

ECE 445  
Senior Design Laboratory

Design Document

# **Antweight Battlebot**

Team number 15  
Daniel Rodriguez  
Carlos Carretero  
Troy Edwards

3/6/2025  
TA: John Li  
Professor: Viktor Gruev

<b>Introduction</b>	<b>3</b>
<b>Problem</b>	<b>4</b>
<b>Solution</b>	<b>4</b>
<b>Visual Aids</b>	<b>4</b>
<b>High-level requirements list</b>	<b>5</b>
<b>Design</b>	<b>6</b>
<b>Block Diagram</b>	<b>6</b>
<b>Subsystem Overview</b>	<b>6</b>
<b>Cost Analysis</b>	<b>16</b>
<b>Ethics and Safety</b>	<b>17</b>
<b>References</b>	<b>19</b>

# Introduction

A complete description of the Antweight battle bot that will be created for competitive applications.

## Problem

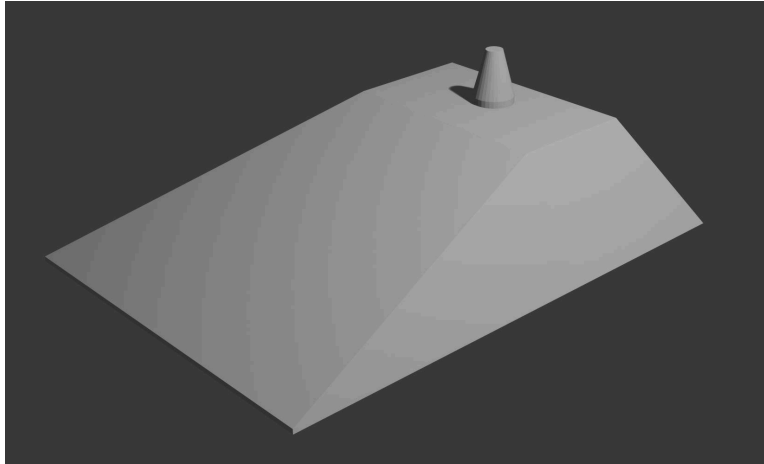
Our project revolves around the Ant Weight Battle Bot competition held by Prof. Gruev. The competition is an elimination-style battle where each team's robot fights to survive 2-minute rounds against their opponent. This competition has many restrictions and requirements that introduce design challenges and considerations.

- The Battle Bot must be under 2 pounds in weight.
- Only 3D-printed parts are allowed for the chassis and weapon and can only be made out of PET, PETG, ABS, or PLA/PLA+.
- The robot is to be controlled via a Bluetooth or Wifi Enabled Microcontroller.
- The functional movement must be displayed with an indicator LED showing power is on.
- The battery voltage can not exceed 16V.
- A manual disconnect must be included for safety.
- If a pneumatic weapon is used it is limited to 250 psi. The system must also have an easily accessible bleed valve.
- Spinning weapons must come to a complete stop after 60 seconds of power disconnect.
- Finally, a custom PCB must be implemented.

## Solution

For our battle bot to be competent in battle creating a strong defense would be more important than creating a glass cannon. Therefore, for our design, we decided upon a wedge-shaped bot that is as low to the ground as possible to ensure it can't be flipped over. As for a weapon, it will have a protruding lever allowing it to flip opponents that end up on top of it. This entire design will be 3D printed using PETG filament. The battlebot will be controlled through a Bluetooth controller that will send signals to the WiFi antenna that is found onboard the ESP32 Microcontroller. This microcontroller takes the inputs that were received from the controller and sends them to the motors that spin the wheels of the robot as well as the motor that controls the battlebot's weapon system. This will all be powered using a LIPO battery at a higher voltage that will be stepped down using a regulator for the microcontroller and its original voltage fed directly to the motors.

## Visual Aids



*Figure 1: Battlebot Prototype*



*Figure 2: External controller (laptop using wifi)*

## High-level requirements list

- The battle bot must not exceed 2 lbs with all required components for proper functionality. This includes motors, wheels, chassis, weapons, and onboard electronics.

- A strong wifi connection between -50 dBm to -60 dBm to must be established allowing for consistent and responsive movements from the input device.
- Competent movement with
  - Velocity of 4.5 feet/s
  - Wheel Rpm of 900 revolutions per minute
  - Independent bidirectional wheel movement

## Design

### Block Diagram

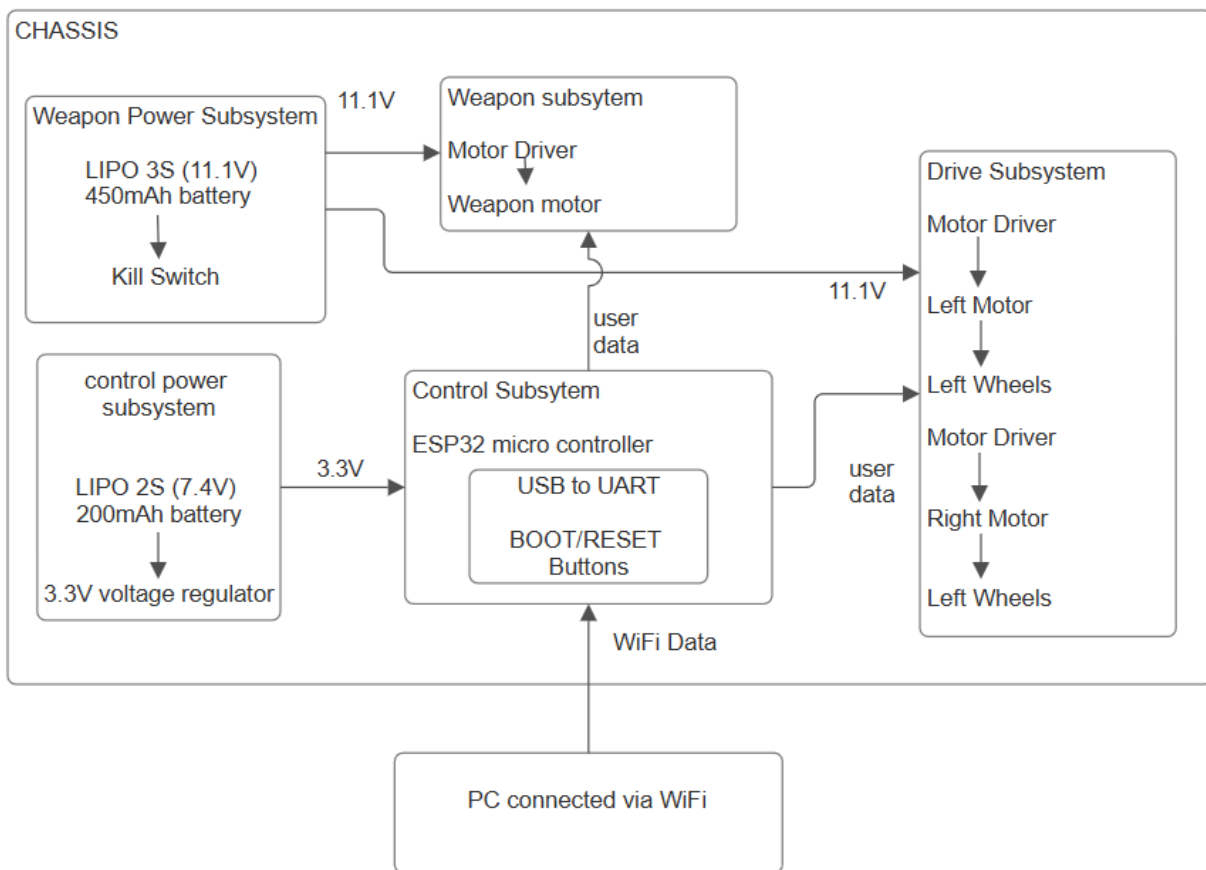


Figure 3: System Block Diagram

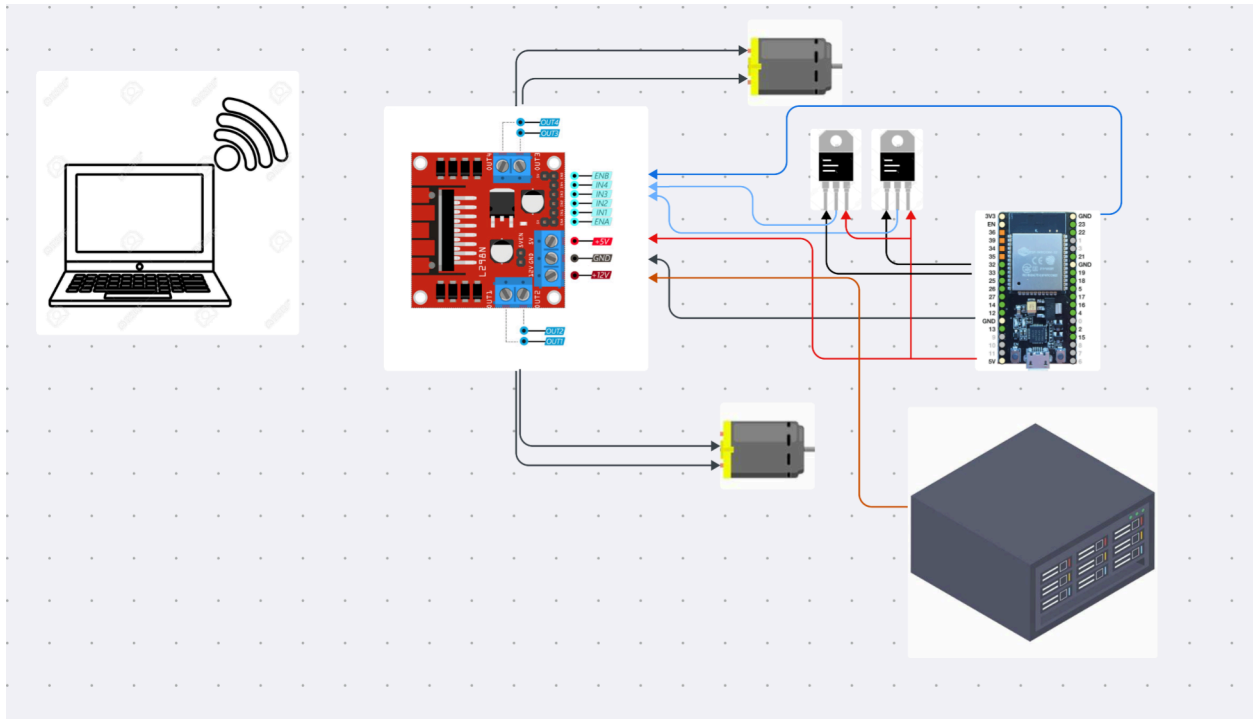


Figure 4: Breadboard Demo Block Diagram

## Subsystem Overview

- Chassis:** 3D printed using PETG to ensure the highest durability as we want any attack to be deflected. Although heavier it will ensure that we can create a strong chassis that can withstand attacks as well as retaliate. The dimensions of this subsystem are yet to be determined as we don't have our PCB completed. This will most likely be one of the final subsystems that will be finished as most of the concrete design choices will be determined by the sizes of our other subsystems and PCB design.

Requirements	Verification
Must be a total weight under 2 lbs.	The chassis will be weighed and the PCB will be weighed and then the total will be added up to hopefully be under 2lbs.
The chassis must be structurally sound to withstand various impacts.	The final chassis design will be dropped from a height of 2m and will be expected to not break.

The chassis must be 3D printed and fit the PCB.	The PCB will be placed inside the chassis and it must close. It must not bulge or bend the PCB.
Must have a front wedge of at least 30 degrees.	We will use a protractor to ensure that the front has a wedge of at least 30 degrees.

- Drivetrain:** The battle bot is going to move using a set of 2 wheels on each side powered by 1 motor per pair of wheels. These motors are going to be connected to the wheels almost directly and the voltage will be limited to ensure correct rpm. The motors will be limited by the MCF8316A1VRGFR motor drivers we are using. The motors we will be using are

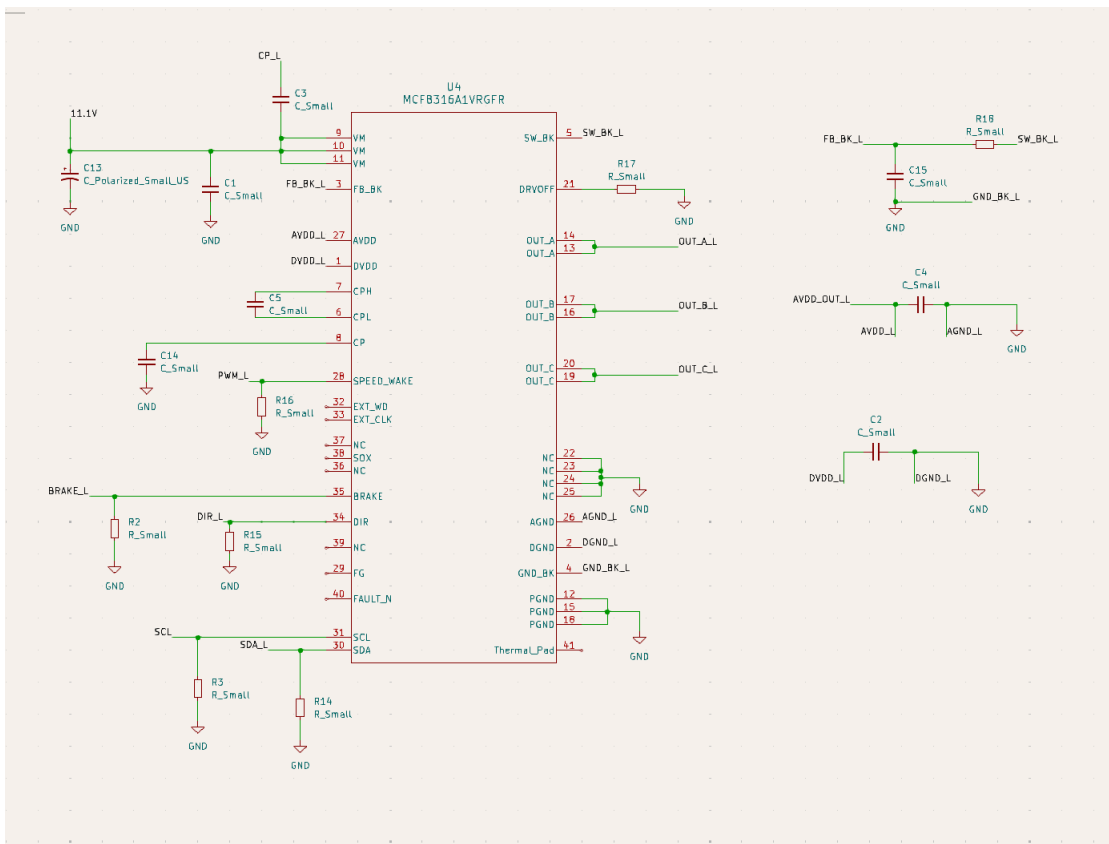


Figure 4: Motor Driver Circuit Diagram

Requirements	Verification
The motors must spin at an RPM of 900	This will be verified using the FG signal

rpm.	coming out of our motor drivers. They work off of I2C which then tells us the exact speed of the motors.
The motor drivers must not spike the voltage coming off of the battery.	The oscilloscope will be used at the power input of the motor driver and the outputs out to the motors to ensure the voltage is not spiking.
The motors must be able to break effectively and change directions. For the braking, we must be able to stop in under 1 foot when going 4ft/s.	Signals from the ESP32 will be sent to ensure that the motors can reverse. This will be done by going forward and backward repeatedly. For the braking, we will make the car run at 4ft/s and then mark a line 1ft away and attempt to stop.
The motor drivers must be properly initialized using the SCL and SDA I2C signals.	We will know that this worked if the motors start to spin. If no movement can be made it may mean other connections don't work but the I2C signals may be the cause. The ESP32 must be made to return responses after we set new speed profiles or initialize the motor drivers.

- **Power:** To power all the motors we are using an 11.1V 3s LiPO battery with a capacity of 450mAh. This amount of capacity will ensure that the battle bot will be able to run for 2-3 minutes. Due to it having 3 cells and a voltage of 11.1 Volts, we can power high torque motors. This battery will be connected through manual disconnect to satisfy the safety requirements. As for the esp32, we are using a secondary battery that is a 7.4V 2s LiPO battery with 200 mAh capacity. We have to add a voltage regulator to decrease the voltage to 3.3V for the ESP microcontroller. Both batteries will be connected to a manual disconnect through a JST plug as the battery is already terminated in a male JST connection.
  - a. Our esp32-wroom-1 microcontroller requires a voltage of 3.3V to operate and has a minimum operating current of 0.5A. Therefore the voltage regulator we choose, the TLV1117-33CDCY, must output 3.3V and also ensure a current higher than 0.5A.
  - b. For our motors that are being controlled by the MCF8316A1VRGFR motor driver we must ensure that the voltage is at least 4.5V which is why we are using a second 11.1V battery to power this subsystem. Although not necessary for our application, this battery is able to output the peak



current output of 8A from the motor driver due to its 45C rating. This is enough for all three motors.

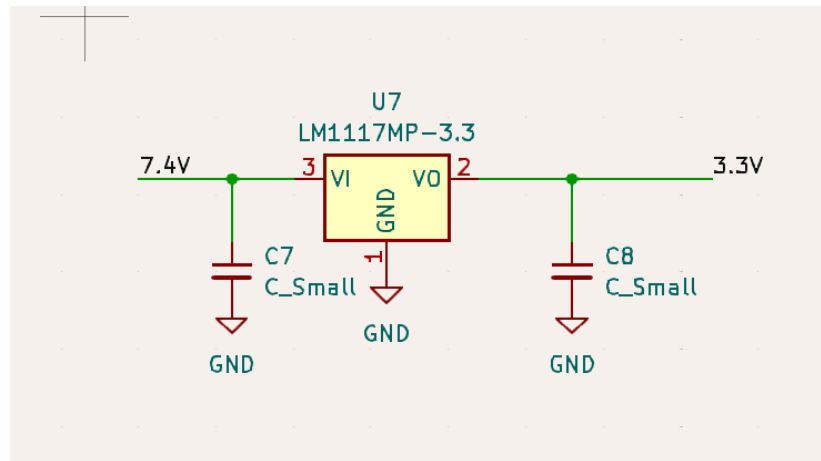


Figure 5: Voltage Regulator Circuit Diagram

Requirements	Verification
The regulator must supply a constant 3.3V	Measure the voltage at the output of the regulator using a voltmeter
Voltage must be stepped down from 7.4V to 3.3V	Compare the voltage from the input of the regulator and the output with a voltmeter
The regulator must supply a constant 0.8 Amps to the microcontroller	This will be measured using an oscilloscope allowing us to see a consistent 0.8A
2nd Battery must be able to handle a continuous discharge of 24 Amps	We will run the motors at max power to ensure that the rating is sufficient in an event were the motors spike. This will be monitored on an oscilloscope to keep a record of our max motor current pull.
Disconnecting either battery will have the subsystems discharge the residual current	We can verify using an oscilloscope connected to ground ensuring the current drops to zero.



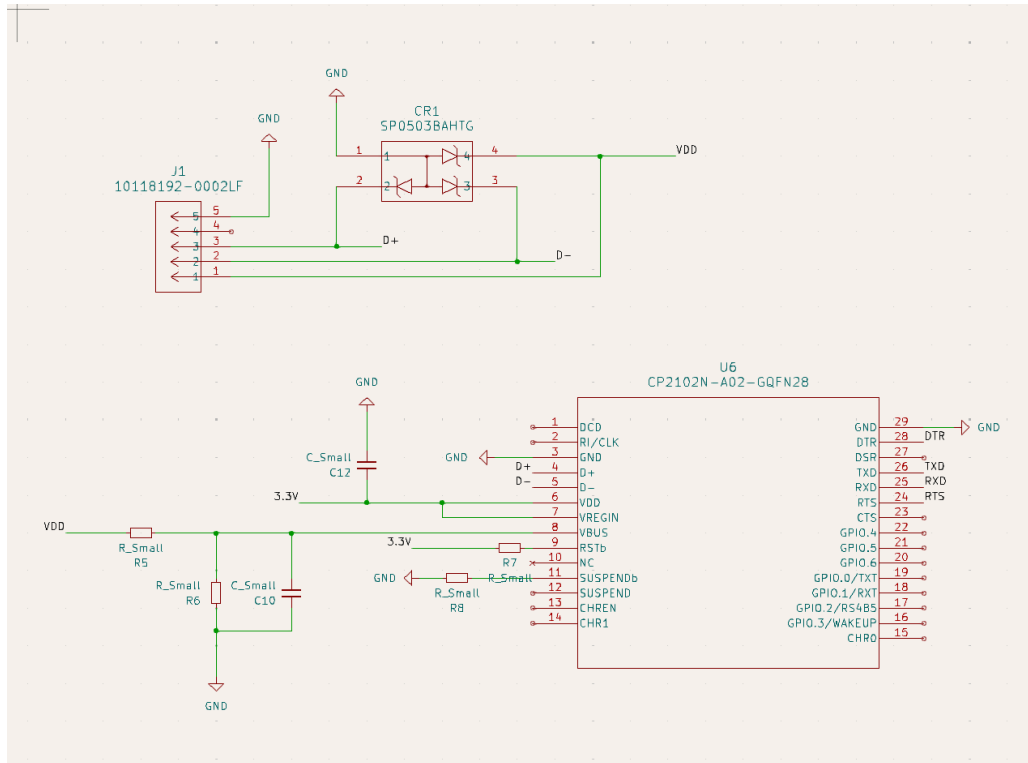


Figure 7: USB to UART Circuit Diagram

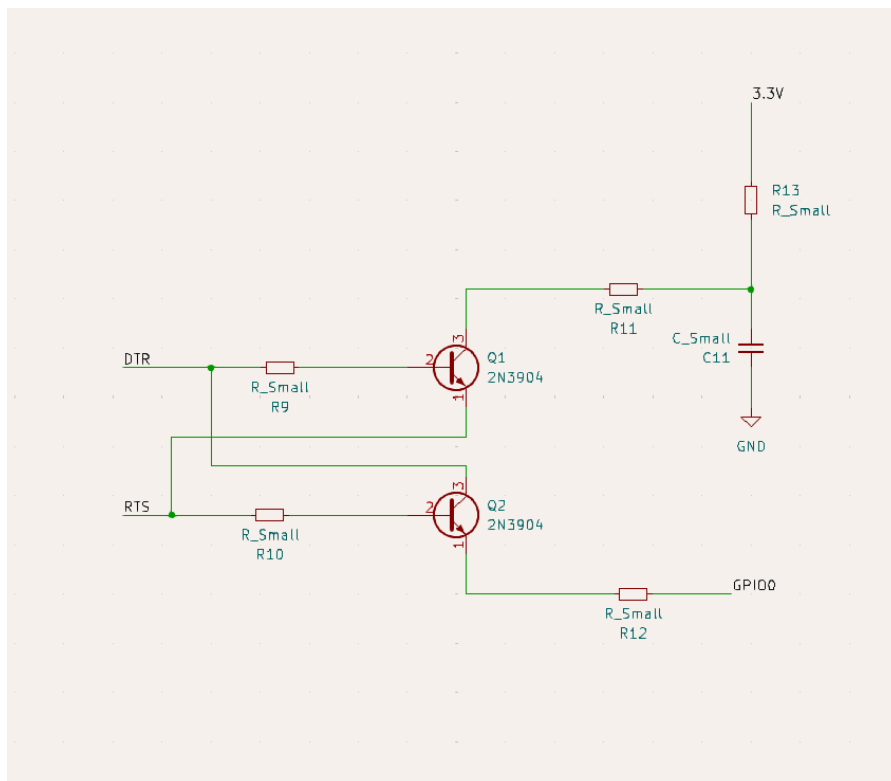


Figure 8: Boot Circuit Diagram

```

1  import socket
2  import keyboard # Install via 'pip install keyboard'
3  import time
4
5  esp32_ip = "192.168.137.120" # Replace with your ESP32's actual IP
6  port = 8080
7
8  # Create a socket and connect to ESP32
9  client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10 client.connect((esp32_ip, port))
11
12 print("Press 'W' for 3.3V, 'T' for 3.0V, and 'P' for 50% PWM output.")
13
14 # Track key states to avoid duplicate sends
15 key_states = {"W": False, "T": False, "P": False}
16
17 while True:
18     # Check 'W' key state
19     if keyboard.is_pressed('w') and not key_states["W"]:
20         client.sendall(b"W\n")
21         print("Sent 'W' → 3.3V output HIGH")
22         key_states["W"] = True
23     elif not keyboard.is_pressed('w') and key_states["W"]:
24         client.sendall(b"W_RELEASED\n")
25         print("Sent 'W_RELEASED' → 3.3V output LOW")
26         key_states["W"] = False
27
28     # Check 'T' key state
29     if keyboard.is_pressed('t') and not key_states["T"]:
30         client.sendall(b"T\n")
31         print("Sent 'T' → 3.0V output HIGH")
32         key_states["T"] = True
33     elif not keyboard.is_pressed('t') and key_states["T"]:
34         client.sendall(b"T_RELEASED\n")
35         print("Sent 'T_RELEASED' → 3.0V output LOW")
36         key_states["T"] = False
37
38     # Check 'P' key state
39     if keyboard.is_pressed('p') and not key_states["P"]:
40         client.sendall(b"P\n")
41         print("Sent 'P' → 50% PWM output")
42         key_states["P"] = True
43     elif not keyboard.is_pressed('p') and key_states["P"]:
44         client.sendall(b"P_RELEASED\n")
45         print("Sent 'P_RELEASED' → PWM output OFF")
46         key_states["P"] = False
47
48     # Optional: Reduce CPU usage
49     time.sleep(0.1)

```

Figure 9: Python Wifi linkage and keyboard polling

```

1  #include <WiFi.h>
2  #include <Arduino.h>
3
4  const char* ssid = "Daniel_laptop";
5  const char* password = "esp32_445";
6
7  WiFiServer server(8080);
8
9  const int wPin = 13; // GPIO13 for W input (3.3V DC)
10 const int tPin = 12; // GPIO12 for T input (3.0V DC)
11 const int pwmPin = 14; // GPIO14 for PWM output
12
13 void setup() {
14     Serial.begin(115200);
15     WiFi.begin(ssid, password);
16     Serial.println("\nConnecting to Wi-Fi..");
17
18     while (WiFi.status() != WL_CONNECTED) {
19         Serial.print(".");
20         delay(100);
21     }
22
23     Serial.println("\nConnected to Wi-Fi");
24     Serial.print("ESP32 IP Address: ");
25     Serial.println(WiFi.localIP());
26
27     server.begin();
28     Serial.println("TCP server started on port 8080");
29
30     // Configure GPIO outputs
31     pinMode(wPin, OUTPUT);
32     pinMode(tPin, OUTPUT);
33     pinMode(pwmPin, OUTPUT);
34
35     digitalWrite(wPin, LOW); // Start W at 0V
36     digitalWrite(tPin, LOW); // Start T at 0V
37     analogWrite(pwmPin, 0); // Start PWM at 0V
38 }
39

```

```

40 void loop() {
41     WiFiClient client = server.available();
42
43     if (client) {
44         Serial.println("[+] Client connected");
45
46         while (client.connected()) {
47             if (client.available()) {
48                 String receivedData = client.readStringUntil('\n');
49                 receivedData.trim();
50
51                 Serial.print("[*] Received: ");
52                 Serial.println(receivedData);
53
54                 if (receivedData == "W") {
55                     digitalWrite(wPin, HIGH); // Set W pin to 3.3V
56                     client.println("W key detected, output set to 3.3V DC");
57                     Serial.println("[+] W Output is now HIGH");
58                 }
59                 else if (receivedData == "W_RELEASED") {
60                     digitalWrite(wPin, LOW); // Set W pin to 0V
61                     client.println("W key released, output set to 0V");
62                     Serial.println("[-] W Output is now LOW");
63                 }
64                 else if (receivedData == "T") {
65                     digitalWrite(tPin, HIGH); // Set T pin to 3.0V
66                     client.println("T key detected, output set to 3.0V DC");
67                     Serial.println("[+] T Output is now HIGH");
68                 }
69                 else if (receivedData == "T_RELEASED") {
70                     digitalWrite(tPin, LOW); // Set T pin to 0V
71                     client.println("T key released, output set to 0V");
72                     Serial.println("[-] T Output is now LOW");
73                 }
74                 else if (receivedData == "P") {
75                     analogWrite(pwmPin, 194); // 50% duty cycle (128 out of 255)
76                     client.println("P key detected, PWM set to 50% duty cycle on GPIO 14");
77                     Serial.println("[+] PWM running at 50% duty cycle");
78                 }
79             }
80         }
81     }
82 }

```

```

79     else if (receivedData == "P_RELEASED") {
80         analogWrite(pwmPin, 64); // Turn off PWM
81         client.println("P key released, PWM turned off");
82         Serial.println("[-] PWM stopped");
83     }
84     else {
85         client.println("Invalid command.");
86         Serial.println("[!] Unknown command received");
87     }
88 }
89 }
90 client.stop();
91 Serial.println("[-] Client disconnected");
92 }
93 }

```

Figure 10: Arduino Wifi connection and pin assignment

Requirements	Verification
The ESP32 must be able to connect with a laptop over wifi to receive signals.	We will know if this worked by being able to check the voltages coming out of the pins using the oscilloscope. If when we set a signal to high and we see over 3V then we know that it is working correctly.
The ESP32 must not reset or reboot when performing various tasks at once.	In order to see this we will run the robot at full power with wifi, all 3 motors, and data tracking for 1 minute and ensure the esp32 does not force a restart or turn off.
ESP32 must receive signals from motor drivers and send those to our PC.	We will run the motors at a certain RPM we already know like 1000rpm and then ensure that the ESP32 receives the correct rpm reading from the motor drivers and that this is sent to us.
The PC must be able to take input from the controller efficiently and effectively with under 0.5s of lag.	The way we will know if this works is if the controller inputs first register in the PC and then check if there is movement at the motors afterward. This will also be timed with a stopwatch to ensure the final latency between the controller and the car is low.
The ESP32 input and output voltages must not spike when under heavy load.	An oscilloscope will be used when the esp32 is sending and receiving signals from the PC and motor drivers at the same time. The oscilloscope will check the voltages at each pin and ensure we are around the 3.3V since that is what we expect.

- Weapon:** The 2 weapons will be the front-shaped wedge on the front of the car that forces opponents atop and then a push rod on top launches them off. The pole is moved using a motor and gear mechanism to convert the rotational force to upward motion allowing for a strong push force.

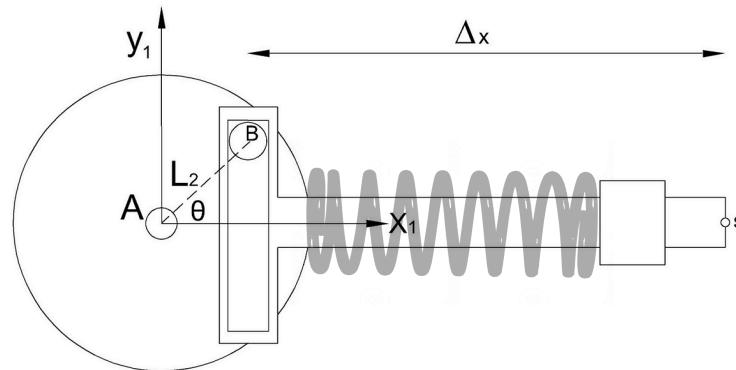
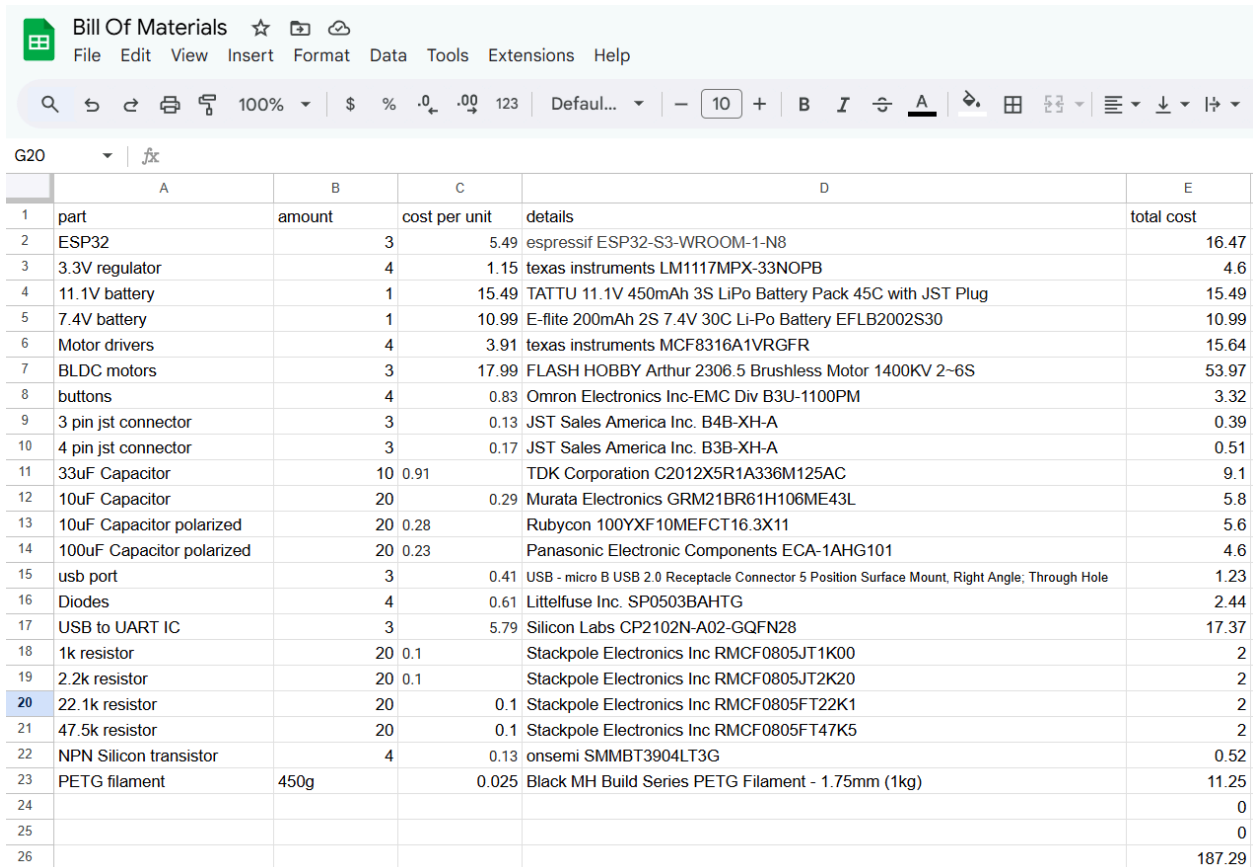


Figure 11: Weapon Diagram

Requirements	Verification
The motor must turn on and off on command after receiving input from ESP32 in under 0.5 seconds.	Using the ESP32 we will send signals at various times and time how long it takes the weapon to stop.
The weapon must sit flush with the top of the car to be ready for an attack. Must be under 0.5cm sticking out.	The motor will be spun and measurements of how flush the weapon is at the end will be taken.
The weapon must not tear itself apart when spinning at a speed of 100 rpm.	The motor will be spun at that rate for one minute and if the weapon stays intact then it will be considered stable.
When the weapon motor spins the motor driver and battery must not become excessively hot.	This will be measured using a thermometer to ensure that no component is overheating. Any component overheating will be given special attention.
Ensure that the voltage powering the motor is not too high so as to not damage the motor.	A voltmeter will be used to check the voltage when full throttle is engaged on the motor.

# Cost and Schedule

## Cost Analysis:



	A	B	C	D	E
1	part	amount	cost per unit	details	total cost
2	ESP32	3	5.49	espressif ESP32-S3-WROOM-1-N8	16.47
3	3.3V regulator	4	1.15	texas instruments LM1117MPX-33NOPB	4.6
4	11.1V battery	1	15.49	TATTU 11.1V 450mAh 3S LiPo Battery Pack 45C with JST Plug	15.49
5	7.4V battery	1	10.99	E-flite 200mAh 2S 7.4V 30C Li-Po Battery EFLB2002S30	10.99
6	Motor drivers	4	3.91	texas instruments MCF8316A1VRGFR	15.64
7	BLDC motors	3	17.99	FLASH HOBBY Arthur 2306.5 Brushless Motor 1400KV 2-6S	53.97
8	buttons	4	0.83	Omron Electronics Inc-EMC Div B3U-1100PM	3.32
9	3 pin jst connector	3	0.13	JST Sales America Inc. B4B-XH-A	0.39
10	4 pin jst connector	3	0.17	JST Sales America Inc. B3B-XH-A	0.51
11	33uF Capacitor	10	0.91	TDK Corporation C2012X5R1A336M125AC	9.1
12	10uF Capacitor	20	0.29	Murata Electronics GRM21BR61H106ME43L	5.8
13	10uF Capacitor polarized	20	0.28	Rubycon 100YXF10MEFCT16.3X11	5.6
14	100uF Capacitor polarized	20	0.23	Panasonic Electronic Components ECA-1AHG101	4.6
15	usb port	3	0.41	USB - micro B USB 2.0 Receptacle Connector 5 Position Surface Mount, Right Angle; Through Hole	1.23
16	Diodes	4	0.61	Littelfuse Inc. SP0503BAHTG	2.44
17	USB to UART IC	3	5.79	Silicon Labs CP2102N-A02-GQFN28	17.37
18	1k resistor	20	0.1	Stackpole Electronics Inc RMCF0805JT1K00	2
19	2.2k resistor	20	0.1	Stackpole Electronics Inc RMCF0805JT2K20	2
20	22.1k resistor	20	0.1	Stackpole Electronics Inc RMCF0805FT22K1	2
21	47.5k resistor	20	0.1	Stackpole Electronics Inc RMCF0805FT47K5	2
22	NPN Silicon transistor	4	0.13	onsemi SMMBT3904LT3G	0.52
23	PETG filament	450g	0.025	Black MH Build Series PETG Filament - 1.75mm (1kg)	11.25
24					0
25					0
26					187.29

Figure 12: Bill of Materials

Our project will require the use of a 3d printer to create the chassis and potentially the wheels. This would be an operational cost on top of the filament cost. However, the IDEA lab has a set base fee of \$4.00 with a material cost of \$0.10 per gram of filament. This will bring our total cost of printing to \$49.00 for a 450g weight chassis. On average a UIUC electrical engineer in Chicago makes around \$55.00 an hour. Based on our production time estimate of around 2-3 hours for soldering as well as 4-5 hours of R&D. our total for a finished product would be \$682.29 with labor, materials, and operational costs.



## Schedule:

Week	Objective	Personel
3/3	Finish Breadboard	All
3/10	First instance of PCB/buy all parts	All
3/17	Spring break	
3/24	Solder pcb and start testing	All
3/31	Second revision of pcb	All
4/7	Solder pcb/third revision of pcb 3d printing	All
4/14	3d printing and final touches	All

## Ethics and Safety

**Ethics** - Since our proposal no significant new ethical considerations need to be considered. We still plan to adhere to the IEEE code of ethics and ensure our project does not pose a fire or chemical threat to the health of the environment or our community. Below is our statement about following ethical guidelines from our proposal. We plan to stay committed to what we said.

In designing and building a battlebot there are several ethical concerns that must be addressed. In accordance with section 1.1 of the IEEE Code of Ethics, we will ensure that the safety, health, and well-being of us and our community are paramount. We will disclose any and all dangers to the public and the environment. Fortunately, our battlebot will be designed to work indoors so we aspire to have minimum impact on the

environment. However, our battlebot and other battlebots are designed to operate in the presence of people making safety a high priority. In addition to section 1.1, we will adhere to section 1.5 of the code of ethics by listening to technical feedback and adjusting our design accordingly. We will be honest about our technical limitations. For instance, in designing a battlebot we will need a motor drive for the wheels with which we do not have substantial design experience. We will research the proper way to implement the system and do so correctly while at the same time maintaining a level of honesty with each other and our professors and teaching assistants. We will not make unsubstantiated claims and ensure design decisions involve grounded analysis whether mathematical or in simulation.

**Safety** - Since our proposal not many new significant safety hazards have emerged regarding the development of our project. However, one important thing that we have taken note of is the temperature of our components such as our ESP32 chip and our motor driver IC. We expected these components to increase in temperature during operation, but we need to monitor the current drawn from the motors and furthermore the temperature of the motor driver IC to ensure we do not burn ourselves or damage our components. Otherwise, below is what we stated regarding safety in our proposal and we plan to stay committed to the safety of ourselves and others.

A small, highly mobile battlebot powered by a Li-Po battery can pose multiple dangers. Misuse or mishandling of Li-Po batteries can result in rapid discharge which can lead to fires, smoking, and thus toxic smoke inhalation. In accordance with Code 1 of the IEEE Code of Ethics, we will ensure to follow common safety practices to ensure that we handle the Li-Po batteries with care to avoid fire, or short-circuiting which could endanger us or the community. For one, we will always charge the Li-Po batteries with the appropriate charger away from flammable materials. When powering the BattleBot we will place the robot in a nonflammable container. This will ensure that a fire is contained as well as that the robot is contained in the case of a short circuit or if the robot operates unprompted. While the robot is charging or in a state of operation we will ensure that one of us is always nearby to neutralize any potential dangerous situation. Testing the robot in an enclosed structure will also allow for safe testing of our motorized weapon. A motorized weapon operating at a high rpm can pose a serious danger if a piece flies off during operation. To ensure this does not occur we will first test our weapon within an enclosure which could protect against the danger of a flying projectile. For our project we do not anticipate the need for potentially dangerous chemicals or biological materials so fortunately our safety concerns rely mostly on proper mechanical and electrical safety.

# References

[1] Espressif Systems, ESP32 Series Datasheet, Espressif Systems, 2023. [Online]. Available: [https://www.espressif.com/sites/default/files/documentation/esp32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf). [Accessed: 10-Feb-2025].

[2] IEEE, "IEEE SA - IEEE Code of Ethics," IEEE, 2024. [Online]. Available: <https://www.ieee.org/about/corporate/governance/p7-8.html>. [Accessed: 8-Feb-2025].

[3] Repeat Robotics, "Repeat Tangent Drive Motors," Repeat Robotics, 2025. [Online]. Available: <https://repeat-robotics.com/buy/repeat-tangent-drive-motors/>. [Accessed: 12-Feb-2025].

[4] Texas Instruments, MCF8316D 4.5-V to 60-V sensorless FOC brushless motor driver with integrated MOSFETs, Texas Instruments, Dec. 2023. [Online]. Available: <https://www.ti.com/lit/ds/symlink/mcf8316d.pdf>. [Accessed: 9-Feb-2025].