gate drivers that are rated much higher than 12V.
The H bridge will accurately control the motors via the PWM input signals, allowing the user to control the direction and the speed of the car.	● We will test this by creating an obstacle and seeing if we will be able to redirect the RC car around the obstacles. ● We will combine the size of the wheels, the RPM and gear ratio of our DC geared motor to calculate the speed of our RC car.
The IRF3205 MOSFETs can take in ± 20V from the gate driver to be fully switched on.	● 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.
The IR2104SPBF gate driver has to be able to take in maximum supply voltage of 25V and 3.3V logic input.	● 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).

2.3.3 Power Subsystem

Power System for Motor Controller:

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.

12V DC Battery Input: The Heart of the Power System

At the core of the power system lies the 12V DC battery. This will serve as the primary energy source for the entire controller. The choice of a 12V 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.

3.3V Solder Jumper and Buck Converter: Powering the ESP32-S3 Module

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 12V 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 12V 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. If the Buck converter is not working properly the ESP-32 will be damaged, however with the solder jumper it will prevent this as it will redirect the high voltage. As mentioned before this will be monitored with our strategically placed test points that can be connected to a voltmeter.

Battery Voltage Sensor and Schottky Diodes: 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, however we will test each diode to make sure it does not turn on when operating backwards . 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.

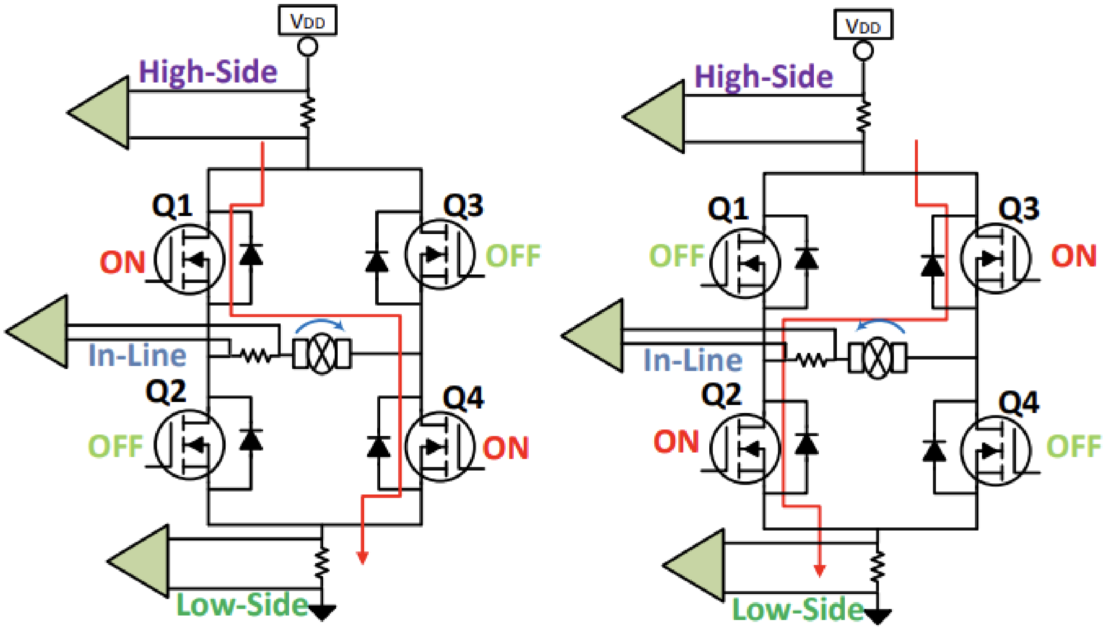
Table 3 Power Subsystem requirement and verification chart

Requirement	**Verification**
The power system will be able to limit the input voltage of the ESP 32 to only 3.3 Volts	● 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 output is a steady 3.3Volts.
The power system will be able to organize the voltage and current to each system for when it is needed.	● To verify, we will make sure the motor is first in a stationary state with the wheels not in motion. Take note of the RPM values for the left and right motors, as obtained from the board's

	microcontroller through serial debugging. Verify that both values are in close proximity to 0 RPM, with a permissible variation of +/- 5 RPM. ● Next, we will instruct both electronic speed controllers to establish a motor current. While both wheels are in motion, we will record its RPM to make sure it aligns with the calculations we have made. ● While doing this test, we will utilize the test points we implemented in our PCB design to monitor current and voltages in each subsystem, checking for any irregularities.

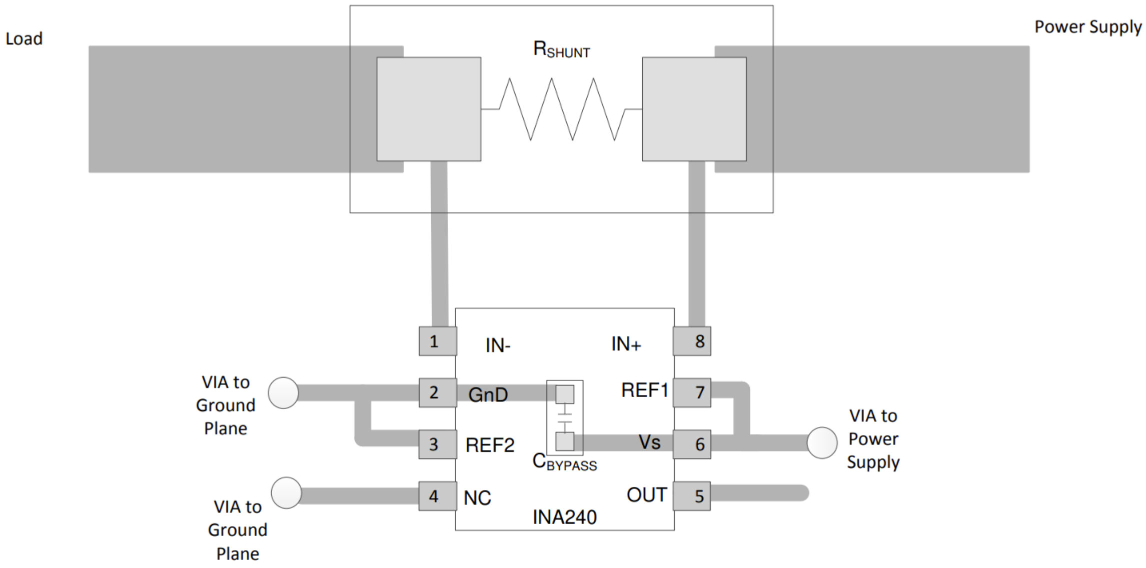
2.3.4 Current Sensing Subsystem

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.

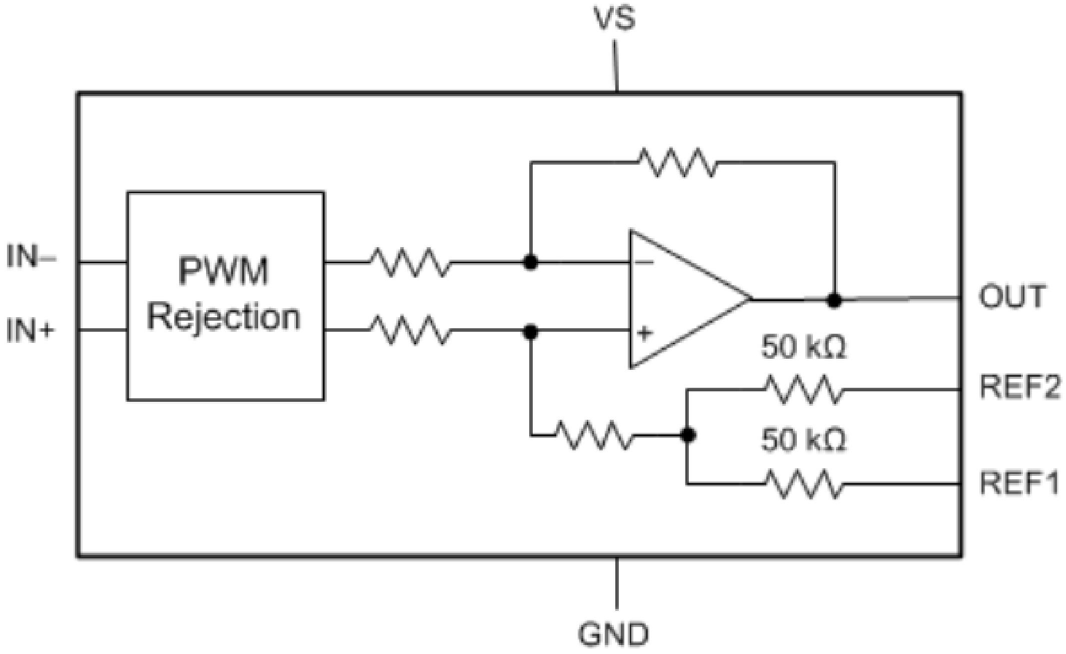


H-Bridge Circuit

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. 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.



This shows the functional block of the INA240 device.



Functional block of the INA240 device

Requirement	**Verification**
The current sensor will be to achieve current measurements in the H-bridge and feedback	- We will be using a multimeter to measure the current flowing across the

the signal to the ESP32 from the INA channel.	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. Is this good? ● After verifying the accuracy of the current sensor, we will connect the INA channel of the current sensor with the GPIO channel of ESP32 to read off the data.
The INA240 current sensor will be able to operate from a supply voltage ranging from 2.7 to 5.5V.	● 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.

2.3.5 Remote Android Phone App Subsystem

The Android Phone App 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.

Table: Android Phone App Subsystem - Requirements & Verification

Requirement	**Verification**
The microcontroller, esp32, must be able to interpret the status of the user input (accelerate, decelerate, move forward, move backward, or stop).	● Ensure that the Android phone app is in the default unpressed state. Record the data transmitted from the app to the ESP32. ● 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).
The Android Phone App shall establish a wireless connection with the Motor	● 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

Controller Subsystem.	to the Motor Controller subsystem. ● Confirm connectivity by checking if the Motor Controller Subsystem correctly responds to the commands by executing the requested actions.

2.3.6 Remote Arduino IDE Subsystem

Arduino IDE programs the ESP32 microcontroller. The code defines how the DC geared motors should respond to inputs received from the Android Phone App. For instance, if deceleration commands are sent via the app, the code lowers the duty cycle to reduce motor speed. In short, the Arduino IDE enables ESP32 to control and fine-tune the motor controller based on user inputs.

Table: Arduino IDE Subsystem - Requirements & Verification

Requirement	**Verification**
The Arduino IDE must program the ESP32 microcontroller to interpret and respond to commands from the Android Phone App specifically for changing the direction of the motor.	● Upload a sample program to the ESP32 via the Arduino IDE, specifically designed for handling direction-changing commands. Send a variety of direction-changing commands from the Android Phone App. ● Verify that the ESP32 consistently interprets and responds accurately to these commands, resulting in the intended changes in motor direction.
The Arduino IDE code should adjust the motor's speed based on input from the Android Phone App.	● Load the code into the Arduino IDE that corresponds to motor control based on deceleration commands from the app. Verify that the duty cycle of the PWM signal sent to the Motor Controller is lowered to reduce motor speed when deceleration commands are received. ● Program the Arduino IDE to respond to acceleration

	commands from the app and verify that the duty cycle is adjusted to increase motor speed accordingly.
The Arduino IDE must ensure that the motor controller responds to user inputs in real-time.	● Continuously transmit varying-frequency commands from the Android Phone App. ● Record motor controller response times. ● Confirm immediate action without noticeable delays, even under rapid command execution. ● Ensure consistent and prompt responses for real-time control of the DC motor.

2.4 Hardware Design

2.4.1 Operating Voltage & Regulation Strategy

In our project, we are designing a remote control system that utilizes an ESP32 microcontroller. To power these components, we plan to draw power from a 12 Volt battery or a 24 Volt source. To ensure seamless operation and compatibility with the ESP32's voltage requirements, we have decided to operate the microcontroller at a stable 3.3V with 3.3V logic levels. This voltage regulation is essential for reliable performance. Power to the system will primarily come from either the 12 Volt battery or the 24 Volt source, and we will incorporate an efficient voltage regulator to step down the input voltage to the 3.3V required by the ESP32. This voltage regulation is crucial to protect the microcontroller and other components from potential overvoltage issues. For convenient power supply and communication with the ESP32-S3 chip, we've included a Micro-USB port on the board. This port not only serves as a power source but also connects to the ESP32 via the on-board USB-to-UART bridge, enabling easy programming and debugging. Additionally, to provide visual feedback to users, we've integrated a 3.3V Power On LED that will illuminate when USB power is connected to the board. This LED indicator serves as a simple but effective way to signal the system's status to users or operators. In addition, we've taken steps to safeguard the ESP32-S3 module and other electronic elements by implementing two strategically placed Schottky diodes. These diodes serve a dual purpose: first, they protect the ESP32-S3 module's analog-to-digital converter (ADC), and second, they ensure that the voltage remains within the safe operating range, never exceeding 3.3V or dropping below ground potential. The choice of Schottky diodes is deliberate, as they offer low forward voltage drop and rapid switching characteristics, effectively limiting voltage and preventing unwanted spikes or deviations that could potentially harm the ADC or other critical components.

2.5 Software Design

The software is central to our project, as it controls the ESP32 microcontroller, allowing us to smoothly manage the motors with encoder. We've crafted the software using the Arduino IDE, making it easy to finely adjust motor speed and direction via wireless communication with a mobile device over Wi-Fi. We'll also consider the option of working with existing Bluetooth controller apps (such as Dabble - Bluetooth Controller) or creating a custom app based on our project progress.

2.5.1 Motor Control

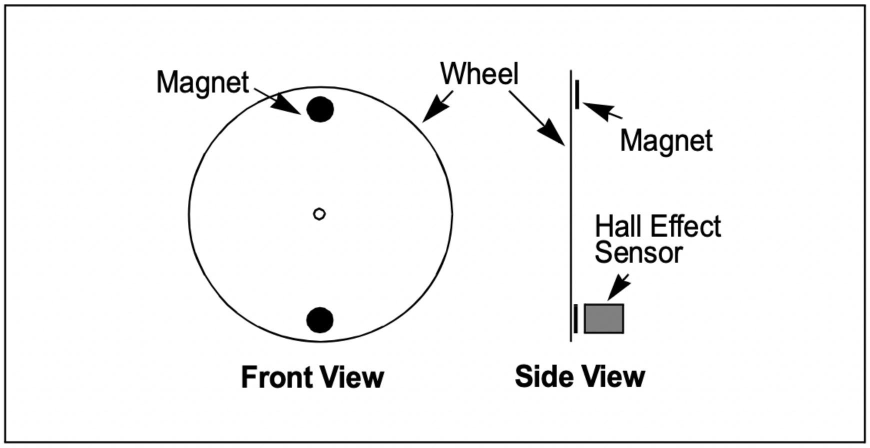
Our motor control algorithm is the core component of the software. It is responsible for interpreting user inputs and adjusting motor behavior accordingly. The algorithm takes into account a range of user commands and behaves as follows:

- Speed Control:
    - ACCELERATE: Command the electronic speed controller to increment the motor speed.
    - DECELERATE: Command the electronic speed controller to decrement the motor speed.
    - NO INPUT: If no input is received, gradually decrease the motor speed to zero, eventually coming to a complete stop.
- Direction Control:
    - STRAIGHT BACKWARD: Command the motors to move backward, rotating the front two wheels backward
    - STRAIGHT FORWARD: Command the motors to move forward, rotating the front two wheels forward
    - LEFT: Command to adjust the middle wheel (ball one) to the left, changing the motor's direction to the left.

- RIGHT: Command to adjust the middle wheel (ball one) to the right, changing the motor's direction to the right.

2.5.2 Hall Effect Sensors

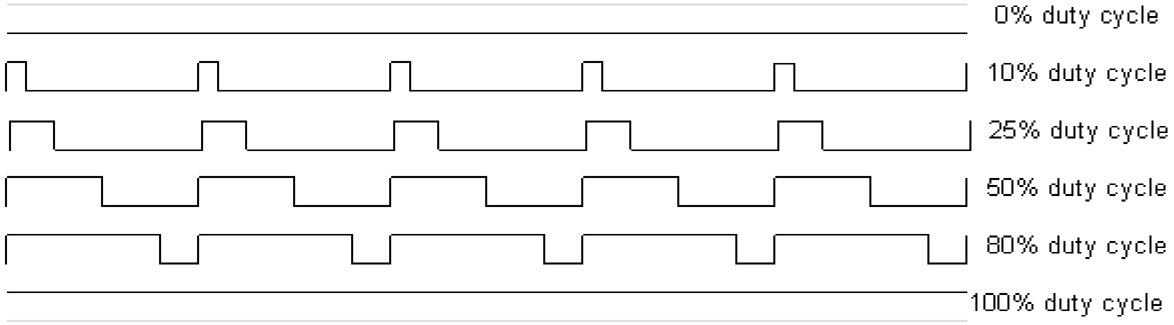
In our project, we use Hall Effect sensors from brushed DC gearmotors to provide accurate feedback on the motor's speed or RPM (Revolutions Per Minute). Placing a magnet on the rotating part of the motor and using a Hall Effect sensor nearby allows us to measure the strength of the magnetic field, which is proportional to the motor's speed. The following figure includes the fundamental components of the hall effect sensor. With these components, it determines the speed of rotating components (motors) at the correct sequence and timing. We are planning to use an encoder that provides 64 counts for every full revolution of the motor shaft. To determine the count per revolution at the gearbox output, we can simply multiply 64 by the gear ratio. We can calculate the speed using this algorithm in the Arduino IDE and display it on the screen to show the user the motor speed.



Hall Effect Sensor

2.5.3 PWM (Pulse Width Modulation)

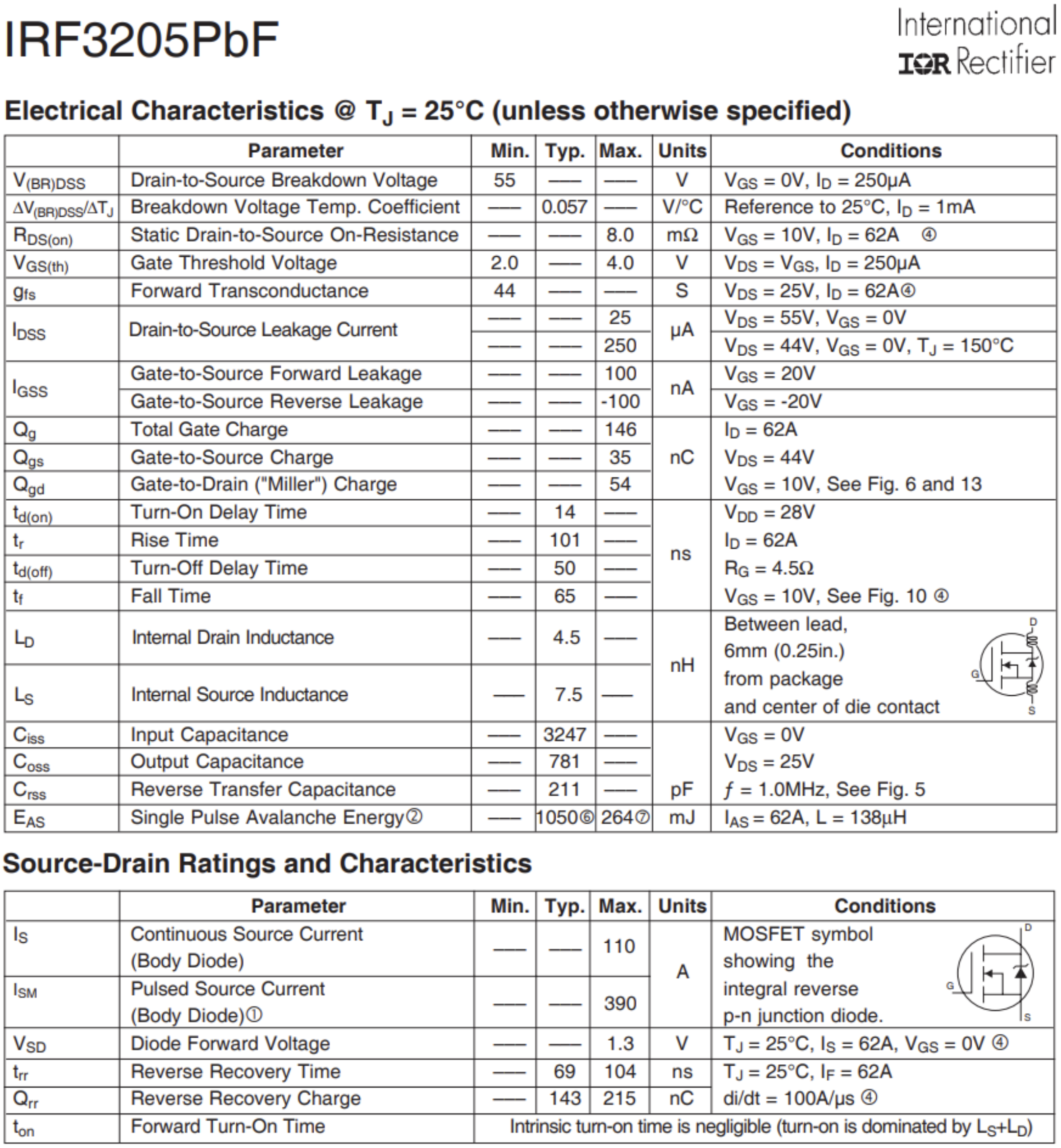
We use PWM to control the speed of motors in remote control systems. By adjusting the duty cycle of the PWM signal sent to the motor controller, we can control the average power delivered to the motor. A PWM signal is a square wave with a fixed frequency and a variable duty cycle. By expressing the duty cycle in percentage, we can represent the fraction of time during one cycle. The following figure 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.



2.6 Tolerance Analysis

- **Voltage Regulator**
  - Because we are using the ESP32-S3-WROOM, we do not require a linear voltage regulator. Our input voltage will be ranging from 12 to 24V, and the motor controller can accept input voltage of this range. ESP32 microcontrollers usually draw a maximum current of 240mA. The dropout voltage would be 24-3.3=19.7V and P=V*I=19.7*240m=4.728W dissipated in the regulator. Therefore, we would need to use a buck converter to step down the high voltage for the 3.3V for the ESP32-S3 module. This provides a much greater power efficiency than the linear regulators.
- **Choosing MOSFETs for H-Bridge**
  - The majority charge carriers of p-channel MOSFETs are holes; it has a lower mobility compared to electrons. This has resulted in higher on-resistance of p-channel MOSFETS compared to n-channel MOSFETS. The formula for power is I^2R, to minimize the power dissipated in the H-bridge circuit, four N-MOSFETs are used at the low side and high side instead of two n-channel types at the low side and two p-channel types at the high side.
  - According to the datasheet, IRF3205 MOSFET has a drain-to-source breakdown voltage of 55V. It is important to keep the voltage across drain-to-source below 55V, so the drain current will not exceed 250µA. To avoid the inductive spike during switching transients exceeding the breakdown voltage, we will derate the voltage range by 60-70%. Since our system uses 12-24 DC, we will use the MOFSETs with a breakdown voltage rating of at least 20.4-40.8V. This range of voltage value is below 55V so the MOFSETs will be able to handle inductiver spikes. Besides, the Rds(on) is specified at 10V VGS and our input voltage is ranged from 12 to 24VDC and the mosfet will be fully switched on.
  - All MOSFETs have in-build reverse Schottky body diodes to prevent the DC brushed motor's reverse current spikes. Some capacitances of value 1000uF, and 470uF will be placed in between the MOFSETs to reduce noise.

- **Choosing Gate Drivers for MOSFETs**
  - In order to drive the MOSFETs, we will need the gate drivers to create a high enough charge to activate the high side MOSFETs in the H-bridge. The logic input voltage(VIN) for the gate driver we chose, IR2104SPBF gate driver, is compatible with 3.3V which is the same voltage supply to ESP32.
  - For the gate driver, the supply voltage(VCC) is from the battery source(12V DC). This is in a safe operating range as the gate driver can take in a maximum voltage supply of 25V. Besides, this value is lower than the maximum VGS of the MOSFET which is ± 20V to prevent the MOSFETs from being destroyed. Last but not least, the bootstrap voltage,VB, is the same as the supply voltage.



IRF3205PbF

International IOR Rectifier

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.)
L_S	Internal Source Inductance	——	7.5	——		from package and center of die contact
C_{iss}	Input Capacitance	——	3247	——	pF	V_{GS} = 0V
C_{oss}	Output Capacitance	——	781	——		V_{DS} = 25V
C_{rss}	Reverse Transfer Capacitance	——	211	——		ƒ = 1.0MHz, See Fig. 5
E_{AS}	Single Pulse Avalanche Energy②	——	1050⑥	264⑦	mJ	I_{AS} = 62A, L = 138μH

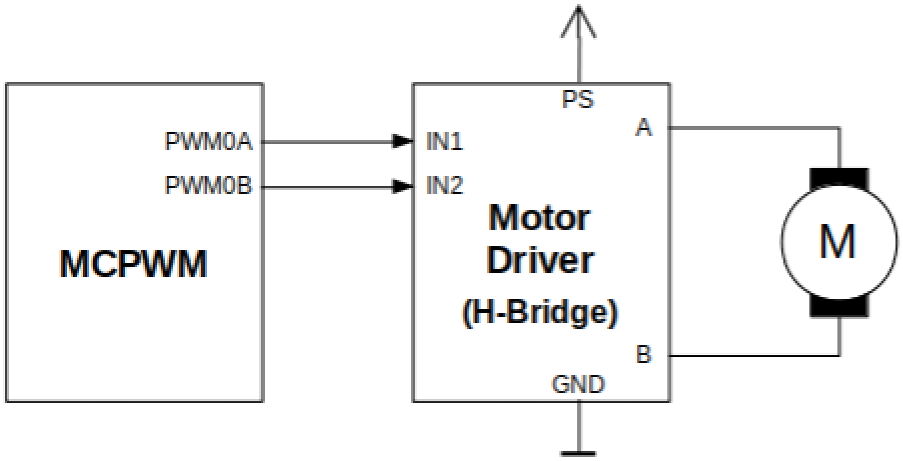
Source-Drain Ratings and Characteristics

	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)				

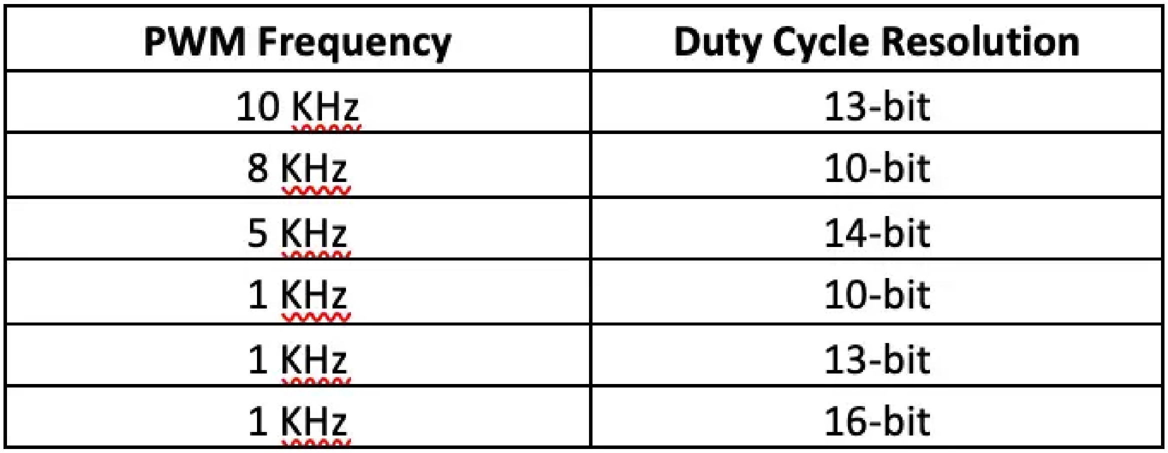
- **Capacity of Microprocessor**
  - The ESP32 microcontroller is a dual-core processor, Wi-Fi and Bluetooth connectivity, and a variety of I/O options. The figure below shows how MCPWM works with H-Bridge. Since we will achieve speed control using the PWM waveform and duty cycle, it is important to discuss the PWM frequency of the microprocessor. The user is allowed to specify PWM frequency from 1kHz to 40MHz and duty cycle resolution from 1 bit to 16 bits.Let’s say with a theoretical frequency of 80MHz, the maximum PWM frequency for ESP32 can

be calculated using the formula below: **Max_PWM_freq = 80,000,000/2^Duty-Cycle-Resolution-Bit**The following table lists some of the recommended PWM frequencies and duty cycle resolution for some common applications.

- The ESP32 microcontroller also has an inbuilt dead time generator that allows the user to specify the dead time on rising edge and falling edge. Besides, we can also generate signal pairs(PWMxA and PWMxB) to be active high, active low, active high complementary, and active high complementary. Furthermore, if we wish to integrate deadtime in the generator module, we are able to bypass the dead time generator module.



Example of Brushed DC Motor Control with MCPWM



PWM Frequency	Duty Cycle Resolution
10 KHz	13-bit
8 KHz	10-bit
5 KHz	14-bit
1 KHz	10-bit
1 KHz	13-bit
1 KHz	16-bit

2.7 Cost Analysis (shopping list)

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. In addition to the parts themselves, we will anticipate a compensation of $40 per hour for each team member working 2.5 hours, resulting in $6,000 per individual. When multiplied by the number of team members, the total labor cost amounts to $18,000 which does not include overtime. Overall the total spending for this project comes out to be around $18,118.

Item:	Cost
2x Current sensors	$6.46
13-V synchronous buck controller with fixed 3.3-V output	$0.63
2x Brushed DC Motor Gearmotor 300 RPM Incremental 12VDC	$29.76
ESP32-S ESP32-S3-WROOM-1-N8R2 Transceiver; 802.11 b/g/n (Wi-Fi, WiFi, WLAN), Bluetooth® 5 2.4GHz Evaluation Board	$3.35
8x MOSFET N-CH 100V 9.7A TO220AB	$12.80
BATTERY PACK NIMH 12V AAA	18.99
PCB	$5
2xGATE driver	$10.68
12x Diode	$1.64
12x Resistors	$1.54
6x Capacitor	$1.32
Inductor	$0.29
Total	$92.46+tax(6.25%)+shipping

2.8 Schedule

Week	Task	Person
Oct. 2nd - Oct. 8th	Order parts for prototyping	Everyone
	Start PCB Design(Power system)(H Bridge)	Aaron, Boon
	Research ESP32 voltage inputs	Kyungha
Oct. 9th - Oct. 15th	Continue PCB Design	Everyone
Oct. 16th - Oct. 22nd	Revisions to PCB Design	Everyone
	Begin Board Assembly	Aaron, Boon
	Start writing code for speed control and app for remote	Kyungha
Oct. 23rd - Oct. 29th	Individual Progress Reports	Individuals
	Continue Board Assembly	Aaron and Boon
	Begin Arduino IDE	Kyungha
Oct. 30th - Nov. 5th	Continue Arduino IDE	Kyungha
	Revisions to Board Assembly	Aaron, Boon
Nov. 6th - Nov. 12nd	Finalize remote System	Everyone
	Integrate Board Control system and Remote System and Test	Everyone
Nov. 13rd - Nov. 19th	Mock Demo	Everyone
Nov. 20th - Nov. 26th	Fall Break	-
Nov. 27th - Dec. 3rd	Final Demo	Everyone
Dec. 4th - Dec. 10th	Final Presentation and Paper	Everyone
	Lab Notebook	Individuals
	Lab Checkout	Everyone

2.9 Risk Analysis

Controlling a Wireless Remote Motor Controller demands a deep understanding of motor control mechanisms. One critical risk involved signal loss between the remote controller and the motor might lead to loss of control or unintended motor behavior. To mitigate this risk, we will employ robust wireless communication protocols and implement error-checking mechanisms to minimize the chance of signal disruption. Extensive testing across diverse environments will be conducted to ensure the reliability of the wireless connection. Another significant concern is the possibility of malfunctions or software bugs in the controller leading to unintended acceleration or deceleration of motors, posing a safety hazard. To prevent such incidents, we would design the algorithm to stop under emergency. Plus, rigorous testing, including simulated emergency scenarios, will be performed to ensure the controller's ability to swiftly and safely respond to anomalies. Additionally, the complexity of the mobile app's user interface poses a risk of user confusion, incorrect inputs, and accidental adjustments that could result in accidents or damage. To address this, we will prioritize a user-friendly and intuitive mobile app interface design, conduct usability testing with potential users, and provide clear instructions and safety guidelines within the app.

3. Ethics and Safety

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 end 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. I, Aaron Chen, Kyungha Kim, and Lee Boon Sheng Adhere to this.

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.

7.No specific author. "ESP32 Technical Reference Manual." Espressif, https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf#mcpwm.

8. 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.

9. Jason🙂