Autonomous Featherweight (30lb) Battlebot

ECE 445 Design Document - Spring 2025

Group #43

Jason Mei (jasonm5), Qinghuai Yao (qyao6), Michael Ko (ykko2)

TA: Michael Gamota

Professor: Viktor Gruev

March 6, 2025

Table of Contents

1 Introduction

1.1 Problem

iRobotics, a 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 especially so in the world of combat robotics. We need to design our product 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’x16’ 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 Visual Aid

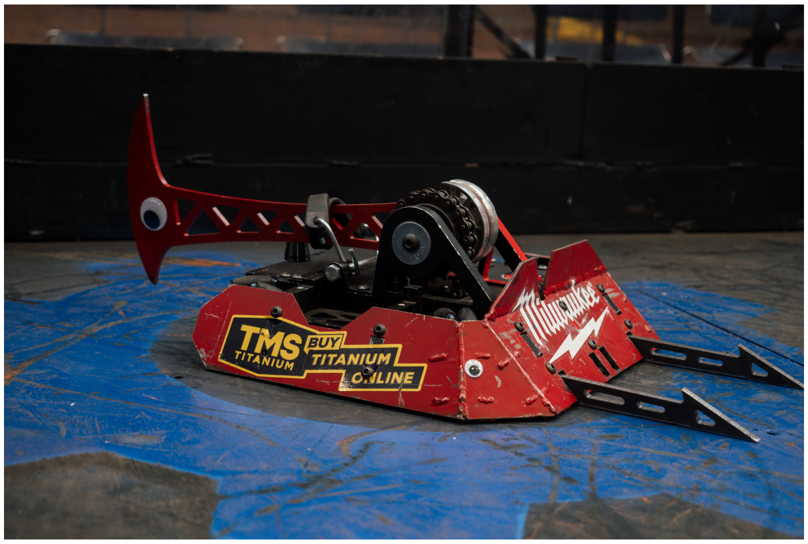


Figure 1: CRACK? V2

CRACK? V2 (henceforth known as “CRACK”) is the battlebot that we will be modifying to add autonomous capabilities. The robot itself has a hammer-axe that can fire down and damage opponents. Given the robot’s primary method of attack, it is quite unlikely that it can eject an opponent out of the arena, so its main option to winning a match is by either controlling the opponent and winning on a judge’s decision, or by puncturing the opponent’s top plate, and piercing a battery, which can trigger a battery fire. CRACK also has a standard 2 motor drive, with the back wheels connected to the front with a belt. In standard operation, the driver sends a total of 3 signals over 2.4GHz radio – left drive, right drive, and weapon – to the receiver. The receiver then converts those signals to PWM (pulse width modulation) signals, and sends them into the ESC (electronic speed controllers) on the robot. These ESCs then drive the motors, and allow the robot to function.

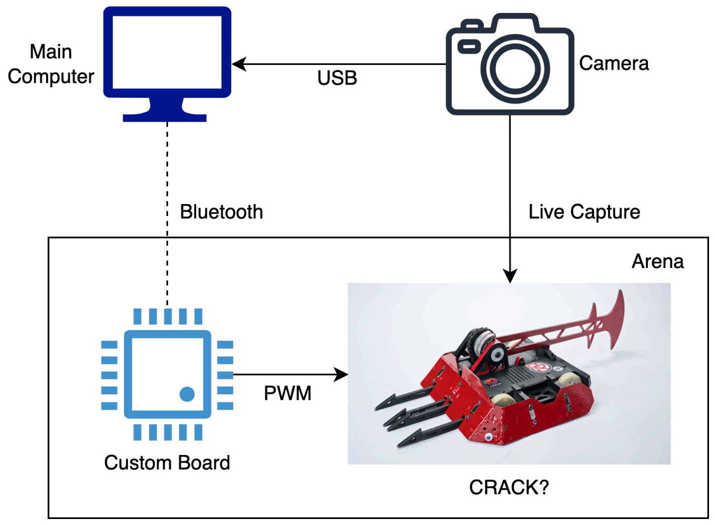


Figure 2: Visual Aid

The main computer will receive the visual information from the camera, performing live capture on the robot within the arena. After performing the calculations, it will send the information to the custom board over Bluetooth. The custom board will take in the PWM information from the receiver, and depending on the input from the controller and radio, the robot will respond accordingly.

1.4 High-Level Requirements

We would define a successful project upon the completion of a specific set of goals:

- The system will track both the robot and an opponent to within +/- 12 inches, and will calculate a path from the robot to the opponent, and send the PWM information to the custom board before the next frame update from the camera (within 16ms).

- The system must be able to shut off safely and immediately if there are ever any safety violations. For example, the robot should be able to E-STOP over Bluetooth, and will fail-safe safely if either Bluetooth or radio signal is lost, by exceeding a range of 100ft.
- On the whole, the robot should be able to track an RC car moving at 5MPH, and move to its location using the entire system.

2 Design

2.1 Block Diagram(s)

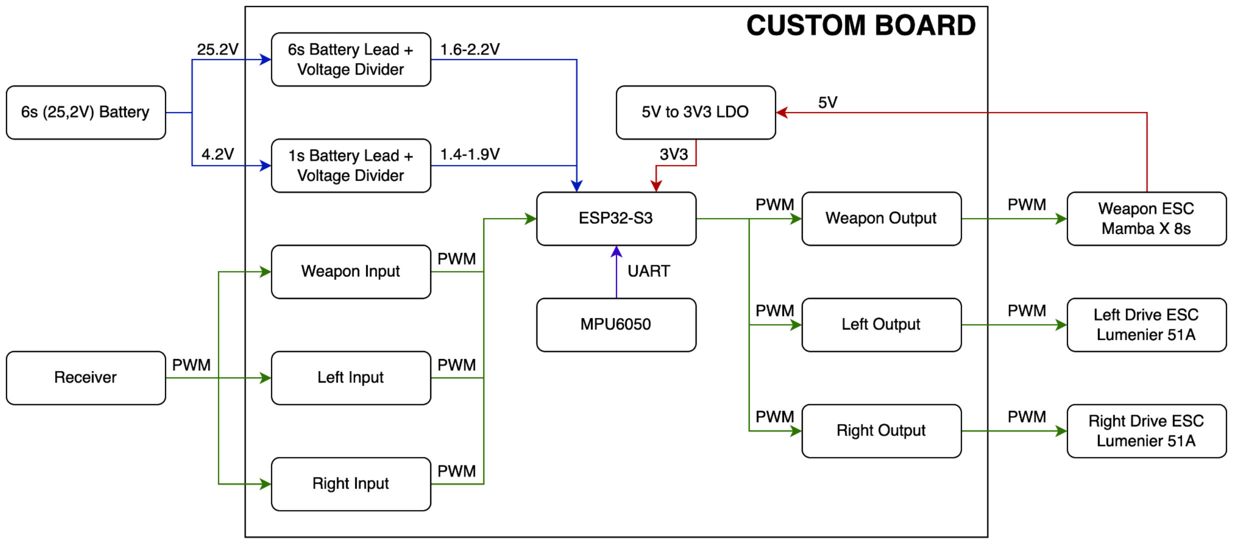


Figure 3: Custom Board block diagram

The physical custom board is relatively simple in design. 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 MPU6050, our IMU will send direct data over UART. 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.

2.2 Physical Specification

As the physical battlebot has already been built, we are required to meet a certain set of physical constraints that allow us to place the on-robot subsystem into the robot without significant changes to the actual robot itself, which would take an unnecessary amount of time. Additionally, upon completion of the final board, we plan to encase the PCB in a Duramic PLA+ case, which will protect the board from being damaged if the robot is thrown around during a match. We have rated the custom board to be a maximum size of 3”x2”x1” within the robot.

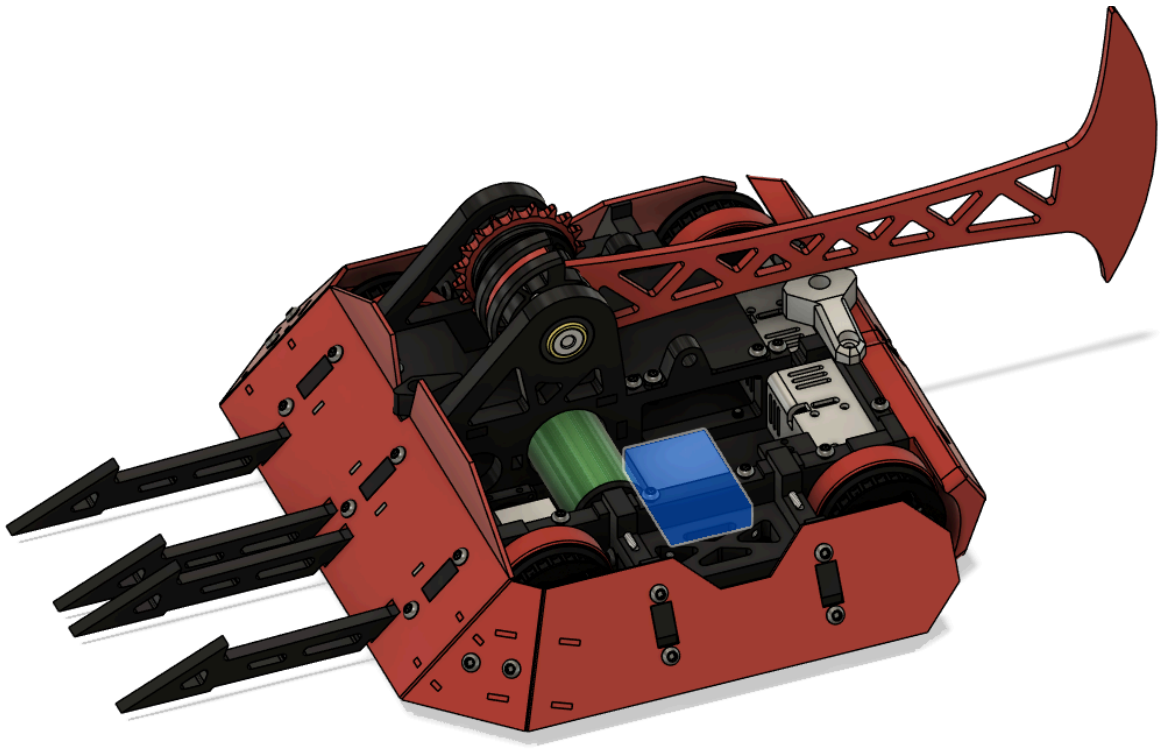
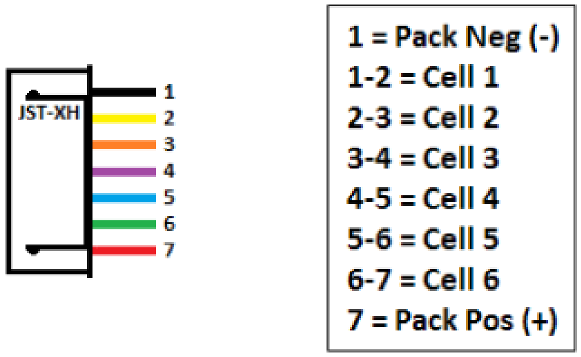


Figure 4: Internal sizing across the internals of CRACK. The custom board is highlighted in blue

2.3 On-robot Subsystem Specifications

2.3.1 Voltage Reader Subsystem

CRACK is powered by two 6s Tattu R-Line 1400mAh Lithium-Polymer batteries in parallel. The batteries’ 6s rating means that each battery has 6 LiPo cells in series, each with a per-cell voltage from 3.2 to 4.2V. One of the dangers of using a LiPo battery is that upon heavy usage, if cell voltages drop below 3.2V, the battery may become permanently damaged[2]. As a result, we want to keep a live update of the voltage of the batteries at all times. This will be done by reading the voltages of the last cell and the first cell to get two readings, and splitting the difference for better accuracy. We can do this by using the balance leads of the battery, which is shown with this pinout:



1 = Pack Neg (-)
1-2 = Cell 1
2-3 = Cell 2
3-4 = Cell 3
4-5 = Cell 4
5-6 = Cell 5
6-7 = Cell 6
7 = Pack Pos (+)

Figure 5: 6s Battery Lead pinout

Specifically, we will be using pin 7 as a 25.2V input, pin 2 as a 4.2V input, and pin 1 as GND. The battery itself uses a JST-XH connector, which allows for a direct connection to the custom board. Since both batteries are wired in parallel, we will be using the balance leads of only one battery, as the parallel connection should balance the positive output of the packs to each other.

Once we have those inputs, we cannot directly connect those voltages to the microcontroller, as 4.2V and 25.2V exceed the 3.3V pin ratings that are standard. We will instead use a voltage divider to pull the voltage down to one that we can send to the microcontroller. We are using resistors with high ohm ratings to minimize the current drawn from each cell. Below is a Falstad simulation of our voltage divider setup:

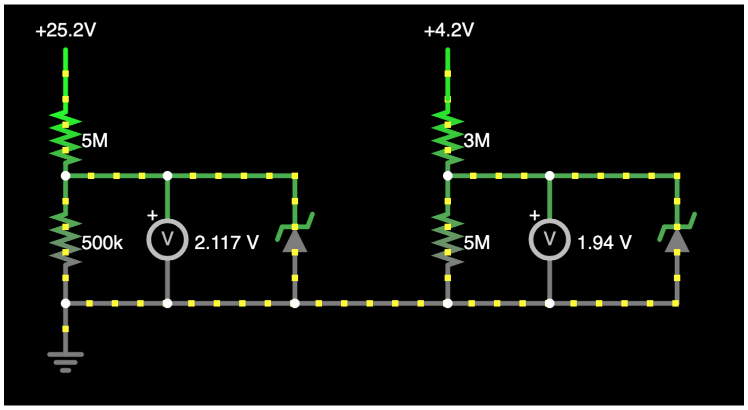


Figure 6: Voltage divider simulation.

At peak, the voltage divider draws 5.37 µA, which should be effectively nothing compared to the standard draw of the actual battery, which is approximately 40A continuous. A zener diode is included to protect the pins as well. If the battery somehow is able to overvolt, and send a voltage above 3.3V, the zener diodes will keep the voltage at 3.3V.

Both voltage dividers are then passed in as inputs to the ADC (analog-digital converter) of the microcontroller. The ADC of the ESP32-S3 has 12-bit (0-4095) accuracy [3], which will allow us to estimate the voltage to approximately 0.001V, or one milli-volt. For our initial design, we have split up the two inputs between the ADC1 and the ADC2 channels, to test the accuracy of both of the converters. We will also perform a hardware calibration (done by the microcontroller itself), and a software calibration to obtain a more accurate reading.

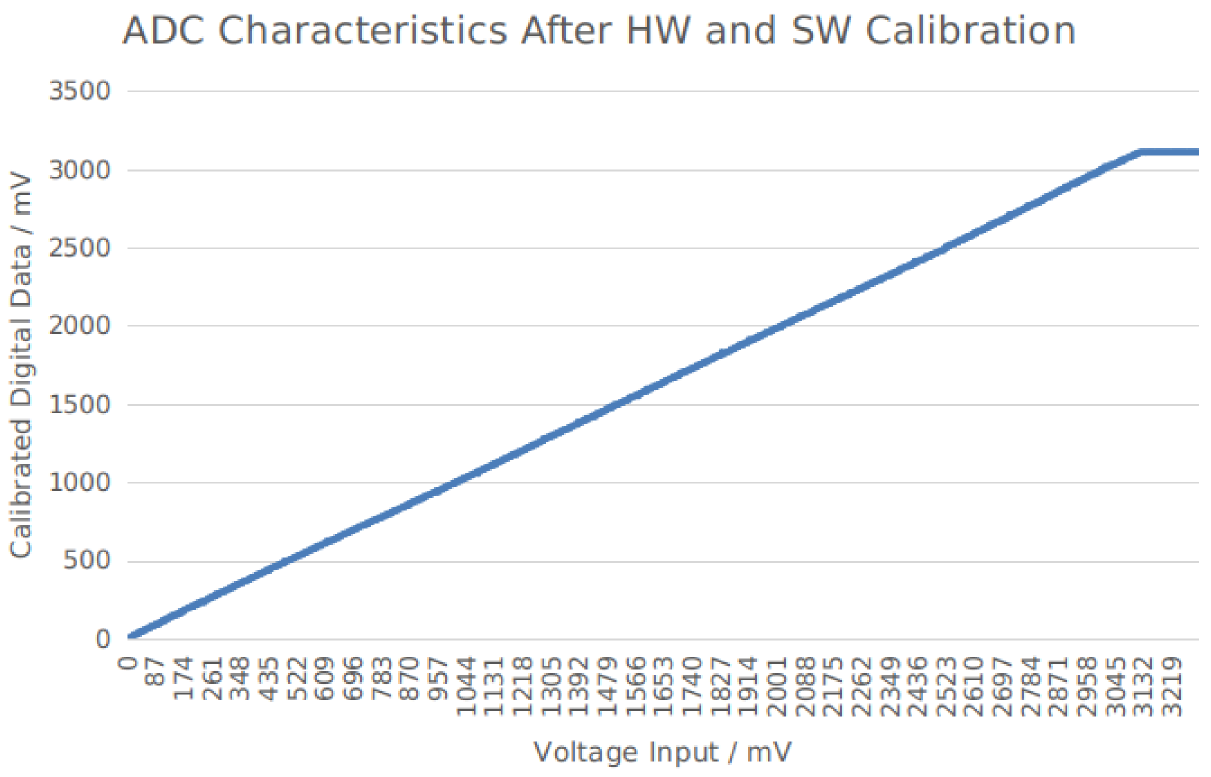


Figure 7: ESP32-S3 ADC Characteristics [3]

Table 1: Voltage Reader Subsystem Requirements and Verification Table

Requirements	Verification
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.
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).

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 will read at 50Hz, and from channels 1-5. Each channel's use is described below:

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

Figure 8: Channel Usage

The controller that we will be using, a Taranis QX7, has a variety of inputs that allow us to adjust how we drive the robot. For example, we may use a knob on the controller to adjust how fast the robot can move at any given time.

Table 2: PWM Input Subsystem Requirements and Verification Table

Requirements	Verification
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.

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.

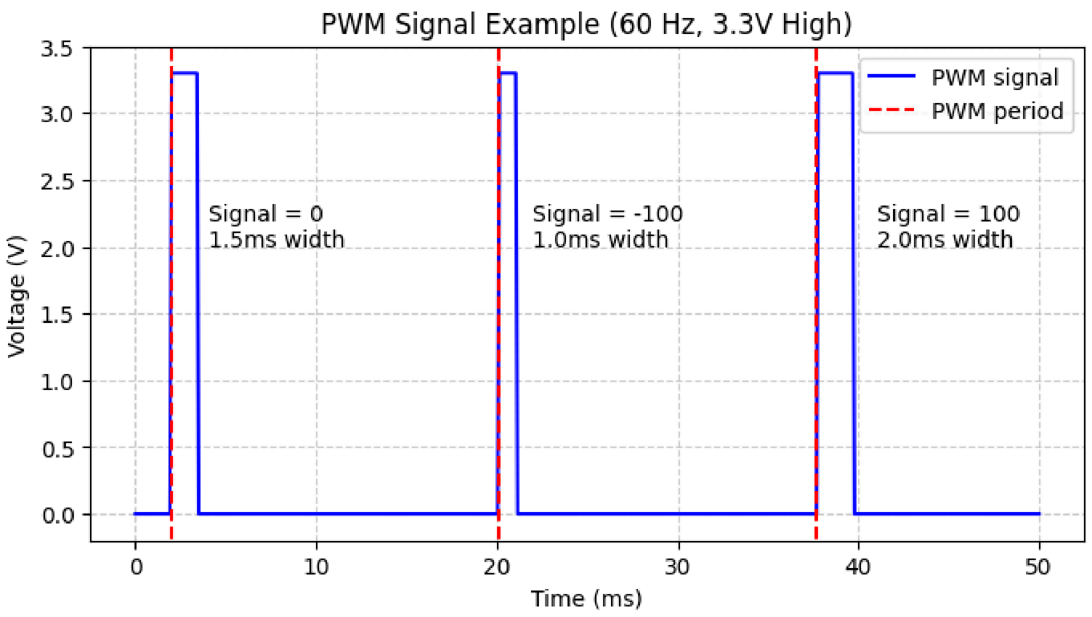


Figure 9: PWM Example for the ESC

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 a high precision for the pulse widths. The following table shows the conversion:

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

Figure 10: Cycle to time conversion table

Table 3: PWM Output Subsystem Requirements and Verification Table

Requirements	Verification
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.

2.3.4 IMU Subsystem

We will be using the MPU6050 IMU for accelerometer 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 whatever the camera is seeing. We will identify with 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.

Table 4: IMU Subsystem Requirements and Verification Table

Requirements	Verification
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.

2.3.5 Remote Camera and April Tag Subsystem

We will be using the NexiGo N980P USB Camera as our camera. It has a 1080p 60fps sensor, alongside a good 120° lens that should allow us to get a wide angle view of the entire arena. To attach the camera, we plan on using a tripod mounted in a specific location, so that the pose of the robot can be calculated without adjusting, and will be much more accurate. The camera connects over USB Type-A, which allows us to directly connect to the computer.

We will be using this information to identify both robots using AprilTags. AprilTags are visual markers that help cameras identify objects and their positions. [5] We are placing one on each side of the robot, on top and on the bottom. These fiducials interface with OpenCV and allow us to identify where the robot is at any given time.

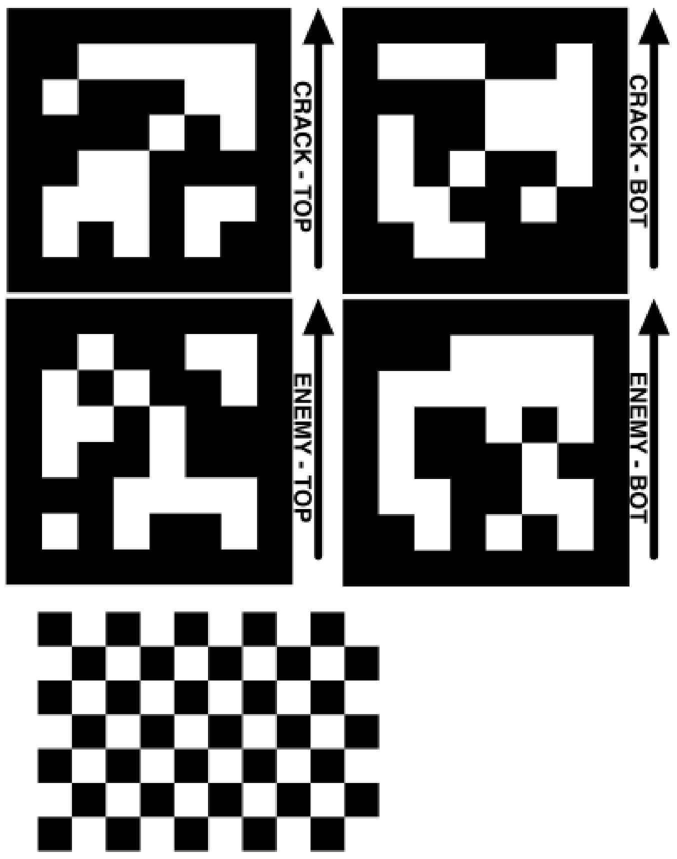


Figure 11: Example AprilTags, as well as the camera calibration tool

The AprilTags themselves are within the 36h11 family, and are standard parts chosen from the AprilTag library posted by the University of Michigan's AprilRobotics team. Using them, we can obtain a full pose (position and orientation) of both robots within the arena. We will be using the AprilTag Python library ported by berwin, which is a pypi port of the original library by swatbotics [6]. We will also be required to calibrate the camera using a standard checkerboard pattern, which will prevent radial distortion and tangential distortion. [7]

The formula for radial distortion is as follows:

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Additionally, there is a given tangential distortion, which occurs when the webcam's lens is not aligned parallel to the imaging plane, given with the following formulas:

$$x_{distorted} = x + [2p_1xy + p_2(r^2 + 2x^2)$$

$$y_{distorted} = y + [2p_2xy + p_1(r^2 + 2y^2)$$

This requires us to find five parameters, k_1, k_2, p_1, p_2, and k_3. We also have to account for a camera matrix, which includes the focal length and optical centers. Thankfully, OpenCV has provided us with a straight codebase that allows us to immediately calibrate the camera with known information.

Table 5: Remote Camera Subsystem Requirements and Verification Table

Requirements	Verification
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.
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.

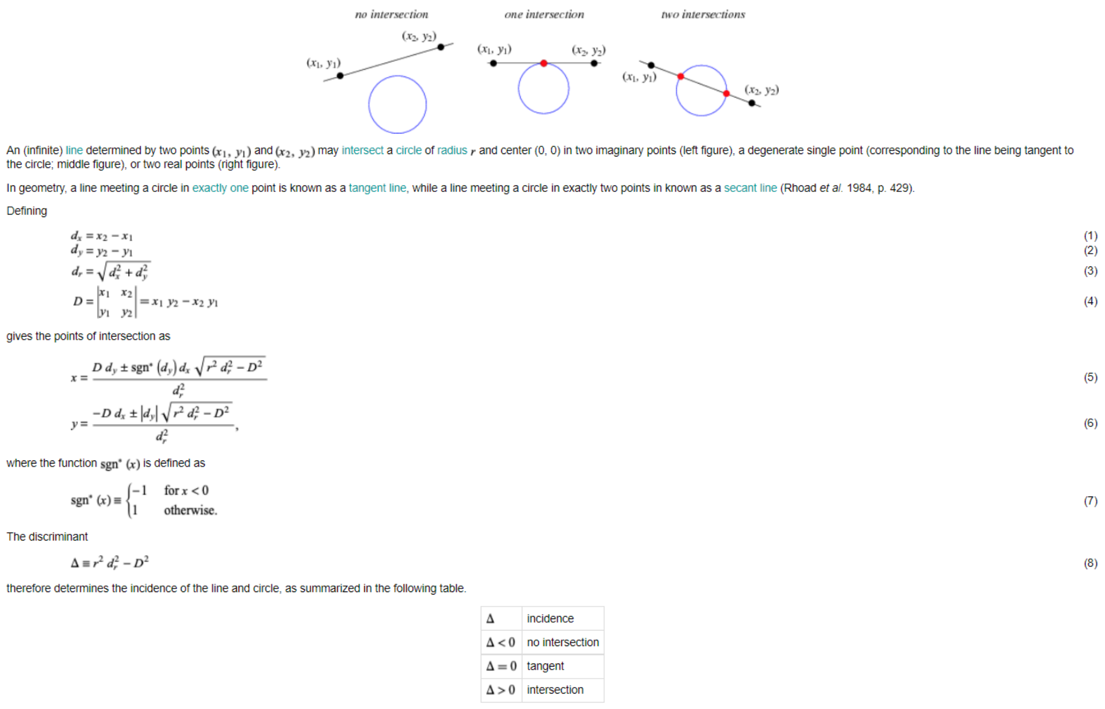
2.3.6 Autonomy Subsystem

The computer takes in multiple inputs:

- Live camera feed of the robots in the arena
- IMU information from the robot
- Previous location data (from previous calculations)

We plan on using a standard pure-pursuit algorithm for the initial pass. We will do this by obtaining the pose from the camera subsystem, as well as the IMU information of the robot.

Once we have two global positions, we will then be able to use line-circle intersection to identify the goal points for the robot. [7]



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

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

gives the points of intersection as

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

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

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

The discriminant

$$\Delta \equiv r^2\, d_r^2 - D^2 \quad (8)$$

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 12: Line-intersection explanation. [8]

Effectively, we will be trying to minimize the angle between the front of CRACK and the opponent, and minimize the distance between the two robots. Before we implement this design in the real world, we have constructed a pygame environment where we can simulate how the robot will behave in the world.

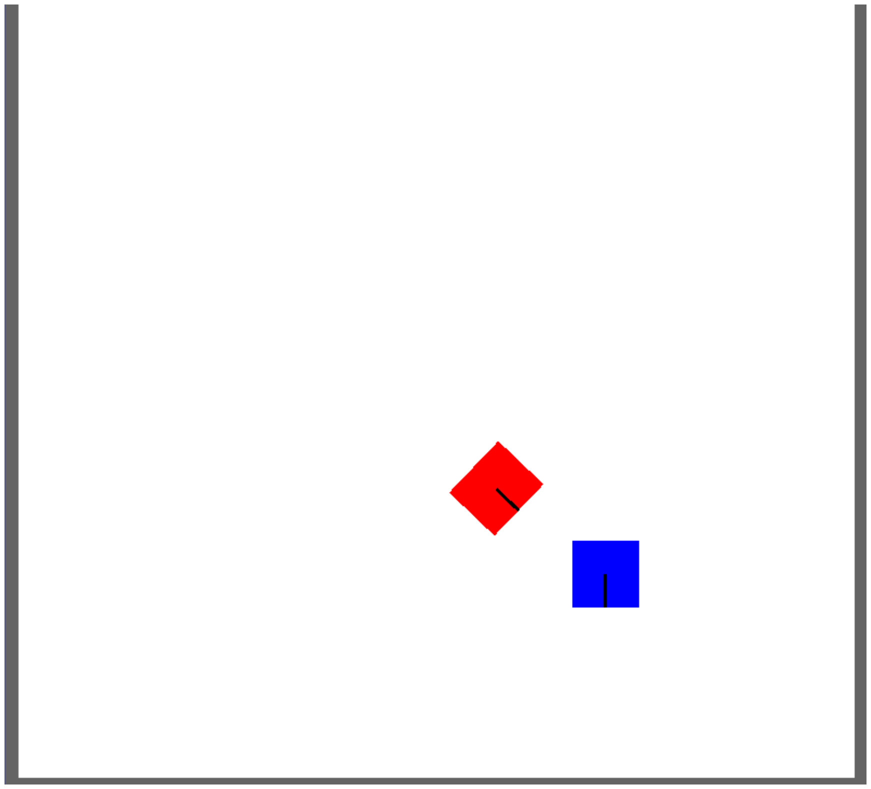


Figure 13: Pygame simulation example

In order to comply with most combat robotics rules, we will be forced to back away from our opponent if we manage to pin them for more than 10 seconds. We could either code this in our autonomy system, or through manual overrides. In our use case, we will still need to implement a manual override to allow the robot to drive to its corner before the fight, as well as to prevent the robot from attacking the other robot after the fight. In this state, all information sent over by the autonomy subsystem will be ignored, and the radio signals from the receiver will be passed through directly to the robot's ESCs.

The outline for the robot's autonomous initialization will follow this flowchart:

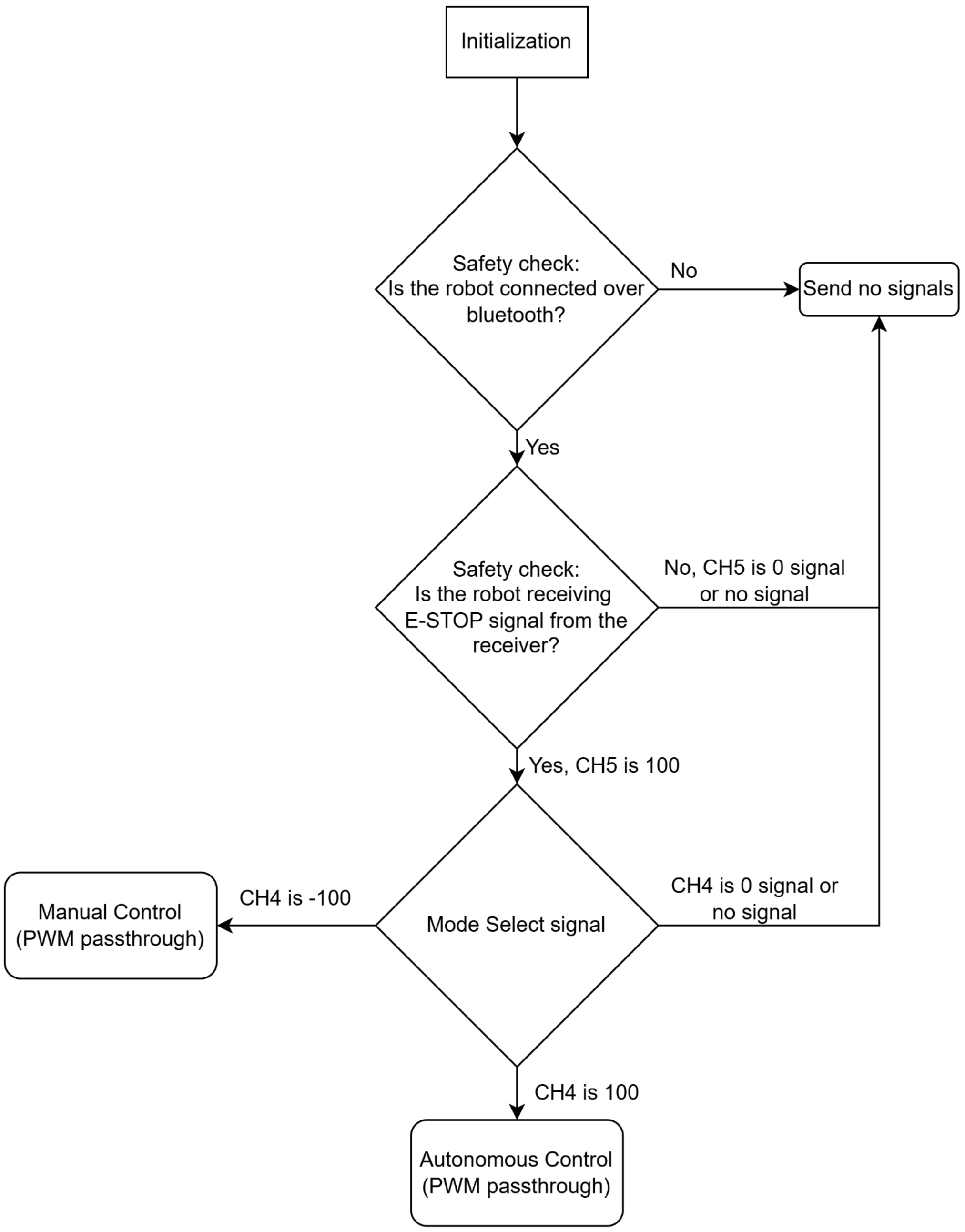

Initialization
Safety check: Is the robot connected over bluetooth?
No
Send no signals
Yes
Safety check: Is the robot receiving E-STOP signal from the receiver?
No, CH5 is 0 signal or no signal
Yes, CH5 is 100
Mode Select signal
CH4 is -100
Manual Control (PWM passthrough)
CH4 is 0 signal or no signal
CH4 is 100
Autonomous Control (PWM passthrough)

Figure 14: Initialization protocol

The robot will follow this procedure for actual autonomous control, assuming the robot is currently in the autonomous control mode:

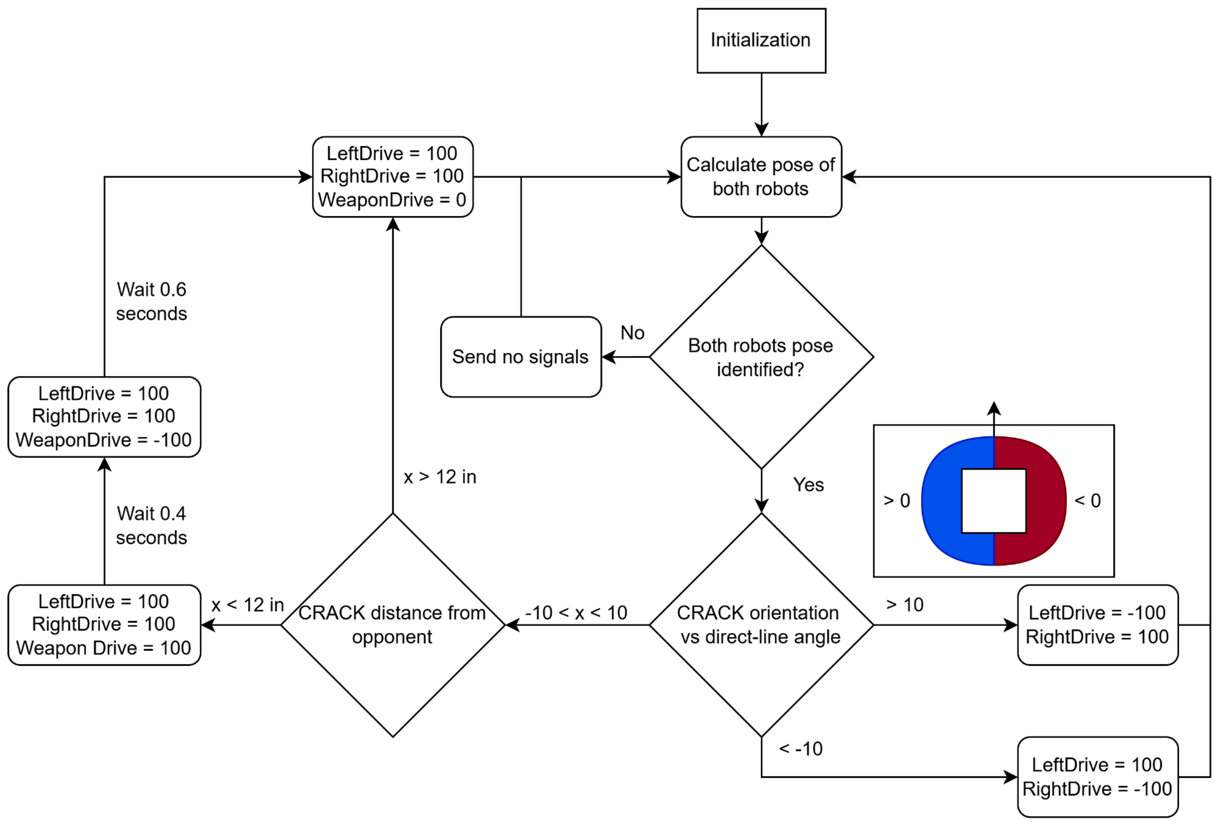


Figure 15: Autonomous protocol

In the future, we may look to using a reinforcement learning algorithm, utilizing machine learning within a custom OpenAI environment.

Table 6: Autonomy Subsystem Requirements and Verification Table

Requirements	Verification
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.
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.

2.4 Tolerance Analysis

We have identified two separate blocks that are critical to the completion of the project: the IMU accuracy, as well as the timing for the autonomous passes.

2.4.1 IMU Accuracy and Kalman Filters

One of the issues with using an IMU to obtain the angle of the robot over an extended period of time, is that slight errors with the measured accuracy of the IMU will add up.

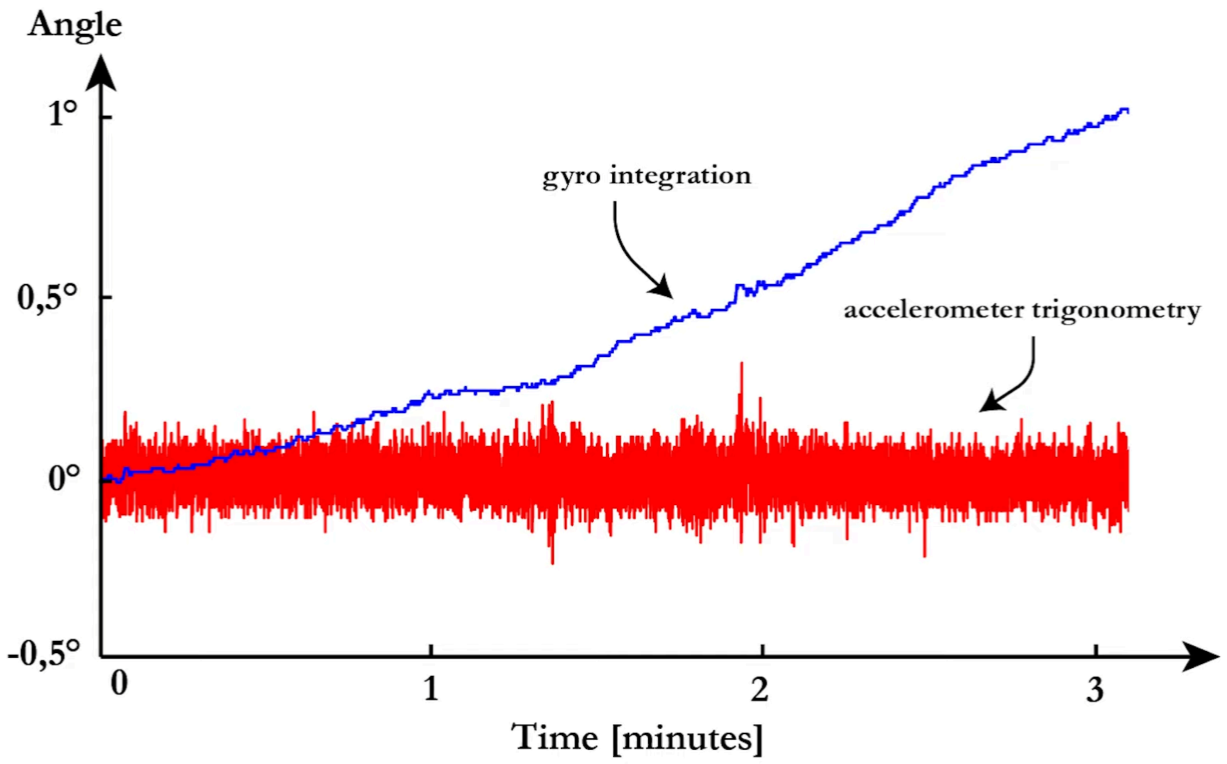


Figure 16: Evidence of error buildup over time [9]

Given enough time and enough error, this can cause all readings of the IMU to be wrong. One solution that can be identified is by occasionally recalibrating the IMU by moving a known position. This is a technique that is commonly used in high-school robotics competitions, but will not work in a battlebots match, as randomly calibrating to a corner in the middle of a fight will give up an attack, and isn't reasonable to expect out of our robot. Instead, we aim to use a Kalman filter to prevent losses from accruing.

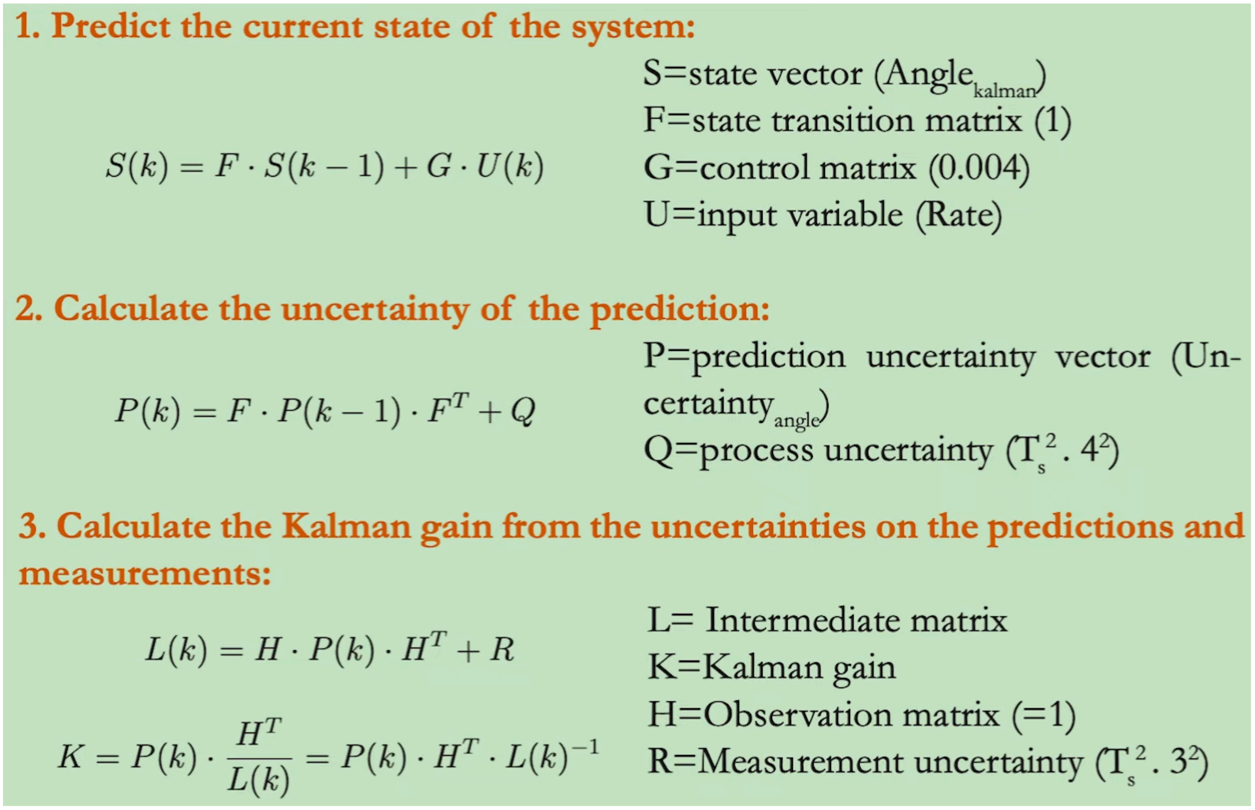
rotation rate [°/s] measured by the gyroscope

$$Angle_{pitch} = \int_0^{k \cdot T_s} \boxed{Rate_{pitch}} \cdot dt$$

$$Angle_{\underline{pitch}}(k) = Angle_{\underline{pitch}}(k-1) + \boxed{Rate_{\underline{pitch}}(k)} \cdot T_s$$

$$Angle_{\underline{kalman}}(k) = Angle_{\underline{kalman}}(k-1) + T_s \cdot \boxed{Rate(k)}$$

Figure 17: Formulas from the Kalman filter



1. Predict the current state of the system:

$$S(k) = F \cdot S(k-1) + G \cdot U(k)$$

S=state vector ($Angle_{kalman}$)
F=state transition matrix (1)
G=control matrix (0.004)
U=input variable (Rate)

2. Calculate the uncertainty of the prediction:

$$P(k) = F \cdot P(k-1) \cdot F^T + Q$$

P=prediction uncertainty vector ($Uncertainty_{angle}$)
Q=process uncertainty ($T_s^2 \cdot 4^2$)

3. Calculate the Kalman gain from the uncertainties on the predictions and measurements:

$$L(k) = H \cdot P(k) \cdot H^T + R$$

$$K = P(k) \cdot \frac{H^T}{L(k)} = P(k) \cdot H^T \cdot L(k)^{-1}$$

L= Intermediate matrix
K=Kalman gain
H=Observation matrix (=1)
R=Measurement uncertainty ($T_s^2 \cdot 3^2$)

Figure 18: Kalman gain calculation

After following the tutorial provided by Carbon Aeronautics, we can minimize the amount of losses from the MPU6050 over time.

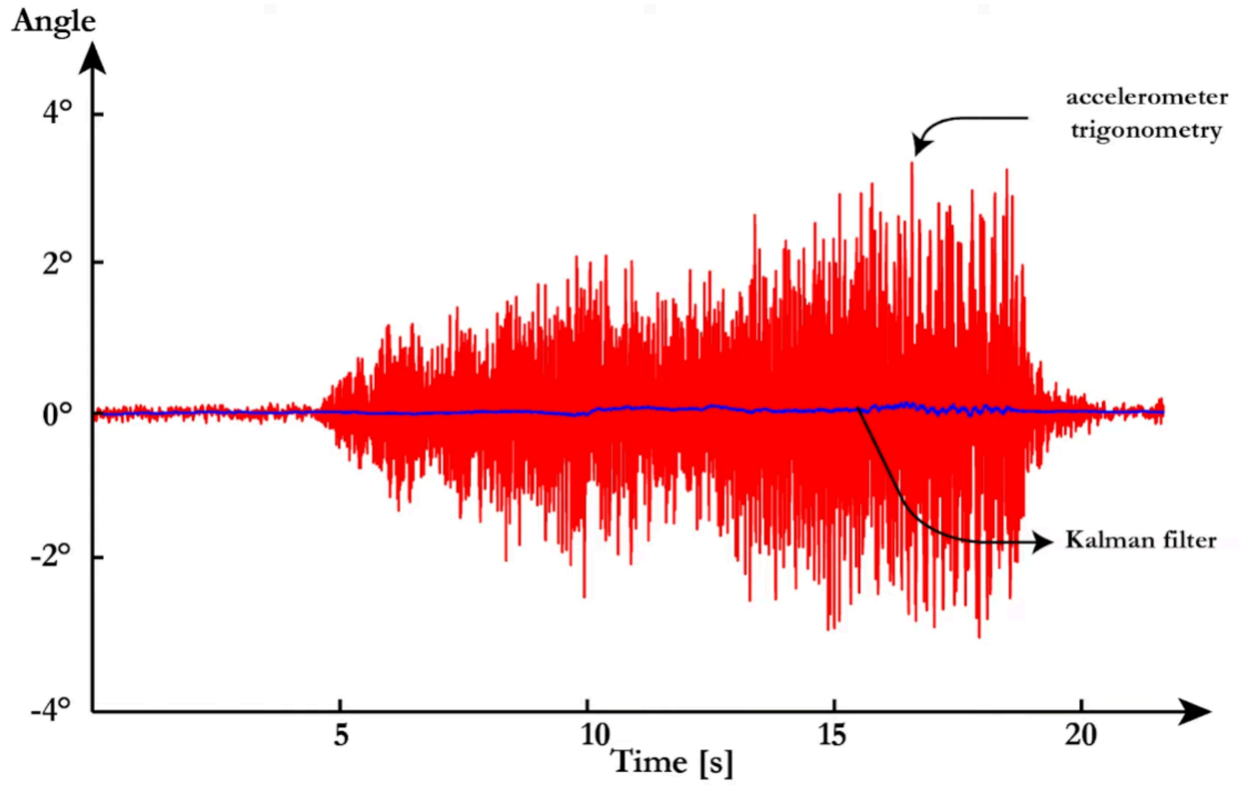


Figure 19: Plot of Kalman Filter minimizing losses over time.

2.4.2 Autonomous Pass Timing

For the meshing of all of the subsystems, the robot as a whole needs to fit within a certain cycle. We are currently limited by the frame rate of the camera, which runs at 60 frames per second. As a result, we need to do processing on the image as soon as we receive it, complete the autonomous protocol (calculate the commands to send to the robot), send the information over bluetooth, have the robot calculate the inputs, then send an ESC signal to all 3 ESCs, all before the next frame. The entire cycle must happen within 16ms, else we risk sending the robot incorrect information, which can cause the robot to behave erratically. The timing of each pass can be shown here:

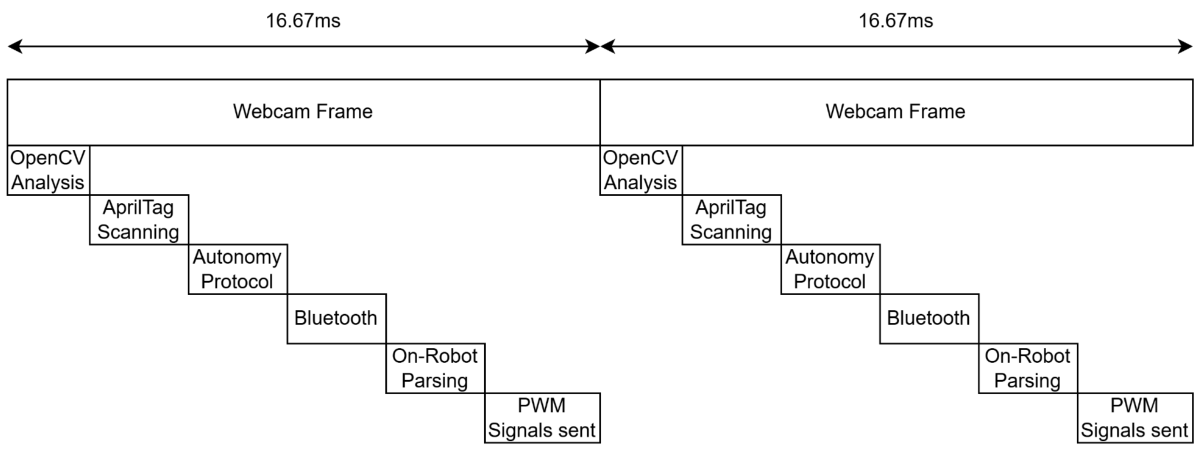


Figure 20: Full system pass timing (not to scale)

We are planning on doing timing tests throughout our tests, to ensure that the robot is capable of completing the pass quickly and efficiently. If we find that it is impossible to do, we can simply duplicate the sent signals, and have the autonomous pass the controls once every 33ms.

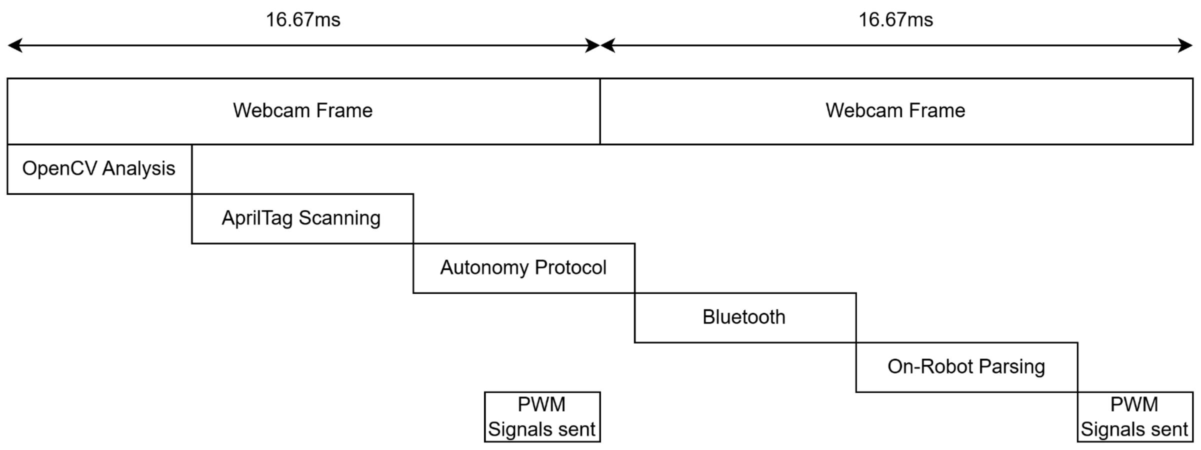


Figure 21: Alternative full system pass timing (not to scale)

3 Cost and Schedule

3.1 Cost Analysis

3.1.1 Labor

The average starting salaries 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

3.1.2 Parts

Description	**Manufacturer**	**Part #**	**Quantity**	**Unit price**	**Total price**	**Link**
CAP CER 22UF 10V X5R 0603	Samsung Electro-Mechanics	1276-1274-2-ND	2	$0.10000	$0.20000	Link
CAP CER 0.1UF 50V X7R 0603	Samsung Electro-Mechanics	1276-1000-2-ND	3	$0.08000	$0.24000	Link
CAP CER 3.3PF 250V C0G/NP0 0603	Vishay Vitramon	720-1318-2-ND	2	$0.49000	$0.98000	Link
CAP CER 1UF 25V X7R 0603	Samsung Electro-Mechanics	1276-1184-2-ND	3	$0.06000	$0.18000	Link
CAP CER 10000PF 50V X7R 0603	Samsung Electro-Mechanics	1276-1009-2-ND	1	$0.08000	$0.08000	Link

CAP CER 10PF 250V C0G/NP0 0603	Johanson Technology Inc.	712-QSCP251Q100G1GV001TTR-ND	2	$0.44000	$0.88000	Link
CAP CER 10UF 6.3V X5R 0603	Samsung Electro-Mechanics	1276-1119-2-ND	1	$0.08000	$0.08000	Link
DIODE ZENER 7.5V 150MW 603	Taiwan Semiconductor Corporation	1801-TSZU52C7V5RGGTR-ND	2	$0.36000	$0.72000	Link
LED RED DIFFUSED 1608 SMD	Rohm Semiconductor	SML-D12U1WT86TR-ND	1	$0.12000	$0.12000	Link
FUSE BOARD MNT 500MA 32VDC 0603	Panasonic Electronic Components	P15127TR-ND	1	$0.32000	$0.32000	Link
CONN HDR 3POS 0.1 TIN PCB	Sullins Connector Solutions	S7001-ND	6	$0.33000	$1.98000	Link
CONN RCPT USB2.0 MICRO B SMD R/A	Molex	WM1399TR-ND	1	$0.92000	$0.92000	Link
CONN HEADER VERT 7POS 2.5MM	JST Sales America Inc.	455-B7B-EH-A-ND	1	$0.26000	$0.26000	Link
CONN UMC RCPT STR 50 OHM SMD	Molex	WM5587TR-ND	1	$0.74000	$0.74000	Link
FIXED IND 2.2NH 1.5A 35 MOHM SMD	EPCOS - TDK Electronics	495-1835-2-ND	1	$0.28000	$0.28000	Link

TRANS NPN 25V 1.5A SOT-23-3	Comchip Technology	641-1790-2-ND	2	$0.24000	$0.48000	Link
RES 5M OHM 0.1% 1/4W 1206	Susumu	408-2038-2-ND	2	$0.35000	$0.70000	Link
RES 10K OHM 1% 1/8W 0805	Stackpole Electronics Inc	RMCF0805FT10K0TR-ND	6	$0.10000	$0.60000	Link
RES SMD 100K OHM 1% 1/10W 0603	Panasonic Electronic Components	P100KHTR-ND	1	$0.10000	$0.10000	Link
RES 100K OHM 1% 1/8W 0805	YAGEO	311-100KCRTR-ND	1	$0.10000	$0.10000	Link
RES SMD 500K OHM 0.1% 1/8W 0805	Stackpole Electronics Inc	738-RNCF0805BTE500KTR-ND	1	$0.13000	$0.13000	Link
RES SMD 3M OHM 5% 1/10W 0603	Panasonic Electronic Components	P3.0MGTR-ND	1	$0.10000	$0.10000	Link
RES SMD 1K OHM 1% 1/10W 0603	Panasonic Electronic Components	P1.00KHTR-ND	1	$0.10000	$0.10000	Link
SWITCH TACTILE SPST-NO 0.05A 24V	Omron Electronics Inc-EMC Div	SW415-ND	2	$0.57000	$1.14000	Link

IC REG LINEAR 3.3V 1A SOT-223-3L	UMW	4518-AMS1117-3.3TR-ND	1	$0.68000	$0.68000	Link
SPARKFUN SERIAL BASIC BREAKOUT -	SparkFun Electronics	1568-1972-ND	1	$9.95000	$9.95000	Link
Multiple Function Sensor Modules MEMS 3Axis Gyroscope 3-axis Accelerometer	Olimex Ltd	909-MOD-MPU6050	1	$8.58	$8.58	Link
IC RF TXRX+MCU BLE 56QFN	Espressif Systems	1904-ESP32-S3R8TR-ND	1	$2.28000	$2.28000	Link
CRYSTAL 16.0000MHZ 10PF SMD	Diodes Incorporated	FL1600080TR-ND	1	$0.56000	$0.56000	Link

Figure 22: Lists and Costs for the Part Components

3.1.3 Sum

The total cost: $33.48 + $6,500 = $6533.48

3.2 Schedule

Week	**Task**	**Person**
Week 1 (1/20)	Discuss and have a brainstorm about the project	Everyone
Week 2 (1/27)	Design block diagram and get the project approved	Jason
Week 3 (2/03)	Write proposal	Everyone
Week 4 (2/10)	Write proposal and prepare 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 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 (3/24)	Program and Test (pure-pursuit, autonomous system)	Everyone

	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 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

4 Ethics and Safety

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, 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.

Additionally, in compliance with ACM codes 1.4, 1.6, and 1.7, our project must align with all of the Robobrawl competition requirements to ensure fairness and transparency [12]. Even though our robot incorporates autonomous capabilities, we must ensure it provides no unfair advantages over competing robots.

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 with 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 ensure that when not in use, the batteries are charged to a safer voltage for an extended period of time.

For our machine learning implementation, we will employ reinforcement learning techniques and conduct thorough testing in simulated environments prior to actual deployment. This testing will progress through controlled settings before advancing to real-world applications. Regarding hardware reliability, we will conduct systematic inspections of all sensors, cameras, and power components to prevent system failures.

5 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