Project Lab Final Report

Jonah Jordan
Electrical and Computer Engineering
Texas Tech University
Lubbock, United States
jonahjor@ttu.edu

Kassav Shakya
Electrical and Computer Engineering
Texas Tech University
Lubbock, United States
Kashakya@ttu.edu

Chinedu Anijekwu
Electrical and Computer Engineering
Texas Tech University
Lubbock, United States
canijekw@ttu.edu

***Abstract*— Battle bots are remotely controlled, armored fighting robots designed to destroy an opponent in an enclosed arena. This paper presents the design and construction of an infrared (IR) controlled battle robot. To build this compact IR-controlled battle Bot, we made use of dual MSP430 microcontroller units (MCU), one of which was used for IR signal transmission and the other used for reception of the signal and control of the two DC motors and attack mechanisms. Our battle bot made use of pulse width modulation (PWM) for speed control. Our bot also made use of MOSFET-based current amplification in order to supply our emitter (IR LED) with enough current for signal transmission and made use of an IR receiver for decoding commands received from the emitter. Our PCB, housed our emitter, pushbuttons (for movement in the four cardinal directions), and our attack mechanisms trigger for the hammer, will allow for rapid configurations while ensuring we adhere to the rules of the tournament.**

KEYWORDS

PWM: Pulse width Modulation

MCU: Microcontroller

DC: Direct current

LED: Light Emitting Diode

GPIO: General-Purpose Input/Output

PCB: Printed Circuit Board

ISR : Interrupt Service Routine

I. Introduction

Autonomous and remotely controlled robots have become the focus of many modern robotics competitions worldwide. The goal of this Battle Bot project is to construct a compact, IR-controlled robot capable of receiving directional commands in real time from the emitter and also activating two distinct attack mechanisms. The system to design has been constrained or forced to make use of a single 9.6V battery, fit in a 16×16×16 inch enclosure, and have a maximum mass of 10 pounds. The main objective of this project is to have a working battle bot (Figure 7) with robust IR communication, minimal response delay, proper motor control, and effective overcurrent detection, all while ensuring we are within the rules of the rubric and comply with all safety protocols.

II. Hardware

The hardware kept on top of the rover primarily consists of an MSP430 board, a comparator circuit on a breadboard and an H-bridge that helps control the DC motors inside the rover chassis.

The pins from the first MSP430 board are connected into an H-bridge that has been set up to control the DC motors placed inside the chassis. The vital pins of H-bridge used in this system are the enable pins, direction pins and the power supply from the 9.6V battery. The enable pins and direction pins take in digital input from the MSP430, therefore allowing movement of stated motors.

The comparator has been set up to compare voltage (Figure 3) from the sense pins of the H-bridge during normal motion to the voltage when the rover is stalled i.e. overcurrent is produced. A potentiometer is then used in order to provide continuous 1V voltage to Input1+ of the comparator, while the sense pin is connected to Input1- so that continuous comparison can take place. In a condition where the system detects overcurrent i.e. Sense pin voltage > 1V, overcurrent protection takes place.

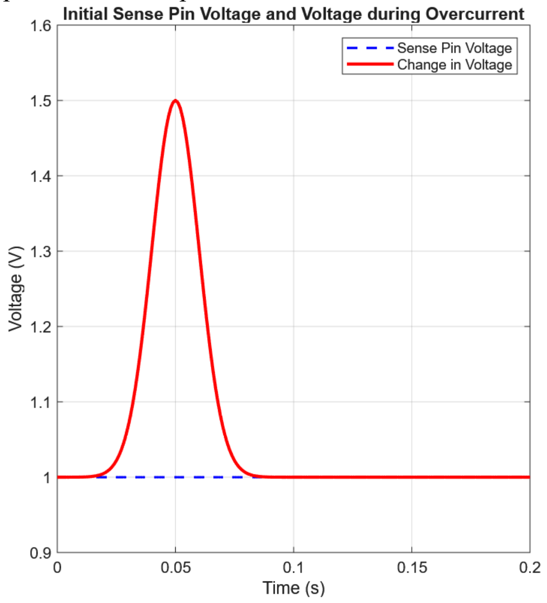


Figure 1: Expected graph for Overcurrent condition

For the hardware of the IR transmitter, the PWM pin is connected to the IRLB8743 MOSFET. The MOSFET is being used to help drive the IR LED high current requirement. The IR LED is connected to the power supply using a 36 ohm resistor to get it close to the 100mA threshold found in the data sheet. This lets the signal travel much further and meet the requirements of the project. The emitter circuit also has 4 push buttons connected to it which will send out the signals for the robot. These buttons are connected to GPIO pins and pulled down to ground using 220 ohm resistors. On a test circuit we have connected LEDs to GPIO pins to make sure the button functions are working. These will soon be removed and the buttons coded to drive the rover. The IR transmission circuit was eventually built on a PCB based on our design (shown in Figure 2). The hardware for the receiving circuit is fairly simple, The receiver is connected to a GPIO pin, power, and ground. A test LED is also connected to test if the receiver is picking up the transmission.

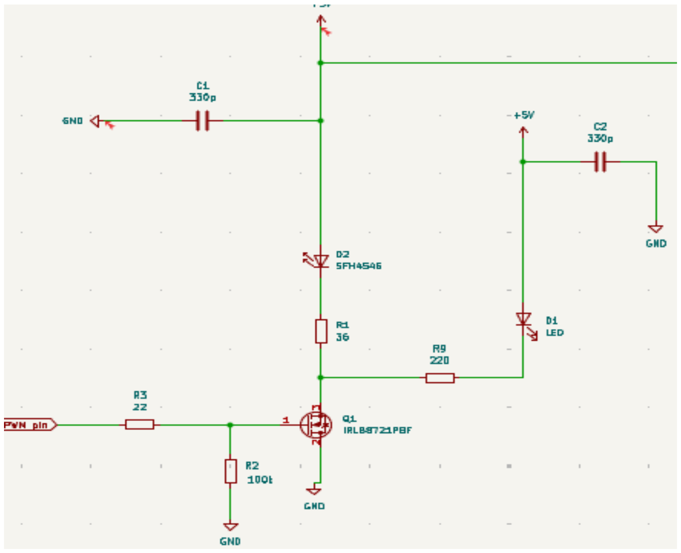


Figure 2: Part of PCB schematic

III. PCB Design

1. *Designing/emulating*

The PCB (Figure 3) for the rover controller was designed with an emphasis on signal integrity, durability, and practical usability. There are a few key differences in the design from our original Hardware configuration. Ground planes were incorporated on the board to provide low-impedance return paths, while grounding through-holes were added to maintain consistent reference potential across layers. Because additional board space was available, the primary trace widths were increased from 10 mils to 50 mils to enhance current capacity and improve mechanical robustness. The IR LED was positioned on the front of the board to ensure clear line-of-sight operation. A 1x10 connector pin was used for connecting the PCB and the MSP. The user-input buttons were arranged in a D-pad configuration to provide an intuitive and ergonomic control interface. A few extra buttons were added in case we wanted to create more attacks for our rover in the future. This PCB had two layers, and there were traces on both sides since the paths cannot cross over each other. After using the built-in design rule checker as well as the electrical rules checker, our PCB was confirmed to be fully functioning. The figure below (Figure 3) is the finalized KICAD design that was sent in to be ordered and printed.

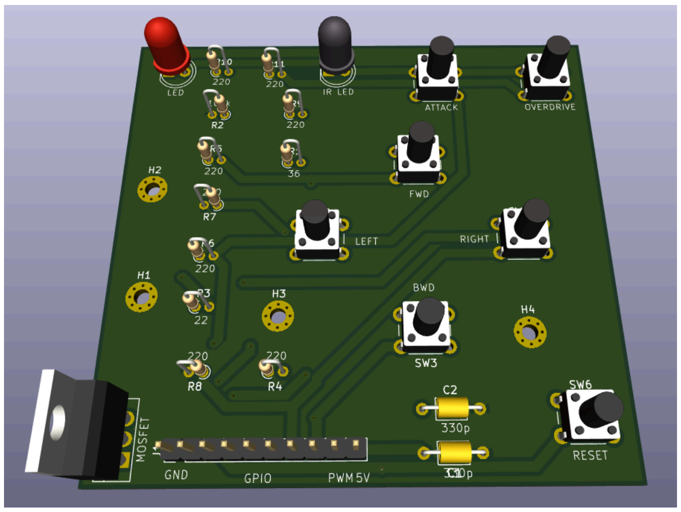


Figure 3: Emulated PCB

2. *Assembling/Debugging*

Though the design passed all automated checks, a slight issue remained. To make the schematic cleaner, the gate, drain, and source pins of a MOSFET were rearranged so that wires would not cross and the design would be easier to read. However, this change was not transferred to the PCB layout. After receiving the boards and soldering all components, the PCB produced a highly distorted signal. Upon reviewing the schematic and layout (Figure 8, (appendix)) , the mismatch was identified. Three jumper wires were soldered to correct the MOSFET connections, and the controller then operated as intended. Another small change that was made was increasing the pull-down resistor values. We got access to larger resistors (10k ohms), so we implemented them on all the pull-downs to further limit current.

IV. Software

The software flow utilizes multiple different versions throughout the project. Firstly the mini project has the motor movement functions as well as overcurrent detection, which will later be implemented into future versions when they have been tested and safely implemented. There is a function for emitting IR signals, and a function for decoding the signals being sent. All of these programs are built and compiled using Code Composer Studio.

1. *Mini Project:*

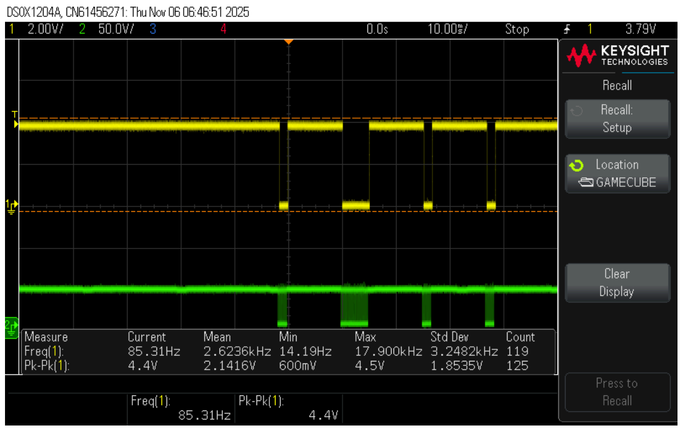
The motor control system operates by manipulating the H-bridge driver inputs to achieve basic movement states: forward/backward motion, turning, and stopping. Forward motion is achieved by setting both of the motors in the same direction, though this configuration will need correction to enable both motors for true straight-line movement. The perpendicular turn is accomplished through differential drive, where Motor A rotates in reverse while Motor B rotates forward, causing the robot to pivot about its center axis. Stopping is implemented by setting all motor driver inputs to LOW, allowing both motors to coast freely until friction brings them to rest. (Going backwards, and turning the opposite are the same process but the logic is inversed.) Speed control for both motors is managed through PWM signals operating at 20 kHz with a 75% duty cycle(Figure 4). There is also overcurrent detection that is mostly detected through a hardware set up but the software shuts everything off if the comparator goes HIGH for a short amount of time. (longer than a spike).

2. *IR emitting software:*

The approach for emitting the IR signal starts with setting the PWM pin (2.7) to supply 38 kHz and setting up the timers. Then the main loop runs continuously until power is cut off.. Each of the 4 buttons is programmed to send out a different amount of time the IR emitting led is turning on and off. This is done by setting the LED to stay on and off using clock cycles. The minimum burst length needs to be greater than 10 carrier cycles, and the minimum gap after each burst has to be greater than 12 cycles. Setting a cycle to be 0 and a different cycle being 1, we can string these together to create binary commands telling our rover what to do. On top of this, it is programmed to send out a “key” bit, which makes it to where other groups can't interfere with our rover.

3. *IR receiving software*:

The IR receiver software is responsible for detecting and breaking down the infrared commands emitted by the transmitting code. After setting up the TSOP receiver input pin (P3.1) as well as enabling the GPIO settings that are needed, the software monitors the status of the signal all the time. When it detects the TSOP output goes LOW, it indicates the 38 kHz IR burst. The length of HIGH and LOW intervals is then counted by the software in clock cycles to recreate the binary command pattern taken from the transmitter. A string of pulse sequences is continuously read as a series of binary bits, which is then compared with pre-stored command patterns to determine which button was used on the transmitter. To avoid interference, the code checks a specific "key" value at the start of every transmission prior to performing any command. Following the detection of a valid signal, the system takes the corresponding action, whether it is moving the rover or executing an attack. The current state of this code is using LED’s to see the signals being received, but the implementation of the mini project motor functions is soon to be added.



```
Measure    Current    Mean       Min       Max        Std Dev    Count
Freq(1):   85.31Hz    2.6236kHz  14.19Hz   17.900kHz  3.2482kHz  119
Pk-Pk(1):  4.4V       2.1416V    600mV     4.5V       1.8535V    125

Freq(1):           Pk-Pk(1):
        85.31Hz             4.4V
```

Figure 4 : Emitter Signal (Green) & Receiver Signal (Yellow)

4. *Transmission Criteria*:
   a. *Initial:*

According to our initial project goal, the program for our transmitter was set up such that a signal, with its respective gaps, would be sent out, as long as the button was pressed, therefore allowing for an easy to control rover with pretty responsive feedback to our input.
The receiver code was then set up to carry out the specified action if and only the signal was actively being received. So with this setup, as long as the right signal was received, the action would take place and then stop when that signal was not received. This was achieved through a switch-case that continuously monitored the signal received from the transmitter, paired with an ISR that would take our rover into a rest state when an expected signal was not received.

   b. *Demo Day:*

But during demo day, a criteria was set in place in order to avoid interference in the contestants’ rovers. Signals had to be set up to only send about 2 signals, that the rover would have to decode, and continue conducting specific action until another instruction was sent. In order to achieve this, the ISR was removed from our initial code and then the emitter was set up to provide only 2 signals of burst for every rising edge.

V. Attack Mechanism

The rover incorporates a compact set of 3D printed offensive and defensive attachments designed to enhance performance during physical interactions. Figure 5 presents the CAD model of the rover with all mechanisms integrated, and Figure 6 shows the completed physical rover.

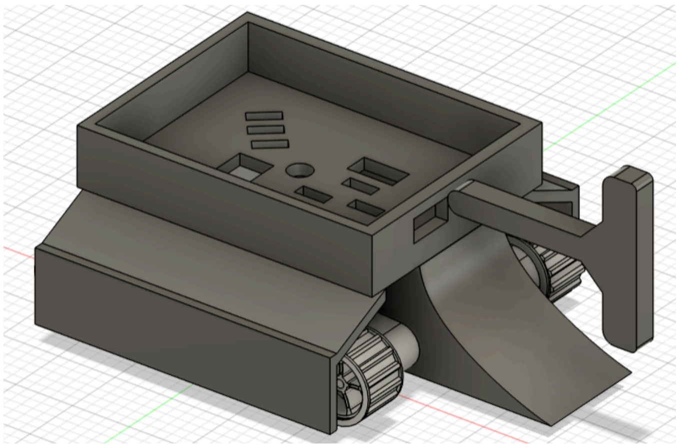


Figure 5 : CAD model of the rover

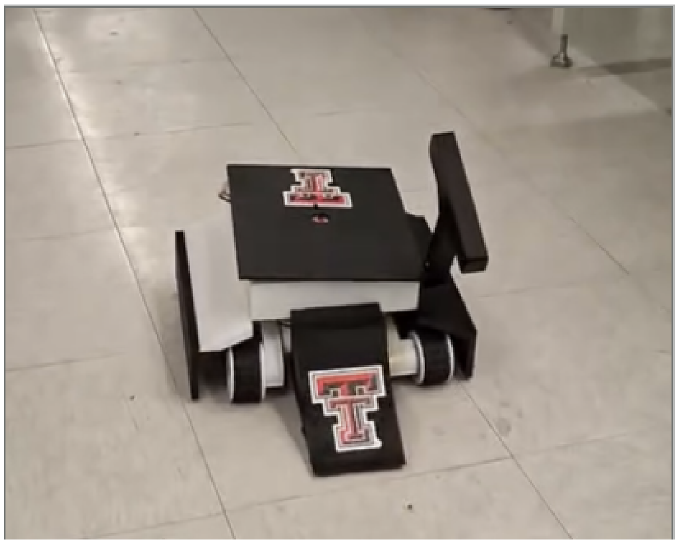


Figure 6: Final Product of the Rover

1. *Hammer Mechanism*

A lightweight 3D printed hammer was mounted on the upper chassis and actuated using a micro servo. The hammer is triggered through an infrared (IR) communication link: when a valid IR command is received, the microcontroller outputs a PWM signal to rotate the servo and deliver a downward strike on its opponents. The design balances mass and speed to maximize impact while keeping servo load low.

2. *Wheel Shields*

Two 3D printed shields were added to protect the drivetrain from side impacts. These attach to the chassis and cover the wheels without restricting motion, reducing the likelihood of entanglement or direct damage during collisions.

3. *Front Ramp*

A simple 3D-printed ramp was installed at the front of the rover to destabilize opposing robots by lifting their wheels or shifting their center of gravity on contact. This mechanism is fully passive, ensuring high reliability with no added electrical complexity.

4. *Software:*

The hammer attack was implemented on the FT5330M servo motor. In order for the MSP to drive this servo, a PWM signal must be provided. The duty cycle of this signal is what determines the position of the servo. In order to mimic a hammering motion, the duty cycle was changed from initial value to desired value for a specific amount of time, and then back to the initial value to get the hammer back to normal.

One thing worth noting is the timers used in order to generate the specific frequency required for each component. To run the motors, we used TimerB that generated close to 20Hz. Since the servo requires a frequency of about 50Hz, pins utilizing that timer would not work in this case. Due to this, the PWM provided to the servo was through a pin that utilized TimerA.

VI. Results and testing

Over the course of this project, we have had to perform comprehensive testing across the PWM motor control, IR communication, and our overcurrent detection system. As a team, we have been able to validate each of these components for both electrical accuracy and real world responsiveness under normal operating conditions.

1. *PWM motor control:*

We were able to generate PWM signals using the MSP430 module at about 20khz shown in Figure 4. This frequency provided our bot with a smooth torque response while minimizing the amount of noise coming from the bot. Left and right motor control through the H bridge driver enabled our bot to have instantaneous turning, which we were able to achieve by turning the treads or wheels of the bot in opposite directions without halting our rover. The PWM duty ratio was adjusted through software and was set at 75% allowing for effective speed control and maneuverability.

2. *IR Transmission:*

The IR LED was modulated at 38kHz(f_c)in order to match the detecting bandwidth of the receiver. We were able to do multiple tests involving IR transmission. The first and easiest test is using a phone camera. The human eye can't see IR light but a smartphone camera is able to. Though it is a very crude way of testing it is a good test to see if your circuitry is right as well as all of your components functioning properly. In slow motion videos you are able to see that it is flickering at different rates depending on which button is being pressed. For precise testing we were able to connect the oscilloscope to see exactly what is going on.

3. Duty Cycle Calculation:

In order to balance transmission intensity with LED longevity, we set the carrier signal to a 33% duty cycle using the formula

$$D = Ton / T * 100\%;$$

Where:

D = Duty cycle,

T(period) = $T = 1 / f_c$ = 26.3µs

T_{on} (ON time) = 8.7µs, resulting in a duty cycle of 33%. Graph of the signal shown in Figure 4.

4. Overcurrent Detection:

Motor overcurrent detection was achieved through a comparator based overcurrent circuit. A potentiometer was used to establish our reference voltage, which represented our maximum allowable current, while the motor's shunt voltage was what served as the sensed input. Whenever the sensed voltage exceeded the reference, the comparator output disabled the motor driver and triggered the fault LED on the MSP430.

5. Equations:

The calculation of time period is required merely to assess the time it takes for a specific signal to be sent through the emitter. The result of the following formula is then multiplied by the number of desired cycles.

$$T = 1 / f$$

Where :

T = Time Period

f = Frequency, 38000 for infrared

When setting up the circuit focused around the IR emitter, current through it could not exceed 100mA, due to which a resistance value that would be below 100mA but as close to it, to maximize safe usage, was needed. The following formula was used to calculate the resistance.

$$R = V/I$$

Where :

R = Resistor value to be used

V = Voltage supplied

I = Desired current

VII. Discussions

While the tournament itself on demo day ended abruptly for our team, the progress we were able to make and the product we were able to showcase at the end of the course is something this team is very proud of. Our expected timeline for progress and predicted parameters for this project were pretty close to what we ended up getting. With responsive remote control, a functional attack mechanism and a sleek body for our rover, we believe this project to be satisfactory.

Overcurrent protection was a feature we wished to have on this rover at the end, but the time constraints forced us to skip it for the demonstration.

VIII. Conclusion

In conclusion, through the combination of many tasks, some being hardware design decisions, others being coding logic, we have successfully made a “battle-ready” bot for demo day. As discussed before, our complete system uses two MSP430 microcontrollers. One of them will be connected to our custom PCB that emits different IR signals based on what buttons are pressed. The other MSP will be connected to the chassis, which will be receiving the signals being sent to it, decoding them, and driving the robot using the decoded signals. Over the last few weeks, the receiving circuit soldered to a protoboard, was placed into our robot armor and the PCB consisted of the transmission circuit. Considering all the progress we made and the knowledge we have gained through research, this project was a massive success, and a remarkable result of teamwork.

References

[1] Z. Ghassemlooy, “Digital pulse interval modulation for IR communication systems,” International Journal of Communication

[2] Suha I. Al-nassar, “Article Title,” DJES (Digital Journal of Engineering and Science),

[3] Texas Instruments, MSP430FR58xx, MSP430FR59xx, and MSP430FR6xx Family User’s Guide, SLAU367P, Rev. Apr. 2020. Dallas, TX: Texas Instruments, 2012–2020.

[4] Infineon Technologies AG, IRLB8743 Power MOSFET Datasheet, Rev. 2.0, Munich, Germany, 2015.

[5] Texas Instruments, LM324-N Low-Power Quad Operational Amplifier Datasheet, SNOSC16L, Rev. Dec. 2019.

[6] Vishay Semiconductors, TSOP382 Series Miniature IR Receiver Modules for Remote Control Systems, Document 82491, Rev. Jun. 2020.

[7] Adafruit Industries, TSOP382 Infrared Receiver – Product Datasheet (Reprint of Vishay TSOP382), 2020.

Appendix

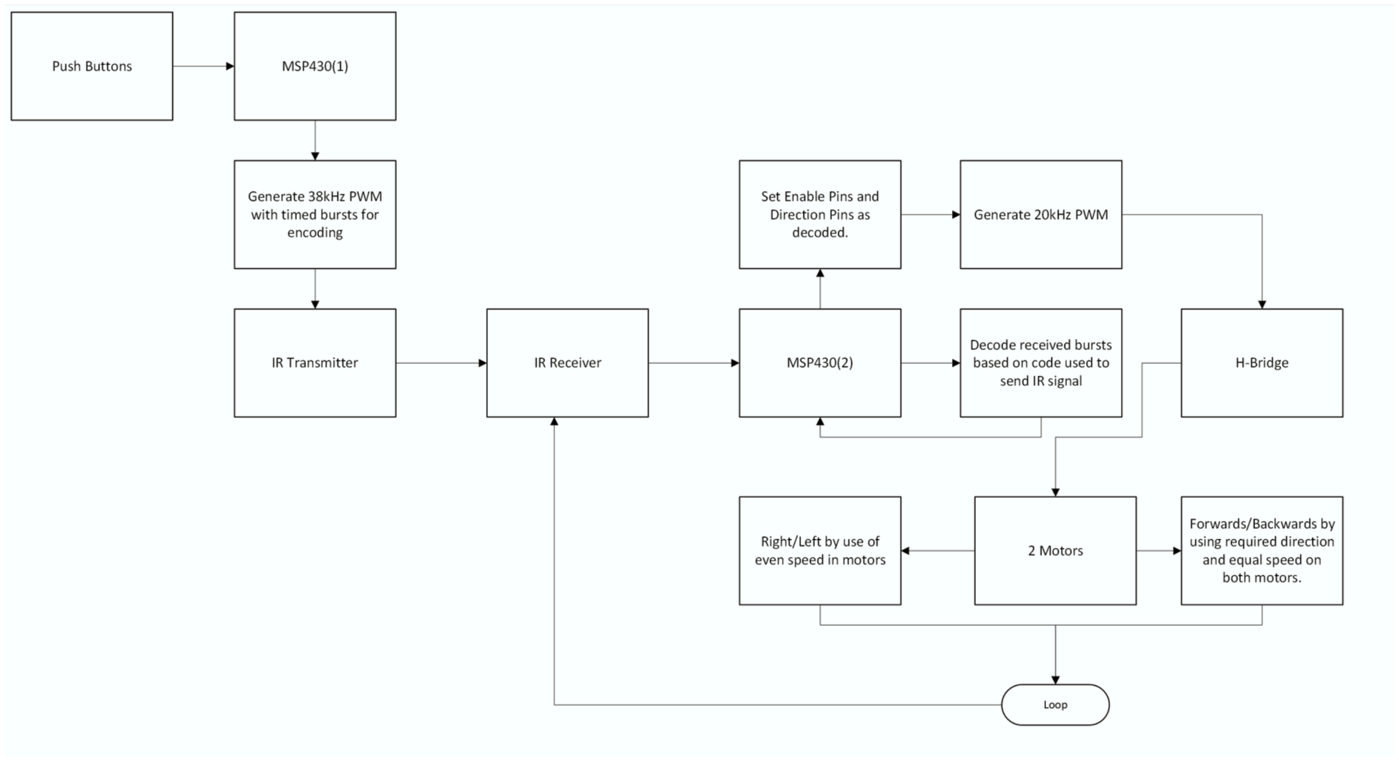

Push Buttons
MSP430(1)
Generate 38kHz PWM with timed bursts for encoding
IR Transmitter
IR Receiver
MSP430(2)
Set Enable Pins and Direction Pins as decoded.
Generate 20kHz PWM
Decode received bursts based on code used to send IR signal
H-Bridge
Right/Left by use of even speed in motors
2 Motors
Forwards/Backwards by using required direction and equal speed on both motors.
Loop

Figure 7: Flowchart

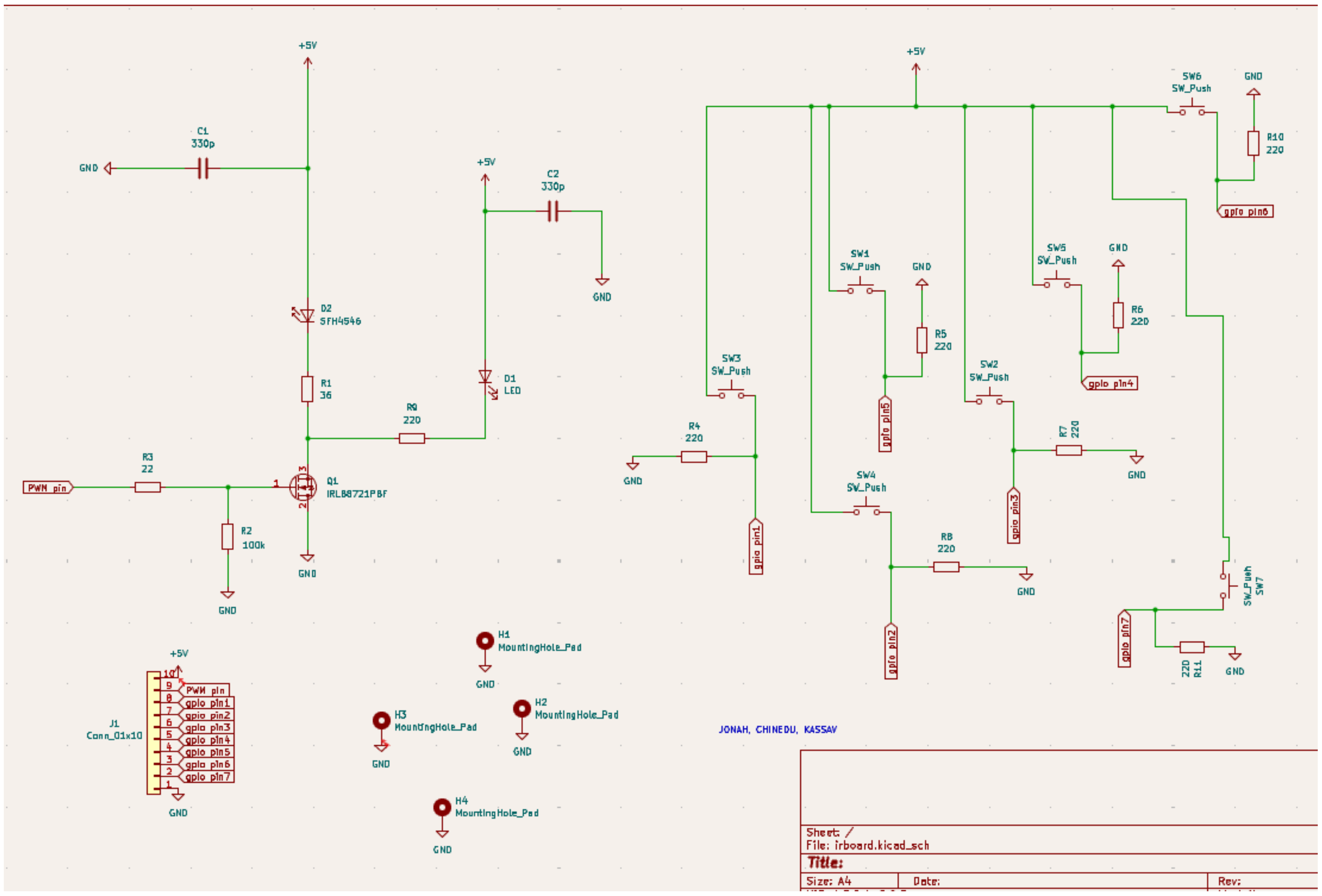


Figure 7: Full PCB schematic