Vertical Spinner Antweight Battlebot

By:

Andrew Bajek

Elise Chiang

Giovanni Escamilla

Final Report for ECE 445, Senior Design, Spring 2026

TA: Jiaming Xu

May 6th, 2026

Project No. 21

Abstract

This report presents the design, implementation, and verification of an ant-weight (≤ 2 lbs) 3D-printed battlebot for the ECE 445 Battlebots Competition. The bot uses a two-wheel differential drive for maneuverability and a vertical spinner weapon for damage output. A custom printed circuit board(PCB) with an ESP32 microcontroller enables Bluetooth Low Energy(BLE) control of the bot with a latency of 60 ms. The bot contains four electrical subsystems: power, drive, weapon, and control. The battlebot maintains a linear speed of 0.41m/s with a maximum weapon speed of .8m/s, allowing the robot to quickly maneuver the arena and deliver powerful attacks to other competitors. The physical design of the bot allows it to be flipped or roll off its side, ensuring continuous operation regardless of its orientation in the arena.

Contents

1. Introduction

1.1 Problem

Dr. Gruev's Antweight 3D Printed Battlebots Competition entails up to eight battlebots competing in a bracket to determine which can outperform the rest. In order to compete in the competition, our robot must meet specific requirements including but not limited to: a maximum weight of 2 lbs; 3D-printed using PET, PETG, ABS, or PLA, PLA+; controlled using Bluetooth or Wi-fi; and the inclusion of a custom PCB. The custom PCB will house the microcontroller, Bluetooth or Wi-fi receiver, H-bridge, and additional sensors. The battlebot should be able to tolerate extreme mechanical stress from other bots while also being able to deliver damage using a weapon.

1.2 Solution

Our solution is a robot with a two-wheel drive and vertical drum spinner weapon. The robot will consist of four electrical subsystems: power, drive, weapon, and control. Our custom PCB will contain an ESP32 microcontroller which will control the weapon and drive subsystems while also monitoring motor current to limit stress to the driver chips. The microcontroller will also connect to a PC using Bluetooth to remotely control the battlebot. We will use keyboard controls, allowing us to drive the battlebot throughout the arena and control the spinner weapon. The exterior of the battlebot will be 3D printed using PLA, a flexible and durable filament. In case our robot is flipped during competition, it will be able to function right side up and upside down by being symmetrical over the horizontal axis.

1.3 High Level Requirements

- **Mobility**: Given that the arena itself is only 10' by 10', we opt to focus more on control rather than speed. Thus the bot should reach a maximum speed of 1.5m/s
- **Latency**: Our control in the field requires us to have as little latency as possible. Ideally, the bot has a latency less than 50ms; however, the minimum requirement is 250ms.
- **Weapon Speed**: A large portion of our bot's weight will be in our weapon itself. This weight distribution will cause our bot to have harder control while our weapon spins. This issue will be worsened with faster spinning speeds from our weapon, however a higher weapon speed will allow for higher damage. Overall we are still leaning for a weapon speed near 27 m/sfor good damage and control in the field.

2. Design

2.1 Block Diagram

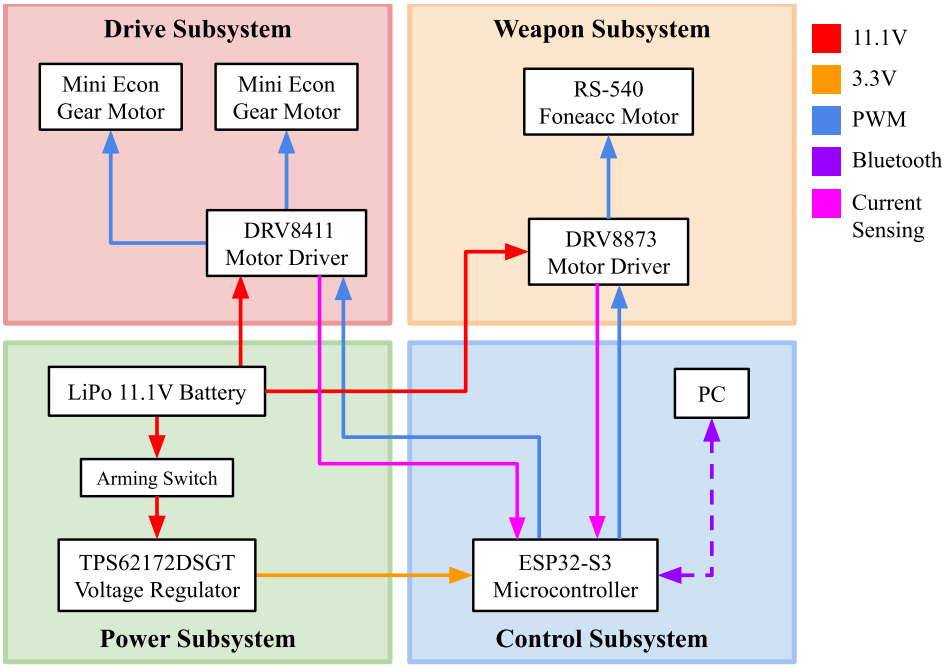

Figure 1: Electrical block diagram

The block diagram in Figure 1 illustrates how each subsystem interfaces with the others. The design is organized into four major subsystems: power, drive, weapon, and control.

The power subsystem accepts the raw battery voltage and uses a buck converter to step it down to the regulated logic rail required by the rest of the board. This rail supplies both the control subsystem and the logic side of the motor driver integrated circuits(IC), while the unregulated battery voltage is routed directly to the driver power stages to feed the motors.

The drive subsystem uses a DRV8411 dual H-bridge driver[2] to control the two drive motors, and the weapon subsystem uses a DRV8873 single H-bridge driver[3] to power the larger weapon motor. Both drivers receive their control inputs from the ESP32-S3-WROOM[4] and their motor-side power directly from the battery rail.

The control subsystem uses an ESP32-S3-WROOM microcontroller[4], which receives user commands from a PC over USB serial, with BLE planned as a secondary control path. The microcontroller translates these commands into the digital control signals with direction inputs and pulse width modulation(PWM), which are needed to operate the two motor driver ICs.

The only signals returned from the drivers to the control subsystem are current-sense outputs and fault flags. The current-sense signals let the firmware monitor motor load in real time, while the

fault flags indicate over-current, under-voltage, or thermal events detected internally by the driver ICs. Together, these feedback paths allow us to implement additional protection logic in software such as current limiting and fault response which are beyond what the driver hardware provides on its own.

2.2 Electrical Subsystems

2.2.1 Power Subsystem

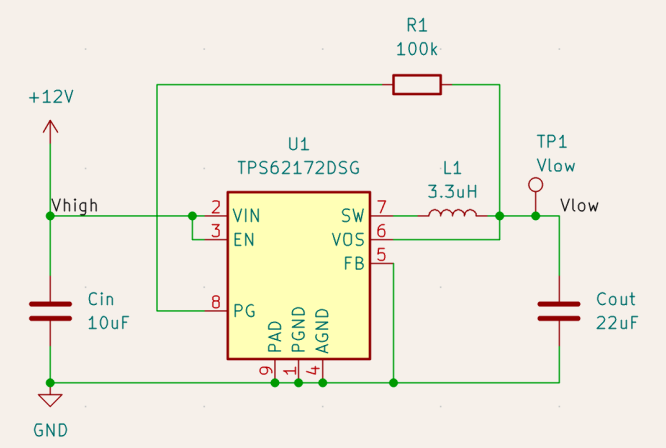

Figure 2: Power subsystem schematic

The power subsystem consists of a 1000mAh lithium polymer(LiPo) battery[5] and a TPS62172DSGT[1] DC-DC voltage regulator, which will step the 11.1V down to 3.3V. The 1000mAh battery was selected due to its lower weight, increased capacity, and power delivery. For safety reasons, we will use an arming plug kill switch between the positive terminal of the battery and the 11.1V bus on the PCB, allowing us to manually disconnect power.

The TPS62172DSGT DC-DC step-down converter can convert a variable input voltage to 3.3V to supply power to the microcontroller[1]. This chip contains only the switching node of the DC-DC converter running at 2.25MHz, meaning we can choose relatively small inductor and capacitor values for a fast transient response.

To ensure our battery is the correct size, we performed load calculations to determine the maximum operation time during normal load conditions and worst case conditions. As shown in Table 1, both the normal load operation time of 2012 seconds and the worst case load time of 493 seconds are greater than the threshold of 180 seconds, so our battery is properly sized. The equation to calculate the maximum operation time is shown in Equation 1.

$$(1\ [Ah]\ *\ 3600[s/h])/\ (Total\ Current\ [A])\ =\ Maximum\ Operation\ Time\ [s] \quad (1)$$

Table 1: Current Draw and Maximum Operation Time

Component	**Qty.**	**Normal Load Current [A]**	**Worst Case Current [A]**
Drive Motor	2	0.3	1.4
Weapon Motor	1	0.89	4.0
ESP32	1	0.3	0.5
Total [A]		**1.79**	**7.3**
Maximum Operation Time [s]		**2012**	**493**

2.2.2 Drive Subsystem

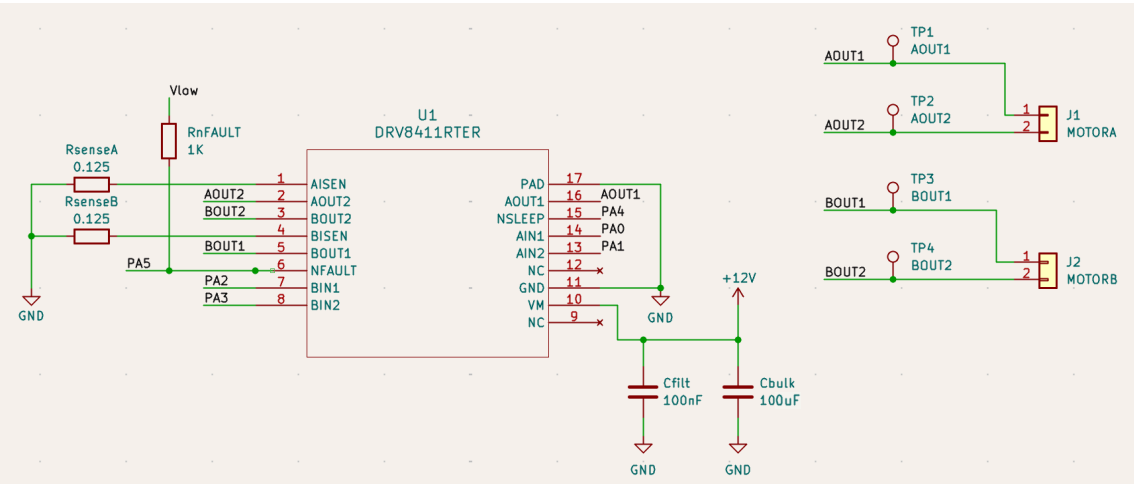

Figure 3: Drive subsystem schematic

The drive subsystem contains two 508 RPM Mini Econ Gear Motors[6] and a DRV8411 Dual H-Bridge Motor Driver[2]. The driver receives 3.3V PWM signals from the microcontroller through XIN1 and XIN2, and outputs 11.1V power signals to the motors through XOUT1 and XOUT2. The duality of this driver allows the motors to move at different speeds and directions, allowing us to create a differential drive system using a single driver chip.

The driver includes two sense pins to allow for current regulation. The trip current I_{trip} is set by a comparator that compares a 200mV reference with the voltage across a sense resistor connected to XISEN. Our drive motors have a stall current of 1.4A, so I_{trip} was selected as 1.6A. Thus, the sense resistors RsenseA and RsenseB were calculated as 0.125 ohms in Equation 2.

$$R_{senseX} = \frac{200mV}{I_{trip}} = \frac{200mV}{1.6A} = 0.125[\Omega] \quad (2)$$

The driver also includes an nFAULT pin that is pulled low during fault conditions, allowing the driver to notify the microcontroller if a fault occurs. For open drain operation, we connected an external pullup resistor RnFAULT, which is between nFAULT and +3.3V. The voltage of nFAULT will fall within the [0.3, 5.75V] range, while the maximum current is 5mA. With this, we can size the pullup resistor in Equation 3.

$$R_{nFAULT} = \frac{V}{I} = \frac{3.3-0.3}{5e-3} = 600\Omega \Rightarrow choose\ 1k\Omega \quad (3)$$

The two capacitors between VM and GND on the driver are a 100nF filter capacitor (Cfilt) and a 100µF bulk capacitor (Cbulk) used to prevent input noise from reaching the driver and stabilize the motor voltage despite parasitic wire inductance.

2.2.3 Weapon Subsystem

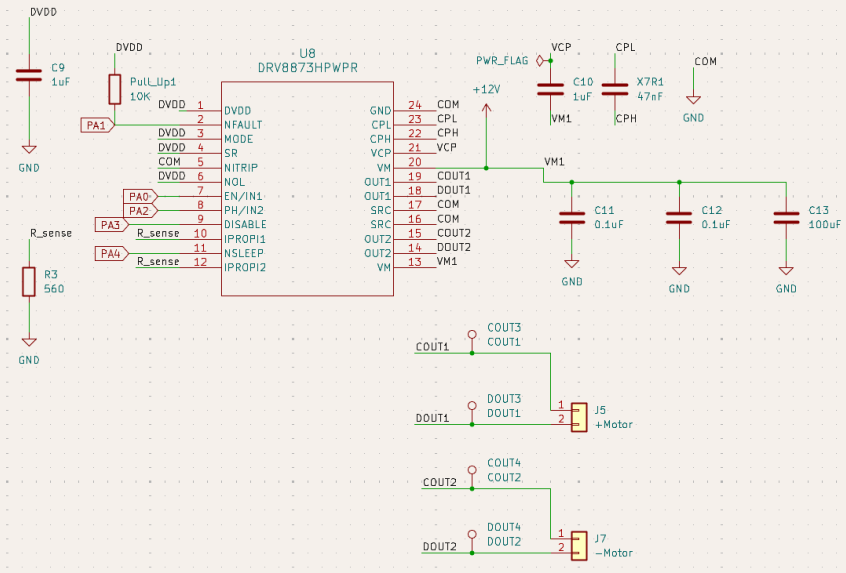

Figure 4: Weapon Subsystem Schematic

The weapon subsystem consists of a vertically mounted solid drum spinner powered by a brushed DC motor[7]. The subsystem converts electrical energy from the 11.1V LiPo battery[5] into mechanical rotational energy using controlled PWM drive signals from the microcontroller. The weapon interfaces with the other subsystems, utilizing the Lipo battery from the power system and PWM control and current sensing from the control system.

In order for the weapon to qualify and succeed, it must have a spin-down time less than or equal to 60 seconds, a mounted operating speed of 7,500 RPM, and a tip speed greater than 27m/s. To ensure the weapon comes to a full stop within the required time, the DRV8873[3] will implement active braking by shorting the motor terminals.

The 3D design of the weapon will implement a cylindrical drum connected to the shaft of the motor. There will also be a bearing on the side without the motor to help rotate and support the drum. The spinner has a tip-to-tip diameter of 70mm, with a width of 35mm.

Given the motor's data sheet, the speed of the tip of the weapon is calculated in Equation 4.

$$v = \frac{2\pi(d/2)\omega}{60} = \frac{2\pi(70/2)(7500)}{60} = 27.4\ [m/s] \tag{4}$$

The drum is powered by an RS-385PH-2270[7] carbon brush motor driven by a DRV8873 H-bridge motor driver[3]. The electrical characteristics of the motor and driver chip are as follows:

RS-385PH-2270:

- Nominal voltage: 11.1 V
- Stall current: 8.7 A
- Normal operating current: 1.6 A

DRV8873:

- PWM-based speed control from ESP32
- Current regulation for stalling
- Protection against overcurrent

A current limit of 6.5 A was selected to remain below the 8.7 A stall current of the RS-385PH-2270[7] motor while allowing sufficient torque during impact events. To achieve this, the following parameters were applied:

- I_0 = 6.5 A
- k = 1100
- V_{ref} = 3.3 V (From ESP32)

$$R_{Sense} = \frac{kV_{ref}}{I_0} = \frac{(1100)(3.3)}{6.5} = 560\ [\Omega] \tag{5}$$

2.2.4 Control Subsystem

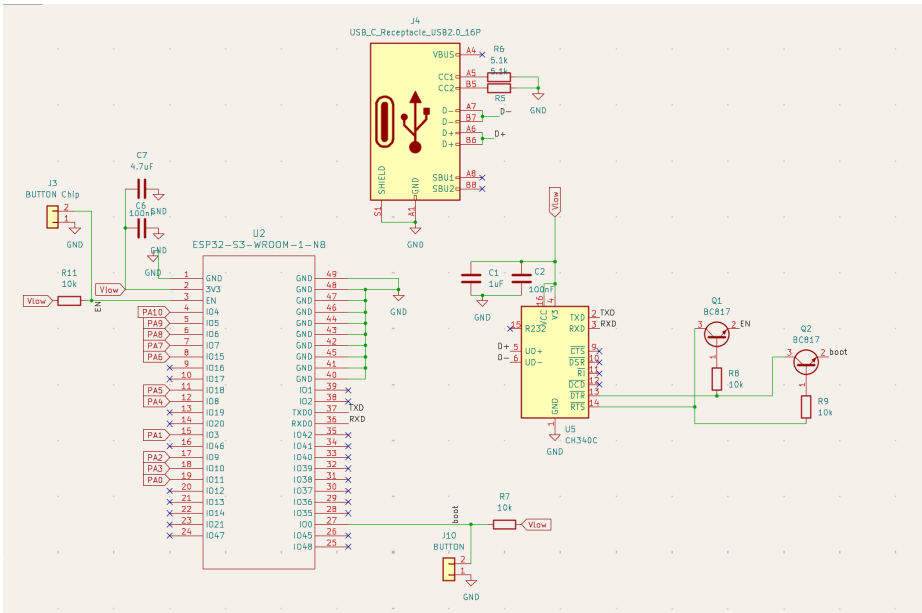

Figure 5: Control Subsystem Schematic

The final design for our control system is centralized around our ESP32-S3-WROOM[4] that sits center stage in our PCB as shown in Figure 5. All subsystems are connected to the control system to either be monitored or controlled by the user via Bluetooth. The two drivers for the drive and weapon subsystems are connected directly to our control in 11 of our general purpose input/output(GPIO) with two additional GPIOs being hooked up by jumper wires to monitor the motor currents in our system.

This final design arose from multiple iterations and changes due to major time constraints and issues with ordering the correct parts. Many of these issues lead to simplifications in the design over time from the STM32-based[8] design to the current one. These changes led to the new programmer, allowing us to communicate to the ESP via a USB-C connection straight from a laptop. We could also remove the antenna system that was used in the STM32-based[8] circuit since the ESP32-S3-WROOM[4] simplified many circuits we originally needed to design. We also dramatically changed the footprints for many components to make the assembly process easier, making soldering far more manageable compared to the previous design.

Our firmware was rudimentary in order to make commands to our controller simple and easily replicated in other devices, with base commands to direct our weapon and drive motors. We also had our ESP constantly sending updates on the voltage being read in GPIO 1 and 2 in order for us to calculate the currents being drawn by any of the motors. Load and stress made it common for the demand from the motor being higher than what was allowed by the driver chips we chose. This led to current sensors having a vital purpose in limiting current through our chips and throwing faults at the chips to prevent them from overheating or damaging the PCB.

On our PC we wrote Python code to latch on the Bluetooth signal being sent when the robot was on. Once latched, we simply used the keys W, A, S, D, I, and K to control the robot, with WASD being our drive control and IK being our weapon control. Knowing that our bot was able to be flipped, our weapon could spin in both directions to allow us to swap the direction of spin if needed in a fight. Additionally, the current data was monitored on our terminal, creating a simple feedback loop of current reading. If the current exceeded our set limit, the motor would be given a fault signal, causing the driver to attempt to push the current elsewhere to cool the chip and limit damage. This saved our chips from constantly burning out, leading to consistent operation.

2.3 Physical Design

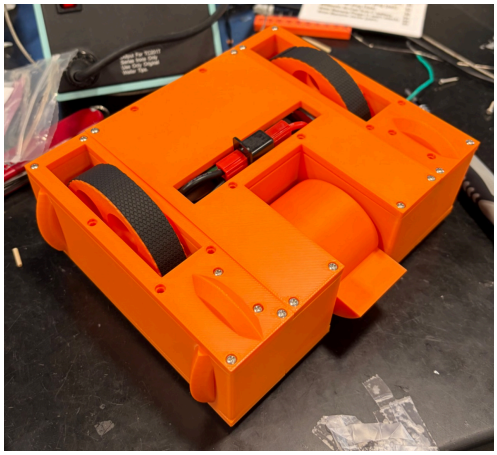

Figure 6: The completed battlebot, “Cheddar Shredder”

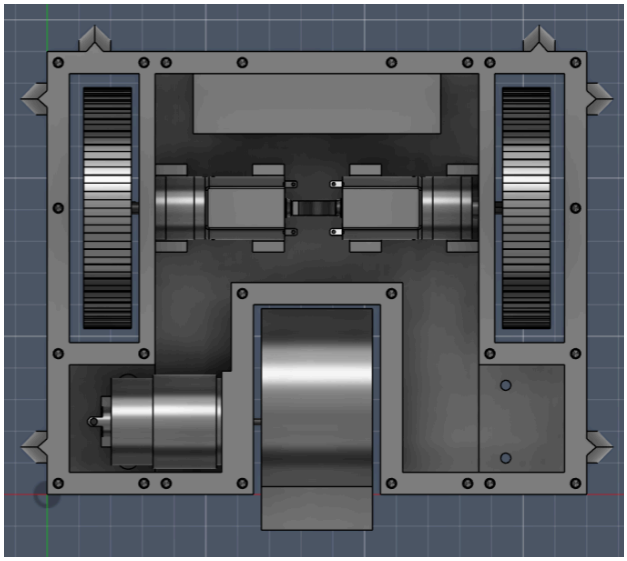

Figure 7: Top-down view of interior of the battlebot in Autodesk Fusion 360

The physical design of our battlebot has to balance durability, weight constraints, and offensive capabilities while adhering to competition standards. We opted for a simple box-like design with additional curved spikes that act as both rollover protection and offensive features for damaging other bots' weapons. To counter flipper-type bots, our robot is symmetric over the horizontal axis so it can still operate when flipped 180 degrees. The wheels have rubber treads and are internal to the body of the robot to prevent external attacks from damaging the drive system. The two-wheel drive causes our bot to drag on the floor; therefore, there are fins positioned in front of the wheels to prop up the bot, increasing ground clearance at the weapon axle and enabling a larger weapon design.

The chassis of the bot has external dimensions of 17.0×14.0×5.5cm in width, depth, and height respectively. Including the wheels and fins, the height is 7.62cm, or 3.0in. The height dimension of the bot and the dimension of the weapon motor limit the weapon size, so our weapon could only have a maximum diameter of 7.0cm and a width of 3.5cm, extruding from the body of the bot by only about 1cm. While this limited the reach of our weapon, it also protected our weapon from attacks to its sides, protecting the motor axle.

We used PLA filament provided by the ECE 445 lab for all the components at different infill percentages based on their usage. To save on weight, for the wheels, top plates, and bottom plates, we used 20% infill since these components are less likely to be damaged by other bots. However, because the body and the weapon must sustain and deliver significant damage, we used 30% and 50% infill respectively.

3. Design Verification

3.1 Power Subsystem

Power verification was relatively straightforward compared to our other subsystems; in most cases, simple voltage readings were sufficient to confirm correct operation. Our primary check focused on the TPS62172DSGT[1] buck converter, which generates the 3.3V logic rail. A test point was placed near the converter output to allow direct probing during bring-up. Across multiple measurements, we consistently read approximately 3.25V against a 3.3V target with a deviation of roughly -1.5%. These measurements were well within the TPS62172's ±3% output tolerance and comfortably inside the supply range required by the ESP32-S3-WROOM[4] and the rest of our logic-side ICs.

A secondary check used the DVDD output on the DRV8873[3], which produces a regulated 5 V rail whenever the chip is powered. DVDD is exposed on pins 1, 3, and 4 of the package, which we routinely probed pin 1 for convenience. Although we hadn't originally planned to use this rail, it served as a reliable indicator of whether the DRV8873[3] was receiving power, and we could check it any time we suspected a damaged chip or a power-delivery issue elsewhere on the board.

Our final check was to confirm that the system could run from the battery[5] for at least three minutes, which is the minimum runtime needed for a single competition fight. We ran the robot against a three-minute timer during initial testing, and it continued to operate well past the requirement with no noticeable drop in performance. We have three battery packs available for the competition and demo, but the first pack performed so well that we kept it installed for the demo, the competition itself, and all subsequent driving. Across competition fights, post-competition driving, and bench testing, this single pack has accumulated at least half an hour of cumulative runtime and at the time of writing still hasn't been swapped out. This significantly exceeds our three-minute requirement and demonstrates that the power subsystem is well over-spec for the actual load.

3.2 Drive Subsystem

The drive subsystem required several checks before we were confident in its operation. The first test verified that the motors themselves would respond to driver signals as expected. We built a discrete H-bridge on a breadboard and used it to drive the motors directly, confirming that they spun in both directions at proportional speeds before we ever connected them to the DRV8411[2]. After this, the remaining work was on the control firmware and the driver chip itself.

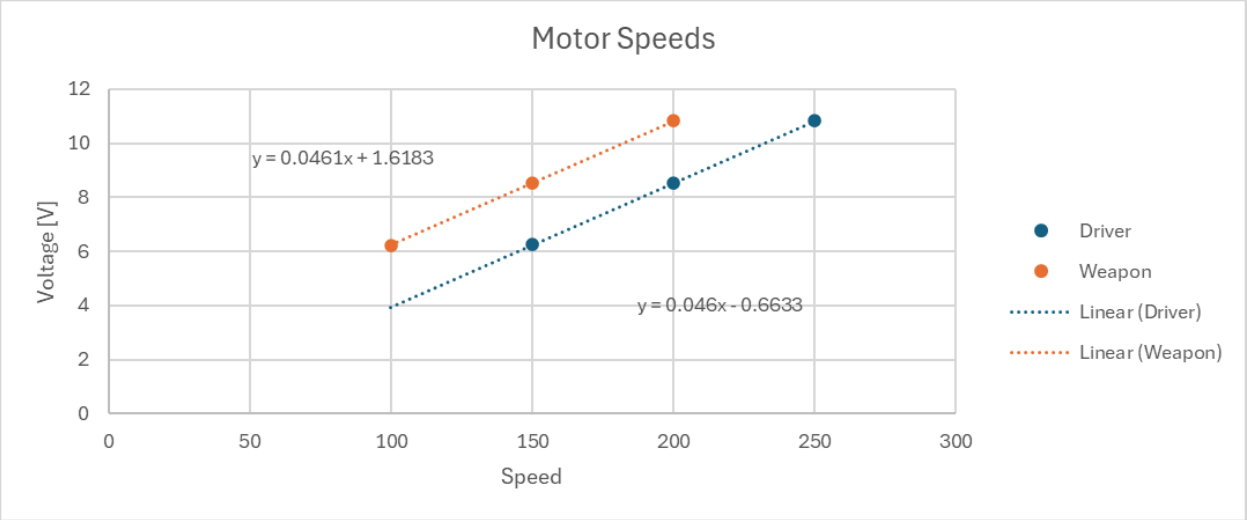

Figure 8: Voltage Readings in Relation to Speed in Software

Once the control firmware was fully running, we tested each drive motor independently to confirm that the DRV8411[2] outputs responded correctly to commands from the ESP32-S3[4]. Discrete commands were sent to drive each motor for 30 seconds in each direction at incremental PWM duty cycles. The voltage measured at the motor terminals at each step is plotted in Figure 8, showing the expected relationship between commanded duty cycle and average motor voltage. This also confirmed that the four GPIOs driving AIN1, AIN2, BIN1, and BIN2 were correctly mapped, and that the left and right wheels could be controlled independently.

With the control side validated, we assembled the bot at near-final weight and tested how the motors held up under competition load. Motor current rose noticeably under load, as expected, while the average motor voltage held near its commanded value. During this same test we measured the bot's physical top speed by timing it across a fixed distance, and at a PWM setting of 200 (out of 255), the bot covered ground at roughly 0.41m/s. This is well under our 1.5 m/s upper bound from Section 1.3, but driving the assembled bot at that speed made it clear that anything close to 1.5m/s would be uncontrollable in a 10ft×10ft arena. The 0.41m/s figure proved much more appropriate for the precision needed in competition.

3.3 Weapon Subsystem

The weapon subsystem consists of a vertical drum spinner to deal damage to other battlebot. In order to work, The DRV8873[3] motor driver needs to be powered by the TPS62172DSGT[1] DC-DC voltage regulator. When the DRV8873[3] reads 5V at the DVDD, the chip will work as intended.

The first verification that was checked was making the weapon motor rotate. To achieve this, the ESP32-S3-WROOM[4] had to communicate with the computer to know when to turn the motor

on. This was accomplished by programming the ESP32-S3-WROOM[4] to turn on the motor when either I or K are clicked (see Section 3.4 for more detail). The second verification was making sure the weapon is able to shutdown within 60 seconds. To test this, the computer needs to disconnect from theESP32-S3-WROOM and time how long it takes to stop. The time that was recorded was about 2 seconds, so well within range. Lastly, the weapon needed to be able to hit objects to ensure damage can be applied. The test for this was using a plastic water bottle as a target and hitting it while the weapon is spinning. The result of this test is the water bottle was hit forward about 5ft, so the weapon worked as intended.

3.4 Control Subsystem

```
t=1064.31s | m1=-245 m2=-245 m3=   0 | weapon= 0.00 A (0 mV) | driveA= 0.11 A (14 mV) | flt=0
t=1064.36s | m1=-245 m2=-245 m3=   0 | weapon= 0.00 A (0 mV) | driveA= 0.10 A (13 mV) | flt=0
ESP: > 1 -245
ESP: m1=-245
ESP: > 2 -245
ESP: m2=-245
ESP: > 3 0
ESP: m3=0
t=1064.41s | m1=-245 m2=-245 m3=   0 | weapon= 0.00 A (0 mV) | driveA= 0.27 A (34 mV) | flt=1 [nFAULT]
t=1064.46s | m1=-245 m2=-245 m3=   0 | weapon= 0.00 A (0 mV) | driveA= 0.14 A (17 mV) | flt=0
ESP: > 1 -245
ESP: m1=-245
ESP: > 2 -245
ESP: m2=-245
ESP: > 3 0
ESP: m3=0
t=1064.51s | m1=-245 m2=-245 m3=   0 | weapon= 0.00 A (0 mV) | driveA= 0.15 A (19 mV) | flt=0
t=1064.56s | m1=-245 m2=-245 m3=   0 | weapon= 0.00 A (0 mV) | driveA= 0.13 A (16 mV) | flt=0
ESP: > 1 -245
ESP: m1=-245
ESP: > 2 -245
ESP: m2=-245
ESP: > 3 0
ESP: m3=0
st=1064.61s | m1=-245 m2=-245 m3=   0 | weapon= 0.00 A (0 mV) | driveA= 0.34 A (42 mV) | flt=0
```

Figure 9: UI for Current reading and Fault Detection

In the early stages, we used Bluetility[9] to send discrete commands to the robot over BLE by writing directly to the Nordic UART Service characteristics. These commands allowed us to turn individual motors and sensors on, run them at specific speeds, and stop them after fixed durations. This let us confirm that the control firmware was responding correctly and that the BLE link was stable end-to-end.

To quantify the responsiveness of the BLE link, we wrote a short Python script that exchanged packets between the PC and the ESP32-S3-WROOM[4] over BLE and measured the round-trip time of each exchange. At a distance of five feet, the round-trip latency held consistently around 60ms. While this figure varies with distance and channel load, it remained well below our 250ms target, confirming that the link is fast enough for real-time control.

Finally, we verified the over-current fault response of both motor drivers. Our nominal trip thresholds are set to 1.2A on the Drive driver[2] and 8A on the Weapon driver[3] were our values chosen to give each chip thermal headroom before reaching its absolute maximum rating. For the verification test, we deliberately lowered the threshold to 700mA so we could trigger a

fault without risking damage to the drivers or motors. As shown in Figure 9, once the measured current crossed 700mA, the firmware-side fault response engaged and pulled the current back down. The reading dropped to roughly 270mA, followed by a smaller 140mA event during recovery. This confirms that our software-side current limiting reacts correctly to a real over-current condition.

3.5 Physical Design

The 3D printed components of the battlebot must be able to withstand attacks from other robots while also delivering damage. The body itself was tested by applying 30N of force by placing the body on a scale and pressing down on critical points. While the body did deform slightly, it did not break. Similarly, the weapon was tested by applying 50N of force to each of the points, which resulted in no deformation. Thus, the main 3D printed components passed the impact tests.

The traction of the wheels was tested by accelerating the robot to maximum speed and observing whether slipping occurred. With the original Flex Tape wheels, the robot lost traction and could not accelerate quickly. Switching the wheel treads to rubber grip tape resulted in a fast acceleration without slipping, meaning the wheels passed the traction test.

Additionally, the entire robot must not exceed a weight of 2lbs. With the internal components weighing in at 0.84lbs, it is crucial that the 3D printed components weigh less than 1.16lbs. The weight of all the 3D printed components was 0.74lbs, meaning that the total weight of our robot is 1lb 9.3oz, which is well under the weight limit.

Finally, while this is not a test that was performed before the final demonstration, it is worth noting the minimal damage our bot sustained during the competition. The weapon shown in Figure 10 was used for four rounds, remaining intact and fully functional, demonstrating its durability and ability to withstand high-speed collisions. The front of the bot shown in Figure 11 also demonstrates the robustness of our design. While the bot has a few scratches and gouges, other bots could not break or pierce through the body, meaning the internal components remained protected and the bot maintained functionality throughout the competition.

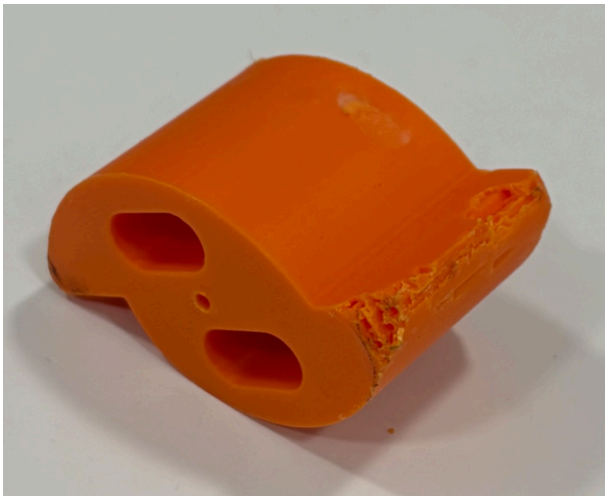

Figure 10: The spinner weapon after the competition

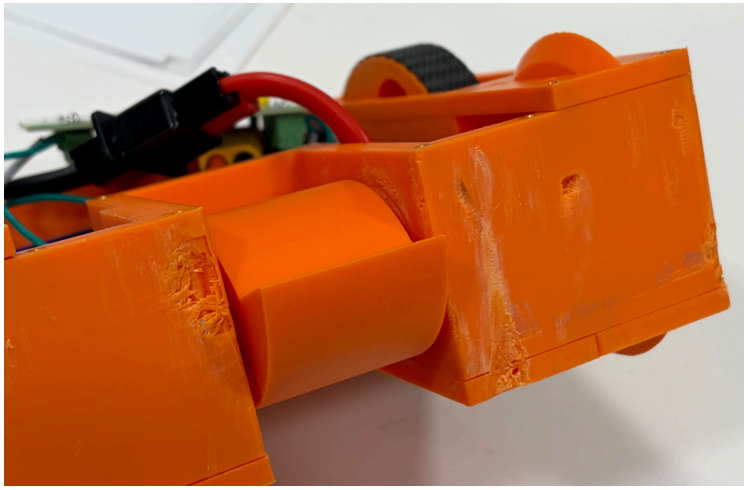

Figure 11: The front of the bot after the competition

4. Costs

The total cost of components used in this project not including shipping was $169.68, as shown in Table 2. The total cost of labor was $6,750, as shown in Table 3. Thus, the total cost for this project was $6,919.68.

4.1 Parts

Table 2: Parts Costs

Description	Provider	Qty.	Bulk Cost	Retail Cost	Total Cost
508 RPM Mini Econ Gear Motor[6]	Servo City	2	$12.99	$12.99	$25.98
DRV8411 Dual H-Bridge Motor Driver[2]	Digikey	5	$1.20	$2.19	$6.00
TPS62172DSGT Buck Converter [1]	Digikey	6	$1.72	$1.72	$10.32
RS-540 Carbon Brushed Micro DC Motor[7]	Foneacc	1	$16.90	$16.90	$16.90
DRV8873HPWPR[3]	Digikey	5	$2.78	$4.15	$10.85
1000mAh 3S (11.1V) 40C Lipo Battery[5]	HobbyKing	3	$8.99	$8.99	$28.97
XT60 Arming Plug	Dave's R/C Electronics	1	$9.99	$9.99	$9.99
PLA Filament	Bambu Store	350g	$4.54	$4.54	$4.54
ESP32-S3-WROOM-1-N8 [4]	Digikey	3	$4.05	$5.66	$5.66
STM32 [8]	Estore	3	$4.66	$4.66	$13.98
256X157C Heat Sinks	Digikey	50	$0.167	$0.26	$8.93
Screws	Digikey	100	$0.04590	$0.04590	$4.95
BC817-25,215 BJT	Digikey	10	$0.074	$0.12	$0.74
CH340C	Amazon	5	$1.79	$1.79	$8.95
10Kohm Resistor	Digikey	10	$0.007	$0.01	$0.07

5.1Kohm Resistor	Digikey	10	$0.005	$0.01	$0.05
0.1uF Cap	Digikey	20	0.093	$0.13	$1.86
1uF Cap	Digikey	20	$0.003	$0.004	$0.06
4.7uF Cap	Digikey	10	$0.038	$0.05	$0.38
100uF Cap	Digikey	10	$0.5840	$0.65	$5.84
22uF Cap	Digikey	8	$0.08	$0.08	$0.64
1Kohm Resistor	Digikey	25	$0.0068	$0.0089	$0.17
USB-C Connection	Digikey	3	$0.89	$0.89	$3.15
XT60 Connectors	Digikey	4	$2.50	$2.50	$10
2.7nH Inductor	Digikey	10	$0.031	$0.04	$0.31
47nF Cap	Digikey	10	$0.075	$0.085	$0.83
125mohm Resistor	Digikey	10	$0.64	$0.93	$6.40
0.8pF Cap	Digikey	5	$0.23	$0.23	$1.15
0.3pF Cap	Digikey	5	$0.38	$0.38	$1.90
LPF	Digikey	3	$0.49	$0.49	$1.47
Antenna	Digikey	5	$0.66	$0.66	$3.30
40Mhz Crystal	Mouser	3	$0.44	$0.44	$1.32

4.2 Labor

Table 3: Labor Costs

Team Member	**Hours Spend**	**Hourly Rate**	**Total**
Andrew Bajek	30	$30.00	$900.00
Elise Chiang	30	$30.00	$900.00
Giovanni Escamilla	30	$30.00	$900.00
Total × 2.5			$6,750.00

5. Conclusion and Further Resources

For this project, the power, drive, weapon, and control subsystems all worked. This meant that we were able to meet the safety, material, weight, and functionality requirements to participate in the battlebot competition. The final design demonstrates various speeds for maneuverability, low-latency via BLE connection, and an effective vertical spinner weapon for damage output.

Improvements made while developing the project include changing from STM32[8] to ESP32-S3-WROOM[4] for simpler circuit design and programming, increasing the footprint sizes of capacitors and resistors to make soldering easier, and 3D printing the body of the robot to optimize the design.

Overall, the final product created is agile, durable, and able to conduct offensive actions toward other battlebots. The project illustrates how electrical and mechanical design work together to make a functional robot.

5.1 Accomplishments

We were able to fully accomplish all of our goals and develop a working battlebot for the competition. Allowing us to fully show our engineering prowess and show off our design during the ECE 445 showcase.

We won both the bracket-type competition and the free-for-all competition, meaning our design outperformed all the other bots. We defeated four opponents during the competitions with clear victory for three of the four fights and winning another by decision.

Our victories demonstrate our success in our design and demonstrate the quality of our work and ability in design and control. Additionally, we received recognition for our wins as the best battlebot design in ECE 445 in the Spring 2026 semester.

5.2 Problem Solving

During the course of this design we ran into several issues that caused delays and forced design changes. The major ones fell into three categories: PCB layout revisions, assembly difficulties, and recurring driver chip failures.

Our PCB went through roughly four revisions over the course of the project. The original design, described in detail in our design document, was built around an STM32[8] microcontroller and required custom antenna routing. We carried this design through audits two and three, where we identified that our antenna spacing did not meet the recommended clearance from the copper ground pour and other noisy components. In response, we pivoted in our final audit to a design centered on the ESP32-S3-WROOM[4], which has an integrated antenna and allowed us to drop

the antenna subsystem entirely. This pivot also reduced points of failure and helped bring the project back in line with our original schedule. However, a footprint error on the ESP32-S3-WROOM in the final audit slipped through review and made the board unusable. With limited time remaining, we placed an emergency order for two redesigned boards: a corrected ESP32-S3 layout, which became our final design, and a revamped STM32 board (shown in Figure 12) kept as a backup in case the ESP32-S3-WROOM board had unforeseen issues. We had both in hand for the demo, but committed to the ESP32-S3-WROOM board to leave time for testing and assembly.

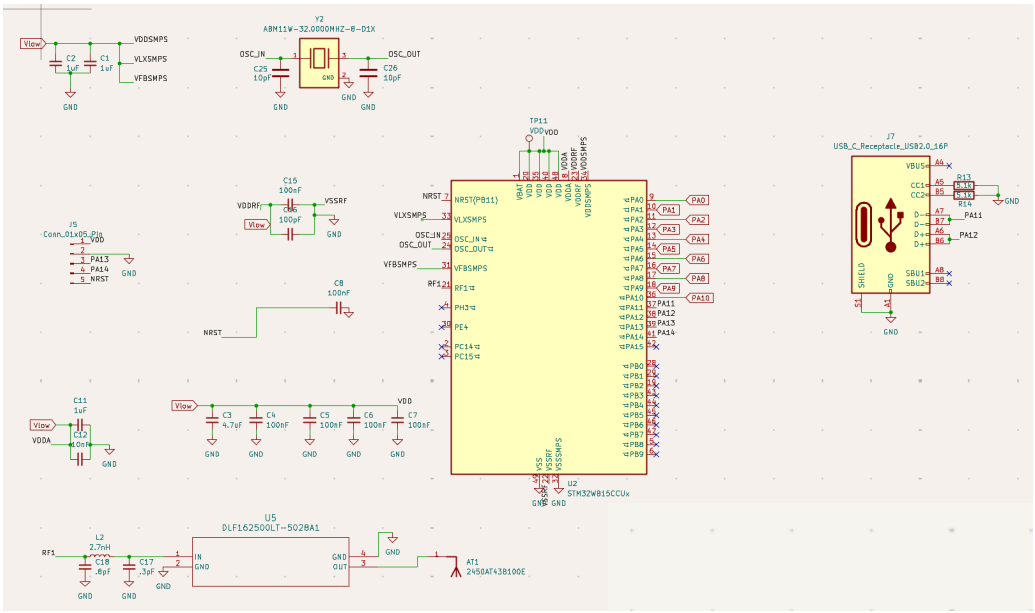

Figure 12: STM Redesign

Assembly itself was the second major source of friction. We had deliberately packed the PCB tightly to leave as much room as possible inside the chassis for batteries, motors, and wiring. The trade-off was that components ended up close enough together to make soldering and especially desoldering particularly difficult. The DRV8411[2] and TPS62172DSGT[1] pads were the most difficult to solder and had to be reworked several times during testing. Our soldering lead was eventually able to place every component cleanly, but the process took longer than we expected. We also discovered several misconnections on the board that we resolved with jumper wires rather than a respin, specifically on the programmer interface and the current-sense lines. Once these were added, the board operated as intended.

Our most persistent problem was the DRV8411[2] burning out repeatedly during testing. The failures occurred when motor current spiked above the chip's safe operating range, most often during stalls or hard impacts. Once we had the current-sense feedback path working in firmware, we were able to detect over-current conditions and assert a fault response that disabled the motor outputs before the chip was damaged. After this software-side protection was in place, we did not lose another DRV8411 for the remainder of the project.

5.3 Ethical Considerations

In order to compete in the 3D Printed Battlebots Competition, we need to comply with the ethical standards set by IEEE[10] and ACM[11]. In ACM 1.2 and IEEE 1.1, it states that avoiding harm is necessary for professional projects. To maintain this, we will test our robot in approved areas that provide protection against danger from the active weapon. We are also making sure our robot can be turned off manually as well as a shutdown via Bluetooth. We will also follow the guidelines laid out by IEEE 1.4 by not taking bribes or participating in unlawful activities. ACM 1.4, 1.6, and 1.7 mentions that we must respect privacy, confidentiality, and other people. Our project will reflect this by following the rules of the Robobrawl Competition. This will be done by not using illegal weapons or material to strengthen our robots' durability.

5.4 Future Work

5.4.1 Electrical Subsystem Improvements

- We should add circuit protection such as diodes or fuses to prevent excess current from burning up the drivers.
- We should implement a feedback control loop from our current sensors to allow them to change the speed of our drive and weapon motors to protect the system from stalling or overstressing any circuit components or motors in our system.

5.4.2 Physical Design Improvements

- The number of wall loops for the weapon should be increased to prevent attacks from chipping into the infill layers, improving the structural integrity and resistance to impact damage.
- We should choose a more compact weapon motor such that the weapon juts out more from the body, giving the bot an increased damage range.
- Repair time should be considered when designing the chassis of the robot. As it stands, replacing the weapon takes up to five minutes, which is tedious to perform many times.
- If operating at maximum speed, the robot bucks due to the fast acceleration. In the future, more of the weight should be concentrated in the front of the bot to keep it on the floor.

References

[1] Texas Instruments, “TPS6217x 3-V to 17-V, 0.5-A Step-Down Converters with DCS-Control.” [Online]. Available: https://www.ti.com/lit/ds/symlink/tps62172.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1775029635331&ref_url=https%253A%252F%252Fso.szlcsc.com%252F

[2] Texas Instruments, “DRV8411 Dual H-Bridge Motor Driver with Current Regulation.” [Online]. Available: https://www.ti.com/lit/ds/symlink/drv8411.pdf

[3] Texas instruments, “DRV8873 H-Bridge Motor Driver.” [Online]. Available: https://www.ti.com/lit/ds/symlink/drv8873.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1770836268292

[4] Espressif Systems, "ESP32-S3-WROOM-1/1U Datasheet." [Online]. Available: https://documentation.espressif.com/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf

[5] HobbyKing, “Turnigy 1000mAh 3S (11.1V) 40C Lipo Battery.” Available: https://hobbyking.com/en_us/turnigy-1000mah-3s-40c-lipo-pack-xt60.html

[6] ServoCity, “508 RPM Mini Econ Gear Motor.” [Online]. Available: https://www.servocity.com/508-rpm-mini-econ-gear-motor/

[7] Amazon, “RS-385PH-2270 Carbon Brushed Micro DC motor.” [Online]. Available: https://www.amazon.com/RS-385PH-2270-Carbon-15000RPM-RS-385PV-2270-Electric/dp/B0GNKSC5X3?th=1

[8] STM32, “STMicroelectronics, 'STM32WB15CC Datasheet.” [Online] Available: https://www.st.com/resource/en/datasheet/stm32wb15cc.pdf

[9] Bluetility, “Bluetility Github and READ ME.”[Online] Available: https://github.com/jnross/Bluetility

[10] IEEE, “IEEE Code of Ethics.” [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8

[11] ACM, “ACM Code of Ethics and Professional Conduct.” [Online]. Available: https://www.acm.org/code-of-ethics

Appendix A - Requirement and Verification Table

Subsystem	Requirement	Verification	Verification Status (Y or N)
Power	Vlow reads ~ 3.3V	Use voltmeter to check output of the buck converter	Y
	Weapon DVDD reads ~ 5V		
	Battery lasts $\geq$ 180s	Connect the battery to PCB and use the robot. Time how long it lasts	
Drive	Motors are able to move	Visual test (Motors can act independentally)	Y
	Robot is able to drive and turn		
	Current limiting works	Send current data to the computer. Record when current peaks	
Weapon	Motor is able to move	Visual test	Y
	Shuts off within 60s for safety	Disconnect bluetooth and time how long weapon takes to stop	
	Weapon is able to hit objects	Use crunched up water bottle to hit	
Control	Command latency of $\leq$ 250ms	Sent a package between PC and ESP and made our code give the time it took for the ESP to acknowledge this	Y
	Controlling the robot works at far distances	Drive robot down a long hallway	