Autonomous Featherweight (30lb) Battlebot

ECE 445 Senior Design Final Report - Spring 2025

Group #43

Jason Mei (jasonm5)

Qinghuai Yao (qyao6)

Michael Ko (ykko2)

TA: Michael Gamota

Professor: Viktor Gruev

7 May 2025

Abstract

This report outlines the design and development progress of an autonomous subsystem for "CRACK?", a 30lb combat robot intended for autonomous operation within a controlled arena. The primary focus is on implementing a computer vision-driven pure pursuit control algorithm that tracks and engages opponent robots based on real-time pose estimation using AprilTags. A custom simulation environment using Pygame was developed to model robot dynamics and verify path-planning logic, including a modified avoidance algorithm that avoids direct weapon-to-weapon contact. The control outputs are transmitted using ESP-NOW between microcontrollers, achieving low-latency communication. Verification tests were conducted to validate algorithm accuracy, communication latency, and PWM signal integrity. The system demonstrates reliable performance in simulation with plans to extend to real-world testing. Ethical considerations regarding autonomy, safety, and transparency were also evaluated in accordance with IEEE and ACM codes.

Figure 1: Image of CRACK?.

Table of Contents

1 Introduction....4
1.1 Problem....4
1.2 Solution....4
1.3 Functionality....5
1.4 Subsystem Overview....5
2 Design....7
2.1 Design Procedure....7
2.2 Alternatives for Design....7
2.3 Design Details....8
2.3.1 Voltage Reader Subsystem....8
2.3.2 PWM I/O Subsystem (Input)....9
2.3.3 PWM I/O Subsystem (Output)....9
2.3.4 IMU Subsystem....11
2.3.5 Remote Camera and April Tag Subsystem....11
2.3.6 Autonomy Subsystem....12
3 Verification....13
3.1 Voltage Reader Subsystem....13
3.2 PWM I/O Subsystem....13
3.3 IMU Subsystem....14
3.4 Remote Camera and April Tag Subsystem....15
3.5 Autonomy Subsystem....16
4 Cost and Schedule....18
4.1 Cost Analysis....18
4.1.1 Labor....18
4.1.2 Parts....18
4.1.3 Total Cost....18
4.2 Schedule....18
5 Conclusion....19
5.1 Summary of Results and Future Work....19
5.2 Ethical Considerations....20
6 References....21
Appendix A. Requirement and Verification Table....23
Appendix B. Data Tables and Example Figures....25
Appendix C. Flowchart and Block Diagram....29
Appendix D. Costs....31
Appendix E. Schedule....34

1 Introduction

1.1 Problem

iRobotics, an RSO on campus, has built multiple battlebots that are entered into competitions across the U.S. One of the robots that has been developed is called "CRACK?", a 30lb hammer-axe battlebot. The robot has already been designed and completed; however, the project would be to upgrade this robot from manual control to autonomous control. One of the main challenges to this project is the transition from the theoretical world to the real world. The designs that we are working on may be feasible in theory and in simulation, but the real world is a lot more complex, and doubly so in the world of combat robotics. Our product has been designed such that it can hold up to the rigors of a typical featherweight match.

In a standard battlebots match, the robots are placed on opposite corners of a 16 ’ x 16’ arena, facing each other. Once the match begins, robots have 3 minutes to attack the other robot and cause enough damage to get the opponent to stop moving. The match can end in 5 ways [1]:

- The opposing robot is “knocked out” by ceasing all translational motion for 10 seconds.
- The opposing robot is sent out of the arena.
- The opposing robot has an exposed battery, which is a hazard.
- The match reaches the 3-minute time limit, and the judges decide the winner.
- The opposing robot’s driver taps out and forfeits the match.

1.2 Solution

For this project, the plan is to use a camera mounted just outside the polycarbonate walls for a live state of the arena, sending information to a computer. The computer can then use image

transforms to get an accurate top-down view of the field, which allows the computer to then calculate the next movements, either by using a pure-pursuit algorithm or a machine learning algorithm, potentially. The control is then passed over to a microcontroller board mounted within the robot, which sends signals to the motors, and drives the robot or fires the hammer.

1.3 Functionality

Overall, there are three high-level requirements for our project:

- Track both robots within ±12 inches and send PWM signals within 16ms per frame.
    - This requirement ensures real-time decision-making, which is critical for a battlebot combat scenario.
    - Timely and accurate robot localization enables the robot to get correct instructions calculated by a pure-pursuit algorithm implemented on a PC.
- Shut off safely on safety violations, with Bluetooth E-STOP and 100ft signal loss fail-safe.
    - Without safety guarantees, the system will not be practical for real-world usage and the competition.
- Track and intercept a 5 MPH RC car using the full autonomous system.
    - This is the core functional goal—demonstrating a robot that can autonomously engage a moving adversary, simulating a combat or pursuit scenario.

1.4 Subsystem Overview

The project is laid out over multiple blocks: the camera, the virtual environment, the pure pursuit algorithm, and the on-robot custom board, which consists of the IMU, voltage reader, and

PWM control.

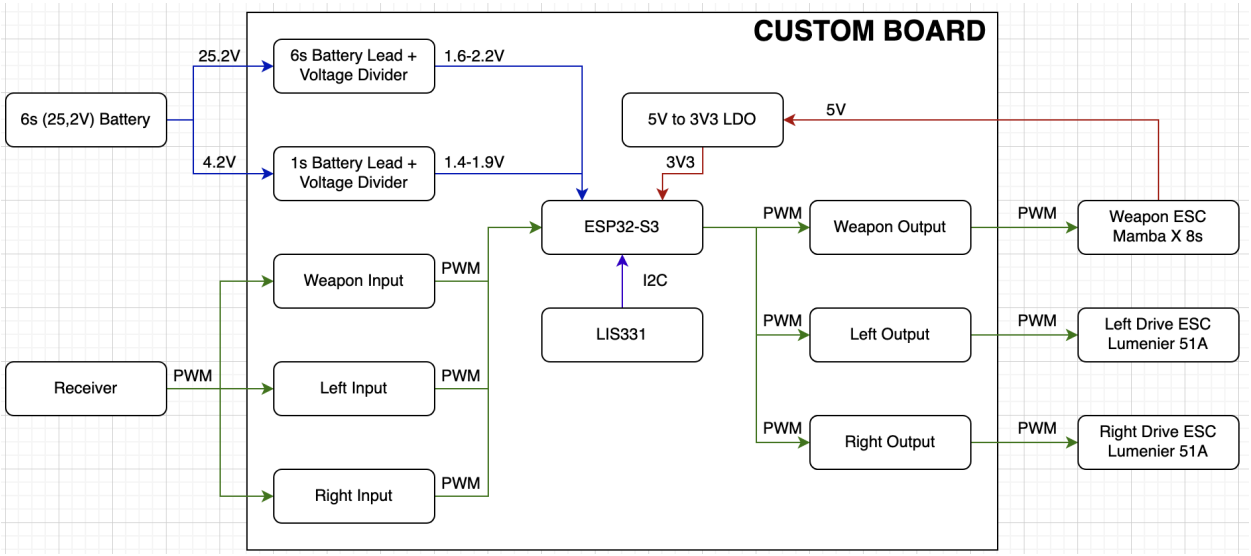


Figure 2: Custom board block diagram.

The physical custom board is relatively simple in design. As shown in Figure 2, the board receives 5V power directly from the BEC (Battery Eliminator Circuit) of the weapon ESC, which is then stepped down to 3.3V using an LDO (low-dropout) regulator. This is used to power the microcontroller, of which we will be using the ESP32-S3. Receiver inputs will also be passed into the ESP32, alongside voltages from the battery leads, converted into an analog voltage from 1.4 - 2.2V to not exceed the rated voltage of the microcontroller pins. Additionally, the LIS331, our IMU, will send direct data over I2C. Lastly, the ESP32-S3 will also receive a Bluetooth signal (not shown, as the majority of the Bluetooth circuitry is integrated within the microcontroller) from the computer. It will then use this information to send a PWM signal to the three ESCs. All the requirements and the verification table are in Appendix A.

2 Design

2.1 Design Procedure

For this project, we want to implement several key functionalities: battery voltage monitoring, PWM-based motor control, robot detection, pure-pursuit algorithm, and reliable and fast communication between a computer and the on-robot PCB.

Overall, the navigation capability, based on the pure-pursuit algorithm, is achieved by the communication between a computer and the ESP32-S3 microcontroller embedded in the on-robot PCB. A detection subsystem, comprising a camera and an IMU, captures the environment information (position of both robots). The environment information is sent to the PC, which processes it with the pure-pursuit algorithm. Then, the control signals are sent to ESP32-S3 via ESP-NOW protocol.

For battery reading, we are using a voltage divider because the input voltage of the battery is larger than the 3.3V pin rating. Then, we are controlling the robot with PWM signals with the microcontroller ESP32-S3. For robot detection, the camera detects the AprilTags on both robots, and the IMU works as a backup when the AprilTags are not detected. The protocol of connection between the PC and the ESP32-S3 is ESP-NOW.

2.2 Alternatives for Design

Several alternatives were considered during the design process, particularly regarding the communication protocol and the selection of the sensor for position estimation.

Initially, we chose Bluetooth as the communication protocol, but the ESP32-S3 supports only Bluetooth Low Energy (BLE), which has limited throughput and high latency.

Subsequently, Wi-Fi was tested, but the high latency rendered it unreliable for low-latency control. Finally, ESP-NOW was selected due to its low latency. We, however, need to have the other ESP32-S3 attached to the host PC, because ESP-NOW supports only peer-to-peer communication.

In terms of robot detection, an inertial measurement unit (IMU) was chosen when AprilTag detection fails. An alternative for robot detection is light sensors, which could potentially provide positional information based on structured lighting. However, this alternative typically requires higher power and more complex environmental calibration. Also, most light sensors have lower update rates than IMUs, which makes them less reliable for fast-moving robots.

2.3 Design Details

2.3.1 Voltage Reader Subsystem

CRACK was powered by two 6s Tattu R-Line 1400mAh LiPo batteries connected in parallel. Each 6s battery consisted of six cells in series, with individual cell voltages ranging from 3.2V to 4.2V. Since discharging below 3.2V risks permanent battery damage [2], we monitored voltage in real time. This was done by tapping the first and last cells using the balance leads—specifically pin 7 (25.2V), pin 2 (4.2V), and pin 1 (GND) on the JST-XH connector—to estimate the full pack voltage with improved accuracy. These leads connected directly to a custom board.

To safely measure voltages beyond the 3.3V tolerance of the ESP32-S3 microcontroller, we used voltage dividers built from high-ohm resistors to reduce current draw, paired with zener diodes for overvoltage protection. At most, the divider circuit drew 5.37 μA—negligible compared to the battery's ~40A continuous output. The scaled-down voltages were then fed into

the ESP32-S3's 12-bit ADC, which provided millivolt-level resolution (0.001V) [3]. Figure 3 shows a Falstad simulation of the voltage divider setup.

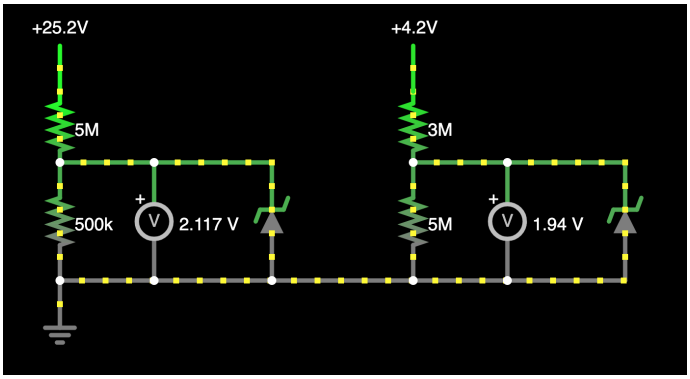


Figure 3: Voltage divider simulation.

2.3.2 PWM I/O Subsystem (Input)

The receiver outputs a PWM voltage, where 0V is a zero, and 3.3V is high. PWM (pulse width modulation) is a method of representing a signal as a rectangular wave with a varying duty cycle. Every period, the receiver will pull each channel high for a certain amount of time, and the width of the pulse represents the signal that the radio is sending. The ESP32-S3 can capture pulses to the same accuracy as it can output, which is described below. This PWM system reads at 50Hz, and from channels 1-5. Each channel's use is described in Table 1 in Appendix B.

2.3.3 PWM I/O Subsystem (Output)

The standard within RC cars (which are the ESCs that CRACK uses) is that a "0" signal is 1.5ms every 20ms (50Hz), a "-100" signal is 1ms every 20ms, and a "100" signal is 2ms every 20ms. However, because the duty cycle can be variable (all that matters is that the period is between 50-200Hz), we will be sending one signal for each frame that we receive from the camera. A PWM signal example is shown in Figure 2 of Appendix B.

Once the angle difference is obtained, we then need to calculate the outputs that we will send to the robot. Since the robot is driven with 3 total inputs, we sent those three inputs as int16s (to minimize the total amount of data sent, from 1000-2000) - Left drive, right drive, and weapon drive. Technically, the ESCs are precise down to 32 bits [5], but 8 bits is enough to get a fairly accurate drive, and is still much more precise than human input. We calculated the values sent using the following formulas, where d_{robots} represents the distance between robots in meters, and θ_{diff} represents the angle difference in degrees:

$$output_{right} = clip((d_{robots} + 0.4) - 1.5 * \frac{\theta_{diff}}{180}), \text{-1.0, 1.0}) \quad (3)$$

$$output_{right} = clip((d_{robots} + 0.4) - 1.5 * \frac{\theta_{diff}}{180}), \text{-1.0, 1.0}) \quad (4)$$

The left and right drives will be converted to PWM using the following formula:

$$PWM_{left/right} = output_{left/right} * 200 + 1500 \quad (5)$$

The ESP32-S3 has a specific peripheral for this - the MCPWM (Motor Control Pulse Width Modulator) is a versatile PWM generator, which contains various submodules to make it a key element in power electronic applications like motor control, digital power, and so on. [4] We will be supplying the microcontroller with a 24 MHz crystal oscillator, so that it can precisely clock for each period, and give us high precision for the pulse widths. Table 2 in Appendix B shows the conversion.

2.3.4 IMU Subsystem

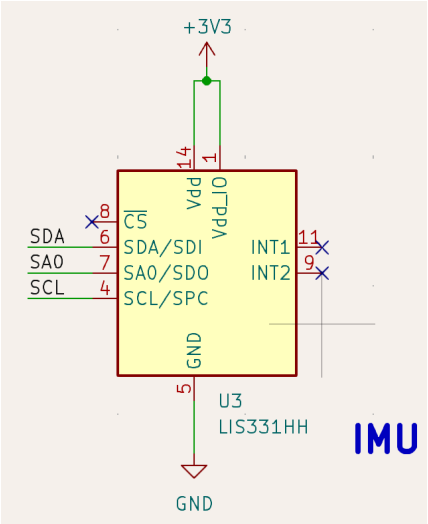


Figure 4: IMU schematic.

We used the LIS331 IMU for acceleration data. This allows us to read live acceleration data from the robot and then send that information over to the computer over Bluetooth. This IMU allows us to obtain a ground truth, regardless of what the camera is seeing. We will identify by testing which method of measuring orientation works best, whether it is the April tag or the IMU. We will perform a calibration at the start of each test, which should allow us to get global orientation data based on the initial placement of the robot. We understand that the IMU could potentially be used for global position data, but with the possibility of the robot getting launched into the air, the global position information would be a lot less accurate.

2.3.5 Remote Camera and April Tag Subsystem

We used the NexiGo N980P USB Camera, which features a 1080p 60fps sensor and a 120° lens, providing a wide-angle view of the arena. The camera was mounted on a tripod in a fixed location to ensure consistent positioning, allowing for accurate pose estimation without

requiring adjustment. It is connected to the computer via USB Type-A, enabling a direct and reliable data stream for processing.

To identify robot positions, we used AprilTags—visual fiducials that aid in object localization [5]. Each robot had tags mounted on the top and bottom, which were detected using OpenCV. We selected 16h5 family tags from the AprilTag library developed by the University of Michigan's AprilRobotics team. An example AprilTag is shown below in Figure 5 of Appendix B. Detection was handled through the AprilTag Python library by Berwin, a PyPI port of the original Swatbotics codebase [6]. OpenCV provided an efficient calibration workflow, allowing us to quickly set up the camera using known parameters.

2.3.6 Autonomy Subsystem

The computer took in multiple inputs: live camera feed of the robots in the arena, IMU information from the robot, and previous location data (from earlier calculations). We used a standard pure-pursuit algorithm for the initial pass. We did this by obtaining the pose from the camera subsystem, as well as the IMU information of the robot. Once we had two global positions, we were able to use line-circle intersection to identify the goal points for the robot. [7]

Effectively, we tried to minimize the angle between the front of CRACK and the opponent, and minimize the distance between the two robots. Before implementing this design in the real world, we constructed a Pygame environment where we simulated how the robot would behave in the world. An example of a Pygame simulation is shown in Figure 4 of Appendix B

The outline for the robot's autonomous initialization follows the flowchart, which is shown in Figure 1 of Appendix C. The robot follows the procedure, which is shown in Figure 2 of Appendix C, for actual autonomous control, assuming the robot is currently in the autonomous control mode.

3 Verification

3.1 Voltage Reader Subsystem

We successfully completed both tests, which are outlined in Table 1 of Appendix A.

Input voltage	Measured voltage	Expected Value
1.4 V	0.81 V	0.875 V
1.6 V	0.91 V	1 V
1.8 V	1.01 V	1.125 V
2.0 V	1.11 V	1.25 V
2.2 V	1.21 V	1.375 V
5 V	0.40 V	0.45 V

Table 1: Voltage Divider Test Result

The expected value is calculated with the formula: $V_{out} = (V_{in} \times R_2) / (R_1 + R_2)$

All measured values fall within +/- 0.2 V of the expected values, which indicates the accuracy of the voltage monitoring subsystem. For the 5 V input case, the measured value remains safely below 3.3 V, which shows that the voltage divider effectively protects the ADC pins from overvoltage.

3.2 PWM I/O Subsystem

We completed both tests, which are outlined in Tables 2 and 3 of Appendix A, with a pass rate of 95% over 5 sweeps being measured.

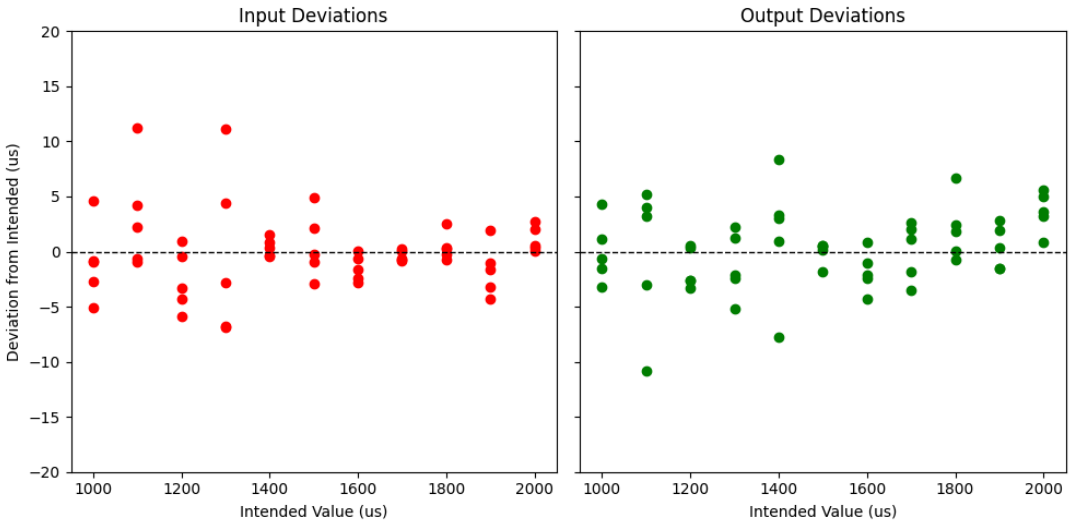


Figure 5: Deviation plots for each intended value over 5 sweeps.

The output deviations averaged 0.27 above the intended value, while the input deviations averaged about 0.14 below the intended value. The maximum deviation was 10.8µs, so all samples were within bounds, and the test was a success.

3.3 IMU Subsystem

Unfortunately, we cannot test the IMU subsystem on the robot because of the inability to program the PCB in the end. However, we still did some tests and verified the hardware implementation of the IMU. We completed the test, which is outlined in Table 4 of Appendix A.

Test condition	Expected (m/s^2)	Outputs (m/s^2)	Pass/Fail
Face-up (Z-axis)	X = 0, Y = 0, Z = 9.8	X = -0.167, Y = -0,199, Z = 9.689	Pass
Sideways (X-axis)	X = 9.8, Y = 0, Z = 0	X = 9.591, Y = 2.229, Z = 2.844	Pass
Sideways (Y-axis)	X = 0, Y = 9.8, Z = 0	X = 0.539, Y =9.862, Z = 1.340	Pass

Table 2: IMU test results

Because we hold the IMU with our hand for sideways tests, the measurement of sideways has minor deviations. Nonetheless, the results remain consistent with the expected values along the corresponding axes.

3.4 Remote Camera and April Tag Subsystem

In order to verify the accuracy of the subsystem over time, we completed the tests from Appendix A, Table 5. We placed the robot in known locations and compared the calculated position of the robot to the ground truth distance. As shown in Figure 6, the system manages to approximate locations down to the worst case of 5.9 inches, with a very small bias towards the positive x direction. The measured distance is acceptable for our standards and completes the test.

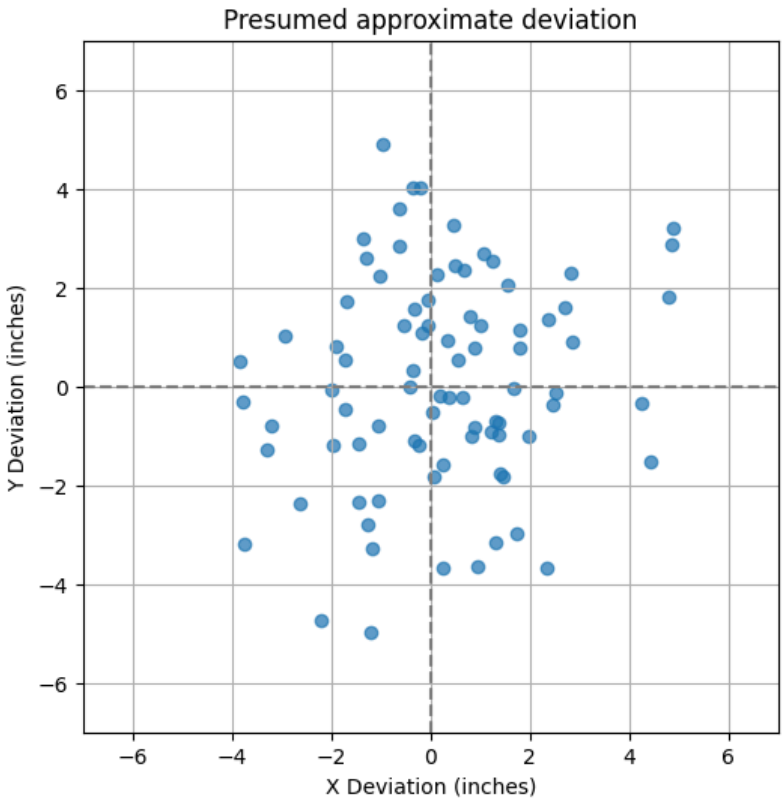


Figure 6: Approximate deviation from ground truth.

3.5 Autonomy Subsystem

To verify the pure-pursuit algorithm, we completed the tests from Appendix A, Table 6. A successful test would be a 95%+ hit rate over at least 100 simulations. I placed the simulated robot directly in the center of the arena and randomized the location of the spawned target. Figure 7 shows the location of starting target points for each test.

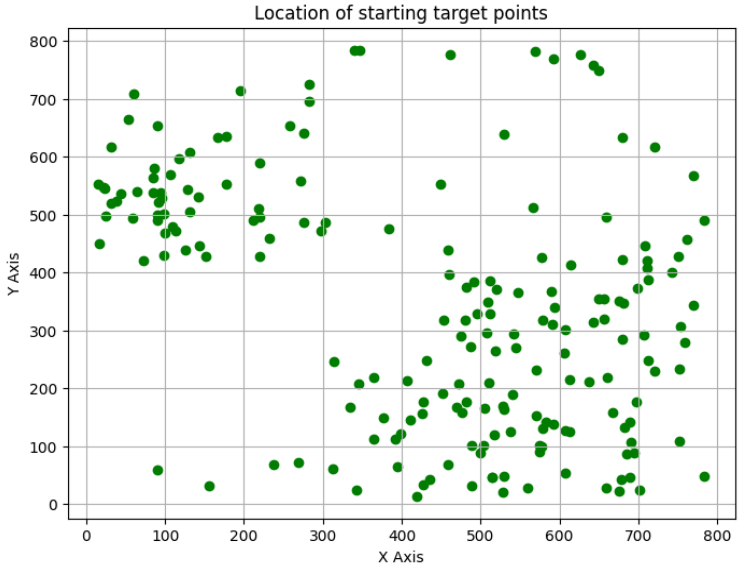


Figure 7: Diagram of the starting target locations

Out of every test, the algorithm never missed and got to the target in under 5 seconds. The measured average “time-to-target” was approximately 2.3 seconds. The reason for the mildly sparse groupings for the target along the y=x line is due to the randomization algorithm, which avoids placing the target directly on top of, or near, the robot. However, the entire arena is symmetrical in all 4 ways. This test was a success.

For the time for each autonomous pass, we were forced to adjust to the “update once every other frame” protocol as considered in the tolerance analysis of the design document. Shown below is the measurement for the tests:

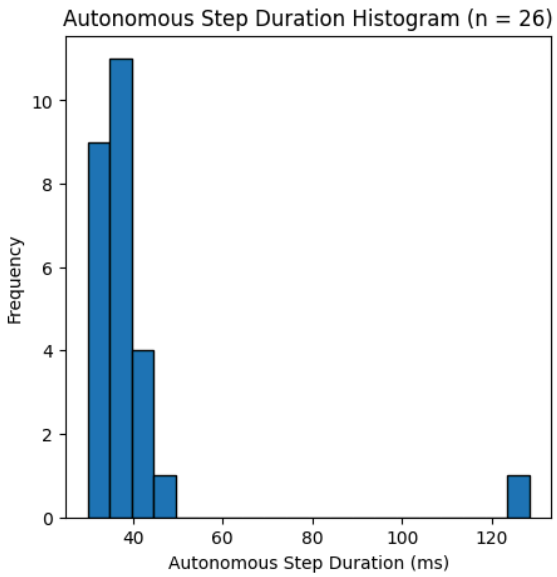


Figure 8: Autonomous Step Duration histogram.

The measured step duration concentrates on approximately 35ms, which is within our adjusted 2-frame window. We have one outlier of 130ms within our sample, primarily due to occasionally doing a full quad-decimate reading of the entire video frame to recapture AprilTag detection. However, this is still acceptable for our standards and completes the test.

4 Cost and Schedule

4.1 Cost Analysis

4.1.1 Labor

The average starting salary of Computer Engineering students is $109,176[10]. Assume a full-time job with 40 hours per week and 52 weeks per year. $109,176/(40 * 52) = $52/hour. For each team member, 5 hours per week will be spent on the project. Thus, 5 * 10 = 50 hours will be spent by each team member. Then, a reasonable salary for each team member is: 52 * 2.5 * 50 = $6,500

4.1.2 Parts

We had several orders for parts. All information on parts costs is included in Table 1 of Appendix D. The total cost for parts is $144.58.

4.1.3 Total Cost

$$\$6,500 + \$144.58 = \$6644.58$$

4.2 Schedule

The entirety of the schedule is listed out in Table 1 of Appendix E. The schedule is what we intended from week 1 until the end of the semester. There were a few changes to the actual schedule due to the parts arriving late and having trouble with the PCB orders. However, we still managed to stick mostly to the plan and have a successful project for the final demo.

5 Conclusion

5.1 Summary of Results and Future Work

We successfully completed the objective and the high-level requirements. The robot was able to move around autonomously based on the RC car's movement, meaning that the pure-pursuit algorithm was accurate, along with the camera being able to detect the AprilTags on the robots. The PCB was also programmed as we intended. However, the PCB was not working for the final demo as the RX pin of the MCU had some issues. We thought it was an issue with the RX pin due to other pins functioning properly, as we were able to get the necessary outputs on the serial monitor, but simply could not upload it. One other uncertainty we had was that the IMU output was not accurate because we had to hold the IMU with our hand when we tried to get the data, when held sideways. This had the data off by a bit since the IMU was shaking and was not in the ideal position. We also struggled with the connection from the computer to the on-robot board as we switched three times from Bluetooth to WiFi and finally to ESP-NOW. Lastly, the Micro-USB port was connected to the wrong pins on the ESP as the D+ and D- pins were inversely connected. Therefore, we had to use a USB-to-UART bridge to solve this issue. In the future, we will create a bigger PCB than what we currently have to avoid any soldering issues. We could also have a camera with higher resolution, since an expensive camera would be able to detect AprilTags much better. Unfortunately, our project had a budget limit, forcing us to purchase a cheap webcam instead. With a higher quality camera, we could have the battlebot move around in a larger environment rather than the 6' x 6' environment we have as of now.

5.2 Ethical Considerations

Our project must adhere to all IEEE and ACM ethical guidelines to maintain safety standards. Several potential ethical considerations warrant attention. Regarding ACM code 1.2, we should avoid harm and ensure safety. The autonomous mobile robots can bring in hazards like failure in the hardware, or showing erratic behavior. To address safety and ethical concerns, we will implement a robust emergency shutdown mechanism that responds instantly through Bluetooth connectivity to prevent loss-of-control scenarios. As a backup measure, we will incorporate multiple redundant safety systems, both physical and wireless [11]. For testing and validation purposes, we will be performing controlled testing in simulated environments.

Following ACM code 1.6, our vision system must maintain privacy standards by avoiding the storage of unnecessary data [12]. Since our robot uses a camera to view the movements of the robot, we must only contain necessary information and avoid collecting private personal information that might invade someone's privacy. To meet the FCC regulations, the Bluetooth connection must not have any interference with other robots [13].

Our robot is powered by lithium-powered batteries and must follow the OSHA guidelines. With these types of batteries, there might be potential fire risks. Following the OSHA 1910.1200, we must have proper hazard communication. We will have proper labelling and follow the procedures for hazardous materials such as the lithium batteries [14]. We will also be charging the batteries at their rated amperage and ensuring that when not in use, the batteries are charged to a safer voltage for an extended period of time.

6 References

[1] “Robobrawl - Rules 2025.” Accessed: Mar. 06, 2025. [Online]. Available: https://robobrawl.illinois.edu/robobrawl/rules

[2] “Beginners Guide to LiPo Batteries,” FPV Freedom Coalition. Accessed: Mar. 06, 2025. [Online]. Available: https://fpvfc.org/beginners-guide-to-lipo-batteries

[3]“Analog to Digital Converter (ADC) - ESP32-S3 - — ESP-IDF Programming Guide v4.4 documentation.” Accessed: Mar. 06, 2025. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/v4.4/esp32s3/404.html

[4]“Motor Control Pulse Width Modulator (MCPWM) - ESP32-S3 - — ESP-IDF Programming Guide v5.4 documentation.” Accessed: Mar. 06, 2025. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/v5.4/esp32s3/api-reference/peripherals/mcpwm.html

[5]“AprilTag Introduction — FIRST Tech Challenge Docs 0.3 documentation.” Accessed: Mar. 06, 2025. [Online]. Available: https://ftc-docs.firstinspires.org/en/latest/apriltag/vision_portal/apriltag_intro/apriltag-intro.html

[6] *swatbotics/apriltag*. (Feb. 18, 2025). C. swatbotics. Accessed: Mar. 06, 2025. [Online]. Available: https://github.com/swatbotics/apriltag

[7] “OpenCV: Camera Calibration.” Accessed: Mar. 06, 2025. [Online]. Available: https://docs.opencv.org/3.3.1/dc/dbb/tutorial_py_calibration.html

[8]“Basic Pure Pursuit | Purdue SIGBots Wiki.” Accessed: Mar. 06, 2025. [Online]. Available: https://wiki.purduesigbots.com/software/control-algorithms/basic-pure-pursuit

[9]“15 | Combine a gyroscope and accelerometer to measure angles - precisely- YouTube.” Accessed: Mar. 06, 2025. [Online]. Available: https://www.youtube.com/watch?v=5HuN9iL-zxU&t=472s&ab_channel=CarbonAeronautics

[10]G. E. O. of M. and Communications, “Salary Averages.” Accessed: Mar. 06, 2025. [Online]. Available: https://ece.illinois.edu/admissions/why-ece/salary-averages

[11]“IEEE Code of Ethics.” Accessed: Mar. 06, 2025. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[12]“The Code affirms an obligation of computing professionals to use their skills for the benefit of society.” Accessed: Mar. 06, 2025. [Online]. Available: https://www.acm.org/code-of-ethics

[13]“Title 47 of the CFR -- Telecommunication.” Accessed: Mar. 06, 2025. [Online]. Available: https://www.ecfr.gov/current/title-47

[14]OSHA, “Law and Regulations | Occupational Safety and Health Administration,” Accessed: Mar. 06, 2025. [Online]. Available:https://www.osha.gov/laws-regs

Appendix A. Requirement and Verification Table

Table 1: Voltage Reader Subsystem Requirements and Verification Table

Requirements	Verification	Verification Status (Y or N)
The voltage reader on both ADCs must be accurate to within +/- 0.2V.	The reader will be tested with a power supply on a sweep from 1.4 - 2.2V (standard operation voltages) in steps of 0.1V, and results will be compared.	Y
The voltage reader on both ADCs must protect against a >3.3V input.	The reader will be tested with a power supply at a voltage of 5V, and the output must be within a safe value (> 3.3V).	Y

Table 2: PWM Input Subsystem Requirements and Verification Table

Requirements	Verification	Verification Status (Y or N)
The PWM input read must be accurate to within +/- 50µs.	The ESP32-S3 will read in a sample signal sweep from 1000µs to 2000µs in steps of 100µs from a signal generator, and results will be compared.	Y

Table 3: PWM Output Subsystem Requirements and Verification Table

Requirements	Verification	Verification Status (Y or N)
The PWM output read must be accurate to within +/- 50µs.	The ESP32-S3 will output a sweep from 1000µs to 2000µs in steps of 100µs, and results will be analyzed using an oscilloscope for accuracy.	Y

Table 4: IMU Subsystem Requirements and Verification Table

Requirements	Verification	Verification Status

		(Y or N)
The orientation measured will be accurate to +/- 10° the true angle of the robot.	The robot will be calibrated in a known orientation, then will spin around for 30 seconds at an approximate rate of 1 full rotation every 3 seconds. After that time, the orientation output will be compared to the true orientation of the robot.	N

Table 5: Remote Camera Subsystem Requirements and Verification Table

Requirements	Verification	Verification Status (Y or N)
The camera is capable of reading the 16'x16' play field from a fixed position.	The camera will be mounted in a specific location, and the robot will drive to all four corners of the arena. The robot's AprilTag should be visible at all corners.	Y
The camera can calculate the pose of both robots with an accuracy of within +/- 12 inches.	The robot and the opponent will be placed at specifically known locations within the arena, and the poses of the robots will be compared.	Y

Table 6: Autonomy Subsystem Requirements and Verification Table

Requirements	Verification	Verification Status (Y or N)
The simulated robot can drive directly at the opponent, and if the opponent is still, it can reach it within 5 seconds.	We will run the simulation, and place the robot at a set of given locations, with the opponent at another. Upon starting the simulation, the robot should always be able to navigate to the opponent.	Y
The robot completes a single autonomous pass within the 16ms timing window for each frame.	The autonomous system will run a timer at the start of the protocol, and the time it takes for each pass will be measured.	Y

Appendix B. Data Tables and Example Figures

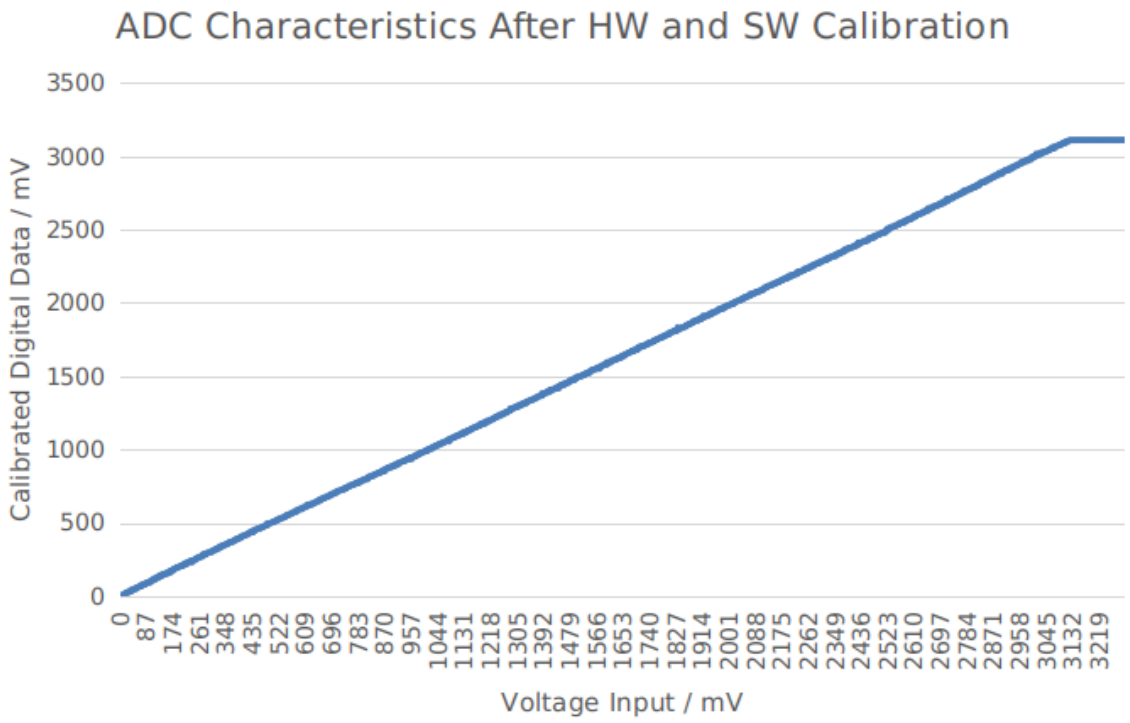


Figure 1: ESP32-S3 ADC Characteristics [3]

Channel #	Use
1	Left Drive Input
2	Right Drive Input
3	Weapon Input
4	Mode Select: -100 = Manual Override 0 = OFF 100 = Autonomous Mode
5	Safety Switch: 0 = OFF 100 = ON

Table 1: Channel Usage

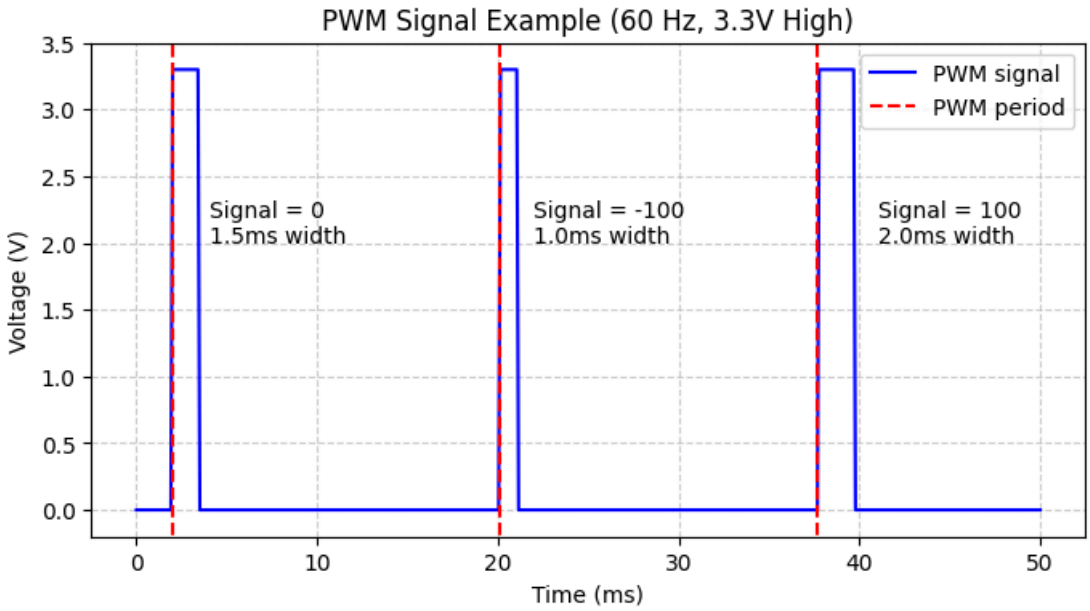
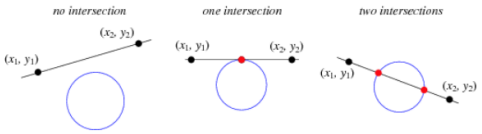


Figure 2: PWM Example for the ESC

Signal	Time (µs)	Number of cycles
PWM period cycle	16666.666	400000
-100 signal pulse width	1000	24000
0 signal pulse width	1500	36000
100 signal pulse width	2000	48000

Table 2: Cycle to time conversion table



An (infinite) line determined by two points (x_1, y_1) and (x_2, y_2) may intersect a circle of radius r and center (0, 0) in two imaginary points (left figure), a degenerate single point (corresponding to the line being tangent to the circle; middle figure), or two real points (right figure).

In geometry, a line meeting a circle in exactly one point is known as a tangent line, while a line meeting a circle in exactly two points in known as a secant line (Rhoad *et al.* 1984, p. 429).

Defining

$$d_x = x_2 - x_1 \tag{1}$$
$$d_y = y_2 - y_1 \tag{2}$$
$$d_r = \sqrt{d_x^2 + d_y^2} \tag{3}$$
$$D = \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} = x_1 y_2 - x_2 y_1 \tag{4}$$

gives the points of intersection as

$$x = \frac{D d_y \pm \operatorname{sgn}^*(d_y) d_x \sqrt{r^2 d_r^2 - D^2}}{d_r^2} \tag{5}$$
$$y = \frac{-D d_x \pm |d_y| \sqrt{r^2 d_r^2 - D^2}}{d_r^2}, \tag{6}$$

where the function $\operatorname{sgn}^*(x)$ is defined as

$$\operatorname{sgn}^*(x) \equiv \begin{cases} -1 & \text{for } x < 0 \\ 1 & \text{otherwise.} \end{cases} \tag{7}$$

The discriminant

$$\Delta \equiv r^2 d_r^2 - D^2 \tag{8}$$

therefore determines the incidence of the line and circle, as summarized in the following table.

Δ	incidence
$\Delta < 0$	no intersection
$\Delta = 0$	tangent
$\Delta > 0$	intersection

Figure 3: Line-intersection explanation. [8]

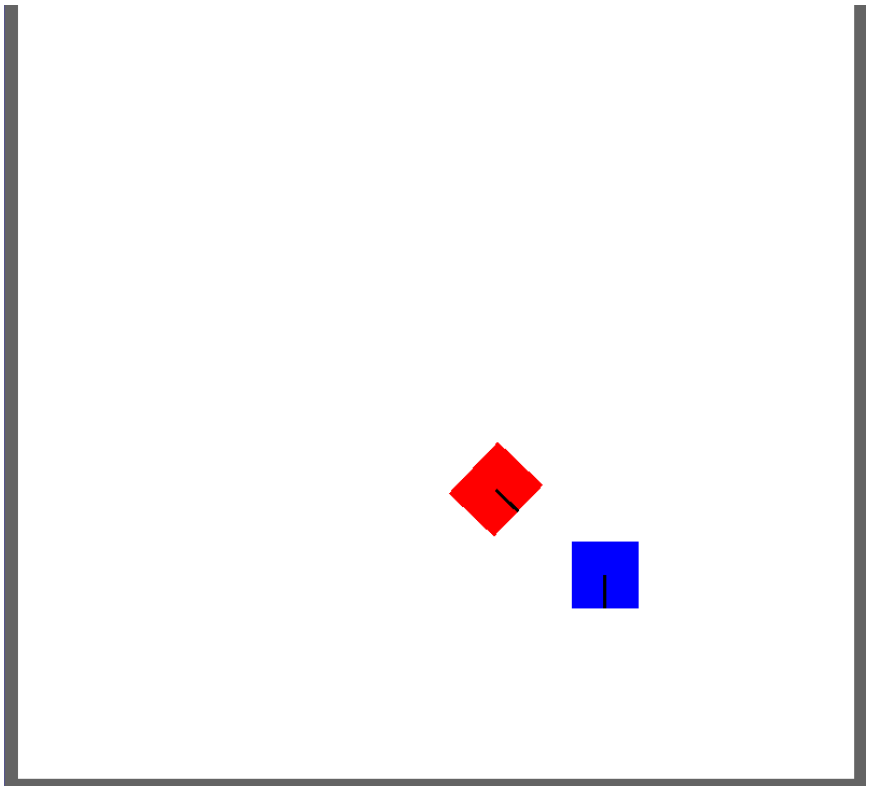


Figure 4: Pygame simulation example

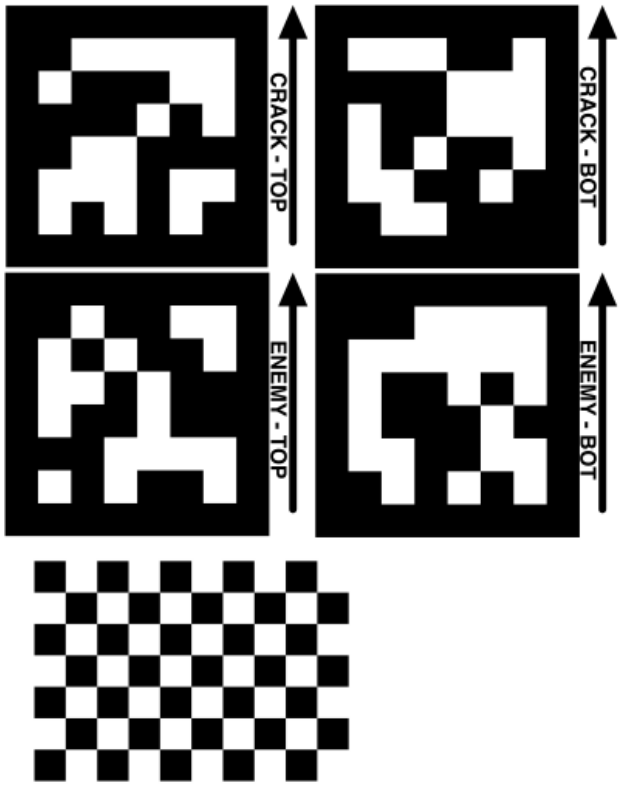


Figure 5: Example AprilTags, as well as the camera calibration tool

Appendix C. Flowchart and Block Diagram

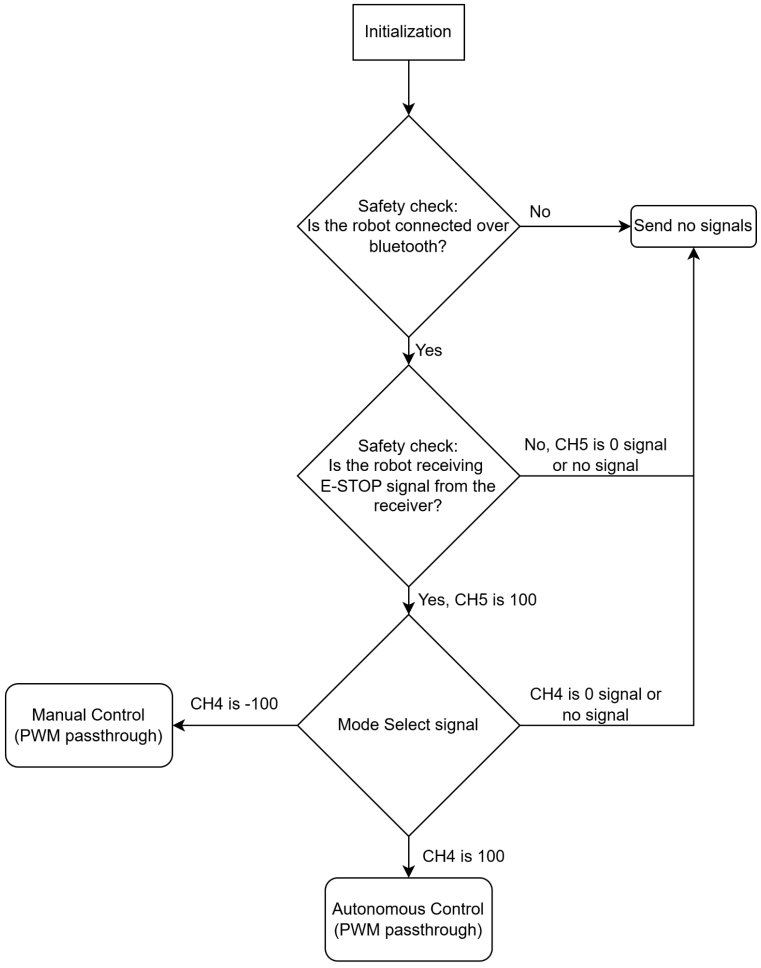


Figure 1: Initialization protocol

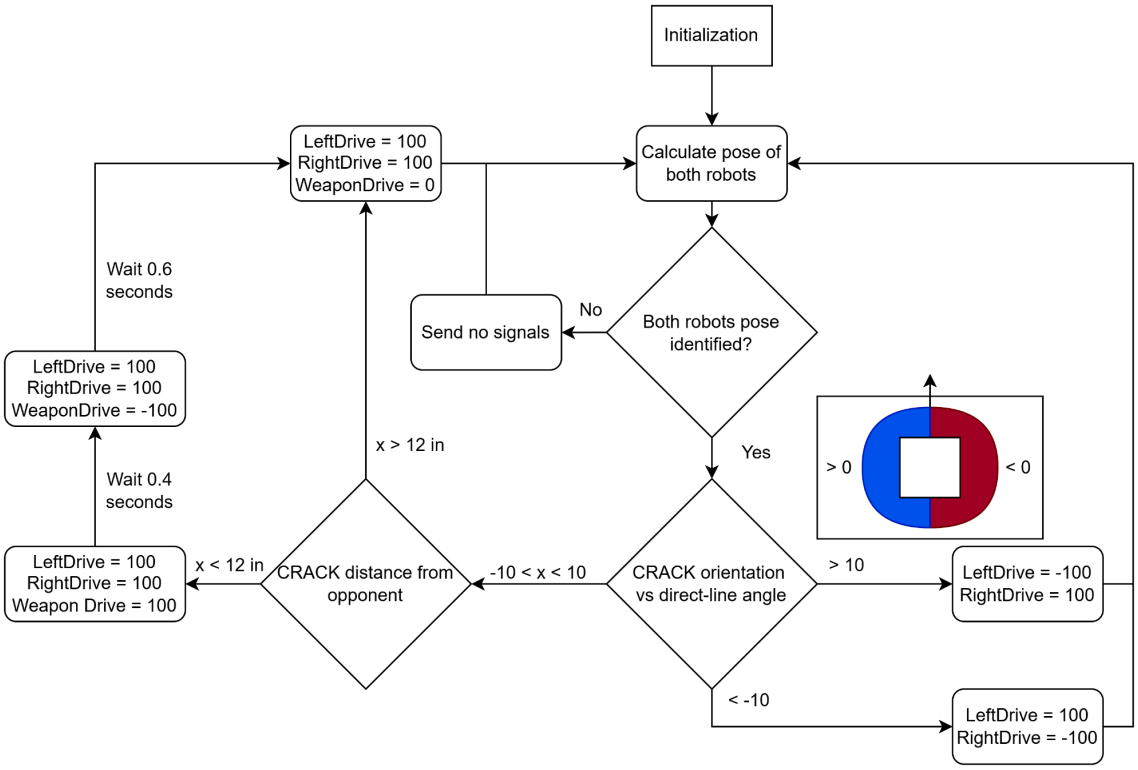


Figure 2: Autonomous protocol

Appendix D. Costs

Part	Manufacturer	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
22UF CAP (0603)	Samsung Electro-Mechanics	$0.08000	$0.04700	$0.40
0.1UF CAP (0603)	Samsung Electro-Mechanics	$0.08000	$0.00600	$0.09
3.3PF CAP (0603)	Vishay Vitramon	$0.49000	$0.30000	$3.00
1UF CAP (0603)	Samsung Electro-Mechanics	$0.08000	$0.01400	$0.14
10000PF CAP(0603)	Samsung Electro-Mechanics	$0.08000	$0.00900	$0.09
10PF CAP (0603)	Johanson Technology Inc.	$0.44000	$0.26300	$2.63
10 UF CAP (0603)	Samsung Electro-Mechanics	$0.08000	$0.02400	$0.24
LED	Rohm Semiconductor	$0.19000	$0.19000	$0.95
CONN 3POS	Sullins Connector Solutions	$0.33000	$0.28200	$4.23
CONN MICRO B	Molex	$0.92000	$0.92000	$2.76
CONN 7POS	JST Sales America Inc.	$0.21000	$0.21000	$1.05

CONN 5POS	Sullins Connector Solutions	$0.42000	$0.42000	$2.10
SS8050-G	Comchip Technology	$0.24000	$0.14800	$1.48
5M RES (1206)	Susumu	$0.32000	$0.28900	$2.89
10K RES(0805)	Stackpole Electronics Inc	$0.10000	$0.02500	$0.75
100K RES(0603)	Panasonic Electronic Components	$0.10000	$0.03500	$0.35
100kRES (0805)	YAGEO	$0.10000	$0.01400	$0.14
500K RES (0805)	Stackpole Electronics Inc	$0.12000	$0.13000	$0.65
3M RES (0603)	Panasonic Electronic Components	$0.10000	$0.03300	$0.33
1K RES (0603)	Panasonic Electronic Components	$0.10000	$0.03500	$0.35
SWITCH	Omron Electronics Inc-EMC Div	$0.57000	$0.57000	$2.85
AMS117	UMW	$0.68000	$0.68000	$3.40
LIS331	STMicroelectronics	$3.05000	$3.05000	$6.10
10k RES	Vishay	$0.527	$0.527	$10.54
10K RES	Vishay	$0.344	$0.344	$3.44
1k RES	Panasonic	$0.187	$0.187	$3.74
10UF CAP	TAIYO YUDEN	$0.256	$0.256	$2.56
Zener Diode	Taiwan Semiconductor	$0.296	$0.296	$2.96

ESD protection Diode	Littelfuse	$0.63	$0.63	$3.15
ESP32-S3-WROOM-1-N16R2	Espressif	$3.62	$3.62	$7.24
RC CAR	Amazon	$54.99	$54.99	$54.99
Camera	NexiGo	$18.99	$18.99	$18.99
Total				$144.58

Appendix E. Schedule

Week	Task	Person
Week 1 (1/20)	Discuss and brainstorm about the project	Everyone
Week 2 (1/27)	Design a block diagram and get the project approved	Jason
Week 3 (2/03)	Write proposal	Everyone
Week 4 (2/10)	Write a proposal and prepare for the proposal review	Everyone
Week 5 (2/17)	Design the schematic and PCB	Michael, Qing
Week 6 (2/24)	Modify the schematic and PCB	Everyone
Week 7 (3/3)	Finish the Design document	Everyone
	Work on the breadboard	Jason (order, assemble) Michael(assemble), Qing (assemble)
	First Round PCB Order 3/3	Everyone
Week 8 (3/10)	Start PCB assembly	Michael, Qing
	PCB Revision	Qing
	Prototype microcontroller	Jason
	Second Round PCB Order 3/13	Everyone
	Breadboard Demonstration	Everyone
Week 9 (3/17)	Spring Break	Everyone
Week 10	Program and Test (pure-pursuit,	Everyone

(3/24)	autonomous system)	
	Finalize microcontroller prototype	Michael
	Program and Test (voltage reader)	Michael
	Program and Test (PWM and IMU)	Qing
	Program and Test (Remote camera and April tag)	Jason and Qing
	PCB Revision	Michael
Week 11 (3/31)	**Robobrawl 4/4 - 4/5**	Everyone
	Modify any changes for the mock demo	Everyone
	Third Round PCB Order 3/31	Everyone
Week 12 (4/7)	Modify any changes for the mock demo	Everyone
	PCB Revision	Jason
	Fourth Round PCB Order 4/7	Everyone
Week 13 (4/14)	Prepare for the mock demo	Everyone
Week 14 (4/21)	Mock demo and final adjustment	Everyone
Week 15 (4/28)	Final demo	Everyone
Week 16 (5/5)	Final presentation	Everyone
	Final Paper 5/7	Everyone

Figure 23: Project Progression Schedule