ECE 445

Spring 2025
Design Document

Antweight Combat Robot

Team 11
Ryan Middendorf
Teodor Tchalakov

TA: Micheal Molter

Introduction

Problem

The Ant-weight 3D printed Battlebots competition hosted by Professor Gruev puts custom built remote controlled robots to battle each other in an enclosed arena till either robot is disabled making the other the winner of that round. The competition's rules follow the official National Robotics Competition guidelines [1]. The constraints defined for the competition are that the robot is less than 2 pounds in weight, is 3D printed in PET(G), ABS, or PLA(+), is remotely controlled via Bluetooth or WiFi, is equipped with a “fighting tool” for use against other robots, and has an easy manual shutdown and automatic shutdown during loss of a wireless connection. Creating a winning robot requires balancing offense and defense to enable the robot to execute power physical blows while also not becoming disabled from the blows of others. We are also challenged with building a custom PCB and control solution to survive the 2 minute matches and win the competition by disabling the other robot. With many mechanical, electrical, and software choices for making a winning robot [2], our design needs to carefully assess the tradeoffs in design and implementation.

Solution

We split our problem into designing a movement and weapon subsystem. Movement subsystems often utilize wheels or walking mechanisms [2], but we think that the use of a novel walking mechanism that “scuttles” when paired with a spinning shell “fighting tool” will provide improved stability to impacts with other robots due to each leg acting as a solid contact point compared to the wheels which are more likely to slip. The spinning shell will completely cover the robot providing 360 degree protection while also acting as a kinetic weapon with an axis of rotation located at the center of the robot that can spin up before colliding with other robots to deal damage. All extra available weight up to 2 pounds will be allocated to thickening up the defensive structure of the robot. The walking mechanism works by having 6 legs with 2 joints, one horizontally rotating joint followed by a translational joint that translates vertically. On each half of the robot, 3 legs are placed in a group where the robot can walk forward and backwards as well as rotate in place by moving 3 legs at a time. This is achieved by moving one leg on one side and the two farthest legs on the other side together and alternating the legs with the other group of 3. To control these legs, we will create a custom PCB that uses the PCA9685PW [4] servo driver to control all 12 servos, an ESP32-S3-WROOM-1-N16 [5] to communicate with a PlayStation 4 controller over Bluetooth to command the robot and control the servo driver over I2C. The ESP32 also interfaces over PWM with an DRV8313PWP [6] Mosfet driver chip to spin the shell.

Visual Aid

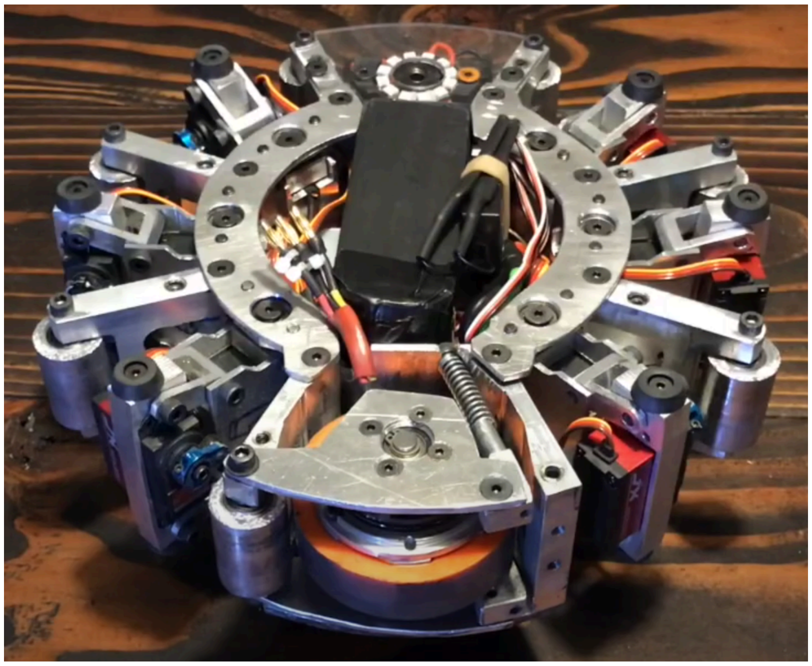

Figure 1: Sample chassis of a 30-lb walking robot

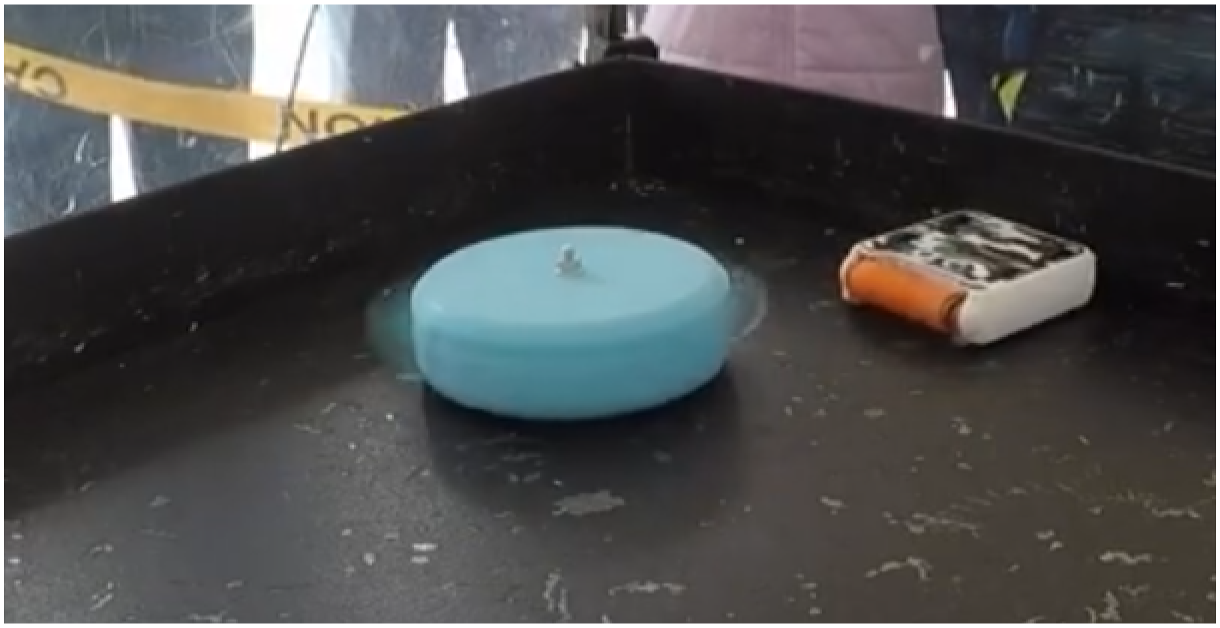

Figure 2: Sample weapon of a 2-lb gyro-walking robot

Figure 1 above features the walking mechanism I aim to replicate in this much smaller weight class. It has 6 legs arrayed in a circular pattern with each controlled by 2 servos 25 kg servos. We plan to use 9g micro servos as my robot will be significantly smaller but the concept will be very similar. Credit for figure 1 goes to dapperrogue on YouTube [7].

Figure 2 above shows Ryan's previous 2 lb bot with a plastic shell spinner. We will be using a similar shell spinner as it has good structural integrity at lower weight classes. Credit to Ryan for figure 2.

High Level Requirements

1. The robot upon powering on should be able to pair with the controller in under 15 seconds.
2. The weapon should have a maximum tip speed of at least 100 mph and should be able to recover to that speed within 10 seconds after a collision.
3. The robot must be able to move controllably for a whole match (2 minutes) and be able to cross the arena in under 10 seconds.

Design

Block Diagram

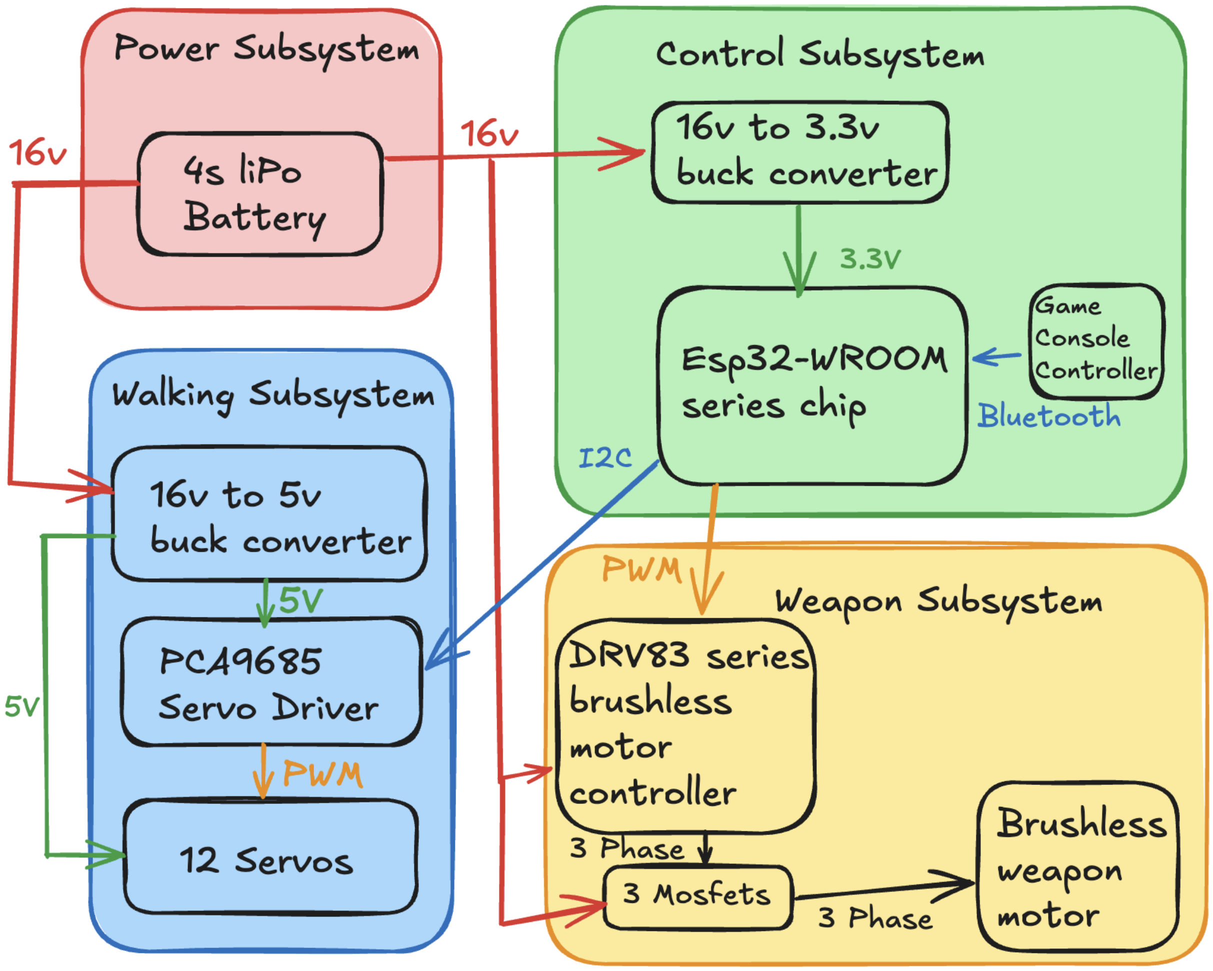

Figure 3: System Block Diagram

Physical Design

A physical diagram of the project indicating things such as mechanical dimensions or placement of sensors and actuators. The physical diagram should also be accompanied by a brief one paragraph description.

Functional Overview and Block Diagram Requirements

Power Subsystem

The power subsystem is in charge of powering all the electronics and actuators on the robot for at a minimum the duration of the 2 minute match while also providing enough power to the weapon subsystem so that it can deal significant damage to other robots. The robot will use 3 different voltages across its components, 16V, 5V, and 3.3V. Due to the 16V line being the output from the 4S lithium ion polymer (LiPo) battery, we derive our 5V and 3.3V lines using buck converters (switching regulators) because they are more power and heat efficient than other options like a linear regulator. The 16V voltages will be supplied to the DRV8313PWP MOSFET driver and MOSFETs which power and control the shell weapon motor to achieve a high tip speed. The 5V line will be used to power the PCA9685PW servo driver and all 12 S51 servos. The 3.3V line will be used to power the ESP32-WROOM-32E microcontroller. Note that the microcontroller will only be used to send data and control signals to other subsystems and will not be used to power other chips or motors on the robot. This is necessary because the ESP32 can only provide 3.3V and each pin can only output a maximum of 20ma per pin. The 16V weapon subsystem and minimum of 3.8V servos can not be powered by the 3.3V line. Even if we stepped up the voltage, the servos also require a peak current draw of 750±10ma which the ESP32 pins can only output 20ma max.

The battery comes with a male XT30PW connector which we will plug into a female XT30PW connector on the PCB. We chose a 4S 850 mAh 75C LiPo battery for our robot. We chose 4S so our robots weapon spins faster and is more powerful since the speed of a brushless motor is dependant on its kv * voltage. We chose 850 mAh to make sure we have a high enough capacity to last the full match time. We chose 75C to make sure our battery has a high enough discharge current to meet our desired 60 A peak current draw. The battery will be connected to a standard screw switch and then our pcb, allowing for an easy manual shutdown of the robot. TODO Also talk about the switch for the safety subsystem

Requirements	Verification
Provide 25 A continuous current	Operating at or under 30C discharge, 25.5 A for our battery, will guarantee it lasts the full 2 minutes of a match. To test this we can run our bot on a full battery and see how long it

	takes to fully discharge. If it's more than 2 minutes it passes.
Provide 60 A peak current	To test this we can hook our battery up to a load like a resistor and check the current with a multimeter. If the measured current is greater than 60 A (should be 63.75) it passes.
Provide a stable 5V for the servos and 3.3V for the esp32 chip	To test this we can run our servos and esp32 chip and check the voltages supplied to them with a multimeter. If it reads the correct voltage without fluctuating it passes.

Table 1: Power Subsystem Requirements and Verification

Walking Subsystem

For a walking subsystem, we chose to make a design similar in functionality to this example walking robot shown in figure 1 [1], but scaled down to our competition and 3D printed. We were looking for a design that for a walking robot is not complex to implement, stays in a stable position when not moving, and can handle the impacts of the weapon systems of other robots. By using a design where the legs are short and each step is small, the robot won't be able to topple over compared to larger legs and bigger steps. Due to the center of mass being closer to the ground and at any given time the robot will have 3 or 6 legs on the ground, the robot will be stable and resistant to blows from other robots compared to those that use two wheels for instance, where there are only 2 points of contact and maybe a castor wheel for a third.

Each leg is designed to use two joints to move two links as shown in figure 4 below. The first joint is a rotational joint connected to the robot's body which can move the entire leg towards the front and back of the robot. This joint performs the horizontal translation and rotation that the robot will experience. In the side view, the translation joint moves the last link up and down so it can dictate when a leg is touching the ground or not. The servo for the second joint is mechanically attached to the translating link such that its rotational motion will turn into translational motion. To describe how the leg will enable motion of the robot. Assuming that the leg starts with the translational joint fully to the ground and the rotational joint fully towards the robot back, we start by lifting the leg so that it is no longer in contact with the floor using the translational joint. We then rotate the leg from the backside to the front side using the rotational joint and then put the leg back down into contact with the floor using the translational joint. Lastly, we use the rotational joint to move from the front side position to the back side position. This will cause the robot to move forward because the bottom of the leg is in contact with the floor. From here we repeat the cycle.

Each leg can move individually but they must all be coordinated together to allow the robot to move forwards, backwards, and turn in place. To illustrate this, in figure 5 a top view of the robot shows what legs need to move in which direction and which should be touching the floor such that the robot moves forwards. Legs 2, 4, and 6 move together to push the robot forward while they are in contact with the ground, while at the same time, legs 1, 3, and 5 move to the front without touching the floor so they can repeat this process. As such, the groups of 3 legs alternate to move the robot forwards. To move the robot backwards, the color coding shown in the figure is flipped such that the legs that are touching the floor move towards the front to push the robot backwards. To have the robot turn in place counter clockwise (to the left relative to front), we follow the directions shown in figure 6 where now legs 2,4, and 6 pick up the robot and turn it counter clockwise while legs 1,3, and 5 get into position to do the same thing. To turn the robot clockwise, the direction of all the arrows is flipped.

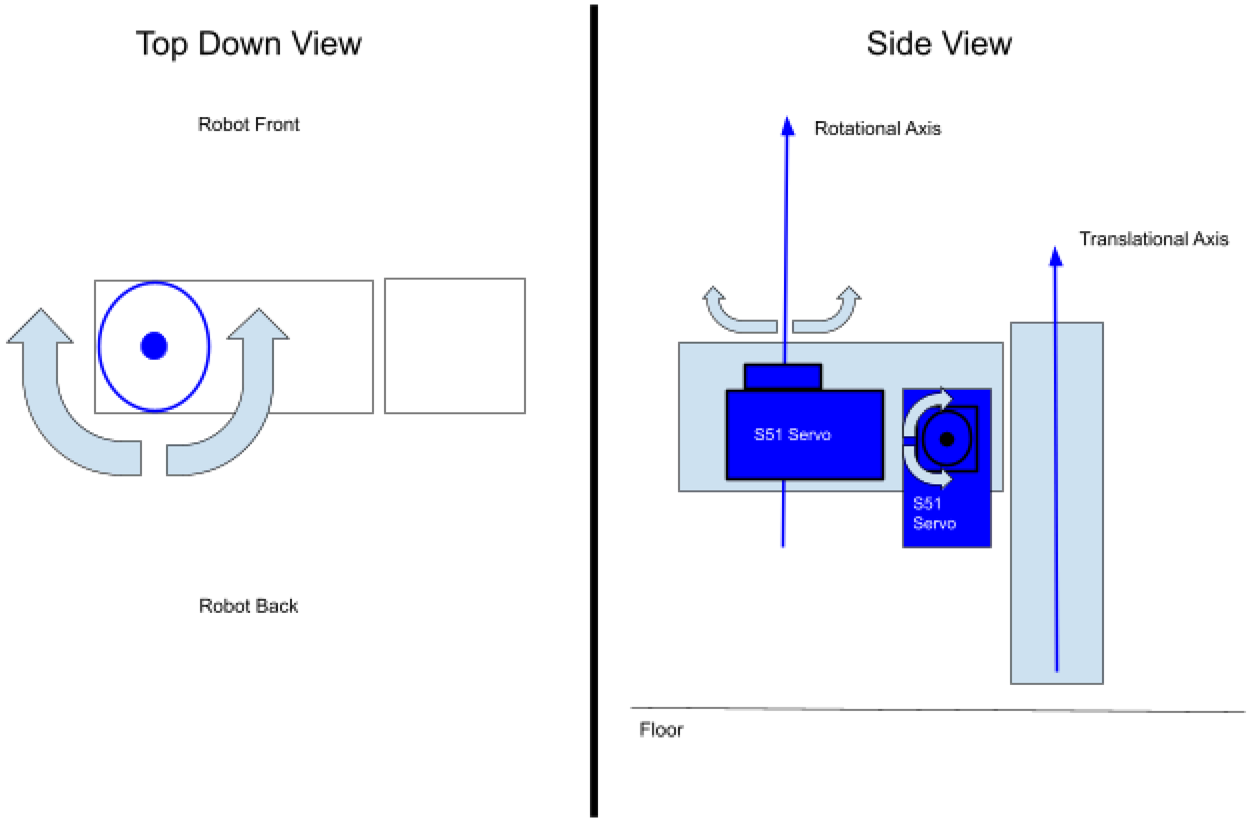

Figure 4: Top and Side View of Leg

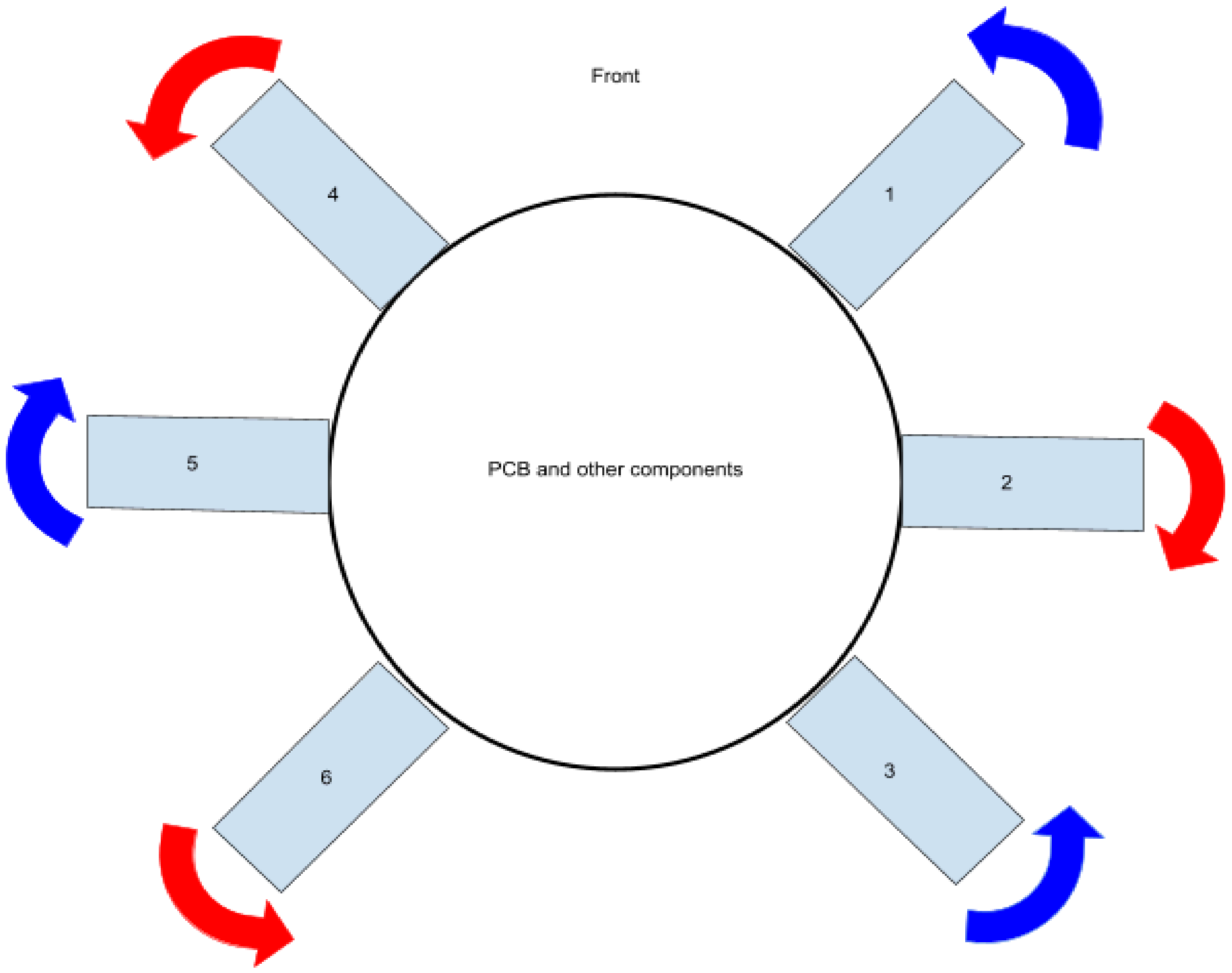

Figure 5: Top View of Walking Subsystem and Forward Walking Motion. Red means the leg is touching the floor while blue means the leg is not touching the floor.

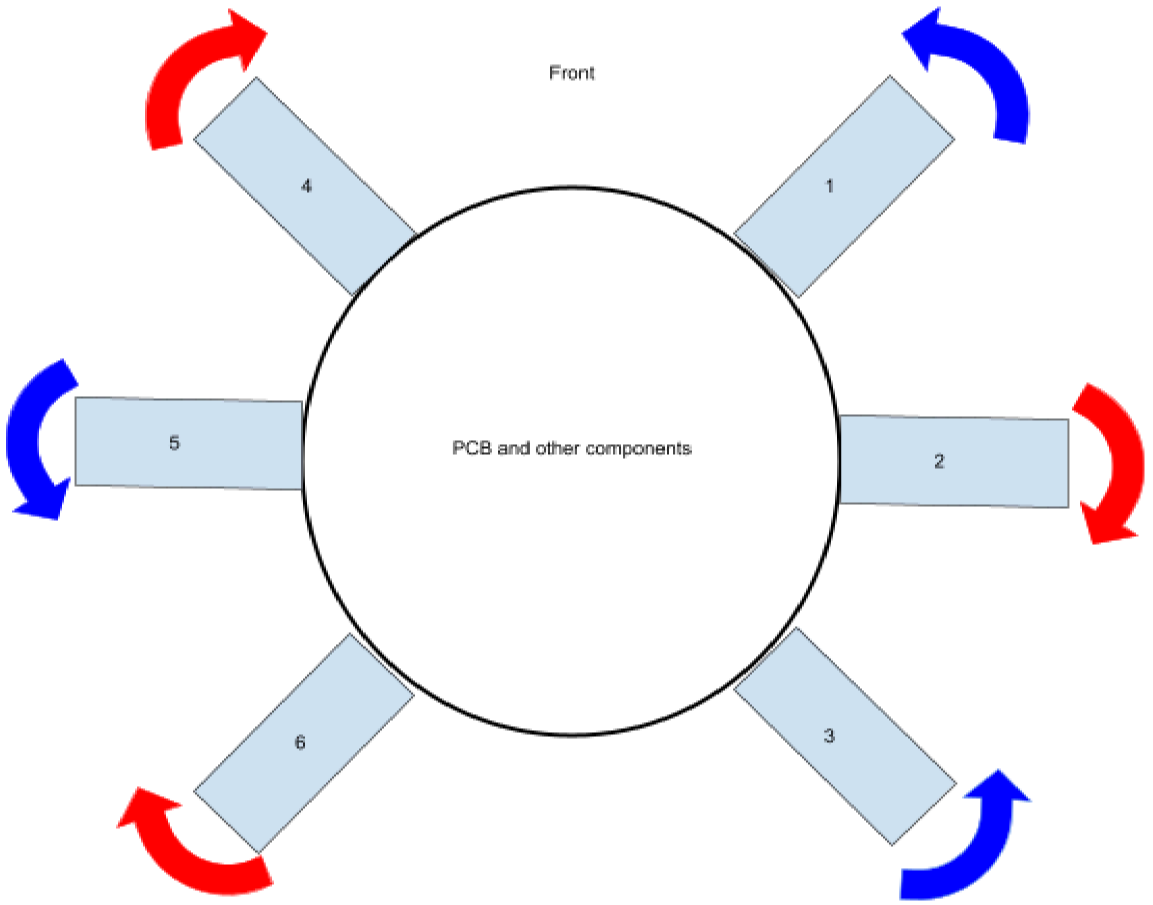

Figure 6: Top View of Walking Subsystem and Rotating Counter Clockwise in Place Motion. Red means the leg is touching the floor while blue means the leg is not touching the floor.

Above we explain how the mechanics of the walking subsystem function. To control the servos we utilize a PCA9685PW servo driver chip that receives positions over I2C from the control subsystem's ESP32. The PCA9685PW provides the PWM signals that the S51 servos need to track and position themselves to. The PCA9685PW can handle 16 channels to control 16 servos, even though we only need 12, we will let all 16 be available to plug into the ESP32

Requirements	Verification
The robot can walk forward and backwards at a speed of at least 10 cm/s	Once the robot is assembled, run the walking code, command the robot to walk 1 meter, and time it with a stopwatch. If the bot covers this distance in under 10 seconds it passes.
Must be able to lift the entire 2lb robot 3mm	Once the robot is assembled, move 3 of the legs to their lifting position and measure the gap under the robot with calipers or a ruler. If the gap is larger than 3mm it passes.

The servo driver must be able to correctly move each servo individually	Once all the servos are connected to the servo driver which is getting commands from the ESP32 chip, have each servo move a set distance sequentially. If all the servos move the correct distance in the correct order it passes.

Table 2: Walking Subsystem Requirements and Verification

- 12v to 5v buck converter to bring 12v from battery down to 5v for the servo driver
- Servo driver to take instructions from the STM32 chip and control the 12 servos so the bot has a fluid walking motion
- 12 servos for the 6 arms of the walking mechanism.

Weapon Subsystem

The weapon subsystem will have a brushless esc and the brushless motor it controls. It will also contain the physical shell. The weapon motor will connect to and spin the shell of the shell spinner. This motor will be a Flash Hobby D3536 750kv motor which will have a high enough kv rating to reach the desired tip speed of 100 mph and have enough torque to recover from impacts in under 10 seconds. The brushless esc will be a custom designed portion of our pcb using a DRV8313PWP chip and 3 MOSFETS capable of providing the current and voltage required to power the motor. The MOSFET driver and the MOSFETS will be powered directly off the battery at 16V and the driver will receive a PWM signal from the ESP32 chip in the control subsystem. The motor leads will be connected to banana plugs that get soldered directly to the pcb for ease of removability and replaceability. The shell and chassis will both have a hole that the weapon stop goes in to prevent the weapon from spinning when it's outside the arena or test box.

Requirements	Verification
Must be able to reach a tip speed of 100 mph	With the shell on the robot and in a safe testing location, spin the shell to its top speed and use a tachometer to measure its rpm. From this the tip speed can be easily calculated and if it's over 100 mph it passes.
Must recover from impacts by returning to its original speed in under 10 seconds	With the shell on the robot and in a safe testing location, spin the shell to an intermediate speed, roughly 50 mph tip speed, and have it hit a test object. Measure it with a tachometer the whole time and if it returns to its original speed before the impact within 10

	seconds it passes.
The esc must be able to provide the motor with enough current that it doesn't burn out during operation	With the shell on the robot and in a safe testing location, spin the shell to its top speed and then stop it and make sure the esc still works and was able to handle the current running through it and the back emf as the motor slowed down. It passes if the esc is still in its original condition

Table 3: Weapon Subsystem Requirements and Verification

Control Subsystem

The control subsystem contains the ESP32-WROOM-32E-N8 module [8] which will communicate using Bluetooth V4.2 with a PlayStation 4 controller to receive commands from a user and to display status LED colors on the PlayStation controller. We decided to connect the PlayStation controller directly to the ESP32 to reduce system complexity. The ESP32 will then take the controller joystick inputs to combine and convert them into walking commands to send to the walking subsystem over I2C. The ESP32 will also send a PWM signal to the MOSFET driver in the weapons subsystem which will provide a percent output of power to spin the shell. The ESP32 is powered by the 3.3V rail from the power subsystem and does not provide power to other subsystems. The 3.3V PWM and I2C signals are respectively compatible with the DRV8313PWP MOSFET driver and PCA9685PW servo driver chip. The control subsystem also contains a manual and automatic system safety stop implementation where a button on the PlayStation controller can be toggled to enable or disable the robot and if the Bluetooth connection is lost, the robot will automatically cut power to the weapon and walking subsystems.

Requirements	Verification
Robot will automatically disable all motors when the Bluetooth connection is lost.	Connect the PlayStation controller to Bluetooth. Hold the joystick forward to walk forward and press the PlayStation button to turn off the controller. With the last command sent being to walk forward, the test passes if the robot stops moving.
Will pair with the bluetooth controller in under 15 seconds	Turn the robot on and start a stopwatch at the same time. If the controller connects in under 15 seconds it passes.
Sends the correct PWM signal to the weapon subsystem allowing precise control of the weapon speed	With the robot in a safe testing location, slowly increase the weapon output on the controller while measuring the shell's speed with a tachometer. If the values on the tachometer climb synchronously with the

	commands being sent to the ESP32 it passes.

Table 4: Control Subsystem Requirements and Verification

Hardware Design

Operating Voltage and Regulation

We will need to provide a constant voltage of 3.3V to our ESP32 chip, and 5V to our servo driver chip and all 12 servos. We will be using buck converters implemented into our PCB to accomplish this. The 5V buck converter will also have to provide enough current ~7A to power all 12 servos simultaneously.

Software Design

Remote Controller Using Bluetooth

We will be using an ESP32 library to connect to our PS4 controller and correctly translate the incoming signals. Credit to styxnanda for the library [6]

Commercial Component Selection

Servo Motors

We will be using standard 9g servos for our robot because they are very common, cheap, light, and robust. Since we are making such a light robot, we have to use the lightest servos that are strong enough to lift our robot. We have decided 9g servos meet all of our requirements.

3D Printing

We will be using an Ender 3 Pro with Duramic 3D PLA+ to print our robot. We will be using an Ender 3 pro because Ryan already has one. We will be using Duramic 3D PLA+ because it is much stronger than standard PLA and has great impact resistance. Ryan has had a lot of success using it in many 1lb combat robots.

Tolerance Analysis

Weapon Subsystem Analysis

We think the weapon subsystem poses the greatest risk as it will have to be powerful enough to damage the other robots and thus will be powerful enough to harm someone. We have a requirement of reaching a tip speed in excess of 100 mph or 44.704 m/s. To accomplish this we need a motor with a high enough

KV rating. Since the entire shell of my robot will be the weapon, it will have a very large weapon diameter. We will estimate this to be 150mm and the battery is 4s or 16V.

$$\frac{44.704m/s}{0.15m \times \pi} = 94.8648RPS \times 60s/min = 5691.89RPM \times \frac{1}{16V} = 355.743KV$$

We are planning to use a 750 KV motor which is comfortably above 356 KV so hitting 100 mph tip speeds should be very achievable even if the shell diameter ends up being a bit smaller than our estimate.

Power Subsystem Analysis

The power subsystem is arguably the most important subsystem since it provides power to every other subsystem in the bot. We need to guarantee our battery is large and powerful enough to power our robot for an entire 2 minute match. The control subsystem, weapon subsystem, and walking subsystem all need to get powered simultaneously. This means our battery must provide the peak current drawn by all these subsystems combined. The ESP32 chip draws 500mA at peak. Each servo needs 550 mA for 12 servos that is 6.6A peak. The weapon motor is quite large and will be able to draw a lot of current, around 40A at peak draw. All the subsystem's max draws add up to 40 + 6.6 + 0.55 = 47.15A which is why we have chosen a 850 mAh 75C battery capable of providing 63.75A.

$$850mAh * 75C = 63.75A$$

To make sure our robot lasts the entire match, it needs to only draw the 30C equivalent from the battery.

$$\frac{60\frac{C}{min}}{2min} = 30C \qquad 850mAh * 30C = 25.5A$$

This means the weapon motor can draw an average of 25.5 - 6.6 - .55 = 18.35A over the course of a match which is plenty to keep it spinning the whole time with periodic spin ups before hits.

Cost Analysis

Description	Manufacturer	Quantity	Unit Price	Order Price	Seller Link
S51 Micro Servos	Smarza	12	$1.67	$19.99	Link
ESP32-WROOM-32E-N8 Module	Espressif Systems	1	$5.28	$5.28	Link
DRV8313PWP	Texas Instruments	1	$4.99	$4.99	Link
XT30PW	Amass	1	$0.69	$0.69	Link
68001-203HLF Header Pins	Amphenol ICC	16	$0.21	$3.36	Link

PCA9685PW, 118 Servo Driver	NXP USA Inc.	1	$2.76	$2.76	Link
LM61495Q5RPHRQ1 Buck Converter 16V to 5V	Texas Instruments	2	$6.41	$12.82	Link
NCP1117LPST33T3G Buck Converter 16V to 3.3V	Onsemi	1	$0.82	$0.82	Link
RA1113112R Power Switch	E-Switch	1	$0.71	$0.71	Link
TMK107BJ474KA-T .47μF Capacitor	Taiyo Yuden	1	$0.13	$0.13	Link
Flash Hobby D3536 750kv Motor	Samsung Electronics	3	$0.06	$0.18	Link
CL05B103KB5NNNC .01μF Capacitor	Samsung Electronics	1	$0.08	$0.08	Link
CL31A107MQHNNNE 100μF Capacitor	Samsung Electronics	1	$0.52	$0.52	Link
RES 10K OHM 1% 1/10W 0603	Yageo	2	$0.10	$0.20	Link
RES 221 OHM 1% 1/10W 0603	Yageo	12	$0.10	$1.20	Link
CSD18536KCS	Texas Instruments	3	$4.48	$13.44	Link
2171750001 USB C Connector	Molex	1	$0.75	$0.75	Link
Tattu 14.8V 4S 850mAh LiPo Battery	Tattu	1	$19.49	$19.49	Link
D3536 750KV Brushless Motor	Flash Hobby	1	$19.99	$19.99	Link
DURAMIC 3D PLA+ Filament 1.75mm 1Kg	Duramic 3D	1	$19.99	$19.99	Link
Playstation 4 Control	Sony	1	$59.99	$59.99	Link
ESP32-DevKitC-32E Development Board	Espressif Systems	1	$10.00	$10.00	Link
Servo Driver Module	AiTrip	1	$5.00	$5.00	Link
Total				$202.38	

Table 5: Cost analysis for every part on PCB and in robot.

Our project has 2 partners that we will count as getting paid for their labor. For high skill engineering labor at 40$/hour person with about 6 hours of work per person per week left with 7 weeks left gives 40 * 2.5 * 8 * 7 = $5600 per person.

Schedule

Week	Tasks
March 10th	Ryan: - Finish basic CAD design for test legs and chassis Teodor: - Finish breadboard demo software for controlling all 12 servos Everyone: - Assemble breadboard for demo - Finish PCB, pass audit, and **submit PCB second order** - Order all PCB components
March 17th	Spring Break
March 24th	Ryan: - Put together test chassis and legs - Add the breadboard to the test chassis for walking tests Teodor: - Assemble PCB - Implement Robot Walking Motion Profile on test legs and chassis Everyone: - Debug PCB - Revise the PCB design
March 31st	Ryan: - Finalize chassis and leg design - Start working on shell design Teodor: - Debug and Test Robot Walking Motion Profile Everyone: **- Submit third PCB order**
April 7th	Ryan: - Finalize shell design and any finishing touches to overall design - Assemble all the mechanical parts to guarantee integrity and review the first design Teodor: - Assemble PCB and Debug for final PCB design Everyone: **- Submit fourth PCB order** - Assemble printed robot and PCB, test running software
April 14th	Ryan: - Make any desired changes to design after initial assembly - Assemble new mechanical parts along with the PCB and electronics to create the robot Teodor:

	- Assemble PCB and Debug - Perform verification for power and control subsystems Everyone: - Test the assembled robot and make final software adjustments
April 21st	Ryan: - Make any final changes to the CAD design if issues came up in testing - If needed, assemble new robot Teodor: - Perform verification for walking and weapon subsystems Everyone: - Test full robot assembly functionality
April 28th	Ryan: - Print replacement parts for competition day - Final assembly check and make any small adjustments necessary Teodor: - Consolidate all design documents, code, and other information and prepare to put it into the final paper. Everyone: - Get ready for the competition!
May 5th	Everyone: - Demo

Table 6: Schedule for Project Progression

Ethics and Safety

Ethics

As outlined in Section I of the IEEE Code of Ethics [4], I will be disclosing any factors that could pose a risk to the public or the environment, as ensuring safety is of the utmost priority. One of the major risks our robot poses is having a lethal weapon that can harm humans. It's my responsibility to only turn on the robot during competition settings in the allowed zones or in contained testing environments, as well as guaranteeing a weapon lock is used properly.

Safety

With a dangerous robot, safety is the utmost priority. I will follow safety procedures similar to that of other competitions. First I will never power the robot when it's not inside of a safe and approved test box. This ensures that no body part of mine or anyone else is in the way of the

spinning weapon. Second, I will follow the standard protocol for turning on the robot when entering the arena/test box.
1. Place robot inside of test box
2. Turn on the transmitter/computer
3. Turn on the robot
4. Ensure robot is connected to transmitter/controller and that robot does not have any motors that are actively trying to spin
5. Remove the weapon stop
6. Close arena doors
7. Move robot to check functionality

By following these procedures, I should be able to create a safe environment. These procedures are based on NHRL (combat robotics competition) procedures [5].

Our competition spec also contains some specific safety-related rules that we must implement:

1. If WiFi or Bluetooth connection is lost between the robot and PC, the robot will automatically go into shutdown mode: it will stop moving and the fighting tool will stop rotating.
2. All electrical power to fighting tools and drive systems must have a manual disconnect that can be activated within 15 seconds without endangering the person turning it off.
3. Spinning blade must come to a full stop within 60 seconds of the power being removed using a self-contained braking system.

References

[1] "The World's First Robotics Competition," National Robotics Challenge, https://www.thenrc.org/contest-manual (accessed Mar. 6, 2025).

[2] "Antweight Robots," BattleBots Wiki, https://battlebots.fandom.com/wiki/Category:Antweight_Robots (accessed Mar. 6, 2025).

[3] P. Garnache, "Know Your Combat Robots! A Field Guide to Competition Weight Classes and Weapons," Make, https://makezine.com/article/technology/robotics/know-your-combat-robots-a-field-guide-to-competition-weight-classes-and-weapons/ (accessed Mar. 6, 2025).

[4] IEEE. "IEEE Code of Ethics." (2024), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (accessed Feb. 11, 2025).

[5] "Safety," NHRL, https://wiki.nhrl.io/wiki/index.php/Safety (accessed Feb. 11, 2025).

[6] "DS4-32," styxnanda, https://github.com/styxnanda/DS4-32 (accessed Mar. 6, 2025)

[7] "How Scuttle Walks," dapperrogue, https://www.youtube.com/watch?v=U8zYyy_4mZI (accessed Feb. 12, 2025).