

Dual Smash: Ant-Weight Battle Bot

ECE 445 Design Document, Fall 2026

Team 51

Aditya Saxena (saxena11) and Laksh Sharma (lsharma2)

Professor: Arne Fliflet

TA: Xiyue Zhu

1. Introduction

1.1 Problem and Solution

This project is our team's entry for the Fall 2026 ECE 445 Battlebots competition [1]. The goal of the competition is to build a robot that weighs less than 2 lb, is controlled wirelessly from a laptop, and can fight other robots inside an arena. Robot combat in general follows a set of shared safety and design rules, such as the SPARC ruleset [2], and these rules put a lot of weight on things like a reliable failsafe, a way to cut power quickly, and safe battery handling. So beyond just building something that can fight, the robot also has to be safe to handle and has to shut itself down if something goes wrong.

Most small combat robots use one main attack, like a spinner, a wedge, or a single lifting or hammering arm. Spinners are very effective, but they are also the most dangerous weapons to test and they put a lot of stress on a small robot. We wanted to try something different that is still effective but easier to control and safer to test. Our idea is to use two hammers, one mounted on each side of the robot, that swing horizontally. Each hammer can be fired on its own, or both can be fired at the same time in a mode we call Dual Smash. The hammers are meant to strike, push, and knock an opponent off balance while the robot stays responsive and easy to drive.

To make this work, we are building a 3D-printed robot controlled by an ESP32-S3 over WiFi. The robot uses two drive motors in a differential drive setup, so it can move forward, backward, and turn in place. Each hammer is driven by its own small gearmotor and motor driver, and each hammer has two limit switches so the firmware always knows if the hammer is at its home position or at the end of its strike. The robot also has an onboard camera that streams video to the driver's laptop, four time-of-flight distance sensors facing the front, rear, left, and right, a piezo impact sensor that detects when a hit lands, and a small speaker for sound feedback. The laptop interface shows the video feed, the four distance readings, the battery level, and the state of each hammer, which helps the driver line up a hit. Everything runs on a custom PCB powered by a 2S LiPo battery, with a fuse, a hardware motor cutoff, an external kill switch, and a firmware failsafe that disables the motors if the WiFi connection is lost.

What makes our design different from a typical ant-weight is the combination of two independent hammers with sensing. The distance sensors on the left and right sides tell the driver when an opponent is actually within reach of a hammer, so the driver is not just swinging blindly, and the impact sensor confirms when a hammer actually connects. We also took a lesson from a previous ECE 445 hammer robot, the Spring 2026 DC Hammer project [3], which used end-stop switches so the hammer motor would stop before the mechanism jammed. We are using that same idea with two switches per hammer, wired so that a broken switch wire stops the hammer instead of letting it run, along with a hardware current limit on each hammer motor.

1.2 Visual Aid

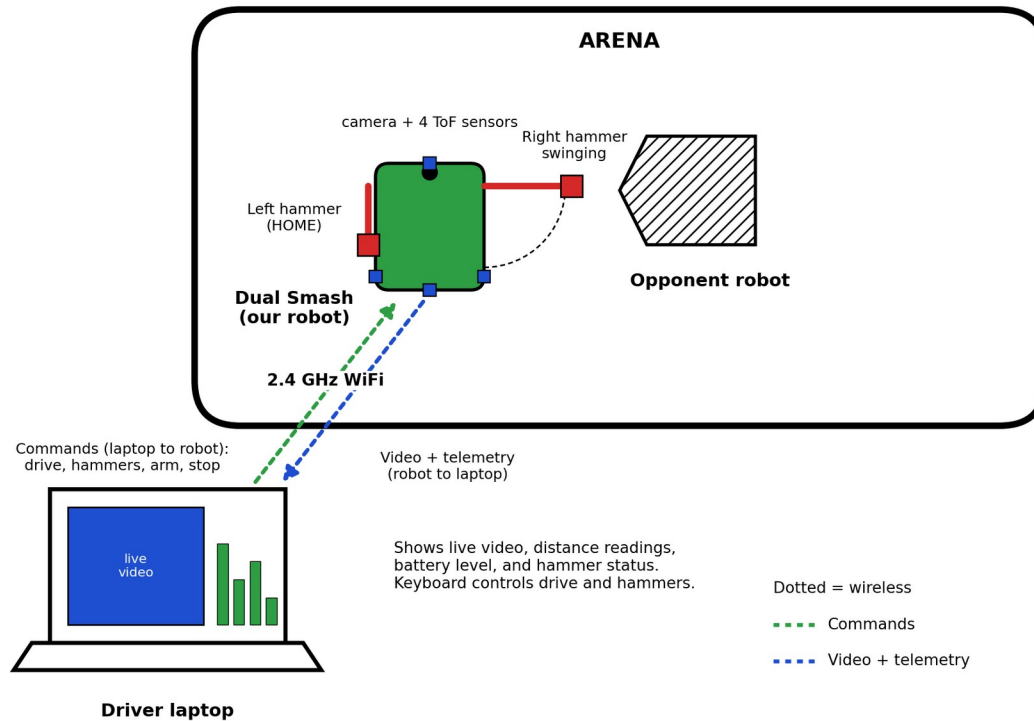


Figure 1: Visual aid of Dual Smash in use

Figure 1 shows how the robot is used during a match. The driver sits outside the arena with a laptop. The laptop connects to the robot over WiFi and sends drive and hammer commands, and the robot sends back the live video feed along with the distance readings, battery voltage, and hammer status. The driver uses the video and the side distance readings to get close to the opponent, and then fires the left hammer, the right hammer, or both hammers at once.

1.3 High-Level Requirements

To consider our project successful, the robot must meet the following requirements:

- The fully assembled robot, including the battery, PCB, camera, sensors, drivetrain, and both hammer mechanisms, weighs less than 907 g (2 lb).
- The robot disables power to all drive and hammer motors within 500 ms of losing its WiFi control connection to the laptop.
- When Dual Smash is commanded, both hammers start moving within 100 ms of each other, and each hammer can complete at least 20 strike and reset cycles in a row without a mechanical or electrical failure.
- During a continuous 2-minute run, the robot carries out drive and hammer commands with less than 200 ms of command latency while streaming video at 10 frames per

second or more and reporting all four distance sensors at 10 Hz or more, without the ESP32 resetting.

2. Design

2.1 Block Diagram

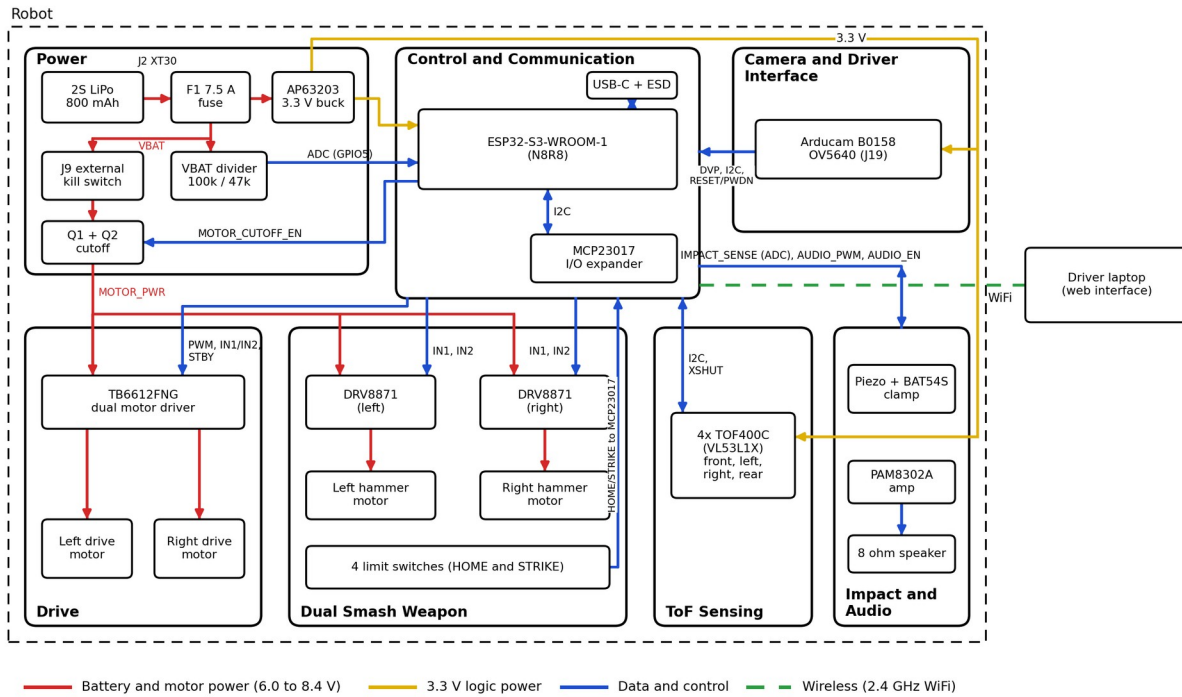


Figure 2: Dual Smash block diagram

The robot is split into seven subsystems. The Power subsystem takes the 2S LiPo battery through a 7.5 A fuse and creates two rails: a 3.3 V rail for all of the logic, and a switched MOTOR_PWR rail for the motors that can be cut off by both an external kill switch and the ESP32. The Control and Communication subsystem is the ESP32-S3, which talks to the laptop over WiFi, runs the failsafe, and controls everything else on the robot. An MCP23017 I/O expander on the I2C bus gives the ESP32 extra pins for signals that do not need to be fast. The Drive subsystem is a TB6612FNG dual motor driver and two gearmotors. The Dual Smash Weapon subsystem is two DRV8871 motor drivers, two gearmotors, and four limit switches. The Time-of-Flight (ToF) Sensing subsystem is four VL53L1X sensors on the shared I2C bus. The Camera and Driver Interface subsystem is the OV5640 camera connected to the ESP32 camera interface, along with the web-based control page that runs on the laptop. Finally, the Impact Detection and Audio Feedback subsystem is a piezo impact sensor read by the ESP32 and a small speaker driven by a PAM8302A amplifier.

2.2 Physical Design

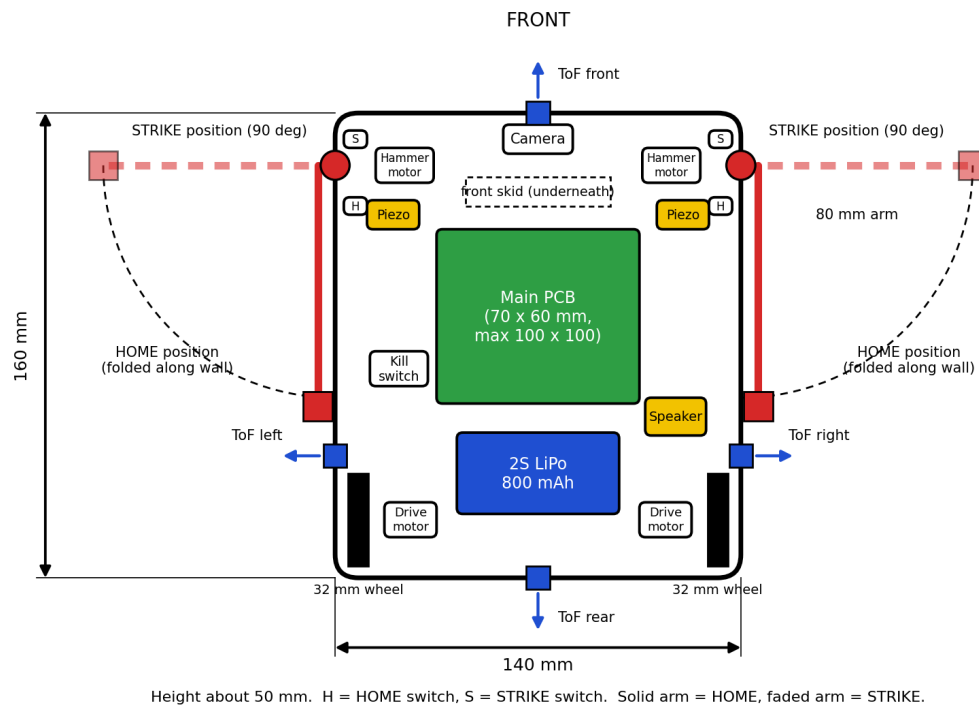


Figure 3: Physical design of Dual Smash (top view)

The chassis is a 3D-printed PETG shell that is about 160 mm long, 140 mm wide, and 50 mm tall. We chose PETG over PLA because it is tougher and less brittle, which matters for a robot that is going to get hit. The PCB sits in the middle of the chassis on standoffs, and the battery sits right behind it, low in the robot to keep the center of mass down. The PCB is limited to 100 mm by 100 mm by the course PCB order, and it will have mounting holes in each corner. The two drive motors are mounted near the back with 32 mm wheels, and the front of the robot rests on a low printed skid. The external kill switch is mounted in the top shell where it can be reached from outside the robot.

Each hammer is mounted on a vertical motor shaft near the front of its side wall. The hammer arm is about 80 mm long with a small steel head of about 20 g at the end. In the HOME position, the arm folds back along the side wall, which keeps it protected and keeps the robot narrow. When the hammer fires, the arm swings 90 degrees outward so it sweeps the area directly beside the robot. A small tab on the hammer hub presses the HOME switch at 0 degrees and the STRIKE switch at 90 degrees. The camera is mounted at the top front of the robot facing forward. The front and rear ToF sensors sit behind small windows in the middle of the front and rear walls. The left and right ToF sensors sit near the back of the side walls, behind where the folded hammer arms end, so the arms do not block them when they are at HOME. The piezo impact sensor is glued to the inside of the chassis near the hammer pivots so it picks up the vibration from a hit, and the speaker sits behind a small grille in the top shell.

Table 1 shows our weight budget. Since staying under 2 lb is one of our high-level requirements, we will weigh each part as it arrives and replace these estimates with measured values. The printed chassis is the largest and least certain number, so we will weigh the first printed chassis before printing the final one.

Table 1: Weight Budget

Component	Estimated Mass (g)
3D-printed chassis and armor (PETG)	330
Hammer arms and steel heads (2)	70
Drive gearmotors (2)	20
Hammer gearmotors (2)	20
Motor brackets (4)	6
Wheels (2)	8
Tattu 2S 800 mAh LiPo battery	45
Main PCB with components	30
Camera breakout	10
ToF breakout boards (4)	8
Limit switches (4)	6
Speaker and piezo element	5
Fuse and fuse holder	3
Kill switch, connectors, and wiring	30
Fasteners and inserts	30
Total	621
Limit	907
Margin	286 (about 32%)

2.3 Subsystems

2.3.1 Power Subsystem

The Power subsystem supplies every part of the robot from one 2S LiPo battery. A 2S pack is 6.0 V when empty and 8.4 V when fully charged, with a nominal voltage of 7.4 V. We picked a Tattu 800 mAh 2S 45C pack (TA-45C-800-2S1P-XT30) [4], which can supply far more current than our worst case (see Section 2.5). The battery plugs into J2, a PCB-mounted Amass XT30PW-M connector rated for 15 A continuous [5]. The XT30 connector is keyed, and J2 pin 1 is battery positive and pin 2 is ground, so the battery cannot be plugged in backward.

This subsystem supports high-level requirements 2 and 4. The hardware motor cutoff is what physically removes motor power during a failsafe, and the 3.3 V rail has to stay up under worst-case load so the ESP32 does not reset during a match.

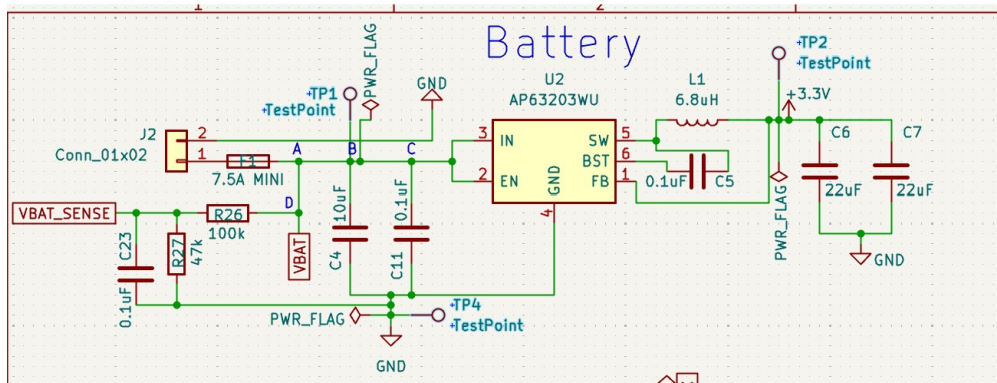
Right after J2, the battery goes through F1, a 7.5 A MINI blade fuse in a Keystone 3568 holder. The fused node is called VBAT, and everything on the robot is powered from it. The fuse protects the battery and the wiring if anything on the board or in the motor wiring shorts. Our worst-case running current is about 5.4 A (Section 2.5), so a 7.5 A fuse will not blow during normal use but will open on a real short.

The 3.3 V rail comes from an AP63203WU buck converter [6]. This part takes an input from 3.8 V to 32 V, has a fixed 3.3 V output, and can supply up to 2 A. We chose a buck converter over a linear regulator because the drop from 7.4 V to 3.3 V is large. A linear regulator supplying our roughly 0.8 A worst-case logic load would have to burn $(7.4 \text{ V} - 3.3 \text{ V}) \times 0.8 \text{ A}$, which is about 3.3 W, as heat. That is too much for a small sealed robot. The buck converter only needs a few external parts: input capacitors (10 μF and 0.1 μF), a Würth 74438357068 6.8 μH inductor, a 0.1 μF bootstrap capacitor, and two 22 μF output capacitors, following the datasheet application circuit [6]. The buck runs straight off VBAT, not off the motor rail, so the ESP32 stays powered even when the motors are cut off.

The MOTOR_PWR rail feeds all three motor driver chips. To get there, VBAT goes out through connector J9 to an external manual kill switch mounted on the chassis, comes back through J9, and then goes through Q1, a DMPH3010LK3-13 P-channel MOSFET used as a high-side switch [7]. Q1's gate is pulled up to its source through R12, so Q1 is off by default. When the ESP32 drives MOTOR_CUTOFF_EN (GPIO47) high, the 2N7002 MOSFET Q2 turns on and pulls Q1's gate down, which turns Q1 on. Q2's gate has its own pull-down resistor to ground. This means that if the ESP32 is booting, resetting, crashed, or unpowered, MOTOR_CUTOFF_EN floats and the pull-down keeps the motors off. We think this default-off behavior is the most important safety feature on the board. Each motor driver has its own local bulk and bypass capacitors, including a 47 μF capacitor at each DRV8871.

We also added a battery voltage sensing divider. VBAT goes through a 100 k Ω and 47 k Ω resistor divider to VBAT_SENSE on GPIO5, with a 0.1 μF filter capacitor (C23). At 8.4 V, the ADC sees $8.4 \times 47 / 147$, which is about 2.69 V, so it stays inside the ESP32 ADC range with its highest attenuation setting. The firmware uses this reading to show the battery level on the laptop, warn the driver when the battery is getting low, block the hammers when the battery is too low for the hammer drivers (Section 2.5), and scale the motor PWM so the 6 V drive motors never see more than about 6 V on average.

(a) Battery input, fuse, 3.3 V regulator, and battery sense



(b) Motor power cutoff

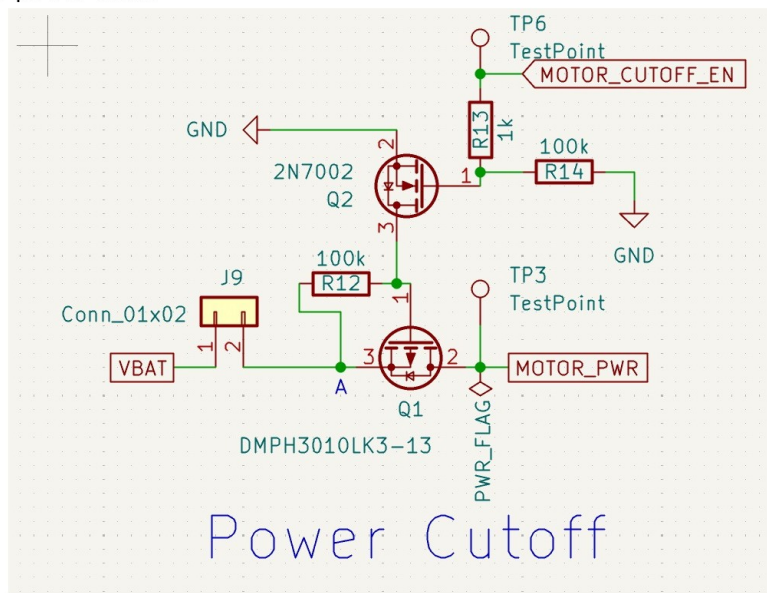


Figure 4: Power subsystem schematic: (a) battery input, fuse, 3.3 V regulator, and battery sense; (b) motor power cutoff

Table 2: Power Subsystem Requirements and Verification

Requirements	Verification
The 3.3 V rail stays within 3.3 V $\pm 5\%$ (3.135 V to 3.465 V) for any battery input from 6.0 V to 8.4 V and any load from 0 A to 1.0 A.	1) Power the board from a bench supply instead of the battery. 2) Connect an electronic load to the 3.3 V rail. 3) Set the input to 6.0 V, 7.4 V, and 8.4 V, and at each input step the load through 0 A, 0.5 A, and 1.0 A. 4) Measure the rail with a multimeter each time and confirm every reading is between 3.135 V and 3.465 V. 5) Record all nine readings in a table of input voltage, load current, and output voltage.
Q1 turns off within 10 ms of MOTOR_CUTOFF_EN going low, and stays off whenever the ESP32 is held in reset.	1) Power the board at 7.4 V and load a test build that keeps the drive PWM running after a cutoff. 2) Put the robot on a stand and spin the drive motors at half throttle.

Requirements	Verification
	3) Probe MOTOR_CUTOFF_EN at TP6 and Q1's gate-to-source voltage on an oscilloscope. 4) Command the cutoff and confirm the gate-to-source voltage drops below 1 V in magnitude within 10 ms, and that the motors stop. 5) Hold the reset button and confirm the gate-to-source voltage stays below 1 V and no motor spins. 6) Repeat 5 times and record the turn-off time for each trial.
Opening the external kill switch removes motor power no matter what the firmware is doing.	1) Spin the drive motors on a stand at half throttle. 2) Open the kill switch. 3) Confirm the motors stop and MOTOR_PWR at TP3 falls below 0.5 V within 2 seconds on an oscilloscope. 4) Repeat 5 times and save one scope capture for the report.
The ESP32 measures battery voltage within ± 0.15 V of the true value from 6.0 V to 8.4 V.	1) Power the board from a bench supply. 2) Set it to 6.0 V, 6.8 V, 7.4 V, and 8.4 V. 3) Compare the voltage printed over UART (J18) to a multimeter reading at J2 and confirm the difference is 0.15 V or less each time. 4) Record the true and measured values and the error in a table.
With the battery at a resting voltage of 7.4 V and all four motors stalled, MOTOR_PWR stays at or above 6.5 V.	1) Charge a battery to a resting voltage of 7.4 V. 2) Block both wheels and both hammer arms so they cannot move. 3) Command full drive in both motors and a strike on both hammers at the same time. 4) Capture MOTOR_PWR at TP3 on an oscilloscope and confirm the lowest point stays at or above 6.5 V. 5) Save the scope capture and report the minimum voltage.

2.3.2 Control and Communication Subsystem

The Control and Communication subsystem is the brain of the robot. It uses an ESP32-S3-WROOM-1-N8R8 module [8], which has 8 MB of flash and 8 MB of PSRAM. We picked the ESP32-S3 because it has built-in WiFi, a dedicated camera interface, native USB for programming, and enough processing power to stream video and run the control loop at the same time on its two cores. We chose a version with PSRAM because the camera driver needs it for frame buffers. Our board includes the standard support circuit for the module: a pull-up resistor and capacitor on EN for a clean power-on reset, reset and boot buttons, and a USB-C port (GCT USB4085) with a Würth WE-TVS ESD protection array, 22 Ω series resistors on the data lines, and 5.1 k Ω pull-downs on the CC pins. A 100 k Ω and 150 k Ω divider on USB VBUS lets the firmware detect when USB is plugged in, and UART0 is broken out to header J18 for debugging. This subsystem directly supports high-level requirements 2 and 4, since it runs the WiFi failsafe and handles all of the command, video, and telemetry traffic.

The ESP32 runs as a WiFi access point, and the laptop connects to it directly. Control commands and telemetry go over a WebSocket connection, and the video goes over a separate HTTP stream. The laptop sends a control packet every 20 ms (50 Hz). Each packet has a

sequence number, the left and right drive values, and the hammer command. If the robot does not receive a valid control packet for 300 ms, the firmware treats the connection as lost. It sets every motor PWM to zero, sets all hammer driver inputs low, and drives MOTOR_CUTOFF_EN low, which physically disconnects MOTOR_PWR. The robot then waits for the laptop to reconnect and for the driver to re-arm it. We also enable the ESP32 task watchdog, so if the firmware ever hangs, the chip resets, and the pull-down on Q2 cuts motor power on its own.

The ESP32 has a lot of pins, but the camera alone takes 12 of them, so we use an MCP23017 I/O expander [9] on the I2C bus for signals that do not need to be fast. These are the four ToF XSHUT pins, the camera reset and power-down pins, the drive driver's standby and direction pins, the four hammer limit switches, and the audio amplifier enable. The MCP23017 datasheet specifies GPA7 and GPB7 as output-only pins [9], so we only use those two pins for outputs (AUDIO_EN and CAM_RESET_N). The MCP23017 interrupt outputs are not used. Instead, the control loop reads the MCP23017 inputs every 10 ms, which is fast enough for our 20 ms limit switch requirement. Anything that needs to be fast or analog, like PWM signals, the hammer driver inputs, the impact sensor, and the battery voltage, goes straight to the ESP32. Table 3 shows the full pin assignment. All analog inputs use ADC1 pins, because ADC2 on the ESP32 cannot be used while WiFi is running.

Table 3: ESP32-S3 and MCP23017 Pin Assignment

Pin	Signal	Notes
GPIO1, GPIO2	DRIVE_L_PWM, DRIVE_R_PWM	TB6612FNG PWMA, PWMB
GPIO4	IMPACT_SENSE	ADC1, piezo impact sensor
GPIO5	VBAT_SENSE	ADC1, battery divider
GPIO6, GPIO7	CAM_VSYNC, CAM_HREF	Camera sync
GPIO8, GPIO9	I2C_SDA, I2C_SCL	Shared I2C bus, 4.7 k Ω pull-ups (R7, R8)
GPIO10 to GPIO18, GPIO48	Camera data and clocks	D0 to D7, PCLK, XCLK
GPIO19, GPIO20	USB_D-, USB_D+	Native USB
GPIO21	USB_VBUS_SENSE	USB detect
GPIO38, GPIO39	HAMMER_L_IN1, HAMMER_L_IN2	Left DRV8871 (U5)
GPIO40, GPIO41	HAMMER_R_IN1, HAMMER_R_IN2	Right DRV8871 (U6)
GPIO42	AUDIO_PWM	PAM8302A input
GPIO47	MOTOR_CUTOFF_EN	Q2 gate, pull-down to ground
TXD0, RXD0	UART	Debug header J18
GPIO3, GPIO45, GPIO46	Not connected	Strapping pins left unused
GPIO35 to GPIO37	Not connected	Reserved by octal PSRAM

Pin	Signal	Notes
MCP GPA0	DRIVE_STBY	TB6612FNG standby
MCP GPA1 to GPA4	DRIVE_L_IN1, DRIVE_L_IN2, DRIVE_R_IN1, DRIVE_R_IN2	Drive direction
MCP GPA5, GPA6	HAMMER_R_HOME, HAMMER_R_STRIKE	Right hammer limit switches
MCP GPA7	AUDIO_EN	PAM8302A enable (GPA7 is output-only)
MCP GPB0 to GPB3	TOF_FRONT, LEFT, RIGHT, REAR XSHUT	ToF address setup
MCP GPB4, GPB5	HAMMER_L_STRIKE, CAM_PWDN	Left STRIKE switch, camera power-down
MCP GPB6, GPB7	HAMMER_L_HOME, CAM_RESET_N	Left HOME switch, camera reset (GPB7 is output-only)

Table 4: Control and Communication Subsystem Requirements and Verification

Requirements	Verification
The firmware disables all motor outputs and turns off MOTOR_PWR within 500 ms of the last valid control packet.	1) Load a test build that prints a timestamp over UART (J18) for every received control packet and for the moment it drives MOTOR_CUTOFF_EN low. 2) Put the robot on a stand and drive it at half throttle. 3) Probe TP6 (MOTOR_CUTOFF_EN) and TP3 (MOTOR_PWR) on an oscilloscope. 4) Turn off WiFi on the laptop. 5) From the UART log, compute the time from the last packet to MOTOR_CUTOFF_EN going low, and add the Q1 turn-off time measured on the scope. Confirm the total is under 500 ms. 6) Repeat 10 times and record the minimum, maximum, and mean delay.
All motor outputs stay off from power-up until the laptop connects and the driver arms the robot.	1) Probe both drive motor outputs and both hammer motor outputs on an oscilloscope. 2) Power cycle the robot 10 times without connecting the laptop. 3) Confirm no motor output shows any pulse during the first 10 seconds after each power-up. 4) Record a pass or fail for each of the 10 trials.
The robot processes at least 45 control packets per second while video and ToF data are streaming.	1) Connect the laptop and start the video stream. 2) Have the firmware count received control packets over 60 seconds and print the total over UART. 3) Confirm the count is at least 2,700. 4) Repeat 3 times and record each count.
If the firmware hangs, the watchdog resets the ESP32 and motor power turns off within 2	1) Load a test build with a hidden command that puts the main loop into an infinite loop. 2) Drive the robot on a stand and send that command.

Requirements	Verification
seconds.	3) Probe TP6 and TP3 on an oscilloscope and confirm MOTOR_CUTOFF_EN goes low and the motors stop within 2 seconds. 4) Repeat 3 times and record the time for each trial.

2.3.3 Drive Subsystem

The Drive subsystem moves the robot. It uses two E-S Motor 6 V 50:1 LP micro metal gearmotors [10] in a differential drive setup, driven by a TB6612FNG dual H-bridge driver [11]. At 6 V these motors spin at about 300 RPM with no load and stall at about 0.36 A. With 32 mm wheels, the top speed is about $300 / 60 \times \pi \times 0.032$ m, which is roughly 0.50 m/s with no load. We chose these low-power motors because they are light, inexpensive, and draw very little current, which leaves more of the battery's current for the hammers. The tradeoff is a slower robot, but in a small arena, controllability from a video feed matters more to us than top speed. The motors connect to the board through 2-pin JST-XH connectors J3 and J4. This subsystem supports high-level requirement 4, since the robot has to respond to drive commands reliably and with low latency for a full match.

The TB6612FNG takes a motor supply up to 13.5 V, so it handles our 8.4 V battery with margin, and it accepts 3.3 V logic directly from the ESP32 and MCP23017. It can supply 1.2 A continuously and 3.2 A peak per channel, which is more than three times the stall current of our drive motors, and it has built-in thermal shutdown. The ESP32 drives the two PWM inputs on GPIO1 and GPIO2 at 20 kHz, which is above human hearing, and the MCP23017 sets the direction pins and the standby pin. Turning in place is done by spinning the two motors in opposite directions.

Since these are 6 V motors and the battery can be as high as 8.4 V, the firmware limits the maximum PWM duty cycle based on the measured battery voltage. The maximum duty cycle is 6.0 V divided by VBAT, so at 8.4 V the duty cycle is capped at about 71%, and at 6.6 V it is capped at about 91%. This keeps the average motor voltage at or below 6 V, which protects the motors.

Table 5: Drive Subsystem Requirements and Verification

Requirements	Verification
The robot moves forward, backward, turns left, and turns right correctly in response to laptop commands.	1) Place the robot in the arena or on the lab floor. 2) Send each of the four commands 20 times. 3) Confirm the robot moves in the correct direction all 80 times. 4) Record the number of correct responses for each command.
The average voltage across each drive motor at full throttle stays at or below 6.3 V for any battery voltage from 6.6 V to 8.4 V.	1) Power the robot from a bench supply with the wheels off the ground. 2) Set the supply to 6.6 V, 7.4 V, and 8.4 V. 3) At full forward throttle, measure the DC voltage across each motor with a multimeter.

Requirements	Verification
	4) Confirm every reading is 6.3 V or less. 5) Record all six readings in a table.
The robot reaches a straight-line speed of at least 0.35 m/s on the arena floor.	1) Mark a 2 m straight section on the floor. 2) Drive at full throttle with a running start and time the 2 m section with a phone video. 3) Confirm the time is 5.7 seconds or less. 4) Repeat 3 times and record each time and the calculated speed.

2.3.4 Dual Smash Weapon Subsystem

The weapon subsystem is the main feature of our robot. Each side hammer is driven by its own E-S Motor 6 V 210:1 HP micro metal gearmotor [12]. At 6 V this motor spins at about 150 RPM with no load, has a stall torque of about 3.4 kg·cm (0.33 N·m), and has a stall current of about 1.6 A. We chose a high gear ratio so the hammer has plenty of torque to push through an opponent, while the motor itself stays small and light. We looked at using hobby servos instead, since a few are available in the ECE 445 cabinet, but two servos would weigh about 120 g, need their own 6 V regulator, and leave very little hardware for us to design. This subsystem is what high-level requirement 3 tests: both hammers must start within 100 ms of each other during Dual Smash, and each must survive 20 strike and reset cycles in a row.

Each hammer motor has its own DRV8871 H-bridge driver [13]: U5 for the left hammer and U6 for the right hammer. The DRV8871 runs from 6.5 V to 45 V, can handle up to 3.6 A peak, accepts 3.3 V logic, and has built-in undervoltage lockout, overcurrent protection, and thermal shutdown. It is controlled with two inputs, IN1 and IN2. Driving one input with a PWM signal while the other is held low spins the motor in one direction at a speed set by the duty cycle, swapping them reverses it, setting both high brakes the motor, and setting both low lets it coast. The left hammer inputs come from GPIO38 and GPIO39, and the right hammer inputs come from GPIO40 and GPIO41. Both inputs have internal pull-down resistors, so if the ESP32 resets, the driver defaults to coasting instead of driving the hammer. The motors connect through 2-pin JST-XH connectors J10 and J11, and each driver has a 47 uF bulk capacitor on its supply.

The most useful feature of the DRV8871 for us is its built-in current limit. A resistor from the ILIM pin to ground sets the current at which the driver starts chopping the motor current. We use 32 kΩ for both R17 and R18, which TI specifies as a 2 A limit [13]. This means that even if a hammer jams against an opponent or the firmware has a bug, the driver will not let that motor pull much more than 2 A. Since the hammer motors are rated for 6 V and the battery can be as high as 8.4 V, the firmware caps the hammer PWM duty cycle the same way it does for the drive motors, at about 6.0 V divided by VBAT. This keeps the average voltage across each hammer motor at or below 6 V, and the current limit protects the motor and battery if the hammer gets stuck.

We also looked at the DRV8874 as an alternative [14]. It is a stronger part on paper: it runs down to 4.5 V, handles more current, and has a current-sense output that the ESP32 could read. We decided to keep the DRV8871 because it already covers our current needs with margin, it has a simple two-pin interface, and its hardware current limit protects the motors without needing any firmware. Its one weakness for us is the 6.5 V minimum supply, which is close to the bottom of a 2S battery's range. We handle that with a low-battery rule in the firmware, which we analyze in Section 2.5.

Each hammer has two Omron D2F-L-D3 lever microswitches [15], HOME at 0 degrees and STRIKE at 90 degrees. They connect through 2-pin JST-XH connectors J12 and J13 (left HOME and STRIKE) and J14 and J15 (right HOME and STRIKE), and all four go to the MCP23017. At each switch, COM is wired to ground and the normally closed contact is wired to the signal line, which has a 10 k Ω pull-up to 3.3 V. So an unpressed switch reads LOW, and when the switch is actuated, the normally closed contact opens and the input is pulled HIGH. We chose this wiring on purpose: if a switch wire breaks or comes unplugged, the signal also reads HIGH, and the firmware treats that the same as reaching the end of travel, so the hammer stops instead of driving into the mechanism. If both switches on one hammer ever read HIGH at the same time, the firmware knows something is wrong and reports a switch fault.

When the driver fires a hammer, the firmware drives the motor forward until the STRIKE switch is actuated, then brakes, then drives the motor backward until the HOME switch is actuated, and then brakes again. After the hammer returns home, there is a 250 ms cooldown before that hammer can fire again. If a switch is not reached within a timeout (300 ms for the strike and 400 ms for the return), the firmware brakes that hammer and reports a fault to the laptop, since a timeout means the hammer is blocked or jammed. The firmware reads the switches every 10 ms, and reading the MCP23017 over I2C at 400 kHz takes well under 1 ms, so the hammer is braked within about 11 ms of a switch being actuated in the worst case. This meets our 20 ms target. At full speed (15.7 rad/s), the hammer can travel about 10 degrees in that time, plus a little more while the brake stops it, so we will mount the STRIKE switch tab a few degrees before the 90 degree end point and leave room in the mechanism for this overtravel.

Dual Smash is a single command that starts both hammers in the same function call, so both hammer drivers turn on within a few microseconds of each other. This easily meets our 100 ms requirement.

To get an idea of hammer speed, we modeled each hammer as a 20 g head at the end of an 80 mm arm, plus an 8 g arm. The moment of inertia is about:

$$J = 0.02 \times 0.08^2 + (1/3) \times 0.008 \times 0.08^2 \approx 1.45 \times 10^{-4} \text{ kg}\cdot\text{m}^2$$

We used a linear motor model, where torque drops from the stall torque at zero speed to zero at the no-load speed. The no-load speed is 150 RPM, which is 15.7 rad/s, and the stall torque is 0.33 N·m. This gives a mechanical time constant of:

$$\tau = J \times \omega_0 / T_{\text{stall}} = 1.45 \times 10^{-4} \times 15.7 / 0.33 \approx 6.9 \text{ ms}$$

So the hammer is nearly at full speed within about 7 ms. A 90 degree swing (1.57 rad) then takes about:

$$t \approx \theta / \omega_0 + \tau = 1.57 / 15.7 + 0.0069 \approx 0.107 \text{ s}$$

This is about 107 ms. Since the firmware caps the hammer's average voltage at about 6 V, this calculation matches how the motor is actually driven. Gearbox friction, PWM losses, and the real mass of the arm will slow it down somewhat, so we set our requirement at 150 ms and will confirm it on the real mechanism.

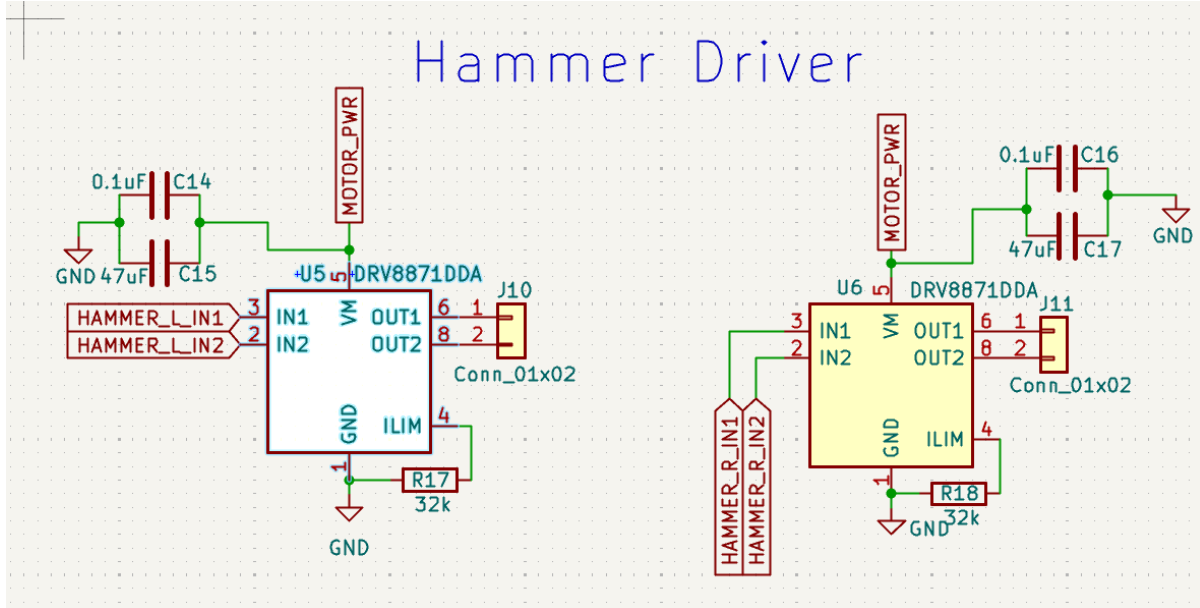


Figure 5: Hammer driver schematic (left hammer)

Table 6: Dual Smash Weapon Subsystem Requirements and Verification

Requirements	Verification
Each hammer moves from HOME to STRIKE in 150 ms or less.	1) Probe the HOME and STRIKE switch signals for one hammer on an oscilloscope. 2) Fire the hammer from the laptop. 3) Measure the time from the HOME signal going LOW (released) to the STRIKE signal going HIGH (actuated) and confirm it is 150 ms or less. 4) Repeat 5 times for each hammer and record the minimum, maximum, and mean time.
The hammer motor is braked within 20 ms of its STRIKE or HOME switch being actuated.	1) Probe the STRIKE switch signal and both hammer motor outputs on an oscilloscope. 2) Fire the hammer. 3) Measure the time from the switch signal going HIGH to the motor outputs going into brake, and confirm it is under 20 ms. 4) Repeat using the HOME switch on the return. 5) Repeat 5 times for each switch and record the maximum delay.
Each hammer motor's current is	1) Clamp one hammer arm so it cannot move.

Requirements	Verification
limited to 2.3 A or less, even when the hammer is blocked.	2) Power the robot from a bench supply at 8.4 V. 3) Command a strike and measure the motor current with a current probe. 4) Confirm the current stays at or below 2.3 A (the 2 A setting plus margin for part tolerance), and confirm the firmware stops the hammer after the 300 ms timeout and reports a fault. 5) Repeat for the other hammer and save a scope capture of each.
If a limit switch wire is disconnected, the firmware treats that switch as actuated and does not drive the hammer into the mechanism.	1) With the hammer at HOME, unplug its STRIKE switch connector. 2) Fire the hammer. 3) Confirm the hammer brakes almost immediately instead of swinging, and that the laptop shows a fault. 4) Repeat for the other hammer and record a pass or fail.
When Dual Smash is commanded, both hammers start moving within 100 ms of each other.	1) Probe both hammer motor outputs on an oscilloscope. 2) Command Dual Smash 10 times. 3) Measure the time between the two motor outputs turning on and confirm it is under 100 ms every time. 4) Record all 10 delays.
Each hammer completes at least 20 strike and reset cycles in a row without failure.	1) Fire the left hammer 20 times in a row, waiting for each cooldown. 2) Confirm the laptop shows all 20 cycles completing with no fault. 3) Repeat for the right hammer and for Dual Smash. 4) Record the number of completed cycles for each test.

2.3.5 Time-of-Flight Sensing Subsystem

This subsystem gives the driver distance information around the robot. It uses four VL53L1X time-of-flight sensors [16] on ACEIRMC TOF400C breakout boards [17], one facing each side of the robot. Each board has VIN, GND, SDA, SCL, INT, and SHUT pins. The boards connect through 5-pin JST-XH connectors J5 to J8, with pin 1 as 3.3 V, pin 2 as ground, pin 3 as SDA, pin 4 as SCL, and pin 5 as XSHUT. We leave INT unconnected because the firmware reads the sensors on a timer. The left and right sensors are the most important, because they tell the driver when an opponent is inside the 80 mm reach of a hammer. The front and rear sensors help with lining up and with knowing how close the robot is to the arena wall. This subsystem supports high-level requirement 4, which requires all four sensors to report at 10 Hz or more during a full match.

All four sensors share the I2C bus with the camera and the MCP23017. The problem is that every VL53L1X starts up with the same I2C address, 0x29. To fix this, each sensor's XSHUT pin is connected to the MCP23017 (GPB0 to GPB3). At startup, the firmware holds all four sensors in shutdown by driving XSHUT low, then releases them one at a time and gives each one a new address (0x30, 0x31, 0x32, and 0x33). To release a sensor, the firmware switches its XSHUT pin to an input so the breakout board's own pull-up enables it, rather than driving the pin high. After that, every device on the bus has its own address: 0x20 for the MCP23017, 0x30 to 0x33 for the sensors, and 0x3C for the camera. We run the sensors in short distance mode,

which works up to about 1.3 m, with a 33 ms timing budget, so each sensor updates about 30 times per second. The firmware sends all four readings to the laptop 20 times per second.

Our PCB already has 4.7 k Ω I2C pull-ups (R7 and R8), and breakout boards like these usually add their own pull-ups too. During bring-up we will measure the combined pull-up on the bus, and if it is too strong we will remove the extra resistors from the breakout boards.

Table 7: Time-of-Flight Sensing Subsystem Requirements and Verification

Requirements	Verification
All four sensors start up with unique addresses every time the robot is powered on.	1) Power cycle the robot 10 times. 2) After each power-up, run an I2C scan from the firmware and print it over UART. 3) Confirm addresses 0x30, 0x31, 0x32, and 0x33 all appear every time. 4) Record a pass or fail for each power cycle.
Each sensor measures distance within ± 30 mm of the true distance from 50 mm to 1,000 mm.	1) Place a flat piece of cardboard in front of one sensor at 50 mm, 300 mm, 600 mm, and 1,000 mm, measured with a tape measure. 2) Record the reading shown on the laptop at each distance. 3) Confirm each reading is within 30 mm of the measured distance. 4) Repeat for all four sensors and record the results in a table of true distance, measured distance, and error.
The laptop receives updated readings from all four sensors at 10 Hz or more.	1) Log every telemetry message the laptop receives for 60 seconds. 2) Confirm at least 600 updates were received for each sensor. 3) Record the update count for each sensor.

2.3.6 Camera and Driver Interface Subsystem

The camera gives the driver a view from the robot itself. We use an Arducam B0158 OV5640 autofocus camera breakout [18] connected to the ESP32-S3's camera interface through the 20-pin header J19. The breakout's VCM (focus motor) supply pin is connected to 3.3 V, and its reset and power-down pins are controlled by the MCP23017. The main reason we picked the OV5640 [19] is that it compresses each frame to JPEG in hardware, so the ESP32 does not have to do any image compression itself. We stream at 320×240 resolution, where each JPEG frame is roughly 8 to 12 KB. At 15 frames per second that is about 1.4 Mbps, which is easy for 2.4 GHz WiFi. The ESP32 supplies the camera's 20 MHz clock on XCLK and configures the camera over the shared I2C bus. This subsystem supports high-level requirement 4, which requires video at 10 frames per second or more and command latency under 200 ms.

The driver interface is a web page that the ESP32 serves to the laptop. It shows the live video in the middle of the screen, the four distance readings as bars around the video, and the battery voltage, the state of each hammer (HOME, STRIKING, RETURNING, COOLDOWN, or FAULT), and a hit indicator from the impact sensor at the bottom. The driver uses the keyboard: W, A, S, and D to drive, J to fire the left hammer, L to fire the right hammer, K for Dual Smash, and the spacebar as a software stop that disarms the robot. When a left or right distance

reading is inside hammer range, that side's bar turns highlighted, which helps the driver know when to fire. On the ESP32, the camera streaming runs on one core and the control loop runs on the other core at a higher priority, so a slow video frame cannot delay a drive or hammer command.

Table 8: Camera and Driver Interface Subsystem Requirements and Verification

Requirements	Verification
The laptop shows a continuous video feed at 10 frames per second or more for a full 2-minute run.	1) Place the laptop about 3 m from the robot. 2) Drive the robot around for 2 minutes with the video on. 3) Use the frame counter on the interface to confirm the average frame rate is at least 10 fps and that the feed never freezes for more than 1 second. 4) Record the average frame rate and the longest freeze.
Each keyboard control triggers the correct action on the robot.	1) Press each control key 10 times. 2) Confirm the robot does the correct action every time and that the hammer states on the screen match what the hammers are actually doing. 3) Record the number of correct responses for each key.
The command round-trip time between the laptop and the robot stays under 200 ms during a 2-minute run with video streaming.	1) Have the laptop log the time it takes for each control packet to be acknowledged by the robot. 2) Run for 2 minutes with video and ToF data streaming. 3) Confirm that at least 95% of round trips are under 200 ms. Since the one-way latency is less than the round trip, this confirms the command latency requirement. 4) Plot a histogram of round-trip times for the report.

2.3.7 Impact Detection and Audio Feedback Subsystem

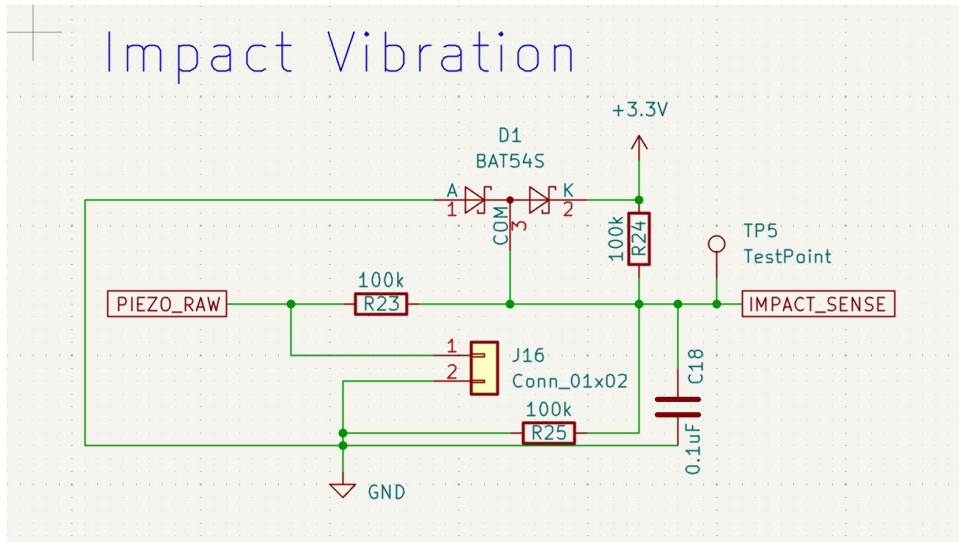
This subsystem lets the robot tell when a hit actually lands and gives the driver and the audience sound feedback. For impact detection, we use a SparkFun SEN-10293 piezo element [20], which produces a voltage when it is bent or shaken. It is glued to the inside of the chassis near the hammer pivots and connects through 2-pin JST-XH connector J16. A piezo can produce voltage spikes of tens of volts from a hard hit, so the signal is conditioned before it reaches the ESP32 (Figure 6a). R24 and R25 (100 k Ω each) bias the IMPACT_SENSE node at 1.65 V, half of the 3.3 V rail, so the ADC can see the spike whether it swings up or down. R23 (100 k Ω) sits in series with the piezo and limits the current it can push into the node. D1, a BAT54S dual Schottky diode, clamps the node between ground and 3.3 V so it cannot damage the ESP32. C18 (0.1 μ F) filters out high-frequency noise; together with the resistors (about 33 k Ω in parallel) it forms a low-pass filter with a time constant of about 3.3 ms, or a corner frequency of about 48 Hz. That smooths the signal enough that one hit shows up as a single clear bump instead of a burst of ringing. If testing shows the filter makes hits too hard to tell apart from driving vibration, we can lower C18 to raise the corner frequency. The conditioned signal, IMPACT_SENSE, goes to GPIO4, which is an ADC1 pin, and test point TP5 gives us access to it for testing. This subsystem is not tied to a single high-level requirement. It supports

the overall design by telling the driver when a hit actually lands, and its tones make the armed, disarmed, and failsafe states easy to hear, which supports the safety behavior behind high-level requirement 2.

The firmware samples `IMPACT_SENSE` quickly and looks for a spike that rises above a threshold set during testing. Normal driving produces small vibrations, while a hammer hit produces a much larger and sharper spike, so the threshold is set between the two. If a spike is detected within 300 ms after a Dual Smash command, the firmware counts it as a Dual Smash hit, sends a hit flag to the laptop, and plays a sound.

For audio, we use a PAM8302A class-D audio amplifier [21], powered from the 3.3 V rail (Figure 6b), driving a Same Sky CMS-15113-078L100-67 8 Ω , 0.7 W speaker [22] through 2-pin JST-XH connector J17. The ESP32 generates tones with a PWM signal on GPIO42 (`AUDIO_PWM`), and the MCP23017 controls the amplifier's enable pin (`AUDIO_EN` on GPA7), so the amplifier stays off and silent until the firmware needs it. Since our circuit feeds the amplifier directly from a PWM signal without a dedicated reconstruction filter, we are planning on simple tones and beeps rather than recorded audio. The robot plays a tone when it is armed, a different tone when it disarms or the failsafe triggers, a warning when the battery is low, and a short sound after a Dual Smash hit.

(a) Impact detection (piezo input)



(b) Audio amplifier

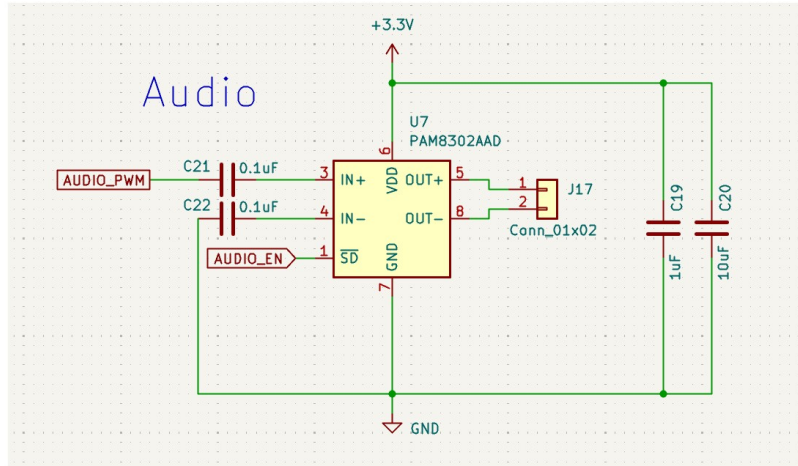


Figure 6: Impact detection and audio schematic: (a) piezo input and clamp; (b) PAM8302A audio amplifier

Table 9: Impact Detection and Audio Feedback Subsystem Requirements and Verification

Requirements	Verification
A hammer impact against a test block is detected in at least 18 of 20 strikes.	1) Place a 1 lb test block next to the robot within hammer reach. 2) Fire one hammer into the block 20 times. 3) Count how many strikes the laptop shows as a detected hit. 4) Confirm at least 18 of 20 are detected, and record the count for each hammer.
Normal driving without contact produces no more than 2 false impact detections per minute.	1) Drive the robot around the arena for 2 minutes without hitting anything, including turns and stops. 2) Count the number of hits shown on the laptop. 3) Confirm the count is 4 or fewer, and record it.
After a Dual Smash hit is	1) Probe IMPACT_SENSE at TP5 and the speaker signal at J17 on

Requirements	Verification
detected, the speaker starts playing within 200 ms.	an oscilloscope. 2) Fire Dual Smash into the test block. 3) Measure the time from the start of the spike on TP5 to the start of the speaker signal and confirm it is under 200 ms. 4) Repeat 10 times and record the maximum delay.
The amplifier stays silent until the firmware enables it.	1) Power on the robot and listen for any sound and probe the speaker signal at J17 for 10 seconds before arming. 2) Confirm there is no output. 3) Arm the robot and confirm the arming tone plays. 4) Record a pass or fail.

2.4 Software Design

The firmware is written using the ESP-IDF framework and is split into tasks. The control task runs every 10 ms on one core at the highest priority. It reads the latest control packet, checks the failsafe timer, reads the limit switches from the MCP23017, updates the drive PWM, and runs the hammer state machines. The communication task handles the WebSocket connection and sends telemetry 20 times per second. The camera task runs on the other core and sends JPEG frames to the video stream. The sensor task reads the four ToF sensors, the battery voltage, and the impact sensor, and the audio task plays tones when asked.

Each hammer runs its own copy of the state machine in Figure 7. Dual Smash simply sends the fire command to both state machines in the same control cycle.

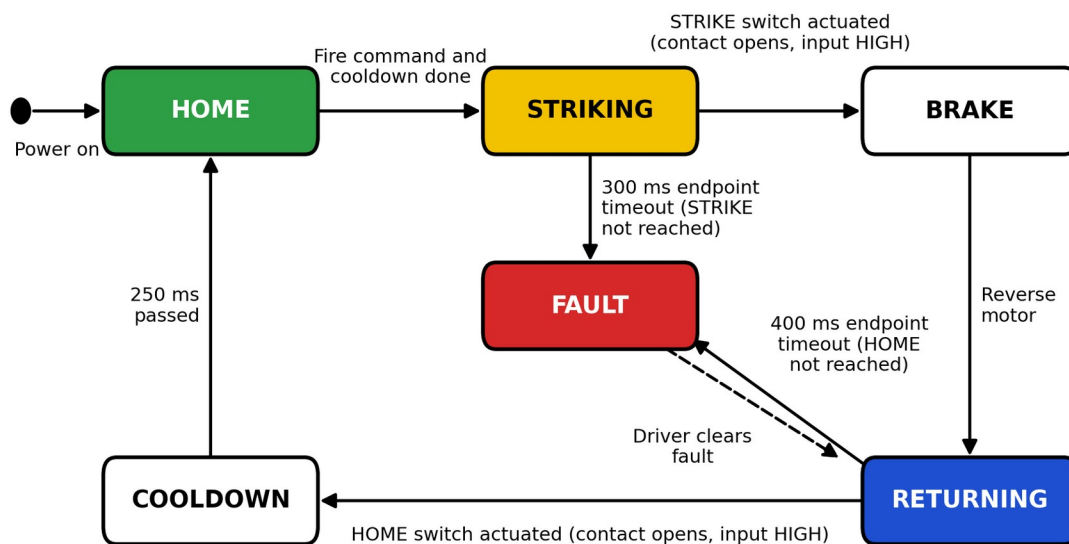


Figure 7: Hammer state machine

The overall robot has three states, shown in Figure 8. In DISARMED, all motor power is off and MOTOR_CUTOFF_EN is low. This is the state the robot starts in after power-up and goes back to after a failsafe or a software stop. In ARMED, the driver has connected and pressed the arm button, so MOTOR_PWR is on and the drive and hammers respond to commands. In FAULT, a hammer timeout or switch fault has been detected. The affected hammer is braked and locked out until the driver clears the fault, while driving still works.

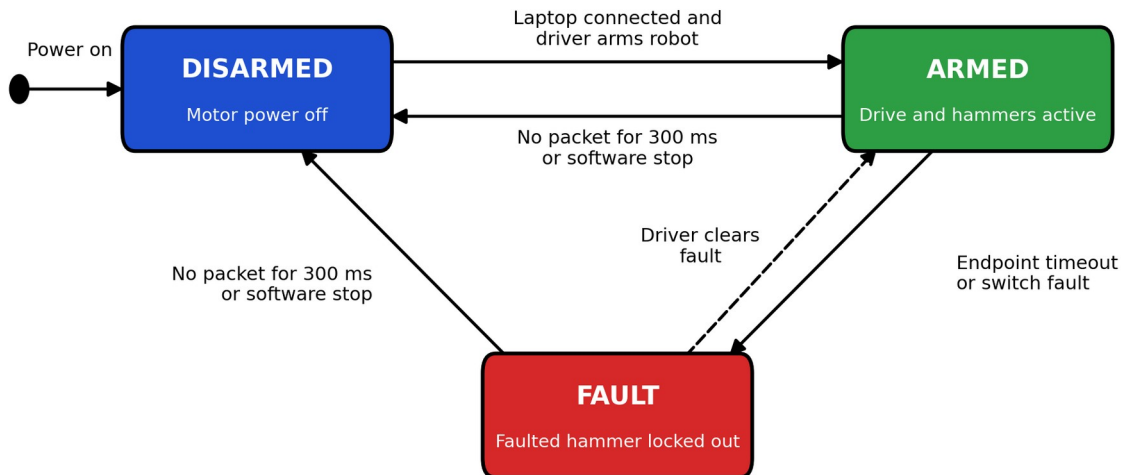


Figure 8: Robot state diagram

The control packet sent from the laptop is 8 bytes. It has a 16-bit sequence number, a signed 8-bit value for each drive motor (from -100 to 100), a hammer command byte (bit 0 fires the left hammer and bit 1 fires the right hammer, so setting both bits in the same packet is Dual Smash), a flags byte (arm and software stop), and a 16-bit checksum. The robot rejects any packet with a bad checksum or an old sequence number, so a corrupted or delayed packet cannot cause an unexpected move. The telemetry packet sent back to the laptop has the four distance readings in millimeters, the battery voltage, each hammer's state, the impact flag, and the fault flags.

2.5 Tolerance Analysis

The tightest electrical constraint in our design is the battery voltage under load. The robot has four motors that can all draw high current at once, for example if we are pushing against another robot with both wheels stalled while firing Dual Smash. When that happens, the battery voltage sags because of its internal resistance and the resistance of the fuse, connectors, switch, and wires. Two parts of our design care about this sag. The first is the AP63203 buck converter, which needs at least 3.8 V on its input [6]. If its input drops below that, the 3.3 V rail falls and the ESP32 resets, which would break our fourth high-level requirement. The second, and the tighter limit, is the DRV8871 hammer driver. It needs at least 6.5 V to operate, and it shuts its outputs

off if its supply falls to between 6.1 V and 6.4 V [13]. If that happens, the hammers stop working in the middle of a match.

First we estimate the resistance in each path. For the battery, we estimate about 15 mΩ per cell, so about 30 mΩ for the 2S pack, and we double it to 60 mΩ to account for an older or cold battery. The XT30 connector and battery leads add about 5 mΩ, and the fuse and fuse holder add about 15 mΩ. So the path from the cells to VBAT is about:

$$R_VBAT = 60 + 5 + 15 = 80 \text{ m}\Omega$$

The motor current also goes through the kill switch, the J9 connector and its wires (about 15 mΩ together), and Q1 (at most 10 mΩ with a -4.5 V gate drive, and less at our roughly -7 V gate drive [7]), which adds:

$$R_MOTOR = 15 + 10 = 25 \text{ m}\Omega$$

Next we estimate the current. A stalled drive motor has a winding resistance of about 6.0 V / 0.36 A = 16.7 Ω, so it draws V / 16.7 Ω. A stalled hammer motor has a winding resistance of about 6.0 V / 1.6 A = 3.75 Ω, but the DRV8871 limits it to 2 A. For the logic, the ESP32-S3 can draw about 350 mA while transmitting over WiFi [8], the camera about 150 mA, the four ToF sensors about 80 mA together, the MCP23017 about 1 mA, and the audio amplifier up to about 250 mA when playing a sound. That is about 0.83 A at 3.3 V, or 2.74 W. Assuming 85% buck efficiency, the battery has to supply about 3.22 W for the logic.

Case 1, logic: We first check the buck converter with a completely empty battery at a resting voltage of 6.0 V and every motor stalled. Each drive motor draws 6.0 / 16.7 = 0.36 A, and each hammer motor draws 6.0 / 3.75 = 1.6 A, which is under its 2 A limit. The logic draws about 3.22 W / 5.6 V = 0.58 A. So:

$$I_total = 2 \times 0.36 + 2 \times 1.6 + 0.58 = 4.50 \text{ A}$$

$$V_VBAT = 6.0 - 4.50 \times 0.080 = 5.64 \text{ V}$$

This is 1.84 V above the AP63203's 3.8 V minimum, so the ESP32 will not reset even with an empty battery and everything stalled. The resistance from the cells to VBAT would have to be (6.0 - 3.8) / 4.50, or about 0.49 Ω, before the logic browns out. That is about six times our already pessimistic estimate.

Case 2, hammers: Next we find the lowest resting battery voltage where the hammer drivers still get at least 6.5 V with every motor stalled. We set the firmware's hammer cutoff at a resting battery voltage of 7.4 V (3.7 V per cell) and check that case. Each drive motor draws 7.4 / 16.7 = 0.44 A, and each hammer motor would draw 7.4 / 3.75 = 1.97 A, which we round up to the 2 A limit. The logic draws about 3.22 W / 7.0 V = 0.46 A. So:

$$I_motor = 2 \times 0.44 + 2 \times 2.0 = 4.89 \text{ A}$$

$$I_total = 4.89 + 0.46 = 5.35 \text{ A}$$

$$V_VBAT = 7.4 - 5.35 \times 0.080 = 6.97 \text{ V}$$

$$V_MOTOR_PWR = 6.97 - 4.89 \times 0.025 = 6.85 \text{ V}$$

This is 0.35 V above the DRV8871's 6.5 V minimum, and 0.45 V above the highest point where its undervoltage lockout could trigger. These numbers treat each hammer as drawing its full current-limit value the whole time. Since the firmware caps the hammer duty cycle, the real average current is lower, so this is a conservative case. If we solve the same equations for the resting battery voltage where MOTOR_PWR lands exactly on 6.5 V, we get about 7.05 V. So below about 7.1 V, the hammer drivers are no longer guaranteed to work under a worst-case load, and our 7.4 V cutoff gives us about 0.35 V of margin. That margin also has to cover voltage ripple from the motor drivers' switching, which we did not model here. We will measure the real minimum on MOTOR_PWR during testing (Table 2), and if the ripple uses up too much of the margin, we will add a bulk capacitor on MOTOR_PWR in the next PCB revision.

The 7.4 V rule does not limit us much in a real match. If the robot averages about 2 A over a 2-minute match, it uses about 67 mAh of our 800 mAh battery, so a fully charged pack at 8.4 V stays well above 7.4 V for several matches.

We also checked a few related numbers. The battery's discharge rate in Case 2 is 5.35 A / 0.8 Ah, or about 6.7C, well under the pack's 45C rating. The 5.35 A total is also below the 7.5 A rating of fuse F1, so the fuse will not open during a worst-case stall but will still open on a short circuit. Q1 carries the motor current of about 4.89 A, so with at most 10 mΩ of on-resistance it dissipates $4.89^2 \times 0.010$, which is about 0.24 W. That is easy for its package with a copper pour on the PCB.

Based on these numbers, the logic will not reset under any battery condition we expect, and the hammers will keep working as long as the resting battery voltage is at least 7.4 V. The firmware measures the resting battery voltage while the motors are idle. It warns the driver below 7.6 V, blocks the hammers below 7.4 V, and disarms the robot completely below 6.8 V to protect the battery.

3. Cost and Schedule

3.1 Cost Analysis

Labor: Grainger reports that computer engineering graduates from Illinois have an average starting salary of about \$103,222 [23]. Divided by 2,080 working hours in a year, that is about \$49.63 per hour, which we round to \$50 per hour. We expect each of us to spend about 150 hours on this project.

$$\text{Labor per person} = \$50/\text{hour} \times 2.5 \times 150 \text{ hours} = \$18,750$$

$$\text{Total labor for two people} = \$18,750 \times 2 = \$37,500$$

Parts: Table 10 lists every part we plan to buy. Most resistors, capacitors, and diodes will come from the ECE Electronics Services Shop at no cost, and we budget up to \$10 for any small parts and wire we have to buy, and the PCB will be ordered through the course from PCBWay. Prices are estimates based on current vendor listings, and we will refresh them before ordering. Parts over the course budget will be purchased by our team.

Machine shop labor: 0 hours. The chassis and hammer parts will be 3D printed, and all other hardware is commercially available, so we do not need the ECE machine shop.

Table 10: Parts List

Description	Manufacturer	Part Number / Model	Quantity	Estimated Cost
WiFi microcontroller module, 8 MB flash / 8 MB PSRAM	Espressif Systems	ESP32-S3-WROOM-1-N8R8	1	\$6.32
5 MP autofocus camera breakout	Arducam	B0158 OV5640	1	\$24.99
Time-of-flight sensor breakout	ACEIRMC	TOF400C VL53L1X	4	\$19.82
Drive gearmotor, 6 V 50:1 LP, 300 RPM	E-S Motor	50:1 Micro Metal Gearmotor LP	2	\$11.90
Hammer gearmotor, 6 V 210:1 HP, 150 RPM	E-S Motor	210:1 Micro Metal Gearmotor HP	2	\$15.00
Dual drive motor driver	Toshiba	TB6612FNG	1	\$2.04
Hammer motor driver	Texas Instruments	DRV8871DDA	2	\$5.96
3.3 V buck converter	Diodes Incorporated	AP63203WU-7	1	\$0.80
I2C I/O expander	Microchip	MCP23017-E/SO	1	\$1.62
P-channel motor cutoff MOSFET	Diodes Incorporated	DMPH3010LK3-13	1	\$1.16
N-channel cutoff-control MOSFET	Various	2N7002	1	~\$0.20

Description	Manufacturer	Part Number / Model	Quantity	Estimated Cost
6.8 uH power inductor	Würth Elektronik	74438357068	1	~\$0.35
USB ESD protection array	Würth Elektronik	WE-TVS 82400102	1	~\$0.86
USB-C receptacle	GCT	USB4085	1	~\$1.00
MINI blade fuse holder	Keystone Electronics	3568	1	~\$3.00
7.5 A MINI blade fuse	Various	7.5 A MINI	1	~\$1.00
2S 800 mAh 45C LiPo with XT30	Tattu	TA-45C-800-2S1P-XT30	1	\$10.29
PCB XT30 connector	AMASS	XT30PW-M	1	~\$2.00
Hammer limit switch	Omron	D2F-L-D3	4	~\$8.40
Piezo impact element	SparkFun	SEN-10293	1	\$1.75
Audio amplifier	Diodes Incorporated	PAM8302AAD	1	~\$0.67
8 Ω speaker	Same Sky	CMS-15113-078L100-67	1	~\$3.10
JST-XH 2-pin PCB header	JST	B2B-XH-A	10	~\$5.00 total
JST-XH 5-pin PCB header	JST	B5B-XH-A	4	~\$3.00 total
32 mm N20-compatible wheels	TBD	TBD	2	~\$3.90
N20 motor brackets	TBD	TBD	4	~\$6.00
External manual kill switch, ≥ 10 A	TBD	TBD	1	~\$3.00
Resistors, capacitors, diodes, headers, wire	Various / E-Shop	Various	-	\$0-\$10
PETG chassis material	ECE shop / purchased filament	PETG	1 lot	~\$22.00
Steel hammer heads, fasteners and inserts	Various	-	1 lot	~\$15.00
2-layer custom PCB	PCBWay through ECE 445	-	1	\$0.00
Subtotal				\$190.13

Using the upper end of the \$0 to \$10 estimate for passive parts and wire, the parts subtotal is \$190.13. Adding 5% for shipping (\$9.51) and 10% for sales tax (\$19.01) gives a parts total of \$218.65. Adding the \$37,500 in labor gives a grand total of \$37,718.65.

3.2 Schedule

Table 11: Schedule

Week	Task	Person
Oct 5	Design Review (Oct 5 to 7)	Everyone
Oct 5	Finalize schematic, add TP6, pass PCBWay audit	Aditya
Oct 5	Order all parts	Everyone
Oct 5	Bring up ESP32-S3 dev board with WiFi control and failsafe	Laksh
Oct 8	First PCB order (second PCBWay round, Oct 8)	Aditya
Oct 12	Breadboard Demo (Oct 12 to 14): WiFi control, drive, one hammer driver, and limit switches on breadboard	Everyone
Oct 12	Camera streaming on dev board	Laksh
Oct 12	Chassis CAD, first version	Aditya
Oct 15	PCB fixes for third PCBWay round if needed (Oct 15)	Aditya
Oct 15	ToF bring-up with XSHUT address setup	Laksh
Oct 19	Print chassis, weigh it, mount drive motors and wheels	Aditya
Oct 19	Driver interface web page, first version	Laksh
Oct 26	Assemble PCB, test power rails, fuse, and motor cutoff	Aditya
Oct 26	Drive control firmware and PWM duty cap	Laksh
Oct 29	Final PCB revision if needed (fourth round, JLCPCB, Oct 29)	Aditya
Nov 2	Progress Demo (Nov 2 to 4)	Everyone
Nov 2	Hammer mechanism CAD, print, and limit switch mounting	Aditya
Nov 2	Hammer state machine and current limit testing	Laksh
Nov 9	Impact sensor threshold tuning and audio tones	Laksh
Nov 9	Integrate all subsystems on the PCB and weigh the robot	Everyone
Nov 16	Mock Demo (week of Nov 16) and R&V tests	Everyone
Nov 23	Fall break, buffer for fixes	Everyone
Nov 30	Final Demo (Nov 30 to Dec 2)	Everyone
Dec 3	Final Presentation (Dec 3 to 4)	Everyone
Dec 9	Final paper due	Everyone

3.3 Risk Analysis

The biggest risk in our design is the WiFi link. During the competition, many robots and laptops will be using 2.4 GHz WiFi in the same room, and our robot is also streaming video over the same link. If the link gets congested, control commands could get delayed. We reduce this risk in a few ways. The control packets are tiny and go over their own connection, the control task runs at the highest priority on its own core, and the video resolution is kept low. If the link drops completely, the failsafe cuts motor power within 500 ms, so the worst outcome is a robot that stops moving, not a robot that keeps moving on its own.

The second risk is battery voltage. As shown in Section 2.5, the DRV8871 hammer drivers need at least 6.5 V, which is close to the bottom of a 2S battery's range. If we start a match with a partly drained battery, the hammers could stop working under a heavy load. The firmware's 7.4 V hammer cutoff and low-battery warnings handle this, and we will always start a match with a fully charged battery.

The third risk is the hammer gearboxes. Small metal gearboxes are not designed for repeated impacts, so a hard hit could strip a gear. The 2 A current limit and the switch timeouts help by stopping the motor quickly when the hammer is blocked, and we plan to buy a spare motor for each hammer. The last risk is weight. Our budget has about 286 g of margin, but the printed chassis estimate is the biggest and least certain number, so we will weigh the first chassis print early and make it thinner if needed.

4. Discussion of Societal Impact, Engineering Standards, Ethics, and Safety

4.1 Societal Impact

Our robot is a competition project, so its direct effect on public health and welfare is small. However, robot combat is a popular way to get students interested in engineering, and a lot of the skills in this project carry over to useful real-world systems. Wireless control with a reliable failsafe, motor control with current limits, and sensor-assisted operation from a camera feed are the same ideas used in inspection robots, remote-controlled rescue robots, and other machines that work where people cannot easily go. Our design also shows that a capable robot can be built from low-cost parts, which makes this kind of project more accessible to students and schools with small budgets. On the environmental side, the main concerns are LiPo batteries and 3D printing waste. We will recycle used batteries through the proper channels on campus, keep our printed parts small, and reprint only what breaks.

4.2 Engineering Standards

Several standards apply to our project. The WiFi link follows IEEE 802.11 [24]. The ESP32-S3 module we use is pre-certified under FCC Part 15 rules for unlicensed radio devices [25], and using a pre-certified module while following Espressif's antenna and layout guidelines reduces the RF compliance burden of our design. The sensors, camera, and I/O expander communicate

using the I2C bus specification [26]. The USB-C port follows the USB 2.0 and USB Type-C specifications [27], including the 5.1 k Ω CC resistors that identify the board as a device. We will follow IPC-2221 [28] guidelines for PCB trace widths and spacing, especially for the VBAT and MOTOR_PWR traces, which carry up to about 5.4 A and are protected by the 7.5 A fuse. Lithium cells are covered by safety standards such as IEC 62133-2 [29], so we will buy our batteries from an established manufacturer and follow their handling instructions. Finally, the robot will follow the ECE 445 competition rules and general robot combat practices such as the SPARC ruleset [2].

4.3 Ethics

Our project relates to several parts of the IEEE Code of Ethics [30]. The first is to hold paramount the safety, health, and welfare of the public. This is why our robot defaults to having its motors off, cuts power when the connection is lost, and has an external kill switch and a fuse. We are designing the robot so that a failure leads to a robot that stops, not a robot that keeps going. The second is to be honest and realistic in our claims. Our hammers are not going to destroy other robots, and we will not claim they will. We will report our test results as they are, including any requirements we do not fully meet. The ACM Code of Ethics [31] also asks us to avoid harm, which applies to everyone around the arena and the lab, not just to us.

There is also an academic honesty side to this project. The idea of using end-stop switches on the hammer came from the Spring 2026 DC Hammer project [3], and we have credited that project. All datasheets, figures, and outside data in this document are cited, and all of the design and code in the final robot will be our own work.

4.4 Safety Concerns

The main safety concerns in our project are the LiPo battery, the moving hammers, and the electrical system.

LiPo batteries can catch fire if they are overcharged, over-discharged, punctured, or shorted. They can also get damaged inside a combat robot that is getting hit. Our hammers are a moving weapon, and even though they are small, they can pinch fingers. Electrically, the motor wiring carries several amps, so a short circuit could damage parts or heat up wires very quickly. Since our proposal, the main new safety concern is that a hammer can jam against something and keep pushing. We addressed this with the 2 A hardware current limit, the switch timeouts, and normally closed limit switch wiring, so a broken switch wire stops the hammer instead of letting it run. The speaker is low power and is not a safety concern, but we will keep its volume reasonable.

4.5 Safety Procedures

We will follow the course safety guidelines and use the following procedures:

- We will only charge the LiPo with a proper balance charger, inside a LiPo-safe bag, and never leave it charging unattended.

- We will store batteries at storage voltage (about 3.8 V per cell) when not in use, inspect them before every use, and stop using any battery that is puffy or damaged.
- The battery uses a keyed XT30 connector so it cannot be plugged in backward, and the 7.5 A fuse F1 protects against short circuits.
- The firmware warns the driver about low battery, blocks the hammers below 7.4 V, and disarms the robot below 6.8 V.
- The robot always powers up DISARMED with the motors off, and the driver has to connect and arm it before anything can move.
- The external kill switch can be reached from outside the robot, so anyone can cut motor power without relying on software.
- The limit switches are wired normally closed, so a broken switch wire stops the hammer.
- Hammer testing on the bench will only be done with the robot on a stand, with hands away from both hammers, and with safety glasses on.
- Full-speed driving and hammer testing will only be done inside the arena or an enclosed test area.
- When we are working on the robot with the battery connected, motor power will be turned off with the kill switch.

We will write a lab safety document for our battery and hammer testing. If the course asks for a full safety manual because of the weapon, we will write one and show at the demo that we follow it.

5. Citations

- [1] ECE 445 Senior Design Laboratory, University of Illinois Urbana-Champaign, 2026. [Online]. Available: <https://courses.grainger.illinois.edu/ece445/>
- [2] SPARC, "Standardized Robot Combat Rules." [Online]. Available: <https://www.sparc.tools/>
- [3] C. Sprague, G. Gathman, and I. Purkis, "Ant-Weight Battlebot - DC Hammer," ECE 445 Senior Design Project, University of Illinois Urbana-Champaign, Spring 2026. [Online]. Available: <https://courses.grainger.illinois.edu/ece445/project.asp?id=12310>
- [4] GetFPV, "Tattu 800mAh 2S 45C LiPo Battery - XT30," product page. [Online]. Available: <https://www.getfpv.com/tattu-800mah-2s-45c-lipo-battery-xt30.html>
- [5] LCSC Electronics, "Changzhou Amass Elec XT30PW-M (C431092)," product page. [Online]. Available: <https://www.lcsc.com/product-image/C431092.html>
- [6] Diodes Incorporated, "AP63200/AP63201/AP63203/AP63205: 3.8V to 32V Input, 2A Low IQ Synchronous Buck Converter," datasheet. [Online]. Available: <https://www.diodes.com/assets/Datasheets/AP63200-AP63201-AP63203-AP63205.pdf>
- [7] Diodes Incorporated, "DMPH3010LK3: 175°C P-Channel Enhancement Mode MOSFET," datasheet DS38123. [Online]. Available: <https://www.diodes.com/assets/Datasheets/DMPH3010LK3.pdf>

- [8] Espressif Systems, "ESP32-S3-WROOM-1 / ESP32-S3-WROOM-1U Datasheet." [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_u_datasheet_en.pdf
- [9] Microchip Technology, "MCP23017/MCP23S17 16-Bit I/O Expander with Serial Interface," datasheet DS20001952. [Online]. Available: <https://www.microchip.com/bin/downloaddocument?documentuuiid=e862a7c9-2efa-44cd-ab21-597681ca4f86>
- [10] RobotShop, "E-S Motor 6V 50:1 Micro Metal Gearmotor LP, 300rpm," product page. [Online]. Available: <https://www.robotshop.com/products/e-s-motor-6v-501-micro-metal-gearmotor-lp-300rpm>
- [11] Toshiba Electronic Devices and Storage, "TB6612FNG Driver IC for Dual DC Motor," datasheet. [Online]. Available: <https://www.digikey.at/en/htmldatasheets/production/1857561/0/0/1/2927>
- [12] RobotShop, "E-S Motor 210:1 Micro Metal Gearmotor HP - 6V 150RPM," product page. [Online]. Available: <https://www.robotshop.com/products/e-s-motor-2101-micro-metal-gearmotor-hp-6v-150rpm>
- [13] Texas Instruments, "DRV8871 3.6-A Brushed DC Motor Driver With Internal Current Sense (PWM Control)," datasheet SLVSCY9B, 2016. [Online]. Available: <https://www.ti.com/lit/gpn/drv8871>
- [14] Texas Instruments, "DRV8874 H-Bridge Motor Driver with Integrated Current Sense and Regulation," product page. [Online]. Available: <https://www.ti.com/product/DRV8874>
- [15] Omron, "D2F Ultra Subminiature Basic Switch," product page and datasheet. [Online]. Available: <https://components.omron.com/sg-en/products/switches/D2F>
- [16] STMicroelectronics, "VL53L1X Time-of-Flight Ranging Sensor," datasheet. [Online]. Available: <https://www.st.com/resource/en/datasheet/vl53l1x.pdf>
- [17] ACEIRMC, "TOF400C VL53L1X Time-of-Flight Distance Sensor Module, 4 Pack," Amazon listing B0DC6M6G7W. [Online]. Available: <https://www.amazon.com/dp/B0DC6M6G7W>
- [18] Arducam, "5MP OV5640D CMOS AF Camera Module 1/4-Inch Module (B0158)," product page. [Online]. Available: <https://www.arducam.com/5-mp-ov5640d-cmos-af-camera-module-1-4-inch-module.html>
- [19] OmniVision Technologies, "OV5640 Color CMOS QSXGA (5 Megapixel) Image Sensor with OmniBSI Technology," datasheet, ver. 2.03, 2011. [Online]. Available: https://www.acmesystems.it/www/roadrunner_isc/OV5640.pdf
- [20] SparkFun Electronics, "Piezo Element, SEN-10293," product page. [Online]. Available: <https://www.sparkfun.com/products/10293>
- [21] Diodes Incorporated, "PAM8302A 2.5W Filterless Class-D Mono Audio Amplifier," product page and datasheet. [Online]. Available: <https://www.diodes.com/part/view/PAM8302A>
- [22] Same Sky (formerly CUI Devices), "CMS-15113-078X-67 Speaker," datasheet. [Online]. Available: <https://www.cuidevices.com/product/resource/cms-15113-078x-67.pdf>
- [23] The Grainger College of Engineering, "Computer Engineering." [Online]. Available: <https://grainger.illinois.edu/academics/undergraduate/majors-and-minors/computer-engineering>
- [24] "IEEE Standard for Information Technology, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE Std 802.11-2020, 2021. [Online]. Available: <https://standards.ieee.org/ieee/802.11/7028/>

- [25] Federal Communications Commission, "47 CFR Part 15: Radio Frequency Devices." [Online]. Available: <https://www.ecfr.gov/current/title-47/chapter-I/subchapter-A/part-15>
- [26] NXP Semiconductors, "UM10204: I2C-bus Specification and User Manual," Rev. 7.0, 2021. [Online]. Available: <https://www.nxp.com/docs/en/user-guide/UM10204.pdf>
- [27] USB Implementers Forum, "Universal Serial Bus Type-C Cable and Connector Specification." [Online]. Available: <https://www.usb.org>
- [28] IPC, "IPC-2221B: Generic Standard on Printed Board Design," 2012. [Online]. Available: <https://www.ipc.org/TOC/IPC-2221B.pdf>
- [29] International Electrotechnical Commission, "IEC 62133-2: Safety Requirements for Portable Sealed Secondary Lithium Cells and Batteries," 2017. [Online]. Available: <https://webstore.iec.ch/en/publication/32662>
- [30] IEEE, "IEEE Code of Ethics," 2020. [Online]. Available: <https://www.ieee.org/about/corporate/governance/p7-8.html>
- [31] ACM, "ACM Code of Ethics and Professional Conduct," 2018. [Online]. Available: <https://www.acm.org/code-of-ethics>

Note: AI tools were used to help word and format this document and to draw some of its figures. All technical content was reviewed and verified by our team.