3D Printed Antweight Battlebot

By
Don Lazatin (dlazat2)
Brian Pau (bnpau2)
Shashank Sangireddy (ssangi2)

Final Report for ECE 445, Senior Design, Spring 2025

TA: Jason Jung

7 May 2025

Project No. 10

Abstract

This final report details the design, testing, and implementation of a 3D-printed antweight battlebot. The battlebot operates using a two-wheel drivetrain with a servo-activated wedge as the weapon. It is controlled from a PC using WiFi and has a fully 3D-printed chassis. The battlebot also includes two kill-switches, a physical toggle switch and a software-based shutdown, to ensure safe operation. The chassis is designed to be horizontally symmetrical to allow the robot to operate even if it is flipped over. The final implementation of the battlebot adheres to all competition rules and serves as a competitive entrant in the arena.

Table of Contents

1. Introduction

1.1 Problem

Combat robotics competitions have experienced a significant increase in popularity in recent years. These competitions offer participants the opportunity to develop and practice engineering, design, and programming skills in an all-hands-on, competitive environment. Professor Gruev's Antweight Battlebot Competition presents a unique challenge where each team must construct a fully functional battlebot with strict design limits and constraints, such as a 2 lb weight limit, control over Bluetooth or WiFi, and built-in shutdowns, etc. [1].

The goal of this project is to design and implement an Antweight Battlebot that is eligible and capable of competing against other Battlebots in a competition arena environment. More specifically, the Battlebot must be able to disrupt the functionality of competing Battlebots with a fighting tool while ensuring its own functionality.

1.2 Solution

The overall solution to this task is a combat-ready battlebot, shown in Figure 18 in Appendix D, equipped with an opponent-destabilizing wedge flipper weapon, a durable yet lightweight 3D-printed chassis body, and a wireless control system. Our battlebot is powered by an ESP32-S3-WROOM-1 microcontroller with built-in WiFi capabilities, allowing seamless remote operation and communication. The battlebot's movement system uses two N20 motors to drive the wheels with DRV8231 motor controllers, which have integrated h-bridges, controlling the wheel direction.

A wedge serves as the primary combat mechanism, actuated by a Micro HDD Servo. This weapon is designed to lift and destabilize the opponent robots by utilizing its mass and motor-driven activation. To ensure the bot remains functional regardless of its axis orientation, the chassis is symmetrically structured about its horizontal axis. This allows the robot to remain functional even if it is flipped. A rechargeable 3S LiPo battery provides power to all subsystems of the design. Voltage regulators are implemented to step down the battery voltage to the different subsystems as necessary. The battlebot also contains multiple kill-switch functions, including both a physical switch as well as a software implementation, to comply with competition safety requirements and constraints. By integrating mechanical engineering principles with embedded systems and wireless communication, the battlebot was a competitive entry in Professor Gruev's Antweight Battlebot Competition.

1.3 High-Level Requirements List

- The battlebot must not exceed a maximum acceleration of 5 m/s^2 to ensure controlled maneuverability across the competition arena.
- The wedge weapon system must be capable of lifting and displacing objects of up to 2 lbs and must return to its original position within 2.5 seconds after activation.
- The battlebot's WiFi connection must maintain reliability with the control PC at a range of at least 15 feet, with a command response time of less than 200 milliseconds in order to ensure real-time and precise control.

1.4 Block Diagram

The robot consists of 5 total subsystems: power, control, drivetrain, weapon, and chassis. Figure 1 displays the interconnections between the different subsystems. The Chassis Subsystem is not depicted in the block diagram as it is a mechanical component and does not directly interact with any of the other subsystems electrically.

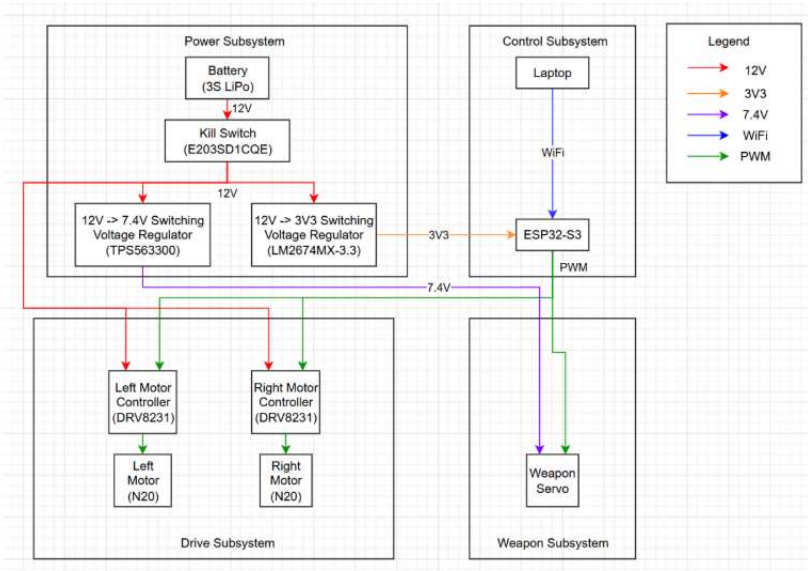


Figure 1: High-Level Block Diagram for Robot

Minor changes were made at the block-level throughout the semester. Most notably, a switch was made from Bluetooth to WiFi, affecting the Control Subsystem. Originally, a dedicated motor controller was planned for the weapon servo, but after further research, the group decided to remove it and connect the servo directly to the microcontroller. This led to the addition of the 7.4 V switching voltage regulator to provide power directly to the servo. No major changes were made with regards to the number or purpose of subsystems.

2. Design

2.2 Subsystem Overview and Requirements

2.2.1 Power Subsystem

The Power Subsystem is responsible for supplying power to the entire robot. This subsystem consists of a 3S 650 mAh 75C LiPo battery, a LM2674MX-3.3 switching voltage regulator, a TPS563300 switching voltage regulator, and an E203SD1CQE toggle kill-switch. The LM2674MX-3.3 is used to step down the battery voltage from 12 V to 3V3 at 500 mA, which is used by the microcontroller. The TPS563300 is used to step down the battery voltage from 12 V to 7.4 V at a maximum of 3 A output current, which is used by the weapon's servo. This subsystem also includes a physical toggle kill-switch between the battery and the rest of the robot. The 3S LiPo battery is nominally 11.1 V, but can reach a maximum of 12.6 V when fully charged. Circuit schematics for the Power Subsystem can be seen in Figure 3 in Appendix A.

The ESP32-S3-WROOM-1 microcontroller draws 500 mA at 3V3 [2]. The N20 motors for the drivetrain individually draw a maximum of 1600 mA, for a combined 3200 mA [3]. Lastly, our Micro HDD Servo motor for the weapon draws a maximum of 3000 mA [4]. Combined, the absolute maximum current draw of our robot is 6.7 A as shown in Equation (1). Thus, our battery needed to be able to accommodate this for at least 2 minutes, even though the expected current draw was much less.

$$I_{max} = I_{ESP} + I_{Drive} + I_{Weapon} = 500mA + 3200\,mA + 3000\,mA = 6700\,mA \quad (1)$$

$$Required\ Capacity = (6.7A) * (1/30\ hours) = 67/300\ Ah \approx 223\,mAh \quad (2)$$

$$Minimum\ C\ Rating = (6.7\ A) / (67/300\ Ah) = 30C \quad (3)$$

Equations (2) and (3) show the minimum required capacity and C rating to support a 6.7 A continuous current draw for two minutes. The C rating describes the rate at which the battery can discharge its current. A higher C rating means that the battery can discharge at a quicker rate, safely supplying larger amounts of current than a battery with a lower C rating. It should be noted that a higher C does not necessarily mean the battery has a shorter runtime. This led to using a 3S LiPo battery with a 650 mAh minimum energy capacity and a discharge rate of 75C, with a maximum burst discharge rate of 150C [5]. This means that the battery is able to provide a maximum continuous current of 48.75 A with a maximum burst current of 97.5 A. Additionally, Equation (4) shows the minimum runtime for the robot, which significantly exceeds the competition time limit.

$$Runtime \;=\; (0.650\ Ah)\ /\ (6.7A) \;\approx\; .097\ hrs \;\approx\; 5.8\ minutes \qquad (4)$$

Both regulators being used, the LM2674MX-3.3 and the TPS563300, are switching regulators. The team decided to use switching regulators for the improved heat dissipation. The LM2674MX-3.3 has an efficiency of up to 96% and does not require extra heat sinking [6]. The TPS563300 also has an efficiency of over 90% in our operating range and also has built-in overvoltage, undervoltage, and overcurrent protection [7].

Since the TPS563300 is an adjustable voltage regulator, the output voltage is set by a voltage divider connected to the feedback pin. Equation (5) was provided by the manufacturer to identify a suitable R_{FBT} value, with recommended values of V_{REF} = 0.8 V and R_{FBB} = 10 kΩ. The TPS563300 is being used for the servo, which has an operating range of 4.8 to 8.4 V, but gives measurements for current and torque for a 7.4 V input. Thus, to achieve a 7.4 V output from the TPS563300, we used an 82.5kΩ resistor as R_{FBT}.

$$R_{FBT} \;=\; \frac{V_{OUT} - V_{REF}}{V_{REF}} \times\; R_{FBB} \qquad (5)$$

An E203SD1CQE toggle kill-switch is used between the battery input and the rest of the electronics. This kill-switch was chosen for its high current rating (7.5 A) and for its easily accessible form factor. The kill-switch can be turned on or off to provide a manual way to shut off the robot.

The battery connects to the PCB through an XT-30 connector. Since the connector has a unique form factor, it ensures that the battery cannot be plugged in incorrectly, preventing shorts or other safety hazards.

2.2.2 Drivetrain Subsystem

The Drivetrain Subsystem is responsible for the battlebot's movement and maneuverability. It utilizes two N20 micro motors, each rated for 12V DC operation, that provide a no-load speed of approximately 460 RPM and a current draw between 100 mA to 1600 mA [3]. The DRV8231 motor driver chip utilizes an H-bridge mechanism and operates within an input voltage range of 4.5V to 33V and supplies a continuous current of up to 3.7A per motor channel. The driver can receive control signals between 0 and 5.5V from the microcontroller and manages the speed and direction of the motors, which enables precise maneuverability [8]. Circuit schematics for the Drivetrain Subsystem can be seen in Figure 4 in Appendix A.

The N20 micro motors were chosen by the team for a few reasons. Researching common antweight battlebot motors pointed towards using N20 motors. Furthermore, the specific N20 motors used have a listed weight of 9.5 g [3], which is incredibly light given the 2 lb weight limit, allowing for more flexibility with the chassis design. Early plans were to use a 4 inch diameter wheel, so after using Equation (6) to estimate the maximum possible speed with the 460 RPM motor, the team believed it to be the best balance between speed and control of the robot's drive.

$$Speed = \frac{(2 * \pi * r * RPM)}{60} = 2.447\ m/s \quad (6)$$

The DRV8231 motor controllers were selected due to its compatible input voltage ranges for both power and logic, as well as its high current output. Since the N20 micro motors advertised a maximum pull current of 1.6 A, the team decided to use the DRV8231 as its 3.7 A output current limit would easily support any demand from the motor [8]. It was also compatible with the 12 V power coming from the battery and the 3V3 logic signals coming from the microcontroller.

The DRV8231 receives PWM signals from the microcontroller to dictate the movement of each motor. During testing, the team found that the speed at maximum settings was manageable, so the duty cycle for the PWM signals are 100% which can also be interpreted as simple "high" and "low" outputs.

2.2.3 Control Subsystem

The Control Subsystem is responsible for processing user command inputs and translating them into movement commands for the battlebot. This subsystem uses the ESP32-S3-WROOM-1 microcontroller. This controller has a WiFi module which is used to wirelessly communicate with the laptop. We used WiFi as a communication protocol to ensure a reliable connection between the laptop and the battlebot. The ESP32-S3-WROOM-1 processes these commands and sends appropriate PWM control signals to the drivetrain motor controllers and the weapon servo. The Control Subsystem also includes a software-based kill-switch: the user can manually disconnect from the WiFi communication protocol by pressing 'x' on the external controller, which will terminate the connection to the robot, causing it to stop operating. Also, if the hosting WiFi device is disconnected for any reason, the microcontroller will acknowledge this event and automatically halt ongoing operations. Circuit schematics for the Control Subsystem can be seen in Figure 5 in Appendix A.

The team chose to use the ESP32-S3 microcontroller for its larger flash memory, built-in antenna, flexibility between Bluetooth and WiFi, and popularity in antweight battlebots. The S3 also has two motor control PWM (MCPWM) peripherals with 3 PWM operators each, for a total of 6 signal outputs [2]. This made the S3 appealing as the robot is controlled entirely off of

PWM signals to the drivetrain motors and weapon servo. Extra circuitry was built into the PCB to allow the ESP32 to be flashed via an external USB-to-UART chip.

A concern facing the ESP32 and the overall Control Subsystem was inrush current, which is large spikes in current draws when the motors start or stop. This was concerning as it could cause drops in voltages that could negatively affect our electronics, including resetting our microcontroller. To best mitigate this, a large capacitance of 470 μF was included near the voltage input pin to the ESP32 microcontroller, which is seen in Figure 6 in Appendix A. Additionally, the DRV8231 motor controller ICs being used for the drivetrain motors have built-in current regulation features to help mitigate some of the effects of inrush current as well.

The team initially attempted to control the ESP32-based battle bot using Bluetooth using the NimBLE library. However, they encountered issues that impacted reliable communication. Specifically, the ESP32 device frequently failed to properly advertise itself. This made it very difficult for external controllers to detect and connect to it. Additionally, the process of correctly configuring and handling the required Bluetooth services and characteristic UUIDs proved to be complex and error-prone. These overall challenges made Bluetooth an unreliable option for real-time control. As a result, the team decided to switch to Wi-Fi, which offered a more stable and straightforward method of communication using HTTP requests. This transition allowed for quicker development, more reliable connections, and simplified the external controller's implementation.

The team implemented a web-controlled robotic system using the ESP32 microcontroller and a Python-based keyboard interface. The ESP32 is programmed to connect to a specific Wi-Fi network and host a simple HTTP server that listens for specific POST requests to control a battle bot's motors and weapon system. It manages two motors connected via GPIO pins to drive the robot forward, backward, left, and right, and includes a servo-controlled weapon mechanism that can be activated in normal or inverted modes. Each movement or weapon command is mapped to an HTTP endpoint (e.g., /forward, /weapon), and the ESP32 executes the appropriate GPIO outputs to drive the motors or actuate the weapon servo accordingly.

On the controller side, a Python script uses the requests library to send HTTP POST commands to the ESP32 based on keyboard input detected using the pynput library. Users control the robot in real-time using W, A, S, D, X, space, and shift keys for directional movement, stopping, weapon activation, and mode inversion, respectively. The script uses a background thread to manage command sending with a short throttle time to prevent flooding the ESP32 with requests. This architecture enables low-latency, keyboard-based remote control over Wi-Fi without the need for specialized hardware on the client side.

2.2.4 Weapon Subsystem

The team implemented a wedge flipper actuated by a servo motor mounted at the front of the robot. Through experimentation, we determined that rotating the servo to 0 degrees raises the flipper to its maximum height, effectively flipping opponents, while 90 degrees serves as the neutral position where the wedge rests flat. To accommodate the bot's inverted driving mode, we introduced a separate control function that moves the servo to 180 degrees instead of 0. This ensures that the flipper still operates in the intended direction even when the robot is being driven in reverse relative to the user's perspective. The toggle between standard and inverted control modes allows consistent and intuitive weapon performance regardless of the bot's orientation on the battlefield.

The Weapon Subsystem consists of a Micro HDD Servo with a high power to weight ratio and the physical wedge that is part of the robot's body, located directly in front of the chassis. These two parts work in tandem to act as a high-power lever with wide surface area, fit to lift and displace objects of up to 2 pounds. The weapon motor operates within a 4.8 to 8.4 V input voltage range with a current draw of up to 3A. This motor is able to jolt up to 60 degrees within 0.22 seconds with no load. The weapon motor is controlled by PWM signals directly from the microcontroller [4]. Circuit schematics for the Weapon Subsystem can be found in Figure 7 in Appendix A.

During our research, we found that this servo was the most ideal for its power to cost ratio compared to the other options. Several other servos on the market had either a slower turning speed, in degrees per second, or a monetary cost that was too high for its marginal edge in performance.

Originally, the team planned to use a separate motor controller IC to control the servo's rotation, which would have added I^2C to the design. However, after further research, the team learned that the servo could be connected directly to the microcontroller and receive the PWM signals directly without a middleman.

2.2.5 Chassis Subsystem

This subsystem provides the structural base for the battlebot. It houses the PCB, drive motors, weapon servo, and battery. This subsystem is 3D printed and constructed using PLA filament in order to balance durability and weight management. Its symmetrical design along the horizontal axis ensures identical operation even if flipped. Figures 19 and 20 in Appendix D show CAD model representations of the internal structure of the chassis.

Originally, we had planned to 3D print all of the relevant parts with PLA+ as our choice of filament. After reviewing the available resources in the ECE 445 lab room, and the costs of acquiring PLA+ filament, we had come to the conclusion that we would move forward with the PLA filament available to finish construction of our robot designs. This decision was further supported by our research findings, which revealed that the quality standard of PLA+ may be different depending on the manufacturer, and that PLA filament would be adequately sufficient for our purposes.

The drivetrain wheels are protected by the chassis itself, with all wheels being inset into the chassis. The holes in the sides of the body are for the N20 motor shafts to insert through, which then connect to holes in the wheels that are appropriately shaped to tightly fit the N20 motor's D-shaped shafts. For the assembly of the robot, this set of parts was attached by press-fitting and was further secured using superglue (GH1200). The visual reference for the drivetrain wheels can be seen with Figure 22 in Appendix D.

The actuator for the fighting tool, the high power servo, is seated inside the main body near the front. The "arm piece" connecting our wedge weapon to the servo needs to move through the chassis, which necessitated the shown opening at the front of the body, which extends from the top to the bottom of the main body. The visual reference for the wedge weapon and related parts can be seen with Figure 23 in Appendix D.

We initially included two sets of small, free-moving wheels with one set on the top and another on the bottom of the robot. These free-moving wheels would allow the robot to stay level with the ground and provide a little extra stability. Since they were free-spinning, the team believed that it would not detract from the overall maneuverability of the robot. A visual reference for this part can be found with Figure 24 in Appendix D.

Other major design changes from the original version included the use of U-Brackets to hold down the drivetrain motors (shown in Figure 26 in Appendix D), and an approximate one inch decrease for all major dimensions of the robot. This decrease in size was made primarily in order to more easily satisfy the weight limit of the competition. The final significant change from the original design has to do with the front set of wheels. The decrease in the chassis's height would have caused the top and bottom sets of front wheels to conform to mechanically unreliable dimensions, so the decision was made to instead incorporate a single set of front wheels.

During testing, we had discovered that the press-fit assembly of these newer wheels and fasteners was hindering our robot's performance by adding unnecessary friction between the robot's body and the floor. The solution to this challenge was to remove those wheels entirely, which resulted in a significantly improved drivetrain performance.

3. Verifications

3.1 Subsystem Verifications

3.1.1 Power Subsystem

The full R&V table and supporting figures can be found in Appendix B, section B.1. All requirements for the Power Subsystem were successfully met during testing. For these measurements, the benchtop multimeters and power supplies available in the lab were used. Table 1 shows the measured voltages from the robot's power rails.

Table 1: Battlebot Voltage Rail Requirements and Measured Values

Required Voltage	**Measured Voltage**
Battery Voltage: 12 ± 0.6 V	12.549 V
TPS563300 Output (7.4 V): 7.4 ± 1 V	7.450 V
LM2674MX Output (3V3): 3.3 ± 0.3 V	3.287 V

There were some initial struggles with getting the correct output from the TPS563300. The first iteration of the board used a 4.7 µH inductor, which was the minimum inductance recommended by the datasheet, which gave an output of roughly 2 V. Increasing the inductance to 22 µH resulted in a correct output and is the final inductance currently on the board.

To measure the current draw of the robot, a current sense resistor of 10 mΩ was placed in between the battery and the system. Figure 11 in Appendix B, section B.1 shows a multimeter reading of the maximum voltage seen across the current sense resistor when running the robot at full load. Knowing the resistance and voltage across the resistor, Ohm's law was used, resulting in a 0.2749 A current from the battery. Additionally, when connected to the power supply, the highest peak current seen at maximum load was 0.836 A, also significantly below the 6.7 A requirement.

To test that the battery lasted for 2 minutes, the team ran the robot at full load (continuous use of drive motors and constant servo actuation) to simulate a competition environment. Using a timer, the team verified that the robot maintained functionality for over two minutes. This is further supported by the fact that our robot remained operational for the entire match duration in Prof. Gruev's battlebot competition.

3.1.2 Drivetrain Subsystem

The full R&V table can be found in Appendix B, section B.2. During testing, our RPM requirement was not met by our robot. During testing, the team measured the linear speed of the robot by taking position measurements against a measuring tape while noting the time it took to reach said positions. Figure 2 shows two such position measurements which correlate to a linear speed of roughly 0.782 m/s. Knowing the wheel circumference to be about 0.065 m, Equation (7) was used to find an RPM of 228 RPM. This failure can be mainly attributed to mechanical issues. The motor's mounting to the chassis and the wheel's mounting to the motor were either press fit or involved super glue, causing issues like wobbling, rubbing up against the chassis, and slipping between the axis and wheel. All of these factors contributed to a slower than expected motor RPM in practice.

$$RPM = \frac{Linear\ Speed * 60}{Wheel\ Circumference} = 228\ RPM \tag{7}$$

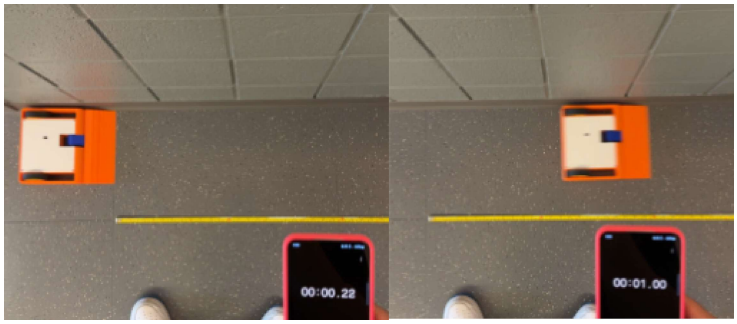


Figure 2: Position Measurements of Robot (0 ft at 0.22 s) and (2 ft at 1 s)

The second requirement regarding acceleration passed, but the requirement could have been written more specifically. To measure our robot's acceleration, we positioned ourselves 10 meters away and used a timer to record how long it would take the robot to cross a 10 meter distance. Following the equations outlined in Appendix B, section B.2, Table 3, since the robot failed to reach the 10 meter mark in under two seconds, we considered it a success. However, since our robot's drive is always at max throttle, this is measuring continuous acceleration. Since the listed requirement does not specify what kind of acceleration, it could also be interpreted as instantaneous acceleration, which we were unable to verify. To measure the instantaneous acceleration, an IMU or accelerometer would have needed to be included on the PCB. Additionally, the team could have instead specified that it is measuring continuous acceleration in the original requirement.

3.1.3 Control Subsystem

The full R&V table and supporting figures can be found in Appendix B, section B.3. All requirements for the Control Subsystem were successfully met during testing. To measure the operating range, we positioned the control PC and the robot approximately 20 feet away from each other. It was marked as successful as the robot still received and executed commands despite the range. Additionally, the robot was observed to remain functional when running it down a hallway at ECEB, away from the control PC. Between these two tests, the team feels confident that the operating range is over 15 feet.

To measure latency for the motors and the software killswitch, an Agilent Technologies DSO-X 3034A oscilloscope was used to measure the output from the microcontroller and the output signal to a drive motor. The oscilloscope was set to trigger at the rising edge of the signals, which is captured in Figure 14 in Appendix B, section B.3. The measured latency between the microcontroller and the motor output was 43 μs, below the required 200 ms.

3.1.4 Weapon Subsystem

The full R&V table can be found in Appendix B, section B.4. The second two requirements were successfully met during testing. The wedge never sustained significant damage during activation and could be used numerous times in quick succession without mechanical failure. Additionally, after testing using a video recording and timer, the team observed that the weapon could be activated three times in under four seconds, clearly surpassing the required 2.5 second reset time.

However, the robot failed to lift the required minimum of 2 lbs, instead reaching a maximum of roughly 0.5 lbs. The cause of failure was a lack of friction between the Micro HDD Servo's shaft extension and the Wedge Arm. This assembly can be seen more clearly in Figure 21 in Appendix D. In this image, the shaft extension is the white long plastic piece that inserts into the blue Wedge Arm. There is enough friction between these two parts for the Weapon subsystem to operate as one part, but only with no load applied to the Wedge Blade. With no load applied, the Weapon Subsystem operates at a native stall torque of 0.019 kg/cm [4].

However, after applying a load of exactly two pounds, we observed that the press-fit connection between the aforementioned parts had broken apart and slid around each other. In other words, the subsystem no longer operated as a single part, which led to a significantly reduced stall torque of 0.219 kg/cm, which translates to our wedge weapon only being capable of lifting objects up to about 220 grams, or half a pound. This max load was estimated by testing the Weapon Subsystem on lighter objects and gradually decreasing the load on the wedge until the subsystem functioned as expected.

3.1.5 Chassis Subsystem

The full R&V table can be found in Appendix B, section B.5. All requirements were successfully met during testing. The chassis effectively protected all internal electronics during testing and competition and no wires or components were exposed. Additionally, the robot was able to stay functional regardless of its orientation, which proved to be useful during competition.

Just before assembling the robot together, each individual part was weighed using a digital scale, set to the grams setting. Figure 27 in Appendix D shows the total weight of an older, but strictly heavier, iteration of our robot design. Therefore, our robot cannot weigh any more than 518 grams, or 1.12 pounds, and safely clears the 2 pound weight limit.

Outside of this subsystem's requirements, the assembly process of the robot led to several instances of mechanical instability which led to failures in some of the other subsystems' high level requirements.

Much of this instability stems from a combination of the minute precision flaws involved with 3D printing and the press-fit assembly strategy chosen for the project design. Quite a few of the smaller details in the 3D printed parts relied on sub-millimeter accuracy in order for proper and complete press-fitting. Some examples for this include the D-Shaft slots in the drivetrain wheels, the circular hole on the wedge arm for the servo shaft to insert into, the holes in the chassis designated for the now-removed free moving wheels and fasteners, and the free-moving wheels themselves. All four of these examples are individual parts that were meant to either be press-fit together or expected to function with a sub-millimeter level of clearance.

The combined result of these assemblies was an ultimately wobbly, improperly secured construction that significantly affected the robot's performance. Applying this method of assembly for most of the critical components showed to not be worth the extra margin of error for total weight.

After initial testing, we resorted to using superglue (GH1200) as a stopgap to more effectively secure the internal components. The robot's functionality was partially recovered thanks to this adjustment, but the robot's assembly was now mostly permanent. While this helped secure the servo and drivetrain motors temporarily, it was still a weak connection overall compared to a designed solution, such as using screws.

4. Cost and Schedule

4.1 Component Costs

Component	Part #	Manufacturer	Quantity	Cost	Link
ESP32-S3-WROOM	ESP32-S3-WROOM-1-N4	Espressif	1	$5.06	Link
E203SD1CQE Toggle Switch	E203SD1CQE	C&K	1	$15.99	Link
LM2674-3.3 (Voltage Reg)	LM2674MX-3.3/NOPB	Texas Instruments	2	$2.58	Link
3S 650mAh 75C LiPo battery	TA-75C-650-3S1P-XT30	TATTU	1	$13.80	Link
SK12 Capture Diode	SK12	Diotec Semiconductor	1	$0.19	Link
Micro HDD Servo	EL-3760	ez-robot	1	$22.49	Link
TPS563300DRLR Servo Regulator	TPS563300DRLR	Texas Instruments	1	$0.91	Link
N20 Motors	638126	ServoCity	2	$25.98	Link
DRV8231 (Motor Driver Chip)	DRV8231DDAR	Texas Instruments	2	$2.58	Link
ESP32-S3-DEVKITM-1-N8	ESP32-S3-DEVKITM-1-N8	Espressif	1	$13.30	Link
SOIC-8 TO DIP-8 SMT ADAPTER	PA0001C	Chip Quik Inc.	1	$6.49	Link
PCB Passives	N/A	DigiKey	N/A	$47.92	Link
3D Printing	N/A	N/A	N/A	$0	N/A

SUM OF ALL COMPONENTS: $157.39

4.2 Cost Analysis

We estimated that a fair hourly rate is $45 per hour due to the rigorous and tedious aspects that this project involved. Based on this information, we have calculated individual salary and group salary below.

> *Hours Per Week **x** Pay Per Hour **x** Total Weeks (Individual Salary)*
> 10 hours * $45 * 8 Weeks = **$3,600**
>
> ***x** Number of Group Members*
> $3,600 * 3 Group Members = **$10,800**

Grand Total (Sum of Costs): $10,800 + $157.39 = $10,957.39

4.3 Schedule

Week	Tasks
Week of March 10th, 2025	• Order Remainder of Components Needed **(Don)** • Finalize Codebase for Breadboard Demonstration **(Don)** • Finalize Breadboard Demonstration **(ALL)** • Finalize 1st Schematic & PCB Design **(Brian)** • Submit 1st Finalized PCB Order **(Brian)** • Get Familiar with 3D-Printing Resources **(Shashank)**
Week of March 17th, 2025	• Request for More Components Needed **(Brian & Shashank)** • Order Final Components Needed **(Don)** • Finalize MCU to Laptop Connection WiFi connection **(Don)** • Begin 3D-Printing Design **(Shashank)**
Week of March 24th, 2025	• Receive 1st PCB **(ALL)** • Solder and Assemble 1st PCB (**Brian**) • Begin troubleshooting 1st PCB **(Brian)** • Make modifications for 2nd PCB Design **(Brian)** • Begin Implementation of MCU to External Controller Connection **(Don)** • Continue 3D-Printing Design **(Shashank)**
Week of March 31st, 2025	• Testing code on 1st PCB **(Brian & Don)** • Submit 2nd Finalized PCB Order **(Brian)** • Finalize Battlebot Codebase **(Don)** • Finalize Chassis Design **(Shashank)**
Week of April 7th, 2025	• Receive 2nd PCB and test custom footprint adjustments **(Brian)** • Continued testing and troubleshooting using 1st PCB (**Brian & Don**) • Begin Printing Chassis **(Shashank)** • Begin Formation of Battlebot Components **(ALL)**

Week of April 14th, 2025	• Test Finalized Code on 2nd PCB (**Brian & Don**) • Begin Adjustments and Printing Non-Chassis Parts **(Shashank)** • Prepare for Mock Demo (**ALL)**
Week of April 21st, 2025	• Solder and Assemble 3rd PCB (**Brian**) • Test and Verify Functionality of 3rd PCB (**Brian**) • Test Finalized Code on 3rd PCB **(Brian & Don)** • Finalize Adjustments and Finish Assembly of Robot **(Shashank)** • Mock Demo (**ALL**) • Prepare for Final Demo (**ALL)**
Week of April 28th, 2025	• Final Demo (**ALL**) • Reworking chassis and final adjustments to battlebot (**ALL**) • Compete in Battlebot Competition (**ALL**)
Week of May 5th, 2025	• Prepare for Final Presentation (**ALL**) • Write Final Paper (**ALL**)

5. Conclusion

Overall, the major accomplishments for this project were making a working battlebot, including a fully working PCB, whose design can be found in Figures 8 and 9 in Appendix A, and codebase, to compete in Prof. Gruev's battlebot competition. While not every requirement was met, developing a functional robot that could operate either flipped or upright within the short time frame of the semester is an accomplishment in and of itself.

One of the biggest shortcomings and uncertainties regarding the design of the robot was with the mechanical design and assembly. The chassis assembly contained a lot of missing parts or parts that failed to translate to the real world which led to unsecure connections, loose fits, and a generally unstable robot. These mechanical shortcomings led to a lot of errors with the final performance of the robot. Given more time, the team would like to revisit the CAD model for the design and make it more robust by adding more screws and fasteners to secure the drive motors and weapon servo, as well as working on more designs for how to attach the wheels and wedge to their respective motors.

Other future improvements that could be made would be increasing the complexity of the PCB by including sensors such as IMU to help with detection of orientation and measuring acceleration, adding support for a handheld-controller, and more further improvements of the chassis' overall design. Overall, this battlebot was a formative experience in designing a multi-disciplinary project from top to bottom with lots of lessons learned amongst all three team members.

5.1 Ethics

As described by Section I Part 1 of the IEEE Code of Ethics [9], we will always put the safety and health of the public first and disclose any potential risks where appropriate. Combat robots pose an inherent threat to public safety, so we will be responsible for ensuring that the robot is designed, handled, and operated responsibly so as to eliminate any risk to the safety of ourselves and others.

As described by Section I Parts 5 and 6 of the IEEE Code of Ethics [9], we will act responsibly as engineers, making sure to openly accept criticism and feedback and to ensure that we have the proper knowledge to accomplish things safely and correctly. Making a robot is a multi-disciplinary effort that requires knowledge in many different fields and technologies. We will make sure that we are doing the proper research, learning, and asking for help in order to complete our project safely and responsibly.

As described by Sections II and III in their entirety of the IEEE Code of Ethics [9], we will make sure to create and maintain a positive, healthy, and collaborative working environment. This includes avoiding using harmful language and holding each other accountable for our actions. We will strive for open and frequent communication and a willingness to help each other in order to make a positive working experience for all those involved.

5.2 Safety

Combat robots and battlebots are inherently dangerous and require thorough safety guidelines to ensure they do not pose a threat to the public. We will be following the safety regulations outlined by the NRC, specifically regarding antweight battlebots [1]. Additionally, we will also be abiding by the safety guidelines for batteries given by the ECE445 course staff [10] as our robot will be battery-powered. On top of the existing guidelines and rulesets, we will be exercising caution during testing and operation. As our design uses motors and weaponry, we will make sure that they are only operational when being actively powered and controlled by a human operator. Additionally, we will not be operating the robot or its motors outside of a safe competition environment or testing area. Finally, we will be implementing multiple kill-switches that will give the user control to disable the robot at any time as needed.

6. References

[1] National Robotics Challenge, “2025 Contest Manual,” Jan. 06, 2025. https://irp.cdn-website.com/9297868f/files/uploaded/NRCContestRules2025-7677ffcb.pdf (accessed Feb. 10, 2025).

[2] “ESP32-S3-WROOM-1 ESP32-S3-WROOM-1U Datasheet 2.4 GHz Wi-Fi (802.11 b/g/n) and Bluetooth ® 5 (LE) module Built around ESP32-S3 series of SoCs, Xtensa ® dual-core 32-bit LX7 microprocessor Flash up to 16 MB, PSRAM up to 8 MB 36 GPIOs, rich set of peripherals On-board PCB antenna.” Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf

[3] “Premium N20 Gear Motor (50:1 Ratio, 460 RPM),” *Servocity.com*, 2025. https://www.servocity.com/460-rpm-micro-gear-motor/ (accessed May 05, 2025).

[4] “Micro Servo - Heavy Duty Servo Motor,” *EZ-Robot*, 2019. https://www.ez-robot.com/store/p19/servo-motor/Micro-Servo/Mini-Servo-Motor.html?srsltid=AfmBOoq0k-v2cL-Rph1v2ZgusYVouHCFWIdlEUb8HsnDK2FcBKooFIHY (accessed May 06, 2025).

[5] “Tattu 650mAh 3S 75C 11.1V Lipo Battery Pack with XT30 Plug,” *GensTattu*, 2025. https://genstattu.com/tattu-3s1p-75c-11-1v-650mah-lipo-battery-pack-with-xt30-plug.html#product-reviews (accessed May 06, 2025).

[6] “LM2674 SIMPLE SWITCHER ® Power Converter, High-Efficiency, 500-mA, Step-Down Voltage Regulator.” Accessed: May 07, 2025. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm2674.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1746630683469&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Flm2674

[7] “TPS563300 3.8-V to 28-V, 3-A Synchronous Buck Converter in SOT583 Package TPS563300 Efficiency, V IN = 24 V, f SW = 500 kHz TPS563300,” 2022. Accessed: May 07, 2025. [Online]. Available: https://www.ti.com/lit/ds/symlink/tps563300.pdf?ts=1746592455085

[8] Texas Instruments, “DRV8231 3.7-A Brushed DC Motor Driver with Integrated Current Regulation.,” Nov. 2021. https://www.ti.com/lit/ds/symlink/drv8231.pdf (accessed Mar. 5, 2025).

[9] IEEE, “IEEE Code of Ethics,” *ieee.org*, Jun. 2020.

https://www.ieee.org/about/corporate/governance/p7-8.html (accessed Feb. 10, 2025)

[10] ECE445 Spring 2016 Course Staff, “Safe Practice for Lead Acid and Lithium Batteries,” Apr. 13, 2016. https://courses.grainger.illinois.edu/ece445/documents/GeneralBatterySafety.pdf (accessed Feb. 10, 2025).

Appendix A

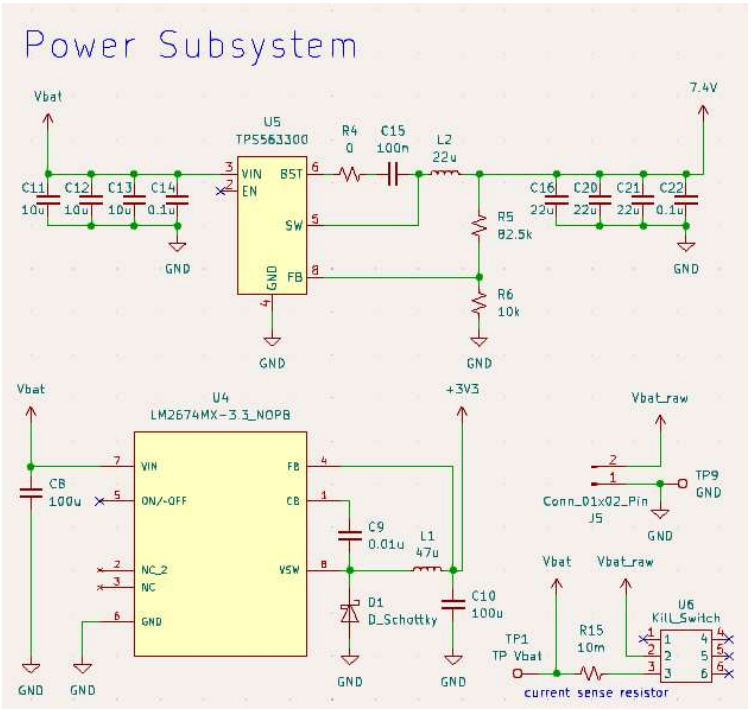


Figure 3: Circuit Schematic for Power Subsystem

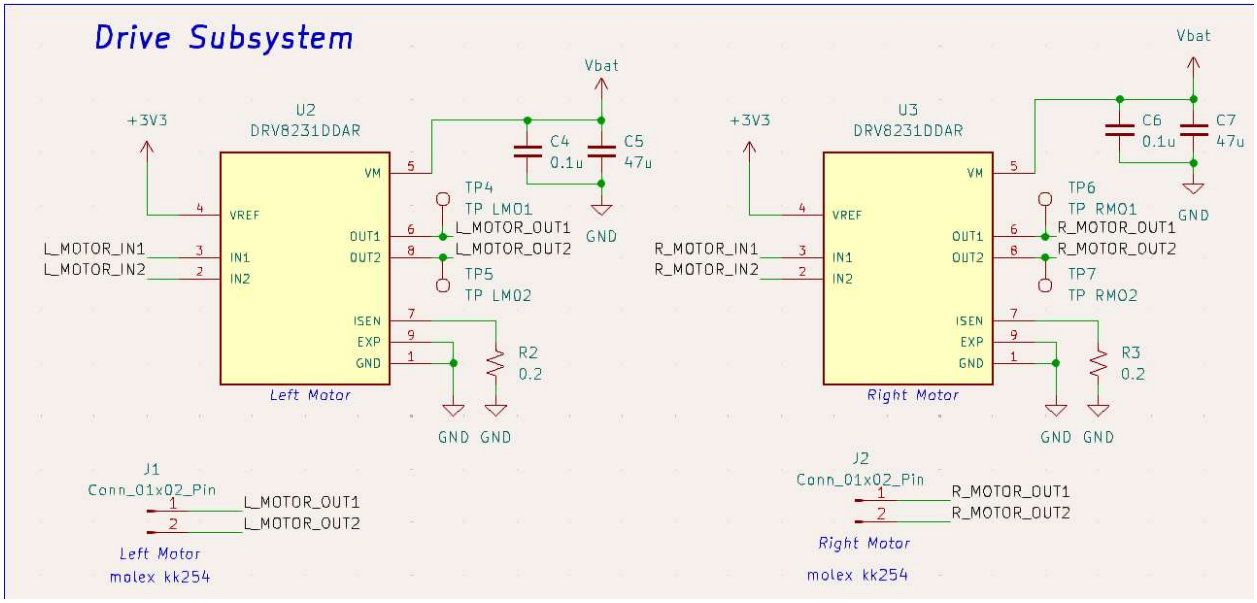


Figure 4: Circuit Schematic for Drivetrain Subsystem

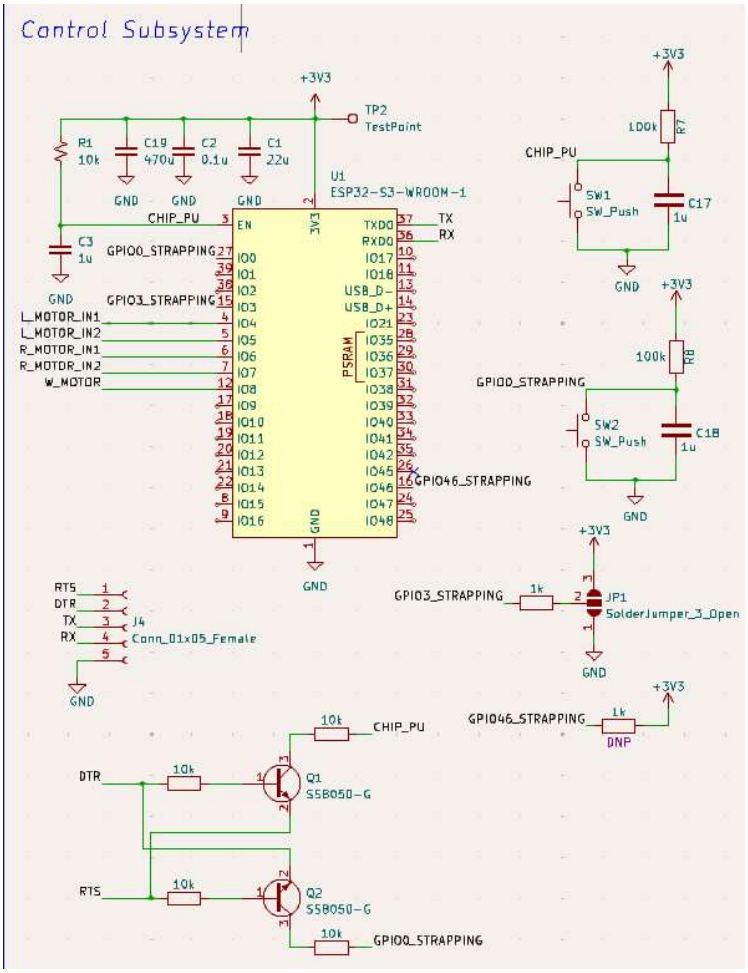


Figure 5: Circuit Schematic for Control Subsystem

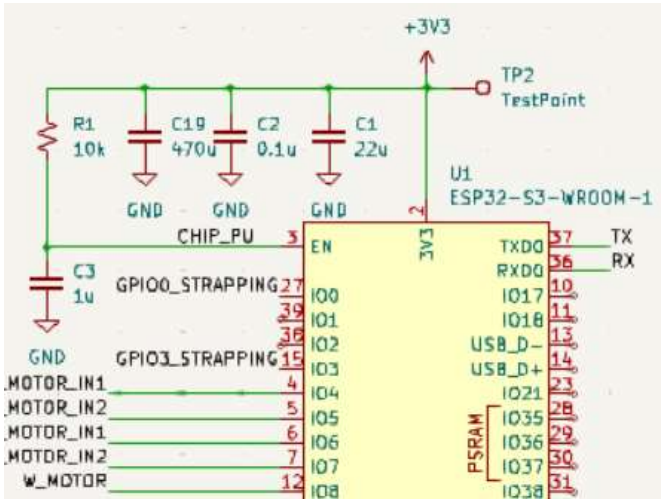


Figure 6: Bulk Capacitance at Voltage Input to ESP32-S3 for Mitigating Inrush Current

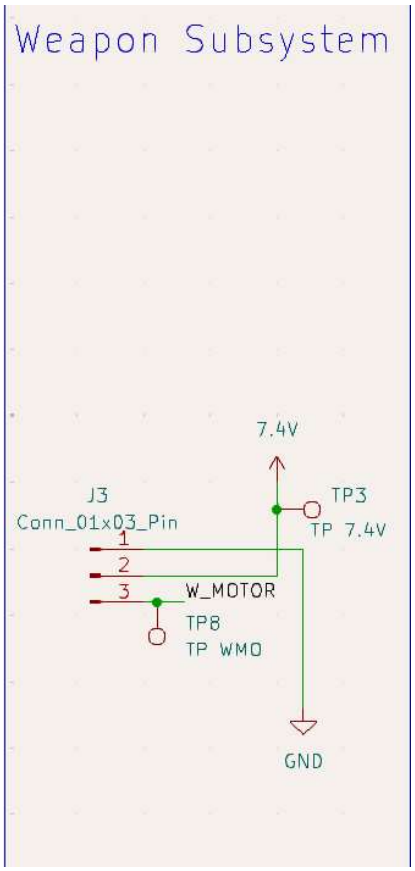


Figure 7: Circuit Schematic for Weapon Subsystem

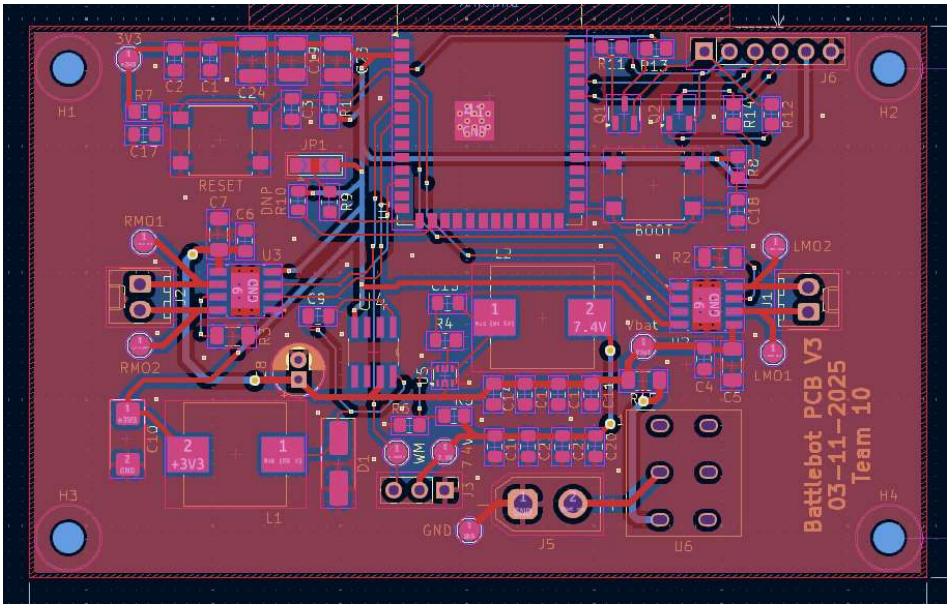


Figure 8: Battlebot PCB Layout

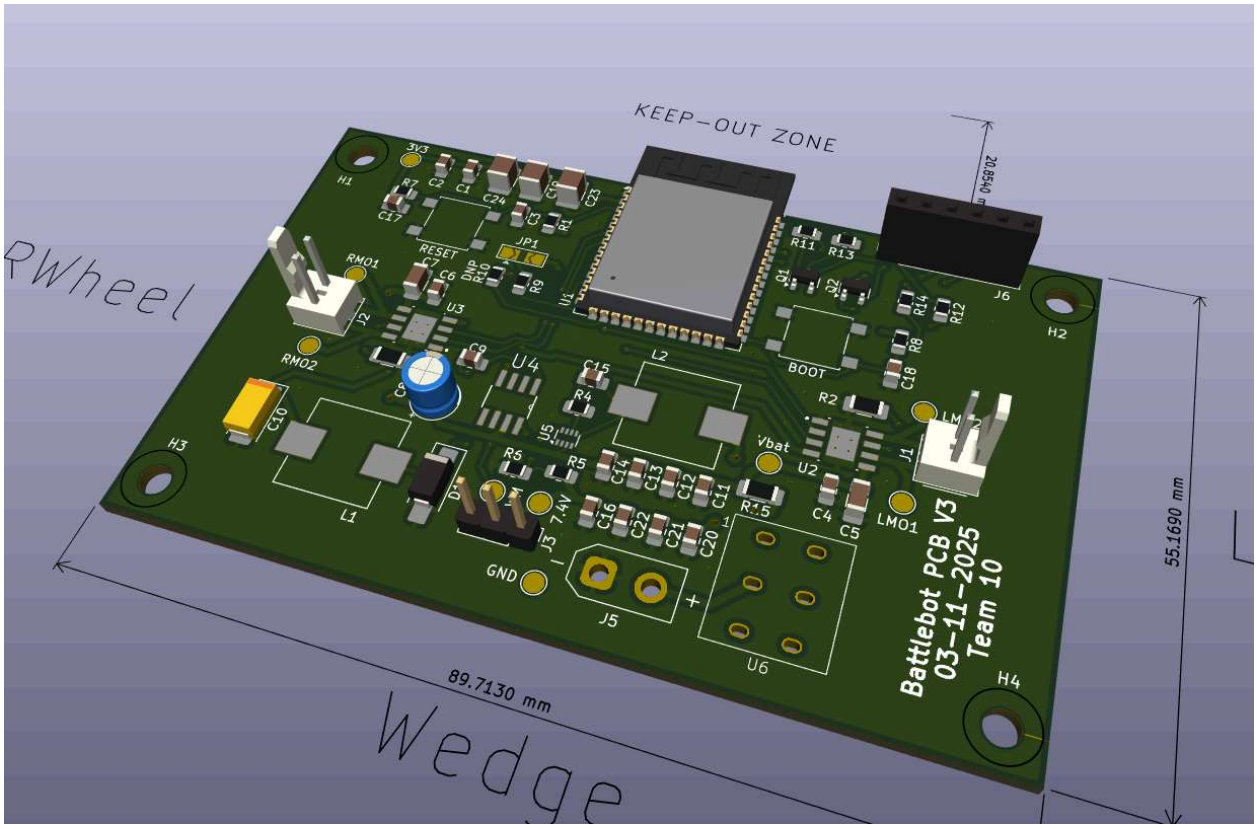


Figure 9: Battlebot PCB in KiCAD 3D Viewer

Appendix B

B.1 Requirements and Verifications for Power Subsystem

Table 2: Power Subsystem R&V Table

Requirements	Verification Method
The battery must be able to supply up to 6.7A at 12V±0.6V.	Test: - To measure the voltage: using a DMM or oscilloscope, probe the battery test point (TP1). - To measure the current: using a DMM, measure the voltage drop across the current sense resistor R15. Then, using its known resistance of 10mΩ, calculate the current using I = V/R. The voltage and current measurements should be recorded as singular numerical values and verified that they fall under the required range.
The LM2674MX-3.3 voltage regulator must be able to step down the battery voltage to 3V3±0.3V	Test: To measure the voltage: using a DMM or oscilloscope, probe the ESP32 test point (TP2). The voltage measurement should be recorded as a singular numerical value and verified that it falls under the required range.
The TPS563300 voltage regulator must be able to step down the battery voltage to 7.4±1V	Test: To measure the voltage: using a DMM or oscilloscope, probe the servo test point (TP3). The voltage measurement should be recorded as a singular numerical value and verified that it falls under the required range.
The battery must be able to power the robot for at least 2 minutes	Test: To test the battery lifetime, we will plug the battery into the robot and prepare a stopwatch. We will start the stopwatch and begin simulating a real combat scenario, continuously activating the drivetrain and weapon motors as if we were in competition.

	This will be recorded as a pass if the robot, under real competition simulation, stays powered and operational for the full 2 minutes. In the case that it does not, it will be recorded as a fail and the time that the robot stops working will be recorded.

Figure 10: Vbat Reading of 12.549 V

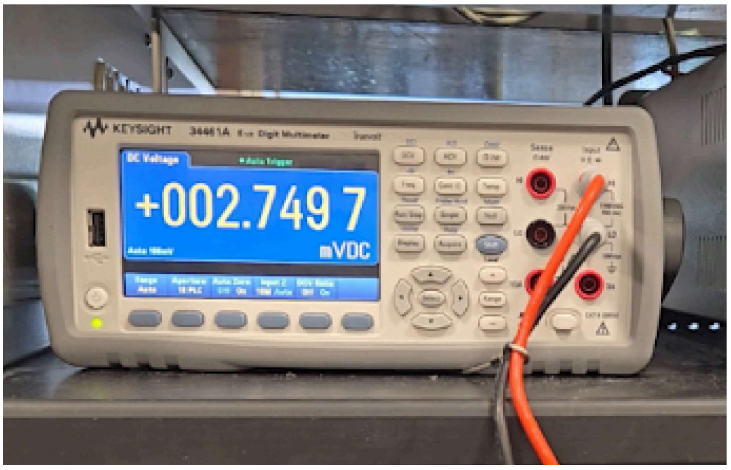


Figure 11: Peak Voltage Measurement Across 10 mΩ Current Sense Resistor

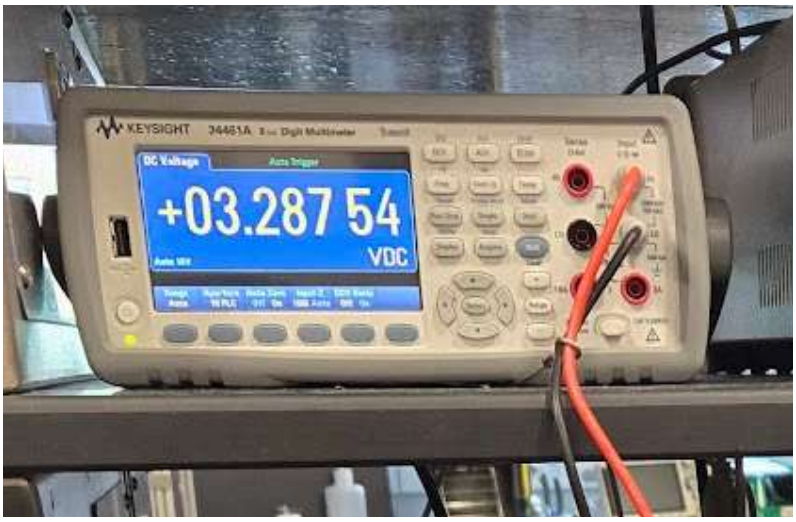


Figure 12: 3V3 Reading of 3.287 V

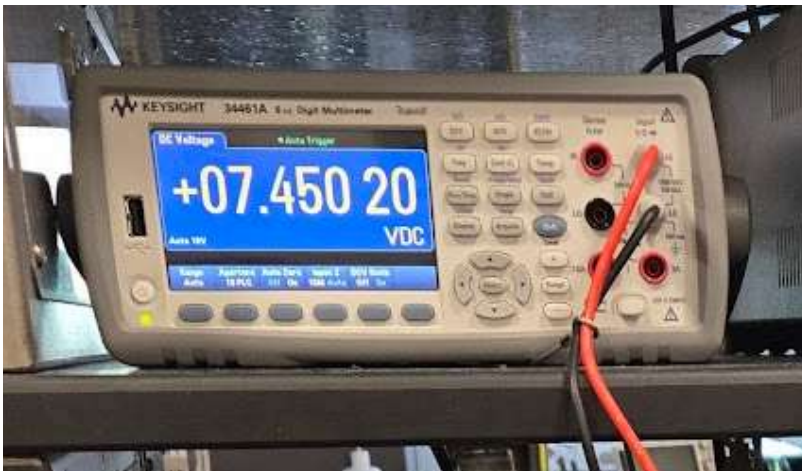


Figure 13: 7.4 V Reading of 7.450 V

B.2 Requirements and Verifications for Drivetrain Subsystem

Table 3: Drivetrain Subsystem R&V Table

Requirements	Verification Method
Each N20 motor must operate at 12V, providing a speed between 310 RPM to 460 RPM and a pull current between 100 mA to 1600 mA.	Test: To ensure each motor is providing a speed between 310-460 RPM, we will use the equation below. *RPM = (Linear Speed * 60) / Wheel Circumference* We note that *Linear Speed* is the maximum velocity that our battlebot is able to achieve. While the motor is rotating the wheel, we will use a multimeter to ensure that the current falls within 100-1600 mA under

	loaded conditions.
Drivetrain subsystem must prevent the battlebot from exceeding a maximum acceleration of 5 m/s^2, ensuring controlled maneuverability across the competition arena.	Test: To measure acceleration, we will run tests in order to verify that the battlebot does not exceed an acceleration of 5 m/s^2. To do this, we will have our battlebot start from rest. We will begin the battlebot's motion at maximum acceleration (full-throttle) and verify the time it takes to pass a marker that is exactly 10 meters away from the starting point. Analysis: Given the data through testing, we will utilize the following kinematic equation to obtain acceleration. We note that v_0 is equal to 0 since we are starting our battlebot at rest and t is the time it takes for our battlebot to cross the 10 meter marker. $s = v_0t + .5at^2$ $10 = .5at^2$ $a = 20 / t^2$
The subsystem must respond to control inputs within 200 milliseconds to ensure accurate, precise maneuverability.	Test: To test latency, we will send a control signal to the microcontroller. Using an oscilloscope and a timer, we will verify that the time from input to motor response is less $\leq$ 200ms.

B.3 Requirements and Verifications for Control Subsystem

Table 4: Control Subsystem R&V Table

Requirements	Verification Method
The microcontroller must maintain a reliable connection with the control PC at a range of at least 15 feet, with a command response time of less than 200 milliseconds.	Test: - To test the range requirement, we will place the robot at a distance of 15 feet from the control PC. After sending a command from the PC, we will visually evaluate whether the robot behaves as expected. - To test the command response requirement: we will probe the respective test points at our motors using an oscilloscope or the ADALM. We will send a command from the

	command PC and measure the signal at the motor. Then, using the measurement we can calculate the time difference between when the command was sent and when it was received. The range requirement will be recorded as pass/fail, where if it happens to fail, the test will be repeated in order to find the maximum distance. The command response time will be recorded as a singular numerical value and evaluated against the maximum allowable response time.
The software-based kill-switch must deactivate the robot within 1 second.	Test: To test the kill-switch mechanism, we will use an oscilloscope or the ADALM to probe a drivetrain motor test point. We will continuously run the motor and record the control signal. Then, we will disconnect the Bluetooth connection and measure the time it takes for the control signal at the motor to cease, even when a command is still being input from the control PC. This will be recorded as a singular numerical value and evaluated against the maximum allowable time of 1 second.

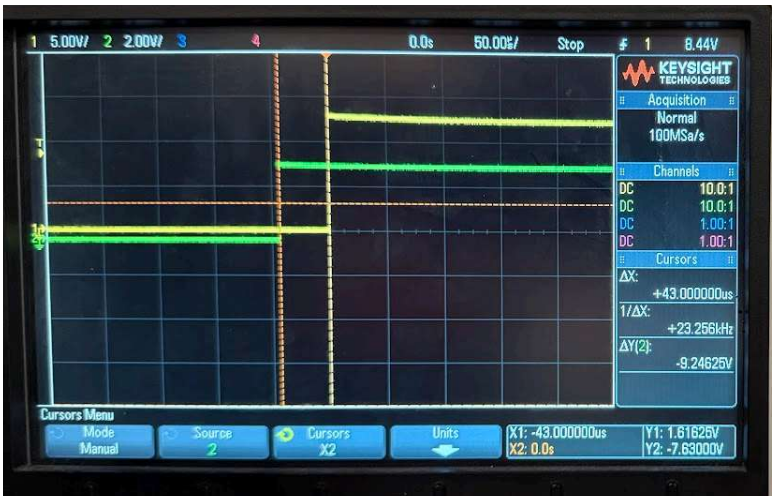


Figure 14: Oscilloscope Reading of Output Signals from Microcontroller and Motor Controller

B.4 Requirements and Verifications for Weapon Subsystem

Table 5: Weapon Subsystem R&V Table

Requirements	Verification Method
The wedge must be capable of lifting and displacing objects of up to 2 lbs.	Test: To test this, we will use a scale to first find an object that weighs at least 2 lbs. We will then verify that the wedge is able to lift this object, unassisted by any other external sources of help. This will be recorded as a pass/fail. If it is a fail, the test will be repeated to find what the maximum weight the wedge can lift is.
The weapon must return to its original position within 2.5 seconds after activation.	Test: Since 2.5 seconds is long enough for the eye to process, we will just be using a timer. We will time how long it takes for the wedge to return to its original position following activation. This will be recorded as a pass/fail. If it is a fail, the test will be repeated to find how long it takes for the weapon to return to its original position.
The wedge must not compromise its structure after any actuation, such that this subsystem can be activated repeatedly.	Test: To test this, we will inspect the wedge's physical structure visually and look for any deformities or other such physical defects. Additionally, we will activate the wedge multiple times in succession, to ensure that it can be activated repeatedly. This will be recorded as a pass/fail. If it is a fail, notes will be recorded detailing the type of defect and on what number activation it failed on.

B.5 Requirements and Verifications for Chassis Subsystem

Table 6: Chassis Subsystem R&V Table

Requirements	Verification Method

The chassis must be able to properly house internal components.	Test: To test this, we will visually inspect the fully assembled robot and ensure that no electronics, besides the physical kill-switch, are exposed in both an upright and flipped orientation. This will be recorded as a pass/fail.
The robot must maintain functionality in both upright and flipped orientations.	Test: To test this, we will operate the robot in both an upright and then a flipped orientation, ensuring that it is still functional in both orientations. This will be recorded as a pass/fail for the upright and flipped orientations individually.

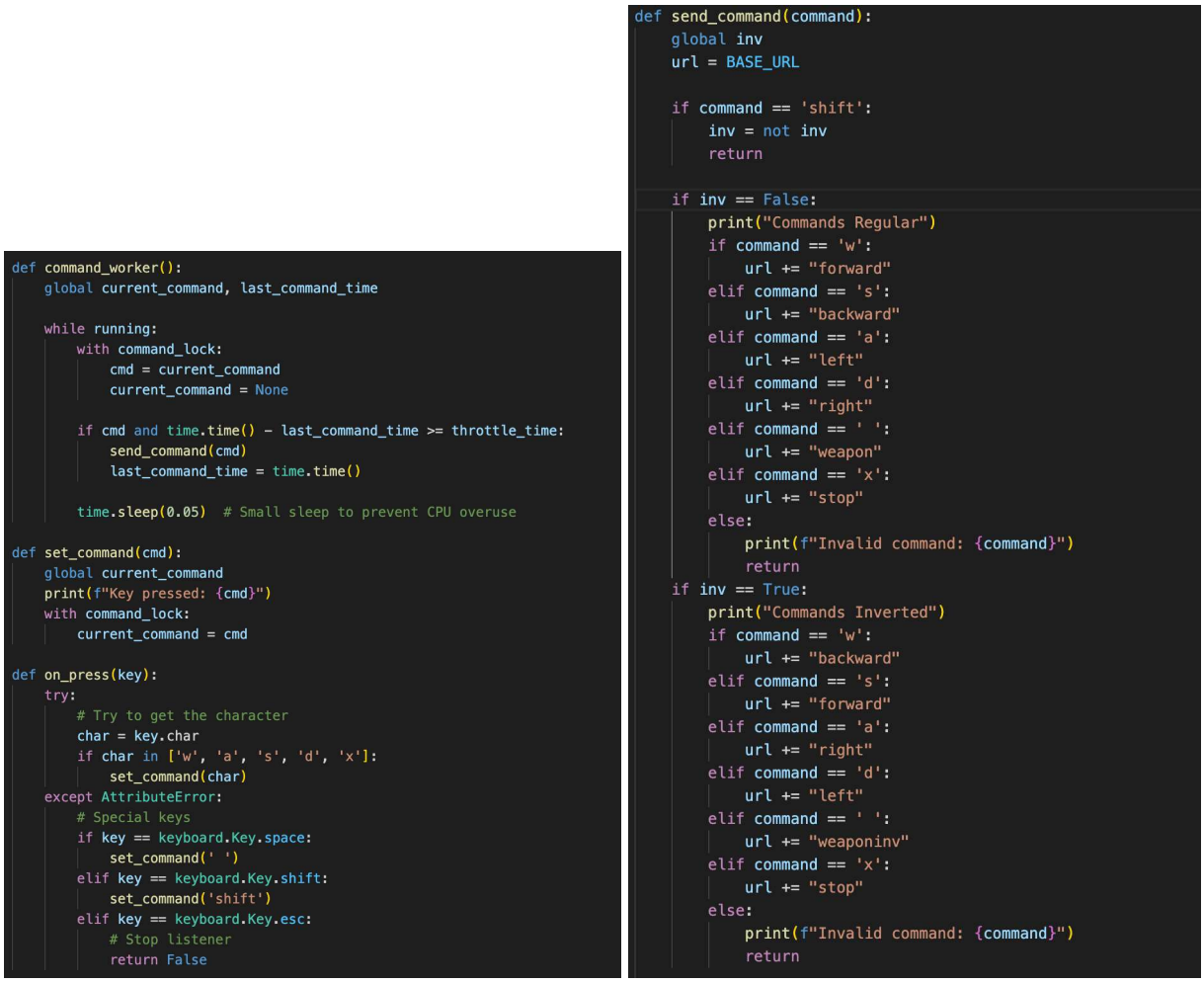
Appendix C

```
void setup() {
  // Initialize Serial Monitor
  Serial.begin(115200);
  Serial.println("BattleBot starting (Wi-Fi)...");


  // Initialize Motor Pins for chip to OUTPUT
  pinMode(IN1_M1, OUTPUT);
  pinMode(IN2_M1, OUTPUT);
  pinMode(IN1_M2, OUTPUT);
  pinMode(IN2_M2, OUTPUT);

  // Initialize weapon servo
  weaponServo.attach(WEAPON_SERVO);
  // Set servo to neutral position initially
  weaponServo.write(90);

  // Stop motors initially
  stopMotors();

  // Wifi mode using 802.11 protocol
  WiFi.mode(WIFI_STA);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);


  // Debug Statement
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("Wi-Fi connected");

  // Use IP address for connection via Python Script
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  // Set up HTTP request handlers
  server.on("/forward", HTTP_POST, handleForward);
  server.on("/backward", HTTP_POST, handleBackward);
  server.on("/left", HTTP_POST, handleLeft);
  server.on("/right", HTTP_POST, handleRight);
  server.on("/stop", HTTP_POST, handleStop);
  server.on("/weapon", HTTP_POST, handleWeapon);
  server.on("/weaponinv", HTTP_POST, handleWeaponinv);


  server.onNotFound(handleNotFound);


  // Start the server
  server.begin();
```

Figure 15: ESP32-S3-WROOM-1 Loaded Set-Up Code

```
// === Motor control functions ===
void moveForward() {
  Serial.println("Move Forward");
  digitalWrite(IN1_M1, HIGH); digitalWrite(IN2_M1, LOW);
  digitalWrite(IN1_M2, HIGH); digitalWrite(IN2_M2, LOW);
}


void moveBackward() {
  Serial.println("Move Backward");
  digitalWrite(IN1_M1, LOW); digitalWrite(IN2_M1, HIGH);
  digitalWrite(IN1_M2, LOW); digitalWrite(IN2_M2, HIGH);
}


void turnLeft() {
 Serial.println("Turn Left");
 digitalWrite(IN1_M1, LOW); digitalWrite(IN2_M1, HIGH);
 digitalWrite(IN1_M2, HIGH); digitalWrite(IN2_M2, LOW);
}


void turnRight() {
  Serial.println("Turn Right");
  digitalWrite(IN1_M1, HIGH); digitalWrite(IN2_M1, LOW);
  digitalWrite(IN1_M2, LOW); digitalWrite(IN2_M2, HIGH);
}


void stopMotors() {
  Serial.println("Stop Motors");
  digitalWrite(IN1_M1, LOW); digitalWrite(IN2_M1, LOW);
  digitalWrite(IN1_M2, LOW); digitalWrite(IN2_M2, LOW);
}

void activateWeapon(){
  Serial.println("Weapon Activated");
  // Rotate servo to maximum position
  weaponServo.write(0); // Full rotation for maximum power
  delay(500); // Keep active for 1 second
  weaponServo.write(90); // Return to neutral position
}

void activateWeaponinv(){
  Serial.println("Inverted Weapon Activated");
  weaponServo.write(180);
  delay(500);
  weaponServo.write(90);
}
```

Figure 16: ESP32-S3-WROOM-1 Code for GPIO Pin Modification



```python
def command_worker():
    global current_command, last_command_time

    while running:
        with command_lock:
            cmd = current_command
            current_command = None

        if cmd and time.time() - last_command_time >= throttle_time:
            send_command(cmd)
            last_command_time = time.time()

        time.sleep(0.05)  # Small sleep to prevent CPU overuse

def set_command(cmd):
    global current_command
    print(f"Key pressed: {cmd}")
    with command_lock:
        current_command = cmd

def on_press(key):
    try:
        # Try to get the character
        char = key.char
        if char in ['w', 'a', 's', 'd', 'x']:
            set_command(char)
    except AttributeError:
        # Special keys
        if key == keyboard.Key.space:
            set_command(' ')
        elif key == keyboard.Key.shift:
            set_command('shift')
        elif key == keyboard.Key.esc:
            # Stop listener
            return False
```

```python
def send_command(command):
    global inv
    url = BASE_URL

    if command == 'shift':
        inv = not inv
        return

    if inv == False:
        print("Commands Regular")
        if command == 'w':
            url += "forward"
        elif command == 's':
            url += "backward"
        elif command == 'a':
            url += "left"
        elif command == 'd':
            url += "right"
        elif command == ' ':
            url += "weapon"
        elif command == 'x':
            url += "stop"
        else:
            print(f"Invalid command: {command}")
            return
    if inv == True:
        print("Commands Inverted")
        if command == 'w':
            url += "backward"
        elif command == 's':
            url += "forward"
        elif command == 'a':
            url += "right"
        elif command == 'd':
            url += "left"
        elif command == ' ':
            url += "weaponinv"
        elif command == 'x':
            url += "stop"
        else:
            print(f"Invalid command: {command}")
            return
```

Figure 17: External Controller User-Input Handler

Appendix D

Figure 18: Orthographic View of Robot's Final Iteration

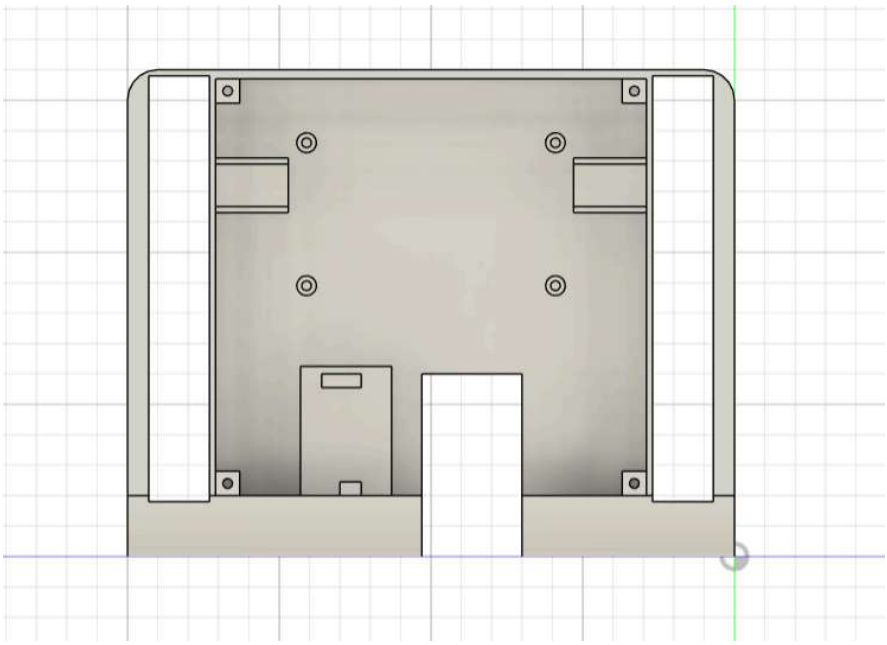


Figure 19: Top-down View of CAD Model of Chassis

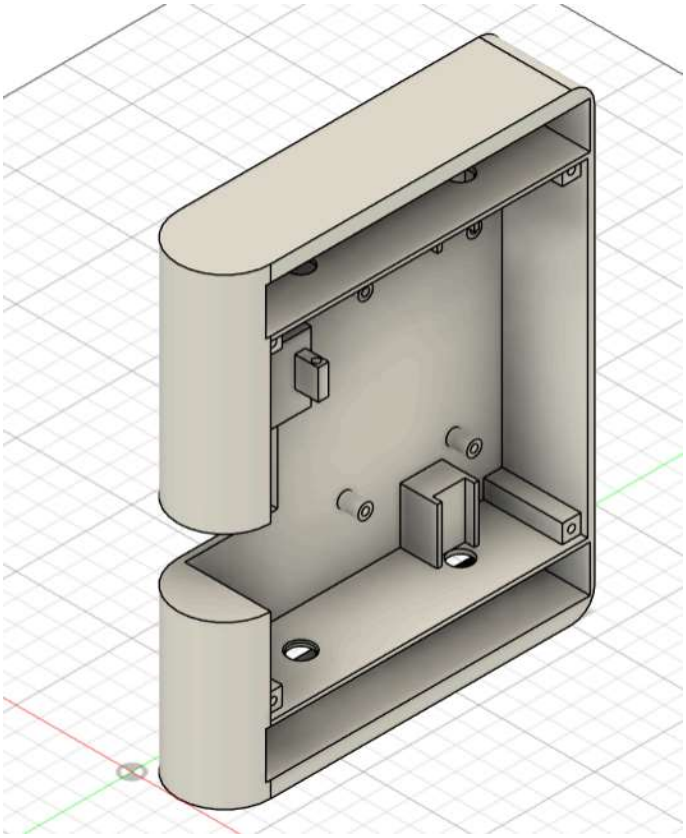


Figure 20: CAD Model of Chassis

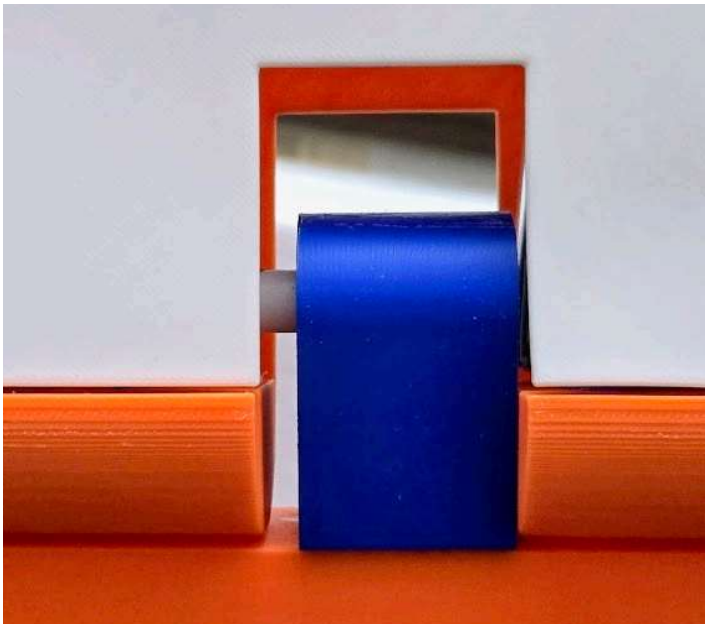


Figure 21: Close-up of Servo-Wedge Arm Assembly

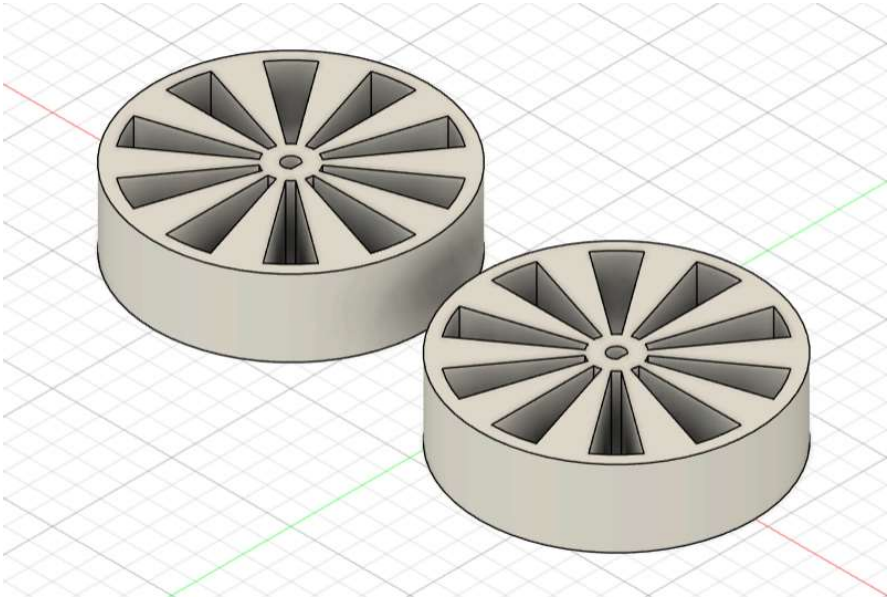


Figure 22: CAD Model for the Drivetrain Wheels

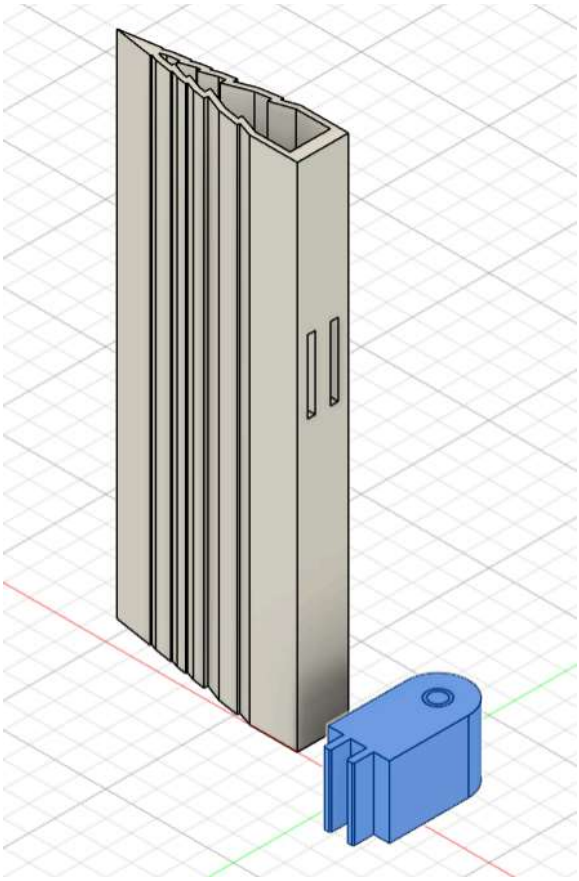


Figure 23: CAD Models of Wedge Blade and Wedge Arm

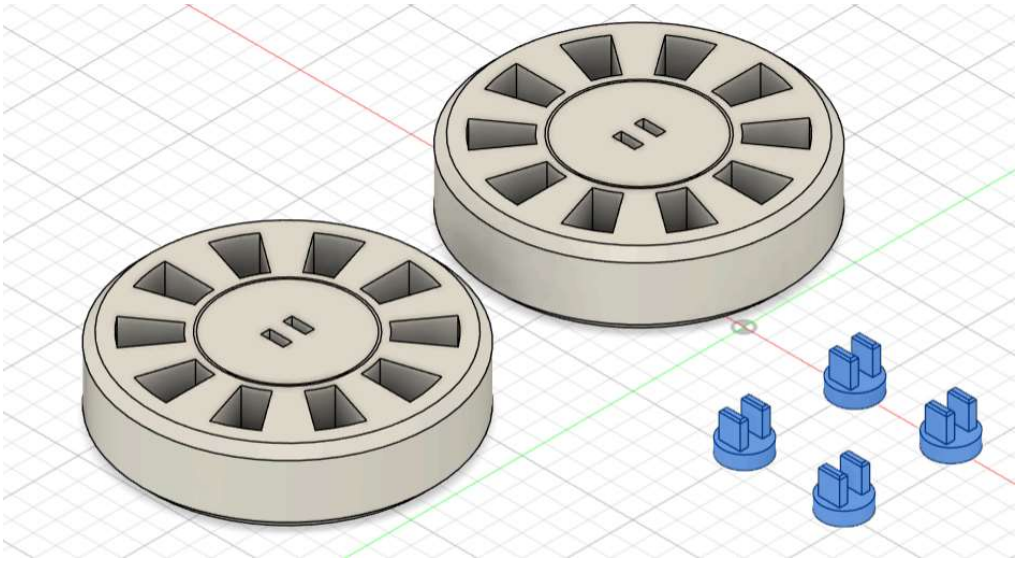


Figure 24: CAD Model for Free Moving Wheels and Fasteners

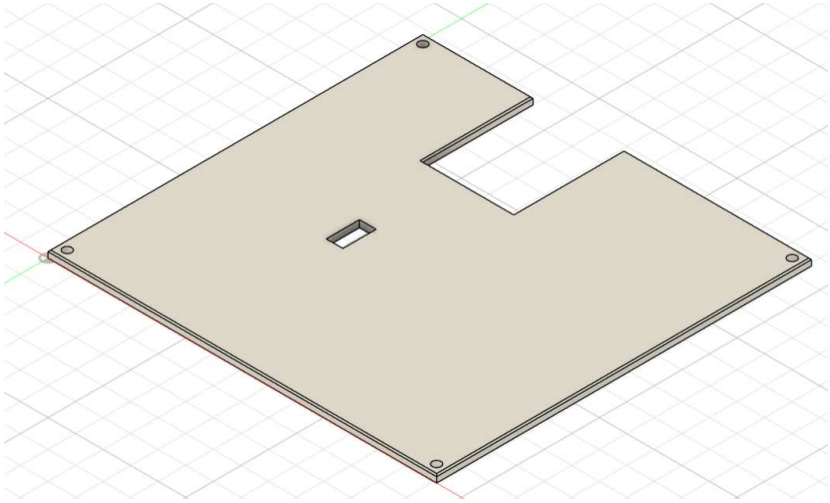


Figure 25: CAD Model for Lid

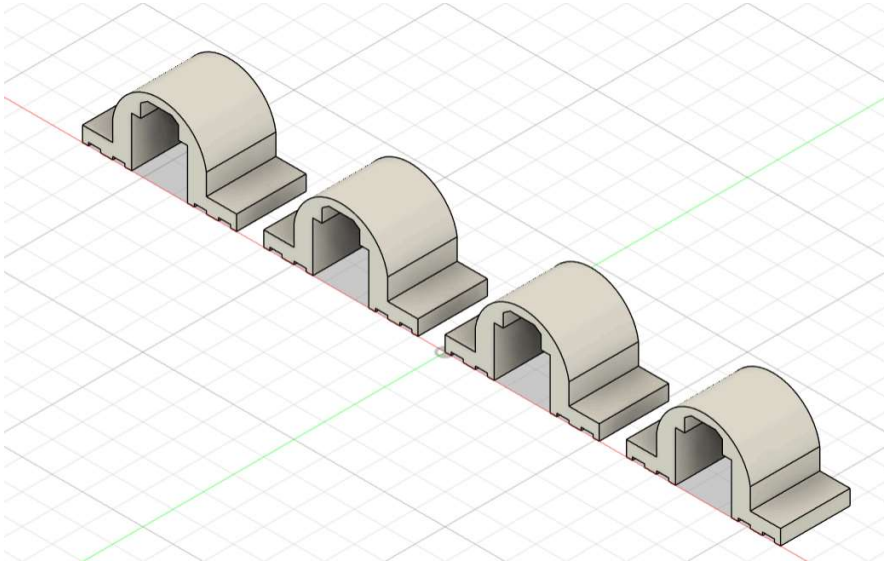


Figure 26: CAD Model of U-Brackets

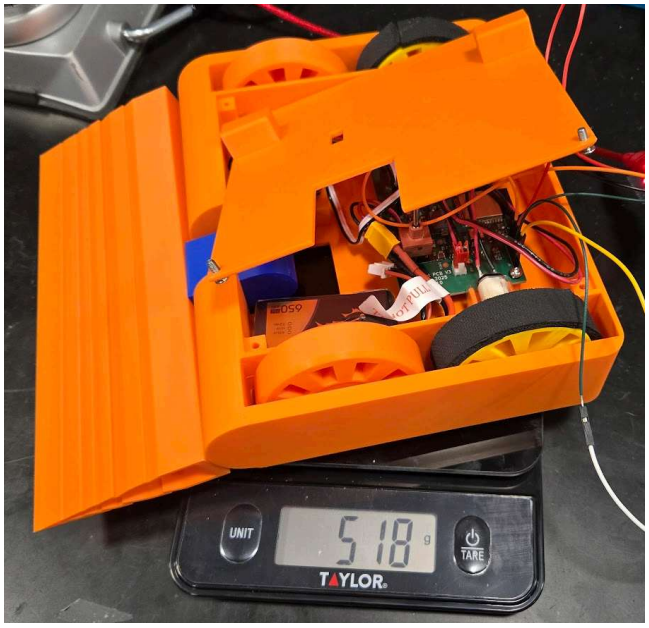


Figure 27: Displayed Weight of Heaviest Iteration of Robot