Fire Flighters Final Report

Date: May 5th, 2025
To: Sara Lego, and Vítor Valente
From: Mark Yazemboski, Jaisel Singh, Achyutan Srinivasan, Trusha Patel, Ankur Ghorai
Subject: Final Report
Course: AERSP 403B

1. Summary

With the rapid advancements in autonomous drone technology, drones are being used in almost every industry. This project focuses on designing a drone to assist firefighters in combating wildfires. The drone must autonomously execute all mission phases, including taking off, locating an active fire, deploying a suppressant, and landing safely. The drone must weigh less than 1 kg, carry 25 grams of fire suppressant, and transmit data to a ground station for monitoring. The mission flight profile begins with the drone taking off to a height of 2 feet before navigating to the fire zone and cruising at 10 feet. A rectangular spiral search pattern is used for fire detection, allowing thermal sensors to cover the area efficiently. Upon locating the fire, the drone faces the fire and hovers above it for 5 seconds, descends to 3 feet, and deploys the suppressant. The mission concludes with a controlled landing 1.5 meters from the fire. At the system's core is an Arduino-Uno, which processes data from sensors, including GPS, magnetometer, barometer, and sonar. It computes control outputs for the motors and communicates with ESCs, which send three-phase power signals to the brushless motors. A wireless receiver enables manual override, enhancing operational flexibility.

The drone's modular and replaceable design incorporates a 3D-printed frame optimized for durability and simplicity. The arms feature a C-beam structure to prevent deflection under load while providing space for concealed wiring. The drone's two housing levels separate key components: the lower level contains the battery, power distribution board, and thermal camera, while the upper level houses the Arduino and Pozyx module. A servo-controlled payload mechanism is mounted beneath the bottom layer for suppressant deployment. The compact drone measures 13 inches in span and 7.5 inches tall.

The software suite includes a waypoint setter to manage navigation, a finite state machine to oversee mission phases, and a Kalman filter to integrate sensor data for precise positioning. Control systems consist of five PID loops: one for position, velocity, acceleration, angle, and angular velocity. Thermal camera data is processed to identify fire locations, isolating hot pixels and calculating a centroid to direct the drone's movements.

In the final implementation, the team successfully conducted hand-carry tests that validated the payload delivery mechanism and navigation system. The ground control station performed well, displaying real-time data including position, orientation, and waypoint tracking. The drone achieved stable flight—taking off and maintaining a controlled hover for approximately five seconds. While processing limitations of the Arduino prevented simultaneous operation of autonomous flight and control systems, the team demonstrated the core functionality of their design through separate testing of these components.

Figures

Tables

2. Introduction

The increasing prevalence and intensity of wildfires pose significant threats to human life, property, and ecosystems. With the limitations of conventional firefighting methods-particularly in remote or hazardous areas-there is a pressing need for innovative solutions to address this growing challenge. For this preliminary design capstone course, our mission is to develop an autonomous drone that can provide rapid and precise wildfire suppression. This project requires prototyping a drone system that can navigate autonomously to a target area ("wildfire location"), assessing conditions using onboard sensors, and deploying a small payload mechanism that takes place of the fire suppression material. By optimizing the navigation, sensing, and delivery systems, our design aims to enhance the effectiveness and safety of suppressing wildfires.

The mission entails creating a quadcopter with strict constraints: total weight must be under 1 kilogram; the payload cannot exceed 25 grams and the total build must be within the $500 budget. The drone must operate autonomously, relaying real-time sensor data and operational statuses – from takeoff to safe landing, the entire flight must be unassisted, with no manual intervention. The drone must also hover over the identified target zone ("active fire") for at least 5 seconds before dropping its payload, and navigating 1.5 meters away for safe landing. These constraints challenge our team to apply engineering principles creatively and rigorously, balancing technical innovation with practical feasibility.

We recognize the importance of adhering to competition rules, including the PSU Policy SY45 and the Flight Readiness Review requirements. By focusing on innovation within these parameters, this mission represents not only a technical challenge but also a real-world scenario to deliver a reliable, cost-effective solution that advances wildfire suppression technology.

3. Mission Overview

The mission for this project focuses on designing an autonomous drone system that is capable of performing wildfire suppression. The drone must complete a series of specific tasks, including take-off, navigating to the target zone based on provided coordinates, identifying the zone using onboard sensors, hovering over the fire, releasing the payload, and lastly returning to the landing location that is 1.5 meters away from target zone.

3.1. Mission Requirements

An important part of any large system is a comprehensive list of system requirements. These requirements ensure that engineers can evaluate whether the design they are building aligns with the project's objectives and constraints. By clearly defining the mission requirements, the design process becomes more structured and traceable, allowing for systematic verification and validation throughout the development lifecycle. This section outlines the mission requirements for the drone, serving as the foundation for the design and implementation phases. Our team broke the mission requirements into 7 overarching umbrellas the first of which is shown below in Table 1.

Table 1: Mission Requirements-FR1

Number	Requirement	Source
FR1	The UAV shall be able to take off autonomously and navigate towards the wildfire zone.	RFI
FR1.1	The team will perform a flight systems check that ensures that the drone is ready to fly safely	FR1
FR1.12	Will make sure motors spin smoothly, batteries are in good condition and connected properly, propellers are secured, and remote control is fully charged.	FR1.1
FR1.2	Create a logic in flight controller software that shall make sure the drone takes off and searches for the fire	FR1
FR1.21	Code shall have a separate mode for take off that will set upward acceleration and velocity and cruising altitude.	FR1.2
FR1.22	Code shall have a separate mode that will set waypoints and have the drone follow a predetermined search pattern.	FR1.2

FR1 - Our first parent requirement FR1 is the first major step in our mission, the ability to autonomously take off and navigate to the known wildfire zone. In order to achieve this a few sub-requirements are needed. Firstly to ensure the safety of any bystanders a flight systems check is needed (FR1.1). This will include various motor, sensor, and battery checks to make sure that the drone is operating properly and is not a danger to its environment. The next sub-requirement is the need for the drone to actually operate autonomously (FR1.2). This means that we need to create logic for the drone's controller that will let it automatically take off and search for the fire. Then once we are stabilized in the air the flight computer navigates to an imputed waypoint to reach the wildfire zone.

Once in the zone it will initiate a set search pattern until it receives a high enough heat signal. It will then adjust its course and fly until it hovers over the highest temperature reading.

Table 2: Mission Requirements-FR2

FR2	The UAV shall be able to release the payload directly on the detected target (wildfire zone) from the necessary altitude.	RFI
FR2.1	The UAV shall hover over the wildfire zone for at least 5s.	FR2
FR2.2	UAV shall descend to the drop altitude above the wildfire zone, must not go lower than 1m from the fire.	FR2
FR2.3	Will integrate the target zone in code that UAV sensors shall detect to drop the payload.	FR2
FR2.31	This mode shall calculate the angle the drone needs to rotate to face the fire	FR2.3
FR2.32	This mode shall figure out the flight path to the center of the fire zone.	FR2.3
FR2.33	This mode shall command the drone to release the payload	FR2.3

FR2 - The second major requirement is that the UAV will be able to release the payload accurately on the target. This is imperative to compleating the mission as if this does not happen the entire mission falls apart. To achieve this the UAV needs to additionally be able to descend to the proper attitude above the wildfire zone to drop the payload. The drone also needs to be able to be able to use its sensors to locate the fire. In order to do this. Once it does this it will calculate the direction it needs to travel to get to the center of the fire and then will drop the payload once it restabilizes over the fire by hovering for about 5 seconds. This also will need to transmit some data back to the ground station about the payload being successfully dropped.

Table 3: Mission Requirements-FR3

FR3	UAV shall navigate 1.5m away from the fire and prepare for landing.	RFI
FR3.1	Will calculate the flight path after successfully dropping the payload	FR3
FR3.12	Will select a flight planning software compatible with drone, and log flight information into the software.	FR3.1
FR3.13	This mode shall raise the drone back up to cruising altitude	FR3.1
FR3.2	This mode shall pick a random direction away from the fire zone and face the drone in that direction	FR3
FR3.3	This mode shall plot a waypoint 1.5m away from the fire zone in the direction above and fly to it.	FR3
FR3.31	This mode shall lower the drone safely at a certain velocity till it reaches the ground.	FR3.3

FR3 - Post-Delivery Navigation and Landing: Following successful fire suppression payload deployment, the drone must execute a safe evacuation sequence. The navigation system then generates a randomized exit waypoint to about 1.5 meters from the fire zone, positioning the drone in a safe zone while maintaining sensor coverage of the target area for effectiveness monitoring. The landing sequence involves a precisely controlled descent rate.

Table 4: Mission Requirements-FR4

FR4	Will communicate all sensory information and data back to the control/ground station.	RFI
FR4.1	Will record telemetry data, fire location, payload drop, and landing drop	FR4
FR4.2	Control station shall have wireless communication modules	FR4
FR4.3	Control station shall have computing/storage units with large enough capacity	FR4

FR4 - Communication Systems: The drone maintains continuous data transmission to the ground control station throughout the mission (FR4). The system records comprehensive telemetry data including fire location, payload deployment confirmation, and landing information (FR4.1). The ground station features robust wireless communication modules (FR4.2) and sufficient computing/storage capacity (FR4.3) to manage all mission data effectively.

Table 5: Mission Requirements-FR5

FR5	Drone Structure will be able to carry 4 motors, a battery and all related electronics with < 2mm deflection	RFI
FR5.1	Drone frame shall be strong enough to handle multiple uses in a short amount of time with minimal repairs	FR5
FR5.2	Drones Landing legs will be able to absorb a fall from 1m on a concrete surface without any damage to the landing legs.	FR5
FR5.3	Drones center of mass will be 2 inches below the plane of the propellers.	FR5
FR5.4	Payload mechanism Shall be reliable and easily resettable simple and lightweight.	FR5
FR5.41	Payload mechanism shall be capable of dropping a clay type payload without the payload sticking to the mechanism.	FR5.4

FR5 - Structural Design: The drone's structure accommodates four motors, battery, and electronics while maintaining less than 2mm deflection (FR5). The frame is engineered for multiple rapid deployments with minimal maintenance (FR5.1) and includes landing gear capable of withstanding 1-meter drops onto concrete without damage (FR5.2). The center of mass is positioned 2 inches below the propeller plane (FR5.3), with a reliable, resettable payload mechanism (FR5.4) specifically designed for clay-type materials (FR5.41).

Table 6: Mission Requirements-FR6

FR6	The total Mass of the drone shall be no more than 1kg	RFP
FR6.1	Mass of the payload shall be no more than 25g	FR6
FR6.2	Mass of the onboard electronics will be no more than 60% of total mass	FR6

FR6 - Mass Distribution: Total system mass is limited to 1kg (FR6), with the payload not exceeding 25g (FR6.1). The onboard electronics comprise no more than 60% of total mass (FR6.2), requiring careful component selection and integration to maintain performance within these constraints. This aligns with the 500 dollar budget constraint while ensuring all functional requirements are met.

Table 7: Mission Requirements-FR7

FR7	The UAV shall have all the necessary sensors to accomplish the mission as well as power for said systems	RFP
FR7.1	The drone's battery shall be able to handle 5 of the longest flights our search pattern will yield	FR7

FR7 - System Integration: The UAV incorporates all necessary sensors and power systems for autonomous operation (FR7), with battery capacity sufficient for 5 complete missions using the maximum search pattern duration (FR7.1). This integration ensures reliable operation throughout the entire fire detection and suppression mission while maintaining the capability to relay critical sensor data back to the control station.

3.2. Mission Architecture

Our team's mission architecture is detailed in Figure 1 below.

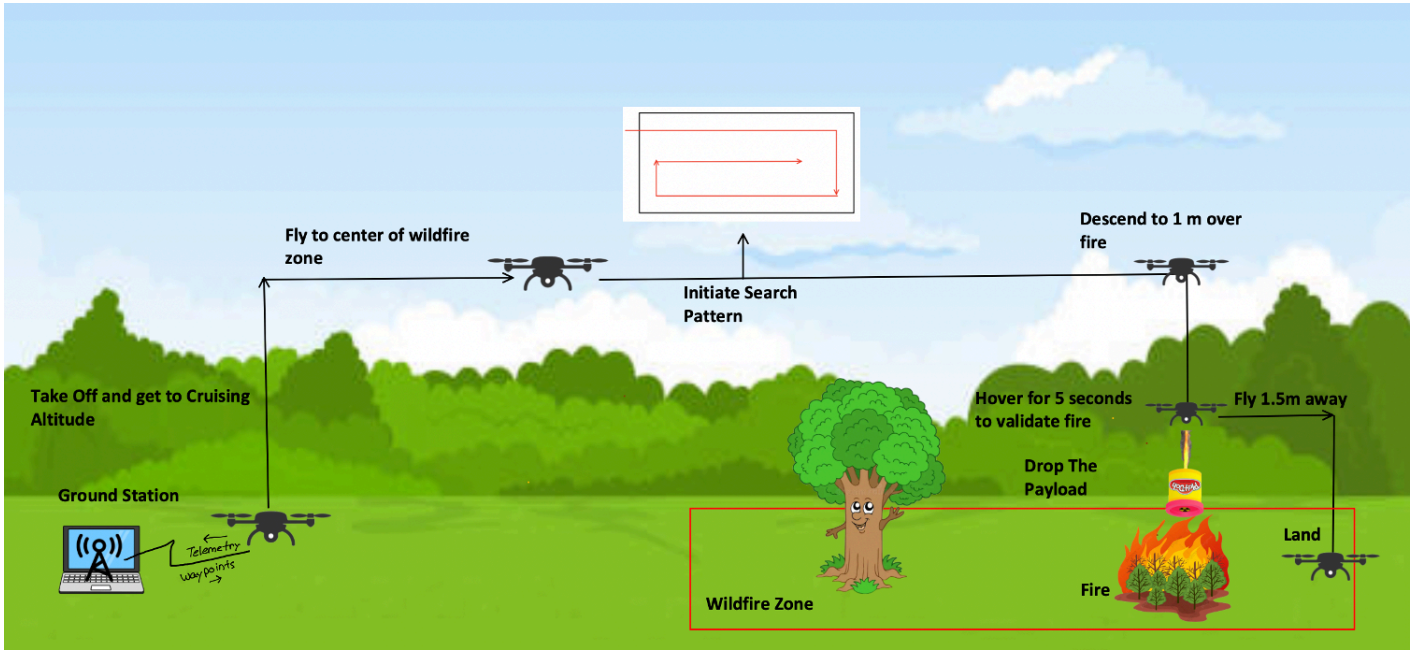

Figure 1: Mission Architecture

It is to be noted that our search pattern is designed to search over previously searched land for safety reasons. This does not impact the performance at all since the drone's heat sensor at a cruise height of 10 ft can accurately survey the entire wildfire zone with the 2 outside segments, the extra pass down the middle is completely redundant and meant to be a safety net incase the sensor is acting up and didn't locate the fire with the first loop.

3.3. Flight Profile

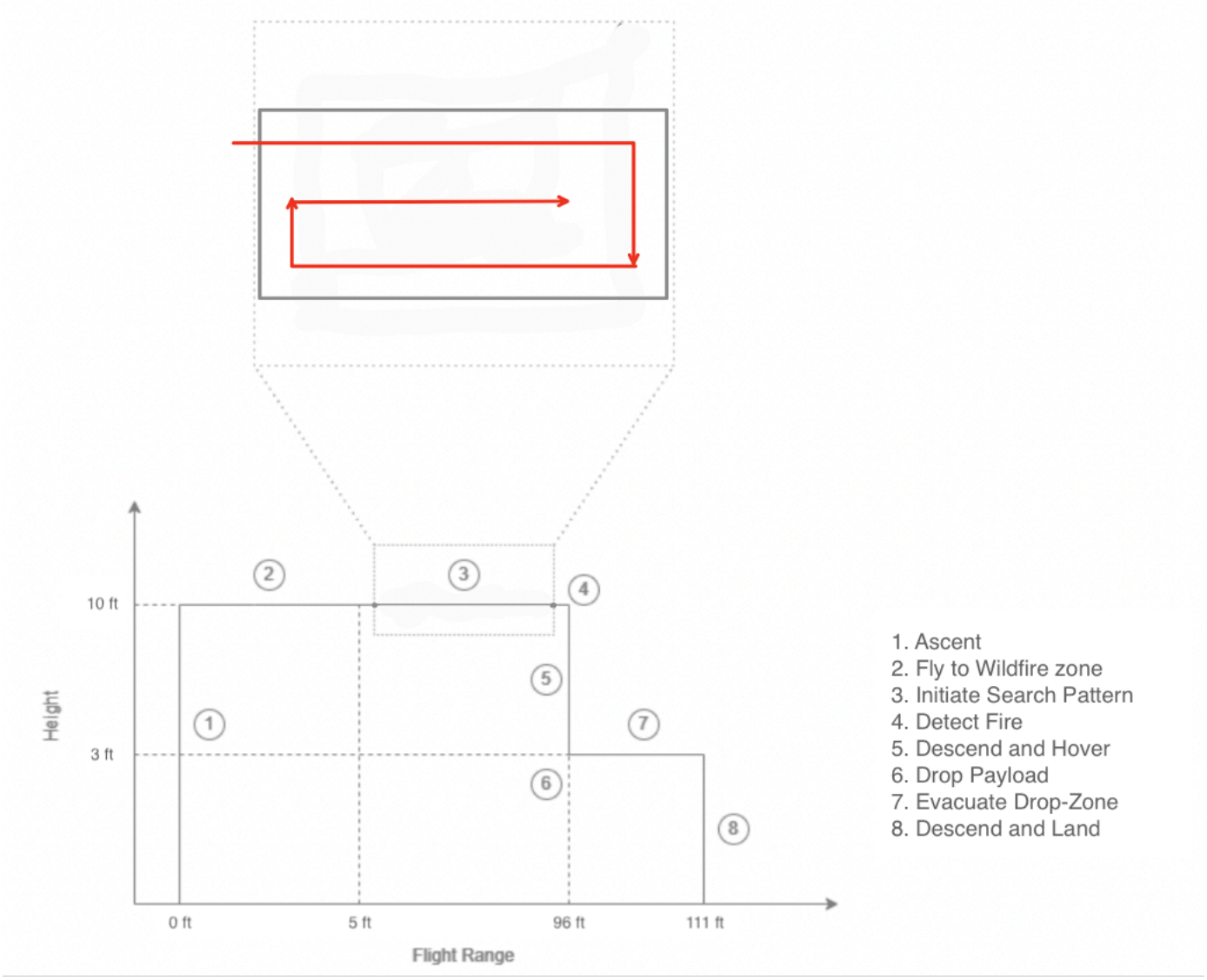

Figure 2: Flight Profile

In the above figure we have outlined our flight profile for our mission. We have the mission broken up into the segments of Ascent, Flying to Wildfire Zone, Initiate Search Pattern, Detect Fire, Descend and Hover, Drop Payload, Evacuate Drop-Zone, Descend and Land.

3.4. Communications Diagram

Understanding how data will flow through the drone is an essential part of making an autonomous vehicle. The communications diagram for the drone design can be seen in Figure 3.

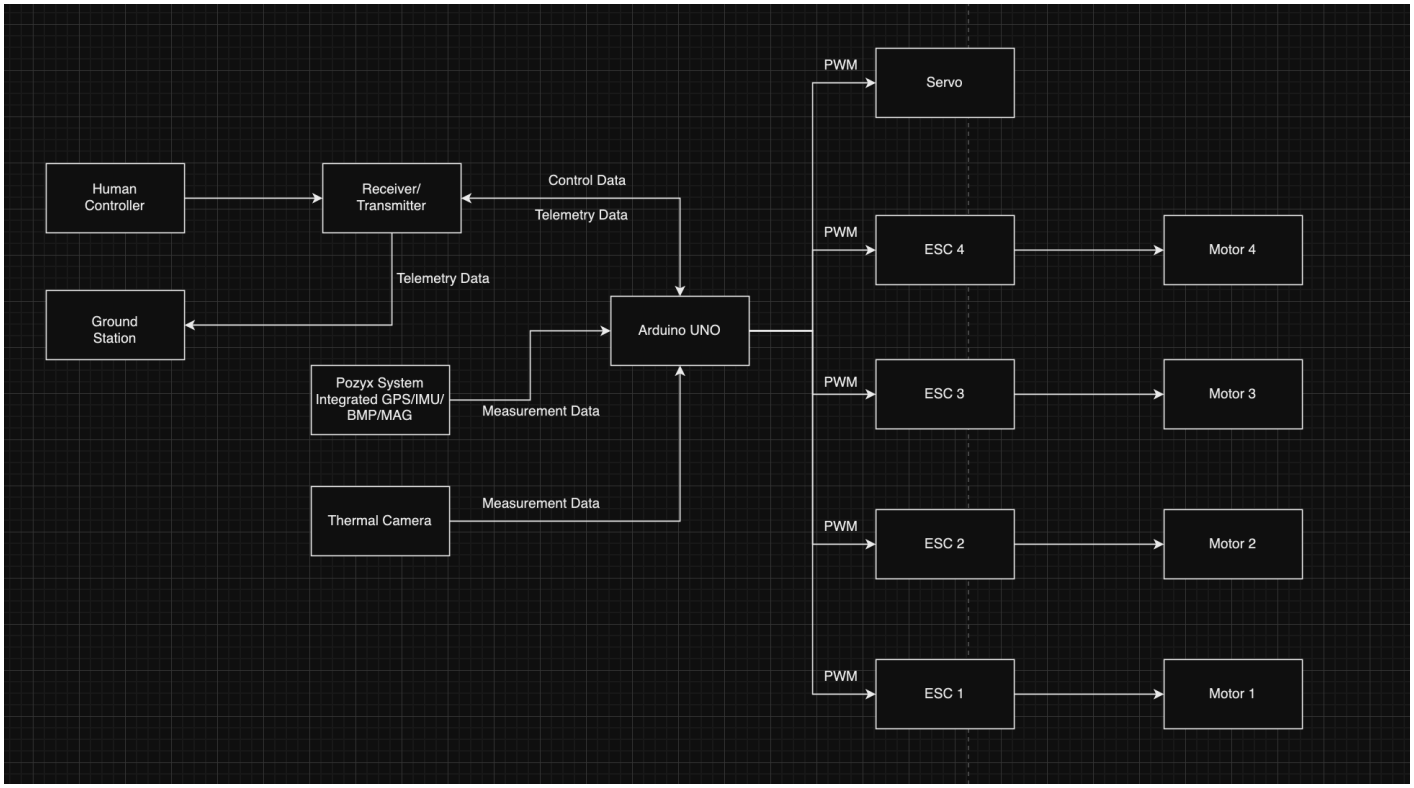

Figure 3: Communications Diagram

The communications architecture for the drone is centered around an Arduino UNO that orchestrates all data flow and control signals. At the heart of the sensing capabilities is the Pozyx module, which provides an indoor positioning system, barometer, magnetometer, and IMU.

While the drone operates autonomously, a ground controller communicating through an RC receiver provides the ability for a drone operator to take control of the drone at any point. The receiver will then transmit the controller data to the Arduino, where the controller inputs will be analyzed. The Pozyx system will continuously send the sensor data to the Arduino so the Arduino can process the data. The Arduino will constantly be sending data about the various states of the drone to the transmitter and then to the ground station.

The Arduino will then take the control signals and process which motors need to spin. It will then figure out what each motor needs to spin at and send a PWM signal to the ESC connected to the motor. The ESC will then convert the PWM signal into a 3-phase AC signal that will cause the motor to spin.

3.5. Power Diagram

The power and wiring diagram serves as a critical blueprint for the drone's electrical architecture, ensuring the functionality of all onboard systems as shown in Figure 4.

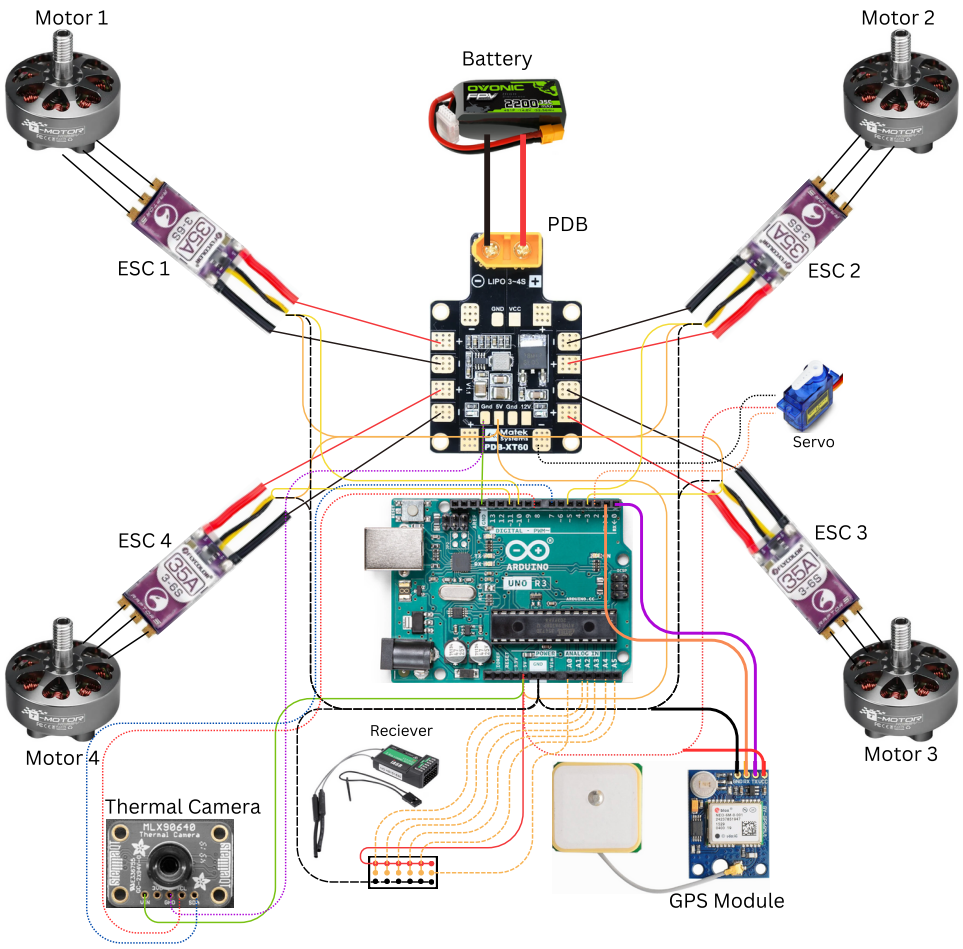

Figure 4: Power Diagram

This diagram outlines the connections between the power supply, flight computer, electronic speed controllers (ESCs), motors, sensors, and communication modules. It also accounts for the distribution of power to essential components while maintaining system efficiency and reliability. By clearly defining power paths and signal flows, the diagram ensures that the drone operates within its specified power constraints and adheres to the competition's safety standards. This section provides a breakdown of the connections and their roles in enabling the drone's autonomous flight capabilities.

The system is powered by a 4S LiPo battery, which connects to a power distribution board (PDB). The PDB is responsible for delivering power to the flight controller, electronic speed controllers (ESCs), servos, receiver, and other peripheral devices while ensuring voltage and current are managed appropriately. The flight computer acts as the central hub, managing the drone's navigation, stabilization, and sensor inputs. It connects directly to the PDB for power and communicates with the ESCs via pulse-width modulation (PWM) signals. Each ESC receives power from the PDB and regulates the current sent to its respective motor, enabling precise control of motor speed and thrust.

The servo motor, used for specific mechanical functions such as payload deployment, connects to one of the PWM output ports on the flight computer. This connection enables the flight computer to send position control signals to the servo, ensuring accurate operation during mission-critical tasks like releasing the fire suppression payload.

The receiver, responsible for communication between the drone and the operator's control station, is also powered via the flight computer. It connects through designated input channels, relaying control signals to the flight computer when necessary. Additionally, the receiver facilitates telemetry data exchange, ensuring real-time updates on drone status and sensor information.

Additional components, such as sensors (e.g., fire detection, altitude measurement) and the telemetry module, are powered through the flight computer or auxiliary power outputs on the PDB. These connections enable data acquisition and real-time communication with the ground station, allowing for effective monitoring and control.

3.6. Mass Budget

The mass budget outlines the distribution of weight across the drone's components to ensure the total mass remains within the design constraint of 1kg (1000g). Careful consideration of each subsystem was essential to achieve optimal performance, balancing structural integrity, payload capacity, and flight efficiency. The table below summarizes the mass allocation across the structure, electronics, and payload categories, culminating in a total mass of 664.8g.

3.6.1. Structural Components

The drone's structure includes the frame and propellers, which together contribute 201g. The frame, being the heaviest single component at 185g, provides durability and supports all other subsystems, while the propellers, at 4g each, account for a total of 16g.

3.6.2. Electronics

The electronics section includes all essential components required for flight control, power distribution, communication, and sensor integration, ensuring the drone operates efficiently and reliably. The total weight for the electronics amounts to 433.8g, which accounts for 43.4% of the overall mass budget.

Arduino R4 WiFi: The flight computer weighs 37g and acts as the central controller, managing sensor inputs and motor outputs. Battery: At 220g, the battery is the heaviest electrical component and provides the power required to sustain all drone systems. Motors: Four motors, each weighing 16.8g, total 67.2g, and generate the necessary thrust for flight. ESCs: The Electronic Speed Controllers (ESCs), weighing 6.5g each, collectively total 26g and regulate motor performance. FlySky Receiver: The 15g receiver enables communication with the ground controller for drone operation. Wires: 30g of wiring connects the electronics, ensuring proper power and signal transmission. Servo: A lightweight servo motor at 10g is used for payload release mechanisms. PDB: The Power Distribution Board (PDB), weighing 25g, distributes power from the battery to all essential components. Thermal Camera: At just 3.6g, the thermal camera integrates with the drone's system for fire detection capabilities.

3.6.3. Payload

The payload system consists of the fire suppression material, represented by 25g of Play-Doh, and a lightweight release mechanism weighing 5g. Combined, these components total 30g.

Section	Component	Mass (g)	No#	Component Total	Total	% of Max
Structure	Props	4	4	16	201	20.1%
	Frame	185	1	185		
Electronics	Arduino R4 Wifi	37	1	37	433.8	43.4%
	Thermal Camera	3.6	1	3.6		
	Flysky Reciever	15	1	15		
	Wires	30	1	30		
	Servo	10	1	10		
	Battery	220	1	220		
	Motors	16.8	4	67.2		
	ESC	6.5	4	26		
	PDB	25	1	25		
Payload	Play-Doh	25	1	25	30	3.0%
Total					**664.8**	**66.5%**

Figure 5: Mass Budget

The final mass budget, totaling 664.8g, leaves a margin of over 300g within the 1kg constraint. This provides flexibility for design iterations, additional components, or reinforcements, ensuring the drone remains lightweight and efficient while meeting all mission requirements.

3.7. Power Budget

The power budget provides a detailed breakdown of the drone's energy consumption during each mission phase, ensuring the system remains within its power limits. The drone operates using a 4S LiPo battery with a voltage of 14.8V, and the calculations assume a total system mass of 670g.

To determine the power requirements, the thrust-to-weight (T/W) ratio was used for each mission phase to calculate the corresponding thrust, current draw, and power consumption. Motor test data was utilized of 2680KV, with current values interpolated to reflect the system's expected performance. The relationship Power (W) = Battery Voltage (V) × Current (A) was applied to estimate power consumption across all flight stages.

Mission Phase	T/W Ratio	Thrust (g)	Current (A)	Power (W)	Battery (V)
Take-off	1.5	251.25	2.8	41.44	14.8
Climb	1.5	251.25	2.8	41.44	14.8
Cruise	1	167.5	2.3	34.04	14.8
Descend	0.5	83.75	1.8	26.64	14.8
Hover	1	167.5	2.3	34.04	14.8
Cruise	1	167.5	2.3	34.04	14.8
Land	0.5	83.75	1.8	26.64	14.8

Figure 6: Power Budget for each Motor

Prop (inch)	Voltages (V)	Throttle (%)	Load Currency (A)	Pull(g)	Power(W)	Efficiency(g/W)	Temperature(in full throttle load 3's)
HQ 3*4*3	14.8	50%	1.8	144	26.6	5.405	58℃
		100%	7.7	411	114.0	3.607	
	22.2	50%	3.2	291	71.0	4.096	75℃
		100%	13.6	767	301.9	2.540	
GF D75	14.8	50%	2	174	29.6	5.878	57℃
		100%	7.4	431	109.5	3.935	
	22.2	50%	3.2	311	71.0	4.378	70℃
		100%	12.3	755	273.1	2.765	
HQ 4*3*3	14.8	50%	2.6	254	38.5	6.601	67℃
		100%	11.7	675	173.2	3.898	
	22.2	50%	4.4	458	97.7	4.689	85℃（Hot）
		100%	18.9	1101	419.6	2.624	
GF5125*3	14.8	50%	2.4	227	35.5	6.391	63℃
		100%	9.9	603	146.5	4.115	
	22.2	50%	3.8	409	84.4	4.848	80℃
		100%	16.9	1035	375.2	2.759	

Figure 7: 2650KV Motor Test Report

Key mission phases include:

Take-off and Climb: Operating at a T/W ratio of 1.5, the drone generates 251.25g of thrust, drawing 2.8A of current and consuming 41.44W of power. Cruise and Hover: With a T/W ratio of 1, thrust decreases to 167.5g, reducing the current to 2.3A and power consumption to 34.04W. Descent and Land: During these phases, the T/W ratio drops to 0.5, requiring only 83.75g of thrust, with a current draw of 1.8A and power consumption of 26.64W.

The table above (Figure 6) summarizes the power requirements for each mission phase. By analyzing these values, the team ensures that the drone's energy consumption aligns with the mission's

operational demands, allowing for efficient battery management and system performance throughout the flight.

The power consumption analysis ensures the drone's electronic components operate within the constraints of the selected power source, maximizing energy efficiency across all mission phases. Each component, including the ESCs, Arduino flight controller, thermal camera, Power Distribution Board (PDB), and servo, is assessed for current draw and power consumption under various operational scenarios. The low power requirements of these components contribute to minimizing energy losses while ensuring consistent performance.

Component	Mission Phase	T/W Ratio	Current (A)	Power (W)	Battery (V)
ESC	Take-off	1.5	<1 mA	<1 mW	14.8
	Climb	1.5	<1 mA	<1 mW	14.8
	Cruise	1	<1 mA	<1 mW	14.8
	Descend	0.5	<1 mA	<1 mW	14.8
	Hover	1	<1 mA	<1 mW	14.8
	Cruise	1	<1 mA	<1 mW	14.8
	Land	0.5	<1 mA	<1 mW	14.8
Arduino	Take-off	1.5	0.05	0.25	5
	Climb	1.5	0.05	0.25	5
	Cruise	1	0.05	0.25	5
	Descend	0.5	0.05	0.25	5
	Hover	1	0.05	0.25	5
	Land	0.5	0.05	0.25	5
Thermal Camera	Take-off	1.5	0.02	0.1	5
	Climb	1.5	0.02	0.1	5
	Cruise	1	0.02	0.1	5
	Descend	0.5	0.02	0.1	5
	Hover	1	0.02	0.1	5
	Cruise	1	0.02	0.1	5
	Land	0.5	0.02	0.1	5
PDB	Take-off	1.5	<1 mA	<1 mW	14.8
	Climb	1.5	<1 mA	<1 mW	14.8
	Cruise	1	<1 mA	<1 mW	14.8
	Descend	0.5	<1 mA	<1 mW	14.8
	Hover	1	<1 mA	<1 mW	14.8
	Cruise	1	<1 mA	<1 mW	14.8
	Land	0.5	<1 mA	<1 mW	14.8
Servo	Take-off	1.5	<1 mA	<1 mW	5
	Climb	1.5	<1 mA	<1 mW	5
	Cruise	1	<1 mA	<1 mW	5
	Descend	0.5	<1 mA	<1 mW	5
	Hover	1	<1 mA	<1 mW	5
	Cruise	1	<1 mA	<1 mW	5
	Land	0.5	<1 mA	<1 mW	5

Figure 8: Power for each Components

Power Consumption Breakdown Electronic Speed Controllers (ESCs): The ESCs regulate the power delivered to the motors and are essential for controlling motor speed. Their power draw is minimal,

with a current consumption of < 1 mA and power usage of < 1 mW across all mission phases. The ESCs are powered by the 14.8V battery, aligning with the primary power system.

Arduino R4 WiFi: Acting as the drone's flight controller, the Arduino R4 WiFi maintains steady current consumption of 0.05A and power consumption of 0.25W across all phases. The component is powered by a 5V source, which ensures sufficient energy for processing tasks, sensor inputs, and system outputs.

Thermal Camera: The thermal camera operates with a low current draw of 0.02A and consumes 0.1W of power during all mission phases. Like the Arduino, the camera runs on a 5V source, enabling lightweight fire detection capabilities while maintaining energy efficiency.

Power Distribution Board (PDB): The PDB distributes power from the battery to all connected components. Its power draw is negligible, with a current consumption of < 1 mA and power usage of < 1mW across all phases. Operating on a 14.8V source, the PDB plays a critical role in providing consistent power distribution.

Servo Motor: The servo motor used for the payload release mechanism has minimal power requirements, drawing < 1 mA current and consuming < 1 mW of power. Like the Arduino and thermal camera, the servo operates on a 5V source.

The power budget analysis highlights the low power consumption of all electronic components, ensuring efficient use of the 14.8V battery for primary systems and a 5V supply for auxiliary electronics like the Arduino, thermal camera, and servo. The ESCs, PDB, and servo motors require minimal power, contributing negligible losses to the overall system.

3.8. Mission Strategy

Our team's strategy to win the competition is grounded in three key pillars: modularity, iterative design, and optimization. We will implement a modular design for our drone to enhance reliability and maintainability. This approach allows for quick replacement of broken or malfunctioning parts, minimizing downtime, and ensuring consistent performance. Modularity also supports streamlined integration of sub systems, enabling us to adapt the design as needed without overhauling the entire vehicle. We will utilize 3D printing and inexpensive material such as PLA for rapid prototyping and testing. This strategy enables us to refine our design through multiple iterations efficiently, balancing structural integrity with weight reduction. Using 3D printing also allows us to produce custom-fit components that optimize space and functionality while keeping costs within budget constraints. To maximize performance, we will prioritize minimizing weight and reducing power consumption. These efforts include selecting lightweight components, designing aerodynamically efficient structures, and optimizing the powertrain for energy efficiency. By improving range and conserving energy, our drone will be better equipped to complete its mission successfully while staying within the 1kg weight limit.

This strategy directly influences our Concept of Operations (CONOPS), requirements, and vehicle design. The modularity ensures that repairs or modifications can be performed quickly, reducing mission downtime. Iterative prototyping with 3D printing aligns with our need for cost-effective and timely development while allowing us to meet performance requirements through incremental improvements. Weight and power optimization affect several aspects of our design. A lighter vehicle with reduced power consumption enables extended flight time and operational range, ensuring the drone can navigate to target locations and return safely. This is also one of the key aspects considered for the scoring of the competition, Keeping the weight and cost as low as possible will allow us to outperform the other groups. Additionally, these considerations shape the selection of components such as motors, propellers, and the battery to achieve an optimal balance of performance and efficiency.

4. Design Overview

4.1. CAD

Designing the full drone in CAD early and accurately was a very important piece the team wanted to focus on. Without being able to visually see the drone, it would be impossible to make the right decisions. This is why the team spent so much time creating our drone in CAD, all the way down to the last detail. Having the entire drone in cad also allowed the team to weight each component and input it into each part. This allowed solidworks to give the team a very accurate estimate of the weight, and moments of interia of the drone seen in Figure 9.

```
Mass properties of Full_Drone_J
    Configuration: Default
    Coordinate system: -- default --

Mass = 672.52 grams

Volume = 368581.01 cubic millimeters

Surface area = 286470.55  square millimeters

Center of mass: ( millimeters )
     X = 0.01
     Y = 26.39
     Z = 0.72

Principal axes of inertia and principal moments of inertia: ( grams *  square millimeters )
Taken at the center of mass.
      Ix = ( 0.97, -0.01,  0.23)        Px = 2340453.33
      Iy = ( 0.23,  0.02, -0.97)        Py = 2492764.73
      Iz = ( 0.00,  1.00,  0.02)        Pz = 4244903.28

Moments of inertia: ( grams *  square millimeters )
Taken at the center of mass and aligned with the output coordinate system. (Using positive tensor notation.)
     Lxx = 2348561.10          Lxy = -3362.97          Lxz = 34134.30
     Lyx = -3362.97            Lyy = 4244424.65        Lyz = -28896.08
     Lzx = 34134.30            Lzy = -28896.08         Lzz = 2485135.59

Moments of inertia: ( grams *  square millimeters )
Taken at the output coordinate system. (Using positive tensor notation.)
     Ixx = 2817161.64          Ixy = -3244.72          Ixz = 34137.52
     Iyx = -3244.72            Iyy = 4244772.91        Iyz = -16126.77
     Izx = 34137.52            Izy = -16126.77         Izz = 2953387.94
```

Figure 9: Mass properties of the designed drone created by solidworks.

This proved to me a huge advantage when trying to simulate the drone on PACE, as having the wrong assumptions for moments of inertia means your controller gains will be very wrong. A lesson we learned the hard way, as seen in Figure 10.

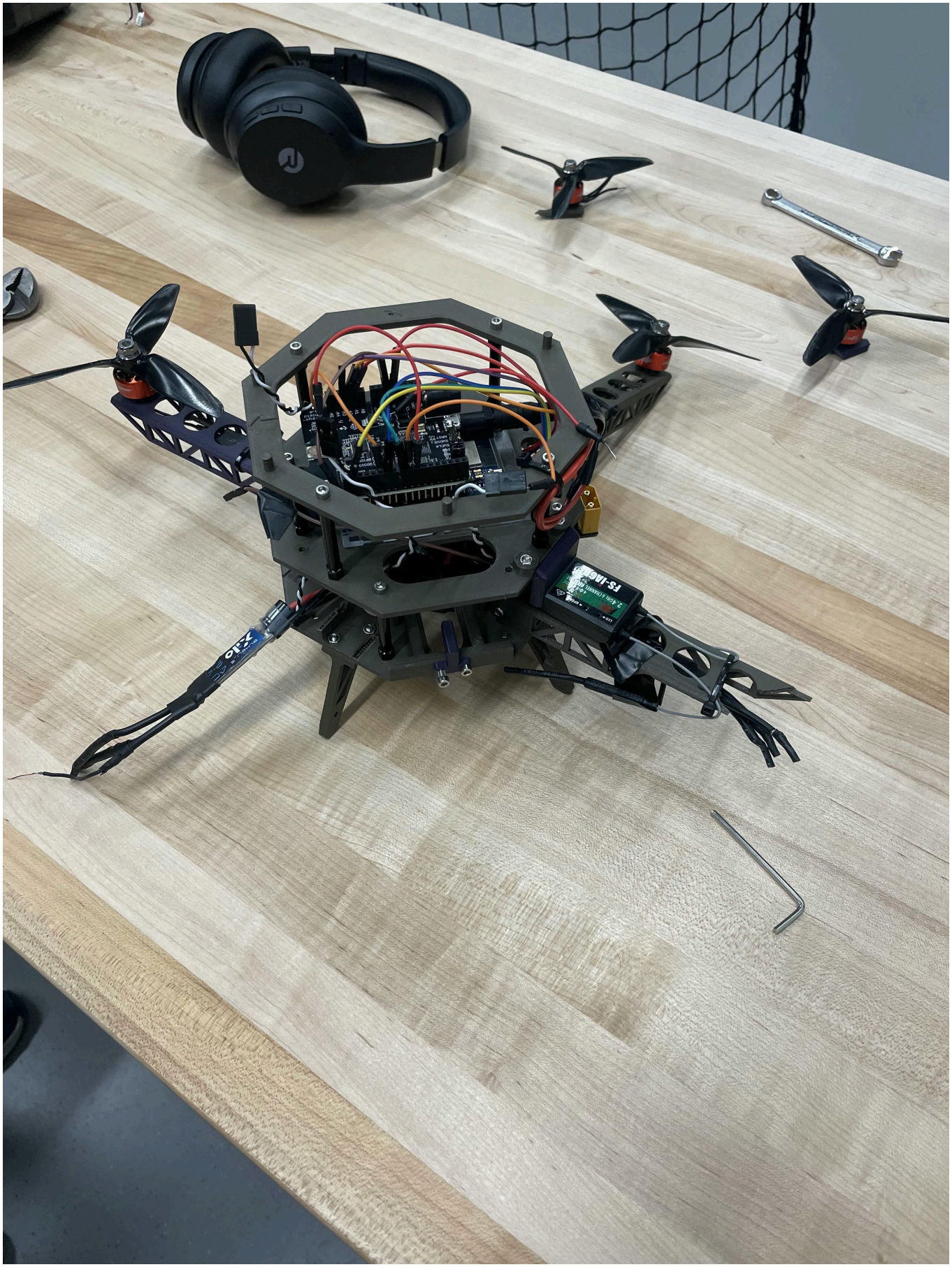

Figure 10: Devastation caused by an improperly tuned controller using incorrect moment of inertia values in PACE.

4.1.1. Frame

When determining how the team wanted to approach the design, the team looked at the previous year's drone designs. The main issue that was found and needed to be avoided at all costs was having issues integrating the electrical components physically on a frame and having messy wiring. The second most important issue was having the ability to rapidly fix our drone in the event of a crash. The obvious choice, when these issues are of the greatest importance, is to design and print a unique drone frame.

There are many 3D printers on campus that allow you to print for free with a variety of materials. So 3D printing the frame is not only a cheap option, but will allow for unbeatable customization and will allow us to place our components wherever and however we want.

The first design choice that was made was the type of frame configuration that was best for the drone. The different frame types considered can be seen in Figure 11.

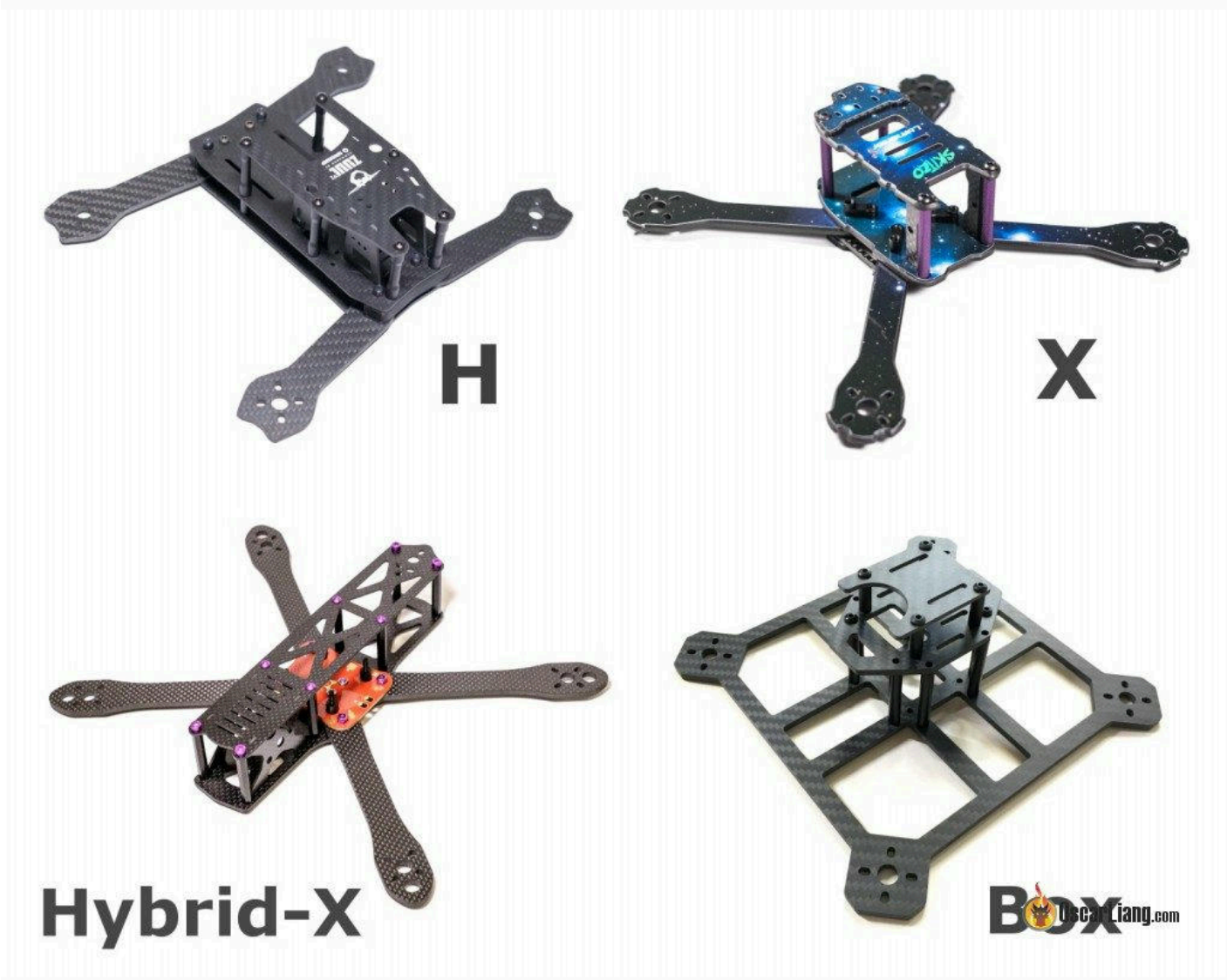

Figure 11: [1] Frame types.

Of the different frame designs, the X frame was the best due to its simplicity. Having only four arms and a small inner body would make it taller than the other options, but it's thought that it will be better as it allows for the adjustment of the center of gravity.

Next, the arms of the drone had to be designed. They are a very important part, as they will be experiencing the majority of the force. The team set a limit of no more than 1mm of deflection under a full-throttle load. Since the team wanted such a low deflection, a simple rectangle couldn't be used. The team also wanted a way to easily mount and in case the wires that run to the motor. The solution that was decided on was a C beam-shaped arm, with a lot of trusses and holes cut out of the beam to reduce weight. Another requirement we had for the arms was that they needed to be wide enough to house the ESC. The proposed design that meets all of these criteria of the arm can be seen in Figure 12. The deflection of the arm under max load can be seen in Figure 13.

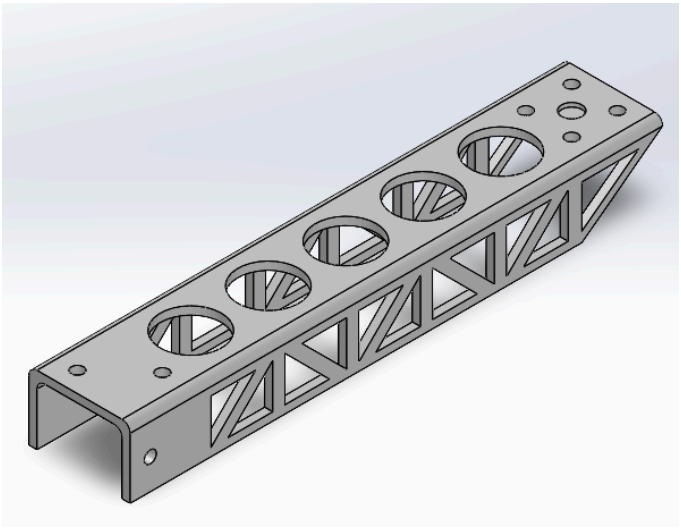

Figure 12: CAD of Drone arm.

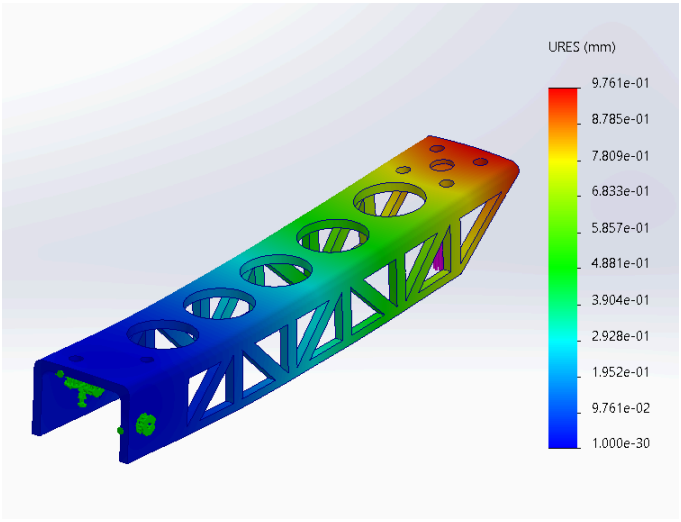

Figure 13: FEA of the arm under max load.

The C beam geometry will cause very low deflection, and as seen in Figure 13, the deflection under a maximum load of 4.905 N only deflects by 0.9761 mm, which is within the specified deflection tolerance. This design also has a natural channel where the wires can be passed to the motor and ESC cleanly. The holes on the top of the air will allow for clean airflow from the propellers downward in order to make the efficiency of the propeller as good as possible. The arm will be connected to the base plate of the drone by four M3 screws, two on the top and two on the sides. The arm will slide over the connection point on the base plate as seen in Figure 14, and it can be seen as fully connected in Figure 15.

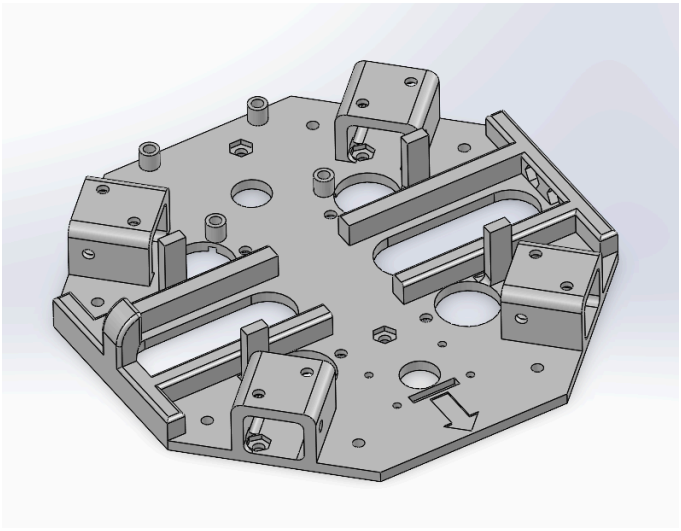

Figure 14: CAD of baseplate.

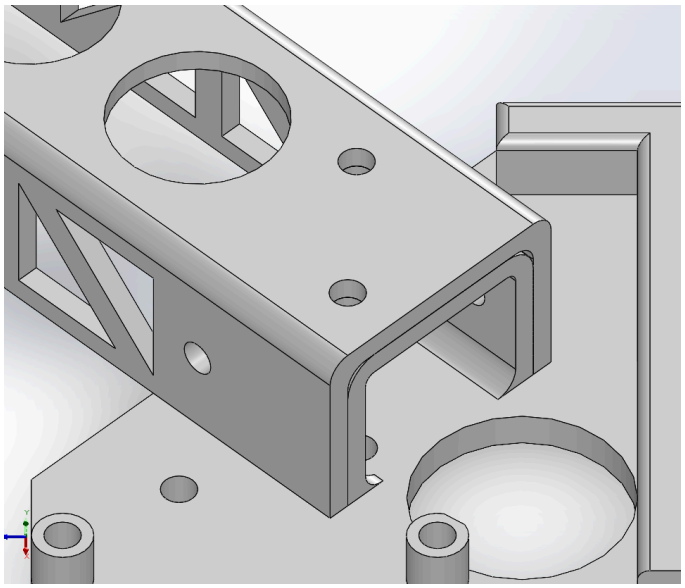

Figure 15: Baseplate-Arm Joint

These types of connections were chosen because they allow the arm to be connected on both the sides and the top, preventing a majority of bending and allowing the wires to pass underneath.

Another feature of the Base plate that is essential for an efficient design is the raised battery holding area in the middle of the baseplate. This was done because the team wanted the battery on the lowest level of the drone, but the battery needed to be raised slightly so wires from the different components could run smoothly through the baseplate. Also, The four cylinders seen at the top of Figure 14 are where the Power distribution board will be mounted. There are raised off of the baseplate because the screw holes for the payload mechanism are right where the power distribution board would be, and it needs to be raised in order to allow room for the screw. Another feature is the hole cut out for the thermal camera to look down, which can be seen at the top of the base plate. In order to keep the battery aligned and stationary in flight, the four rectangular posts in the middle of the baseplate were used. Additionally, two battery brackets which will be shown later were screwed into the base plate in order to prevent the upward movement of the battery. Lastly, an arrow was added to the base plate, along with all of the other layers, in order to let the assembler know the orientation of the different parts.

4.1.2. Full Assembly

The full drone assembly with everything labeled can be seen in Figure 16, Figure 17, and Figure 18.

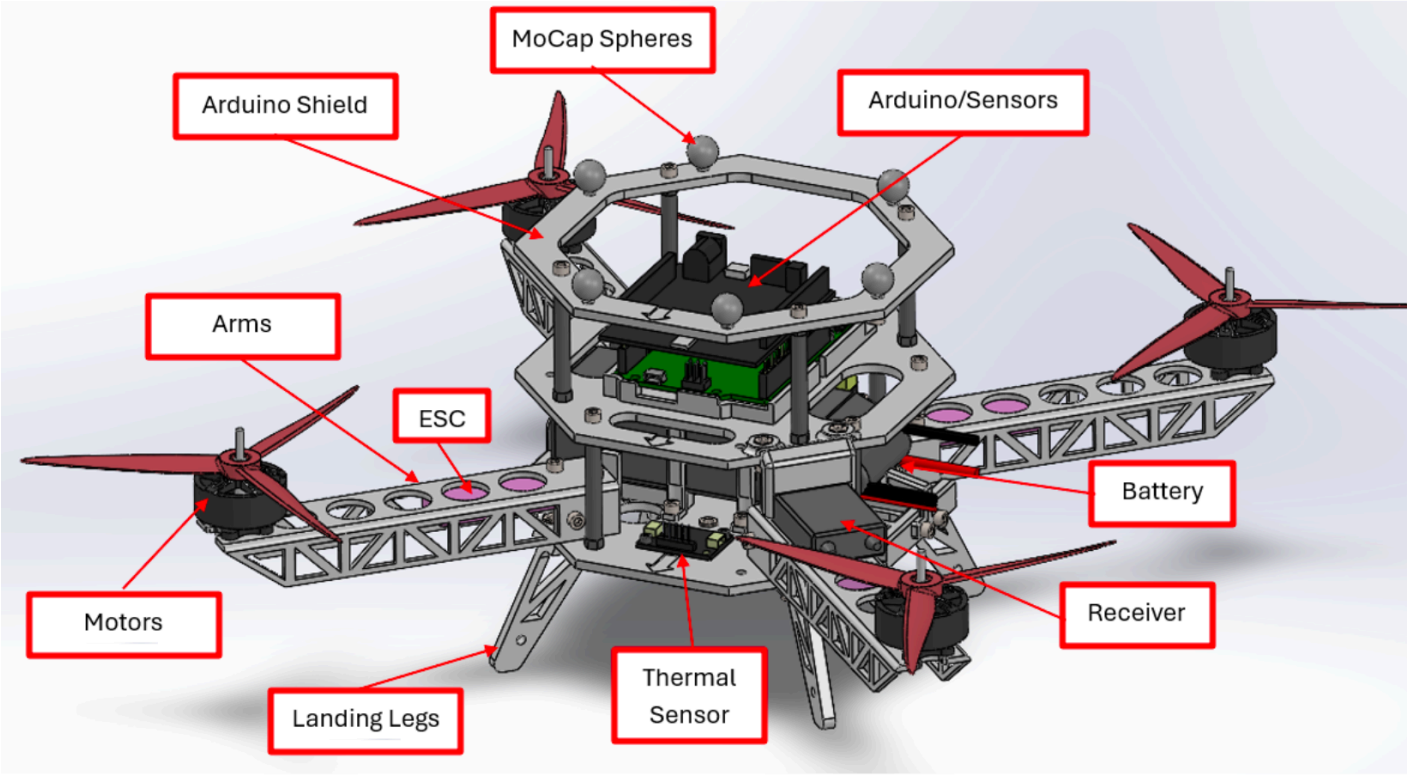

Figure 16: Labeled front view of drone

Figure 16 Shows the front face of the drone. The image shows the ESCs under each arm. Each arm also has a brushless motor with a 5-inch propeller. A thermal camera can be seen on the baseplate facing downward. There is a hole that the camera passes through that is more visible in a later view. The battery is placed in the center of the lowest layer to keep the center of gravity as low as possible. The Arduino and GPS module are placed on the top layer, with no other electronic components, in order to decrease electronic interference. Having the Arduino and the GPS module on top gives a clear view of how the GPS module works. The receiver is placed on one of the arms so its antennas can be guided through the holes in the arm and face downward so it can receive a clear signal from the transmitter. The drone also features a lot of holes cut out of its various pieces. This is to reduce weight and allow for multiple places to run wires. Lastly, from this view, the Arduino shield can be seen, which is used to prevent the Arduino from colliding with the ground if the drone flips over. On the Arduino shield are the MoCap balls. The Mocap system uses these to know the orientation and position of the drone. These balls are screwed in using little 3D-printed screws that are part of the Arduino shield.

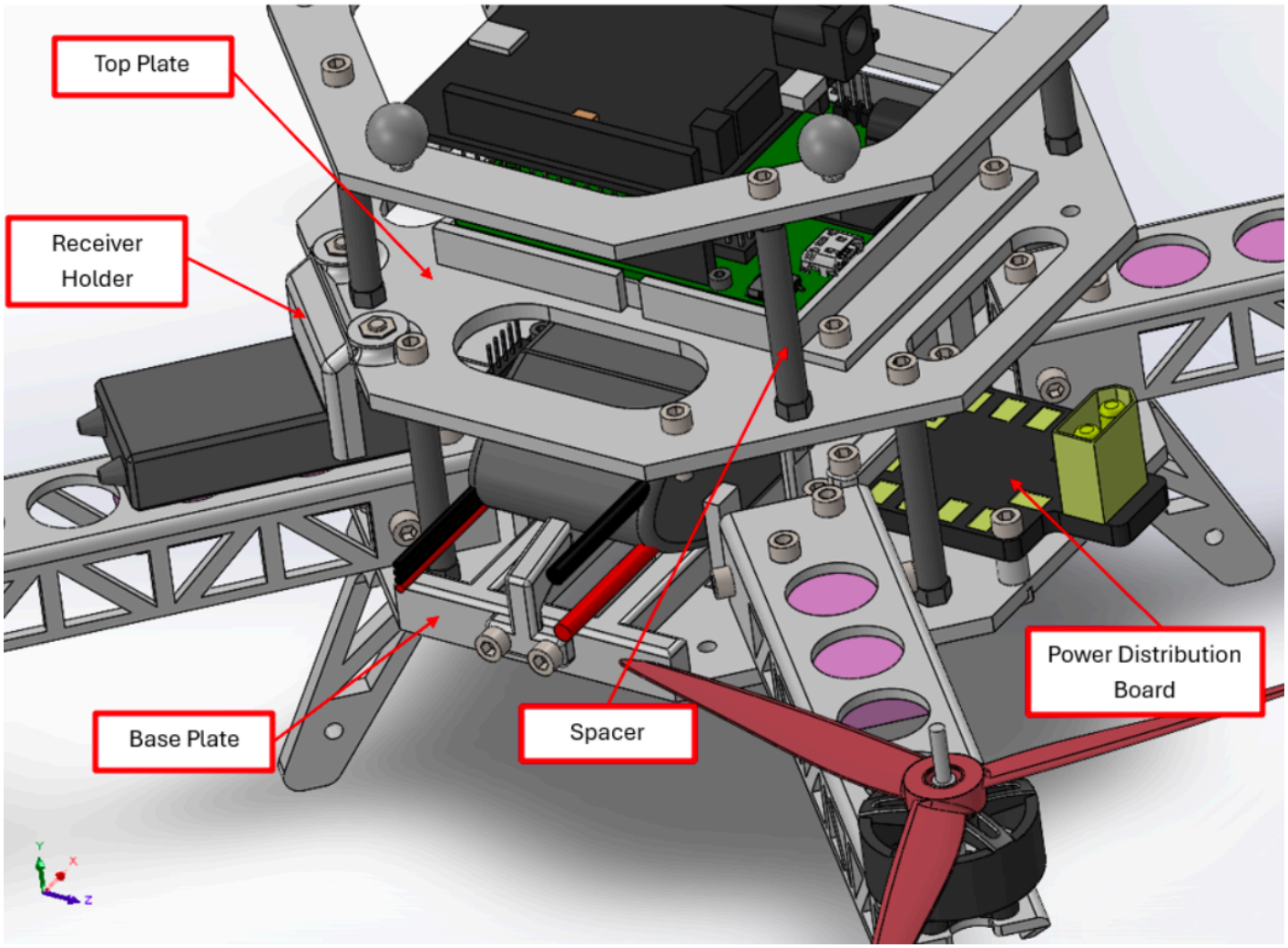

Figure 17: Labeled rear view of drone

Figure 17 shows the back side of the drone, which reveals the power distribution board, which is placed on the bottom level right next to the battery so the wire can be easily connected. When looking closely at the power distribution board, one can notice that it's raised off of the base plate a little. This is because the payload mechanism is mounted from the bottom there, and there needs to be some clearance for the screws. The receiver can be more clearly seen from this angle, and it has been placed in such a way that none of it is in the direct downwash of the propeller. This will ensure the efficiency of the propeller remains high. The receiver is also restrained by just using its own geometry, which is hard to explain without seeing it in person. The spacers are also labeled in this image and are what separates the different layers of the drone. They are purchased and made of aluminum.

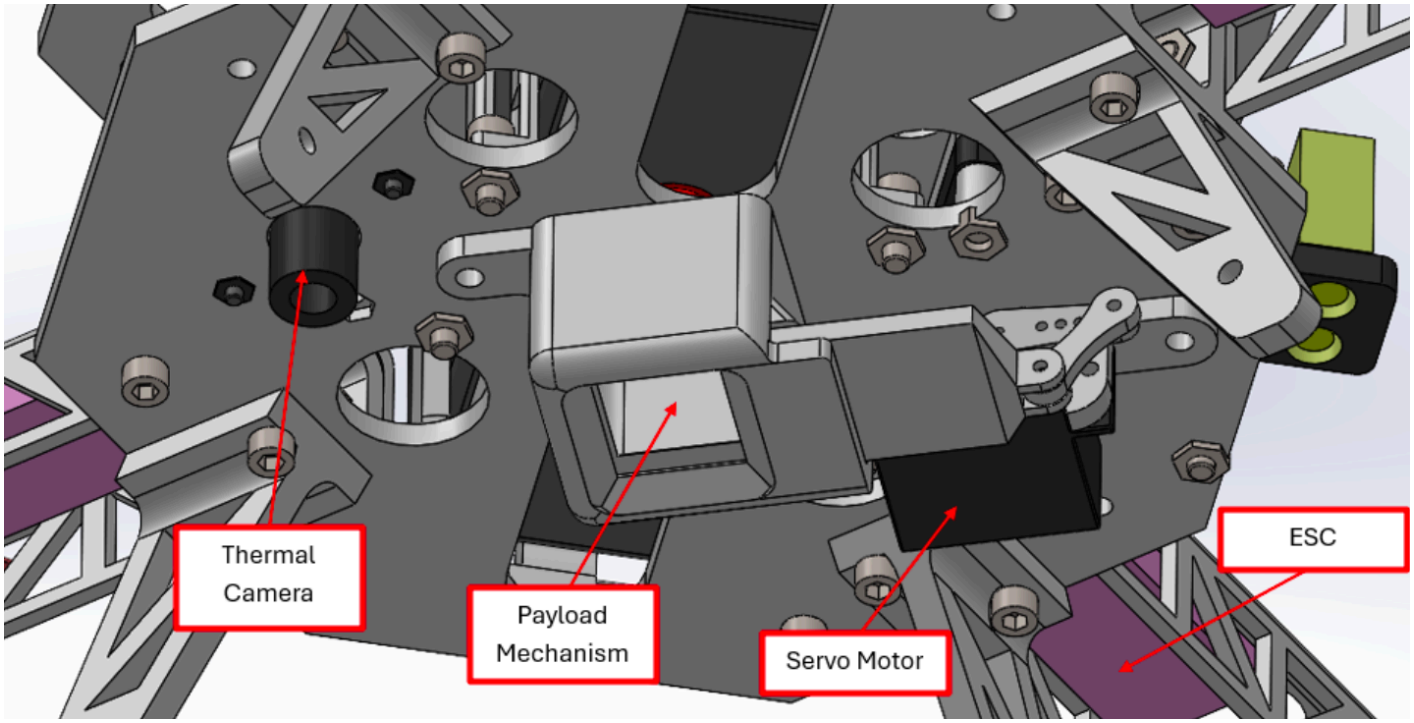

Figure 18: Labeled bottom view of drone

Figure 18 gives a view of the drone's underside, with the payload mechanism visible. The payload mechanism is a sliding door that is operated by a Servo, which is also labeled. This image also shows a better view of the ESC under the arm. As well as the camera section of the thermal camera, which is poking through the baseplate of the drone.

4.1.3. Three View Sketch

The three-view sketch of the drone showing all of the major dimensions can be seen in Figure 19.

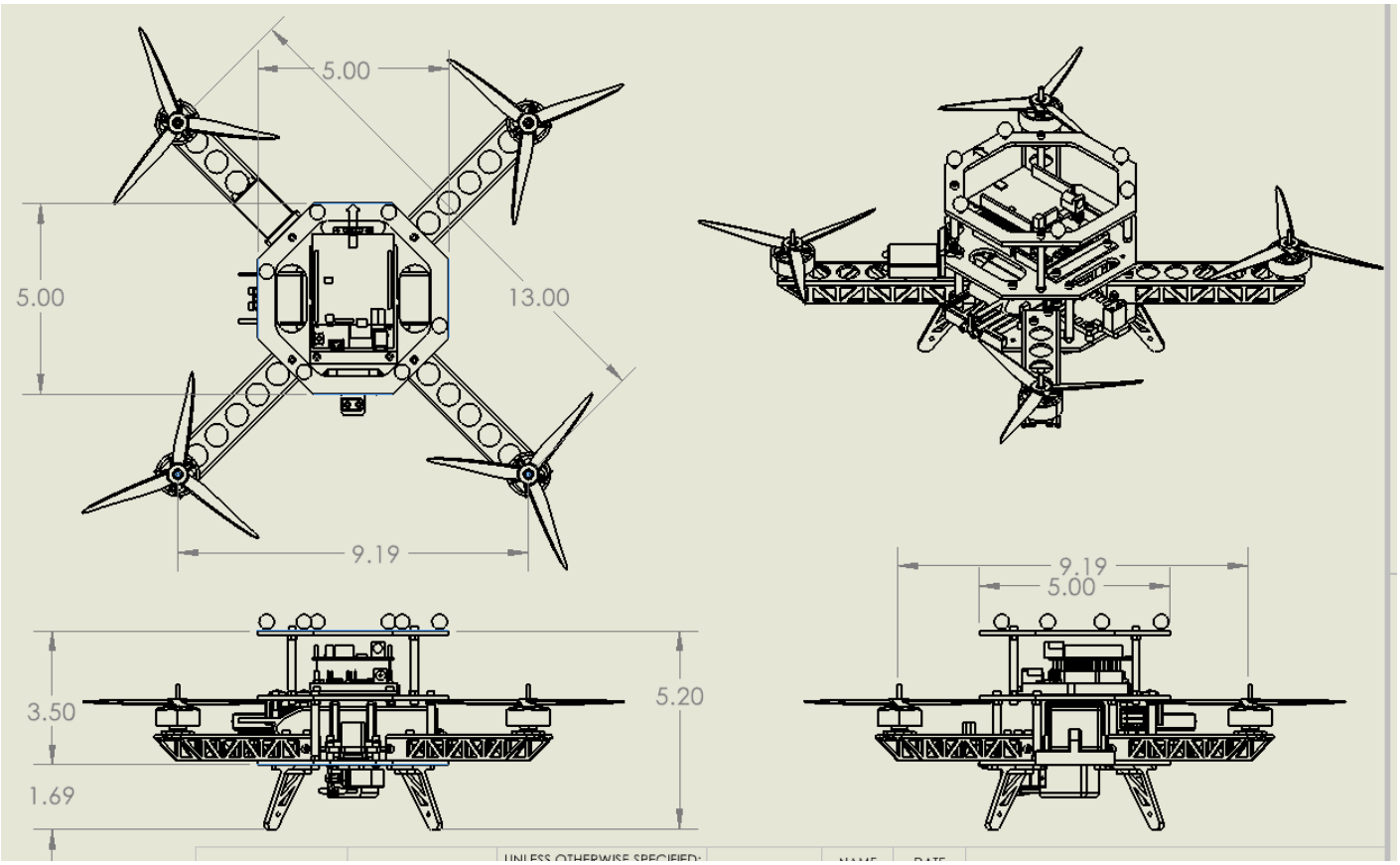

Figure 19: Three-View Draft Drawing

In Figure 19, the drone has a span from the motor to the motor of 13 inches. This is a reasonable size, and the team set a limit of 16 inches of span, so we met that requirement. The max height of the drone from the floor to the top is 5.20 inches. The landing gear gives the drone a clearance of 1.69 inches. The drone's center hub is an octagon, where the height and width are 5 inches. The distance between the layers of the central hub is 1.57 inches.

4.2. Sub System Design

Every complex system can be broken down into smaller problems and subsystems. For our drone project, the team identified eight different subsystems that needed to work together for a successful mission. These subsystems were divided into two categories: electrical components, which included all the parts that needed to be ordered, and software subsystems, which comprised the controller, Kalman filter, waypoints, data link, ground control station, thermal camera, and payload mechanism.

4.2.1. Component Selection

Selecting the right components for the drone was the most critical step in the design process. Without a properly sized battery, suitable propeller, and motor, the drone could have been, at best, overweight and over budget, or, at worst, unable to fly. To address this, significant time was dedicated to trade studies and optimization analysis using ATSV software.

4.2.1.1. Analysis of Electrical Components

To start the design process, the team needed to narrow down the list of components that were viable. This was done using the provided program, ATSV. Within the program, constraints on the diameter, and thrust were placed to filter out all of the invalid components. To filter by preference and their effectiveness in our problem, we used the disk actuator equation,

$$P_{\text{Actual}} = \frac{1}{\text{FoM}} \cdot \frac{T^{\frac{3}{2}}}{\sqrt{2\rho\pi R^2}} \tag{1}$$

where *FoM* is the factor of merit. This correlates to the efficiency of the propeller. From this equation, minimizing the power required consists of maximizing the *FoM* and the radius. Drones built for stability and slow flying tend to have lower pitch angles. These factors were applied in the ATSV search resulting in Figure 20.

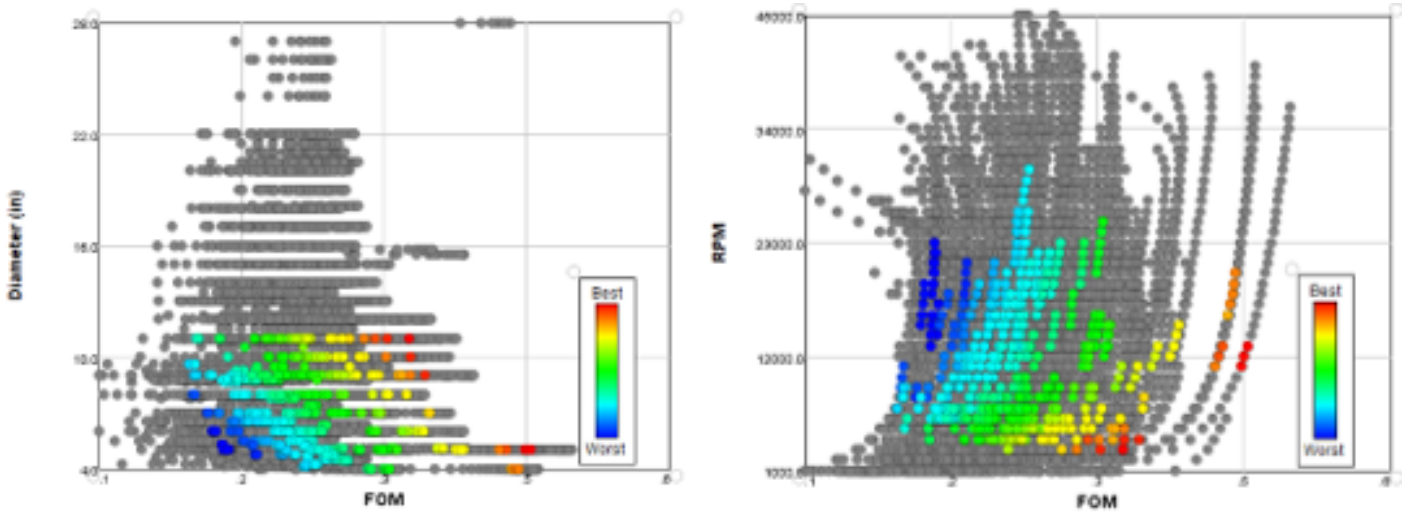

Figure 20: ATSV Propeller Search

The resulting figures showed that there were some optimal diameters to pick propellers from. These ranges would be 4-5 inches and 7-10 inches. The search also favored lower RPM propellers. Using this information the team was able to pick several propellers that could be viable.

4.2.1.2. Optimization Methodology

To achieve the goal of designing an optimal subsystem, the team built an optimization process to apply to multiple system designs. Unlike traditional optimization, the design constraints are discontinuous points an space, so the traditional search methods do not work. In the case of this project, the problem can be simplified to the selection of the propeller. The propeller dictates the energy power required at a given thrust value, which correlates to the capacity of the battery. The size of the propeller also dictates which motor KVs are viable as they are diameter-dependent. While this process can be coded to include all design aspects of the drone, due to the simple nature of the problem, the team chose to work this problem out on a spreadsheet.

From a starting prop, an ESC and motor are chosen. The ESCs at this scale have minimal variations in price and weight. The motors, however, can vary greatly in terms of weight and cost depending on the KV. Within the same KV, the variation in weight and cost is minimal. The motor selection process consisted of finding a reliable motor that satisfies the propeller requirements. This is used as the starting system for the process. The optimization process is shown in Figure 21.

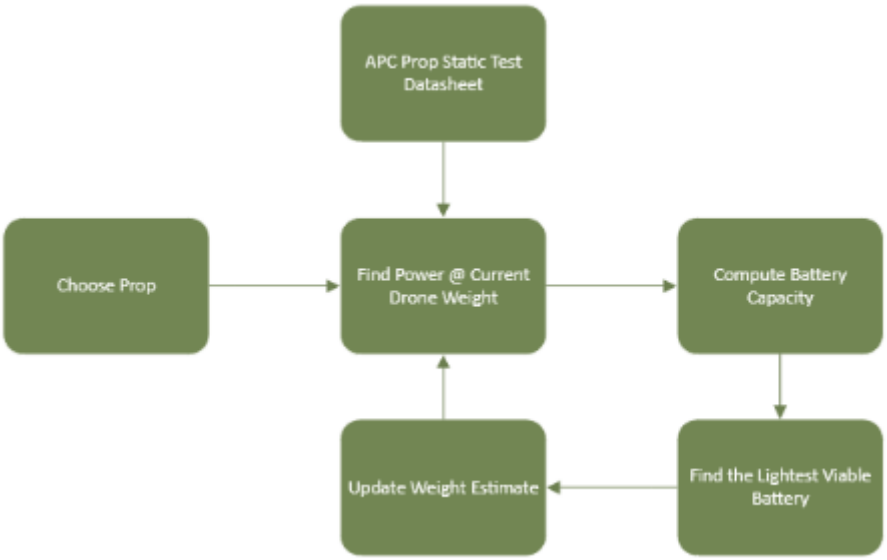

Figure 21: Optimization Flow Chart

Starting from a chosen prop and a drone mass of 1 kg, the required power to hover is found from the APC Prop Database. From the power estimate, an estimated battery capacity can be found using,

$$\text{Capacity(Ah)} = \text{FOS} \cdot \text{ToF} \cdot \frac{P_{\text{data}}}{V_{\text{Battery}} \cdot \text{DoD} \cdot \gamma} \tag{2}$$

where *FOS* is the design factor of safety, *ToF* is the estimated flight time, P_{data} is the power acquired from the database. *DoD* and γ are battery type specific values of the amount to be discharged and the

efficiency of the battery. The typical efficiency for a LiPo is noted as 0.9. The team is choosing a DoD of 0.9 for the final competition as we would never reach the number of cycles to be detrimental to the health of the battery.

After the battery capacity is determined, the lightest viable battery is found at or above the computed capacity. This weight is noted and used to update the overall weight of the drone. The process is now repeated with the new thrust requirement.

4.2.1.3. Trade Study

The selection process for which system to use consists of a quantitative decision matrix. This matrix allowed the team to narrow the subsystem design based on their performance and how it would contribute to the team's overall score within the flight competition. The equation for the score is,

$$\text{Score} = \frac{1000}{W} \cdot \frac{500}{C} \cdot (\text{Mission Completions}) \tag{3}$$

where W is the weight of the drone in grams and C is the cost of the drone in dollars. *Mission Completions* is defined as the set of successful tasks accomplished during flight. This equation was used to determine the preference function. This function was defined as,

$$\text{Preference Function} = 0.5\ \text{Mass} + 0.3\ \text{Cost} + 0.2\ \text{FOS} \tag{4}$$

Mass and cost were used as they directly correlate with the score of the team. The FOS, factor of safety, was also used as it determines how safe the drone may complete its set of tasks The weights were chosen as to the impact the variables have on flight as well as the score impact. As the score in Equation 3 weighted all of the variables equally, the team took the liberty of weighting based on how controllable they were based on design choices. Mass was the most controllable factor so it was weighted at 50%. Cost was weighted less at 30% and the FOS was left at 20%.

Each system was built off of a set of propellers as outlined in the optimization methodology. The normalized scores are shown in Table 8. The results from this table provided valuable insights into how the team should proceed forward.

Table 8: Propeller Trade Study

	5x37e3	5x3E-B4	6x4	4x4E-3
Mass	0.95	1.00	0.00	0.24
Cost	0.94	0.94	0.00	1.00
FOS	1.00	0.61	0.00	0.03
Total	2.90	2.56	0.00	1.27

While the 6 inch diameter propellers were more power efficient, the motors required to run them were significantly heavier due to requiring a higher torque and KV. As such, overall mass was greater, requiring more thrust and power to remain in flight.

The most optimal setups were both of the 5 inch propellers. One of them being a low pitch 2 bladed propeller and the other having a slightly higher pitch with 3 blades. As both require the same setup and offer similar performance as it relates to the success of the project, the team chose to use both propellers in the final design. Allowing for the ability to switch if one is underperforming.

4.2.1.4. Component List

After optimizing the largest contributors to weight, the process of selecting components came to picking the most compatible parts with the chosen propeller, battery, and drone frame.

The battery paired with these propellers was the 2200 mAh 4S battery from Ovonics. The battery weighs about 220g, which is much lower than the estimate from the team's PDR. The rated discharge metric is 50C, which translates to 110A. This value far exceeds the estimated current draw from the systems.

Based on the trade study discussed, the system chosen was based on the 2 and 3-bladed 5-inch propellers. The motor selected was the "Smoox Plus 1507" with the 2680KV option. This motor weighs 15.8g and is more power efficient than others on the market. Based on the power benchmarks supplied by the manufacturer. The motor draws 2.4A at 14.8V. This roughly correlates to 220 gram-force at 50% thrust, which is much larger than the required thrust of the drone, 167gf. The initial current estimate from the optimization procedure was 2.1A, however, it falls within the FOS set by the power calculation.

Based on the maximum current limitation of the motor of 19A and a nominal battery voltage of 14.8V, the ESC was chosen. The ESC would be the "Flycolor Raptor 5" with the option of a 20A limit. This limitation guarantees our motors do not burn out due to excess power. As our team opted for 4 separate ESCs, the Flycolor was an optimal choice as it was lightweight, at 6.5g, and small enough to fit within the arms of the drone.

The power distribution unit chosen was Matek System's "PDB-XT60". Power distribution boards for multi-copter drones have become fairly standardized so the board was chosen based on size and reviews. This board easily fits in the main body of the drone and has enough outputs for 4 ESCs. It also provides a 5V output for the Arduino and sensors. The connections for the ESCs can support up to 25A of continuous current. The 5V BEC output supports up to 2A or current. Both surpass the power requirements of the systems listed above.

4.2.2. Software Subsystems

All of the drone's code was developed using C++; first, all of the code was done using the PACE simulation environment and was then ported over to the Arduino on board the drone.

4.2.2.1. Controller

The controller for the drone is one of the most important parts. The controller takes in the drone's desired position and commands the motors to move the drone. A poorly designed controller can, at best, result in some position error and, at worst, crash the drone.

The controller designed by the team was a cascade controller, consisting of five different PID loops: position, velocity, acceleration, angle, and angular velocity. This division was necessary because PID controllers are effective only for linear systems. By breaking down the drone's movement in this way, we transform a complex nonlinear problem—such as using a motor to achieve a desired position—into a series of linear problems.

The process works as follows: first, we control the motor to achieve a constant angular velocity. Then, this constant angular velocity is used to attain a desired angle. Next, we achieve the desired angle to create a desired acceleration, which in turn yields a desired velocity, ultimately leading to the desired position. Each of these steps represents a linear problem, making it possible for the PID controllers to manage them effectively. This layout is illustrated in the block diagram shown in Figure 22.

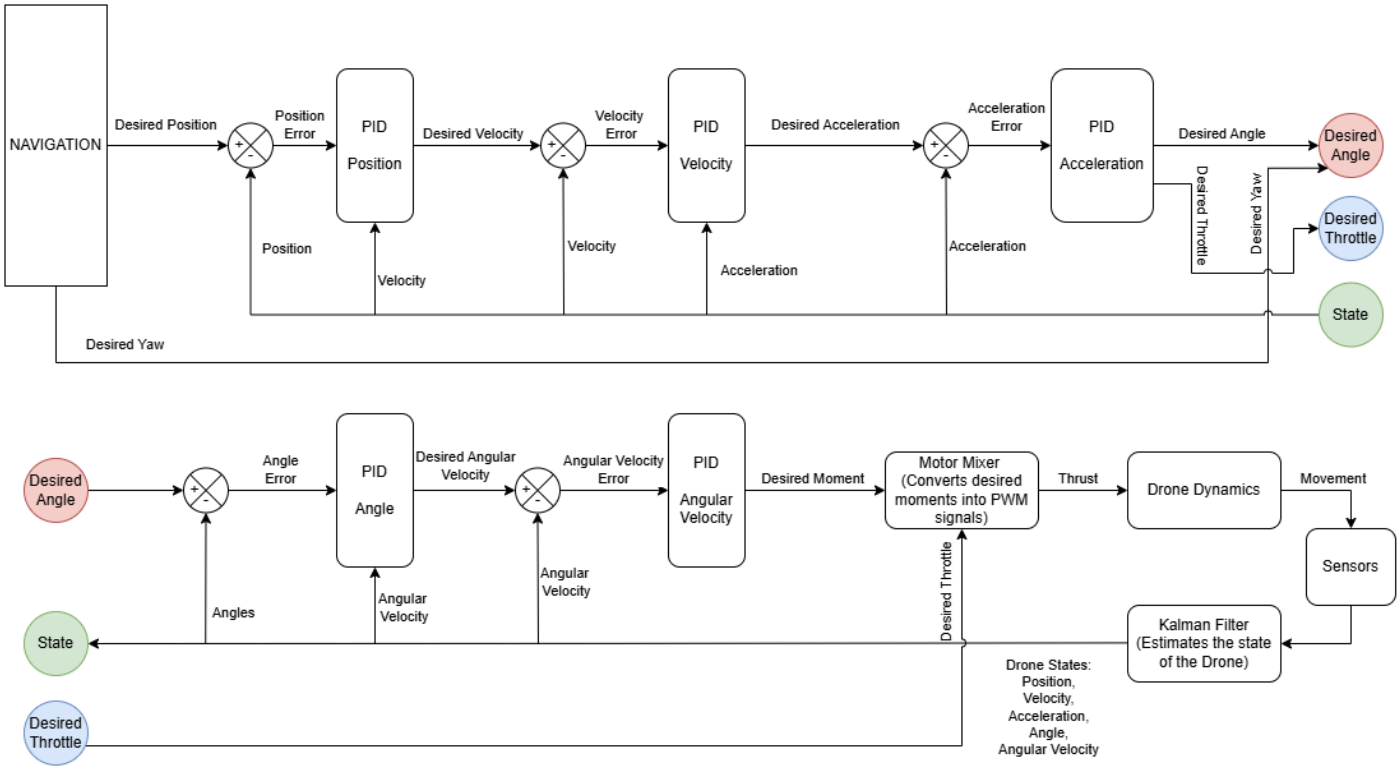

Figure 22: Controller block diagram.

This PID layout is very robust and can deal with disturbances in each of the sections due to the integral gains in each PID section. This setup also allows for a lot of customization when tuning and lets the team have the drone be very responsive and get to waypoints quickly and efficiently; the PID gains used can be seen in Figure 23.

```
float KP_Pos[3] = {.3,.3,.8} : Proportional Pos gain 3rd is throttle;
float KD_Pos[3] = {.1,.1,.2} : Derivative Pos gain 3rd is throttle;
float KI_Pos[3] = {0.001,0.001,0}: Integral Pos gain 3rd is throttle;

float KP_Vel[3] = {3,3,1.5} : Proportional Vel gain 3rd is throttle;
float KD_Vel[3] = {0.7,0.7,0.3} : Derivative Vel gain 3rd is throttle;
float KI_Vel[3] = {0.5,0.5,0.001}: Integral Vel gain 3rd is throttle;

float KP_Accel[3] = {0.6,0.6,0.01} : Proportional Acel gain 3rd is throttle;
float KD_Accel[3] = {0,0,0} : Derivative Acel gain 3rd is throttle;
float KI_Accel[3] = {0.01,0.01,0.1}: Integral Acel gain 3rd is throttle;

float KP_Angle[3] = {3.1,3.2,.3} : Proportional Angle gain 3rd is yaw;
float KD_Angle[3] = {.09,.09,0.2} : Derivative Angle gain 3rd is yaw;
float KI_Angle[3] = {0.001,0.001,0.001}: Integral Angle gain 3rd is yaw;

float KP_Rate[3] = {.012,.012,.1} : Proportional Rate gain 3rd is yaw;
float KD_Rate[3] = {0.003,0.003,0.01} : Derivative Rate gain 3rd is yaw;
float KI_Rate[3] = {0.001,0.001,0.01}: Integral Rate gain 3rd is yaw;
```

Figure 23: PID gain values for the simulated drone.

One interesting thing that was learned through trial and error was that integral gain is both a curse and a blessing. Too much integral gain will cause the drone to have huge overshoots under normal conditions. However, whenever there is wind, the drone performs better with large integral gain, as it's able to correct the disturbance faster. The team realized, however, that since the drone will be inside with little to no wind or disturbances, the integral gain could be kept to really small values like 0.01.

However, the integral gain of 0.01 can still create unwanted results if not handled properly. Since the drone has a velocity limit that we set, whenever the controller tries to tell the drone to move faster than that limit, the code will max out the desired velocity at the set max value. This works great until the integral gain gets involved. The integral gain will only show that the desired velocity isn't being reached and will start increasing the response. To the point where when the drone reaches its position and needs to slow down, the integral gain has taken over, and the drone will shoot past its desired spot. To combat this, the team set up a system where whenever the velocity limit was reached, the integral gain wasn't allowed to continue to grow. This worked great, as the integral gain could still work in order to bring the drone to its maximum velocity, but it would then get rid of the whole overshooting problem.

Another issue that was found in the controller was a weird pathing issue when trying to go to points that required both x and y movement. This weird behavior can be seen in Figure 24. This was caused by the X, Y, and Z controllers essentially being autonomous and not communicating with each other. This resulted in each direction trying to achieve its goal as soon as possible. This can be seen in the Figure 24, where the Y error is immediately reduced. This behavior is bad because it causes the drone to travel a longer distance and accelerate more than needed, draining more battery. The desired behavior can be seen in Figure 25.

Figure 24: Undesired behavior in the controller.

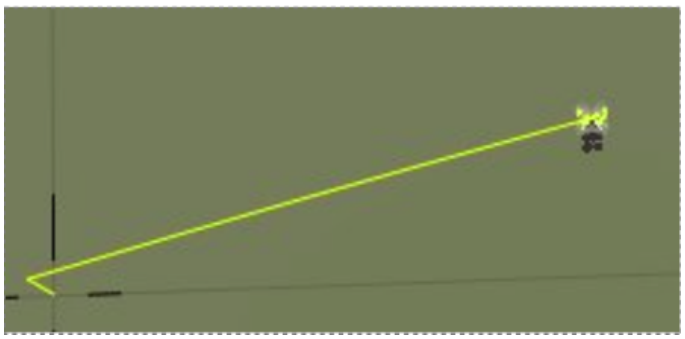

Figure 25: Desired behavior in the controller.

The desired path shown in Figure 25 was achieved by normalizing the desired velocity of each of the different directions by the maximum velocity. This allowed the drone to fly in a straight path and save time and battery.

4.2.2.2. Kalman Filter

The Kalman filter is an essential part of the drone's software, as it lets the drone know where it is. Sensors are great for getting data about where the drone is, but sensors, by themselves, have errors and slow refresh rates. Those things aren't good when you want to control a drone accurately. A Kalman filter will take all of the different sensor measurements and combine them to give a very accurate position at the refresh rates necessary for an accurate controller.

4.2.2.2.1. Kalman Filter Design

At the beginning of the fall semester, all of the teams were given Dr Johnson's Kalman filter paper **(Link)**. that was designed explicitly for IMU and GPS sensor fusion. This paper was excellent for getting a baseline for how to set up a Kalman filter; however, at the beginning of the spring semester, teams were told the competition would now use a MoCap system instead of a "GPS." This would allow the drone to get not only a position measurement but an orientation measurement as well. Seeing this, the team decided to create 2 Kalman filters. Since the position of the drone and orientation of the drone arent coupled through dynamics (In the sense that we were using the IMU for predition update), it was decided that a 2 kalman filter aproach would actually reduce the computational requirment of the flight computer without reducing accuracy. Going from needing to calculate a 16x16 matrix to a 7x7 and a 9x9, which is a 50% reduction in computation. The position and orientation of Kalman filter block diagrams can be seen in Figure 26 and Figure 27.

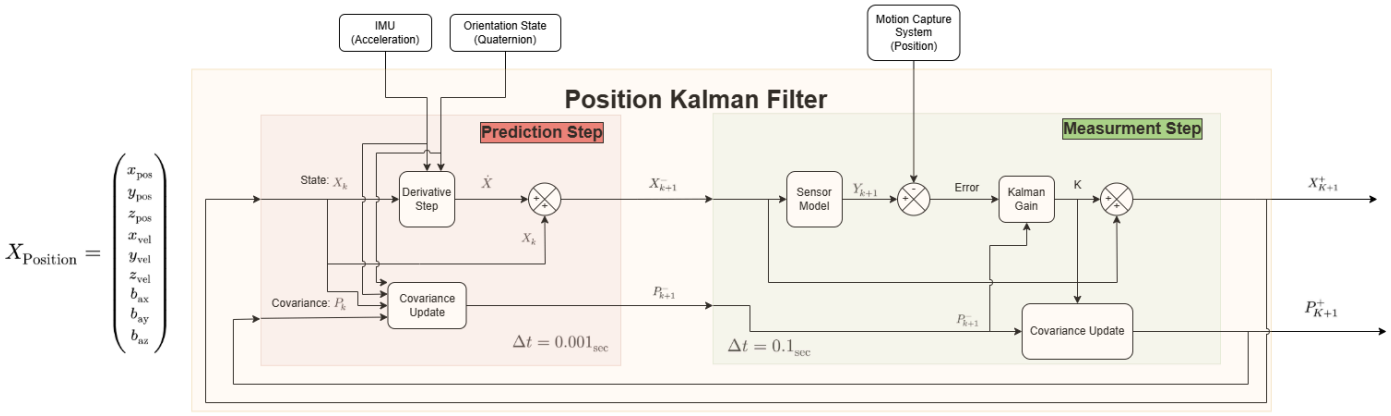

Figure 26: Kalman filter block diagram for position.

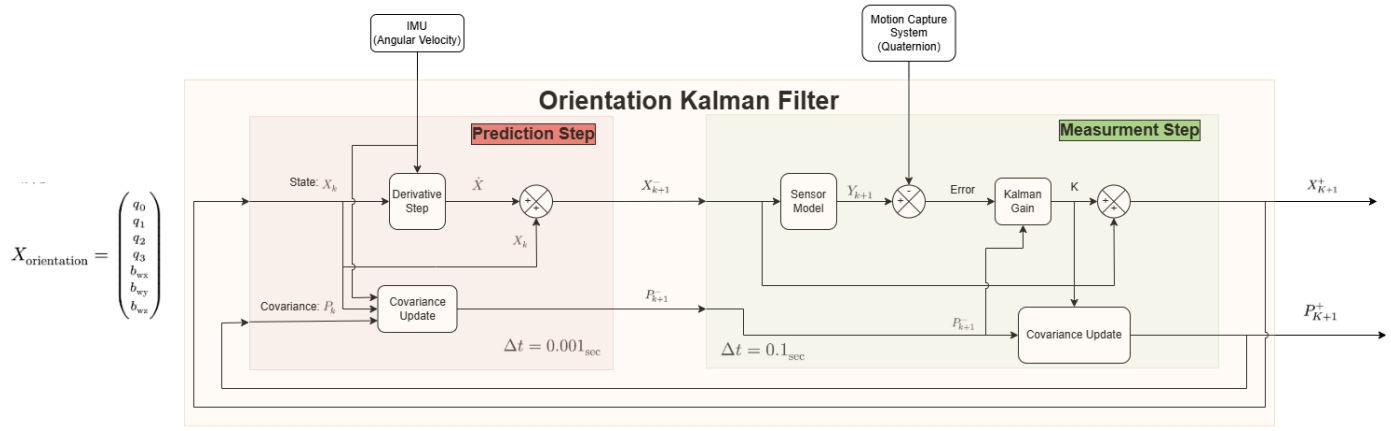

Figure 27: Kalman filter block diagram for orientation.

The Kalman filter for the orientation will try to estimate the orientation quaternions, as well as the IMU gyro biases. Predicting the Biases is essential before IMUs are known for being very inaccurate over long periods of time due to the biases they come with. Predicting them allows the drone to have very good predictions, even if the mocap system stops giving correction measurements. The same goes for the position of the Kalman filter. The only difference is that the states it tries to estimate are position, velocity, and accelerometer biases. The position Kalmn filter also needs the orientation estimate in order to know which way is down and cancel out the gravity vector from the accelerometer.

The Kalman filter uses the IMU data to perform the prediction step. Since the IMU gives acceleration and angular rate data, these can be used as the equations of motion and propagate the motion of the drone. This prediction step can happen every 0.01 seconds. Next, about every 0.05 seconds, the Mocap module gives its position and orientation measurement, which is called the update step. This step will take in the current predicted state and the estimated covariance and will use the position or orientation data to give a better estimate of the position or orientation and recalculate the covariance. Every time step, the Kalman filter sends its most up-to-date state vector to the controller.

4.2.2.2.2. Kalman Filter Implimentation

In order to implement the Kalman filter effectively, the team took many small steps. This proved to be very necessary and decisive, as the FireFlighters were the only team to get a working Kalman filter. The approach was simple. The Kalman filter would first be done in Matlab, as the native matrix multiplication and easy debugging features allowed for mistakes to be minimized and issues in implementation to be quickly tracked down. This process still took hundreds of hours, and Matlab's Kalman filter wasn't fully implemented until late January. An image of the Kalman filter working in Matlab using a truth and estimate cube can be seen in Figure 28.

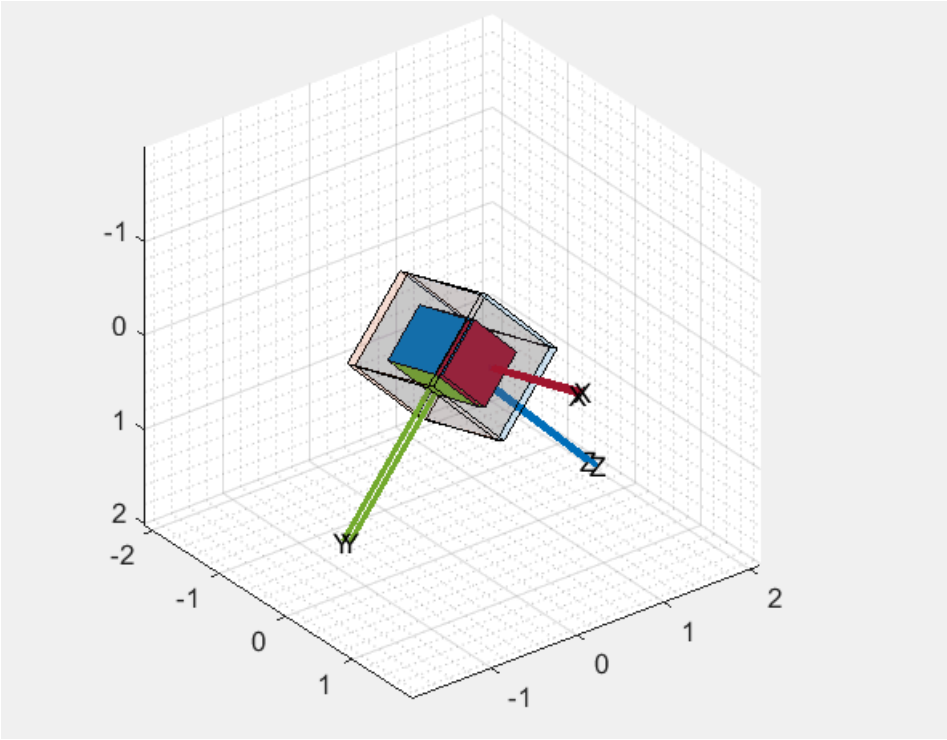

Figure 28: Kalman filter working in Matlab.

Next, the Kalman filter needed to be transferred to PACE. In order to do this, the Matlab code was ported over to C++ using ChatGPT to avoid human errors, and it was inputted into a separate C++ file. This file was used to test the output of each function. A set of the same random inputs would be fed into each function in both the C++ and Matlab code, and the outputs would be checked to make sure they matched. This was done for each function, and then the functions were slowly linked together. While slow at the start, this process allowed for the Kalman filter to work on the first try when copied and pasted over to PACE. The outcome can be seen in Figure 29 with the red drone being truth and the blue drone being the Kalman filter approximation (PACE offsets the estimated and truth by (0.1,0.1,0.1) ft so the user can tell the difference.)

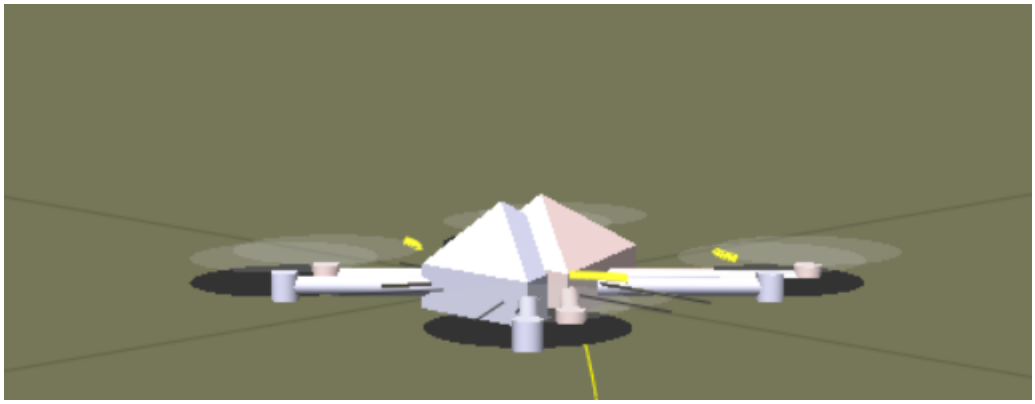

Figure 29: Kalman filter working in PACE.

Lastly, the Kalman filter needed to be put on the Arduino on the drone. This proved to actually be the most difficult, as the mocap system needed to be linked to the drone, and the IMU's orientation needed to be corrected. Once those issues were ironed out, it was found that the Mocap data the team was getting was delayed by a full 1 second. Dr. Valente investigated this, and it turned out to be an issue on his end. This issue was never resolved, and it meant the Kalman filter would always be a full second behind. The estimate of the position and orientation would be awful because the prediction step would predict a good estimate of the drone's position. Still, the measurement step would tell the drone it's actually 1 second in the past, and it would mess up the covariance and IMU biases. Another faital flaw that will be discussed later was that the arduino was slow. Operating around two hz with everything running, it's WELL off the 1000hz expected in the simulation. This caused the Kalman filter to explode with the normal measurement and process covariance settings and caused the team to have first to figure out that was the problem and not the code, and second sit there and change the covariances of the sensor noise till the Kalman filter decided to converge. This further means that the Kalman filter on the drone was very inaccurate. An image of the Kalman filter working onboard the drone and sending the data to the ground control station as the drone was being carried around the room can be seen in Figure 30.

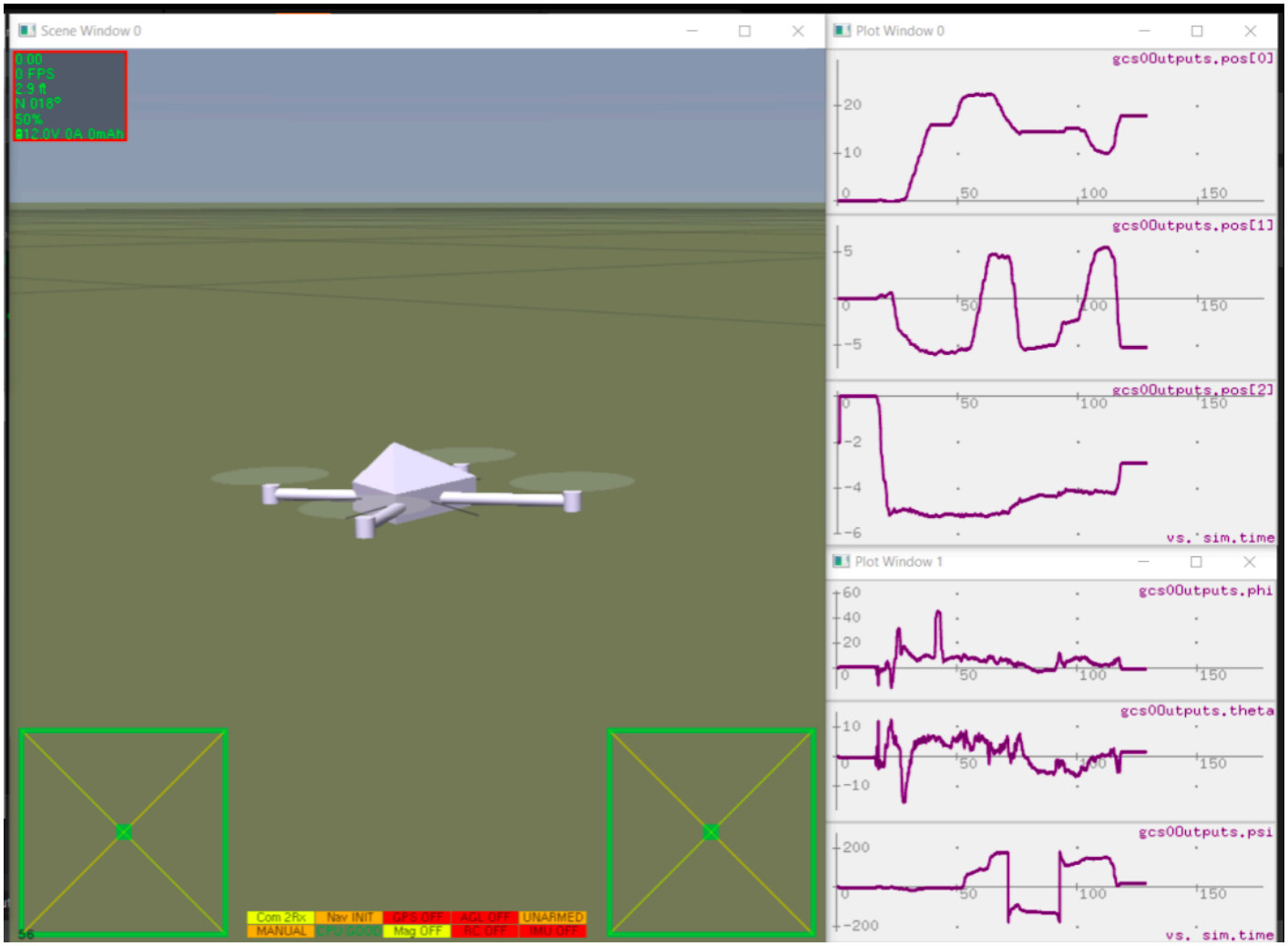

Figure 30: Kalman filter working on the real drone and showing up in the ground control station.

4.2.2.3. Navigation

For the navigation, the drone's code uses waypoints with a lot of different parameters. In order to accomplish this in the code, a waypoint class was used, which stored everything shown in Equation 5.

$$[x, y, z, \text{Radius}, \text{Max\_Velocity}, \text{Hold\_Time}, \text{Turn}, \text{Turn\_Error}, \text{Turn\_Vel\_Error}] \tag{5}$$

The waypoint would have the position information and the error radius around the point. The time the drone needed to stay at the point, the maximum velocity it could have when it is in the radius. The waypoint could also have the "Turn" parameter, which is a true or false value that determines if the drone will turn itself to face the waypoint. If this is true, the angle error and angle velocity error can also be stored. All of these values together give a very robust waypoint class.

Using this system. Waypoints can be rapidly added for the drone to follow and can be seen in Figure 31.

```
Drone.addWaypoint(Waypoint(0, 0, -2, 0.2, 0.2, 2, false, 0.05,0.05));

Drone.addWaypoint(Waypoint(0, 5, -10, 0.2, 0.2, 2, true, 0.05, 0.05));

Drone.addWaypoint(Waypoint(5, 5, -10, 0.2, 0.2, 2, false, 0.05, 0.05));

Drone.addWaypoint(Waypoint(5, 5 + 33 - 4, -10, 0.2, 0.2, 2, false, 0.05, 0.05));

Drone.addWaypoint(Waypoint(-5, 5 + 33 - 4, -10, 0.2, 0.2, 2, false, 0.05, 0.05));

Drone.addWaypoint(Waypoint(-5, 9, -10, 0.2, 0.2, 2, false, 0.05, 0.05));

Drone.addWaypoint(Waypoint(0, 9, -10, 0.2, 0.2, 2, false, 0.05, 0.05));

Drone.addWaypoint(Waypoint(0, 5 + 33 - 4 - 4, -10, 0.2, 0.2, 2, false, 0.05, 0.05));

Drone.addWaypoint(Waypoint(2, 5 + 33 - 4 - 4 - 3, -10, 0.2, 0.2, 2, true, 0.05, 0.05));

Drone.addWaypoint(Waypoint(2, 5 + 33 - 4 - 4-3, -3, 0.2, 0.2, 2, false, 0.05, 0.05));

Drone.addWaypoint(Waypoint(2, 5 + 33 - 4 - 4 - 3, -5, 0.2, 0.2, 2, false, 0.05, 0.05));
```

Figure 31: Drone waypoints in the code

However, there still isn't any logic that actually gets the drone to know how to move onto the next waypoint. In order to do that, a finite-state machine was made, and the block diagram can be seen in Figure 32.

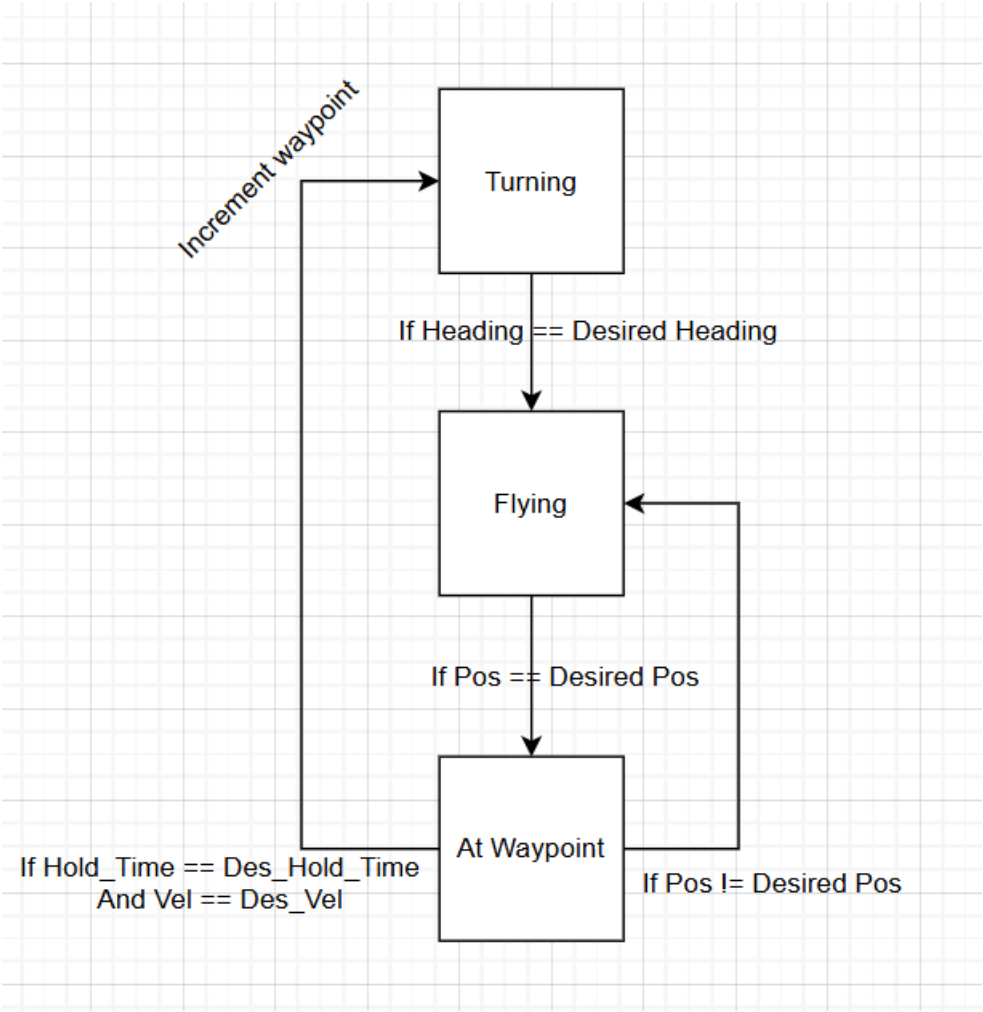

Figure 32: Flowchart for the state machine that handles the waypoints.

The state machine has 3 modes: a TURNING mode, which the drone is only in when the turning parameter is set for the waypoint. This mode will stay until the drone has fully turned to its correct heading and has the desired angular velocity. Once those conditions are met, the next state is set, which is the FLYING state. This state continues until the drone detects that it's within the radius of the waypoint. Then it switches to the AT_WAYPOINT state, where it will check to make sure that the drone doesn't exceed the maximum velocity and holds for the correct amount of time. If the drone ever flies out of the waypoint's radius, the FLYING state will be reapplied, but if everything is good, the drone will increment to the next waypoint in the list, and the TURNING state will be replied. The full state machine can be seen in Figure 33.

```
Waypoint& currentWaypoint = waypoints[currentWaypointIndex];
switch (currentState) {
case TURNING:
    if (currentWaypoint.Turn_to) {
        Waypoint& PreviousWaypoint = waypoints[currentWaypointIndex-1];
        float Delta_Y = currentWaypoint.y - PreviousWaypoint.y;
        float Delta_X = currentWaypoint.x - PreviousWaypoint.x;
        if (Delta_Y == 0 && Delta_X == 0) {
            std::cout << "DIVIDE BY ZERO No Turning----------------------------" << std::endl;
            currentState = FLYING;
        }else{
            psi_des = atan2(Delta_Y, Delta_X);
            std::cout << "Turning" << std::endl;
            if (currentWaypoint.isTurned(psi,psi_dot, psi_des)) {
                currentState = FLYING;
            }
            else {
                Pose_Desired[0] = PreviousWaypoint.x;
                Pose_Desired[1] = PreviousWaypoint.y;
                Pose_Desired[2] = PreviousWaypoint.z;
                Pose_Desired[3] = psi_des;
                return Pose_Desired;
            }
        }

    }
    else {
        psi_des = psi;
        currentState = FLYING;
    }
case FLYING:
    if (currentWaypoint.isAtWaypoint(droneX, droneY, droneZ)) {
        currentState = AT_WAYPOINT;
        currentWaypoint.resetHoldTime();
        std::cout << "Arrived at waypoint " << currentWaypointIndex << std::endl;
    }
    break;

case AT_WAYPOINT:
    if (currentWaypoint.isAtWaypoint(droneX, droneY, droneZ)) {
        currentWaypoint.updateHoldTime(deltaTime);
        std::cout << "Holding at waypoint " << currentWaypointIndex << std::endl;

        // Check if conditions are met to transition
        if (currentWaypoint.canTransition(velocity)) {
            std::cout << "Conditions met, Moving to next waypoint " << currentWaypointIndex << std::endl;
            currentWaypointIndex++;
            cntrl->integralPos[0] = 0;
            cntrl->integralPos[1] = 0;
            currentState = TURNING;
        }
    }
    else {
        // If we are no longer at the waypoint, reset hold time
        currentWaypoint.resetHoldTime();
        std::cout << "Moved away from waypoint " << currentWaypointIndex << ".\n";
        currentState = FLYING; // Try to re-approach
    }
    break;
```

Figure 33: Code for the state machine that handles the waypoints.

4.2.2.4. Datalink

Datalink is the system that gets the drone to send whatever data the team wants over Wi-Fi. It also allows us to send data to the drone via Wi-Fi. Getting this subsystem was the top priority at the beginning of the spring semester. This is because, without datalink, it's really hard to see how your controller is doing and the rise time setting time and overshoot for the controller. So essentially, without datalink, tuning the PID turns from being a scientific purposeful process into a "does this feel better" process. Because of this, the datalink was finished in early February. Datalink allows the drone to send whatever data we want using packages called messages. All of the messages the drone used can be seen in Figure 34.

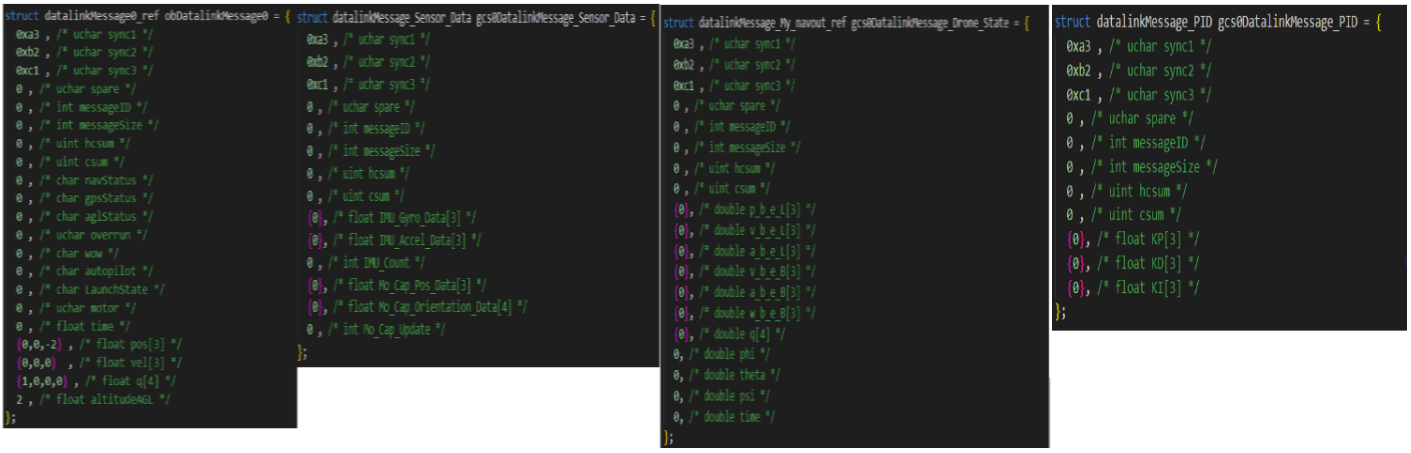

Figure 34: Different messages in the code.

Datalink, however, turned out to be the nail in the coffin for autonomous flight. It was found late into the semester that sending Data using datalink took 100 milliseconds. Without multi-threading, this would severely limit the rate the Arduino could run, meaning no active control loop could adequately control the drone. This meant that the hopes of autonomous flight were scratched, and any autonomy systems needed to be performed using a hard carry demonstration.

4.2.2.5. Ground Control Station

The ground station is an integral part of drone operations, as its role is essential for data collection and allows for the debugging of the drone even if the drone crashes and the onboard computer dies. It's also necessary to have, as spectators want to see how the drone is doing, and just with a quick glance at the ground control station, you should be able to understand the state of the drone and the progress it is making through the mission.

Everything that was displayed on the ground control station was: Position, orientation, Fire location, Fire detection, Waypoint number, Desired location, Payload dropped. The ground control station can be seen in

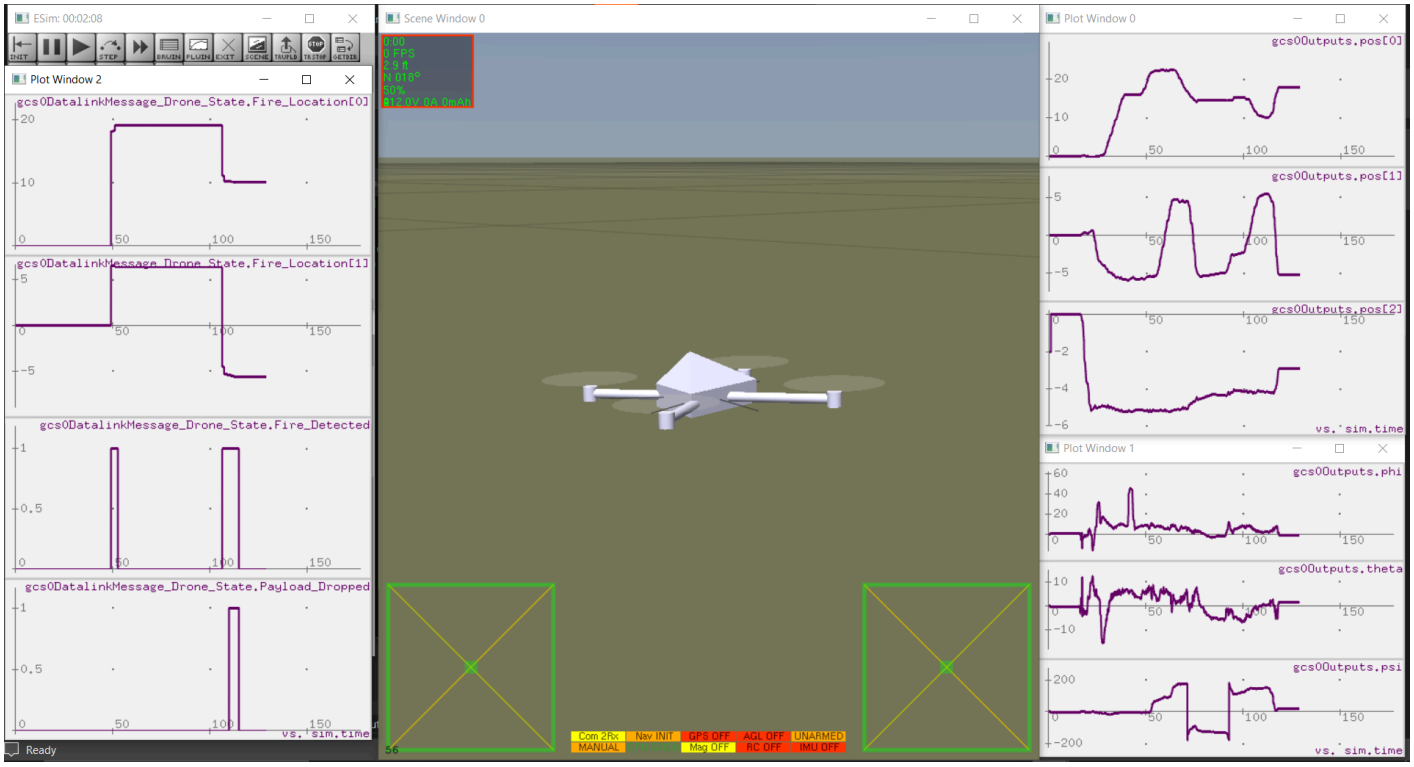

Figure 35: Ground control station after a full hand carry test with fire detection and payload drop.

4.2.2.6. Thermal Sensor

The thermal sensor will be how the drone will detect the “fire” and know where to drop the payload. The plan is for the thermal sensor to send a thermal image to the Arduino for processing. An example of a processed image can be seen in Figure 36.

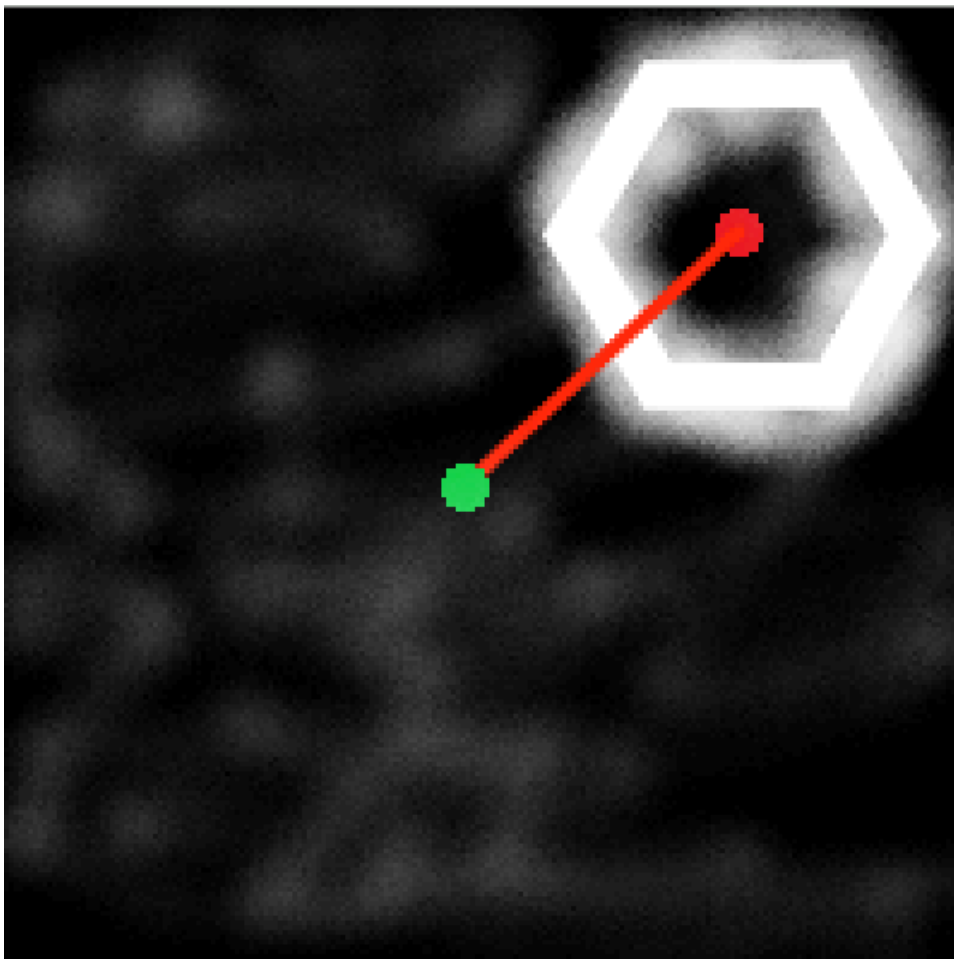

Figure 36: White hot image showing an example of the thermal camera seeing the target and estimating a location to move to.

The team plans on processing the thermal image data by first taking advantage of the fact that we will know the range of temperature at which the resistors will be operating. Then, the drone can filter out all the pixels that aren’t within that range. Finally, by averaging their positions, the drone will find the centroid of all of the “Hot” pixels that it sees in the image. This centroid is shown in Figure 36 as the red dot. Next, the drone knows it is right over the center of the image, which is shown as the green dot. The Arduino will then calculate a line from the centroid to the current location and the heading angle the drone needs to turn. Once the drone is facing the heat source, it will move slowly to it until it is right overtopped. Then, it will initiate the payload drop sequence. Whenever the thermal sensor detects a fire the detection state will be sent to the GCS, and the drone will stop trying to go to the next waypoint, and instead will try to go to the fires centroid location. Once the drone is over the centroid, the payload will drop, and the next waypoint will send the drone back to the takeoff location.

4.2.3. Payload Mechanism

The final subsystem is the payload mechanism. The team focused on this early on because we wanted a reliable and reusable mechanism that could be opened and closed many times. Other teams had mechanisms that, when opened, remained open for the duration of the flight. We wanted ours to be more of a bomb bay and close after dropping. The solution was a linkage mechanism with a sliding door, as seen in Figure 37.

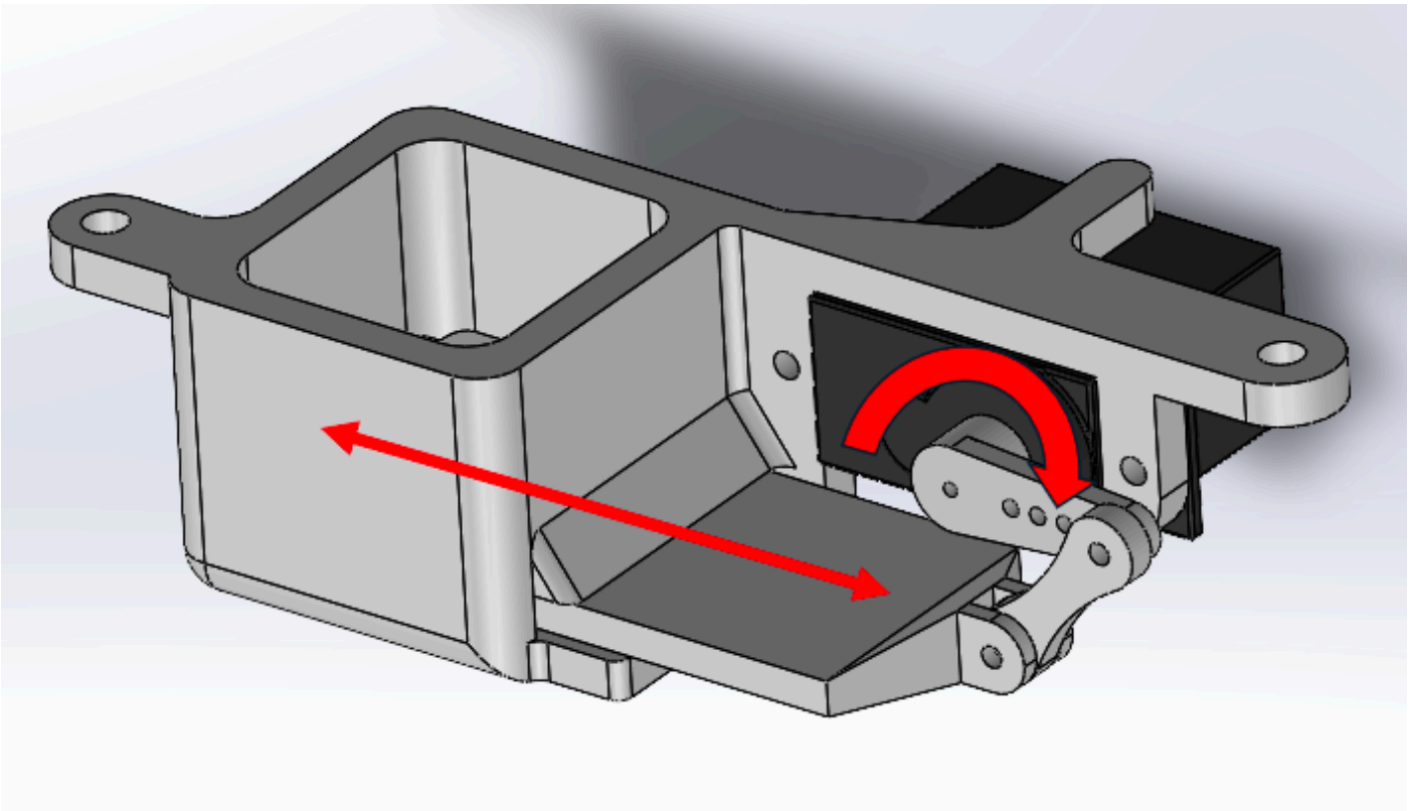

Figure 37: Payload mechanism

Designing the linkage so that the sliding door doesn't get jammed was a tedious and iterative process; the slot the door slides into needed to be extended, and the opening needed to be widened, but after many iterations, the final product seems to be flawless, and after hundreds of opening and closing cycles, it hasn't gotten stuck.

4.3. Full Simulation

When combining all of the parts of the code, a full simulation of the drone's mission can be conducted. All of the parts of the flight profile are simulated. The flight profile that the simulation will follow can be seen back in Figure 2.

The drone taking off can be seen in Figure 38. Then, the drone will turn towards the fire zone and fly to it, which can be seen in Figure 39.

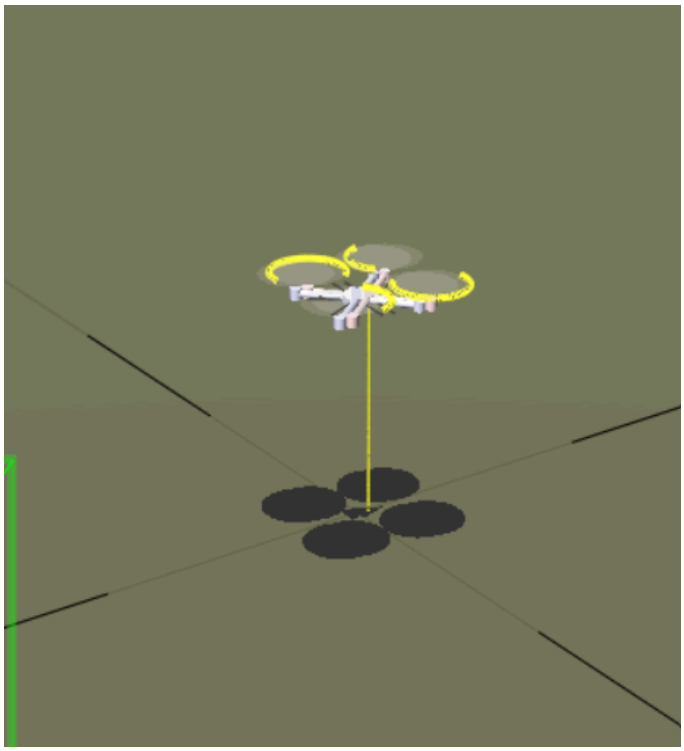

Figure 38: Drone taking off.

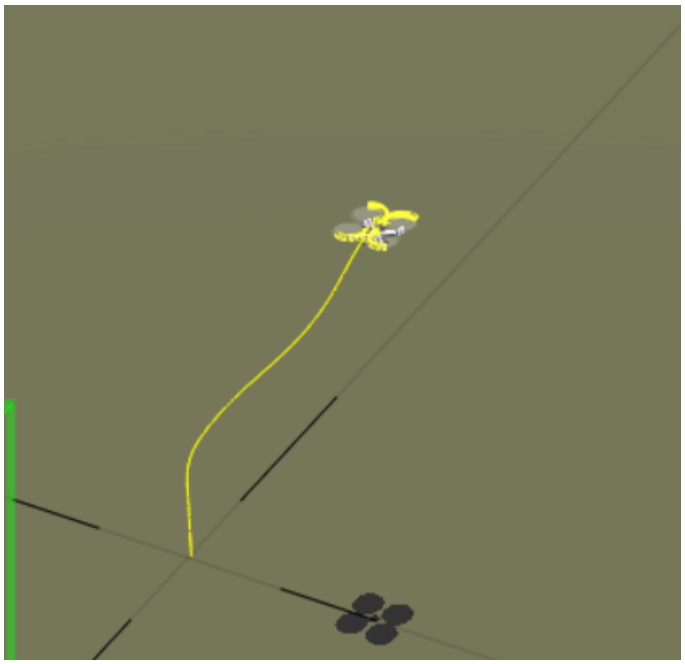

Figure 39: Drone moving to the fire zone.

The drone will take off and hover at 2 feet. Then, it will turn towards the fire zone, rise to 10ft, and move the fire zone edge all in one movement.

Next, the drone will then perform the search pattern, which can be seen in Figure 40. It will then detect the fire, turn to the fire, and move overtop of it, and can be seen in Figure 41.

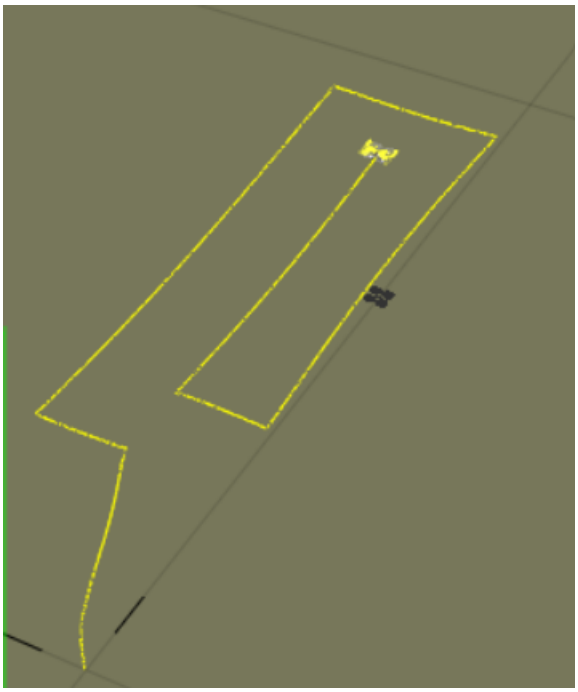

Figure 40: Drone performing the search pattern.

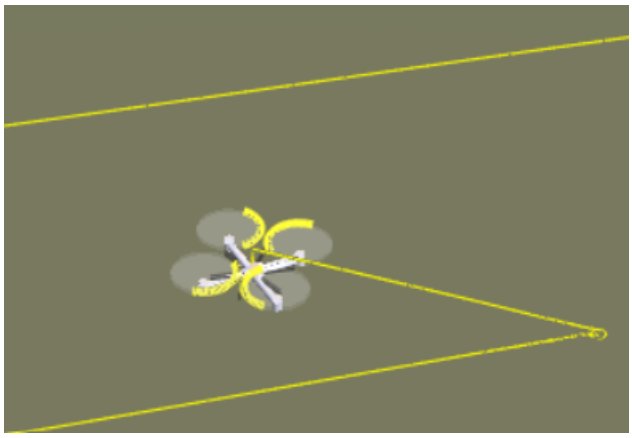

Figure 41: Drone moving to the fire.

The search pattern shown in the first two passes should be able to find the fire. In the simulation, however, the drone flies on the full path. The simulation doesn't have a way for the drone to actually detect the fire yet, so when the drone turns and moves over the "fire," it's just a waypoint.

Next, the drone will descend to the fire and drop its payload, which can be seen in Figure 42. Then, the drone will fly up to 5 feet, as seen in Figure 43.

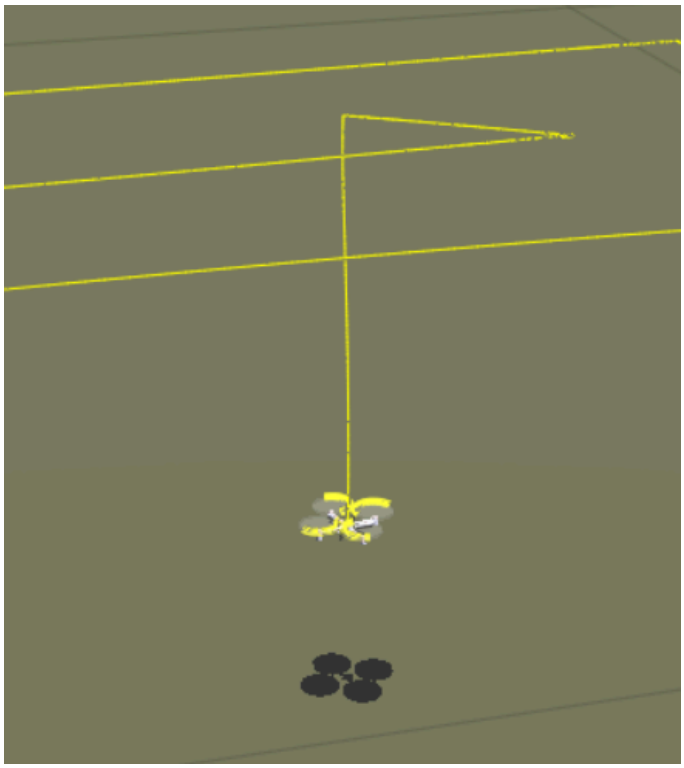

Figure 42: Drone descending over the fire and dropping the payload.

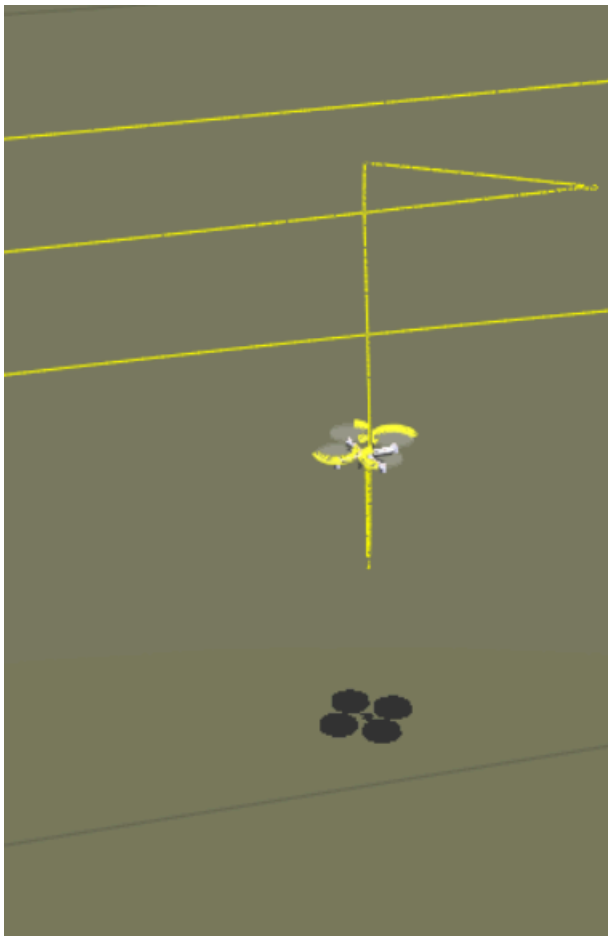

Figure 43: Drone raising its altitude after dropping the payload.

When the drone drops down to the fire, it does so in a controlled manner, and its velocity never exceeds three ft/s. The drone will then hover over the fire for 5 seconds and then release the payload. The drone will fly up to five feet to get away from the fire. It doesn't go all the way up to 10ft because it only cruises at 10ft, so the camera can see more area. The drone will fly at 5 feet in order to save battery.

Finally, the drone will move at least 1.5m away, as can be seen in Figure 44. Lastly, the drone will land softly, Figure 45.

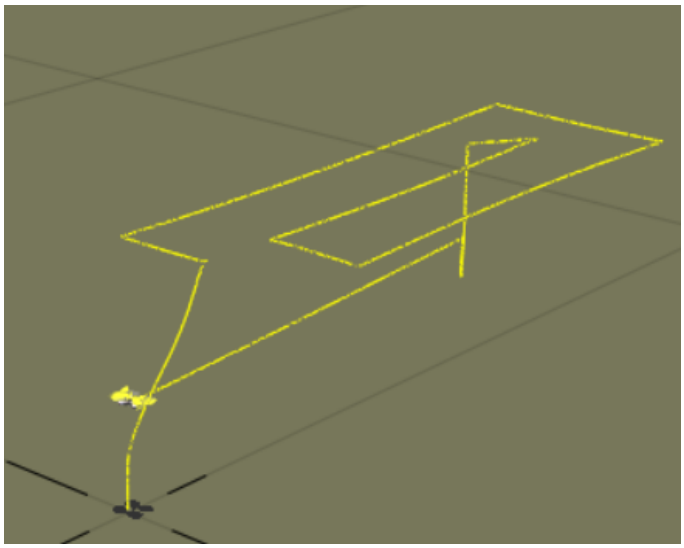

Figure 44: Drone moving away from the fire.

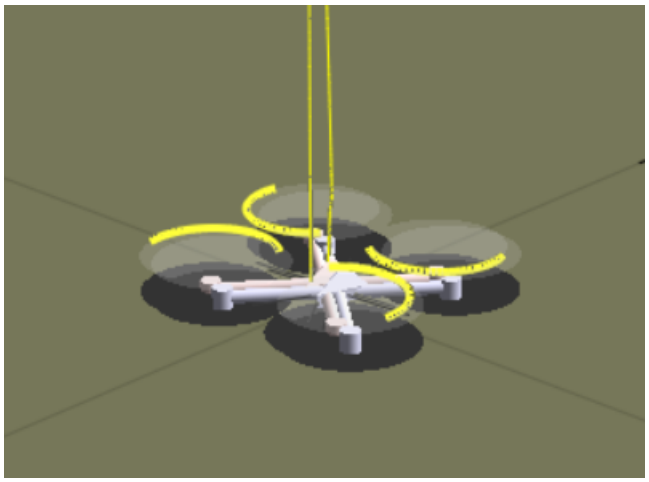

Figure 45: Drone landing.

In the simulation, the drone flies back to the starting position. However, when the real drone is flown for the competition, it will just go 1.5m away to conserve battery. The simulation was set up this way because the team was unsure if the drone could land in the fire zone. The landing sequence isn't just a waypoint on the ground. It's actually a series of points, so the drone can slowly and softly touch the ground.

5. Design Challenges, Risks, and Mitigations

Throughout the design and development process we ran into many challenges some of which was due to our own oversights, some of which from elements beyond our control. One of the biggest problems we ran into as a group was how brittle our drone actually was. Due to our drone being largely 3D printed it let to multiple common breakpoints due to high stress concentrations or the print shearing down the print lines.

5.1. Material Limitations

During early testing one of the common breaks we got was a delayering of the leg when we tried to land our drone. It would often sheer right at the print line causing us to have to shift the print orientation from horizontally down the leg to vertically. Another issue we ran into was the amount of material connecting the arms onto the baseplate. We had a pretty tight C-bracket joint with screws around it so the actual thickness of the material was only around 3mm supporting the entire arm and motor forces. This led to another very common break point of the arm sheering off the baseplate shown in Figure 46.

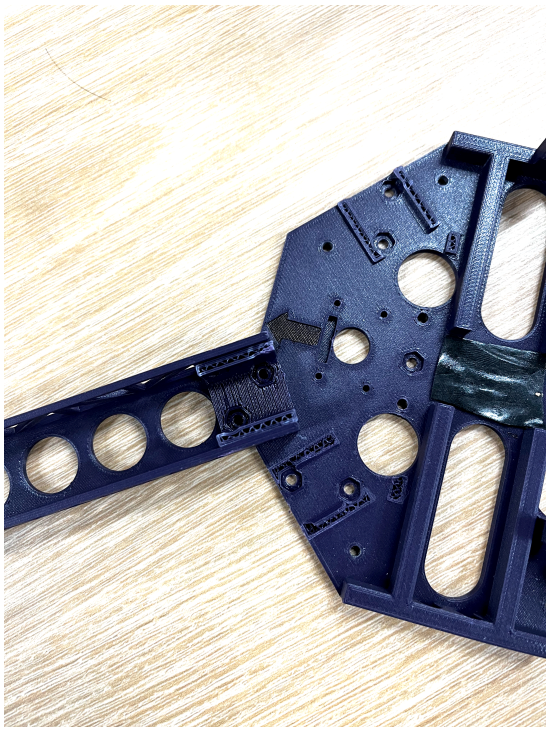

Figure 46: Most common baseplate breakpoint.

5.2. Testing Without Risk

Due to the brittle nature of the drone and the guess and check nature of early gain tuning it led to a bad crash and break more often then not. This led to many hours of repairs before another short flight and break. In order to not get stuck in this destructive loop we decided to take inspiration from another team to create our own test stand so we would have a safe way to test our gain combinations. A picture of our test stand will be included below.

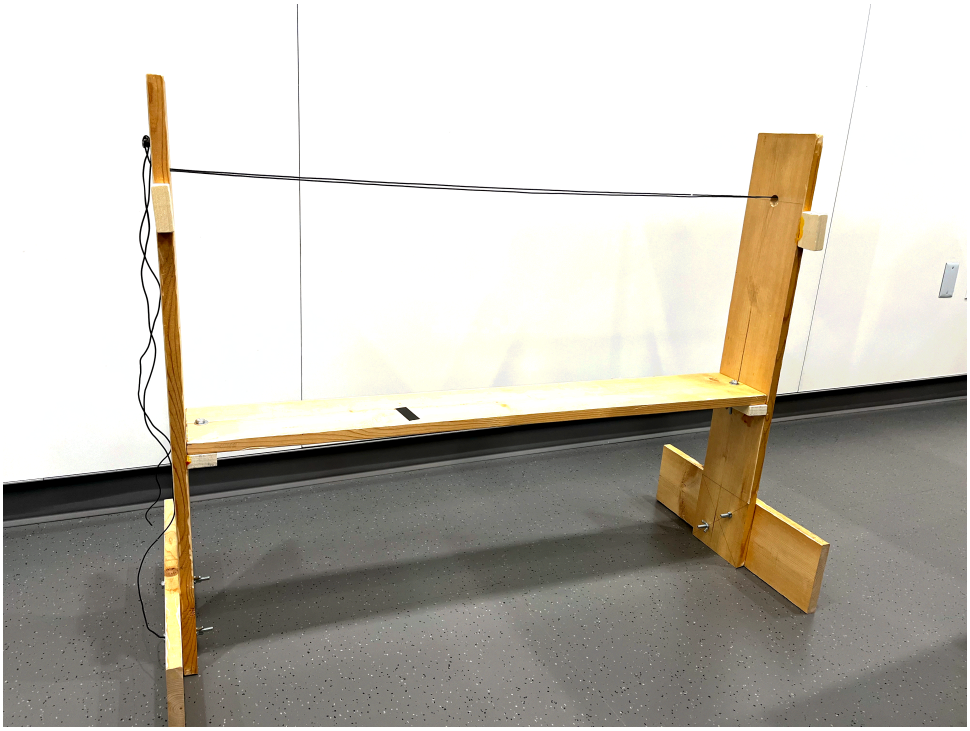

Figure 47: Final version of the test stand

Initially the test stand did not have a string to hang the drone on but a rigid wooded rod that would slide through the holes on each side. This was made to mimic what we saw in YouTube videos on other simple one axis test stands. Later we realized that since our rod is not running through the drone's center of mass we were effectively testing on an inverted pendulum. This led to no real progress in getting better. To fix this our group decided to thread a wire through our center of mass and tie it to the test stand. We assumed since the wire provides less resistance to motion than the rod this would be an even better solution. Which it mostly was apart for if the drone got too much thrust the drone would pull the wire to the side as it tries to escape its confinement which causes the oscillation to go wild. To get around this much of our gain tuning was done at lower thrust until we could get a satisfactory response.

5.3. Loose Wires

The last biggest recurring issue our team ran into was our wires coming out on landings or scrapes with the ground. This caused a couple misfires of the props and some non responsive controls from time to time. Getting around this took a bit of trial and error. At first we would just replace the wire with a tighter one hoping that it would not happen again. But this was growing to be a bigger issue, another team's receiver wire had slipped out of their Arduino causing the kill switch to become unresponsive. Upon seeing this and knowing how much our wires yearned for freedom we coded in a failsafe. If the Arduino saw the same exact output from the receiver for about 2 seconds (which with natural fluctuation of values is near impossible) it would force kill the motors. This was a valuable addition but it still didn't solve the initial problem our wires were still falling out occasionally, especially our main ground. The next solution was to try and tape down the leads which worked for a bit but was still not reliable. It was about this time where we had a bad crash where one of the arms bent upward and severed the antenati of the receiver and would have completely destroyed the drone if not for the failsafe we had built in Figure 48. Eventually to fix this problem we had to buy a screw locking attachment for the Arduino where we could thread our wire leads through and lock them into place as seen in Figure 49.

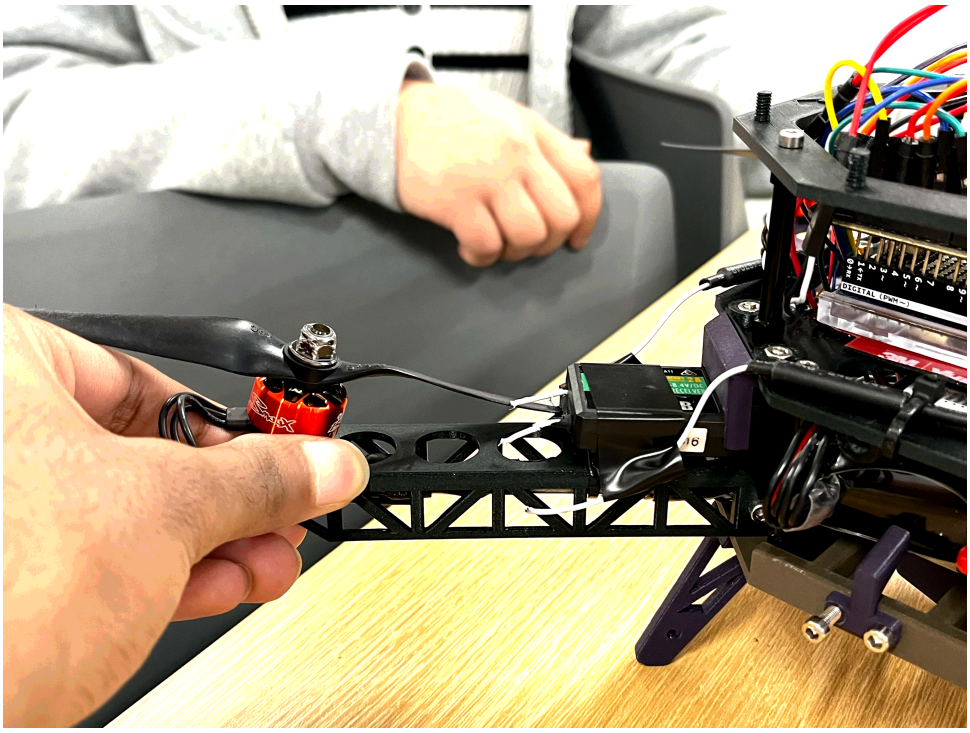

Figure 48: Cut receiver

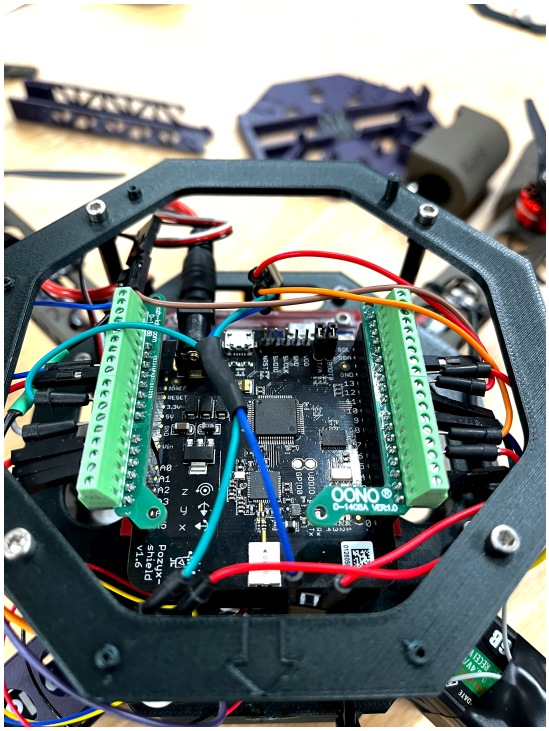

Figure 49: Wire-locking mechanism

6. Development Plan

6.1. Organization Chart

An organization chart is a graphic showing how a team is managed and who is leading each part of development. The organization chart for the Fire Flighters team can be seen in Figure 50.

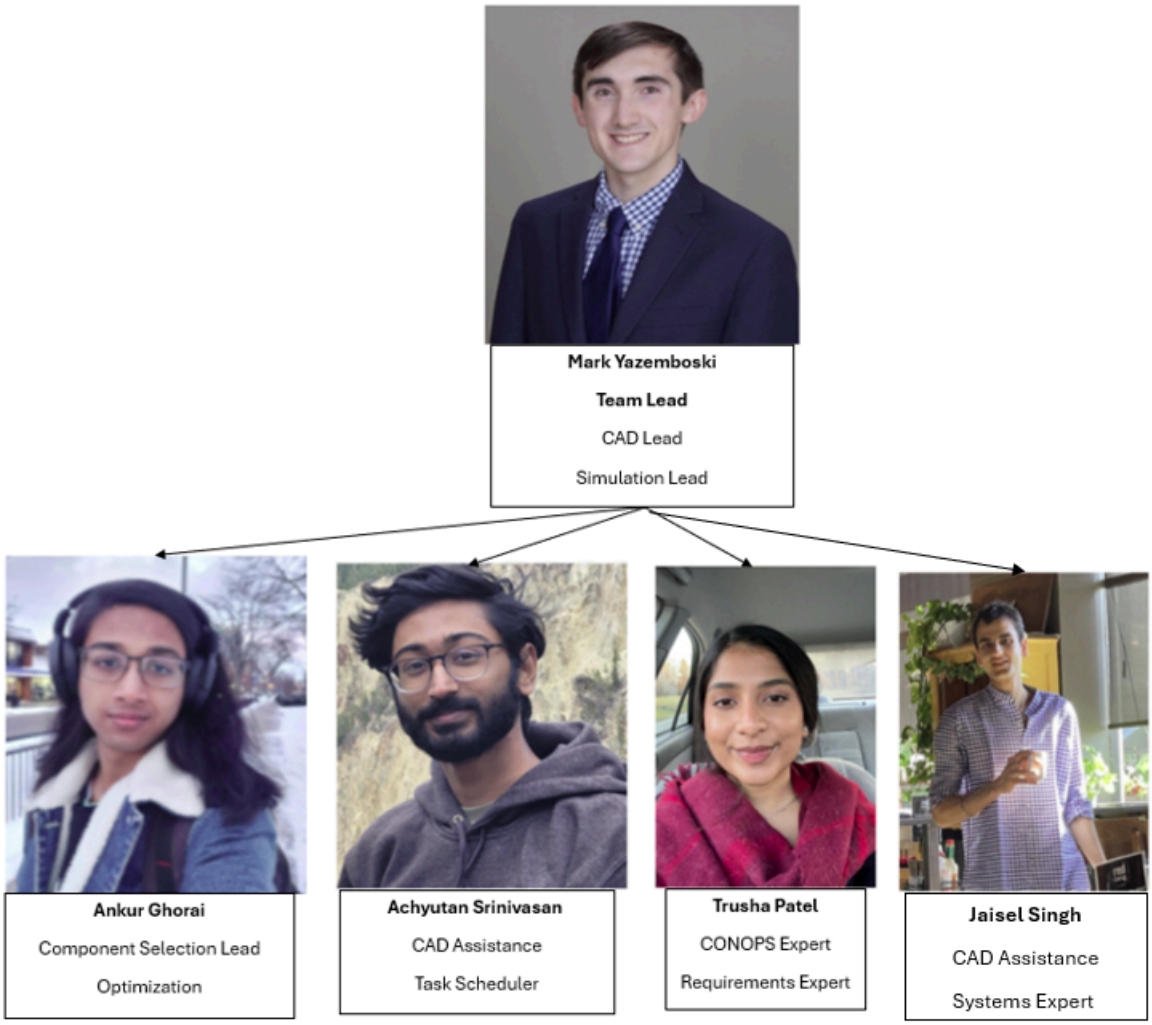

Figure 50: Image showing the organization chart for the team.

Mark took the team lead role due to his previous experience running a technical development team for the Streamline Charging start-up he helped cofound. This gave him tons of leadership and CAD skills that could be utilized very well as a team leader. Mark also had the most programming experience, as he took many CS courses and is currently taking AERSP 424 to get more experience in C++. Because of this, Mark is the team leader, CAD lead, and Programming lead.

Ankur has a lot of experience with component selection and choosing compatible electronics, having been the propulsion lead in the Penn State rocketry club. Ankur has also taken optimization courses from the Aerospace department and was already familiar with the ATSV software. Because of this, Ankur became the components expert and optimization expert.

Achyutan had the second most CAD experience and was very interested in the CAD side of the drone. Because of this, he assisted Mark with the drone CAD. Achyutan also took on the role of task scheduler for the next semester since he is so good with time management from his 22 years of life experience.

Trusha has prior experience in project management and technical knowledge of drone subsystems. Because of this, Trusha was tasked with breaking down the drone into its most basic functions so it could be built effectively and conducting research on connecting the selected components for its optimal functionality. She collaborated with Ankur during the component selection process, contributing to the evaluation and choice of key elements needed for the drone's successful implementation.

Jaisel has experience designing rockets for the Penn State Rocketry Club, which requires blending together many different subsystems. Because of this, he has had a lot of experience with implementing electronics. He also has CAD experience. Because of this, Jay is the systems expert and also helps Mark with the CAD.

6.2. Capability Matrix

To determine our individual strengths as a group we created a table representing Figure 51

	Skill Sets					
Members	**Programming (Arduino)**	**Embedded Systems**	**Structural Design**	**Additive Manufacturing**	**Piloting Capability**	**Structual Analysis**
Mark Yazemboski						
Achyutan Srinivasan						
Ankur Ghorai						
Trusha Patel						
Jaisel Singh						

Green = proficient
Yellow = little experience/ not as confident
Red = no experience

Figure 51: Chart depicting our individual strengths

From this chart we can see that our weakest factor is our piloting capability where only two of our members having little experience in piloting drones. This should hopefully not be an issue as we can develop our piloting skills further as we get our certifications to fly on campus. We are also weak as a team in our programming capability but since there is at least one member who is proficient in this it is not as big of an issue. We expect to reach out and seek help from Professor Johnson and work together as a team to tackle any issues that we might encounter with the programming as we move further in the development process of our drone. In terms of piloting, we also expect that during our testing process as well as simulation testing, we should be able to develop adequate skills over the course of the spring semester. In contrast we are all comfortable with structural design and the other skills have at least 2 proficient members.

6.3. Development Schedule

In order to meet the time constraints our group considered the time left in the year and created a Gantt chart. This gave us a semi-mailable timeline and checklist to get through the year. Our original Gantt chart will be provided below in Figure 52.

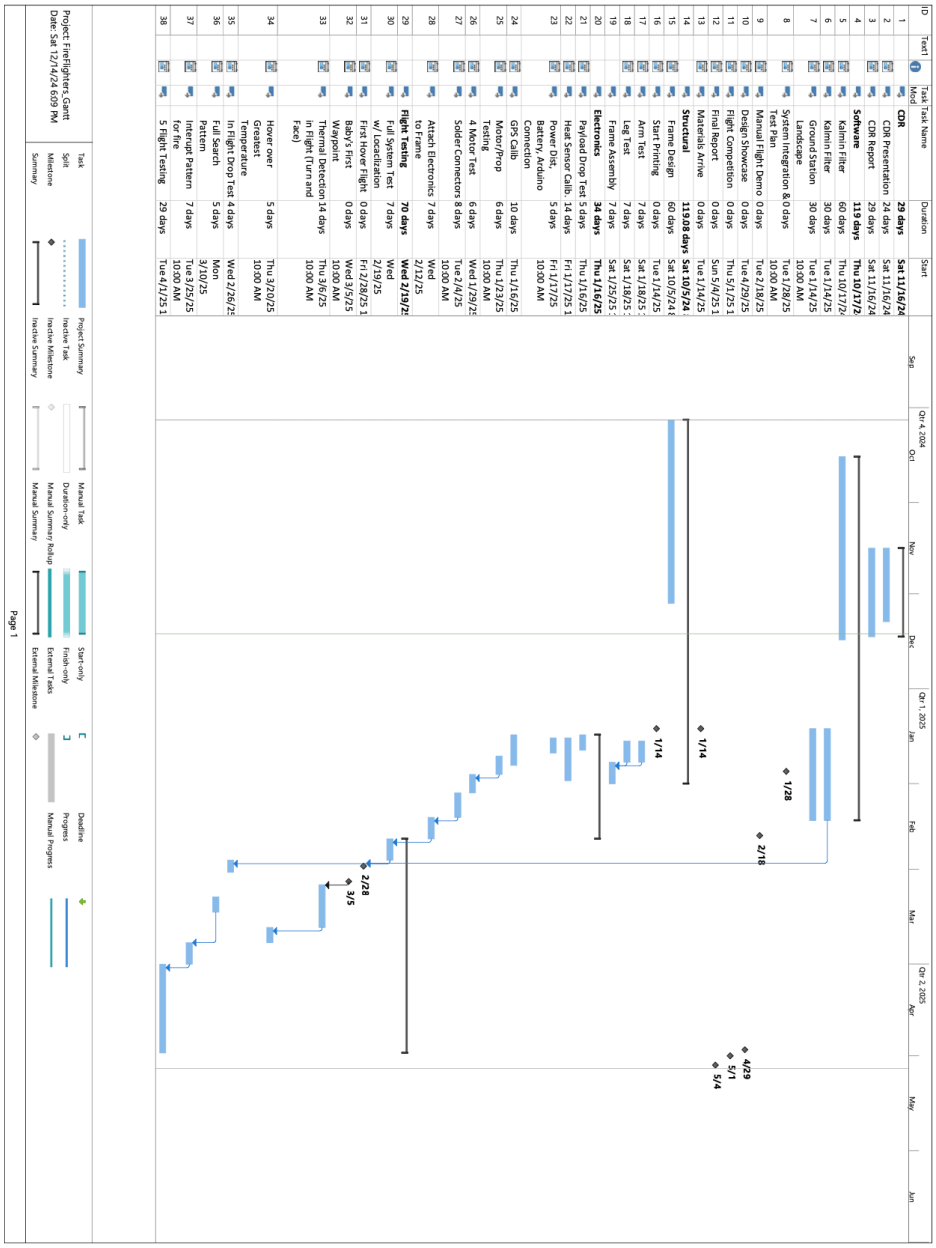

Figure 52: Image of Our Original Gantt Chart

This chart changed a lot as we met and responded to various challenges and delays encountered in development. Our final Gant chart will be provided below. We severally underestimated how long gain tuning would take us. It was our first time doing anything like this and we made many mistakes and un-optimal decisions along the way. If we were to tackle this project again we could probably streamline much of this work.

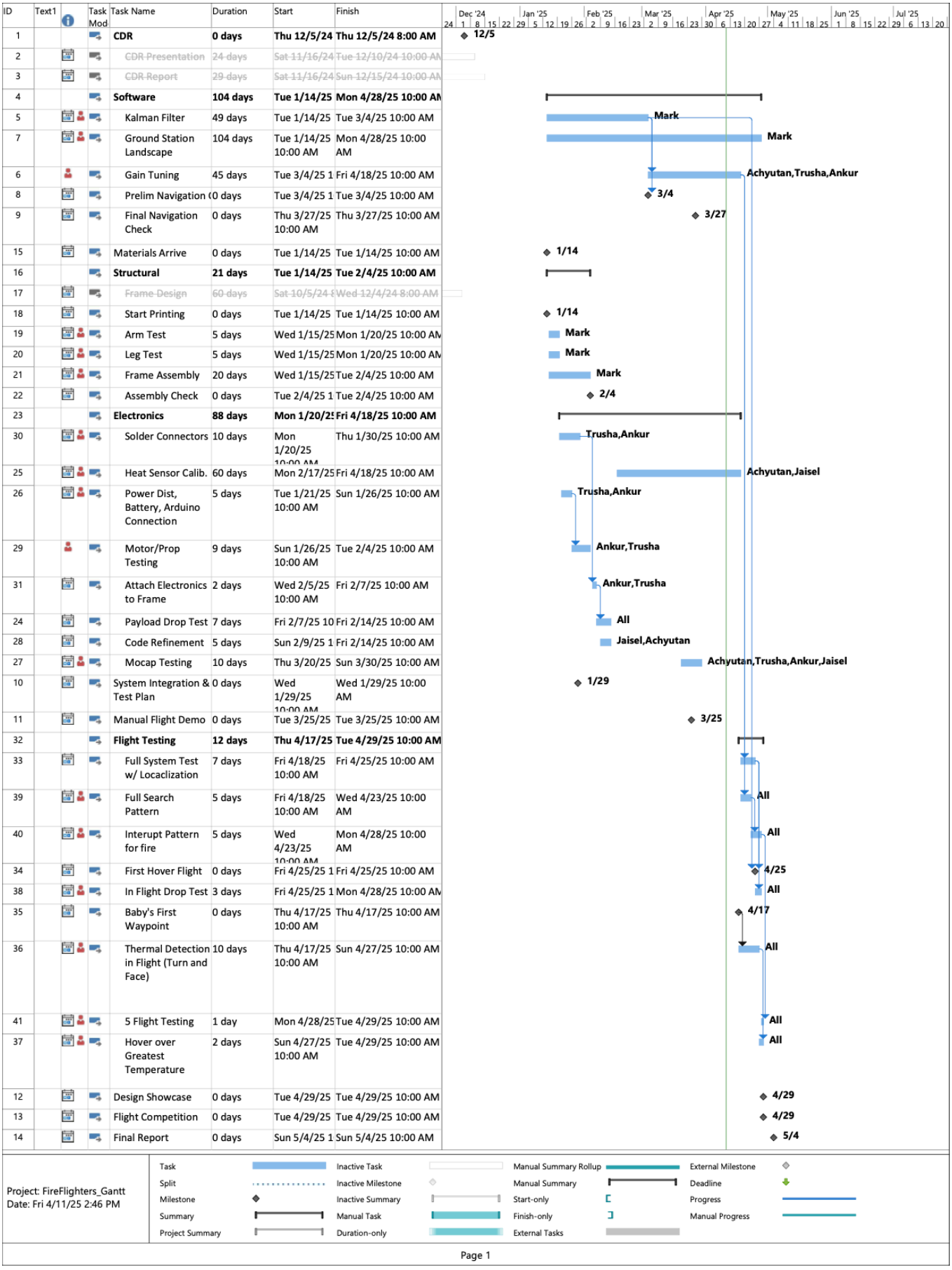

Figure 53: Image of Our Final Gantt Chart

7. Cost Breakdown

The total budged of our drone can be seen in Figure 54. The biggest highlight from this project, is the fact that the drones frame is only $3.38. With the entire drone only coming out to $226.95. A link to the full bill of materials will be provided ***here***.

	Base Drone	Extra	Replacements
Propellers	$ 4.07	$ 15.33	$ 4.07
Motors	$ 75.96	$ 18.99	$ 18.99
ESC	$ 83.96	$ 20.99	-
Battery	$ 25.99	$ 25.99	-
Hardware	$ 36.97	-	-
Frame	$ 3.38	-	$ 10.14
Totals	$ 226.95	$ 81.30	$ 23.06

Figure 54: Image of our cost breakdown

The frame was designed to be fully 3D printed, and this worked out very well. An image of the full frme being printed on a Prusa XL can be seen in Figure 55. The full print time is only 13 hours, and the cost is only $3.38 at a cost of $18 per kg.

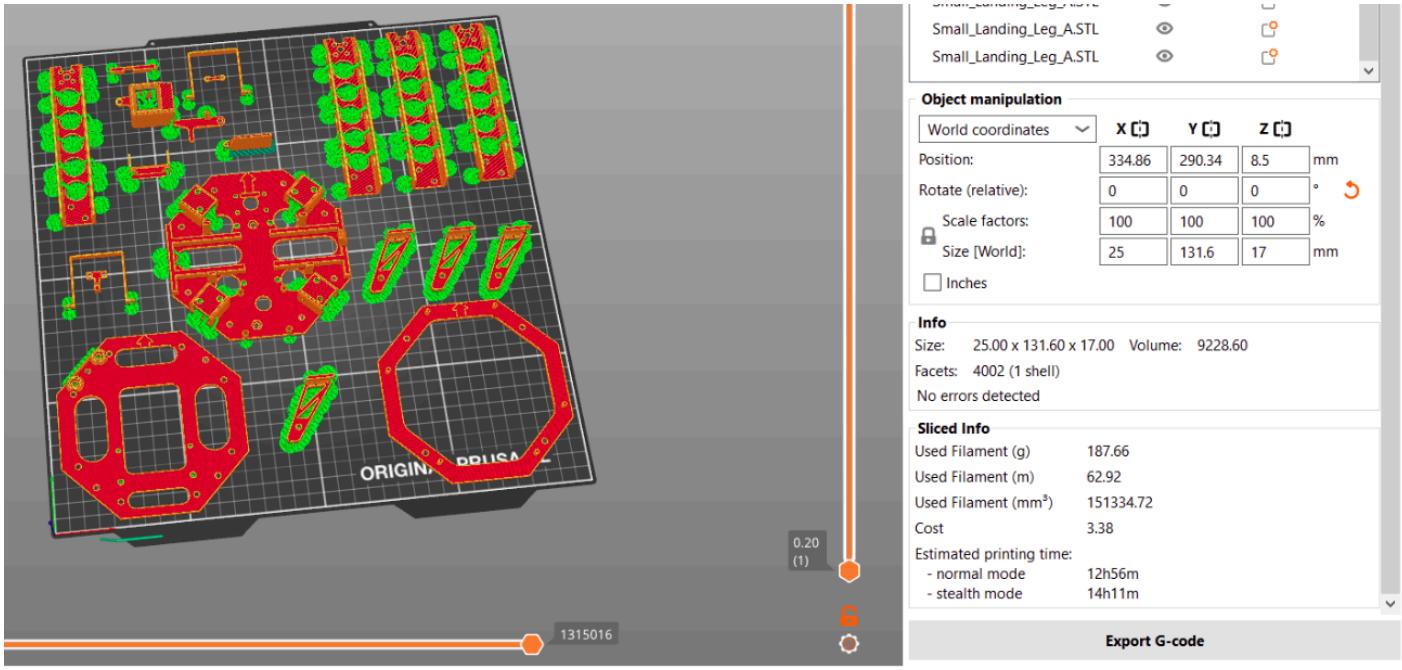

Figure 55: Image of the entire drone in the Prusa slicer software.

8. Testing

8.1. Testing Protocol / Plan

As decided by the team each subsystem component was tested separately before integrating with the drone. The propulsion system (ie. motors, ESC, and propellers) were each tested for proper wiring and directionality. The thermal sensor was tested and calibrated for the resistor pad. The Kalman filter and state estimation would be tested through hand carries. Data Link would be tested by sending and receiving data from the drone. As each subsystem is tested and the code runs through the PACE format, the code would be added to the onboard or grounds station code.

To perform proper flight and to tune the controller, a testing stand was implemented. The drone would be attached to a horizontal wire. The attachment point would be located on the same plane as the center of mass to mimic the dynamic flight stability of the drone. The mount is shown in Figure 56.

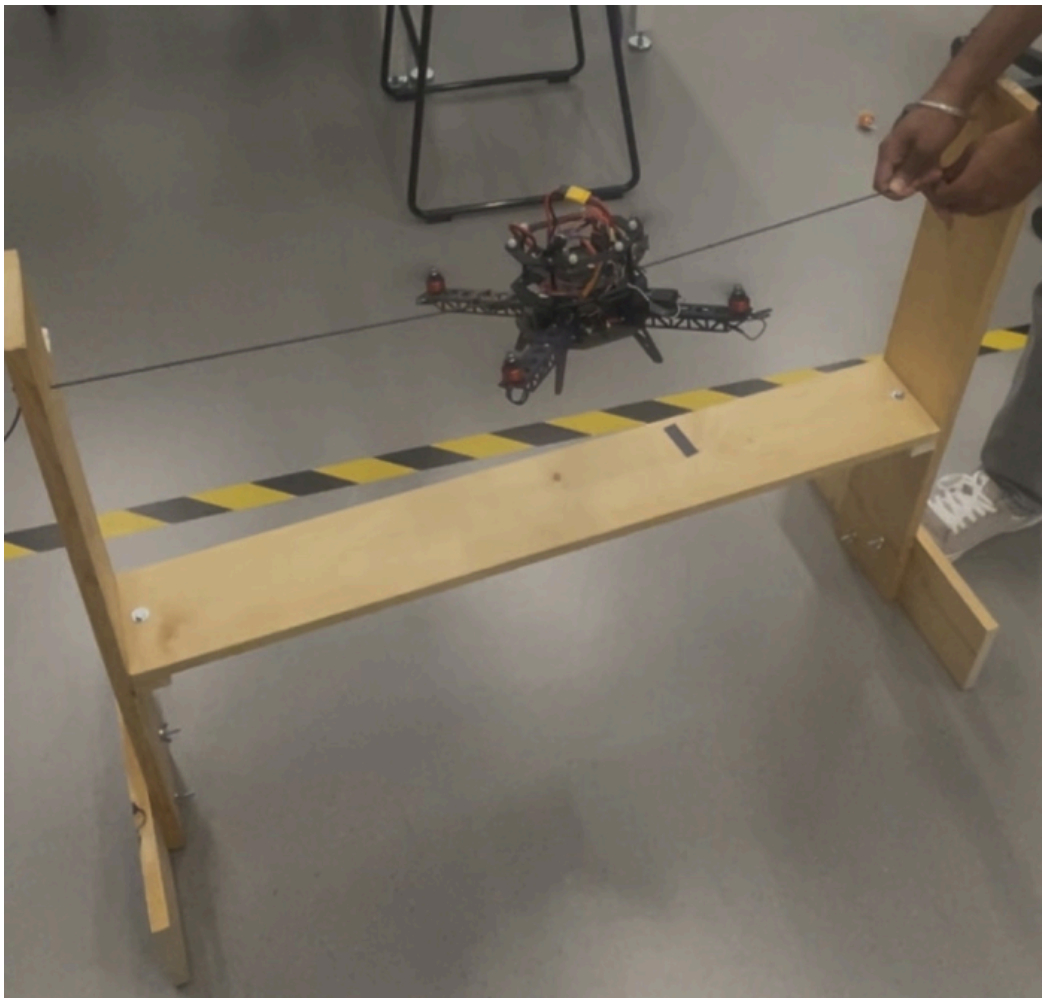

Figure 56: Drone on the testing rig.

Initially the gains for the controller will be set to zero. Starting with the proportional gains, the team would adjust the gains until the drone has oscillatory behavior that does not grow when perturbed. This process would be repeated with the derivative and integration terms. Once all of the terms are set, the team will lightly adjust the values to improve responsiveness to controller inputs.

8.2. Final Test Flight Results

The culmination of our development efforts took place during our final testing day, where we achieved significant milestones despite facing several challenges. Although we were only with an angle controller implementation, due to the limitations we uncovered about the limitations associated with the Arduino's computing and technical capability, as a team we successfully conducted comprehensive hand-carry tests that validated our payload delivery mechanism and navigation system. During these tests, team members physically carried the drone while it actively guided them to the fire zone using its waypoint system, effectively demonstrating the functionality of our navigation code. Our ground control station performed admirably, displaying real-time data including current position and orientation, waypoint tracking, and payload status monitoring alongside the PACE simulation for reference. Most notably, in our final flight attempt, the drone achieved stable flight—taking off, maintaining a controlled hover for approximately five seconds, and then being safely landed by Mark. While not perfect, this achievement represented a significant milestone, validating our core systems' functionality and providing valuable insights for future improvements. The successful hover demonstrated that our component selection, weight distribution, and basic flight controls were fundamentally sound, giving the team confidence in our overall design approach.

9. Conclusion and Lessons Learned

Overall, the team was successful in completing the competition albeit due to several constraints being made to the final mission allowing us to complete the autonomy and flight part separately. The final drone was both cheap and lightweight relative to the other competing teams.

The structure of the drone was sound and modular however the choice to use carbon fiber “reinforced” PLA as the 3D printing material was not the best choice as PLA is one of the more brittle filaments. The combination of PLA and carbon fiber was not a great choice as upon landing too hard, the arms and legs would break. In addition, the wire pathing through the baseplate caused the team to resolder the ESCs when an baseplate breaks. Should the team have more time, the use of ABS plastic for certain components like the legs and arms may lead to less breakages. Furthermore, a redesign of the baseplate’s joints would have prevented the need to resolder connections with every crash.

The overall choices and design of the drone’s subsystems were satisfactory. The propellers, motors, and batteries performed similarly to the predicted performance. The modularity of the structure allowed us to implement them well. The estimated weight of the drone was very close to the predicted value and the team was able to isolate components that were redundant to save 55g more from the total flight weight. However, the normal wire connections to the Arduino were loose and caused fluctuations in the reading from the PWM reading from the receiver. This was a major pain point for the team and caused many crashes. The source was only located by the team very late into the semester and fixed by installing proper connections to the board. In the future, this should be a checklist for any design, especially should it have many vibrations.

The team’s successful flight was the product of a great amount of testing and tuning the gains of the drone. This could only be accomplished easily by testing them on a test stand; something that was overlooked in the design stage of the project and only implemented after multiple crashes and rebuilds. In any future projects that require precise and responsive controllers, a method to test the gains and parameters should be used in a risk-free environment. Had the team implemented this from the start, the team would have saved many days’ worth of time.

On the software side, the team was able to implement the autonomous requirement for the competition via a hand carry. The limitation of the Arduino’s processing speed, and inability to multitieredd prevented the team from being able to fly the drone with the autonomy software running, and instead required the drone to turn off all of the autonomy software to be barely fast enough to manually fly.

The team work within the team was phenomenal. The ability to split up tasks that could be parallelized and the diverse skill set of the team allowed for tackling the various issues that arose during the project. Communication was also a key factor to the success of the project as it kept the work flow up as other members can pick up the tasks.

Overall, the team was satisfied with the conclusion and results of the competition and project. Many of the systems that were designed came together and worked by the end of the project.

10. Appendix

10.1. Engineering Ethics & Social Impact Implications of UAV Design

The advancement of unmanned aerial vehicles (UAVs) for applications like wildfire suppression presents transformative opportunities - yet it also raises critical ethical and social questions. As engineers, we must weigh not only technical feasibility but also the broader consequences, both positive and negative, of autonomous systems on human lives, employment, environmental sustainability, and safety. How do we balance innovation with responsibility when designing UAVs that could displace workers, malfunction in critical scenarios, or contribute to e-waste? This analysis explores the ethical challenges and societal trade-offs inherent in UAV development, emphasizing the need for proactive mitigation strategies, stakeholder engagement, and a human-centered approach to ensure these technologies serve the greater good without unintended harm.

Below, we analyze this system through a futures wheel framework, mapping both first-order (direct) and second-order (indirect) impacts of widespread implementation. This diagram highlights not only the anticipated benefits—such as improved fire response times and reduced risk to human firefighters—but also potential negative outcomes, including job displacement, environmental concerns, and operational risks. [2]

For each identified challenge, we assess its likelihood and propose targeted mitigation strategies, ensuring our design addresses ethical obligations while maintaining operational effectiveness. By confronting these consequences proactively, we aim to develop UAV technology that aligns with societal values, environmental sustainability, and long-term resilience.[3]

10.2. Futures Wheel

Figure 57: Futures Wheel

10.3. FMECA-style table

Consequence	Likelihood	Mitigation(s)
Loss of jobs for firefighters	High	1) Train traditional firefighters to oversee, maintain, and deploy drone fleets. 2) Shift roles from frontline firefighting to AI-assisted fire management.
Potential for accidents, like drone malfunctions causing unintended fires or crashes	Medium	1) Dual motors, backup batteries, and parachute recovery systems for crash prevention. 2) Maintain human operators to intervene during malfunctions 3) Automatic shutdown if critical systems (e.g., lidar, GPS) fail. 4) Pre-programmed emergency landing zones (away from flammable areas).
Manufacturing and disposal of drones contribute to e-waste and resource depletion	Low	1) Use biodegradable composites (e.g., mycelium-based, flax fiber) for non-critical parts. 2) Prioritize recycled metals (aluminum, titanium) and low-impact plastics. 3) Partner with e-waste recyclers to recover materials from old electronics. 4) Optimize drone structures (e.g., hollow frames, honeycomb designs) to minimize raw material use

Figure 58: FMECA-Style table

We would prioritize addressing job displacement for firefighters, drone malfunctions, and e-waste in future design iterations due to their ethical, safety, and environmental implications. These mitigations —such as human-in-the-loop controls (to oversee, maintain, and deploy drone fleets), fail-safe mechanisms, modular eco-design, and renewable energy integration—would increase upfront costs and complexity but ensure long-term sustainability and social acceptance. Lower-priority issues like geopolitical competition or noise pollution would be deferred, as they fall outside immediate technical scope or offer acceptable trade-offs. Balancing competing requirements (e.g., cost vs. sustainability, autonomy vs. jobs) would involve modular upgrades, ensuring core mission effectiveness while adapting to evolving needs. Ultimately, this approach would align with mission goals, user expectations, and funding constraints without compromising safety or ethical responsibility.

11. References

[1] Oscar Liang, “What to Consider in FPV Drone Frames.” [Online]. Available: https://oscarliang.com/fpv-drone-frames/

[2] FAA, “Drones and Wildfires are a Toxic Mix.” [Online]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.faa.gov/sites/faa.gov/files/uas/resources/community_engagement/FAA_drones_wildfires_toolkit.pdf

[3] M. G. Elena Ausonio Patrizia Bagnerini, “Drone Swarms in Fire Suppression Activities A Conceptual Framework.” [Online]. Available: https://www.mdpi.com/2504-446X/5/1/17