

DOMINANCE: Drone Mine Obstacle Avoidance

ECE Senior Design 2 Spring 2020
Group 18



Caleb Jones (CpE)
Hamza Siddiqui (CpE)
Rishi Jain (EE)
Ryan Lucas (EE)

Sponsored By Lockheed Martin

Table of Contents

1.0 Executive Summary (RL)	1
2.0 Project Description	2
2.1 Project Goals and Objectives	2
2.2 Project Motivations (RL)	6
2.3 Project Operation (CJ)	6
2.3.1 Hardware Setup (CJ)	7
2.3.2 Ground Control Station Setup (HS)	7
2.3.3 Drone Power-up Sequence (CJ)	7
2.3.4 Autonomous Mode Operation (CJ)	7
2.3.5 Manual Mode Operation (CJ)	8
2.3.6 Shutdown (CJ)	8
2.3.7 Diagnosing and Resolving Operational Issues (RJ)	8
2.4 Requirements Specification (CJ)	9
2.5 House of Quality (RJ)	11
3.0 Project Research	13
3.1 Similar Projects	13
3.1.1 Nvidia Trail Drone (CJ)	13
3.1.2 MicLoc Sound Detection (HS)	15
3.2 Relevant Technologies	16
3.2.1 Image Recognition Methods (HS)	16
3.2.1.1 Image Classification (HS)	17
3.2.1.2 Object Localization (HS)	17
3.2.1.3 Histogram Back Projection (HS)	17
3.2.1.4 MeanShift Algorithm (HS)	18
3.2.1.5 CAMshift Algorithm (HS)	19
3.2.1.6 Grabcut Algorithm (HS)	20
3.2.1.7 Convolutional Neural Network (HS)	20
3.2.1.8 Linear Regression (HS)	21
3.2.1.9 Object Detection (HS)	21
3.2.1.10 Instance Segmentation (HS)	22
3.2.2 Computer Vision Libraries (HS)	22
3.2.3 Object Detection Models (CJ)	24
3.2.3.1 CenterNet	24

3.2.3.2	RetinaNet	24
3.2.3.3	Single Shot Detector (SSD)	24
3.2.3.4	Model Decision	25
3.2.3.5	Model Training	25
3.2.4	Sound Detection Methods (HS)	26
3.2.4.1	Microphone Triangulation	26
3.2.4.2	Sound and Noise Filtering (HS)	27
3.2.4.3	Noise Filtering Hardware Approach (HS)	27
3.2.4.4	Noise Filtering Software Approach (HS)	28
3.2.4	Flight Controller Software (RJ)	29
3.3	Parts Selection	30
3.3.1	Computer (CJ)	30
3.3.1.1	Computer Requirements	30
3.3.1.2	Raspberry Pi 4 Model B	30
3.3.1.3	Jetson Nano Development Kit	30
3.3.1.4	Comparison & Selection	31
3.3.2	Flight Controller (RJ)	32
3.3.2.1	Flight Controller Requirements	32
3.3.2.2	Parts Comparison and Selection	33
3.3.3	Camera (CJ)	34
3.3.3.1	Camera Requirements	34
3.3.3.2	Single Camera	34
3.3.3.3	Double Cameras (Stereo Vision)	35
3.3.3.4	Depth Cameras	35
3.3.3.5	Method Comparison and Selection	36
3.3.3.6	Part Comparison and Selection	37
3.3.4	Microphones (HS)	38
3.3.4.1	Seeed's ReSpeaker Mic Array v2.0	38
3.3.3.2	Seeed's ReSpeaker 2-Mics Pi	39
3.3.3.3	Seeed's ReSpeaker 4-Mic	39
3.3.3.4	Part Comparison and Selection	39
3.3.5	Batteries (RL)	40
3.3.6	Motors (RL)	41
3.3.7	Wireless Communication (HS)	42

3.3.7.1 Router	42
3.3.7.2 Wi-Fi Transmitter	42
3.3.7.3 Telemetry Radio	42
3.3.8 Optical Flow Camera (CJ)	43
3.3.8.1 Purpose	43
3.3.8.2 PX4FLOW	43
3.3.8.3 Hereflow Optical Flow/Lidar	44
3.3.9 Time of Flight Distance Sensors	44
3.3.9.1 Purpose (CJ)	44
3.3.9.2 HC-SR04 Ultrasonic Range Sensor (HS, CJ)	45
3.3.9.3 HRLV-MaxSonar EZ4 on the PX4FLOW (CJ)	45
3.3.9.4 TeraRanger One (CJ)	45
3.3.9.5 TF Mini LiDAR (HS, CJ)	45
3.3.9.6 Part Comparison and Selection (CJ)	46
3.3.10 Drone Starter Kit (RJ)	47
3.3.11 Part Selection Summary	50
3.4 Possible Architectures and Related Diagrams (RL,RJ)	51
3.4.1 Emergency Stop (RL)	52
3.4.2 Data Flow (RL)	53
3.4.3 Obstacle Recognition and Maneuvering Sequence (RL)	53
3.4.4 Power Distribution (RL)	54
3.4.5 Acoustic Waypoint (RL)	55
3.4.6 Component Connection (RL)	56
4.0 Related Standards, Regulations, & Realistic Design Constraints (RL)	57
4.1 Standards (RL)	57
4.1.1 Dimensional Standards (RL)	57
4.1.2 Coding/Programming Standards (RL and HS)	57
4.1.3 Video resolution standard (RL)	59
4.1.4 IEEE Standards (RL)	59
4.1.5 UL 3030 – Standard for Safer Flights (RL)	59
4.1.6 IPC PCB Standards (RL)	60
4.1.7 IEC60950 (Relevant Power Supply Standards) (HS)	62
4.2 Drone Regulations (CJ)	63
4.3 Realistic Design Constraints (RL)	64

4.3.1 Economic Constraints (RL and RJ)	64
4.3.2 Time Constraints (RL and RJ)	65
4.3.3 Environmental, Social, and Political constraints (RL and RJ)	66
4.3.4 Ethical, Health, and Safety Constraints (RL and RJ)	67
4.3.5 Manufacturability and Sustainability Constraints (RL and RJ)	69
5.0 Project Hardware Design Details	70
5.1 Initial Design Architecture (HS)	70
5.1.1 Power System (RJ)	71
5.1.2 Wireless Communication System	74
5.1.3 Interfacing with Sensors (CJ)	75
5.1.3.1 USB	75
5.1.3.2 UART	75
5.1.3.3 I2C	75
5.1.4 Microcontroller Architecture (RJ)	75
5.1.5 Nano-PixHawk Interface (CJ)	77
5.2 Subsystems (RL)	78
6.0 Project Software Design Details	82
6.1 Drone Software	82
6.1.1 Drone Software Overview (CJ)	82
6.1.2 Drone Computer OS and Framework (CJ)	83
6.1.2.1 Drone Computer OS	83
6.1.2.2 Robotic Operating System (ROS) Framework	83
6.1.3 Drone Software Nodes (CJ)	84
6.1.3.1 Camera Node	84
6.1.3.2 RetinaNet	84
6.1.3.3 Distance Estimation	85
6.1.3.3.1 Node Design & Operation	85
6.1.3.4 Height Sensor Node	86
6.1.3.5 Microphone Node	86
6.1.3.6 Drone Controller	86
6.1.3.6.1 Autonomous Control Modes	86
6.1.3.6.2 Manual Control Mode	90
6.1.3.7 Simultaneous Localization and Mapping	91
6.1.3.8 MAVROS	91

6.2	Ground Control Station Software (HS)	91
6.3	Software: Interference and Failure Modes (RJ)	93
6.3.1	Unidentified Object Mode	93
6.3.2	Flight Recovery Mode	94
6.3.3	Mine Avoidance Mode	94
6.3.4	Object Proximity Mode	95
6.4	Microcontroller Software (RJ)	95
7.0	Project Construction and Coding	96
7.1	Hardware Construction	96
7.1.1	Needed Equipment and Building Space (RJ)	96
7.1.2	Hardware Calibration and Configuration	96
7.1.2.1	Electronic Speed Controllers (ESCs) (RJ)	96
7.1.2.2	Depth Camera Calibration (CJ)	97
7.1.2.3	PX4FLOW Calibration (HS)	97
7.1.2.4	PixHawk Calibration and Configuration (RJ)	97
7.1.3	Phase I (RJ)	97
7.1.4	Phase II (RJ)	98
7.1.5	Phase III (RJ)	99
7.2	Software Development	99
7.2.1	Language Choice (CJ)	99
7.3	Development Methodology	100
8.0	Project Prototype Testing Plan (RL)	101
8.1	Hardware Testing (RL)	101
8.1.1	Hardware Test Environment	102
8.2	Hardware Specific Testing (RL)	105
8.2.1	Lithium Polymer Battery Testing (RL)	105
8.2.2	Printed Circuit Board (RL and RJ)	106
8.2.3	Jetson Nano and component pairing (RL)	106
8.2.4	Flight controller and electronic speed controller (ESC) (RL)	108
8.3	Drone Software Testing (CJ)	108
8.3.1	Unit Testing Nodes	108
8.3.2	Software-in-the-Loop Testing	109
8.3.3	Input Sensor Interface Testing	109
8.3.4	Hardware-In-the-Loop Testing	109

8.4 Wireless Testing (HS)	110
8.5 Ground Station Manual Control Testing (HS)	111
8.6 System Connectivity Prototype (HS)	112
8.7 Sound Triangulation Prototypes (HS)	113
8.8 Sound Channel Prototype (HS)	114
8.9 Reliability and Performance Testing (RJ)	115
8.10 Adversarial Mine Avoidance Hardware (HS)	115
8.10.1 Propeller Shrouds	116
8.10.2 Safety Mesh	117
9.0 Administrative Content (RJ)	118
9.1 Deliverables	118
9.2 Milestone Discussion	119
9.3 Initial Budget and Finance Discussion	121
9.4 Advisors, Meetings, and Communications	123
9.5 Parts Acquisition	123
10.0 Project Results, Major Changes, and Future Considerations	124
10.1 Hardware	124
10.1.1 Drone Frame	124
10.1.2 Optical Flow/Height Sensor	124
10.1.3 Microcontroller/PCB	124
10.1.4 Power Distribution	125
10.1.5 Electronic Speed Controllers	125
10.2 Software	125
10.3 Overall Drone Functionality	126
10.3 Final Cost	126
11.0 Summary	128
12.0 Appendices	A
12.1 Bibliography	A
12.2 Copyright Permissions	C

1.0 Executive Summary (RL)

Drones have been adapted to serve many purposes and functions. Law enforcement and other government agencies use drones for a variety of purposes such as surveillance, traffic monitoring, and search and rescue. However, drones are used in all types of industries. Drones have also been made into a consumer hobby item and even toys for children, where the drones are equipped with a camera and can be utilized to create personal videos, or just simply hobby flying. [1]

With many different advancements in drone technology, autonomous drones have become increasingly popular for programmed scanning and surveying of areas for various purposes, especially due to lower costs and ease of use. An autonomous drone may be used for scanning forests, exploring regions of caves that may be difficult for explorers to reach, and other similar scanning purposes. It becomes particularly important if environmental and ecological conditions tend to interfere with radio, Bluetooth, WiFi, and other forms of remote signals. Using such drones also help reduce safety risks and lower costs in operations. [25]

This project involved designing, building, and demonstration of an aerial vehicle that will be fully autonomous. The drone is a quadcopter and is capable of detecting obstacles and identify what the obstacle is (ring, single pylon, or double pylon). The drone was designed to perform certain pre-programmed tasks after identifying the obstacles, and use the layout of the obstacle's locations to route a path as it will not be equipped with any form of GPS navigation system. It was also planned to be able to withstand or avoid any ground-bases obstacles and interferences created by the "mine" team as it navigates through the obstacle course. As such, the drone will also need to be sturdy to some extent.

The size of the drone was limited to certain dimensions based on the customer's requirements. The drone is able to detect acoustic sound waves that can mark waypoints for it to land for a brief moment and takeoff again. It is also equipped with an emergency shutdown protocol which will turn off power and cause the drone to land in the event of a safety concern.

The drone communicates with a remote computer via WiFi and radio telemetry, and send data such as the type of object detected and its respective distance and level of confidence, route data, and other relevant data as required by the client. It able be able to operated for a flight duration of at least 7.7 minutes. The drone is able to keep a log of the entire flight path including the starting point, and be able to return to its starting point once all required obstacles have been detected and cleared. Two runs were planned to be conducted; the first run without the mines set by the "mine" team, and the second run with the mines set somewhere amongst the obstacles or the flight path.

The success of this project originally depended on the drone's ability to successfully detect and maneuver around all obstacles as instructed through its programming, and to accumulate points. The team that completes the course with the highest accumulated points was slated to the competition. However, with the lockdown and the "stay at home" order due to the COVID-19 pandemic in effect, the team was required to achieve autonomous flight and successfully maneuver through a ring and a pylon, and then autonomously land. The drone was no longer required to be able to autonomously evade a "mine" attack, land near a waypoint, or successfully perform an E-Stop procedure.

2.0 Project Description

The following sections under the project description provide details of the general project concepts, the project's goals and objectives, the different uses of autonomous drones, and the project requirement specifications. This section also includes a House Of Quality figure that weighs the engineering tradeoffs between different design decisions.

2.1 Project Goals and Objectives

The team was tasked to create an autonomous drone with the goal of being fully self-driven. A total of three teams were to compete to design and create their own version of an autonomous drone which will be programmed to detect various obstacles placed throughout the course at Lockheed Martin's drone testing facility.

Points were to be disbursed based on the successful maneuvering of each obstacle, along with multipliers for each consecutive obstacle detected and maneuvered around as per required by the clients/customers. Judges were to be appointed to observe and make decisions on points disbursement based on the performance and accuracy of proper execution of each obstacle avoidance.

An unknown number of acoustic waypoints were to be located at random locations throughout the course where the drone will be able to detect the acoustic signal and identify the waypoints to land and takeoff again and continue with the course. There was also a team (the mine team) whose sole goal is to create mines to take down the drones, so the drones were to be programmed to detect approaching dangers and make attempts to avoid the mines.

Three different types of obstacles were planned to be placed throughout the course which are rings, single pylons, and double pylons. The obstacle course design and layout, along with the number of each type of obstacles and waypoints, was to remain unknown, and will only be revealed on the day of the competition. Our primary motivation during the competition is to maximize the points earned. The point breakdown is listed below in Table 1.

Table 1: Obstacle Navigation Values

Obstacle	Points
Ring	1
Single Pylon	2
Double Pylon	3
Acoustic Waypoint	4

This project is a sponsored project, as such, all designs and programming codes for the drone were to be executed with the requirements of the customers in mind. Two advisors from the sponsoring company has been assigned to overlook and provide guidance on the progress of the project. This project was an interdisciplinary project with each team typically comprising of two electrical engineering, two computer engineering, two mechanical, and two aerospace engineering students. The purpose of this senior design project was to challenge the students to utilize all the knowledge and experience gained from classes taken throughout the years spent in college and applying it to a practical scenario.

The main focus for the electrical and computer engineering (ECE) students of the team was to design and implement a system that will enable the drone to detect an obstacle, identify the type of obstacle detected, access the maneuvering sequence for the particular type of obstacle detected, and execute the programmed set of instructions to continue and complete the course. The team also designed and create a printed circuit board (PCB) to to manage laterally mounted ultrasonic range sensors.

The central processing unit (CPU) was to manage horizontal distance sensor data, propellers, flight controller, cameras, microphones, and sensors. The ECE team was to incorporate the appropriate algorithm to enable the various components to communicate with each other. We also sent the data collected from the cameras, microphones, sensors to the remote computer via WiFi to be recorded and viewed. Our team was also responsible for implementing a flight path log to enable the drone to return to the starting point once it has completed the course.

The drone will be quadcopter with four motorized propellers. We built our own frame for our final project, and though we planned on utilizing a kit for prototyping, our team developed in houses prototypes as well. This decision was unanimous since the option was available, and would indefinitely same time and effort, especially due to time and funding constraints. We considered creating a

mesh to protect the drone from mine attacks, but dropped this idea this plan due to the time constraints and reduction of requirements.

A camera with depth perception was used to determine the distance of any detected obstacles. Computer vision software was utilized to detect and navigate to the obstacles. Moreover, microphones was to be used to detect acoustic waypoints and communicated with the onboard CPU to instruct the drone to land and takeoff again.

Our team attempted to create a small, portable drone that can autonomously track and detect obstacles, navigate towards them, and maneuver around them. After maneuvering, the drone was to search for the next obstacle within its field of vision and repeat the same process until it reaches the end of the obstacle course. All navigation and maneuvering must be done without the use of GPS navigation. The drone must also be able to detect acoustic waypoints and land in front of them before continuing on. Additionally, the drone must have the ability to identify, approach, and navigate around the obstacles regardless of their orientation or location. The obstacles in the course include pylons and rings as shown in Figure 1.

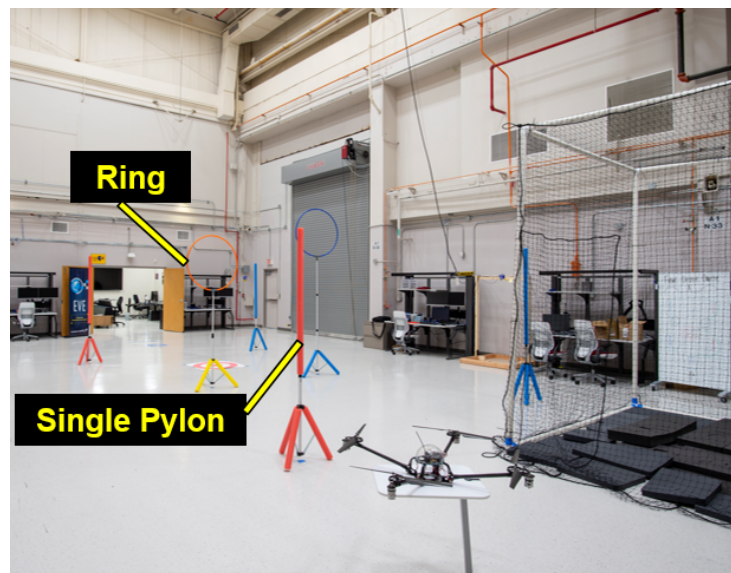


Figure 1: Lockheed Martin IRIS Drone Lab with Pylons and Rings, image referenced from Lockheed Martin DOMINANCE Kick-Off slides

The drone has several operational modes available for use. Auto Navigation will be used to detect, approach, and autonomously navigate around objects. Auto-manuever will be used to perform specific autonomous actions around identified objects. Manual mode will be used in the event that the previous two modes fail to work correctly during demonstration. E-Stop will be used in emergency situations where in which the drone fails to operate correctly and cannot be controlled manually. Lastly, Take-off/Land mode will be used to start and end the drone's mission. In order to ensure that the drone can successfully perform its

desired mission, it was to have the power to sustain flight and its onboard components for at least 10 minutes.

Several key components to implementing these features include a camera to detect objects, a sonar sensor to detect proximity, and a microphone to detect acoustic signals. The video feed from the drone was to be broadcasted back to the computer where a red 'X' will overlay the identified object as seen in Figure 2. In addition to identifying the object, the drone's identification confidence level, distance to the object, and the drone's estimated arrival time to an object will be displayed for the end user to see. The drone's communication with the host computer will allow the user to test and control the drone manually, if needed. This channel of communication will be done wirelessly, either through WiFi or radio telemetry.



Figure 2: Symbology Example Metadata box and Red X for Target Identification, image referenced from Lockheed Martin DOMINANCE Kick-Off slides

In addition to navigating obstacles, one other team's objective to create a set of adversarial "mines" to disrupt our drone's ability to navigate the course. These mines were designed to track our drone's position and attempt to knock our drone off course using projectiles. In addition to launching projectiles, the mines may use other techniques to try to impede the ability of the drones to navigate. These mines were to be placed throughout the course in unannounced locations, and our objective is to ensure our drone is sturdy and adaptable enough to avoid or withstand any attacks from these mines, allowing us to complete the obstacle course. However due to the COVID19 pandemic, the competition was dropped.

Our drone was to be assessed on its ability to navigate to and around obstacles in the course. Moreover, the performance of our drone was to be compared to the drone's of other teams with the same objectives during two rounds. The first round will only involve navigating around identified obstacles as required, while the second round will include the adversarial mines. Each team was to receive points for being able to successfully complete the course during the two rounds. An inoperative video data feed or uncontrolled UAVs was to result in forfeiture of the round. These requirements were to be accomplished initially, before the COVID-19 pandemic occurrence. With the new requirements for the teams, the

project will now be assessed on the team's ability to achieve autonomous flight, successfully navigate a ring and a pylon, and then autonomously land. The competition for the project was cancelled, and point accumulation will no longer be necessary.

2.2 Project Motivations (RL)

The motivation for this project entails several applications, each with its unique beneficial properties. From a military perspective, an autonomous drone may be used for reconnaissance or area scouting. This property can be particularly useful in saving the lives of troops. Autonomous drones can also be useful in exploration applications such as surveying a forest region, or a cave or tunnel system. This property is particularly useful as it is very cost effective, at the same time, reduces the risk of safety concerns in the form of unforeseen injury prevention (in the case of unexplored territories).

Aside from the practical applications of an autonomous drone, the project in general involved working with people with diverse backgrounds, each bringing their own set of skills and expertise to make the project successful. This type of project environment is quite common in the real world, and being able to work with a team becomes crucial and often challenging at times. In many ways, this project acts as a form of preparation of what to expect when entering a career path. Growth is an essential factor in any career path, and is particularly important in the case of an engineer. This project emphasized the importance of good communication and collaboration between colleagues to reach a common goal. If a team is not able to communicate between members, any project they may be working on will most likely not succeed.

This project required the knowledge and skills of electrical engineering, computer engineering, mechanical engineering, aerospace engineering, and computer science students. However, in the case of our group, since the group lacks students in the computer science field, all coding and programming responsibilities will be handled by the computer engineering students. In other words, the desire to successfully complete the project and execute the expected results of the customers, the team was required to work in unity. The mechanical and aerospace engineering students worked on the frame of the drone, along with the motors and propellers, flight controller, electronic speed controller (ESC) of the drone. The electrical and computer engineering students worked on all the electrical and communication components, as well as the coding and programming algorithms that will be utilized for all of the onboard components to communicate and work with each other, and with the ground computer/laptop.

2.3 Project Operation (CJ)

This section details the operation of the drone from the user's perspective. The purpose is to help guide our design based on how we wish the users to operate

the drone. This section guides a potential user on how to set up, power-up, operate, and shutdown the drone. We also discuss what can be done during our project demonstration to overcome possible issues.

2.3.1 Hardware Setup (CJ)

The drone does not need a large amount of assembly before operation. The operator will only need to strap in the battery pack. The battery pack is detachable in order to make charging the battery pack easier and to allow for quick replacement of the battery for a fully charged one. Setting up the ground station would require connecting a laptop that is able to run our ground station software to the wifi access point that the drone produced. Additionally, in order to manually fly the drone, a radio controller was connected to the computer.

2.3.2 Ground Control Station Setup (HS)

A laptop is needed to operate the drone in manual control modes. In autonomous mode, the ground control station is used to initiate flight and display the camera feed from the drone.

The laptop will be setup with a router, and the router will need to be turned on and be in the range of the drone's wifi module (within 100 feet). An operator needs to SSH into the shell of the drone computer and run the necessary programs to start the autonomous flight. During the flight, the drone displays the output from the vision algorithm to the Graphical User Interface (GUI) of the operating system of the drone computer. The ground station then receives the the feed of the GUI of the drone computer and displays it.

2.3.3 Drone Power-up Sequence (CJ)

The drone powers up once the drone battery is connected to the rest of the drone components. The flight controller attempts to turn on and does pre-flight checks to ensure that the drone is ready to fly. If the drone passes all pre-flight checks, the light on the flight controller will change to a flashing blue. If the drone fails any pre-flight checks, the flight controller will display a different color, usually flashing yellow. At the same time, the drone computer then turns on and starts the wifi access point for the ground station to connect to.

2.3.4 Autonomous Mode Operation (CJ)

The drone's Autonomous Mode can be started by running the autonomous program from the ground control station. The drone will then carry out its autonomous actions and land. If an safety issue occurred, the drone can be sent an E-stop command from the ground station computer. This will force the drone to immediately stop and land.

If the drone ever loses communication with the ground station for more than 5 seconds, the drone will immediately attempt to land. This will prevent the drone from flying without being able to throw it into the E-Stop submode. If the drone battery life ever drops below 5%, the drone will attempt to land if currently navigating to an obstacle. This is to prevent the drone from falling out of the sky from losing power and causing damage to the drone.

2.3.5 Manual Mode Operation (CJ)

In Manual Mode, the drone is controlled via a radio controller. In order to arm the flight controller and enable flight, the left stick that controls altitude must be held to the down and right. Once the light on the flight controller turns from a flashing blue to a solid blue, then the drone is ready for flight. When no inputs are given by the radio controller, the flight controller causes the drone to hover in place if airborne and stay on the ground if it has not taken off.

Flight commands are sent to the flight controller via the radio controller and receiver. To fly the drone upwards, tilt the left stick up. To fly the drone downward, pull the left stick down. To turn the drone left, push the left stick left. To turn the drone right, push the right stick right. To move the drone forward, push the right stick forward. To move the drone backwards, pull the right stick backwards. To move the drone left, move the right stick left. To move the drone right, move the right stick right.

2.3.6 Shutdown (CJ)

Once the drone has landed and the motors are off, users should wait for the flight controller to switch from solid blue to flashing blue. This indicates that the drone is disarmed and safe to approach. A user can then go to the drone and unplug the battery. All onboard systems including the flight controller and the computer will immediately shut off.

2.3.7 Diagnosing and Resolving Operational Issues (RJ)

In the competition, we were permitted to repair the drone if necessary without losing any points. For safety reasons, we would only attempt to repair and reset the drone once it is on the ground. In addition to our manual ESTOP feature, the drone is also programmed to land automatically if it experiences a communication failure. Moreover, the drone will cease flying if it is unable to sustain an altitude of 1 foot for at least 10 seconds.

In the event that the drone is unexpectedly on the ground in the middle of the obstacle course, then we would know the drone experienced a communication, power, or hardware failure. We would first check whether the drone is connected to the ground control station. If not connected, we will attempt to reconnect communications via the laptop. If we are unable to reconnect, we will physically approach the drone and perform a hard reset. Once the drone is reconnected, we will verify the power level of the battery. The drone was to be programmed to

land before fully depleting the battery, allowing the drone to safely land. If the battery is low, we will swap the battery with the spare. However, if we find that the communication and power systems of the drone are working as expected, we would first verify whether or not the drone was damaged. The most susceptible components to damage are the propeller, which could crack when colliding with an object. If the propeller is damaged, it can easily be swapped with a replacement by unscrewing the old propeller. Before doing this, the user would unplug the battery to ensure that the propeller doesn't spin during the repair.

If we were to find that the drone is completely unresponsive, a hard reset will be performed by removing the battery and powering up the drone again. We would also reset the ground control station on the computer to ensure that all systems have been restarted.

If all systems have been reset and there is no apparent damage to the drone, then further diagnosis into the wiring would be required. It is possible that during flight or in a crash, some of the components may have become loose or unplugged. If all the wiring is found to be intact, then components will have to be removed and individually tested to isolate the problem.

2.4 Requirements Specification (CJ)

We were given constraints by our project sponsor, Lockheed Martin, for the basic design and implementation of our drone. However, since our drone was also be competing in a competition, there were performance requirements that were necessary in order to make the drone pass through obstacles in the competition. The requirements specification will serve as a basic guideline for the designing of the drone's dimensional structure, as well as certain functional attributes that will be necessary for the successful execution of the project. The following set of tables below outlines the original requirements we considered for our drone before disrupted by the COVID19 pandemic.

Table 2 below outlines the physical constraints that we must abide by for our drone.

Table 2: Physical Constraints

#	Requirement
1	The maximum drone size shall be 1.5 ft. x 1.5 ft. x 1.5 ft.
2	The drone shall be able to fly through a 3 ft diameter ring.
3	The drone shall weigh less than 5 pounds.

The Table 3 below outlines the operational requirements for our drone.

Table 3: Operational Requirements

#	Requirement
4	The drone shall not fly higher than 45 ft above ground level.
5	The drone shall navigate without use of GPS.
6	The drone will have 5 flight modes: <ol style="list-style-type: none"> 1. Auto-Navigation (AutoNav) 2. Auto-Maneuver 3. Manual 4. E-Stop 5. Take-Off/Land
7	In AutoNav flight mode, drone will detect, identify, and navigate to an obstacle
8	In Auto-Maneuver flight mode, drone will autonomously perform obstacle-specific maneuver.
9	In Manual flight mode, autonomous functions cease, and drone controls are transferred to human pilot.
10	In E-Stop mode, the drone stops all lateral movement and reduces the thrust of the propellers to initiate a controlled crash landing
11	In Take-Off/Land mode, the drone shall perform a controlled take-off and/or landing.
12	The drone will be able to identify and distinguish between the following obstacles: <ol style="list-style-type: none"> 1. Rings 2. Single Pylons 3. Double Pylons 4. Acoustic Waypoints.
13	When the drone encounters a ring, it will fly through it.
14	When the drone encounters a single pylon, it will loop around it once.
15	When the drone encounters a double pylon, it will loop around both and then pass between them.
16	When a drone detects an acoustic waypoint emitting a frequency between 0.5 - 1kHz, it will land in front of it for about 5 seconds before moving on.
17	The YOLO object detection algorithm cannot be used.
18	The minimum horizontal field-of-view for obstacle detection is 60°
19	Minimum flight time is 10 minutes under normal autonomous operation.
20	After drone reaches a wall approximately six feet in front, it returns to start position.

The Table 4 below outlines the communication requirements for our drone. These requirements are important to ensure the system is able to accurately communicate with the ground station. The system also transmits various measurement data, mapping data, and several other data types that will need to be logged and recorded on the ground station. This data is made easily accessible by both the user, and the drone system, as these data will be used to track the flight path. (RL)

Table 4: Communication Requirements

#	Requirement
21	The drone must respond to commands sent by a transmitter within 100 feet of the drone.
22	The drone will stream video wirelessly to a ground station laptop that is within a 100 feet radius.
23	Drone obstacles will be identified with a red “X” overlaid on the video stream
24	A metadata box will be displayed on the computer detailing the following information on the currently identified obstacle: <ol style="list-style-type: none"> 1. Type of obstacle 2. Confidence of identification 3. Distance to obstacle 4. Approximated time to arrival 5. Height Above Ground Level (HAGL)

All of the requirements mentioned above above are essential and vital to the success of this project. The team will have to ensure all requirements are strategically implemented into all design phases of the project. The requirements implementation will be particularly important for the prototype designing phase as it will allow the team to make adjustments and modifications to the design before final implementations on the main design. (RL)

2.5 House of Quality (RJ)

For this project, we were given customers for whom the UAV must be designed around. The customer’s expectations are a primary consideration when designing and implementing this drone, however, many of these aspects are directly correlated to each other. Therefore, it is important to identify the ways in which customer requirements can be set. This was accomplished through a House of Quality matrix. This matrix outlines the way that a customer’s requirements can be met, and provides traceability for its implementation. This highlights the relationship between marketing requirements and engineering

requirements. Marketing requirements are the requirements largely driven by the customer, and include some of the goals they hope to accomplish with the product. Engineering requirements are guidelines for implementation. A marketing requirement may affect the design choices for a product, and at the same time, an engineering requirements may affect certain marketability aspects of a product.

The House of Quality table below outlines the major engineering and marketing requirements. It also shows the relationship between engineering and marketing requirements, as well as between two engineering requirements. The up arrow shows if there is a positive correlation, while the down arrow shows there is a negative correlation between accomplishing the goals for both items. The double arrow indicates that a correlation between the two items is stronger, while a single arrow shows that they are weaker. If there is no appreciable correlation between two items, then the box shared between the two items might be left blank.

Moreover, the House of Quality matrix also shows the polarity on each engineering and marketing requirement, indicating whether we seek to minimize or maximize that attribute. The polarity briefly summarizes the targets for each requirement, and allows us to attempt improving the design to surpass minimum expectations. As an example, in our matrix, we show that we want to increase maneuverability of the drone (positive polarity), though the size and weight of the drone will apply a negative pressure on that marketing requirement. Moreover, we have specified that the size and weight of the drone have a target engineering requirement, in order to make sure that the technology can support what sought to create. However, this also gives us leeway in our design to be flexible as long as we do not cross any important thresholds.

In our analysis, we have found that the primary customer requirements included minimizing the completion time of the build and its associated costs, while increasing the maneuverability of the drone, effectiveness of the user interface, accuracy of object detection, and ability to defensively avoid adversarial mines.

As seen below in Figure 3, the matrix also includes specific engineering requirements that we must target when designing our drone. Some of the requirements, such as the 18" x 18" x 18" size, greater than 10 min of flight time, and less than \$1100 were specifically requested by our customer and must be met. However, other engineering requirements such as the weight, communication range, and field of vision are discretionary and provides a great deal of flexibility when designing the final product.

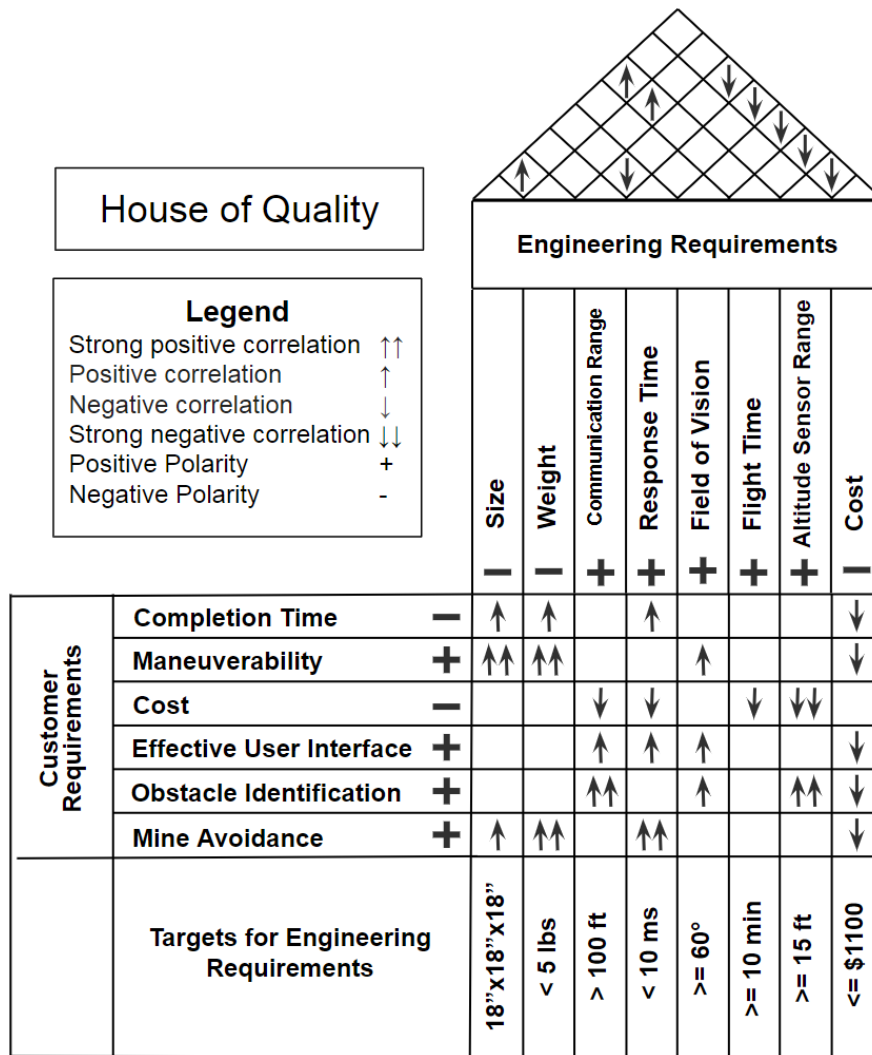


Figure 3: House of Quality Matrix (HS)

3.0 Project Research

The following sections of the project research will focus on project related products, technologies, parts selection and comparison, possible architectural designs and diagrams, along with the parts selection summary.

3.1 Similar Projects

3.1.1 Nvidia Trail Drone (CJ)

In July of 2017, NVIDIA researchers Smolyanskiy, Kamenev, Smith, and Birchfield published the paper *Toward Low-Flying Autonomous MAV Trail Navigation using Deep Neural Networks for Environmental Awareness* [17]. This paper details the methods they used to get a drone to navigate a course without

a GPS. This was done using a combination of neural networks, a single webcam, a Simultaneous Location And Mapping (SLAM) algorithm.

The hardware of their drone was composed of a 3DR Iris+ quadcopter with Pixhawk hardware and autopilot, the Jetson TX1, the PX4FLOW optical flow sensor, and a Microsoft webcam running in 720p at 30fps. Pixhawk is an open-hardware project that provides designs for flight control hardware, and Pixhawk is compatible with the PX4 open source flight control software that is also used on this project. The Jetson TX1 is actually very similar to the Jetson Nano that is described below in parts selection. Both the Jetson TX1 and the Jetson Nano use the same CPU (Cortex-A57), and the only major difference between the two is that the TX1 has double the number of CUDA cores in its GPU (256 cores). The PX4FLOW sensor is designed for optical flow calculations to approximate distance travelled and current speed.

The drone software uses the Robot Operating System (ROS) framework in order to standardize communication between different kinds of software. The basic flow of the program starts with the camera sending its images to three different software groups known as nodes: the researchers' own Deep Neural Network (DNN) called TrailNet, the YOLO network to detect objects, and the direct sparse odometry (DSO) SLAM algorithm.

TrailNet is a DNN designed to detect the drone's offset from the center of the trail and orientation with respect to the trail. The network's orientation output was trained with cameras pointed left off the trail, right off the trail, and center on the trail. The offset output was trained using three cameras facing forward on the trail with one in the center, one on the right edge, and one on the left edge. This neural network is run using TensorRT to take better advantage of the computer's GPU. If we decide to go with similar NVIDIA architecture (like the Jetson Nano) then we will want to implement our vision algorithms using TensorRT.

YOLO was used to detect obstacles such as other hikers and pets in order to avoid them. In a similar manner, we intend to use an algorithm similar to YOLO in order to identify objects, but will instead navigate our drone towards them rather than away from them. The YOLO algorithm was also implemented using TensorRT like TrailNet to improve its performance.

The DSO SLAM algorithm is used in conjunction with location information from the PX4FLOW to create a 3D point map of obstacles along the trail. This is done to ensure that the drone does not encounter any obstacles that may be located near the middle of the trail. This method was considered by us for our drone as a method to determining distance away from obstacles and for navigating around them.

The output from each of these modes are then fed into the controller node that determines the necessary action to take and sends it to the MAVROS node that

manages the Pixhawk directly. Additional input from a joystick is also used in order to control the drone manually and to create disruptions for testing purposes.

From this study, we found several technologies that we found useful in our own project. The ROS was a useful tool in implementing communication between different sections of our code. We thought that MAVROS could be used to enable easier communication between the computer and our flight controller. However, we found the MAVROS commands to be somewhat difficult to use. Additionally, we found that TensorRT greatly improved the performance of our object detection model on our computer.

3.1.2 MicLoc Sound Detection (HS)

MicLoc is an open-source project documented by author kryptor from the blog rural hacker. MicLoc uses sound triangulation to pinpoint an incoming sound in both 2D and 3D space (two versions outlined in the article). The sound detected in this particular project is a loud one shot noise that propagates from over 3 meters away and can pinpoint the sound within 5 centimeters of error.

Each microphone hears the sound at a different time, and the Time Difference of Arrival (TDoA) between each microphone is used to determine the direction of the sound (microphones that pick up the sound earlier will be closer to the target sound). TDoA is also used to also to determine how far away the sound is originating from as seen in Figure 4.

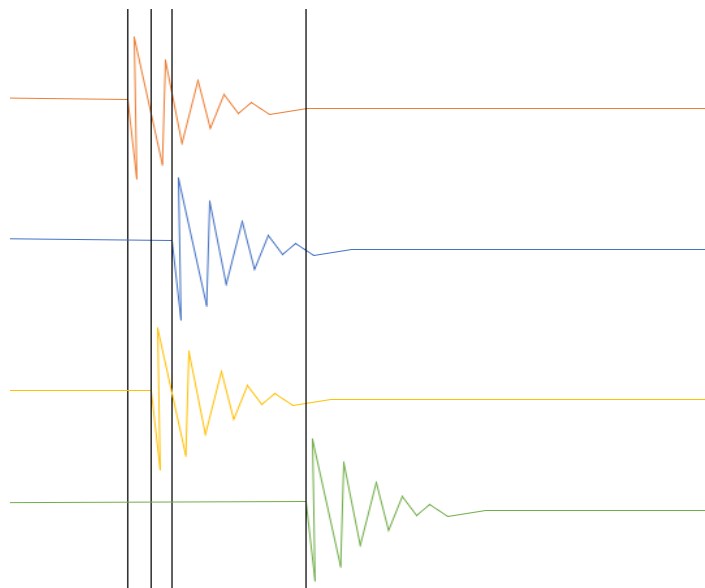


Figure 4: Example Diagram of individual microphone readings and comparisons between TDoAs

The hardware used for the MicLoc is a Teensy 3.1 microcontroller as well as 4 SparkFun Electret Microphone Breakout three pin microphones. For the 2D sound triangulation, the microphones are set up in a square shaped array.

Although the project is built using a Teensy 3.1 Microcontroller, this project can be adapted to work with an Arduino Uno which is outlined in an earlier version of the MicLoc on the same blog. [16]

3.2 Relevant Technologies

3.2.1 Image Recognition Methods (HS)

Object recognition is the use of a collection of related tasks for identifying objects in an image. Image classification involves assigning a class label to an image, whereas object localization involves drawing a bounding box around one or more objects in an image. Object detection combines these two tasks and draws a bounding box around each object of interest in the image and assigns them a class label, which together is referred to as object recognition. In our project, image classification was used to determine whether the object in view is a pylon, a double pylon, or a hoop to travel through. Object localization was planned to be used to draw the red colored X in the object to be traversed, as well as to determine the relative position of the obstacle in view so the drone can fly in the correct direction to complete the specified obstacle. Object recognition method hierarchy can be seen in Figure 5 below.

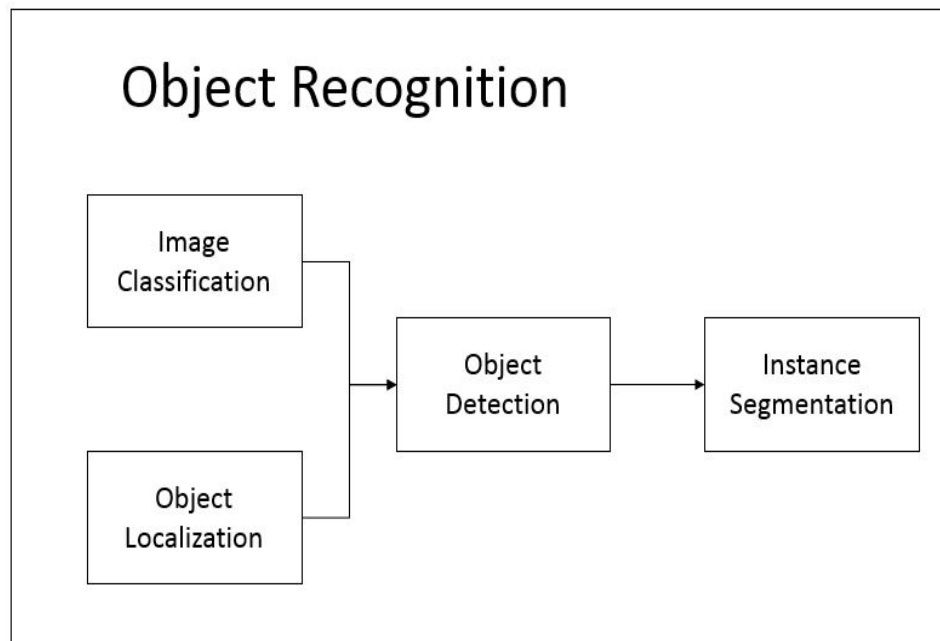


Figure 5: Object Recognition hierarchy diagram (HS)

3.2.1.1 Image Classification (HS)

Image classification involves predicting the class of one object in an image. The input would be images of a single object, such as an image of the pylons and hoops. The output would be a class label or several integers that are mapped to class labels, and a percentage tied to the class label indicating how similar or unsimilar the image in view is to the target.

Input for our drone would consist of a large pool of images of the pylons and hoops that will be used in the obstacle course for our drone. These images were used to make up a dataset for our deep learning algorithm to utilize to have a better understanding of what should be deemed an obstacle and what type of obstacle is directly ahead. For our drone, TensorFlow will be utilized to build the dataset of images for the obstacles. Images taken of the obstacles will be fed into TensorFlow, and a model will then be exported for use in OpenCV. Additionally, to aid in image classification, images that do not constitute an obstacle can be fed into a separate dataset.

3.2.1.2 Object Localization (HS)

Object localization refers to identifying the location of one or more objects in an image and drawing a bounding box around their extent. The input, like image classification, was images of a single object, such as an image of the pylons and hoops. The output of object localization is a bounding box, or multiple bounding boxes around the object which is identified within the image, defined by a point (x and y coordinates relative to the center of the camera), as well as width and height. There are a variety of TensorFlow and OpenCV algorithms that can produce a list of object categories present in the image, along with an axis-aligned bounding box indicating the position and scale of one instance of each object category. There are several methods for object localization, including Histogram back projections for fast image detection, MeanShift Algorithm, CAMshift Algorithm, Grabcut Algorithm, Convolutional Neural Network to predict four corners for the rectangle (CNN based Bounding Box regression model), and a Linear regression loss function to train the network.

3.2.1.3 Histogram Back Projection (HS)

Histogram back projection is used for image segmentation or finding objects of interest in an image. It creates an image of the same size and converts it into a single channel (black and white) of the input image. In the single channel image, each pixel corresponds to the probability of that pixel belonging to the target object. The output image will have the object of interest in the color white compared to the rest of the image.

Color and shape are both contributing factors to discerning if the object is in view. Multiple images of the same object can be fed into this algorithm, and each image will contribute a hue (color) and the location of each hue relative to each

other. These pixels constitute the resulting backprojection image. The values stored in the backprojection image represent the probability that a pixel within the image belongs to the object to be tracked. An example of image alteration for histogram BackProjection can be seen in Figure 6 below.

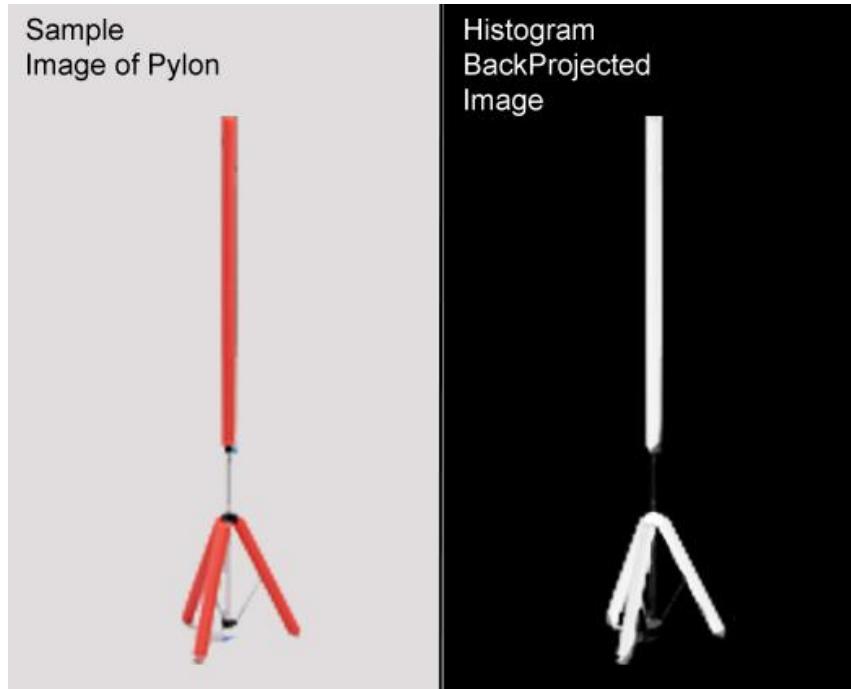


Figure 6: Example of image alteration for histogram BackProjection

3.2.1.4 MeanShift Algorithm (HS)

MeanShift is used to compare all the pixels that constitute the object that needs to be tracked, and then find the centroid of that object. For example, if you have a set of points and are given a small frame, to find the centroid of the points you have to move the frame to the area of maximum pixel density, or maximum number of points. This process is repeated in multiple iterations until the true centroid is found with the least amount of error.

For MeanShift algorithms, the histogram backprojection image is passed in as the initial target location. When the object moves, the movement is reflected in histogram backprojection image. As a result, the MeanShift algorithm moves the frame to the new location which contains the maximum density of pixels. An example of MeanShift movement across BackProjected image can be seen in Figure 7 below.

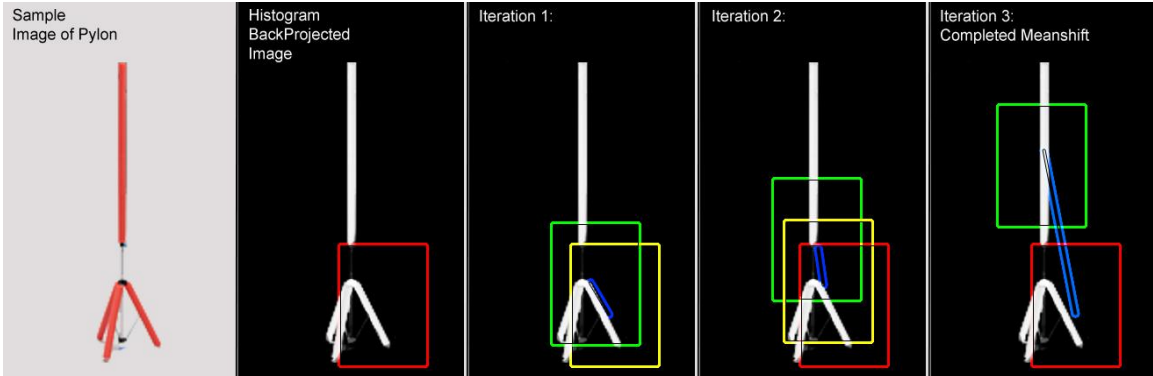


Figure 7: Example of MeanShift movement across BackProjected image

3.2.1.5 CAMshift Algorithm (HS)

Identifying the cluster of pixels using MeanShift is not enough to obtain the bounds of an object to be tracked. Not having the bounds of the object means we only know its origin, and not its relative width and height in order to determine the relative distance the object has between it and the camera.

To remedy this issue, CAMshift (Continuously Adaptive MeanShift) can be used to scale the frame of the points based on the size and rotation of the object. MeanShift works by using the completed MeanShift, then updating the scale of the frame.

Additionally, it applies a best fitting ellipse to the frame to determine object orientation. Then, MeanShift is applied again with the newly scaled frame until convergence. An example of CAMshift movement across BackProjected image can be seen in Figure 8 below.

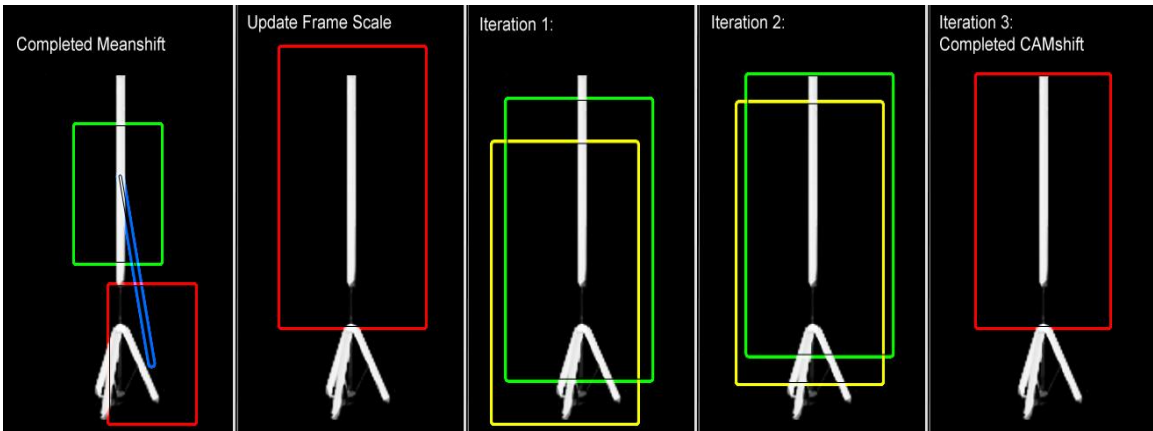


Figure 8: Example of CAMshift movement across BackProjected image

3.2.1.6 Grabcut Algorithm (HS)

The Grabcut algorithm is useful for highlighting the object in view, and is primarily used to live trace an object, or render everything in the background behind the object invisible. Grabcut works by using preset outlines set by the user which resemble the object to be tracked. The outlines are used to find and trim the object in view.

For our project, we need to indicate the object being tracked with a red X, but if we would like to take a step beyond just this, we can also highlight the object with a colored glow to indicate that we are tracking a specific object in case there are multiple objects in view behind the red X. An example of GrabCut being performed on a designated obstacle can be seen in Figure 9 below.

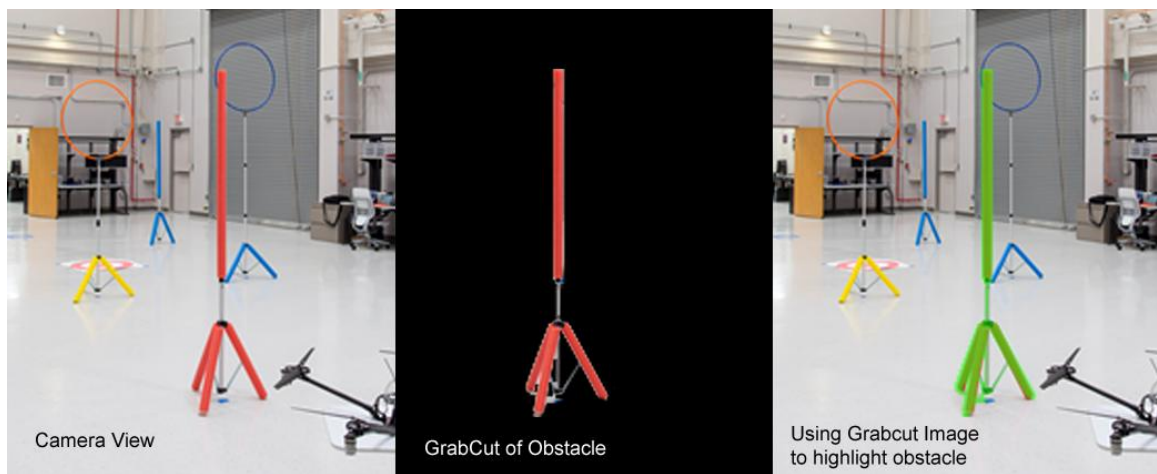


Figure 9: Example of GrabCut of a designated obstacle

3.2.1.7 Convolutional Neural Network (HS)

The role of the Convolutional Neural Network is to reduce the images into a form which are easier to process, without losing features which are critical for getting a good prediction. This is important for an algorithm to learn features, and to be usable for massive datasets.

Each color channel in an image is separated into their corresponding RGB layers, and then each of the three color layers are run under the convolution operation to extract the edges of the object, as seen in Figure 10. This data is then used to draw an outline around the desired object, with an x-coordinate, y-coordinate, width, and height.

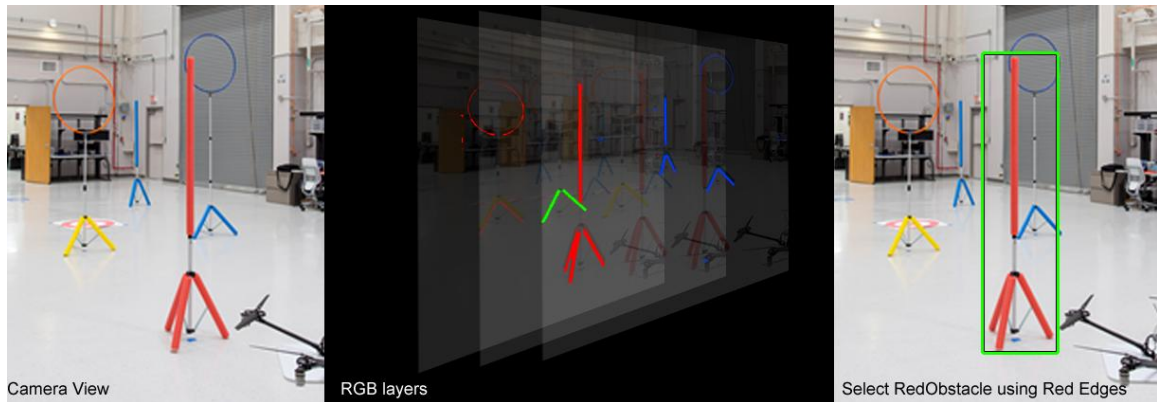


Figure 10: Object tracking using RGB layer separation (detecting red edges)

3.2.1.8 Linear Regression (HS)

Linear regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. It's used to predict values within a continuous range, rather than trying to classify them into categories. Linear regression training is used to improve the prediction equation by looping through the dataset multiple times, updating the weight and bias values in the direction indicated each iteration by the slope of the cost function (gradient).

Linear regression training is complete when we reach an acceptable error threshold, or when training iterations fail to reduce cost. simple linear regression draws and updates a regression line through points plotted using dataset images. The variables used in this algorithm represent the attributes and distinct pieces of information we have about each image.

3.2.1.9 Object Detection (HS)

For object detection, we are prohibited by the rules set by our sponsor to use the YOLO (You Only Look Once) object detection system. Instead, our drone will utilize the Single Shot Detector (SSD) object detection system to locate and track the obstacles to traverse.

Single Shot Detector achieves a good balance between speed and accuracy. SSD runs a convolutional network on input image only once and calculates a feature map. A small 3x3 sized convolutional kernel on the feature map is then used to predict the bounding boxes and classification probability. To handle size, SSD predicts bounding boxes after multiple convolutional layers. Since each convolutional layer operates at a different scale, it is able to detect objects of various sizes.

3.2.1.10 Instance Segmentation (HS)

When performing object detection the bounding box for each object is computed, then a class label is associated with each bounding box. The limitation of object detection is that it indicates nothing regarding the shape of the object itself while only displaying a bounding box. Instance segmentation, on the other hand, computes a pixel-wise mask for each object in the image. With instance segmentation foreground objects can be segmented from the remaining background. Instance Segmentation works with the help of the GrabCut algorithm used in object localization. A direct visual comparison between Object Detection and Instance Segmentation can be seen in Figure 11 below.

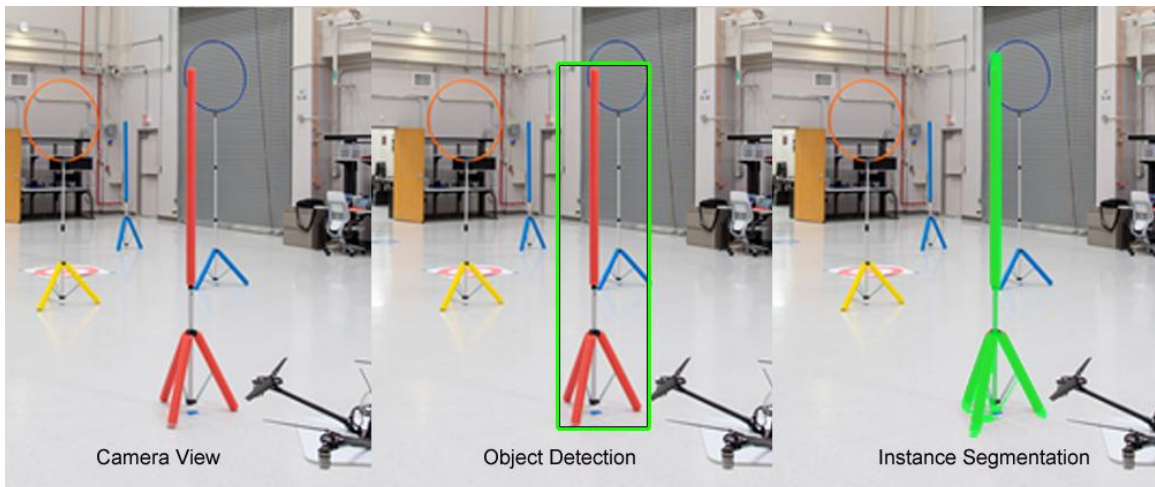


Figure 11: Object Detection versus Instance Segmentation

3.2.2 Computer Vision Libraries (HS)

There exists a wide range of computer vision algorithms in robotics for uses such as object recognition, image recognition, and simultaneous localization and mapping (SLAM) navigation. Cameras perceive colors differently than the way the human eye perceives the same colors. Human vision cannot accurately discern the true color of an object, particularly when a color is surrounded by other dark colors versus when a color is surrounded by other light colors. To measure the absolute, “real” color values of an object, the Red Green Blue (RGB) color coding system should be used to store the color data.

However, representing color in a format which is representative of how the human eye perceives the colors is very difficult. Rather than trying to create a sensor that operates like the human eye, Hue Saturation Value (HSV) color coding scale is used instead. The crux of our drone project is the detection of obstacles throughout the challenge course. In order to perform these tasks, there is little to no apparent need to take into consideration how the human eye might need to perceive the color of obstacles in the challenge course. Therefore, the drone should utilize the absolute color values and the RGB color coding scheme,

to compare the colors of the objects it detects. This ensures that the colors of the objects are consistently represented in a form that our drone can understand.

For this autonomous drone, Lockheed Martin prohibited the use of the YOLOv3 computer vision algorithm entirely. Four other open source deep learning algorithms that were researched for use in object detection, was OpenCV, Keras, PyTorch, and Caffe. 3 categories analyzed to compare the algorithms are image classification, object detection, and object tracking.

A test run on a 2 core CPU with 8GB RAM, no GPU, and on a Ubuntu 16.04 Operating system was conducted by Dr. Satya Mallick comparing the four algorithms with 100 runs each. For image classification, Caffe averaged 2200ms being the slowest, Keras averaged 500ms, OpenCV averaged 320ms, and PyTorch was the fastest at 280ms. For Object Detection, Keras, PyTorch, and Caffe averaged a very slow 27.832 seconds per frame using OpenMP. YOLOv3 with Darknet took 12.730 seconds per frame on average, and OpenCV was the most optimal, at a very fast 0.714 seconds per frame to detect an object as seen in Figure 12. Object tracking when compared to OpenCV was a landslide in OpenCV's favor, as it ran 6 times faster on average than the other algorithms. [6]

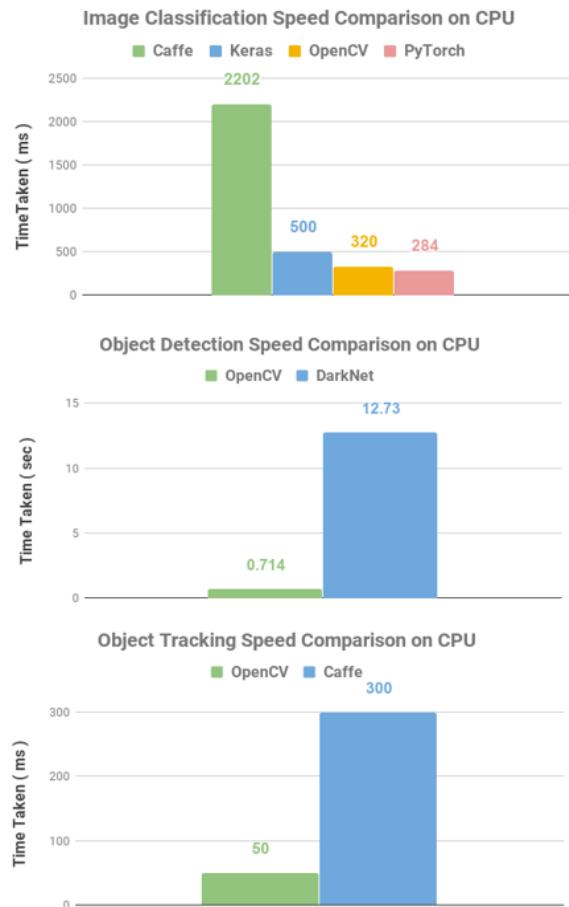


Figure 12: CPU Performance Comparison Deep Learning frameworks [6]

However, when we were looking for examples to build our object detection network, we were not able to find as many resources for OpenCV. For this reason, we decided to switch to use Tensorflow. In addition to being able to find more resources, we were able to more easily convert our Tensorflow model into a form that could be used by TensorRT. If we weren't able to use TensorRT, our model would have run significantly slower on our drone computer.

3.2.3 Object Detection Models (CJ)

In this section, we will determine which kind of object recognition algorithm we will use. Object detection is integral to the operation of our project. Our drone must be able to navigate to and around obstacles in order for the drone to complete the obstacle course. We believe the best way for the drone to determine where these obstacles are is to use a camera and process the image using object recognition. We will discuss several different algorithms and their performance. We will be excluding YOLO based algorithms from this list as our customer has requested that we use a different model.

Primarily, we will be looking at one-stage detection algorithms. This is because one-stage algorithms tend to be quicker and require less processing than multi-stage detectors. However, this comes with the tradeoff of less accurate results. In our project where the speed of the algorithm will effectively determine the speed at which we will be able to alter our course and make corrections, we are willing to prioritize speed over accuracy.

3.2.3.1 *CenterNet*

CenterNet is a relatively new algorithm created by Zhou, Wang, and Krähenbühl (2019). This detection algorithm finds objects first by identifying the center points of objects and then determining the bounding box. According to their study, CenterNet works faster than both YOLOv3 and RetinaNet while maintaining a similar if not greater level of accuracy. However, due to the model being so new, CenterNet does not have a lot of resources available for efficient implementation of the algorithm.

3.2.3.2 *RetinaNet*

RetinaNet is an object recognition algorithm created by Lin et al. (2017). It is a slightly older algorithm that is outperformed by YOLOv3 and CenterNet but provides better performance and accuracy than algorithms like YOLOv2 and Single Shot Detector (SSD). RetinaNet has been around for a while so there are some (but not many) resources to help us implement the algorithm.

3.2.3.3 *Single Shot Detector (SSD)*

SSD was developed by Liu et al. (2016). SSD works by creating a fixed set of bounding boxes and determining if there is an image in each of them. SSD is the oldest out of the three being compared. The algorithm is the least accurate out of

the three discussed, but it is a well supported algorithm that can work as fast as RetinaNet.

3.2.3.4 Model Decision

Between these two options, RetinaNet seemed initially to be the better choice. Though not as fast or accurate as CenterNet, it had more info on how to implement it. However, after doing more research while trying to implement a RetinaNet, we found it extremely difficult with our entry level knowledge of computer vision applications.

Due to the increased number of resources, we found that a SSD was the easiest to implement. In addition to being easier to implement, an SSD with a MobileNet V2 image recognition algorithm backing was found to run at 39 frames per second on our Jetson Nano according to Nvidia [26]. We could have chosen a different image recognition backing, but it wouldn't have operated nearly as fast. However, the speed comes at the cost of accuracy. However, in testing we found the model to be accurate enough for our project.

3.2.3.5 Model Training

An SSD MobileNet V2 model must be trained using a unique data set in order for the algorithm to properly detect our obstacles. We took images and pictures of the obstacles, label these images, and then convert them into a format that can be read by TensorFlow or OpenCV. Most of these images were used to train the database, and a small subset that were not be used for training were used to test how well the model operates.

Model training can take a long amount of time due to the number of images and the complexity of our model. Therefore, researched ways that will allow us to train our model in a relatively short amount of time. If we wish to train it ourselves, we will need to use a computer that has both a powerful CPU and GPU. However, while the model is being trained the computer will be practically unusable due to the workload the training will cause. Additionally, no group members have a computer that would allow for relatively quick computation.

Another option is using the Newton Visualization Cluster that is provided by the UCF Advanced Computing Research Center. The Newton cluster is composed of NVIDIA Tesla V100 GPUs that would allow our model to be trained in a few hours as opposed to a day or two. This would have required us to request permission from the Advanced Computing Research Center which would have taken some time, but would be free for us to use.

A third option is to rent GPUs from an online service such as Google's Cloud TPUs or Gradient. This would have allowed us to train our models quickly without the need to wait for approval from the research center. Some resources are available to users for free, but free plans often restrict how much data can be put

on their servers and how long the training to run for. Paid plans would provide us with more resources including memory and much longer run times.

Initially we believed that using cloud GPUs would have been the best option. However, due to the restriction on the free versions of the cloud GPUs and the high cost of using these GPUs, we decided to use one of our own computers. Additionally, it was quicker and more easily set up than working through an online portal.

Our SSD Mobilenet V2 model was trained using one of our personal computers with a GTX 980ti GPU using Tensorflow. Tensorflow was chosen due to its theoretical compatibility with the Jetson hardware. The images used for training were collected by us during our test sessions in the Lockheed Martin drone lab and outside, and these images were labeled using 'labellmg'. The configuration of our training was based on that Tensorflow researchers used for training an SSD for the COCO dataset.

Once the model was trained, the frozen model was transferred to the Jetson where the model was converted into a UFF file, a form usable by TensorRT. TensorRT was then used to optimize the model for use with the GPU located on the Jetson Nano. This process was difficult due to issues in compatibility between particular versions of Tensorflow and TensorRT. In this case, Tensorflow 14.0 was creating nodes in the graph that were unable to be properly interpreted by our version of TensorRT. We were able to resolve these issues after finding a source detailing how these issues could be resolved. [27]

3.2.4 Sound Detection Methods (HS)

Another obstacle that needs to be tracked in the course for the drone, is the acoustic waypoint with an unknown shape and size. This obstacle will emit a specific sound, and will need to be spatially located so the drone can land near or on top of it, and then lift off to detect the next obstacle.

To achieve this, our drone had an array of four microphones; a front facing, back facing, left facing, and right facing microphone. The left and right microphones will be used in tandem with the front microphone to triangulate the location of the acoustic waypoint. Additionally, the left and right microphones can be used with the rear facing microphone to obtain a second triangulation of the sound, in order to have a 360 degree range of sound detection. If this proved to not be accurate enough for our drone, we could have had 4 separate triangulations using the 4 microphones to get the best relative position of the sound. However, we found the initial configuration to have good performance.

3.2.4.1 Microphone Triangulation

The method used to triangulate the sound using microphones is simple and effective. The central microphone will act as a control for the sound, and then

there will be a check with the microphone to the left of the central microphone and a simultaneous check with the microphone to the right. The volume is compared from the side microphones to the central control microphone, and the sound direction is correlated with whichever microphone (left or right) picks up the most volume. Volume intensity can potentially be used to determine the distance of the acoustic waypoint, although we did not have enough time to test this.

3.2.4.2 Sound and Noise Filtering (HS)

There are two methods that can be utilized to filter out the drone's motor noise in order to accurately detect the acoustic waypoint; a hardware approach and a software approach. The hardware approach will focus on using more sound reducing propellers at the potential cost of performance, and the software approach will utilize the on board hardware-enabled algorithms by the XMOS XVF-3000 in the ReSpeaker Mic Array v2.0. Both of these methods were employed together to maximize noise reduction for the microphones.

3.2.4.3 Noise Filtering Hardware Approach (HS)

Minimizing the vibrational noise from the drones moving components can be achieved by focusing on different components of the drone such as the motor and the propeller, which can significantly reduce the amount of noise that comes from a drone.

One method of reducing drone noise is the use of larger slower propellers. Using larger propellers, means the motors can spin at a slower rate, thus making the drone quieter. Factors to consider when utilizing this method of noise reduction include drone max dimensions and weight limitations. Using a larger propeller means the motor will need more torque, and thusly it will need to be a larger, heavier, and more powerful motor, which will add to the overall weight of the drone. Additionally, using a larger propeller will infringe on the dimensional restrictions of the drone which limit the maximum width and length of the drone to 18 inches squared.

Another method of reducing noise is the use of shrouds. Propeller shrouds are a new method of reducing drone noise through the use of funnels made of nano-fiber or a similar solid light sound dampening material to capture most of the drone motor sounds and direct them upwards as seen in Figure 13. These shrouds act as a barrier around each of the propellers on the drone, which mitigates the sound made from the propeller tips cutting against the air surrounding it. This also serves as a dual purpose by protecting the propeller blades from objects, such as the obstacles our drone will pass through and around, as well as the adversarial mine that will be placed in the obstacle course.

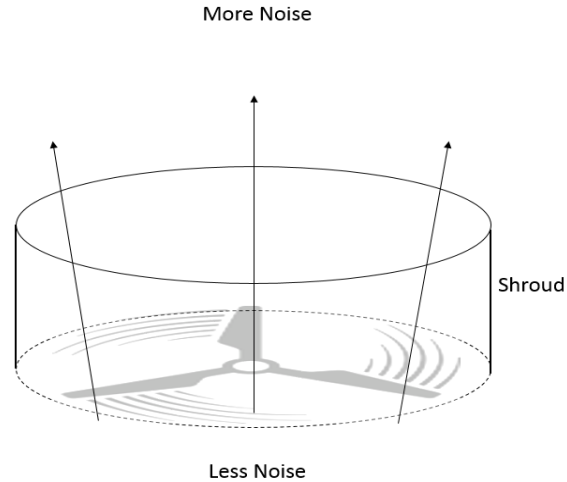


Figure 13: Propeller Shroud Diagram

The last method for reducing drone noise through hardware, is the use of low noise propellers. DJI engineered the Phantom 4 Low Noise propellers for their Mavic Pro drone, which utilize a raked wingtip design to lower the sound volume generated by the propellers by 5 decibels, and reduce the pitch of the sound as seen in Figure 14.

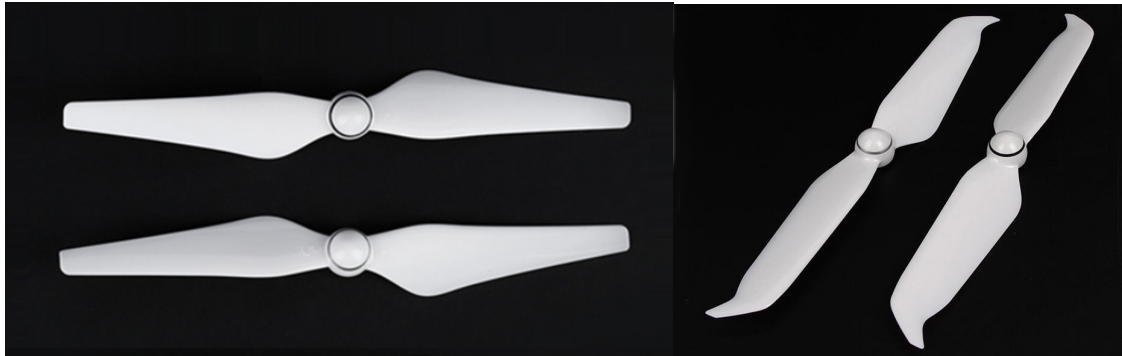


Figure 14: DJI Regular Propellers (Left) versus DJI Low Noise Propellers (Right)

Due to a change of sponsor requirements caused by the pandemic, we did not explore this further.

3.2.4.4 Noise Filtering Software Approach (HS)

Acoustic Echo Cancellation (AEC) is designed to remove echoes, reverberation, and unwanted added sounds from a signal that passes through an acoustic space. As shown in Figure 15, the sound coming from the remote sound, is sent in parallel to a digital signal processor path and to an acoustic path. The acoustic path consists of an amplifier, an acoustic environment, and a microphone returning the signal to the DSP.

The algorithm continuously adapts this filter to model the acoustic path, and the output of the filter is then subtracted from the acoustic path signal to produce a clean signal output with the linear portion of acoustic echoes largely removed. The AEC block also calculates a residual signal containing nonlinear acoustic artifacts. This signal is sent to a residual echo cancellation block that recovers the input signal even further. The signal is then passed through a noise reduction function to produce the filtered sound channel output. The filter stops filtering when it detects sounds in the unrelated direct sound input. This allows sound from the microphone to be added to an additional secondary sound channel independent from the first sound channel, as seen in Figure 15.

The ReSpeaker Mic Array V2 has the capability to utilize AEC to filter out the constant drone motor propeller sound and listen for the frequency emitted by the acoustic waypoint. The sound emitted by the acoustic waypoint will be a set frequency which can be filtered out. The array also has the capability to locate the origin of the sound omnidirectionally, which can then be fed into the Jetson Nano CPU to redirect the flight path of the drone when the sound is within a set vicinity.

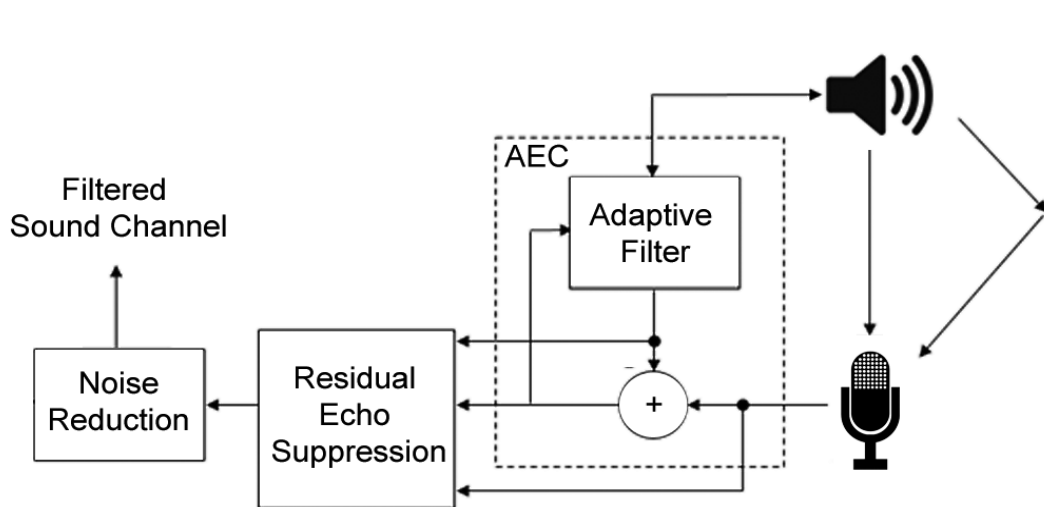


Figure 15: AEC Filter Diagram

3.2.4 Flight Controller Software (RJ)

An important consideration when finalizing a flight controller to use is determining which flight stack we will use. The flight stack is the firmware that the flight controller operates on, and can provide varying feature sets. The two primary open source flight stacks are PX4 and ArduPilot. The PX4 has been around since 2012, while the ArduPilot has been around since 2009 so they both have good documentation. The choice, for our purposes, will come down to preference and ease of use. For maximum flexibility when creating our drone, we wanted to have the option of using either flight stack as necessary even though we selected ArduPilot in the end.

3.3 Parts Selection

3.3.1 Computer (CJ)

3.3.1.1 Computer Requirements

The computer of the drone was used to control the main functions of the drone. These functions include image recognition on frames received from the camera, receiving messages from the ground station, running the autonomous loop code, sending commands to the flight controller, and sending video to the ground station. The computer needed to be powerful enough to run these programs simultaneously.

Image recognition required the most intensive processing, so computers that are able to speed up this processing will be preferable. Specifically, we looked at computers that have a Graphical Processing Unit (GPU). GPUs are processors specifically designed to do parallel computations. This makes them especially fast at doing the image convolutions necessary for identifying the obstacles our drone will need to fly towards and navigate around.

In addition to the capabilities of a computer, we also needed to account the weight and size of the computer. A very large computer will be unable to fit in the small amount of space we have on our drone, and a very heavy computer will require more power to keep the drone flying. For this reason, we looked for small, lightweight computer. Considering the above, we found two CPUs that could provide us both with the processing power we need as well as being small and lightweight.

3.3.1.2 Raspberry Pi 4 Model B

The Raspberry Pi 4 Model B (Pi 4B) is a computer that uses a BCM2711 chip, which contains a quad-core Cortex-A72 64-bit processor that can run at 1.5 GHz. The chip also contains 4GB of LPDDR4 memory and a VideoCore VI GPU that can be used by OpenGL ES, an embedded graphics API. The Pi 4B has 4 USB ports (two USB2, two USB3) and 40 GPIO pins that can support UART, I2C, and SPI input/output. It also has an ethernet port, two micro-HDMI ports, a Wifi module, and a microSD card slot (with included micro SD card) that is used for storage. The Pi 4B has a size of 85 x 56 x 16 mm and a mass of 46 grams (1.6 oz). The Pi 4B requires 5 volts to run. The processor itself uses 1 amp, but the board in total can use up to 3 amps.

3.3.1.3 Jetson Nano Development Kit

The Jetson Nano Development Kit is a computer that uses a quad-core Cortex-A57 64-bit processor that can run at 1.42 GHz. The computer contains 4GB of LPDDR4 memory and an NVIDIA Maxwell (128 CUDA core) GPU that can be utilized using NVIDIA's libraries. The Nano has 4 USB3 ports, 40 GPIO pins that can support UART, I2C, I2S, and SPI input/output. It has one HDMI A port, an

ethernet port, and a microSD card slot that is used for storage (no microSD card included). The Nano has no way out of the box to work over wifi, so wifi capability can be added via the M.2 E key slot. The Jetson Nano Development Kit has dimensions of 100 x 80 x 29 mm and a weight of 4.1 ounces. The Nano processor requires 5 volts to run and can run at 5 or 10 watts. The development board with peripherals can draw up to 4 amps. The Nano also comes with a heat sink.

3.3.1.4 Comparison & Selection

Comparing the processors, the computers were fairly similar in many ways. Both use a microSD card for storage and 4GB of LPDDR4 memory. Both have 40 GPIO pins that can be also be used for various kinds of serial communication. Both have ethernet ports. Both run on 5 volts. However they differ in other significant ways. They both have 4 USB ports, but the Nano’s ports are all USB3, which are faster than USB2, while the Pi 4B only has two USB3. This could enable faster data collection from USB based sensors. Wifi is also an important factor as the Nano does not come with a wifi module, meaning that it must be connected via ethernet or an external wifi module to retrieve important code bases from the internet.

GPUs of these devices weres very different. The Pi 4B has uses a VideoCore VI while the Nano uses a Maxwell containing 128 CUDA cores. There is not a lot of information on the technical specifications of the VideoCore VI. While OpenGL ES can be used to utilize the cores, the exact processing power of the VideoCore VI is not documented. The CUDA cores in the Maxwell are designed by NVIDIA, and TensorRT, a machine learning framework based on TensorFlow, is optimized to work on the Nano’s GPU. Additionally, these two devices have similar, but not identical processors. The Nano runs at a slightly slower clock rate (1.42 GHz) compared to the Pi 4B (1.5 GHz). The best way to determine which computer would be most Comparative benchmarking of object recognition by Alasdair Allan indicates that the Nano runs faster than the Pi 4B using TensorFlow when the TensorFlow code is optimized for TensorRT. [18] A comparison of both computers can be seen below in Table 5.

Table 5: Computer Comparison

Computer	Jetson Nano	Raspberry Pi 4B (4GB)
Processor	Cortex-A57 (4 cores)	Cortex-A72 (4 cores)
Clock Rate	1.42GHz	1.5GHz
Power Consumption	10 W or 5 W	5 W
GPU	Maxwell (128 CUDA cores)	VideoCore VI
Weight	4.1 oz	1.6 oz
Price	\$99	\$55

Using the benchmark along with the above information, we decided to use the Jetson Nano Development Kit. This kit provided quicker image recognition using TensorFlow than using the Pi 4B. Even though a wifi module needed to be bought to connect to the internet, this allowed us to pick a wifi module that will assuredly connect to our ground station from at least 100 feet away. Also, though the Nano is larger and heavier, it was not so much as to prevent its use in our project. Additionally, in the same benchmark Allan found that the Nano draws less current both when idle and when operating, though the power savings may be offset by the additional weight of the Nano. For these reasons, we have chosen the Jetson Nano.

We found the Jetson to work well for our project. It was able to process our images very quickly and was able to run all the software we needed to run. However, we had issues running it in the 10W mode when the computer was connected to the battery. For this reason, we consumed less power than we initially thought was needed. However, this was at the cost of things running slightly slower.

3.3.2 Flight Controller (RJ)

3.3.2.1 *Flight Controller Requirements*

The purpose of a flight controller is to control the movement of the drone by adjusting the power delivered to each motor via the electronic speed controllers. The flight controller also determines the orientation and movement of the drone using sensors such as accelerometers, gyrometers, and barometers. Flight controllers typically have a recursive control system to account for environmental factors in order to provide stable flight across a variety of conditions. Generally, flight controllers and their software provide a way for the user to configure their remote control in order to fly the drone. Many flight controllers have an SPI and I2C bus to connect peripherals to the board, and directly use those components to support flight through built-in functionality on the flight controller's firmware. Since we are planning on connecting the camera and microphone to our CPU so that our program can analyze the data and determine how the drone should respond, we need to interface the flight controller with the CPU. To accomplish this, we required the software of the flight controller to be open source so that we can modify the code to configure the exchange of information between the sensors, computer, and flight controller. Instead of using a remote control to command the inputs for flight, the CPU will provide the inputs.

To interface the CPU with the flight controller, we needed a flight controller that has a serial connection to implement the MAVLink protocol.

Flight controllers also come with 8-bit processors up to 32-bit processors. This processor bit count correlates to the performance of the flight controller, as the 32-bit ones are usually faster, thus operating the drone more smoothly. We would be flying in an environment with minimal tolerances; for example, the

limited width of a ring that our drone is required to fly through and the need to avoid adversarial mines at short notice, we will need a responsive flight controller.

One option for a flight controller would be to create our own using an Arduino microcontroller (or similar). The benefit to creating our own would be to have a flight controller that has all of the features we need, potentially at a lower cost. For example, several flight controllers include a GPS which we would not need for our project. We would, however, need to find the peripherals we want our flight controller to have and implement them ourselves. An Arduino Nano 3.x costs about \$10, and a 9-axis altitude-gyrometer-magnetometer combination sensor costs about \$8.49. The cost for producing our own Flight Controller would be less than \$70, however, we expected that going this route would involve a substantial amount of time to configure, debug, and test the flight control capabilities for this board. For this reason, we will opt to find an off the shelf Flight Controller that satisfy our needs. Moreover, selecting an off the shelf solution came with a considerable amount of product support and documentation that will aid us in trying to implement our autonomous drone.

3.3.2.2 Parts Comparison and Selection

A popular option for a flight controller is the HGLRC F4.V2 flight control board. It uses a 32-bit CPU which is one of the main features that we required. It also includes a barometer for altitude detection, and gyroscope, magnetometer, and accelerometer for orientation and speed detection. The downsides of this flight controller, however, it only runs on Betaflight flight stack, which is relatively newer than the other mentioned flight stacks and therefore might not have the documentation and support as those flight stacks. Specific to the HGLRC, though its price is \$33.99, it also does not have the capability to add many external components through the use of I2C and SPI. We will be using the CPU to manage many of the sensor inputs for object detection and tracking, however, we may utilize several of the flight stack features for other purposes.

A popular flight controller is the Readytosky Pixhawk. This flight controller costs \$72.99, but it uses the 32-bit ARM CortexM4 and has the ability to run the NuttX RTOS real-time operating system. The Readytosky Pixhawk also includes sensors including the barometer for altitude detection, and the gyroscope, magnetometer, and accelerometer for orientation and speed detection. This model Pixhawk allows support for both the PX4 and the ArduPilot flight stacks, which will allow us to have greater flexibility once we implement our drone. Most importantly, the hardware on this flight controller allows us to connect our Nvidia Jetson to send commands for flight. While we expect this to be our primary method of interfacing with the flight controller, this Pixhawk additionally provides an I2C and SPI bus which would allow us to also take data from our sensor components and use it directly into our flight stack for certain features that may not need to be implemented on the CPU. A comparison of the flight controllers is shown below in Table 6.

Table 6: Flight Controller Comparison

Flight Controller	HGLRC F4.V2	Readytosky Pixhawk
Processor	32-bit	32-bit
Flight Stack	BetaFlight	PX4 or Ardupilot
I2C	No	Yes
SPI	No	Yes
Price	\$33.99	\$72.99

We selected the Pixhawk not only because of its advanced features, but also because we were able to receive one free at cost from our sponsor. We found that our flight controller worked well enough for what we needed. However, it was a first generation model and we believe that choosing a more recent generation may have improved the performance of our position hold. In addition, our PixHawk did not come with all of the necessary cables to connect items to the flight controller. This created project delays as we had to wait for cables for certain components.

3.3.3 Camera (CJ)

3.3.3.1 Camera Requirements

The purpose of the camera on our drone was to identify and navigate towards obstacles using object recognition. In order to do so, we needed a camera that will meet the field of view requirement set by Lockheed Martin (not yet determined). In addition, we also needed to determine the distance to an obstacle both to report to the users over the video stream and for the purpose of maneuvering around the obstacle. In the following sections, we look at possible camera layouts and technologies that will help accomplish both goals.

3.3.3.2 Single Camera

A single front-facing camera would provide us a video stream to process and identify objects in its field of view. However, the main problem with this approach is the ability to determine distances to an object. Monocular (as in single camera) depth calculation is possible. In the NVIDIA Trail Drone study, the drone was able to create a 3D point cloud based using the direct sparse odometry (DSO) simultaneous localization and mapping (SLAM) method created by Engel, Kulton, and Cremers (2017).

The point cloud created using DSO is used by the trail drone to determine objects that are in the path of the drone and steer away from them as well as keep track of the position of the drone. This method works well for the trail drone

because it simply needs to avoid the large number of obstacles located on the edge of the trail. However, we believe that determining the exact distance to an object would be difficult with such a sparse cloud of points. After identifying an object via object recognition, we would need to determine which region(s) we would need to take points in the current image and use those to estimate the distance to the obstacle, and we believe this could be a difficult and time-consuming task to develop and make sufficiently accurate.

The benefit to using this method would be the ability to utilize the SLAM that is done during the calculation of the point cloud. This SLAM could be used along with our internal measurement unit (IMU) that can be found in our flight control board to give our movement commands greater precision.

3.3.3.3 Double Cameras (Stereo Vision)

The use of two front facing cameras would enable us to use stereo vision to calculate depth from the cameras by comparing the images of the two cameras. By determining the difference in locations of an object within both cameras' field of view, we are able to determine the distance away that object is. Closer objects will have a greater displacement between the two images, while objects farther away will have little displacement. By calculating this for the entire image, this gives us entire regions of dense depth information that we believe would make calculating the distance to an obstacle easier than with a sparse point cloud.

The main downside to the stereo vision method is the need for the drone computer to calculate the depth of the entire image. Because of this, it may be more computationally intensive compared to the monocular DSO method. Additionally, it may be harder to use a single video stream to do object recognition from due to each of the cameras being slightly off center.

3.3.3.4 Depth Cameras

A depth sensor is a component that is able to determine the depth to some object. A depth sensor accomplishes this by sending out a signal and measuring the amount of time it takes to return. The most basic forms of depth sensors simply send out a single signal straight out ahead of it. This can be good for estimating distance to large objects, but our obstacles, especially the ring, will be hard to measure due to its thin nature. The depth camera has two separate sensors, as shown below in Figure 16.

Instead, we considered depth cameras that can give us a 2D depth image of the environment in front of them. Such cameras include the Intel RealSense D400 series cameras. These sensors use two image sensors to calculate the depth of objects in front of them up to 10 meters away. This calculated image could then be sent over USB from the camera to the computer. [14]

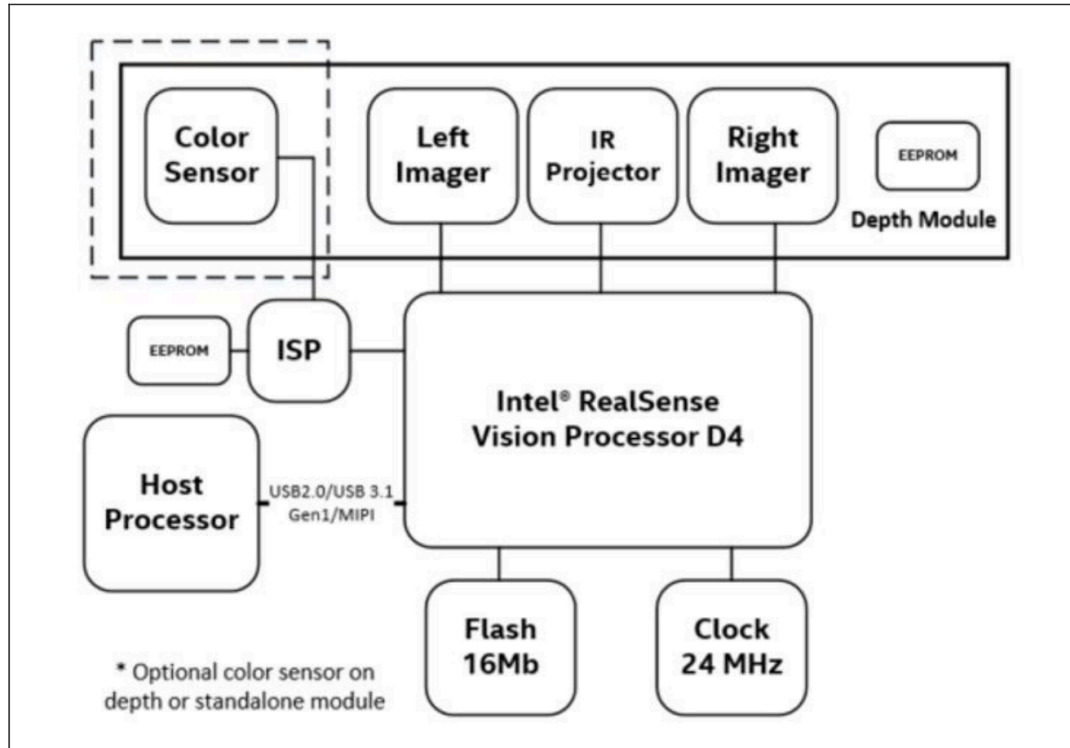


Figure 16: Intel RealSense Camera Architecture, courtesy of Intel [14][E]

The main benefit of this camera over standard cameras is that it calculates the entirety of the depth image on its own computer. This allowed us to free up a significant amount of processing on our main drone computer that can instead go towards object identification and determining the distance of an obstacle based on the depth image.

3.3.3.5 Method Comparison and Selection

We believe that computing an entire depth image based on stereo vision will prove to cause too much strain on our CPU. This leaves the two camera options that are less computationally expensive. By using DSO SLAM with a single camera, we would have both a way to calculate the distance to an obstacle and provide us with location information. The main downside to this method is the added computational cost of the method and the possible difficulty in determining distance to an object. If we were to use a depth camera, the calculation of depth image would be done entirely in the camera itself, leaving the computer to simply calculate the distance to an obstacle based on the depth image and the object identified. A SLAM algorithm, while not implemented by the camera, could be implemented by the computer to improve drone navigation.

Based on the above, we have decided to use a depth camera both to reduce the need for computation on the drone computer and the work needed to determine the distance to an object.

3.3.3.6 Part Comparison and Selection

There are two brands which we are looking at for depth cameras. The first brand we are looking at are the Intel RealSense D400 series depth cameras. These cameras use two monochrome image sensors to calculate the depth of any object within both sensors field-of-view (FOV) up to 10 meters. In order to improve the operation of these sensors in dim lighting conditions, the camera is equipped with infrared (IR) projectors to improve vision. These sensors produce a depth image with a resolution up to 720p @ up to 90fps. In addition these Intel depth cameras have an RGB image sensor (1080p @ 30 fps) with a FOV of 70° x 42° (Horizontal x Vertical). These cameras weigh 72 grams (2.5 oz) and can be plugged in via USB3 and can interface with the ROS [19] or the Ubuntu that runs on the drone computer.

These cameras come in three major SKUs. One is called the D415 and has a depth image FOV of 63.4° x 40.4° and a rolling shutter, meaning that the pixels values are taken one at a time as opposed to all at once. One is called the D435 and has a depth image FOV of 85.2° x 58° and a global shutter, which means that all pixel values are taken at once. The last one is called the D435i and is the same as the D435 with the addition of an inertial measurement unit (IMU).

The second brand we are looking at are the MYNT EYE S1030 cameras. These cameras have no RGB sensor, but make up for it in a FOV of 122° x 76° and a depth range of 18 meters. Both the output depth image and the stereo image sensors have a resolution of 480p @ 60fps. These also come standard with an IMU. These cameras come in two SKUs: one with an IR projector to make detecting depth in low light easier, and one without an IR projector. Either SKU can be plugged in via USB3 and can interface with the ROS or the Ubuntu that runs on the drone computer. A comparison between all of the cameras can be seen in Table 7 below.

Looking at our needs, we believe that having an IMU in our camera will be redundant due to being able to receive data from the IMU on the flight controller. This means paying extra for the Intel D435i would not make sense. This leaves us with four options. The cheapest of these options is the Intel D415 at \$149. This would likely work, but having a greater depth FOV is likely something that could help in identifying obstacles.

The next cheapest options are the Intel D435 and the MYNT EYE S1030-non-ir, both priced at \$179. The D435 has better depth resolution than the S1030-non-ir, but a smaller and shorter depth field of view. The D435 also has an RGB sensor and an IR projector that the S1030-non-ir does not. The IR sensor is not likely to be of much use to us due to the well lit conditions our drone will typically fly under. The RGB sensor on the other hand is very useful, especially if we are training our object recognition software with color images. The RGB sensor can also assist manual users in identifying important objects while using the drone remotely. The most expensive camera is S1030 at \$239, which when compared

to the S1030-non-ir only has added an IR projector which we believe to be unnecessary.

Table 7: Camera Comparison

Camera	D415	D435	D435i	S1030	S1030(no ir)
Depth FOV	63.4°x40.4°	85.2°x 58°	85.2°x 58°	122° x 76°	122° x 76°
Depth Resolution	720p @ 90fps	720p @ 90fps	720p @ 90fps	480p @ 60fps	480p @ 60fps
RGB FOV	70° x 42°	70° x 42°	70° x 42°	N/A	N/A
RGB Resolution	1080p @ 30 fps	1080p @ 30 fps	1080p @ 30 fps	N/A	N/A
IMU	No	No	Yes	Yes	Yes
Cost	\$149	\$179	\$199	\$239	\$179

Since having a smaller FOV would not preclude us from accomplishing our task and that a RGB video feed to assist in image recognition will be useful, we chose to use the Intel D435.

Our Intel D435 worked well and its ROS node allowed for easy integration. However, due to being unable to meet as a team after the COVID19 breakout, we were unable to implement the feature utilizing the depth image to determine the distance to objects. This wasn't required under the updated requirements.

3.3.4 Microphones (HS)

The purpose of the microphones on our drone is to identify and navigate towards the designated acoustic waypoint using sound recognition and detection. In order to do so, we needed an array of microphones that will be able to listen for the designated sounds the Lockheed Martin will provide us, as well as communicate with the on-board CPU. In the following sections, we looked at possible microphone arrays that will help accomplish this goal.

3.3.4.1 Seeed's ReSpeaker Mic Array v2.0

The ReSpeaker Mic Array v2.0 is a USB powered is a microphone array with 4 microphones, as well as a built in chip that contains sound triangulation and noise filtering. The board is capable of detecting voices up to 5 meters away with the presence of background noise. The onboard hardware-enabled algorithms are powered by the XMOS XVF-3000 enable the device to know the direction of a source, allows the device to focus only on sounds that come from the target direction, as well as ignore background noise.

Specifications include the XVF-3000 from XMOS audio processor for enhanced sound processing and filtering, four ST MP34DT01TR-M digital microphones, twelve programmable RGB LED Indicators, a maximum sample rate of 48Khz, and a voice capture radius of 10 feet, at a price of \$69.00. The greatest feature of this microphone array is that it has an advanced voice processor with built in Acoustic Echo Cancellation (AEC) to filter out the drone's noise to better hear for the designated frequency.

3.3.3.2 Seeed's ReSpeaker 2-Mics Pi

HAT is a dual-microphone expansion board for Raspberry Pi designed for AI and voice applications. The board is developed based on WM8960, a low power stereo codec. There are 2 microphones on both sides of the board for collecting sounds. it also provides 3 APA102 RGB LEDs, 1 User Button and 2 on-board Grove interfaces for expanding your applications.

This microphone array is light, minimalistic, and although it is designed for the Raspberry Pi, it is possible to integrate this device with other Linux based CPUs. Specifications Include the WM8960 low power stereo codec for audio processing, two MSM321A3729H9CP high performance analog microphones, a maximum sample rate of 48Khz, and a voice capture radius of 10 feet, at a price of \$9.90.

3.3.3.3 Seeed's ReSpeaker 4-Mic

Linear Array Kit is an extension board, aka HAT designed for a Raspberry Pi. It's a linear microphone array kit that comes with four microphones and designed for AI and voice applications. That means you can build a more powerful and flexible voice product with small CPUs. Although it is designed for the Raspberry Pi, it is possible to integrate this device with other Linux based CPUs. This microphone array specifications includes two X-Power AC108 ADCs for audio processing, four MSM321A3729H9CP high performance analog microphones, a maximum sample rate of 48Khz, and a voice capture radius of 10 feet, at a price of \$24.90.

3.3.3.4 Part Comparison and Selection

All three options listed above are compatible with the Jetson Nano. The stereo codec is the chip associated with sound recognition for all of the attached microphones in the array. This was important for our drone so it could pinpoint the specified sound among other noises within the obstacle room, including the drone's own motors and propellers.

The AC108 ADC is a high power codec with an emphasis on word detection, which is useful for detection of key words or sounds, as opposed to the WM8960, a low power stereo codec, more suited for sound detection on a small low power CPU, and not for precise audio recognition. Both the 4-Mic Linear Array Kit and 2-Mics Pi HAT share the same specifications for microphones, audio sample rate, and a voice capture radius of 3 meters.

The ReSpeaker Pi series is the cheaper brute force method to sound detection, while the Seeed's ReSpeaker Mic Array v2.0 is the more expensive and easier to implement option. The Seeed's ReSpeaker Mic Array v2.0 also is equipped with higher quality microphones that can detect sound from 5 meters away, as well as containing it's own on board chip for noise filtering and sound positioning. Considering the large budget with this project, and the capabilities of the XVF-3000 chip, the ReSpeaker Mic Array v2.0 will be the best suited choice for this drone to detect the acoustic waypoint. A direct comparison can be viewed in Table 8 below.

Table 8: Microphone Comparison

Microphone	Mic Array v2.0	2-Mics Pi	4-Mics Pi
Sound Processor	XMOS XVF-3000 (stereo-AEC voice processor)	WM8960 (low power stereo codec)	X-Power AC108 ADC (x2)
Microphones	MP34DT01TR-M (x4) (digital)	MSM321A3729H9 CP (x2) (analog)	MSM321A3729H9C P (x4) (analog)
Max Sample Rate	48Khz	48Khz	48Khz
Sound Capture Radius	16.4 feet	10 feet	10 feet
Cost	\$64.00	\$9.90	\$24.90

We chose the Mic Array v2.0 due to its 4-microphone configuration for sound triangulation. We were easily able to detect the angle of the sound relative to the forward position of the drone, and navigate to the source until we found that the sound was behind the drone.

3.3.5 Batteries (RL)

The Venom 4s, 30C, 3200mah, 14.8V Lithium Polymer (LiPo) battery was chosen to be utilized for the system power source based on several factors. LiPo batteries are popular batteries commonly used for drones. This is because the LiPo batteries have a softer outer casing, making the overall weight of the battery significantly lighter compared to other types of batteries, and thus makes this type of batteries an ideal choice for drone power supplies.

LiPo batteries are also capable of storing and delivering large amounts of power, which is ideal since the motors of the UAV will require a good amount for power to maintain thrust for takeoff and flight maneuvering. LiPo batteries have a high energy density and discharge rate which is essential for good power output. Lipo

batteries are typically rated by four key specifications; the charge capacity (usually displayed in mAh), the discharge or C-rating (usually displayed with a number followed by the letter "C"), the cell count (usually displayed with a number followed by the letter "S"), and the supply voltage (usually displayed in volts [V]). Thus, the Venom LiPo battery has 4 cells, with a discharge rate of 30 columns, a capacity of 3200 milli-amp hours, and a supply voltage of 14.8 volts.

The Traxxas 2890X was another possible LiPo battery to be considered for the drone project. It has a 6700mAh charge capacity which is slightly higher than the Venom. The number of cells and the output voltage of the battery is the same as the Venom as well. However, the battery's discharge rate is less than the Venom, being at 25C compared to the Venom's 30C discharge rate. The Tarras 2890X is also more costly than the Venom, making it economically less desirable as it makes budget management a bit difficult.

When comparing the two batteries, and taking into consideration the amount of power that will need to be delivered to each component, as well as the total flight time for the drone, the Venom was decided to be the better choice of the two. The Venom is expected to be sufficient to provide the power needed, at a lower price.

Proper precautionary steps need to be taken when it comes to dealing with LiPo batteries. Overcharging of such batteries can cause the battery to explode and catch fire, thus proper knowledge of charge time for the particular battery is important. It is also important to ensure the batteries' do not have any form of damages as it may cause the battery to explode and possibly lead to severe injuries.

During our project build, we estimated the battery to supply us with 7.7 minutes of flight time at an 85% discharge level. We were planning on ordering a higher capacity battery to swap out between rounds, but since the competition was cancelled, it was no longer necessary. Additionally, we were unable to procure additional parts from the university once spring break began.

3.3.6 Motors (RL)

For this project, the Cobra CM-2206 KV=2400 motors were chosen, due to its compatibility with the electronic speed controllers (ESC) and battery Components range from 2-5 cell, which meets the power demands and thrust requirement for the drone. The specs of the motor were determined by the mechanical and aerospace engineering students of the team.

We found that the motors we chose worked well enough for what we needed, but they needed to run at a fairly portion of their maximum power in order to keep the drone in flight. Because of this, the MAE team determined that newer, more powerful motors were to be ordered along with a higher capacity battery would

increase flight time. However, we were unable to obtain these due to ordering using our sponsor funds being suspended because of the pandemic.

3.3.7 Wireless Communication (HS)

We will use wireless communication to interact with our drone from the ground control station and to transmit the video feed. These wireless communications will consist of WiFi and radio.

3.3.7.1 Router

A router was used to transmit the WIFI signals back and forth between the drone's stereo camera and ground monitoring station. The project specifications require that the drone send camera feed data to device. Having a router means ensuring a strong secure signal enabling the ability to see what the drone sees and the data displayed on the OSD as well as targeting obstacles and displaying the red X on specified targets. We also considered hosting a Wi-Fi hotspot on the drone, but ended up using a network router as a connection bridge between both devices due for the greater performance (bandwidth) it provides.

3.3.7.2 Wi-Fi Transmitter

To interface with a designated router and laptop to display the camera feed with the overlay, the best option would be to connect via Wi-Fi to the on-board CPU using a USB Wi-Fi adapter. The Jetson nano does not inherently come with a Wi-Fi module, so this addition will be necessary. The Jetson Nano is a relatively newer CPU, and the only documented Wi-Fi module that works for it is via USB. The best option was the Geekworm Jetson Nano Wi-Fi Adapter Dual Band Wireless USB 3.0 Adapter. It features a 5dBi sma antenna, a Frequency: 2.4GHz and 5GHz, and a Transmission Power of less than 20dBm(EIRP). This is also one of the few small, long range dongles that work with Linux based operating systems.

3.3.7.3 Telemetry Radio

In order to manually control the drone from the ground control station, a telemetry radio is needed to connect to both the Pixhawk PX4 and the laptop, so it can be controlled via ground station software. One reliable telemetry radio device is the 3DR Radio Telemetry Kit 915Mhz Air and Ground Data Transmit Module. This device has one module connected directly to the flight controller, and the other module connected directly to the laptop via USB. We ended up connecting the module to our drone remote controller in manual flight, and using Wi-Fi for autonomous flight.

3.3.8 Optical Flow Camera (CJ)

3.3.8.1 Purpose

An optical flow camera is a camera designed to detect how far points in the scene have shifted in order to estimate how far the camera has travelled. We intended to use this camera along with a SLAM algorithm in order to determine current position and ground speed since we are unable to use GPS. By having two sources of data for positioning and ground speed, we hope to reduce the drift that may occur in our SLAM algorithm.

There were two main options for our optical flow camera. The first is to use a standard camera to get a video feed and then use an optical flow algorithm on the drone computer to calculate horizontal distance travelled. The second option is to use an already made optical flow module that will calculate the distance travelled on a separate computer.

Because we believe that an optical flow algorithm running alongside our object detection algorithm will significantly reduce the number of updates to the flight each second, we intended to use a pre-made board designed to output horizontal movement information.

3.3.8.2 PX4FLOW

The PX4FLOW is an optical camera designed to be interfaced with the PixHawk. It provides data on the horizontal distances covered by the drone. It updates its shifted position at a rate of 400Hz. In addition to the optical flow camera, it also has a built-in gyroscope to detect motion. This gyroscope is redundant due to the gyroscope in the PixHawk, but may be useful if we find it to be more accurate. The P4FLOW interfaces with the PixHawk using an I2C bus as seen in Figure 17 below, and the drone computer can get this information using serial communication with the PixHawk.

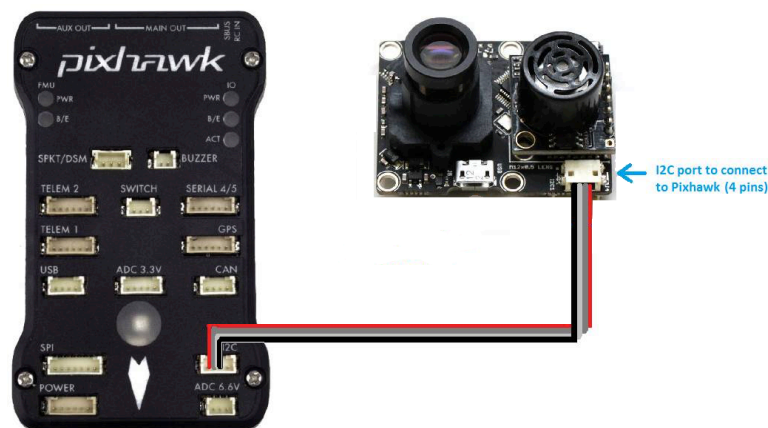


Figure 17: I2C Interface for Pixhawk [2]

The PX4FLOW also comes with a sonic distance sensor. This distance sensor is used by the PX4FLOW to estimate the distance from the ground to improve the estimation of velocity and distance travelled. If this sensor is accurate enough, we can use it as our main height sensor. This is discussed in the next section.

As we attempted to get our drone to hover in a single spot using the PX4FLOW, we were having issues with the drone making erratic adjustments and with the motors cutting out for no discernable reason. While trying to troubleshoot the problem, we discovered the PX4Flow was sending incorrect data to the flight controller. For this reason, we decided to get a HereFlow, which is described below.

3.3.8.3 Hereflow Optical Flow/Lidar

A HereFlow is designed in much the same way as the PX4Flow. It uses a camera to determine how much the drone is moving based on the movement of pixels in the camera. It also contains an accelerometer to assist in this determination. The HereFlow uses a LiDAR distance sensor to determine height. This LiDAR distance sensor was fairly weak as outdoors it was only supposed to work up to half a meter. Indoors it was able to work up to 6m. While trying to integrate it, we had similar issues with correction as when we were using the PX4Flow though the data from the sensor seemed to be more correct. These corrections were not as extreme however, and we used it in our final build.

3.3.9 Time of Flight Distance Sensors

3.3.9.1 Purpose (CJ)

The use of a Time of Flight (TOF) distance sensor is imperative for calculating the distance between the drone and the surface beneath it as well as the distance between the drone and obstacles horizontal to it. TOF distance sensors measure the amount of time it takes for a signal to travel from the emitter, bounce off and object, and return to the receiver. The TOF distance sensors will be used in two different manners. First, a distance sensor facing downward will be used to estimate the height above ground level of the drone. Second, the drone will use sensors placed at the front, back, and two sides to detect any obstacles close to the drone.

We require a height sensor that will give us an approximate height of up to at least 20 feet. Since we are required to stay below 45 feet, we know if we lose the signal from the distance sensor that we will have to return to a lower height. Additionally we want a sensor that will operate at a frequency of at least 10 Hz to ensure that our computer is supplied with height data regularly.

As for the horizontal object detection sensors, we needed a range of at least 5 feet to prevent the drone from coming too close to objects and a frequency of at least 10 Hz for the computer to receive distances at a regular pace.

3.3.9.2 HC-SR04 Ultrasonic Range Sensor (HS, CJ)

This very cheap and economical sensor provides 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm. Each HC-SR04 module includes an ultrasonic transmitter, a receiver and a control circuit. This sensor can interface directly with the Jetson Nano via GPIO pins or via the GPIO pins on an Arduino microcontroller.

In order to get the output distance, a trigger pin must first be raised for 10 μ s. The length of time it takes for the echo pin to rise after the trigger is the amount of time it took for the signal to travel. Using the speed of sound, this was used to calculate the distance of the object it bounced off from.

3.3.9.3 HRLV-MaxSonar EZ4 on the PX4FLOW (CJ)

This distance sensor uses ultrasound to determine the distance to an object and is attached to the PX4FLOW. The PX4FLOW provides distance information digitally via an I2C connection to the PixHawk flight controller. The height data can then be retrieved by the drone computer via a serial link. The PX4FLOW is planned to be installed downward in order to approximate the horizontal distance the drone is travelling, so using the ultrasonic sensor attached to it would require no additional hardware if used for height detection. The sensor is able to operate at 10Hz and has a range of 5 meters.

3.3.9.4 TeraRanger One (CJ)

This sensor uses an infrared beam to determine the distance and is designed to interface with a computer via UART or I2C. The sensor requires 12V, so a separate voltage regulator may be necessary in powering it. Additionally, the UART output voltage is 5V, which is too large for the Jetson Nano. If UART communication was to be used, the output voltage would have to be reduced. I2C communication is used, the output from the I2C conversion cable would provide the output in 3.3V, which is accepted by the Nano. However, this sensor is able to operate at 1000 Hz and provide a range of up to 14 meters, which is better than the other two sensors.

3.3.9.5 TF Mini LiDAR (HS, CJ)

Within 12 meters the product can measure 100 times within one second and produce centimeter-level reliable data. This sensor can interface with a computer or directly to the pixhawk flight board. The upside to this sensor is that it runs off of 5V, which can be taken directly from the Jetson Nano Dev Board. Additionally, it outputs 3.3V which can interface with the Nano without the need for voltage division.

3.3.9.6 Part Comparison and Selection (CJ)

First, we will consider the needs for the height sensor. Table 9 shows a comparison of all of the distance sensors being considered. The HC-SR04 ultrasonic sensor is inexpensive, but the implementation of this device will require creating a function that is able to send and receive signals from the GPIO pins. Additionally its range means that we will not be able to tell how high the drone is when it is above 4 m (13 ft). The HRLV-MaxSonar sensor works on the same principle as the HC-SR04, but communicates via USB, making it significantly easier to interface with. The sensor works between 30cm and 5m (16ft), which gives us 3 feet more range than the HC-SR04. Additionally, the HRLV-MaxSonar comes with the PX4FLOW, meaning that we would not have to spend any additional money to acquire it.

Table 9: Distance Sensor Comparison

	HC-SR04	HRLV-MaxSonar EZ4	TeraRanger One	TF Mini LiDAR	HereFlow LiDAR
Operating Voltage	5V	3.3V	12V	5V	5V
Average Operating Current	15mA	2.5mA	50mA	120mA	100 mA
Average Power Consumption	0.075 W	8.25mW	0.6W	0.6W	0.5 W
Sensor Range	2cm to 400cm	30cm to 500cm	Up to 14m	30cm to 1200cm	80 mm +
Range Resolution	N/A	1 mm	0.5 cm	1 cm	N/A
Field of View	15°	~3°	3°	2.3°	27°
Update Rate	40Hz	10Hz	1000Hz	100Hz	50 Hz
Communication	GPIO	USB	UART, I2C	UART	CAN
Price	\$3.95	\$109.99 w/PX4FLOW	\$59.50	\$39.95	\$49.99

The TeraRanger One is the most expensive option and would require a regulator to provide it 12V. Additionally, the UART signals required for the UART are too high, so an I2C adapter would be necessary. This I2C adapter would provide us the flexibility to either interface with the Nano or the PixHawk. If the hurdles of this device are overcome, it has the longest range (14m) and the fastest operating frequency (1000Hz). The TF Mini is twenty dollars cheaper than the One. It operates off of 5V and uses 3.3V UART signals. This means that it would be very easy to attach to the Jetson Nano. It provides a fast operating frequency of 100Hz and works up to 12m.

Based on our analysis above, we originally chose to go with the HRLV-MaxSonar on the PX4FLOW for our height sensor. Though it does not have as great of a range as the TeraRanger One or the TF Mini LiDAR, the range is suitable for our needs and will not require the incorporation of additional hardware. Once we determined the performance was not high enough for our position hold feature, we experimented with the HereFlow Optical Flow/Lidar sensor.

For our horizontal distance sensors used for object detection, we needed sensors that are fairly cheap due to our need to have five of them. For this reason, we plan to use four HC-SR04 sensor. Since these sensors require the use of interrupts in order to be used most accurately, we plan on creating a printed circuit board (PCB) with a microcontroller on it to manage the sensors and send data back to the Jetson Nano via a UART connection. This is to prevent interrupts on the Nano from interfering with processes integral to the flight of the drone.

3.3.10 Drone Starter Kit (RJ)

To begin developing our drone, we needed to prototype our implementation on an existing product to speed up development. The advantage to using a drone starter kit is that we don't need to build a frame from scratch, which would add to the amount of time spent designing and testing our drone. However, we would need to readapt our starter kit for our final product down the road, and between the starter kit and readapting our drone to meet the final requirements, a significant amount of our budget would be spent. Our primary motivation for the beginning of our project is to make sure that we can begin collecting data for our object recognition algorithm, and to ensure that the flow of data operates correctly.

The first drone starter kit we looked at was the LHI 240mm starter kit that includes the frame, 4x brushless motors, Omnibus Flight Controller, and a camera. The transmitter, receiver, and battery were not included and would need to be purchased separately. This kit cost \$189.00, but we were not able to source it domestically in accordance with UCF's purchasing policy. Additionally, the basic flight controller and camera may not have been suitable for our autonomous functionality, and we may not have been able to reuse these parts for our final drone project.

The next kit we looked at was the DJI Flamewheel F450. This kit includes the frame and the motors, but not the flight controller, camera, receiver, transmitter, or battery. At \$239.99, this drone kit is substantially more expensive than the initial kit we were looking at while coming with less components.

The benefit to using our own flight controller and camera during our prototyping phase is that we can directly transfer our implementation to our final product with minimal reengineering since we would not need to determine how to transfer our

data to another platform. Another added benefit of going with the DJI Flamewheel F450 is that DJI is a reputable company in the U.S., and there is a lot of documentation and support for their products.

The project requirements specify that the drone cannot exceed a size of 1.5 ft x 1.5 ft x 1.5 ft, however the F450 would exceed this requirement. For our final implementation, considered modify the frame to comply with the size requirements, and we will build our own frame. Though we would be able to reuse the motors, flight controller, and cameras from our prototype into our final build, we would be challenged with having to convert the frame into one that is compliant with our engineering requirements with a limited remaining budget.

In addition to our constrained budget and strict size requirements for our drone, we became pressed on time as we did not have the authorization to order parts using our allocated funds until mid-November. Instead, in collaboration with the mechanical and aerospace engineering (MAE) group members of our team, we decided to split the prototyping of our project into separate categories and began working towards each category before receiving authorization to order parts.

In one category, the MAE team began designing a frame from scratch that would meet the size requirements, and then fabricate it using 3D printing. Creating a modular design for the different components of the drone would allow us to easily modify individual parts and quickly reprint them if we found that the design needed to be modified to mount a new component. We would also be able to quickly repair our drone in case it crashes and cracks some components in the frame. In the second category of our prototyping, the ECE team would begin configuring the CPU for object detection and autonomous maneuvering.

We, the ECE team, began collecting sample data through images of the obstacles taken in various locations, lighting conditions, angles, and heights so that we could begin building a dataset to train our algorithm. Additionally, we began working on configuring the flight stack and determining how raw sensor data can be translated into meaningful instructions for the movement of the drone. Once we began to combine both categories of prototyping, we we planned to tweak our design and tune our implementation to reach our final product without a significant amount of last minute reengineering during the second semester.

We had a preliminary drone frame to begin our prototyping that was 3D printed using polyethylene terephthalate glycol, also known as PET-G. The preliminary design made by our MAE team is shown below in Figure 18-A. PET-G is advantageous because it has strong impact resistance while also having some flexibility. We expect that the drone may crash during testing, so having strong impact resistance is crucial for our design. Moreover, the preliminary design is modular, meaning that the base and individual arms of the frame can be disassembled and easily replaced as necessary. We did not have to pay for the

printing of our frame, but we estimate the PET-G filament to cost approximately \$25.99 per kilogram. The weight of the frame is 292 grams, making our estimate cost to be \$7.59.

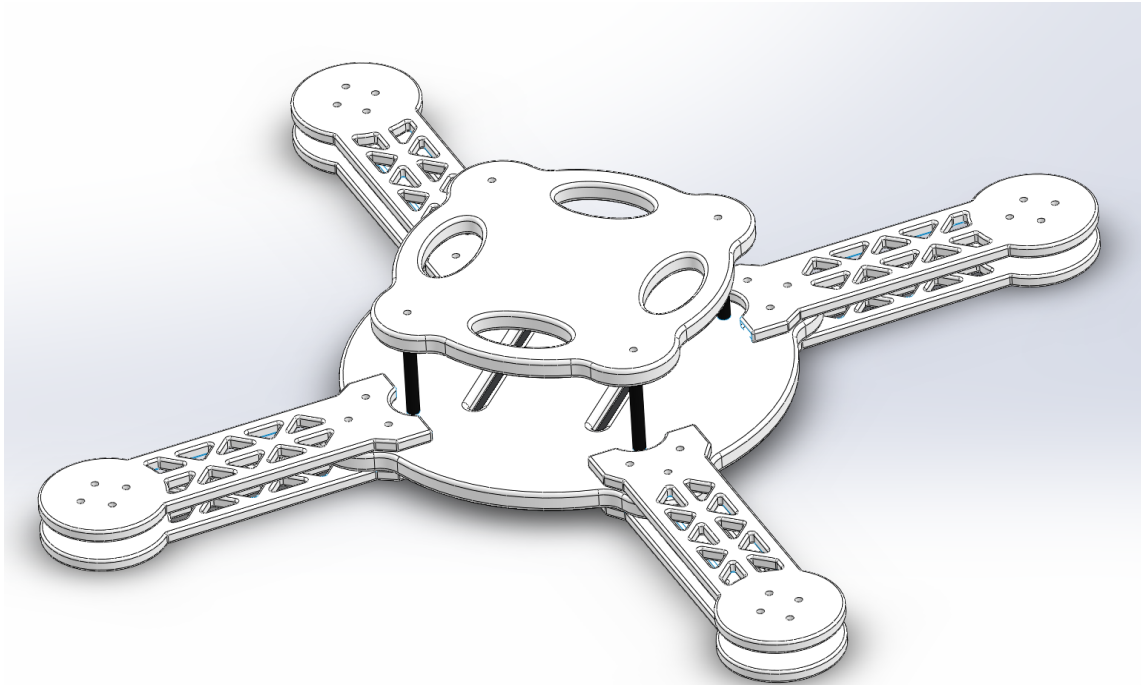


Figure 18-A: Preliminary 3D printed drone frame

In case the frame turns out to be unsuitable (due to design flaws, manufacturing errors, or size constraints) by the time we begin prototyping by the start of the spring semester, then we would will order a frame-only kit to build our drone upon. However, we found that the initial design was good and we were able to begin prototyping right away. The only major changes that needed to be made were mounts to house our components. The final 3D printed drone frame can be seen in Figure 18-B.

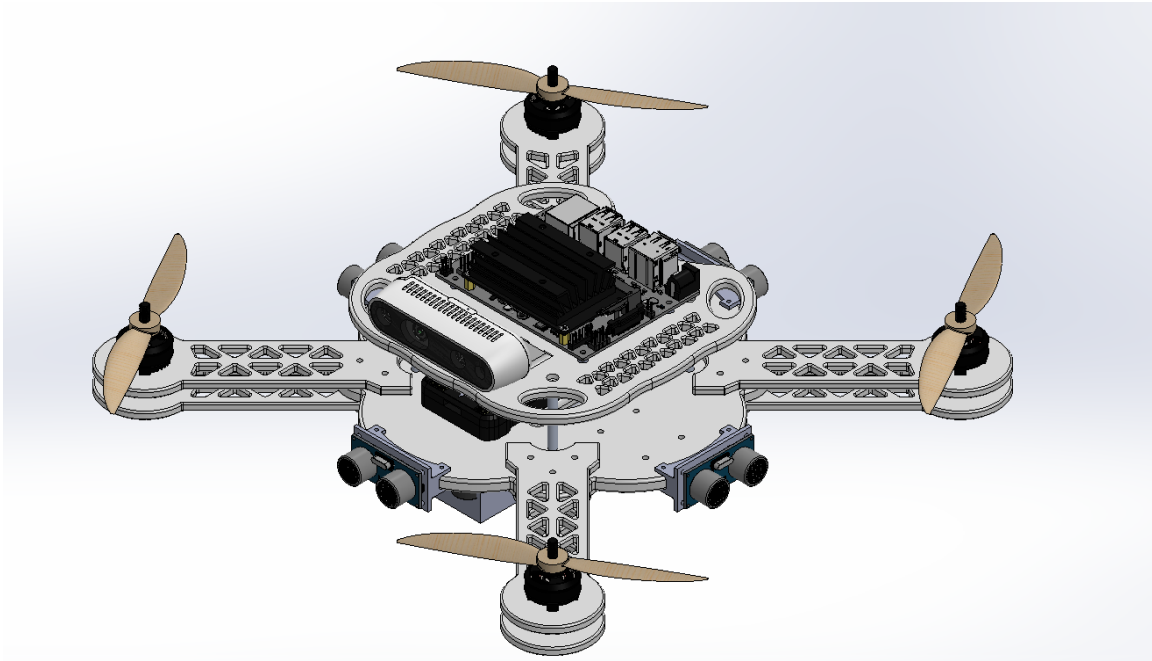


Figure 18-B: Final 3D printed drone frame with components

The frame size is 330 mm (which is roughly 12.99 inches), as measured diagonally from the placement of the motors. Since our project cannot exceed 18" x 18" x 18", that means our maximum diagonal distance, including the tips of the propellers, is 25.46 inches. For this frame, the recommended propeller size is 6 inches, which means roughly 3 inches of the propeller will hang off each corner of the frame, bringing our diagonal distance from 13 inches (frame only) to 19 inches (frame plus motors).

Though we are allocated around 6 extra inches for our maximum allowed size, it is difficult to find a frame kit that is in the 350-450 mm range. Moreover, a smaller sized frame provides an advantage that we could capitalize on, allowing us to maneuver the drone in and around obstacles with a larger margin of error. The USAQ Alien 300 mm frame costs \$26.95, which would not be a significant burden to our budget. However, this was unnecessary as we were able to use our team's developed frame.

3.3.11 Part Selection Summary

Table 10 below summarizes the chosen parts for our initial project prototype. We received the flight controller and PX4Flow sensor free of cost as it was donated to us by our sponsor. However, we included the cost as a reference to our budget.

Table 10: Initial Build Parts Summary and Prices

Part	Price
NVIDIA Jetson Nano	\$99.00
Drone Frame	\$7.59
PixHawk Flight Controller	\$72.99
Geekworm Jetson Nano WiFi	\$18.99
Venom 4s 30c	\$69.99
Intel D435 Camera	\$179.99
PX4FLOW	\$109.99
CM-2206/17-V2 Multicopter Motors (4)	\$91.96
3DR Radio Telemetry Air and Ground Data Transmit Module	\$22.99
ReSpeaker Mic Array v2.0	\$64.99

Though these parts were used for our initial prototype, they were carefully chosen based on technology we want to use for our final build. Therefore, most of these parts will be transferred to our final project and we will be able to save time by not switching to and needing to reimplement our autonomous functionality for different parts down the road. Since we will be reusing these parts, the amount taken out of our prototype budget will end up being the cost of components that were not used in the final build. Therefore, we were significantly under budget at the beginning of prototyping.

3.4 Possible Architectures and Related Diagrams (RL,RJ)

Figure 19 below outlines the basic architecture of the drone. The 'Sensors' box includes the camera, microphone, and sonar detection. Raw data is sent to the CPU, and used to determine where the drone should move to next. The CPU will send the data inputs to the flight controller (similar to a remote control input). Finally, the flight controller directly controls the motors using electrical signals.

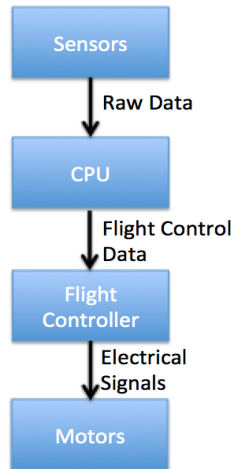


Figure 19: Conversion of data to electrical signals

3.4.1 Emergency Stop (RL)

The emergency stop system for the drone will primarily be sent from the ground computer via a wireless router. The drone CPU will receive this command and execute the preprogrammed emergency landing procedure where it will stop all lateral movements of the drone and reduce power to the motors rather than completely shutdown power. This will cause the drone to make a soft landing. Once the CPU has determined the drone is completely grounded, it will completely shutdown all power systems. Figure 20 below demonstrates the emergency stop sequence.

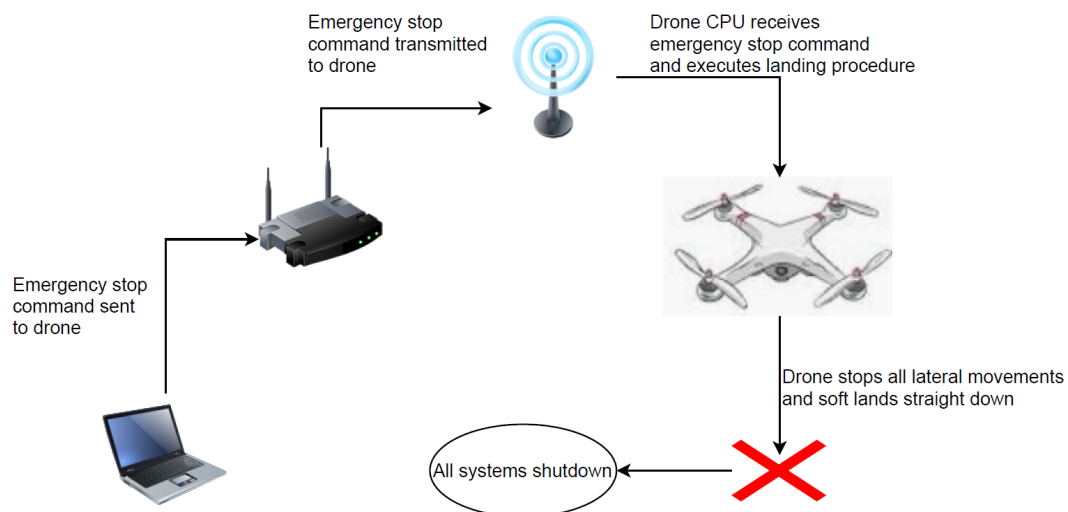


Figure 20: Emergency stop process (RL)

3.4.2 Data Flow (RL)

Certain sets of data will be transferred between the ground computer and the drone CPU. Figure 21 shows the different data transferred between the ground computer and the drone CPU. Some data will be transferable back and forth between the devices, while most will be transferred in one direction.

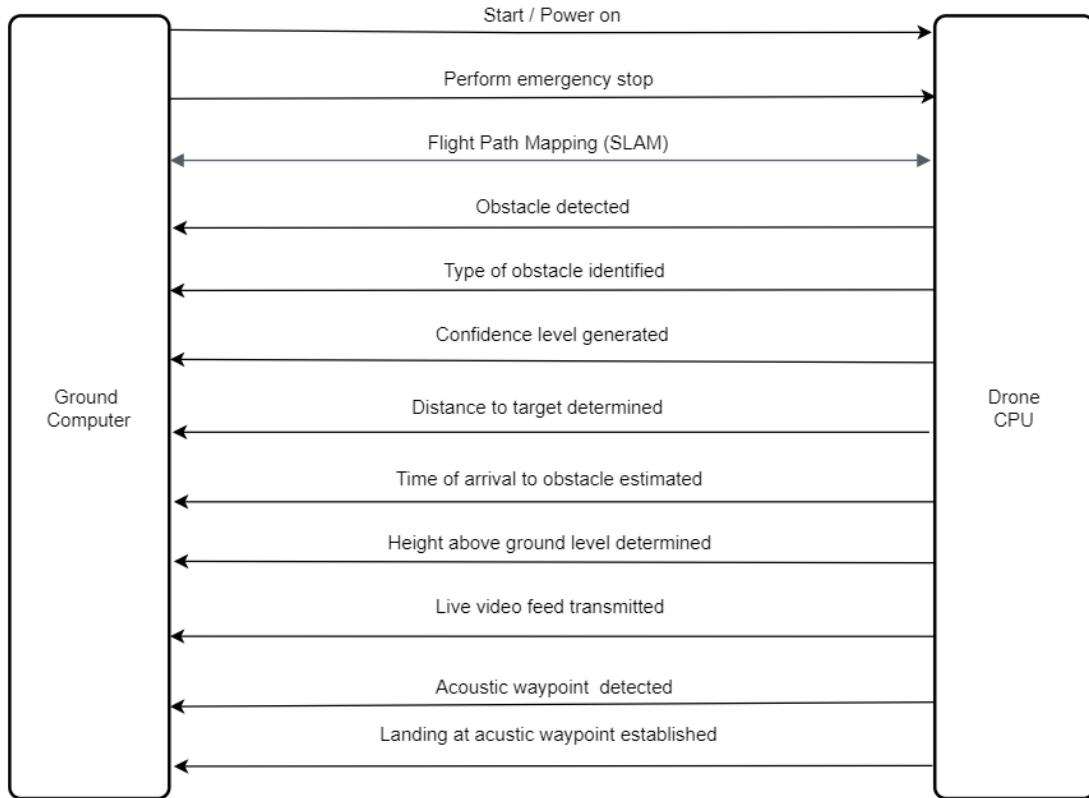


Figure 21: Datapath between ground computer and drone CPU (RL)

3.4.3 Obstacle Recognition and Maneuvering Sequence (RL)

The CPU processes image data received from the camera and use it to identify obstacles amongst objects it detects in the images. Object recognition coding is be implemented into the system's programming. It then pairs a set of maneuvering instructions that are preprogrammed into its coding and specific to each type of obstacle. There are three types of obstacles the system will need to be able to recognize and pair the maneuvering sequence with. Figure 22 shows the steps of the CPU image processing and maneuvering instructions pairing.

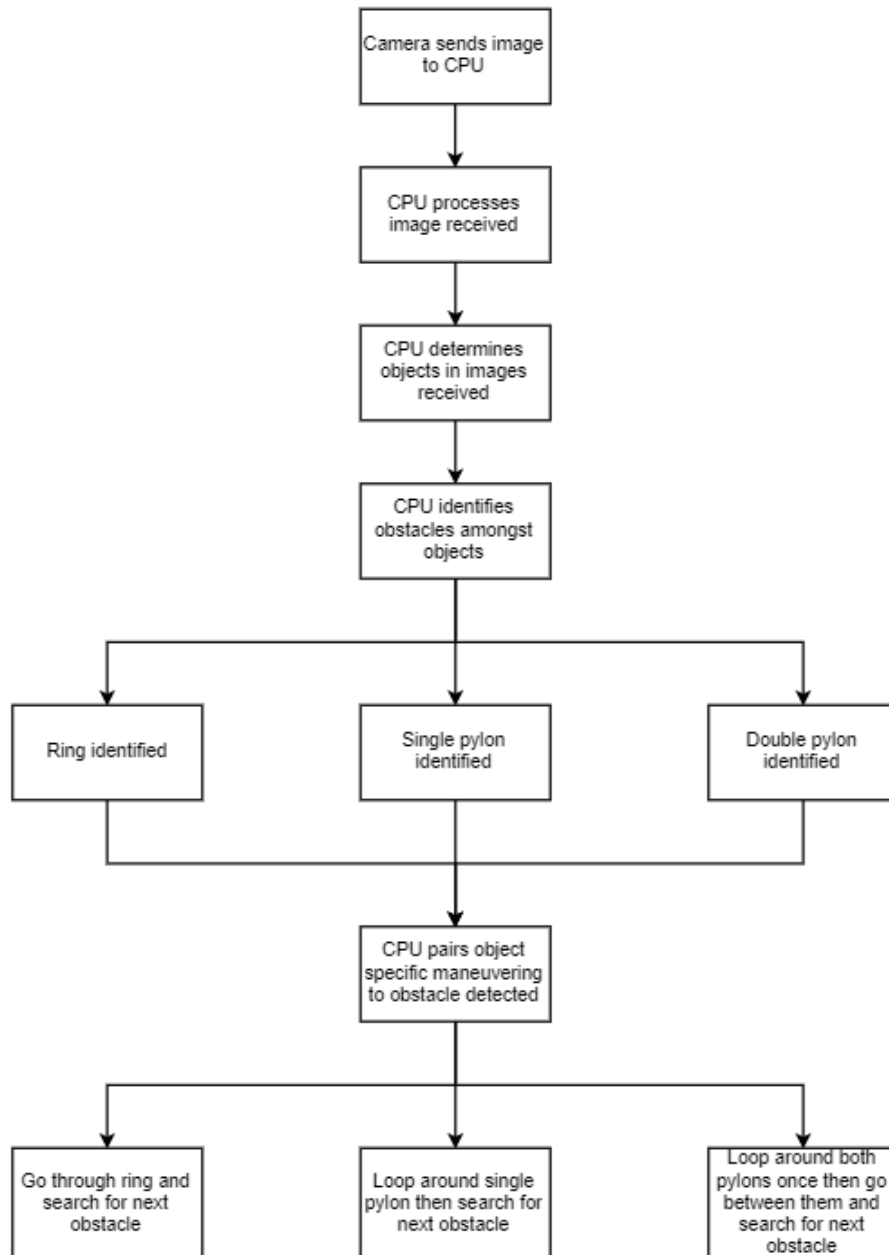


Figure 22: CPU obstacle identification and maneuver pairing (RL)

3.4.4 Power Distribution (RL)

Power supplied from the lithium polymer battery will be distributed to various components for the drone; it was originally planned to be done via the printed circuit board (PCB). However due to the high current of the motors/ESCs (30A), we were advised not to do this and purchased a power distribution board (PDB). The distribution of power from the battery to the various components through the PDB is shown in Figure 23.

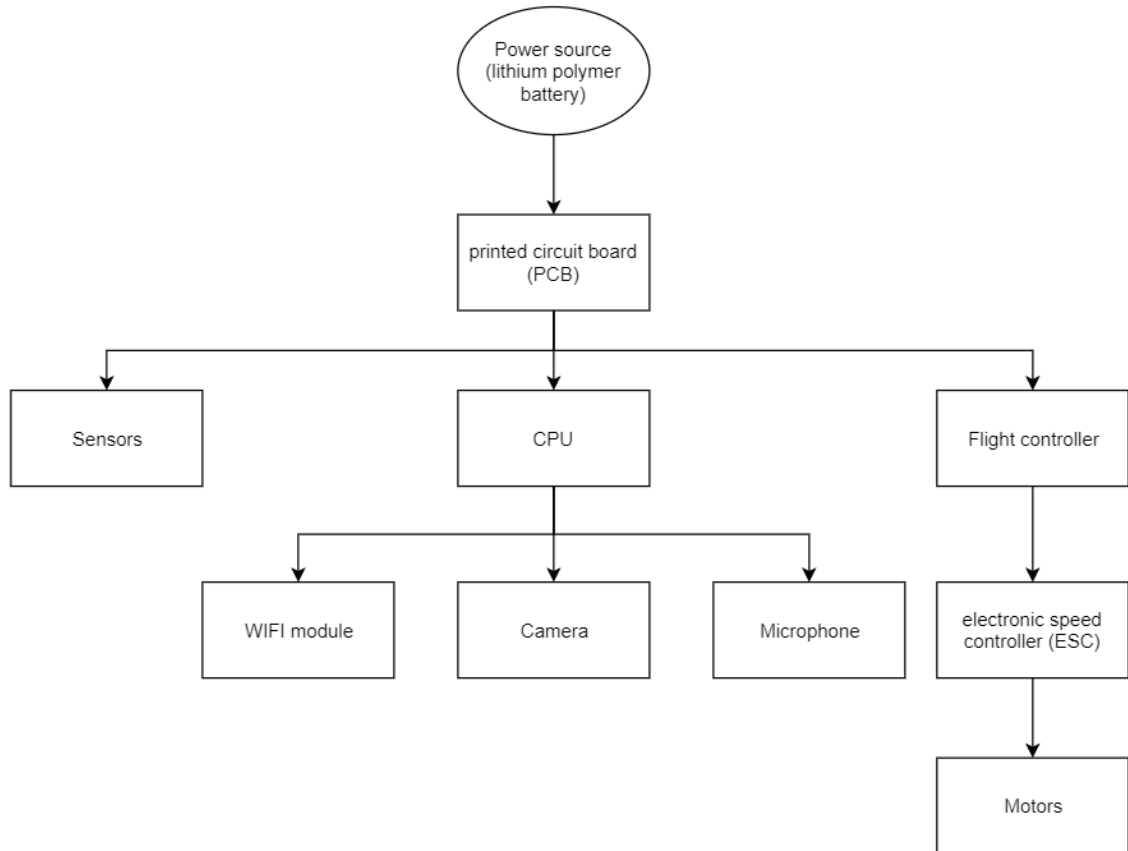


Figure 23: Power flow distribution chart (RL)

3.4.5 Acoustic Waypoint (RL)

Acoustic signals will be used to mark waypoints for the drone to detect and land for a brief period, then takeoff and continue on with the course. The drone system will be equipped with a microphone that will be used to detect the acoustic signal. The CPU of the system will receive the signal from the microphone and process it to determine whether a waypoint is located nearby or not. Once the CPU identifies the acoustic waypoint signal, it will then instruct the system to get within a 3 feet proximity of the signal and execute a landing sequence where the drone will land near the waypoint for 5 second, then takeoff and continue with the course. This process is demonstrated in Figure 24 below.

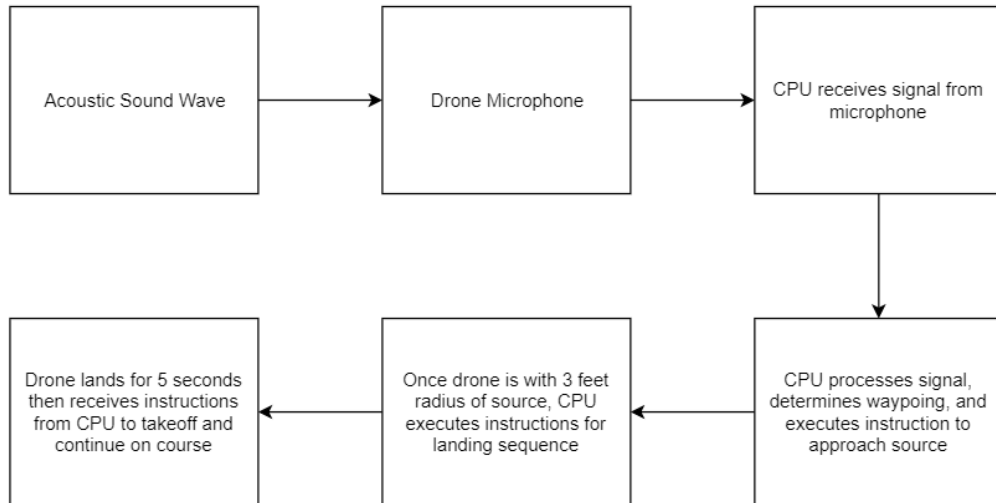


Figure 24: Acoustic signal processing of CPU and waypoint landing sequence execution (RL)

3.4.6 Component Connection (RL)

The components of the drone are to be interconnected with each other. The major components will be connected to the CPU, while some of the components will be connected partly to the CPU and partly to the PCB. the connection web of all the components of the system is demonstrated in Figure 25 below.

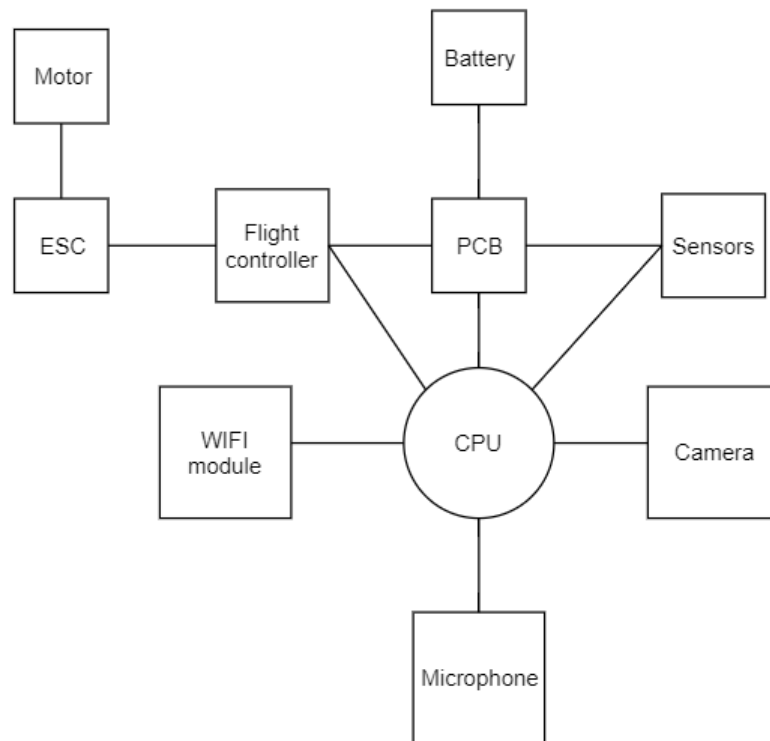


Figure 25: Component Connection Web (RL)

4.0 Related Standards, Regulations, & Realistic Design Constraints (RL)

4.1 Standards (RL)

It is vital to understand the constraints for the project as these can impact the choice of specific parts or lead to delays if the constraints are not met. Operating within limitations is a vital part of the engineering process, as you must meet certain criteria such as physical design limitations (size, weight, shape, etc.) and environmental limitations (use of non-toxic materials, rugged for outdoor environments). Design constraints can also be software related, such as a limitation to the use of a certain programming language. For this project, several limitations were set by our sponsor Lockheed Martin to ensure a baseline for all teams competing to make the most efficient autonomous drone.

In addition to this, there were many standards to consider for this project. There are also many government and city regulations to take into account when planning and constructing for the UAV. Law enforcement regulations become particularly important when it comes to the testing phase of the project as certain permissions need to be taken for outdoor testing. Permission from UCF authorities would also be required to conduct testing on school grounds. If the product is not built or found to not meet the required standards, obtaining such permissions may become difficult.

4.1.1 Dimensional Standards (RL)

The UAV is required to be within a certain bounded dimension limit. This limit was imposed due to the size of the obstacles used and the spacing clearance between the obstacles. This standard incorporates a safety concern as there may be a risk of collision with the obstacles which may be hazardous to an extent. This standard is imposed more so from the sponsors rather than state regulations. This standard needed to be closely observed and maintained as it reduces the amount of space available on the frame for the CPU and other key components that will be necessary for the UAV to operate.

4.1.2 Coding/Programming Standards (RL and HS)

While writing the code for the particular algorithm to be used for the system, several coding standards will need to be used. The code will need to be easily readable and understandable by other users who may be analyzing the code. As such, it will be important to choose an algorithm that will be compatible with all components to be used for the UAV, while still maintaining the coding standard guidelines. [5]

According to the article “Coding Standards and Guidelines”, the purpose of having coding standards are:

- To maintain a uniform and consistent appearance of codes written by various engineers
- To make the code easier to read and maintain, and reduce complexity
- To make error detection easier and help make the code reusable
- To increase programming efficiency and encourage global practices

Some programming standards mentioned in the article include:

1. Limited use of globals: this standard identifies the data types that can be declared as global
2. Standard headers for different modules: This standard emphasizes on the importance of having a standard format and information for headers of different modules. In general, the header should have the following:
 - Name of the module
 - Date of module creation
 - Author of the module
 - Modification history
 - Synopsis of the module and its functions
 - Functions supported in the module and respective I/O parameters
 - Global variables utilized by the module
3. Naming conventions for local variables, global variables, constants, and functions. This standard includes the following:
 - Use of meaningful and understandable names for variables
 - Local variables should be named with lowercase letters, whereas global variable names should start with capital letters
 - Avoid using digits in variable names
 - Function names should be written in lowercase letters
 - Function name should concisely describe the reason for its use
4. Indentation: Indentations are used to increase the readability of the code. Indentations may be of the following forms:
 - Having a space after a comma between two function arguments
 - Nested blocks need to be properly indented and spaced.
 - Proper indentation need to be utilized at the beginning and end of each block
 - All braces should start from a new line with the code following the end of the braces also starting from a new line

5. Error return values and exception handling conventions: Functions that encounter an error condition should either return 0 or 1 for simplifying the debugging.
6. Avoid coding styles too difficult to understand: Complex code makes maintenance and debugging difficult and expensive
7. Avoid using identifiers for multiple purposes: Using an identifier for multiple purposes may confuse the reader.
8. Code should be well documented: Proper commenting should be used to increase ease of understanding.
9. Length of functions should not be very large: Long functions are difficult to understand and should be broken into smaller ones for smaller tasks.
10. Avoid using GOTO statements: GOTO statements make the program unstructured and difficult to debug.

4.1.3 Video resolution standard (RL)

For the camera(s) to be used the EIA-1956 standard and RoHS compliance may be used to determine the accurate camera resolution required for this project. To be able to detect an object and be able to differentiate between a redundant object and an obstacle, the camera resolution needed to be good enough to an extent. Also, to be able to calculate the distance of the system from the obstacle, a clearer image received by the CPU from the camera will improve the calculation accuracy. Recent improvements in depth perception cameras may be useful to meet this standard as they typically tend to have good resolution.

4.1.4 IEEE Standards (RL)

IEEE P2025.2 standard for consumer drones: privacy and security; focuses on drones that are available in the market for consumer or commercial uses. This standard specifies requirements, systems, methods, testing and verification for consumer drones to preserve the privacy and security of people and properties within range of the drones (Definition adapted from the IEEE standards website). This standard however, does not affect this project as the drone being built will not be for any sort of consumer or commercial use, and thus will not be available in the market. [13]

4.1.5 UL 3030 – Standard for Safer Flights (RL)

The use of drones have become increasingly popular in the world today, and are readily used for accomplishing many tasks including performing tasks that would be otherwise time consuming and expensive without the use of drones. Drones are also used for activities that reduces risks and can be operated from a safe distance. With all the uses for drones, it is important for such devices to be

equipped with reliable batteries that pass all safety standards, safe mode capabilities, and a safe load, charger and battery coordination. The UL 3030 Standard for Unmanned Aircraft Systems, addresses electrical system requirements for UAVs operated by trained pilots. This new standard covers commercial and tactical applications (Definition adapted from the UL website) [A]. This standard is useful for the project as it is focused on ensuring the safety of the system's power supply and load coordination. [8]

4.1.6 IPC PCB Standards (RL)

Institute of Printed Circuits (IPC) standards are the electronics-industry-adopted standards for design, PCB manufacturing and electronic assembly. Every step of PCB design, production, and assembly is associated with an IPC standard (definition adopted from "History and Basics of IPC Standards" article by Nick Davis). According to the article, the first version of the IPC-A-600 standard, known as the "Acceptance of Printed Boards" was published in 1964. In 2008, to address the lead-free solder problem, IPC got together with the Electronic Components Association (ECA) and Joint Electron Device Engineering Council (JEDEC) to develop IPC-J-STD-075 standard known as the "Classification of Non-IC Electronic Component for Assembly Processes. [24]

For this project, the team needed to closely follow the IPC standards for the PCB to be used the electronic signals sent to the other major components, as the IPC standards mainly apply to printed circuit boards (PCB). Figure 26 shows a flow chart of the IPC PCB standards list. Following the standards properly will eliminate the risks of electrostatic discharge which can completely destroy a circuit.

The IPC standards typically incorporate:

- Product documentation
- Design specifications
- Specifications of materials to be used for the PCB
- Performance scope of the PCB
- Solder requirements for electronic assemblies
- Acceptability of electronic assemblies
- Requirements and acceptance for cable and wire assemblies
- Inspection and testing acceptability standard



IPC STANDARDS — EVERYTHING YOU NEED FROM START TO FINISH

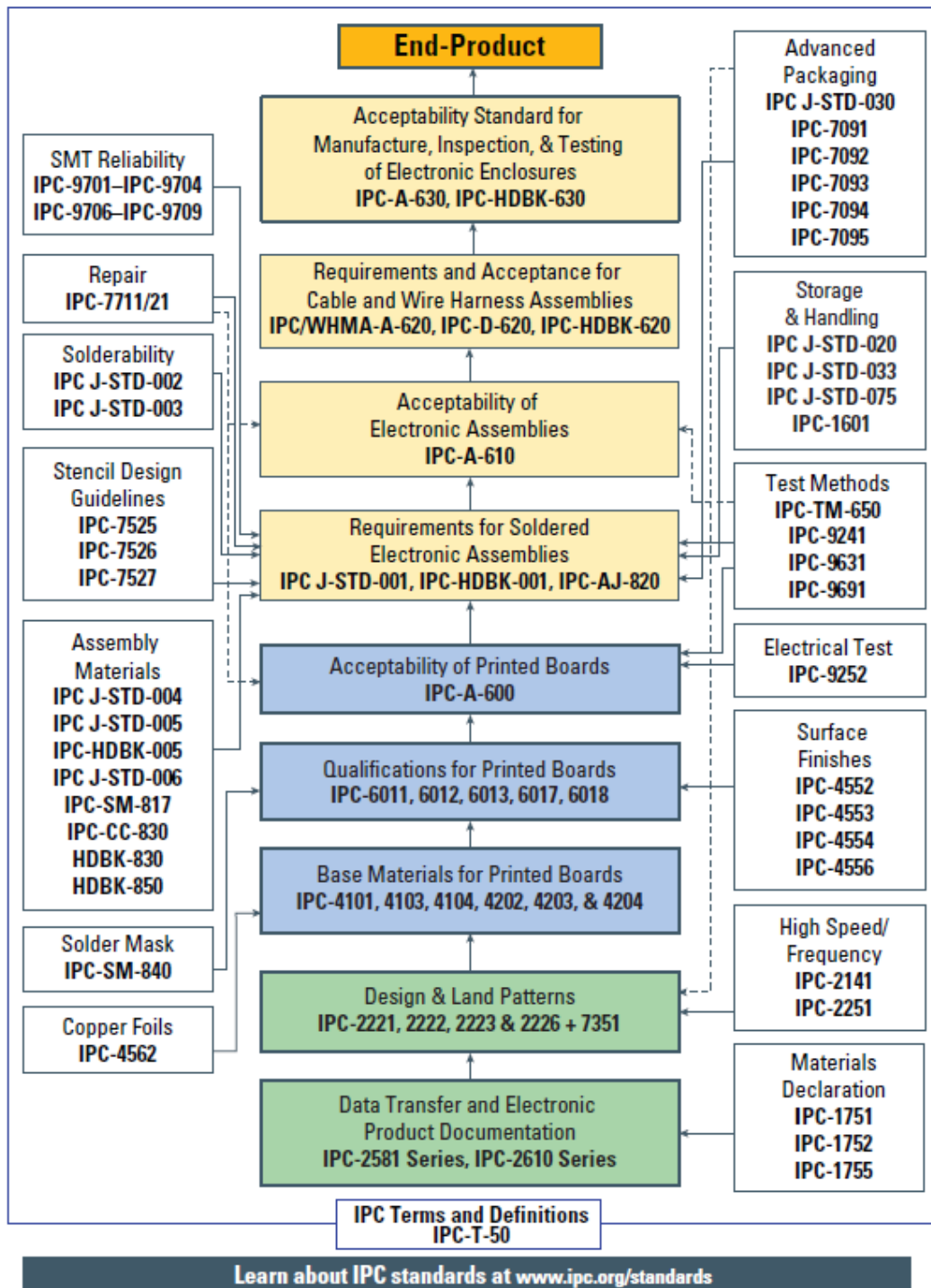


Figure 26: IPC Standards List, image courtesy of IPC Association [B]

4.1.7 IEC60950 (Relevant Power Supply Standards) (HS)

This standard is intended to ensure that products are insulated correctly in order to prevent serious injuries from occurring due to fire hazards, electrical shock, or high temperatures. A hazardous voltage according to this standard is any AC voltage that is larger than 42.2 volts peak or any DC voltage that is larger than 60 volts. These apply as long as the circuit is not a limited current circuit which is a circuit that guarantees that a hazardous current will not be reached even if there is a fault. This standard separates equipment into three different classes with each being isolated through different methods:

1. Class I: For class I devices, shock protection is performed by grounding to protective earth or using basic insulation material to separate live wire. The protective earth grounding must have a certain colored insulation (green, yellow, or clear) and must not contain a switch or fuse.
2. Class II: Class II devices, unlike Class I devices, do not require a protective earth ground and only requires that shock protection be performed by utilizing reinforced insulation (double insulation)
3. Class III: Class III devices must contain a secondary circuit known as a safety extra-low voltage (SELV) circuit. This circuit makes it impossible for dangerous currents to be produced by the equipment and the SELV circuit should be separated from primary circuits and other dangerous voltages by two other types of insulation or protection.

Hazardous voltages are stopped by utilizing insulation of which there are different types. The most important types of insulation specified in IEC 950 are the basic insulation, supplementary insulation or operational insulation.

1. Basic Insulation: Protects against shocking hazards by utilizing a single level of protection. There is no minimum thickness for this insulation type however, there must be a secondary protection layer such as a protective earth ground.
2. Supplementary Insulation: A layer of protection that is often utilized as a failsafe for basic insulation. This type of insulation must be at least 0.4 mm
3. Operational Insulation: This type of insulation is the minimum level of insulation needed for the equipment to work correctly and are not meant to provide any protection from fire or shock hazards.

Ultimately, IEC 950 outlines the basis for hazard protection for power supplies and demonstrates the importance of clearance as well as solid insulation in ensuring the safety of the user of the product. These standards are vital in protecting the consumer from short circuits and shocks which could in turn cause fires. The reason this standard was discussed was due to the importance of

power supply protections. This can be viewed in the relatively recent “Hoverboard” fires or Samsung Galaxy Note 7 fires that were caused by overheating power supplies and batteries. [7]

4.2 Drone Regulations (CJ)

This project will fall under Part 107 of Title 14 of the Code of Federal Regulations. [20] This is because our project is not being flown purely for hobby or recreational purposes and weighs less than 55 pounds. Under part 107, all drones over 0.55 lbs must be registered with the FAA. Drones can be registered online using the FAA DroneZone. In addition, drone pilots must also register and be certified by the FAA. This process includes passing a short test to demonstrate an understanding of the laws governing the use of drones.

Once a drone is registered and a pilot certified, a drone can only be flown without prior approval in uncontrolled airspace known as Class G airspace. While in Class G airspace, pilots are free to fly between 400 feet and ground level, assuming that all other regulations are followed. Flight in any other airspace requires approval beforehand. Controlled airspace typically exists around airports, stadiums, and areas of national importance, and one can ensure they are flying in uncontrolled airspace by looking at a map online.

In addition to staying below 400 feet above ground level in uncontrolled airspace, drone operators must comply with the following regulations:

- Pilots must do a preflight check to ensure the drone is safe for operation.
- Pilots must keep the aircraft in sight and must fly only during the day.
- Pilots must fly at or below 100 mph.
- Pilots must not fly over people other than themselves.
- Pilots must not operate a drone from a moving vehicle.

Most of these regulations will be easy to follow as the drone will not be flying fast far off from the ground. The major regulations that we must be careful to follow are the need to register the drone, to have a certified pilot, and to ensure that we are flying our drone in uncontrolled airspace.

Our drone is be able to be autonomously controlled, so we had to take precautions to ensure that the pilot will be able to take control of the drone at any time in order to comply with the above regulations. This will be implemented using two different switches that can throw the drone into our E-Stop mode or our manual mode. Additionally, we can implement an algorithm that can detect when the drone has lost communication with the ground station and force it to land or return.

The drone was primarily flown by a licensed member of our team (MAE division), and we also conducted some indoor tests.

4.3 Realistic Design Constraints (RL)

Constraints are conditions that place limitations on the designing of products. This section will focus on the different constraints that will affect the design of this project.

4.3.1 Economic Constraints (RL and RJ)

Economic constraints are constraints that place a financial constraint on the project. This project is sponsored by Lockheed Martin, and the customers of the project has a budgetary limit which places a restriction on the amount of finances that can be utilized towards the development of the project.

The project required the team to design and create a drone that is fully autonomous. To ensure the final product is designed to meet the expectations of the customers, a prototype of the drone was needed to be developed first before assembling the final finished product, and present it to the customers.

The team is provided with an initial funding of \$550 to develop a prototype of the autonomous drone. The customers (sponsors) have certain specifications that will need to be met for the final product. These conditions needed to be implemented into the prototype before being incorporated into the finished product.

Due to the limited funding of the prototype, certain components needed to be adopted into the final product to stay within the required budget of the customers. To be able to get the essential components for the drone, the drone kit was not considered to be used for the development of the prototype. Instead, the team decided to utilize a three dimensional printer to design the frame for the drone. This approach was more cost effective as the drone kit that was found to be most suitable for the project cost a little over \$200. Whereas, designing and printing our own frame would cost the team about \$10 per frame.

This approach allowed the team to utilize the funding to obtain more reliable and better quality parts for the drone, and was crucial to the successful completion of the project. The extra funding saved from creating our own frame left more room to get a better and more powerful CPU for the drone system. The team was able to utilize a better camera with depth perception capability, and be able to obtain a better battery with a higher charge capacity.

The team was allowed a fund of \$1100 to be put towards the finished product. If any component was to be reused from the prototype model on the final product, the cost for that component was deducted from the budget provided for the final finished product. This is an economical constraints as it takes funding away from the final product. As such, limitations are placed on certain added components that may be utilized to better the completed autonomous drone. Such

components may include sensors that may be incorporated to improve the drone's object detection and maneuvering capabilities, components to improve the drone system's reaction time, and so on.

4.3.2 Time Constraints (RL and RJ)

Time is a big factor to consider in the design and development of a product. Engineers are often faced with various deadlines that need to be met for successful completion of a project. In the case of this project, the time constraint is no different.

This project began in late September 2019 once we were assigned our groups, and will be due in April 2020 to fulfill our sponsor's completion deadline as well UCF's graduation requirements for the Senior Design 2 course. In addition to this late Spring deadline for the completion of the final project, we have milestones to complete in-between then. By December 2019, we must have a completed outline of our preliminary design to review with our project sponsor, and to submit as part of the requirements for the Senior Design 1 course. By March 2019, we hope to have a finalized design of our drone, and to use the remaining time for tuning and performance improvement.

Time has continued to work against our team since the very beginning we have taken on the project. The team was not assembled in a time consistent period which pushed back the progress of the team orientation. Another setback was the late introduction of certain project requirements that was specific to the customers expectations. These setbacks includes the incorporation of certain functional range limits such as angle of sight for the drone cameras, flight height limits, and so on; some of which are still yet to be clarified.

There were delays in getting access to the funding account. The team was not able to get access to funds until November 2019. This caused a delay in placing orders for parts that were required for the project. This delay in funding access will also cause further delays as the parts to be ordered will need time to be delivered, and there will be loss of time during shipping and handling for the parts.

Prototyping for the drone for the first phase was aimed to be completed by the end of January 2020. With the following next two phases, at the very least, are expected to be executed in one to two week intervals following each phase. This milestone leaves the team with very little time to make any error corrections and adjustments to be implemented on the prototype for the next testing phase.

The prototype testing is expected to be completed by March 2020 and is expected to be ready for presenting to the customer by no later than April 2020. These milestones and due dates do not leave much time for extensive testing, and all necessary updates and adjustments will need to be completed before the presentation due date of the project to the customers.

4.3.3 Environmental, Social, and Political constraints (RL and RJ)

Setting the right environments for the project is very important and places a considerable level of constraint on the drone's testing and demonstration conditions. When prototyping our initial design and developing our final design, we will have to perform several test flights to verify, debug, and improve the flight capabilities of the drone. We will also need to perform test flights to tune the object and waypoint recognition that will allow the drone to command itself autonomously.

Testing the drone in different environmental settings may significantly affect certain components of the system. For instance, if the drone prototype is being tested in a very noisy area, it may be difficult to test the microphones ability to detect the acoustic sound waves. Ensuring the right environmental setting is obtained is also important in the event the drone CPU is unable to differentiate between objects and begins to approach random objects. This may pose a level of danger as there may be risks of injury and damage to public property.

For environments involving the outdoors, effects of natural entities also needed to be considered. Such entities may include bright sunny days, rainy weather, strong wind gusts, wildlife (birds and other animals), pets that are not on a leash, and so on. To avoid such effects of nature, the emergency stop feature, or the manual override feature will need to be implemented into the drone system as a safety precaution.

We made sure to perform test flights in an environment suitable for drone flying. We cannot fly the drone over private property, and in the case of flying the drone on UCF property, pre-notified written approval from the university would need to be obtained before any outdoor testing can be conducted. The UCF police and relevant law enforcement authorities will also need to be notified, and written permission would need to be received before any outdoor testing can be performed. Moreover, we must make sure that we are not flying in controlled or restricted airspaces, as defined by the Federal Aviation Administration (FAA).

Socially, we needed to remain mindful of the perception that people may have towards flying drones. Primarily, individuals may feel uncomfortable being in an area with a flying drone, and may fear the risk of bodily injuries. Furthermore, since our drone has a camera and microphone, individuals may fear that they were being observed and recorded and may feel their privacy is being invaded. Therefore, we needed to remain mindful of these concerns when testing our drone. The best way to avoid these issues is to test in an open and secluded outdoor area, or indoors in a private area. Social perceptions affect the type of area that can be chosen to test the drone as it influences the type of environment to be used.

The use of unmanned drones have become increasingly popular, especially among government agencies. Law enforcement are known to use drones as speed traps, surveillance, pursuit of people observed to be suspicious, and many other tasks. Drones have also become popular for military use and are commonly utilized for a variety of tasks. Due to accidental events leading to loss of lives by military owned drones, the public's opinion towards drones are not the most favorable. Furthermore, the public's fear of government agencies using drones to invade privacy, has caused the public to gear against the idea for drones being allowed to fly freely. The team will need to keep the public's perspective attitude towards drones in mind while working on developing and testing the drone.

To overcome these challenges, we primarily tested the drone on private property owned by a teammate, and indoors.

4.3.4 Ethical, Health, and Safety Constraints (RL and RJ)

The development of an autonomous drone involves a certain ethical factor, especially if the drone is equipped with recording capability. As the developers of an autonomous drone with such capabilities, the team will need to be aware of the ethical limits of the project and design the drone with this particular constraint in mind. [10]

Our drone will have the ability to record pictures, videos, and audio as these are needed for autonomous flight capability. We must make sure to only use data that is required by our algorithms to process the flight control of the drone. For the images used as a template for the object recognition algorithms, we avoided capturing images, video, and audio of individuals that do not consent to being recorded. In the event that we record such an individual, we planned to blur or entirely remove faces and other personally identifying features.

The dataset we create for our object recognition algorithm only needs to include the obstacles that will be used in the course; therefore, sanitizing the images of personal information will not affect our ability to program our drone. We may need to store some live flight footage from the drone, but we will similarly sanitize recordings that we need to keep and delete recordings that we will not use.

Aside from the concern of invasion of privacy of unsuspecting pedestrians, which will also be put into consideration during the assessment of environmental constraints, other ethical factors will also need to be considered during the development of the project. One such concern is the reproduction of coding for the system. Any form of coding the team decides to utilize and incorporate into the project software design will need to satisfy the coding guidelines standards. These standards are mentioned in details in section 4.1.2. The need to follow this and other standards also fall in this ethical constraints section. As engineers aiming to develop a successful product, these standards will need to be closely maintained. As such, restrictions and limitations on both the design and the functionality will be imposed on the drone's systems.

The drone's electrical components is primarily be powered by a lithium polymer battery. This type of battery tends to explode if it is overcharged for an extended period of time. Due to the nature of this type of battery, certain precautions need to be taken when using this battery. If not properly handled, the battery may cause property damage and bodily injury to anyone who may be nearby. To ensure the battery if not overcharged, the charging process of the battery will be closely monitored and timed. If necessary, we planned to implement cooling systems to prevent overheating of the components. However, we found that this was not needed.

Since we are building a drone from scratch, and because it will be designed to work autonomously, we will create a fail-safe system to allow us to immediately kill power to the drone's motors on demand. The reason why this is important is to prevent the drone from injuring an individual and/or causing property damage. If we suspect that the drone is behaving erratically or it becomes completely uncontrollable, we will have the ability to bring the drone down to ground level almost immediately.

The drone will also be equipped with an option to take over manual control. The emergency stop command and the manual control command will be enabled through the ground computer. The drone will be equipped with a WiFi transmitter and receiver and will receive commands for emergency protocols using this system.

People have always been concerned about health concerns produced by radio signals, television signals, WiFi and radio transmissions, and several other entities that emit radiation through transmitted signals. Communication signals that will be transmitted between WiFi and radio devices used for this project is no exception this form of radiation emission. Although the levels of radiation that will be generated by the drones communication system will be extremely low to cause any type of long term health concern, the team will still need to be mindful of the possibility that a member of the public may not be comfortable near our project.

We also needed make sure that these communication systems do not interfere with already established systems that others may be utilizing. For example, we must ensure that our drone does not interfere with the WiFi networks when testing at home, UCF, or Lockheed Martin's facility, as it could prevent others from accessing their services.

Another significant health and safety concern relates to the material of our PCB. We have the option to have our PCB fabricated using lead, which is considered a toxic material. Though we would only need to handle the PCB when soldering components and attaching it to our drone, some people may not feel comfortable being around a device that has a potentially exposed lead PCB. Moreover,

California's Restrictions on the use of Certain Hazardous Substances (RoHS) in electronic devices prohibits lead from being used. [3] The primary focus of this regulation is to prevent the release of heavy metals into the environment once they are disposed of. Though Florida does not have a similar restriction on electronic devices, using a RoHS compliant PCB would be environmentally beneficial.

4.3.5 Manufacturability and Sustainability Constraints (RL and RJ)

This project required using many "off-the-shelf" solutions for various components. However, the team still needed to determine how to make all of the components work with each other and place all the components together on the frame. Initially the team's plan was to design small sub mounts, which would then be attached to a pre-manufactured frame for prototype drone. However, due to economic constraints, the team decided to use a three dimensional printer to design the frame from scratch.

Another reason for the team's decision was the dimensional restriction that was placed on the team by the sponsors. The sponsors required the drone to be no more than 1.5 ft x 1.5 ft x 1.5 ft. because of this size constraint, the team was facing some challenges in obtaining the correct size frame on the market shelves.

When creating our own frame, we needed to make sure the material of the frame is strong enough to support the weight of all the components and avoid being damaged by the forces in flight. We would also like the drone to be strong enough to avoid being damaged from an occasional fall or collision.

The other major component we needed to have manufactured is the printed circuit board (PCB) which we originally planned to also use as a voltage regulator for the various onboard components. We need to create a design that will be small enough to be easily installed to our frame, but large enough to easily solder the electrical components as needed. Several laboratory simulation and testing for the various required circuits will need to be performed before having them implemented on to the final PCB design. The voltage regulator circuit for the PCB will need to be thoroughly tested to ensure it is able to regulate and distribute the correct amount of current and voltage required to supply power from the power source to the various components.

The center of the drone's autonomous system, and perhaps the most crucial component of the entire system that brings all the other components together, is the CPU. The CPU will need to be able to interface with all of the components and process the data that it receives from each to make logical decisions. It acts as the brain of the drone and will maneuver the system through the course. It will also be able to communicate with the ground computer and transmit and receive data and commands while it navigates the drone through the course. Constraints involving the CPU includes the compatibility of other components to be used in

junction with the CPU, such as cameras, microphones, WIFI modules, and sensors.

The cameras for the drone will be the eyes for the system. It will help the system locate objects and allow the processor to process the images received and identify obstacles the system needs to approach and perform various preprogrammed maneuvering sequences. The will also help the system determine the distance of the drone from an obstacle, and estimate the time of arrival to the obstacle. As such, the camera will need to be equipped with depth perception feature and it will need to be compatible with the CPU. However, the bigger constraint may be the ability for the camera and the CPU to be able to identify the type of obstacle the camera detects.

Other constraints involve the sensor's ability to detect acoustic waypoints that will be strategically placed along the flight path to mark temporary landing zones, and approaching dangers set by the "mine" team whose goal is to take down our drone and cause it to crash.

Overall, these constraints, along with the other constraints mentioned in the subsections above, placed limitations of various types on the project. They greatly affected the designing process of the drone, and the team needed to take every constraint into consideration to maintain the quality of the final product and ensure successful completion of the project.

5.0 Project Hardware Design Details

5.1 Initial Design Architecture (HS)

Our initial design architecture involves two main components: the CPU and the flight controller. We planned to connect digital sensors to the CPU and write a program to analyze the sensor data. Once the sensor data is analyzed, and a direction for the movement of the drone is determined, then the CPU will send a signal to the flight controller. The flight controller can then take this input data and use it to directly control the motors, also accounting for the sensors on the flight controller such as the barometer, accelerometer, magnetometer, gyroscope in order to provide smooth, level, and stable flight. An initial outline for our architecture can be seen below in Figure 27.

The power distribution board (PDB) connects to the battery and regulates the voltage for the various components. The PDB has several output wires, including full voltage to the electronic speed controllers and 5V to several of the electronic components such as the flight controller and Jetson Nano. Without the power distribution board, we would not be able to individually control the voltages going to the respective components.

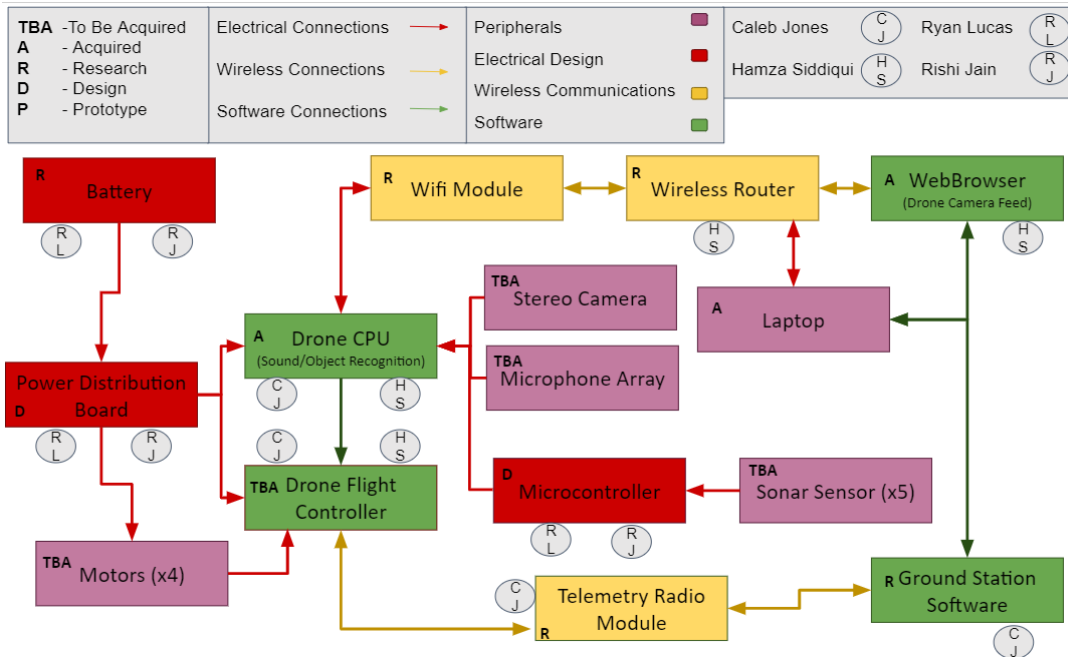


Figure 27: Initial Design Architecture for autonomous drone (HS)

5.1.1 Power System (RJ)

The power system begins with the battery that we connect to the drone. The battery will be higher voltage than what most of the components required, except for the motors. However, we will be able to step down the voltage using one or more voltage regulators depending on the exact voltage requirements for each component. Between the battery and the voltage regulator we planned to include an emergency switch, which will implement our “E-STOP” feature where in the case of an emergency, the drone is immediately stopped. This switch, when open, would essentially disconnect the battery to the entire drone and turn off all components, including the motors. However, we found utilizing a software command to be more convenient.

Our drone had multiple components that need to be powered in order to implement flight and autonomous functionality. These components include: the electronic speed controllers, the motors, the CPU, the camera, and the microphone.

The electronic speed controllers serve as throttle for the motors, controlling the flow of current into the motors, which determine operating speed. The specification for the electronic speed controller we selected is that it is directly connected to a battery ranging from 2S to 6S. Each lithium polymer (LiPo) battery cell outputs 3.7 V, and the S refers to the number of cells a battery has. Therefore, our selected electronic speed controller can directly handle a voltage range of 7.4 V to 22.2 V. We have selected a 4S (14.8V) battery; therefore, we do not need to implement an intermediary solution to power the electronic speed controllers and motors.

However, the chosen CPU (Nvidia Jetson Nano) cannot accept such a large range of voltages. From the specification sheet, the Jetson takes a 5V input at 2-4 A. We need to create a voltage converter in order to power the CPU. The amperage specification depends on the operating mode of the CPU, and only includes the main components on the board; it does not include extra peripherals that will also be powered, such as USB devices.

The typical operating mode of the Jetson only uses 2 A, while utilizing advanced features will bring it up to 4 A. The camera and microphone that we have chosen have a USB interface, and that is how we plan to interface between the CPU and those sensors. USB 3.0 specifications state that each port supports up to 0.9A, therefore, we will consider that we need an extra 2A when designing our DC/DC power converter. We do not expect to utilize the 4A mode on the CPU, nor do we expect our USB devices to each draw the maximum amperage provided by the USB ports. Therefore, designing a 14.8V to 5 V supporting up to 6A will be sufficient to power our CPU and sensors.

The Readytosky Pixhawk flight controller that we selected also requires a 5V input with up to 3 A capability, which means we should be able to use the same DC/DC converter we designed for our CPU without any additional design.

An important consideration when designing our DC/DC power converter is the range of voltages that the battery will output at a given time. The 3.7 V per cell figure is considered a battery's 'nominal' voltage, which is its resting voltage. However, the actual voltage at any given time may be higher or lower. LiPo batteries are fully charged at 4.2 V per cell, and should not be discharged below 3.0 V per cell, as it can severely damage the battery. Therefore, for our original plan for DC/DC converter utilizing a 4S battery would need a 12 V to 16.8 V at up to 6 A.

We initially utilized Texas Instruments' Webench Power Designer to input these desired specifications and select a design that would fit our requirements. When selecting a design, our main concerns are with power efficiency, bill of material (BOM) count, and BOM cost. Power efficiency is an important consideration because we would like to maximize the flight time of our drone.

Bill of material count refers to the number of components required to implement the DC/DC power converter circuit. We would like to minimize this number because it would aid in assembling the printed circuit board (PCB), particularly when we attach the various circuitry components. A lower BOM count makes the PCB easier to assemble, and reduces the chances of failure of the component. Lastly, we need to consider BOM cost. Since we are on a strict budget, we seek to minimize our costs when designing this PCB. Minimizing our PCB cost also gives us the flexibility to order a spare PCB, in case we damage our original board during assembly or testing.

The design we felt that best fits our requirements is the Texas Instruments' TPS56637 circuit. A schematic of the circuit is shown below in Figure 28 below.

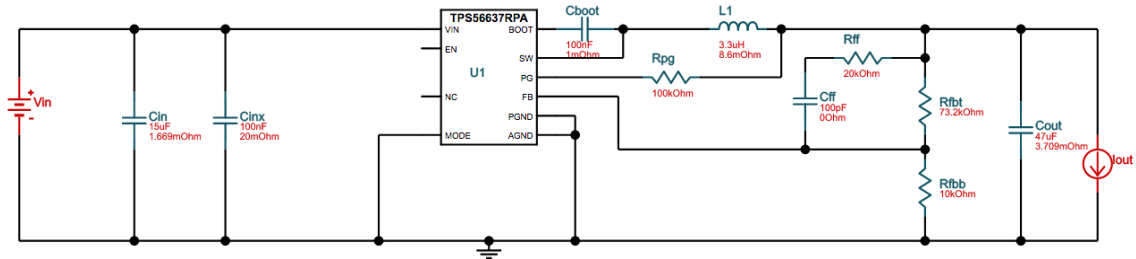


Figure 28: Texas Instruments TPS56637 design, courtesy of Texas Instruments [C]

The TPS56637 has an average efficiency of 93.9%, a BOM count of 11, and a BOM cost of \$9.23 making it an excellent option for our project.

The power efficiency of the circuit fluctuates based on the input voltage and the output current draw. Figure 29 below shows the efficiency of the circuit as functions of input voltage and output current. Our power efficiency will lie in the region between the red (12V) and gray (19V) lines, as those will be the operating conditions for our battery as to not damage it.

Efficiency vs Output Current $V_{OUT} = 5V$

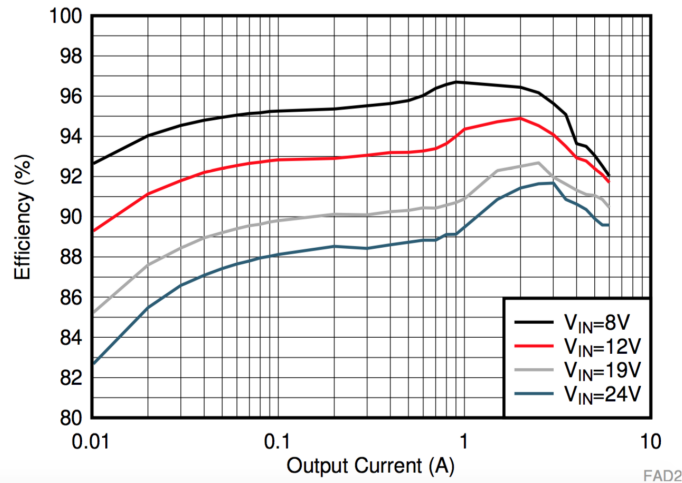


Figure 29: TPS56637 Power Efficiency, courtesy of Texas Instruments

We also utilized Autodesk EAGLE to design our PCB. Though we already have the schematic, we will need to determine the placement of circuitry components ourselves and design the PCB so that it will be small enough to fit securely on our drone, but also be large enough for us comfortably work on. Our PCB will be

using surface mount components, rather than through-hole components, as they are cheaper and faster to install.

The free version of EAGLE we are utilizing allows for an 80 cm² design with two layers of wiring, which is more than adequate for our purposes. We will attempt to minimize the distance of wire traces between the two components and use 45° bends in order to reduce electrical interference in our circuit.

By default, the Jetson Nano is powered by a micro-USB connector. However, the micro-USB only supports 2 A @ 5V, which would likely not be able to support all of our components. The Jetson Nano also includes a 5.5 mm/2.1 mm barrel jack connector which can be used to power the board at a higher amperage, which will be needed for our application. Our PCB planned to include soldering pads for the output voltage, and we will solder in a wire connected attached to a barrel jack connector so that we can power the CPU. The barrel jack cable will cost \$2.42. The components from our BOM, as well as the barrel jack cable, are available from Mouser.

EAGLE has the capability to export our board file into a Gerber file format, which is the standard utilized by the PCB fabrication industry. The manufacturer of PCB that we will use is JLCPCB as they are able to manufacture the board within 1-2 days and offer reasonably priced express shipping options. JLCPCB has a minimum order size of 5 pieces, however, this is suitable as it gives us extra circuit boards to use as spares in case we damage any. The price of an individual PCB with a lead-free finish is \$7.30.

The total cost for our initial PCB design utilizing DC/DC power converter will include the BOM for the circuit, the barrel jack cable, and the fabrication price of the board for a grand total of \$18.95.

When we receive the product and complete the installation of all required components, we planned test the output voltage using an oscilloscope. The reason for this is to ensure that the board is working as expected, and providing the voltages necessary to power our components. Since the components we have selected are expensive, we want to ensure that we do not damage anything. However since we were advised to buy a PDB specifically designed for drones, we elected to power all of our components using that voltage regulator instead of splitting it between our PCB.

5.1.2 Wireless Communication System

Another required system for our drone is its communication system. We would like to be able to control the drone manually if desired, and we must be able to view a live video feed from the drone. We connected a wireless transmitter to the drone CPU, as that is where we will be processing all images and audio. Taking the feed from our written program was helpful in debugging and analyzing the

performance of our drone, while also fulfilling the requirements set out by our project sponsor.

With the wireless transmitter, the drone can connect to a router that will also be connected to a laptop. The laptop will be able to receive the video feed through the on screen user interface. Using WiFi will be a suitable hardware solution for this communication system as it will have enough range to cover the obstacle course during the competition at Lockheed Martin's facility, if it occurred.

5.1.3 Interfacing with Sensors (CJ)

Each sensor must be able to interface with the Jetson Nano. The Nano will use the information from the sensors in order to determine where to go and to avoid obstacles. This section will be used to ensure that each of the sensors will be able to interface with the Jetson Nano Development Board. Additionally, we will be using a microcontroller to interface our multidirectional ultrasonic sensors that will be used to determine how close the drone is to a nearby object.

5.1.3.1 USB

There are a total of four USB 3.0 ports on the Jetson Nano Dev Board. These ports interface with the Intel RealSense D435 camera, the Geekworm Jetson Nano WiFi module, the ReSpeaker MicArray, and the custom PCB. The USB ports may require extra power depending on the connected peripheral, so we have accounted for that extra amperage in our initial PCB design.

5.1.3.2 UART

There is a single UART port on the Jetson Nano pins. This means that only a single device will be able to use this, and we have two devices that could use it. First is the TeraRanger One distance sensor, and the second is the PixHawk. Since communication with the PixHawk is so vital and the TeraRanger has other ways to communicate, the PixHawk will be chosen to use the port. Additionally, we will be using UART to interface our microcontroller with the CPU.

5.1.3.3 I2C

Since the TeraRanger One is unable to use UART communication, the sensor will be able to connect via I2C. This will require a special adapter for the sensor, but it is fairly inexpensive.

5.1.4 Microcontroller Architecture (RJ)

We planned to attach five (later reduced to four, removing the bottom sensor) HC-SR04 ultrasonic distance sensors to our drone as "bumpers" so that we can the drone's proximity to other objects during flight, as seen below in Figure 30. While we could connect the sensors directly to our Jetson Nano, it is preferable to use a microcontroller to manage those sensors as reduces the complexity of

the entire system by dividing it into layers. Then, the microcontroller can interface directly with the CPU to give position data for flight decisions.

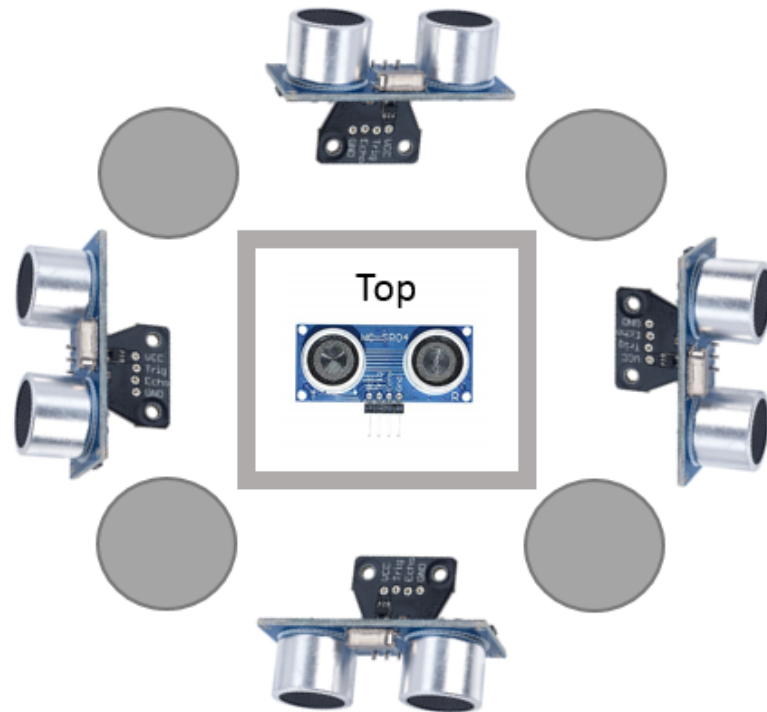


Figure 30: Top View of Five Ultrasonic Sensors Attached to Drone

For our five sensors, we only need one digital pin to send the trigger signal, but five additional (digital or PWM) pins to receive the echo signal from each individual sensor. We considered two microcontroller architectures: the Texas Instruments MSP430 and the Arduino Uno Rev3. While either architecture would be suitable for our purposes, we chose the Arduino architecture because there was more documentation and projects posted online that we could use as a reference.

We built our own PCB, because we do not require all the components on the development board. Building our own PCB allows us to minimize our form factor by combining DC/DC converters and the microcontroller chip, as well as fulfilling our electrical and computer engineering senior design requirements.

The Arduino platform has several chips that we can use, with the main difference being the number of pins. We considered the ATmega256 and ATmega168. The ATmega256 has 54 digital input/output pins, while the ATmega168 only has 14 digital input/output pins. While the cost difference is negligible, we determined that using the 14 pin chip would make the soldering the components significantly easier. Figure 31 below shows the chip and DC/DC converter on a single PCB.

We only need 6 pins from the ATmega168 (1 pin for the trigger, 5 pins for the echo). We are planning an 8 pin header, where the other 2 pins will be connected to VCC and ground. We have also added a status LED to ensure that power is flowing through the circuit.

We were able flash the processor from the Arduino Uno Rev3 development kit that we will use to debug and test the code before ordering the PCB using the ICSP interface. This allowed us to verify functionality of our code, and make any last minute changes to the schematic if needed.

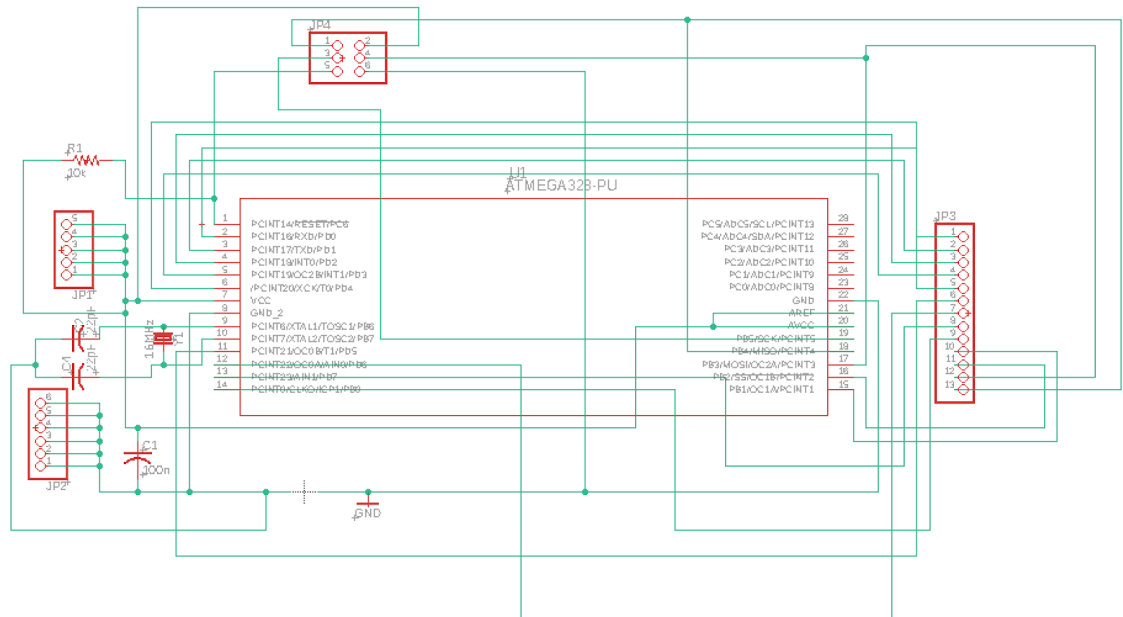


Figure 31: Schematic including Atmega328pu chip with LED indicator

A major components required to get the Jetson Nano to communicate with the PCB was a logic level shifter due to the fact that the Jetson Nano uses a 3.3V bus, while the PCB uses a 5V bus. Without this logic level shifter, the signals may not be registered. We originally intended to implement this into our PCB, however due to time constraints caused by the lengthy university ordering process and delays caused by the pandemic, we elected to use a standalone module separate from the PCB.

5.1.5 Nano-PixHawk Interface (CJ)

We originally planned the Nano-PixHawk interface through a UART interface. According to the ArduPilot website, the UART TX, RX, and GND pins from the Jetson Nano can be connected to the TELEM2 port on the PixHawk. Once the pins are connected, the Baud rate on the PixHawk serial port two must be set to 921600, and the serial protocol on the port set to 1, allowing MAVROS communication [2]. However, for convenience, we elected to utilize the USB ports on both devices to communicate between them.

5.2 Subsystems (RL)

An electronic system typically incorporates several smaller systems working together; a network of systems. These smaller systems are normally referred to as subsystems. Each subsystem offers its unique contributions through its specific characteristics, and these systems communicate with each other to some level to enable the entire network of systems to function as a whole.

5.2.1 Power system (RL)

The power system consists of the battery, the power distribution board (PDB), the sensor PCB, flight controller, electronic speed controller (ESC) (which is connected to the motors), and the CPU system. The PDB was originally to be developed by the ECE students of the team, however we were advised to buy a drone specific one as to not damage any of our expensive parts. Due to the various port requirements for all the different components that will need to be connected to the PDB, it will need to be specifically designed to hold and support all the connectors. Since each component will have specific voltage requirements, several voltage regulators will be incorporated into its circuit configurations.

The battery will be connected to the PDB via an XT60 connector. The PDB will distribute power to the various components, either directly to the individual component, or to the system that the component is connected to and is a part of. The battery's connector port is an XT60 type, as such, the PDB will need to have at least one XT60 type port [11]. With that, we were able to solder connections from the 5V pads to all of our necessary devices.

For this project, the team will use four ESCs, each of which had a brushless motor connected to it. The ESCs were be connected to the PDB for this project, and much like the connection for the flight controller, the ESC can also be connected in one of two ways. The first way to connect the Esc to the PDB is to solder the terminals of the ESC to the pad ports of the PDB. Unlike the six wires used to connect the flight controller, the ESC would only have two wires, and hence soldering the contacts will not be difficult. However, soldering the terminals made the connection permanent, unless the wires were re-soldered to detach.

To make the power terminals easier to attach and detach as needed, a second method could be utilized through the use of bullet connector ports. The bullet connectors can be soldered to both the PDB pad ports, and the ESC power terminals. After that, the connectors can be connected and disconnected as needed.

To connect the motors to the ESCs, the same modes for connecting the ESC to the PDB applies. There will be three wires from each ESC to connect to each of the motors. The orientation of the wires is not of significant importance. If the wires are connected a certain way, the motor will rotate in one direction. To

change the direction of rotation of the motor, any two of the three wires connected to the motor from the ESC can be swapped. Just like the ESC to PDB connection, the wires connecting the motor and the ESC can be soldered together, making it a permanent connection. However, similar to the ESC power connection, bullet connectors can also be utilized to connect the wires of the motor, making it easy for attaching and detaching the motors as needed. Figure 32 below shows the schematic for the hardware configuration of the PDB to the ESCs and flight controller.

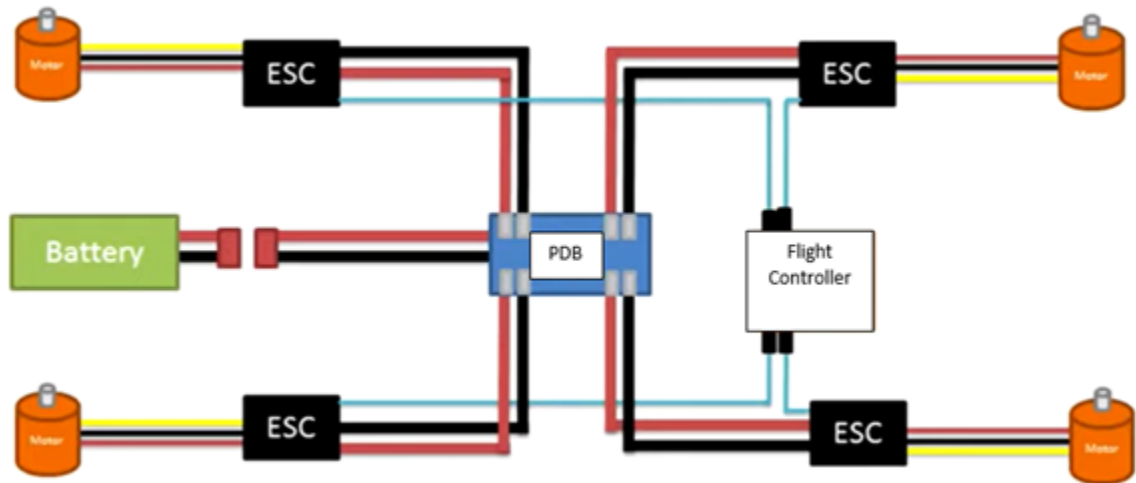


Figure 32: Power distribution schematic for battery to PDB to ESC and motors, courtesy of Painless360 [D]

The PDB will be connected to the sensor PCB with wires soldered to the power port terminals of the board. Supply voltage for the PCB will be regulated by the voltage regulator circuit on the PDB and will supply 5 volts to the sensor PCB. Connectors will not be considered for connecting the PCB to save room on the PDB, as too many connectors may take up space on the board and add unnecessary weight.

The PDB was also be connected to the Jetson Nano via a barrel jack whose wire was soldered to the PDB. The voltage to be delivered through the barrel jack will be 5 volts. The camera, microphone, and the Wi-Fi module will be connected to the CPU's USB ports and will receive power from those ports. As such, these components will not need to be connected to the PDB for power. The necessary voltage and current will be regulated and supplied to these components by the CPU.

5.2.2 Signal Processing and Data Transfer System (RL)

The CPU will be connected to all the other components of the drone either directly or indirectly through other components. It was to process all the signals it receives from the input devices and components and transmit signals to various output devices and components. The CPU will also receive data from the flight

controller regarding battery charge levels which the flight controller will be monitoring.

The CPU would receive input data from the camera, the microphone, the sensor PCB, and from the ground computer through the WIFI module. The CPU can transmit data to the flight controller, which will transit data to the ESC, which then transmits data to the motors. The CPU will also transmit data to the ground computer through the WIFI module.

The camera (Intel RealSense D400) was to be connected to one of the CPU's USB ports. Images will be captured by the camera and sent to the CPU. The CPU was to receive the images and process them for objects. It can then distinguish the objects that are not part of the background of the images. Once the objects are distinguished, the CPU will identify if the objects are one of the three types of obstacles (ring, single pylon, or double pylon) required to approach using the image recognition coding. The CPU will then determine the distance of the obstacle and transmit that data to the ground station through the WIFI module.

The CPU is also able to determine an estimated time of arrival to the obstacle and the confidence level and transmit that data to the ground station as well. It repeats these steps for every object and obstacle, and place a marker on each obstacle, in the form of a red "X" symbol and map the course the drone would take and send the data to the ground station.

Using SLAM the CPU was originally intended make a record of the course layout to enable the system to keep track of the flight path and create a map of the course. The CPU would later use this data to return back to the starting point. All image data the CPU receives from the camera could then be transmitted to the ground station where it can be live streamed. However, due to a change in requirements, this was no longer necessary.

The CPU is also able receive audio signals from the microphone. The microphone is connected to the CPU through one of its USB ports. Once the CPU receives an audio signal, it filters the signal, using a bandpass filter incorporated into the coding. The filter will allow frequencies of 0.5kHz to 1kHz to pass, while blocking noise of other frequencies. The CPU can then use this filtered data to detect acoustic sound wave, which will be used to mark waypoints, and send the data to the ground station for logging and mapping.

The CPU will also receive signals from the sensor PCB. The Sensor PCB will be connected to the CPU through UART. Every time the sensors detect an object, it can then send an interrupt to the CPU. The CPU will then use the signal and the image data it receives from the camera, and process a course of action to be taken, which will be predetermined by the system's coding.

If the signal received is processed and found to be relevant to an obstacle, waypoint, or mines set by the mine team, the CPU transmits the data to the ground station for debugging purposes.

The CPU send signals to the flight controller for drone maneuvering instructions. The flight controller will be connected to the UART port of the Jetson Nano. Each time the CPU processes a signal received from the camera, and determines an obstacle to be detected, it send signals to the flight controller to move closer to the obstacle and perform the preprogrammed maneuvering sequence specific to the obstacle detected.

The CPU also send signals to the flight controller for landing sequences in the event it detects a signal from the microphone that is processed to be from an acoustic waypoint. The flight controller will also receive instructions for evasive maneuvering from the CPU if the CPU receives signals from the sensors due to mines. The CPU will constantly be sending signals to the flight controller for all directional movements.

5.2.3 Wireless and Ground Station System (RL)

The CPU sends all data it processes to the ground station through the WIFI module. The WIFI module will be connected to one of the Jetson Nano's USB ports. Communication between the ground station and the drone's system will be done through the WIFI module. The ground station records all data it receives from the drone system and stores it for viewing and flight logging.

The live video feed is viewable from the ground station and the flight path and mapping will be recorded by it as the drone progresses through the course. The ground station will also be used to transmit the emergency stop signal to the drone system, and the CPU transmits a signal to the flight controller to perform the emergency landing sequence.

The ground station will receive measured flight data such as distance to an obstacle the drone is approaching, the time the drone will arrive to the required proximity of the target, the confidence level of the approach, as well as the marker to be placed on the obstacle detected. The ground station was to place that marker on the course map being created. The ground station will also place other markers on the map such as markers for the starting point of the course, any waypoints detected along the course, and any mines detected. The ground station then communicates this data with the drone system to enable it to keep track of the path taken and to find its way back to the starting point once the course is completed.

5.2.4 Drone Telemetry System (HS)

The telemetry subsystem is to be utilized solely for the manual control mode for the drone. It was originally planned to consist of the laptop for the ground control station, a ground station software application such as QGroundControl, a gamepad such as an xbox 360 game controller, a telemetry radio module for the laptop, and a telemetry radio module for the Pixhawk PX4 flight controller itself. However, during testing, we found that it was easier to SSH into the drone to directly interface with the computer and to run our scripts.

6.0 Project Software Design Details

This section details the design of the software for the DOMINANCE project. The software is broken up into two sections: drone software and ground station software. The purpose of the drone software is to enable the drone to fly autonomously using the various sensors onboard the drone. The drone is also responsible for receiving commands from the ground station and for sending video feed overlaid with detected obstacle data information. The purpose of the ground station is to send commands to the drone and to display the sent video feed sent from the drone. This section describes both groups of software and their communication. (CJ)

6.1 Drone Software

6.1.1 Drone Software Overview (CJ)

The drone software is broken up into two sections: the drone computer software and the flight controller software. The drone computer software will be divided into groups called nodes, and the inputs and outputs of these nodes will be managed by the Robot Operating System (ROS). We used some open source nodes to help us interface with our sensors and our flight control board, and we will create some custom nodes to perform tasks such as acting as the primary controller for the system. The flight controller software selected was Ardupilot. The main purpose of the Ardupilot is to carry out movement commands from the drone computer and to send movement data to the drone computer. A diagram showing the flow of data between components of the drone software can be seen in Figure 33 below.

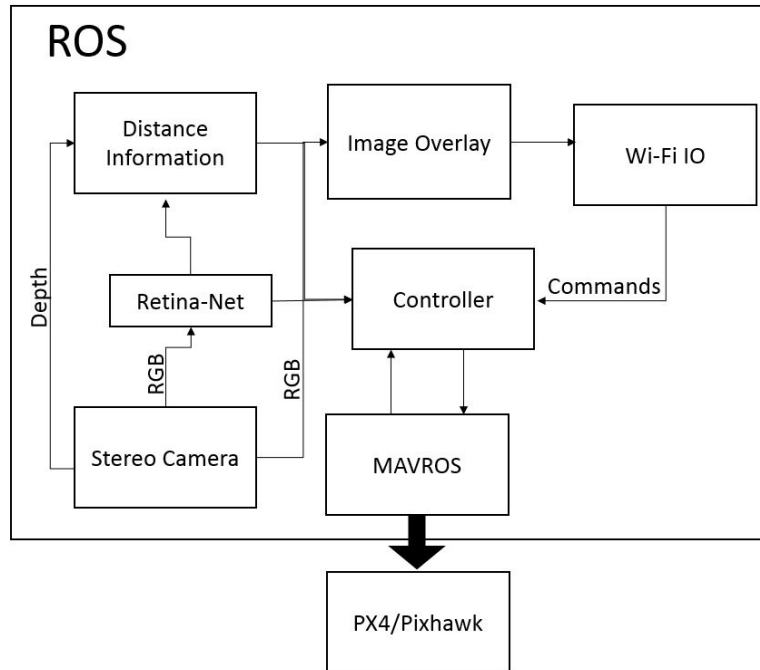


Figure 33: Drone Software Model (HS)

6.1.2 Drone Computer OS and Framework (CJ)

6.1.2.1 Drone Computer OS

The drone computer OS is Linux4Tegra(L4T). This Ubuntu based operating system is provided by NVIDIA and is specifically designed to run on the Jetson series hardware. Along with L4T, NVIDIA provides JetPack, a group of libraries and APIs that are designed to run on L4T and the Jetson hardware. These include TensorRT that can be used to improve object detection and the OpenCV library. In addition, there are many computer vision libraries available for use with our GNU-Linux based operating system. With these benefits in mind, we see no need to choose a different operating system.

6.1.2.2 Robotic Operating System (ROS) Framework

On our drone computer, we chose to use ROS Melodic, which is the only ROS version compatible with our L4T OS. ROS facilitates communication between different sections of code known as nodes. When a node wants to send data to other nodes, the node acts as a publisher and sends a message to a topic. The message, which contains a data structure, is then sent to any nodes that are subscribers to this specific topic. The data received can then be used by the receiving node.

In addition to the ability to publish or subscribe to topics, the ROS also enables a node to send a request to a node and wait for a reply. This effectively acts as a way for nodes to call a remote procedure. A diagram of how message passing and service invocation works can be seen in Figure 34 below.

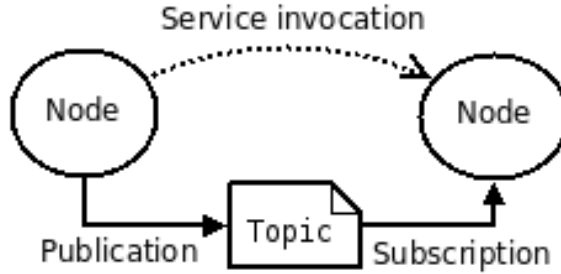


Figure 34: ROS Diagram (from ROS Wiki) [22]

6.1.3 Drone Software Nodes (CJ)

In this section, all the ROS nodes located on the drone computer are detailed. Where there are already developed open-source nodes that can be used, we describe what they are and how they need to be configured. Where there is no available node that can be used for a section, we detail how the node can be designed and what libraries we plan on using.

6.1.3.1 Camera Node

Intel provides a node called `realsense-ros` that is used to produce a ROS node wrapper for the D400 series camera. [12][19] This will enable the camera to automatically send messages to a list of topics once the node is started. These topics can then be subscribed to by the dependent nodes. The main topics that our software will be using will be the raw color image for image recognition and the raw depth image for calculating distances to objects.

The provided Intel node is not created for ROS Melodic, and there are no supported versions of the L4T OS for the Jetson Nano that can support other versions of the ROS. [21] We expected that this may cause some integration issues and that we may have to build the node code ourselves on our Jetson Nano. However, the node worked on our system out of the box.

6.1.3.2 RetinaNet

The RetinaNet node processes the RGB image from the camera and uses it to identify all objects in the image. The node receives an input image from the camera and then use TensorFlow Lite to run a trained RetinaNet model. This produces an output list of each found obstacle. This list will include both the type of obstacle that is found and the coordinates of the bounding box of the obstacle. This information is then published for use by the Distance Estimation node to determine the distance to these objects and to choose the closest one.

The RetinaNet model also needed to be trained on hardware other than the Jetson Nano as the Nano does not provide enough computing power to train one in a reasonable amount of time. We intended to use a either UCFs Newton

cluster or a paid online service such as Google's Cloud TPU to create our models, but then decided to run it on a teammember's personal computer.

We ended up using SSD MobileNet V2 during prototyping, because we found better resources online. Additionally, though slightly less accurate than RetinaNet, we primarily needed the speed it provided. We followed a similar training process as what RetinaNet would require, and after the model was trained, it was then moved onto the Jetson Nano.

6.1.3.3 Distance Estimation

The distance estimation node on the drone computer has two major functions. First, it will calculate the distance from the drone to each of the obstacles detected by the RetinaNet algorithm. As input, the distance estimation algorithm gets the bounding box of each of the detected obstacles along with the type of obstacle that has been detected. As an output, the algorithm publishes which obstacle was closest and the approximate distance to that obstacle.

6.1.3.3.1 Node Design & Operation

Once both the bounding box of the obstacle and the type of detected obstacle is received, the node begins by translating the coordinates from the RGB image to that of the depth image. This is necessary because the RGB image and the depth images have different sizes and may be slightly shifted. The exact adjustments need will be determined during testing.

The node then determines of each obstacle distance based on the type of object and the portion of the depth image that is within the bounding box. If a ring is detected, the drone will take the average of the closest 10% of points within the bounding box. This should allow for the algorithm to detect the approximate distance to the ring. The main issue that could arise using this algorithm is the possibility of objects like pylons in front of a ring may be included if a ring is still detected.

If a pylon is detected, the algorithm takes an average of the depths within the bounding box. Detecting a double pylon will require that we take into account the distance between two pylons. This is because double pylons could be confused by a vision algorithm to be two separate pylons.

In order to remedy this, a list could be made of all the pylons detected by the vision algorithm. The node will then compare each pylon to every other to see if pylon is approximately 5 feet away from another. If this is the case, the node will take the average of the distances away from the drone and add a new double pylon item to the list of obstacles.

Once the distances to each of the objects is found, the node searches through the list of obstacles and publish the information for the obstacle that is closest to

the drone. This information will be received by the Drone Controller to be used in navigation and the Image Overlay node to overlay the video stream.

6.1.3.4 Height Sensor Node

Terabee provides a node that can be used within the ROS to communicate with the TeraRanger One distance sensor via a USB, UART, or I2C. This node is compatible with ROS Melodic and can provide distance data to the controller and the image overlay nodes.

6.1.3.5 Microphone Node

The ReSpeaker Mic Array v2.0 has available for it an ROS node called *respeaker_ros*. This node would allow for information from the microphone board connected by USB to be published to the system. This node is designed for ROS Kinetic, which means that it may not work with the ROS Melodic that will be running on the Nano. If this is not the case, we will need to set up a serial interface through the USB port. This can be done using the ROS node *rosserial*. From there, the information can be published to the subscriber nodes in the rest of the system.

6.1.3.6 Drone Controller

The drone controller acts as the main decision making drone of the hub. The controller receives data from many different nodes including the RetinaNet, Distance Estimation, Height Sensor, Microphone, and MAVROS nodes. What the controller does with that information is dependent on the mode it is currently in. There are main controller modes: Autonomous Control Mode (which contains AutoNav, Auto Maneuver, E-Stop, Take-off/Land) and Manual Control Mode.

6.1.3.6.1 Autonomous Control Modes

The Autonomous Control Mode is the mode in which the drone will navigate the obstacle course. In Autonomous Control Mode, the controller can systematically switch between autonomous submodes to complete the course.

When placed into autonomous control mode, the drone will attempt to navigate an obstacle course. The drone if located on the ground will enter into the Take-Off/Land submode and ascend to a height of 5 feet. Once the drone has taken off or if the autonomous control mode was engaged mid-flight, the controller will enter the AutoNav submode. Once the drone has navigated to an obstacle, the controller will enter the Auto Maneuver submode.

After the obstacle is maneuvered around, the controller enter AutoNav and repeat the process over. The controller will continue this process until it detects an entire wall directly in front of it, at which point the drone will attempt to navigate to its starting position. In the case that the drone battery depletes to 5%

of its maximum charge, the drone will enter E-Stop mode to prevent crashing. The flowchart of controller's operations can be seen in Figure 35 below. The flow chart demonstrates the different decisions used to influence the autonomous control of the drone via the CPU.

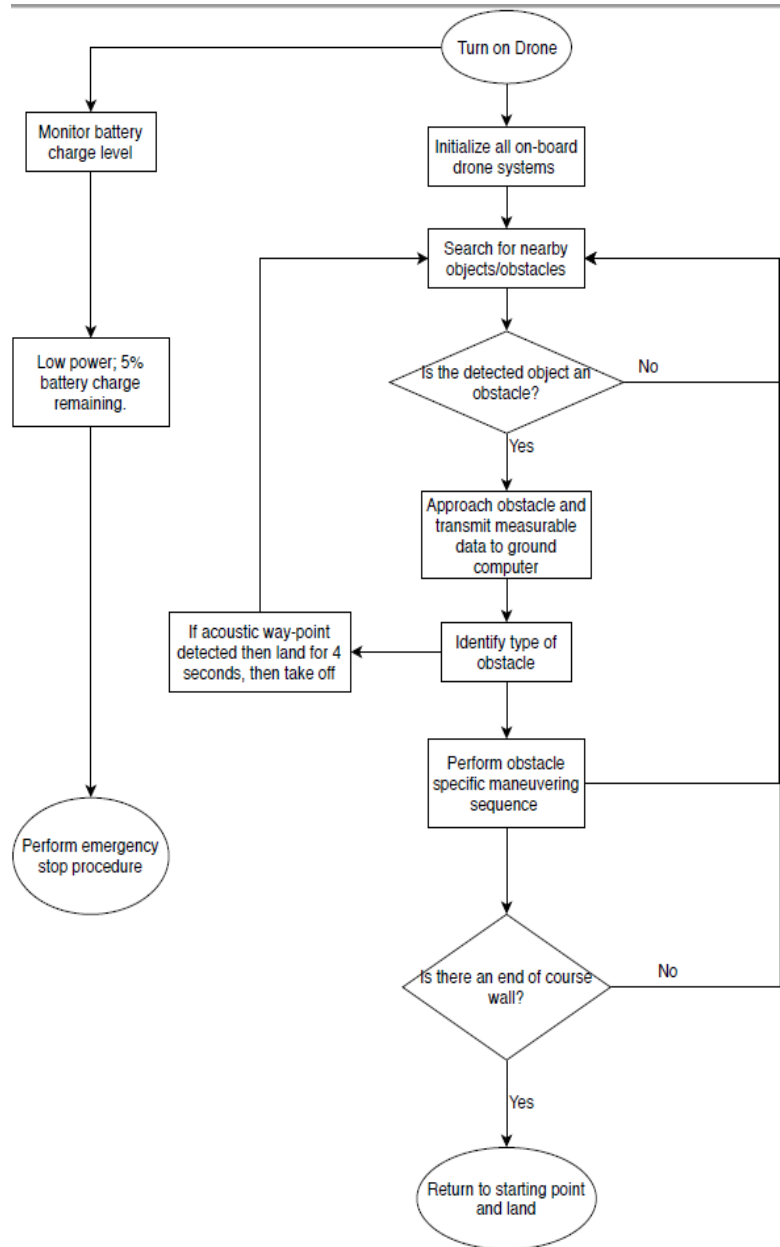


Figure 35: Flow chart of drone operation (RL)

If any interrupt signals from the ground station are received, the current submode operation will be stopped. The controller will then be switched into either the E-Stop submode or Manual Control Mode based on the received signal. If switched

into the Manual Control Mode, the drone will stop all horizontal movement and hover.

Auto Navigation (AutoNav)

In AutoNav mode, the controller will determine how to move based on the obstacle that is closest to the drone. In this submode, the node enters a loop where it will continuously receive data from the Distance Estimation. The Distance Estimation node supplies the distance to the closest obstacle along with its bounding box, and drone will attempt to navigate to this object based on the type of object and the location of the bounding box. This is sent by sending new input commands to the flight controller each loop. In order to prevent the drone from moving too fast or too rapidly, the drone will ensure that all commands sent to the flight control board will be at or below a specific threshold.

The AutoNav node then continues in this loop until it reaches a predefined point in front of the obstacle. The point at which the drone stops is dependent on the obstacle. For rings, the drone will stop centered in front of the rings about 2 feet away. For pylons, it will stop centered 1.5 feet from the pylon. For double pylons, it will stop centered in front of the right pylon about 1.5 feet away. From this point, the drone will enter Auto-Maneuver and navigate around the obstacle.

While the node is within the AutoNav loop, three events will be checked for at least once per loop:

- Command is received from the Ground Station
- Detected frequency from the microphone is between 0.5kHz and 1kHz
- Battery power is at or below 5%

Any of these cases will cause the AutoNav to change submodes.

If a command is received from the ground station, the Drone Controller node will either be switched into Manual Control Mode or switched into the E-Stop submode depending on the command. Before the drone is switched into manual control mode, the drone will stop all horizontal movement and hover in place.

At any point during AutoNav, there could be an audio waypoint along the path. Therefore during AutoNav software will be taking audio input from the microphones and scanning it at least once per loop to see if a frequency of 0.5kHz to 1kHz is being emitted near the ground. In the case that this occurs, the drone will stop lateral movement and enter the Take-Off/Land submode to land. After about five seconds on the ground, the drone will raise, re-enter AutoNav, and continue on its way. Within ten seconds after taking off again, the drone will be prevented from landing again due to the frequency in order to allow it to move past the audio waypoint.

The battery level can be retrieved from the PixHawk via MAVROS. If the battery power is found to be at 5% or below, the drone will enter Take-Off/Land mode to land safely before the drone loses all power and drops out of the sky.

When navigating, the controller will ensure that it does not run into anything that could be in its path. If something is encountered in its path, it will attempt to navigate around it by rising over it. Detecting interfering objects in the path will be done using a combination of the map created by the SLAM algorithm and the four directional HC-SR04 distance sensors placed around the drone. The point map will be useful in navigating around interfering objects in our path that are not within our field of view, and the distance sensors will provide additional data that will help in the case where the SLAM algorithm has not gathered enough data.

The one subsystem of AutoNav that is separate from navigation to obstacles is the return to home function. In this function, it will take the SLAM map developed while navigating the course to plot a way back from its current position. The drone will use its depth camera and HC-SR04 distance sensors to ensure that any obstacles that are encountered on the way back are avoided. Once the drone has reached the starting area, the drone will land and turn off the motors. After landing, the Drone Controller will exit Autonomous Control Mode and enter Manual Control Mode.

Auto Maneuver

In the Auto Maneuver submode, the controller will navigate around an obstacle depending on the type of obstacle it has encountered. When in Autonomous Control Mode, the type of obstacle will be determined by our object recognition software. When placed into this mode from Manual Mode, the drone will carry out the maneuver of the passed obstacle type.

Rings will be the easiest to maneuver through as all that is required is maintaining a forward motion without drifting vertically or to the sides. Maneuvering around pylons is harder. To do so, we will need to our SLAM mapping of the environment to ensure that we maintain a safe distance and complete an entire loop. Once the loop has been made, the drone can then start searching for the next obstacle.

Maneuvering around the double pylons will be the hardest. This will require for the drone to make an approximate 90° turn around one of the pylons, locate the second pylon, navigate near it, and make an approximate 270° turn around it. During the second turn, it is extremely important that the drone not make the turn too far from the second pylon or it will crash into the first pylon.

At any point during AutoNav, if a command is received by the Drone Controller from the Ground Station, the drone will switch into either Manual Control Mode or the E-Stop submode. Before the drone enters Manual Control Mode, the Drone Controller will halt all movement and hover in place.

E-Stop

E-Stop mode will be triggered by a button press on a controller connected to the ground station. The ground station then sends a signal to the drone via wifi. When the signal is received, the controller will immediately send commands to the flight controller via MAVROS to cease all horizontal movement and make a controlled drop to the ground. During this drop, the motors will provide some upward thrust as to reduce the force of the impact, but will still allow the drone to drop quickly. Once the drone has landed according to the height sensor and the movement data from the flight controller, the drone will turn off its motors entirely. This mode is used in case the drone enters into an unsafe condition and must be quickly stopped.

Take-Off/Land

The Take-Off/Land submode manages the taking off and landing of the drone. Take off will occur when the autonomous mode is engaged and the drone detects that it is on the ground. During Take-Off, the drone will fly to a height of 5 feet and hover. Landing mode will occur when the drone has reached the end of the course and has navigated back to the starting area. Take-Off/Land is also used when an audio waypoint is detected by the drone while in AutoNav mode.

When given the command to take off, the drone will use data from the distance sensor mounted on the bottom of the craft to determine the current height above ground. Once the drone has reached 5 feet, the drone will hover in place and enter into the AutoNav.

When given the command to land, the drone will slowly lower while keeping track of the reported height from the distance sensor. When the distance sensor no longer properly operates, the drone will continue to slowly lower until the IMU from the flight controller no longer reports movement.

When an audio waypoint is detected while in the AutoNav submode, a command to the Take-Off/Land submode will tell it to land the drone, turn off its motors for five seconds, and then take off back to the height that the drone started at. Once that process is complete, the controller will return back to the AutoNav mode.

6.1.3.6.2 Manual Control Mode

In manual mode, the drone responds directly to any commands sent by the ground station. The ground station is capable of sending flight commands from the ground station, enabling a human to control the direction of the drone. Additionally, the ground station will be able to send commands to carry out a single specific instance of one of an autonomous submode. For example, the AutoNav submode can be enabled to travel to an object.

Once the drone has arrived, the drone controller node will have the drone hover until the next command is received. This function will be useful in testing specific

operational modes. Additionally, while in manual control mode, the operator will be able to put the drone into Autonomous Control Mode, allowing the drone to be set up to test a specific portion of an obstacle course.

6.1.3.7 Simultaneous Localization and Mapping

Simultaneous localization and mapping (SLAM) is a method to provide feedback to the drone to determine its location in 3D space. By using SLAM, we would have been able to accomplish two things. First, SLAM will be able to keep our drone from drifting while navigating to and around obstacles. This is accomplished by determining if the SLAM algorithm is reporting that we are moving in an unwanted direction and correcting the trajectory. Second, SLAM will enable us to return to places that we have been using a 3D point map. This will enable us to more easily maneuver around the obstacles, and will provide us data to return back to our starting position.

In order to implement SLAM into our project, we would have used the ROS node called `rtabmap`. This node is able to take in a depth image and produce the necessary 3D point clouds. The node explicitly supports our Intel RealSense D435 camera, so integration into our system should be relatively quick and easy. Nodes can subscribe to topics from this node to get information on the cloud map.

6.1.3.8 MAVROS

Micro Aerial Vehicle ROS (MAVROS) is a ROS node that will be used as an interface to facilitate communication between the drone computer and the ArduPilot software of the flight controller. MAVROS will receive commands from the controller node and will process those commands into serial signals that can be read by the ArduPilot. The ArduPilot likewise will be able to send information to the drone computer through the reverse method. MAVROS is designed to work with ROS Melodic and with ArduPilot, so implementing the node should be a simple process.

6.2 Ground Control Station Software (HS)

A ground control station is a program that would allow us to interact with our drone during operation. It provides status information on the drone, and provide a variety of information such as altitude, speed, and trim. We will need to select a Ground Control Station program to run on the laptop from which we will view the video feed during the drone competition.

The two most popular ground control station programs are QGroundControl (QGC) and MissionPlanner. They are similar in available features, however QGroundControl works on Microsoft Windows, Mac OS, Android, and iOS. MissionPlanner only works natively on Microsoft Windows, though it may be possible to run on Mac OS. Being able to view this data from a mobile device

would be helpful, especially if we do any testing outdoors. Therefore, we will pick QGroundControl as the primary interface between the drone.

QGroundControl provides full flight control and vehicle setup for PX4 or ArduPilot powered vehicles. It provides easy and straightforward usage for beginners, while still delivering high end feature support for experienced users. Key features of QGroundControl include flight support for vehicles running PX4 and ArduPilot, and mission planning for autonomous flight. QGroundControl runs on Windows, OS X, Linux platforms. QGroundControl also runs on Android and iOS, allowing us to easily debug the drone during testing from a handheld device. [4]

Controllers such as the NVIDIA Shield or wired alternatives like the Microsoft Xbox gaming controllers; Xbox 360 or Xbox One, can be used to control a drone remotely. In such a scenario, the ground control station machine, or the laptop, needs to run a ROS joystick node and be able to connect to the ROS master running on the NVIDIA Jetson.

In the end, we found that it was easier to sufficiently control the drone by SSH into the computer's shell to execute specific scripts that we needed. Instead of using a gaming controller to fly the drone, we could send specific MAVLINK commands to the drone.

The easiest way to install the required components on a laptop is to use a ROS Docker container. Using Docker allows you to isolate your host system from any changes, such as software installation and configuration updates. QGroundControl will be used in tandem with the ROS Joystick Node to interface with the Drone Pixhawk and PX4 flight board to enable manual control. Autonomous control will be handled on the Nvidia Jetson Nano aboard the Drone itself.

The Drone camera feed will be processed and have image recognition and overlay done on the Jetson Nano aboard the drone and will then have the camera feed streamed over Wi-Fi to the router within the ground control station. The Ground Control Station will have a web browser open to SSH into the drone's camera stream via the Jetson Nano's IP address. This will allow us to remotely access the camera stream from another computer.

The camera on the Jetson would need to be configured and have its data broadcasted to the Wi-Fi Router to enable this feature. This will satisfy the requirement to have a live camera feed from the drone accessible and visible in the ground control station. Design of the ground control station subsystem is outlined in Figure 36. (HS)

The drone will be able to communicate with ROS nodes on the ground station by connecting them on a wireless local area network (WLAN) or on an ad-hoc network. All computer systems running ROS nodes will be able to communicate

with each other as long as the systems allow bi-directional communication between themselves and are advertised on the network. This requires that at least part of our ground control software use ROS nodes, but it will prevent us from creating a custom node on the drone to send and receive TCP or UDP packets. (CJ)

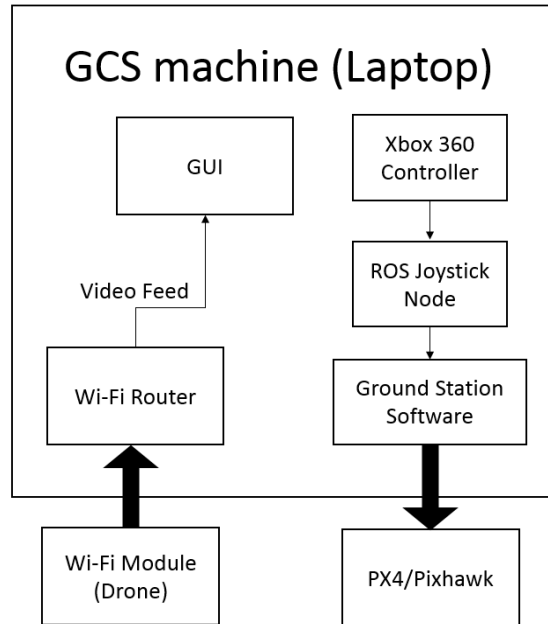


Figure 36: Ground control station setup diagram (HS)

6.3 Software: Interference and Failure Modes (RJ)

Ideally, our drone will be able to complete the entire obstacle course without any issues. However, we recognize that our drone may not be able to perform all missions successfully. For example, our drone may not be able to identify an object, may fail at maneuvering through or around an object, and may get attacked by an adversarial mine. We have outlined several contingency plans in our code to direct the drone in the event it experiences one of these situations.

6.3.1 Unidentified Object Mode

Due to the field of view that our chosen camera provides, we expect to identify an object directly after completing a previous maneuver. However, if the drone is unable to identify an object, it will execute a panning mode to rotate the drone 30° to the left, and then 30° to the right. If the drone is still unable to identify an object while panning, then it will increase its altitude by 2 feet, and then decrease its altitude 2 feet (as long as there is at least 2 feet of ground clearance). If the drone is still unable to identify the object, then the drone creeps forward six inches while trying to identify the object again. It will repeat this procedure at most three times.

If the drone is unable to find any object after three attempts, it will fly up to an altitude of 20 feet and fly back in line of sight to the starting point. The drone will be able to do this due to the location mapping functionality that will be implemented. After arriving at the starting point, the drone will descend down to 5 feet and restart AutoNav mode. When the drone begins AutoNav, it will not repeat an obstacle that it already completed or attempted, which means it will pursue a different direction (from the starting point) in relation to the previous run.

6.3.2 Flight Recovery Mode

If the drone fails at maneuvering about an obstacle or gets attacked from an adversarial mine that results in falling out of the sky, the drone will enter Flight Recovery Mode. The drone will attempt to self-correct its angle of flight in relation to the horizon when encountering external interference. However, if after three seconds the drone is unable to restore level flight, the throttle input to the motors will be gradually reduced to zero so that the drone can land as softly as possible. Then, in this mode, the drone will first attempt to fly at 1 foot above the ground for 10 seconds.

If the drone is unable to reach an altitude of 1 foot, then it can be concluded that the drone is either upside down or that it sustained physical damage. In this situation, the drone will enter a stop mode where it will cease to move until it is reset. However, if the drone is able to maintain level flight at an altitude of 1 foot above ground level for 10 seconds, the drone will then rise to 20 feet and fly back in line of sight to the starting point. Once above the starting point, the drone will descend to 5 feet and restart AutoNav while pursuing a new path.

6.3.3 Mine Avoidance Mode

In the second run of the competition, we will be flying through the obstacle course with the presence of an adversarial mine. We know the mine will not exceed the dimensions of 1.5 ft x 1.5 ft x 1.5 ft, and that the blast zone will not have a vertical height greater than 10 ft and a horizontal distance greater than 3 ft. Therefore, when we run the drone the second time through the obstacle course, the drone will hover 12 ft above the ground, instead of 5 ft above the ground. This should prevent us from being hit from any projectiles.

We will be using an ultrasonic sensor facing downwards which will give us a height reading, in addition to the built-in altitude sensor (via the use of a barometer) in the flight controller. If the flight controller altitude is reading 12 ft, but the ultrasonic sensor is reading 10.5 ft or less, then we will assume that we are above a mine. If we are above a mine, then we will not attempt to maneuver about a detected obstacle that is within 3 ft of the detected mine. Cutting too close to an obstacle may cause the drone to fall.

It is possible that the mine may have the ability to move. Therefore, if we find that the mine follows us to all locations, then the drone will not attempt to maneuver through any obstacles. The drone will land at the original starting point once the

time limit is reached or if the battery power is low, regardless of the number of obstacles completed. The primary objective during the second round is to protect our drone, while the secondary object is to complete the course.

6.3.4 Object Proximity Mode

The other four proximity sensors will be placed on all four sides of the quadcopter, and we will use the readings to determine if we are too close to an object. The sensors will be connected to a single microcontroller, and we will continually check that the drone is not about to hit an object that we are trying to maneuver around. If one of the sensors detect that the drone is two inches or less from any object, the drone will stop horizontal movement, and drift until the distance for that particular sensor is greater than two inches. This will ensure that when we attempt to maneuver around an object, that we have enough space to do so without damaging our drone.

6.4 Microcontroller Software (RJ)

We will be using the Arduino IDE for the PCB we will use to interface the ultrasonic sensors with our CPU. We are primarily interested in reducing the workload and complexity of the CPU software. The microcontroller will read data from the five sensors. We will set a threshold of two inches for the four horizontally placed sensors on each side of the drone

The microcontroller will record the data, and if the distance detected is under the threshold, it will return a unique value over UART to the CPU. A return value of '1' will indicate that the drone is too close to the left side, '2' will indicate that its too close to the right side, '3' will indicate that its too close to the front side, and '4' will indicate that it's too close to the backside. When the CPU receives one of these numbers, it will know what type of corrective action is required to prevent contact with the drone and the object in question. Figure 37 below shows the software flow for the microcontroller.

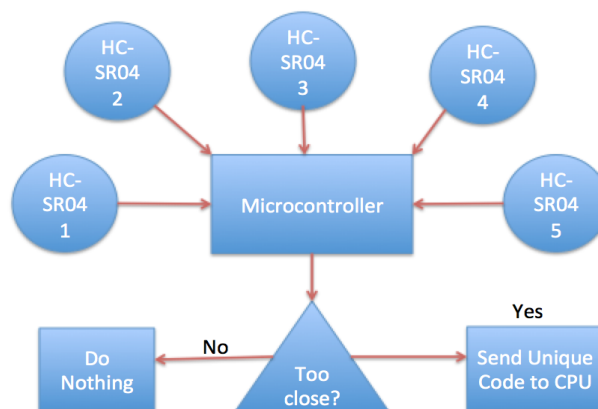


Figure 37: Microcontroller Software flow

7.0 Project Construction and Coding

7.1 Hardware Construction

7.1.1 Needed Equipment and Building Space (RJ)

The primary location of building our drone and testing its components was the Texas Instruments Innovation Lab and the Senior Design lab. We often needed materials and equipment that will easily be found in those labs. A significant portion of building our drone involved soldering components, requiring a soldering iron. We will also need to utilize a reflow oven to mount certain circuitry components onto our PCB. Moreover, we needed to utilize an oscilloscope and multimeter to check the voltage and current flow through various components; this was severely hindered due to the campus closures during the second half of the semester.

Once our drone is physically built, we will primarily conduct our tests indoors. The demonstration facility at Lockheed Martin is indoors, and an outdoor testing location would require us to recalibrate the sensors when switching locations. This could cause performance issues that we would prefer to avoid. Additionally, we would not have to worry about local outdoor airspace restrictions that may exist.

We attempted to conduct our tests in an open garage at low thrusts to verify basic flight functionality. For the competition, we have the ability to fly up to 45 feet; however, we do not need to fly that high during testing in order to train our drone to detect the objects. Moreover, Lockheed Martin has several testing days that will allow us to fly the drone in their facility, allowing us to test the drone in a less constrained airspace.

We will need a space where we can plug in our CPU into a screen and keyboard. We will primarily work on this using our own screens at home, but we also have the opportunity to use the monitors in the MAE Senior Design lab. Alternatively, we occasionally SSH into the terminal of the CPU.

7.1.2 Hardware Calibration and Configuration

In order for the drone to operate properly, several components of the drone need to be calibrated in order to record and produce accurate measurements.

7.1.2.1 Electronic Speed Controllers (ESCs) (RJ)

Electronic speed controllers control the flow of current into the motors, thereby controlling the thrust of the drone. If the electronic speed controllers are not calibrated, the motors may produce uneven amounts of thrust causing the drone to drift or flip over. The Pixhawk has an ESC calibration mode where it will run

the motors at full speed (with the propeller removed) to take a baseline measurement to compare the motors to one another. [9]

7.1.2.2 Depth Camera Calibration (CJ)

The D430 camera software needed to be calibrated to create an accurate depth image. Out of the box, the camera likely will not be extremely accurate due to minor inaccuracies in manufacturing. In order to calibrate the camera, we used the calibration software provided by Intel. [15]

Calibration is done to adjust the rotation and translation values between the RGB image and the left and the right cameras. This is necessary as we will need to know how much to shift the bounding box found on the RGB image to the depth image for calculating distance to an object. Additionally calibration is used to improve each cameras focus and reduce distortion.

7.1.2.3 PX4FLOW Calibration (HS)

The PX4FLOW Camera will be pointed downwards towards the ground and be used to assist with SLAM navigation. To calibrate the PX4FLOW Camera, QGroundControl will be used in tandem with its autopilot feature. With the drone connected via telemetry, the drone will need its propellers removed. To calibrate, the drone will be manually rotated in its respective roll axis through a range of ± 15 degrees within two seconds. This process will be repeated about the drone's pitch axis. This will be repeated Repeat this 10 times to ensure accuracy in calibration.

Camera data will be referenced with gyroscopic data from the flight controller. Data will be analyzed in the autopilot software and adjusted to have both data values matched. Range tests will be calibrated by holding the camera a set distance from the ground, and measuring it physically and comparing it to its digital readout on the autopilot software. [23]

7.1.2.4 PixHawk Calibration and Configuration (RJ)

The PixHawk flight controller has several onboard sensors such as the gyroscope, accelerometer, barometer, and compass. These sensors provide information about the acceleration, orientation, and altitude of the drone to the flight controller, which can then be viewed from the ground control station. Since we will be flying indoors for the competition, we will perform the calibration indoors to get the most accurate readings. [23]

7.1.3 Phase I (RJ)

In the first phase, we planned to assemble the prototype drone using most of the sensors that we require for our final build. Before placing the components onto the drone, we individually tested the components to ensure that they are not defective. Once placed on the drone, we will ensure that the components are

securely fastened, and that the basic drone is flyable. We expected the need to develop additional mounts for the components as necessary during this stage.

Our primary goal for this phase is to make sure that we can have a standard drone flying with the components that we selected. Simultaneously, we will begin working on the computer vision algorithms and configuring the flow of data between the sensors and the CPU. We will work on getting all the raw sensor data and making sure that they are centrally available in the CPU. However, we did not configure the drone to fly with autonomous functionality at this time. We wanted to take this data and process it to match the obstacle course requirements set out by our sponsor.

We planned establish a rudimentary system to utilize sensor data to control the drone, and we will program basic autonomous functionality for the prototype objects that were given to us. To do this, we will also create machine learning models using a dataset of images taken of the objects in different lighting conditions, at different angles, at different heights, and at different distances.

Building an algorithm using all of these data points will allow us to achieve better performance. This allowed us to ensure that we assembled the circuitry correctly, and that we know how to utilize the flight controller, transmitter, and receiver to fly a basic drone. At the same time, working on the sensors and CPU allowed us to prepare the drone for future autonomous functionality that will be implemented in the next phase.

7.1.4 Phase II (RJ)

In the next phase, we planned to combine the basic drone with the autonomous functionality. The drone will already be assembled with most of the components, however, we will need to reattach some additional components that was previously removed for development, such as the CPU and camera. The flight controller will receive input commands directly from the CPU, and not the transmitter/receiver that was manually controlled in Phase I. A significant portion of time would be spent interfacing sensor data with flight movement.

In this phase, we expected that our image recognition algorithm to have low accuracy, but we are primarily interested in verifying that we can maintain control of the drone. We will also implement and verify the flight control modes that were outlined previously are functional. We will perform specific hardware and software tests before we fly our drone. These tests are outlined below, and will help us prevent failures which could damage our drone or cause injury to someone. Once we verify that our drone and it's components pass the the specific hardware and software tests outlined below, we would individually test each specific maneuver we expect the drone to be able to perform. For example, the drone should be able to detect a ring and fly through it.

In a separate event, the drone should be able to detect a pylon and fly through it. Instead of testing both of those items successively, as the drone should be able to perform during the competition, we will first see if the drone can individually pass those tests. Then, we will test if the drone can fly between objects without maneuvering in or around those objects. This will allow us to examine each function separately, and help us identify points of failure that would otherwise be hard to identify.

7.1.5 Phase III (RJ)

Our final phase would include fine tuning the autonomous drone for improved performance, and to make sure that it can complete an obstacle course in preparation for the competition at Lockheed Martin. We will follow the performance testing guidelines we established as a way to focus on specific features of the drone. We may need to purchase additional components if we find that there is a significant deficiency that is preventing us from satisfying our operational requirements.

Object detection is one aspect of the competition, however, we must determine how the drone will fly towards the next object and the decisions it will require to get there. Moreover, a major aspect of the competition will be the adversarial landmines that threaten to bring down our drone. During this phase, we will develop and implement defensive maneuvers to avoid being attacked by a landmine. The reason why we are waiting until the last phase to do this is because we first want to make sure that other features of the drone are operational. Our primary consideration during the second round of the competition will be to protect our drone from any attacks.

7.2 Software Development

The software development process is what we believe will take the longest to accomplish due to the size and complexity of the necessary software. For this reason, it will be important to decide upon how the software will be developed ahead of time. In this section, we will discuss how we will organize our software development to ensure efficient use of our time. (CJ)

7.2.1 Language Choice (CJ)

The choice of language choice will depend on the section of the software being coded. This is because different parts of the software may require different libraries or different levels of efficiency. There are two distinct

A large portion of the project will be made up of ROS nodes. ROS supports C++ and Python, so we had the choose between these two when deciding on what to code the node in. Other portions of the code on the ground station could be coded in another language, but to reduce the need to become familiar with other languages, we will attempt to use languages other than above on the ground station software unless we find it reasonably necessary.

In order to understand what language we need to code each of our nodes in, we will need to understand the tradeoffs for the languages. Python is an interpreted language. This means that the code is executed at run time without need for compilation. This means that with Python that it is sometimes hard to catch errors before running the program. It also means that Python tends to run slower than a compiled language like C++. Additionally, since the Python interpreter is what manages memory and not the program, Python scripts can use up more memory than is absolutely necessary. What Python loses in efficiency it makes up for in user friendliness. The user does not have to manage memory, and most operations are at a higher level, meaning that the same goal can be accomplished with fewer lines. Also several members of the MAE team know how to use Python, and this will help them contribute to the programming of the project.

C++ is an object oriented language based on C. It is a language that requires the programmer to manage the memory when creating data structures and objects. Because of this, it can take several lines of code to implement what Python can do in a single line. However, this enables the programmer to make memory management more efficient. Another benefit that C++ has over Python is that it is compiled to binary before operation. This enables it to run faster than Python since the written code does not have to be interpreted at runtime.

Despite its need for memory management, C++ will be used where possible to improve the speed at which our algorithms will run. This will reduce the time it takes to run through loops during the AutoNav and Auto Maneuver autonomous submodes. If we decide that using a python library will be necessary for a node or that we feel that the node design would benefit significantly from people from the MAE team working directly on the node, we will use Python.

For the microcontroller, we will utilize the Arduino IDE, which can support C and C++. The Arduino IDE simplifies the coding process for the microcontroller aspect of the project, as it contains helpful packages and debugging tools specific to the Arduino platform.

7.3 Development Methodology

There are many software design models such as the Waterfall model, Prototyping model, or Agile model. We have decided to go with Agile model as with this method there are many iterations of software development and this allows for change in design and requirements. A high-level plan of the features to be implemented is suitable with the Agile method and it doesn't require having a detailed design beforehand. By using this method, the developer can design, implement, and debug each module separately with each iteration.

The Agile method is a good process to use for our team as we are working with a large team composed of many engineering disciplines. There is a chance that we will have to change our initial design proposal or change parts of the software we develop so that we can integrate it with the modules the other members will be developing. By using Agile, we will be able to rapidly develop software for various parts of our system and we will have the option to change any parts if needed in an efficient manner.

8.0 Project Prototype Testing Plan (RL)

This section will discuss the different testing phases that was planned to be conducted during the developmental stages of the drone prototype. In order for the project to be successful, thorough testing will need to be done to ensure proper functionality of the different components of the drone. There are several factors to consider while conducting the different testing phases as each component will require its unique set of testing conditioning.

To ensure proper testing is being conducted, appropriate testing environments will need to be established for both hardware testing and software testing. Several testing runs will need to be conducted after each individual component testing has been conducted to ensure all the components function accurately when combined as a finished product. These testing phases are necessary to work out the kinks and make adjustments to both the hardware and software components before incorporating them into the final product. The project prototype testing plan is separated into four main sections to adequately describe each testing phase and selective environments established to perform the tests.

This project has various moving parts considering the size and complexibility of the final product. We needed to prioritize different segments of the project in order to make progress with our design. To do this, we plan to test our project in three phases to divide our goals into manageable segments. These phases emphasize the electrical and computer engineering aspects of this project, however, it is likely that moving from the starter frame (from the DJI F450) to the final frame may occur during phase three.

8.1 Hardware Testing (RL)

The final product requires several components to work together to ensure successful completion of the project. Each hardware components have several sub-components that will need to be considered to ensure all major components function accurately. Communication between each sub-components and major components will mainly be accomplished through coding, and coding based platforms. Software testing and code testing environments will be discussed in the later sections. This section will primarily be focused on the hardware testing.

8.1.1 Hardware Test Environment

For testing each component of hardware to be used, several factors needed to be considered. These factors will be determined based on the individual component being tested. One of the major hardware components to be tested is the power supply circuit. Power from the battery will be distributed to all electrical components of the drone through the use of power distribution board. Each electrical component will have its own individual power requirements. Miscalculations in the amount of power to be supplied to each component may lead to catastrophic failures and can cause components to overheat and burn out. The power supply itself, which will be the lithium polymer battery, will not require much thorough testing as the output of the power supply will be constant, and the charge capacity and discharge rate will be fairly exact to the manufacturers specifications. The only forms of testing that would be conducted on the power supply are:

- Its ability to distribute a steady level of voltage
- The true running times at different levels of operations including durations of operations at high power usage, lower power usage, and optimal power usage.
- Total charging time of the battery with zero stored charge.

These basic tests for the battery will mostly be conducted in the laboratory. The testing for operation duration will be made possible once all components are connected and the drone is taken for a test flight. The operational duration test for the battery will be the only test conducted in a field setting, and will be conducted simultaneously with other tests that will also need to be conducted in the same field setting environment.

Small passive components such as resistors, capacitors, switching regulators, operational amplifiers, transformers, and other similar small electronic parts will not be thoroughly tested as individual components. These parts will have manufacturer specifications which will be used to determine the correct parts to use. The only test that will be involved with the testing for these components is using multimeters that are available in the laboratories to ensure the components are not blown and functional. However, the circuits that will be built using these components will be thoroughly tested and precise measurements will be taken as the circuits built will be used as blueprints for ordering the appropriate PCBs required for the project.

The PDB is one of the most essential circuits to be used for this project. Ensuring accurate voltage levels are supplied to each electrical component is very important for optimal operation of the drone as a whole. Power will be distributed from the battery to the various components of the drone through the PCB circuits.

Simulation software, such as Eagle and Multisim, will first be used to design a circuit that will be suitable for the various power distribution requirements. Once

the simulated circuits are designed, the circuits will then be built on breadboards and tested in a laboratory setting environment. Thorough measurements will be taken to ensure optimal results are achieved. More details on the PCB circuits and schematics are available in section 6.0.

Since the drone systems will be powered by a DC power source, and all the components will require a DC power input, there will be no need for an AC to DC power converter. This simplifies the design process for the PCBs by a bit as it eliminates the need for an extra circuit.

The Jetson Nano will be used as the drone's CPU. It is a small but efficient computer for embedded applications, and with its smaller size, it was decided to be the appropriate CPU to be used for the project. The Jetson Nano requires an input voltage of 5 volts with a low-power consumption of 5 watts and a high-power consumption of 10 watts. This CPU will be tested in several phases and on different aspects and environmental conditions.

The software portion of the testing phase will be explained in later sections. The CPU's input power will be supplied through the PCB. The first phase of the testing will be done in a laboratory setting environment where coding will be implemented in the CPU and other essential components will be connected to it via its USB ports. The components to be connected includes a depth perception camera, a microphone, WIFI and radio telemetry receiver and transmitter, the flight controller, and sensors.

Each component to be connected to the CPU will first be connected individually, and the algorithm will be tested to check for proper functionality. The camera is an important component that will be connected to the CPU. The CPU will need to be able to process the images captured by the camera. Through coded instructions, the CPU will then need to be able to identify the different images sent by the camera.

The first thing the CPU will need to do is match the images received with the instructions in the coding to differentiate if the objects captured in the images are targets (obstacles) of interest. It will then need to place the image in one of the categories of the targets of interest. The CPU will then need to match the target identified with the set of maneuvering instructions for that particular target.

Training will need to be performed using the CPU in both a laboratory setting, and field setting environment. The initial testing phase will be conducted in the laboratory, where the CPU with the camera connected will be moved around the stationary obstacles provided by Lockheed Martin, and checked to see if the CPU is able to process the images and utilize the image recognition software implementation. After thorough testing in the laboratory, the next phase of the testing will require a field test where the CPU will be exposed to external

extremities and influences, and the system will be checked to see if it is able to perform as expected under those conditions.

The CPU and camera combination will also be tested for proper ability to determine its distance for an identified obstacle, and adjust the distance as the system's location gets closer, or further, from the obstacles. The pair will be tested on its ability to determine its height from ground level, the confidence level of the approaching obstacle, and the time it will take for the system to reach the obstacle.

Aside from identifying and differentiating between objects and obstacles, and the type of obstacle, the CPU will also need to be able to detect mines set by the "mine" team. For this the CPU will have sensors and infrared cameras connected to it. The CPU will also be equipped with a set of coded instructions to perform evasive maneuvers, in the event of fast approaching projectiles. Similar environment to testing for the depth perception camera, will be utilized for testing these components.

The CPU will also be tested with a microphone connected to it. The microphone will be used to detect acoustic sound waves which will be used to mark waypoints for the drone to land on and take off. The testing phase for this will be conducted mostly in the laboratory. Testing this feature will prove to be challenging in an external field setting due to noise interruptions of the environment. A band-pass filter may be necessary to be implemented to weed out the noise and detect the acoustic sound waves. A set of instructions will be coded into the CPU to instruct the drone to land at the waypoint for a set amount of time (5 seconds), then take off and carry on the flight path.

The CPU will also be equipped with a WIFI receiver/transmitter. This will be used to send and receive data and instructions to and from a ground based computer. The data collected and processed by the CPU, along with the live video feed will be transmitted to the ground computer. The flight path mapping data will also be transmitted to the ground computer where it will be stored to keep track of the drone's location, along with the locations of the obstacles detected, waypoints, and the starting point.

It is essential to set up the appropriate testing environment for devices to be tested. A set of the devices to be tested and the corresponding testing environment for the respective devices is summarized in the list below:

- Battery: will be tested in a laboratory
- Printed circuit board (PCB): will be tested in the laboratory at different stages. First stage will be conducted by connecting the CPU to the PCB. Second stage will be conducted by connecting additional devices to the PCB and the CPU. The final stage will be conducted with all the

components connected to the PCB. The final stage testing will be conducted in both a laboratory setting and a field setting.

- CPU (Jetson Nano): the CPU will be tested first in the laboratory with individual components connected to it. Then it will be tested with all the components connected together in both the laboratory and in the field during the test flight of the prototype.
- The camera and the microphone: will be tested together with the CPU on the same environmental setting.
- WIFI/Radio module: will be tested the same way as the camera and microphone.

8.2 Hardware Specific Testing (RL)

Each hardware component will be tested using various equipment and measuring devices. The components will be set up at different conditionings and measurement data will be collected and recorded to check for optimal functionality and operations. Hardware specific testing is important to ensure all components are working individually as in junction with other connected components. Performing such tests is essential for the project's success.

8.2.1 Lithium Polymer Battery Testing (RL)

As mentioned in the previous section, the lithium polymer battery will not require extensive testing. However, due to the nature of this type of battery, certain tests will still need to be conducted to ensure its reliability and integrity. The battery will first be physical checked to look for any structural abnormalities such as bumps, cracks, and exposed wires and internal casings. Doing this will ensure the battery does not pose any potential health and safety risks to anyone. It will then be fully charged using a charger with manufacturer recommended settings. This step is important due to the nature of lithium polymer (LiPo) batteries as this type of batteries tend to explode and potentially cause fire and harmful chemical leakage if not charged correctly or overcharged.

The battery's temperature will also need to be closely monitored as overheating may also lead to the risks mentioned above. For records purposes, the total charging time of the battery will be tracked and recorded. This measurement may be useful to help the team determine the need to acquire additional batteries.

During preliminary testing, the battery will be connected to a multimeter to monitor the voltage levels delivered to the PCB and other components that will be connected to it. The battery's total time of operation will be monitored at different levels of operation. These levels include running time will all components connected and operating at full capacity with all components operating at maximum output. Performing this test will help the team determine the battery's ability to operate and supply power to the whole system during the entire duration of the flight course.

8.2.2 Printed Circuit Board (RL and RJ)

Before placing orders for PCBs to be built, the circuits to be implemented will first need to be designed and tested. This will be done using circuit simulation software such as Multisim, Eagle, LT Spice, and the likes. The various circuits needed for this project include voltage regulation circuits, amplifier circuits, voltage to current conversion circuits. Once these circuits are designed and simulated on the relevant software, they will then be built on breadboards, and measurements will be taken to ensure the circuit will work for the project and be able to generate the expected outcome. Once all necessary testing for the circuits are completed, orders for the PCBs will be placed. Details for the PCB designing and ordering can be found in section 6.0 of this report.

For the microcontroller aspect of the PCB, we will first use the Arduino Uno Rev3 development kit using the processor pins we designated for our schematic. Once we test our prototype, we will order our PCB. We will flash the processor in order to upload our code, and then we will test to make sure that our microcontroller works as expected.

The completed PCB will be tested with each individual component of the drone's electrical system to ensure proper functionality. We will first connect our power source to our PCB, and measure the output voltage. Once verified we are receiving the correct output voltage for our sensors, we will connect one of the main components to be tested using the PCB, which is the CPU via UART.

Once the CPU is found to be operating accurately with the PCB, other components will be added to the system, one at a time, to check if the components are able to operate together. This testing sequence will be repeated until all components of the system is integrated and the system is found to be working in harmony. Testing the PCB by slowly adding more peripherals will allow us to understand the causes of failure, if they should occur.

8.2.3 Jetson Nano and component pairing (RL)

The Jetson Nano was chosen to be the CPU to be used for the autonomous drone project. This component will be heavily tested in terms of both hardware functionality and software compatibility. Most of the systems major components will be connected to the CPU, and thus, it will be very important to ensure this device is programmed and tested correctly.

The first step of the testing will involve ensuring the CPU is receiving the accurate level of voltage and current from the PCB needed for it to operate. The CPU then be paired with other crucial components and its ability to process data received from the paired devices will be observed. Adjustments will be performed based on these observations and the pairs will be tested again. This process will be repeated until the desired outcome is achieved from each pair being tested.

The CPU will be programmed with a coding that will allow the system to spot objects in images it receives from the camera. The camera will need to connectable to the CPU, and its intake power will need to be monitored. Should the CPU not be able to provide enough power to the camera, the camera will need to be powered through the PCB. The team has chosen a camera with depth perception capability. The sensor used in the camera for this feature will also need to be power through the PCB as the CPU may not be able to provide sufficient power to the sensor.

The CPU will process the objects it detects in the image received, and identify if the objects detected are obstacles for interest or background objects that can be ignored. The CPU will be programmed to identify three types of objects: rings, single pylons, and double pylons. The CPU's programming will also include a set of instructions to perform obstacle specific maneuvering. The CPU will be tested for its ability to identify the different types of obstacles and pair the respect set of maneuvering instructions for each obstacle.

The CPU will also be programmed to land at waypoints marked by acoustic sound waves. As such, the CPU will be paired with a microphone that will be able to detect the acoustic signal. The same steps, to that for the camera, will be taken for the microphone. Both connectivity and compatibility of the device will be tested to obtain the desired outcome. The CPU will receive the acoustic signal via the microphone and instruct the system to get into a set preprogrammed proximity of the signal and land for a set time frame (5 seconds), and then takeoff and continue with the course. The CPU will be tested with its ability to process the signal and execute the set of instructions to perform the landing sequence.

The system will be programmed with a set of instructions to perform an emergency stop and shutdown sequence as a safety feature to prevent injury or damage to property. The command for the emergency shutdown will be entered into the ground computer and transmitted via a wireless router to the WIFI receiver onboard the drone's system. Similar compatibility, power supply, and connectivity tests will be performed for the WIFI module for the CPU.

The CPU will be tested in its ability to receive and process the command from the WIFI module, and execute the set of instructions for the emergency stop sequence. The emergency stop protocol is a primarily a three-step process once the command is received via WIFI. The CPU will be checked to see if it instructs the system to stop all lateral movements, which is the first step of the emergency stop protocol.

For the next step, the CPU will send a command to the system's flight controller which will reduce power of the drone motors through the electronic speed controller (ESC). This will cause the drone to slowly descend and perform a soft landing. The third step of this process requires the CPU to detect the completion

of the landing and shutdown all power to the system once on the ground. Testing will be performed for all three steps of the emergency stop sequence.

8.2.4 Flight controller and electronic speed controller (ESC) (RL)

The flight controller and electronic speed controller (ESC) will be used to steer the drone, as well as control the takeoff and landing of the drone. Although most of the testing for the flight controller and the ESC will be performed by the mechanical and aerospace engineer portion of the team, the electrical and computer engineering portion of the team will test the power flow to these components, as well as, the compatibility and connectivity of the components to the drone's CPU. We will test for ways to process signals sent to these components for accurate maneuverability of the drone through the course.

All of the component testing will first be conducted in the laboratory. Typical laboratory measurement equipment such as multimeters, function generator, and oscilloscope will be used to collect data for the various circuits to be used. A computer with an operating system that is compatible with the coding to be implemented will be used to test the code and CPU processing capability of data received from input devices (camera, microphone, WIFI module, etc.). The project sponsors have provided sample obstacles that will be used on the final course tryout. The team will use these sample obstacles for CPU training and field testing the prototype.

8.3 Drone Software Testing (CJ)

Software testing is essential to ensuring that our drone operates properly and safely. Ensuring that our programs work properly will prevent damage to other objects, people, and the drone itself. The process of software testing will begin with testing individual portions of the code, which in our case will be our software nodes. After we have shown proper operations of each of the nodes, we will test begin testing software as a whole during our software-in-loop testing phase. Once our drone software has successfully passed both stages, the drone would be tested as a whole with all of the sensors. Unfortunately due to the COVID19 pandemic, we were not able to dedicate much time to software testing.

8.3.1 Unit Testing Nodes

Unit testing of each of the software nodes would be carried out to ensure proper operation. In order to test the nodes, unit testing modules will be programmed. These testing modules will publish information to the inputs of the node being tested. The node will then use that information to carry out its functions. Once the node has published all information out to the subscribed nodes, the testing module will then check to ensure that the published information was correct. If information is not correct, the test is flagged as failed and the output data is logged.

These unit testing modules will be run by a shell script. The shell script will run each of the nodes and the corresponding testing module. This shell script will ensure that only one node and its corresponding testing module will be active at the same time. This is so that other nodes that are subscribed to the input or output topics of the node do not take received data and begin running. This unit testing shell script provides us a simple way to quickly ensure that all nodes are running as intended.

8.3.2 Software-in-the-Loop Testing

Software-in-the-loop (SITL) testing is designed to test how the computer system as a whole operates without needing to observe the operation while the drone is in flight. By doing SwIL testing before trying to fly the drone with the only unit tested nodes, we reduce the risk of our drone crashing and breaking valuable components and reduce the likelihood of someone becoming injured. Two different SITL tests will need to occur to fully test our system. The first will be the SITL of the drone & ground station software, and the second will be the SITL of the Pixhawk.

For the SITL of the drone and ground station system, a testing module will provide predetermined inputs into the entire system. The software will then run as if this data was being received from the various sensors on the drone. As the system runs, the testing module will log all data published to the ROS system. The data will then be compared to the desired output of the system, and all anomalies found will be noted. Once we are able to run tests simulating a flight without incident, the software will be ready for flight testing.

For the SITL of the PixHawk, we will use an ArduPilot SITL program. This allows a simulation of a drone to carry out commands in a virtual environment. In order to ensure the commands given to the PixHawk by the control computer will not cause the drone to crash, the simulated drone will be passed output data from the drone and ground station SITL. If the simulated drone is able to operate and navigate without issue, we know that the output commands from the SITL will work properly.

8.3.3 Input Sensor Interface Testing

To ensure that our sensors properly interface with our software, we will need to test that each software node is able to detect the sensor and publish the data to the ROS. This will be done by connecting the sensor to the drone computer, starting the responsible ROS node, and running a small program that prints out each update from the sensor. This process will allow us to discover any interfacing issues and resolve them.

8.3.4 Hardware-In-the-Loop Testing

Hardware-in-the-loop (HITL) testing is an option that we considered. HITL would enable us to ensure that the drone computer and flight controller are capable of

receiving data from the input sensors and that the drone and flight controller are outputting the expected signals.

This form of testing, hardware is required to interface with each of the physical inputs and outputs of the system, and a computer program on a separate system is responsible for sending signals simulating input along this hardware and for receiving the signals. This is to ensure that the hardware can properly send and receive electrical signals as data. Though we believe that HITL testing would be beneficial to our project, we believe that time and budget constraints prevent us from being able to set up such a system. The above testing procedures should give us enough confidence to begin flight tests.

8.4 Wireless Testing (HS)

We needed verify our wireless communication systems because it will satisfy Lockheed Martin's requirements for the live video stream. Wireless testing will be done solely on the Jetson Nano. To test for wireless connectivity, a video stream will be conducted using the Jetson Nano and a web camera. The Jetson Nano will be configured to broadcast a stream of the web camera footage over IP, and a laptop connected to the same router that the Jetson Nano is connected to, will be able to grab the stream from the designated IP address.

To achieve this, the Jetson Nano will first be linked to the WiFi router. Next, the USB camera will be connected to the Jetson Nano, and will be selected via command console and configured based on the webcam's settings. Additional settings and parameters will be designated for streaming. After the settings are complete, the command will be sent to enable the video live stream of the camera.

After the stream is enabled, the ground control station laptop connects to the Jetson via secure shell (SSH) protocol using the program Putty. Using the Jetson Nano's IP address, the laptop will a web browser open with the address containing the Jetson's IP address. If the wireless connection is working, the browser will display the stream from the web camera connected to the Jetson Nano. If the wireless connection is not working, then video feed will not display on the designated IP address. Design of test is outlined in Figure 38.

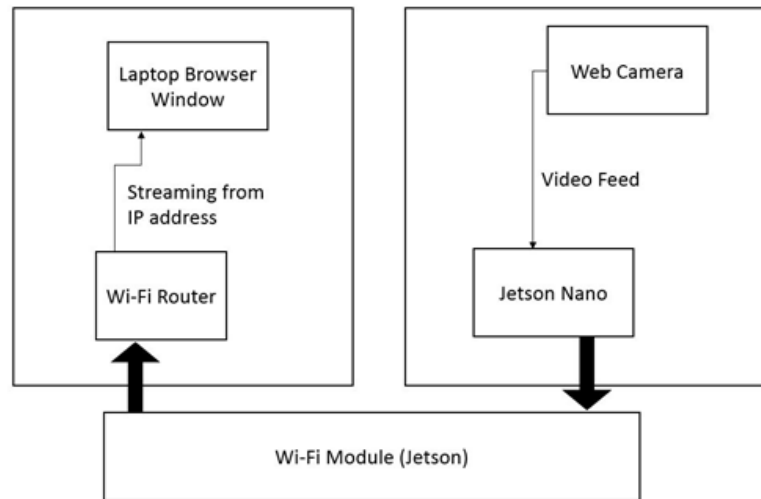


Figure 38: Wireless Testing Diagram (HS)

8.5 Ground Station Manual Control Testing (HS)

Ground Station testing will make sure that the PX4 and the Pixhawk can communicate with the ground station, enabling manual flight of the drone. To test that the ground control station can successfully connect and control a drone, the Pixhawk flight board, as well as a prototype drone or commercial off the shelf drone will be needed. Additionally, a laptop with an Xbox 360 controller will also be utilized.

Based on our original design, the system configuration will consist of the ground station laptop running the QGroundControl software, and a ROS Joystick Node on the laptop to use the Xbox 360 controller with QGroundControl. QGroundControl will enable a connection with the Pixhawk flight board to manually fly the drone using the gamepad controller. If both subsystems are configured correctly, steady and responsive flight via manual control will be established. Design of test is outlined in Figure 39.

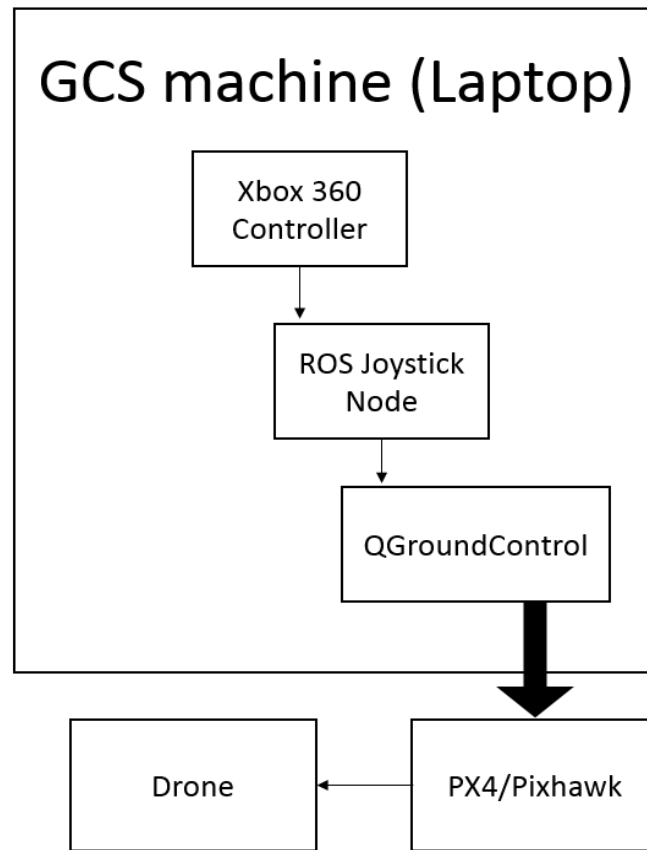


Figure 39: Ground Station Manual Control Testing Diagram (HS)

8.6 System Connectivity Prototype (HS)

To ensure that all the systems are linked and can function cohesively, a simple prototype will be built to demonstrate each subsystem such as the ground control station and the drone, in communication with one another. The prototype will focus less on moving the drone and getting accurate feedback from the camera, and rather focus on the wireless connectivity between the Jetson Nano's Wi-Fi module and the ground control station's wireless router, and the ability to receive output from the drone's camera, and input from the Xbox 360 controller on the ground control station.

The primary component that will be tested in our prototype is the Jetson Nano. Other components in this prototype include a laptop, Wi-Fi Router, USB Wi-Fi module for the Jetson Nano, and a web camera. Software used for this will primarily consist of ROS nodes, and additional software to connect the laptop to the Jetson Nano. Connection between the laptop and the Jetson Nano will be done through serial connection, using a program such as Putty to initialize a serial connection to send a serial command to the CPU via the router.

The prototype would have the Jetson Nano linked with the ground control station laptop via Wi-Fi, through the USB Wi-Fi module to the Wi-Fi Router. A key on the laptop will be pressed and a serial command will be sent to the Jetson Nano, which will turn on an LED connected to the Jetson Nano. Pressing the button again will turn the LED off. Additionally, a web camera will be connected to the Jetson Nano via USB, and will broadcast the video through the Wi-Fi Module, and this camera stream will be accessed via IP address. The setup can be seen in Figure 40. Having the LED functionality as well as the camera stream functionality will ensure that it is possible to communicate both ways between the ground control station and the drone.

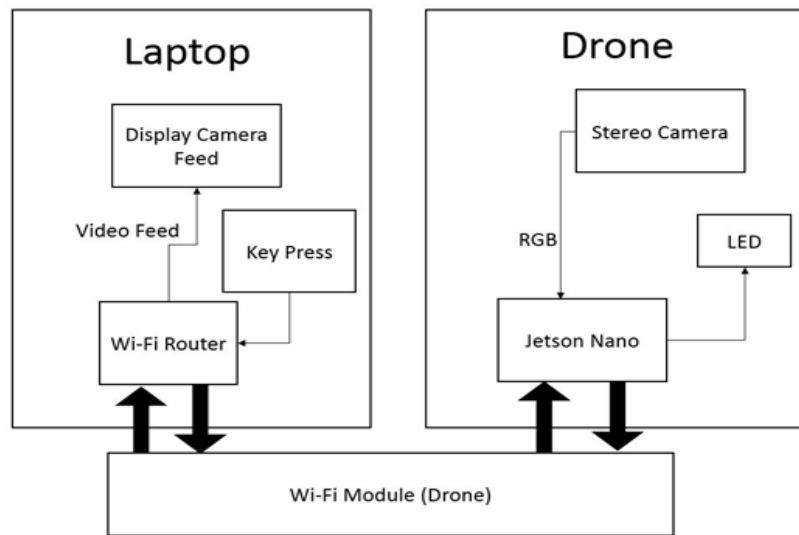


Figure 40: System Connectivity Prototype Diagram (HS)

8.7 Sound Triangulation Prototypes (HS)

To test for accuracy of pinpointing the specified sound from the acoustic waypoint, an initial and basic sound triangulation prototype using an array of three microphones will be constructed. Tests will be run using this prototype to tweak values to determine sound distance based on volume intensity, as well as sound direction based on the differentials between the three microphones. To test and verify the accuracy of this setup, the distance (between 0 and 5 meters) and angle (between 0 and 180) will be printed on the screen of the CPU.

A second prototype consisting of 4 microphones will need to be constructed after successful tests with 3 microphone array. This prototype will function similarly by triangulating sound, but will have 4 simultaneous triangulations occurring to provide a coordinate location relative to the center of the microphone array based on the location of the sound. There will be a print out of the x-coordinates and y-coordinates in meters of the sound relative to the origin, as well as a distance

and angle (0 to 359) to the sound. An example scenario of this prototype can be seen in Figure 41 below.

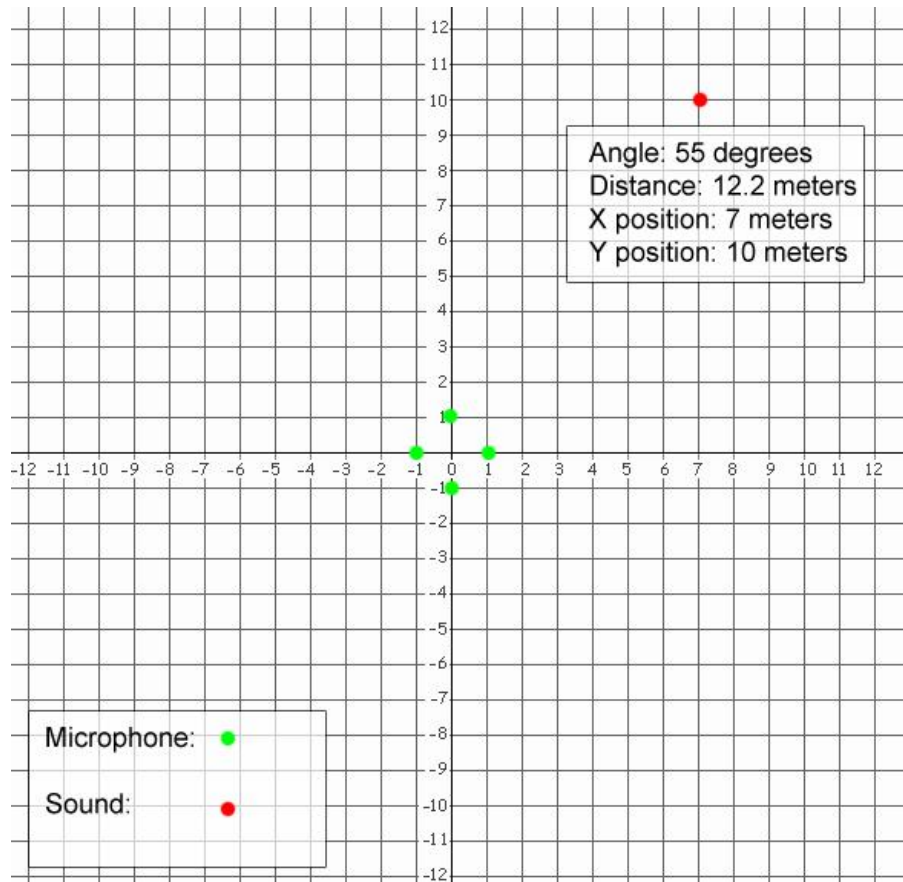


Figure 41: Sound Triangulation Prototype Example (HS)

8.8 Sound Channel Prototype (HS)

Filtering the sound of the drone will be key for locating the acoustic waypoint. To ensure this capability is functional, a prototype will be setup using the Jetson Nano connected to the ReSpeaker Mic Array v2.0. The mic array which is connected to the cpu, will listen to the designated frequency that will be emitted by the acoustic waypoint which will be played on a phone which will sit 10 feet away from the microphone array. Additionally, a loud fan will be on and running making as much noise as possible from 1 foot away from the microphone array as seen in Figure 42.

Utilizing the AEC functions of the XMOS XVF-3000 in the ReSpeaker Mic Array v2.0, two sound channels can be made from both the fan's sound and the phone's sound. Both channels will be compared to the desired frequency, and whichever channel sounds closest to the correct frequency will be designated as the sound emitting device

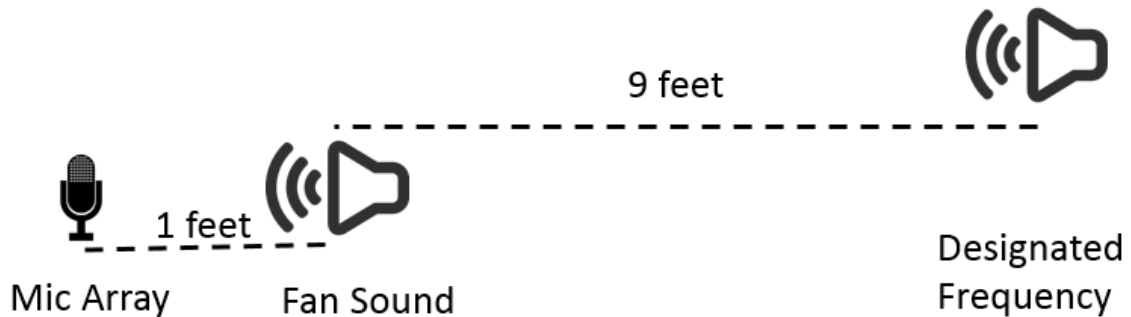


Figure 42: Diagram of sound channel prototype layout

8.9 Reliability and Performance Testing (RJ)

A major aspect of our project was include reliability and performance testing, which are inherently related to each other. Increasing the performance of the drone (such as the speed at which it flies) may decrease the reliability of the drone being able to complete a mission. A mission is defined as being able to maneuver around an identified obstacle from the course.

Moreover, there is a multiplier bonus for the number of missions that are consecutively executed successfully.

When we begin prototyping our physical drone and algorithm, we will fly the drone at a low speed to ensure that it can individually complete a mission. We will repeat this process, however, we will modify the testing conditions such as changing the starting angle of the drone in relation to the obstacle, or changing the lighting conditions of the room. This will allow us to create a better dataset for our autonomous functionality, and prepare us for the unknown environment that our drone will experience during competition day. We will seek to achieve at least 90% reliability for each individual mission. The way this will be measured is by measuring the number of successful missions out of a ten run trial. If the drone successfully completes a mission ten out of ten times during our testing, then we will increase the speed of the maneuver. If the drone is unable to meet this threshold, we will reduce the speed of the drone until we can satisfy the 90% threshold.

8.10 Adversarial Mine Avoidance Hardware (HS)

One of the obstacles our drone will need to traverse past is an adversarial mine created by another senior design team. All that is known about the adversarial mine is that it cannot contain any high velocity projectiles and it will be placed on the floor between two obstacles on the course. Our drone will conduct two runs; one without the adversarial mine and a second run with the mine present. The adversarial mine can only attack the drone within a 3 foot diameter, and 10 foot high cylinder of its origin as seen in Figure 43.

Because the nature of the adversarial mine is unclear, our drone needs to be able to withstand any potential dangers that could strike against it from the mine. Possible attacks from the mine include wind gusts, a large net, or small soft body projectiles. The mine is also limited by being placed on the ground; however, they can employ any number of sensors to detect and track our drone. Additionally, the mine team can create multiple mines that will all be placed between two arbitrary obstacles on the course. Given these potential hazards to the drone's flight on the course, there are several additions to the drone that can help mitigate a crash when the drone is under fire from the mine.

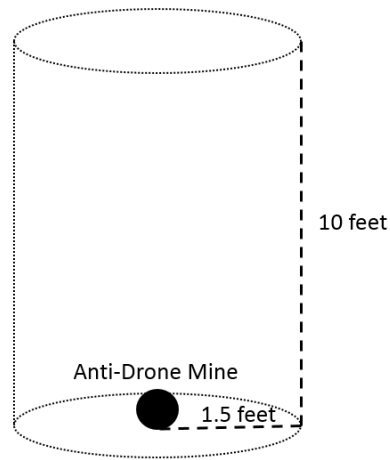


Figure 43: Adversarial mine area of effect diagram

8.10.1 Propeller Shrouds

Mentioned prior in the sound reduction section, the propeller shrouds around the propellers can help to mitigate sound and prevent any object from colliding with the propeller from its sides as seen in Figure 44. If the drone gets knocked out of the air into another object, the drone might be able to recover because the shroud allows for the drone to slide against walls without damaging the propellers. If the mine uses projectiles, this will be helpful, however if the mine uses air blasts to knock the drone out of the air, depending on the height of the shroud, it might increase the drone's overall surface area, which would allow for the air blast to be more effective against the drone.

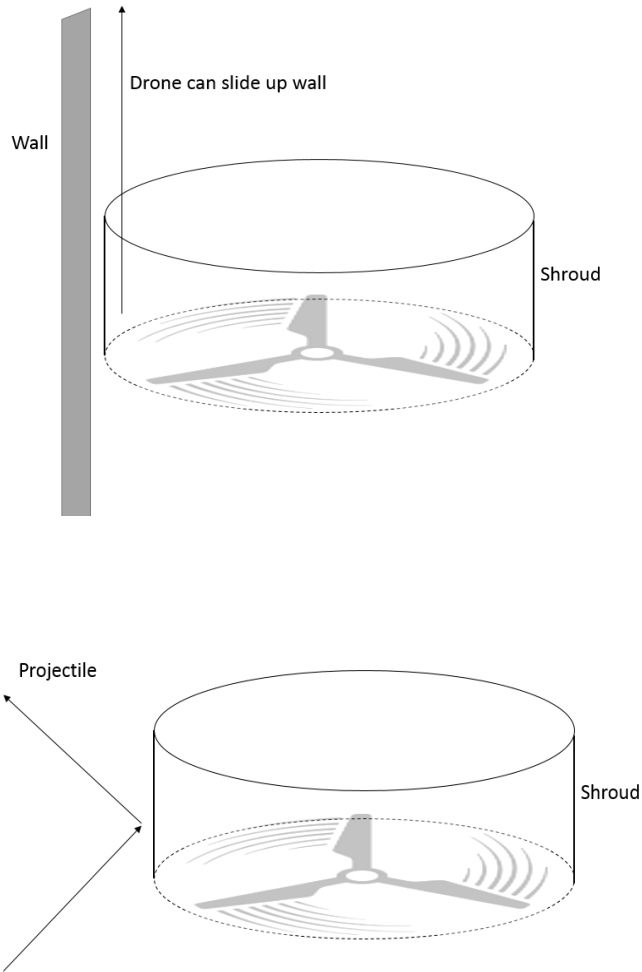


Figure 44: Propeller Shrouds gliding against wall or obstacle (Top) and deflecting projectiles (Bottom)

8.10.2 Safety Mesh

Another safety device that can be added to the drone is a wireframe designed safety mesh made of plastic surrounding the entirety of the drone structure. This will allow for the mesh to absorb the impact of any projectiles that hit the drone. The downside for this addition would be the fact that it will increase the overall size of the drone, allowing for it to be easier to hit, and make it harder for the drone to pass through and around obstacles. An example mesh that can attach to our drone can be seen below in Figure 45.



Figure 45: Example safety mesh to surround the drone

Due to the change of requirements given by our sponsor, we did not pursue any defensive strategies to protect our drone during the second round of the competition.

9.0 Administrative Content (RJ)

9.1 Deliverables

For this project, we have several deliverables that we will produce over the two semesters.

- Weekly Status Report (Lockheed Martin Requirement)
- SD1 Divide & Conquer (Senior Design Class)
- SD1 Initial Project Report (Senior Design Class)
- Preliminary Design Review Presentation (Lockheed Martin Requirement)
- SD2 CDR (Senior Design Class)
- SD2 Midterm Demo (Senior Design Class)
- Video Demo (Lockheed Martin and Senior Design Requirement)
- SD2 Final Project Report (Senior Design Class)
- Website

For our project sponsors and advisors, we had a weekly status report that we must complete, including what was accomplished during the previous week and what will be completed the following week. This document is shared with our project advisors and sponsors the day before our weekly meeting. We also must produce a presentation by Jan 2020 that includes details about the design and

implementation of our drone. Finally, our sponsor and our class requires that we submit a demo of our project.

The primary documents we must produce for the Senior Design class requirements are the Divide & Conquer, the Initial Project Report, and the Final Project Report. The Divide & Conquer is a preliminary document that outlines our requirements and architectural plans. The Initial Project Report is our plan for designing, building, and implementing our project, and it is due at the end of Senior Design 1. The Final Project Report is due at the end of Senior Design 2, and will outline everything we did to implement our project, including any changes. We originally needed to prepare a posterboard, but due to the COVID19 pandemic, that requirement was dropped. However, we still needed to produce a website.

9.2 Milestone Discussion

An important aspect to ensure the success of our project is to divide our tasks into smaller segments. This allows us to focus on smaller, achievable goals while also moving us towards the completion of this project. Creating milestones also helps our project sponsors and advisors know where This project is expected to take two semesters to complete.

Table 11 outlines our project milestones and due dates for Senior Design I, where we primarily hope to complete all of our research for the project. This includes researching the technologies available to us, and deciding how we will choose to implement our drone. These dates were also selected to meet the UCF and Lockheed Martin requirement deadlines.

During the second semester, we were given to present our preliminary design to the Lockheed Martin team, where we received feedback on our implementation.

Table 11: Senior Design I Timeline

Milestone	Completion Goal
Divide & Conquer 1	9/20/19
Create detailed requirement list	9/29/19
Divide & Conquer 2	10/4/19
Drone CPU and/or Microcontroller Research Finished	10/11/19
Order CPU and/or Microcontroller for Prototyping	10/12/19
Order Drone Starter Kit	10/16/19
Begin Data Collection & Training	10/21/19
SD1 Documentation (30 pages)	10/21/19
SD1 Documentation (60 pages)	11/1/19
Finish Algorithm Training	11/4/19
SD1 Documentation (80 pages)	11/8/19
SD1 Documentation (100 pages)	11/15/19
SD1 Final Documentation	12/2/19

Table 12 below outlined our original plan for the second semester in the senior design sequence. Senior Design 2 will primarily encompass building our physical product and testing that it works for our missions. After the initial build, we expect that most of our time will be spent tuning the algorithm to detect obstacles and to navigate through the course autonomously. We found that in several instances, we needed to redesign certain aspects of our drone as it not meeting the performance requirements that we outlined earlier.

Though our initial build burned through most of our budget, we had leftover funding that would allow us to make minor design changes as necessary. Moreover, we will be able to modify a lot of operational functionality through our code, and if the drone does not meet our performance requirements, it is possible that those issues can be resolved through software solution.

Table 12: Planned Senior Design II Timeline

Milestone	Completion Goal
Object Detection Model Training	1/6/20
Coding Modification and Simulation	1/9/20
Initial Power System Testing	1/10/20
Manual Mode Flight Accomplished	1/15/20
Preliminary Design Review Presentation	1/17/20
Land/Take-off Autonomous Submode is Completed	1/30/20
Emergency Stop (E-Stop) system Completed	2/7/20
Modified Program Simulation and Power System Recheck	1/22/20
Testing and Design Changes Phase 1	2/15/20
Laboratory Testing for Power System	2/18/20
Testing and Design Changes Phase 2	3/15/20
Final Build	4/15/20
SD2 Final Report	4/20/20
Website	4/20/20

Due to the COVID19 pandemic emerging during SD2, the original timeline was severely affected. The results of our project are outlined in section 10.

9.3 Initial Budget and Finance Discussion

Lockheed Martin, the project sponsor customer, provided a total of \$1650 dollars to our group for our autonomous drone. \$1100 of that total amount is allocated towards the materials and parts needed for the final product. The additional \$550 is allocated for prototyping and testing. While we are able to spend a total of \$1650, the total cost of the drone cannot be \$1650; it must be \$1100 or less. This budget constraint provides an engineering constraint for us, requiring us to carefully select parts that we know will be subtracted out of the \$1100 build budget. For example, though the CPU, camera, and battery are amongst the

most expensive items in our build, we were confident that they will be used in our final product.

For this reason, we were significantly under budget at the beginning of prototyping. This provided wiggle room in case certain parts turn out to be more expensive than expected, or replacements are needed. Replacement parts might be needed if we damage them during testing, or if we find out that they provide poor performance that is not adequate for our mission. This allowed for part of the budget to go into research and experimentation with potential sensors, propellers, drone frames, CPUs, and flight boards. Below in Table 13 was our estimated budget for this project.

Table 13: Estimated Budget

Part	Estimated Price
CM-2206/17-V2 MULTIROTOR MOTOR (4)	\$91.96 (\$22.99 ea.)
HQ Prop 5x3 Propellers (4)	\$1.96 (\$0.49 ea.)
3D Printed PETG Drone Body	\$7.59
Intel RealSense D435 Depth Camera	\$179.00
NVIDIA Jetson Nano	\$99.00
Geekworm Jetson Nano WiFi	\$18.99
3DR Radio Telemetry Air and Ground Data Transmit Module	\$22.99
Cobra 30A Opto Multicopter ESC (4)	\$111.96 (\$27.99 ea.)
HC-SR04 Ultrasonic Range Sensor (4)	\$15.80 (\$3.95 ea.)
Holybro PX4FLOW Optical Flow Camera	\$109.99
Venom 4s 30c Battery	\$59.99
ReSpeaker Mic Array v2.0	\$64.99
Custom PCB	~\$10.00
Electrical Components (transistors, resistors, capacitors, ICs, etc)	~\$10.00
TOTAL	\$804.22

9.4 Advisors, Meetings, and Communications

This project consists of a multidisciplinary team which includes two electrical engineering (Rishi Jain and Ryan Lucas), two computer engineering (Caleb Jones and Hamza Siddiqui), two mechanical engineering (Moneer Hajiazimi and Chad Bement), and two aerospace engineering (Christopher Rehberg and Joseph Rice) students.

Additionally, our sponsor has given us four advisors that are available to assist us when needed. Two of the advisors, Jonathan Tucker and Andrew Kirk, act in the role of the “customer”. They are our target audience for whom we are building the drone for. Our overall team lead, Christopher, has a weekly call with them to provide status updates and to clarify any customer requirements. Two other advisors, Aaron Phu and George Loubimov, act as our project mentors. They are resources for us to utilize during the research, construction, and implementation of our drone. Finally, we have access to our Senior Design class professors, Dr. Richie and Dr. Wei, that we can contact for specific electrical and computer engineering questions, as well as general concerns relating to the Senior Design courses.

All group members and our project mentors meet weekly to discuss project progress, roadblocks, and major decisions that require approval from both subteams. Additionally, the MAE and ECE subteams frequently break off into smaller groups in order to tackle specific goals.

All group members and project mentors have access to a communication platform called Slack, which allows us to communicate and easily share information with each other. Moreover, most joint assignments are hosted on Google Drive, allowing group members to easily contribute and view the overall progress of a specific goal.

9.5 Parts Acquisition

Since our project is being financed by Lockheed Martin, there is a specific process for acquiring parts. First, we had to decide as an entire team which parts we are going to order. The reason is because most systems are interconnected with each other or may impact the weight/aerodynamics of the drone. Once we reach unanimous agreement on the parts, our team leader will fill out an MAE department order form. Once the form submitted, we will have to inform our project advisor to review the order and approve it. If the advisor disagrees with something decided on, then we will have to reevaluate whether we should pursue a different part. If the advisor agrees, then the parts will be ordered and we will receive an email confirmation.

We also have limited access to parts used on previous Lockheed Martin sponsored projects. We received a PX4flow, a Pixhawk-variant flight controller, and a Nvidia Tegra K1 CPU that we could use for our project. While we did not

have plans to utilize these components since they do not fit within our original architectural design plan, it gives the opportunity to test other components if we find that our chosen components fall short of our performance requirements.

10.0 Project Results, Major Changes, and Future Considerations

Although previous sections of this paper outline our design considerations for the original scope of the project, we experienced varying levels of success and needed to modify some of our design plans. Moreover, the emergence of the COVID19 pandemic and changing requirements from our sponsor abruptly altered our focus in certain areas of the project. This section summarizes: major changes (both completed and pending) made to the software and hardware of the drone; the functionality achieved by our final product; and our final costs.

10.1 Hardware

10.1.1 Drone Frame

For the drone frame, we were originally planning on developing anti-mine hardware (such as propeller guards and a mesh net) for use in the competition. However, since the competition was cancelled, this was no longer a priority. In the end, our multidisciplinary team created an in-house frame that was able to house all of our components.

10.1.2 Optical Flow/Height Sensor

Although we were originally planning on using the PX4Flow Optical Flow sensor, we were not able to stabilize the drone using position hold, which would keep a drone stationary in the air. We replaced the Optical Flow sensor with a HereFlow Optical Flow/Lidar sensor due to its increased vertical range. However, we only found modest improvements in its ability to perform in the outdoors. Indoors, the HereFlow seems to be stable but we were unable to extensively test this due to lack of access to Lockheed Martin's drone facility and stay-at-home orders implemented throughout the state.

10.1.3 Microcontroller/PCB

We successfully designed and tested a schematic for our PCB to manage ultrasonic range sensors to convey distance measurements to objects. Although the fabricated PCB was functional, it experienced some signal interference inside the board making two of the sensor measurements unreliable. We were not able to properly debug this issue due to the lack of access to the senior design lab. In a future iteration of the PCB, we would also include a logic level converter built into the board to reduce the profile of components on the drone.

10.1.4 Power Distribution

Our power distribution board was rated to support 5V @ 6A, however, we experienced intermittent power failures where our Jetson Nano would shut off during high load. We were not able to debug this issue in the lab environment due to campus closures, but as a temporary solution, we were able to manually set the Jetson Nano into a reduced power mode (5W). Although computing performance was reduced, the computer was able to remain powered on and we were able to demonstrate a proof of concept for our project. This could be easily remedied by acquiring a replacement voltage regulator.

10.1.5 Electronic Speed Controllers

The original (Cobra) ESCs that we selected for our project were compatible and provided acceptable performance in our initial testing; however, we found that one of them was nonoperational. We were not able to source Cobra ESCs from the internet in a timely fashion, so we elected to buy DSHOT ESCs locally. Instead of using a PWM signal to drive the motors, the new ESCs were digital and provided better performance in terms of latency.

10.2 Software

We were able to successfully build our dataset and train an image recognition model. An example of our object detection is shown in Figure 46. Additionally, we were able to extract sound measurements to guide our drone.

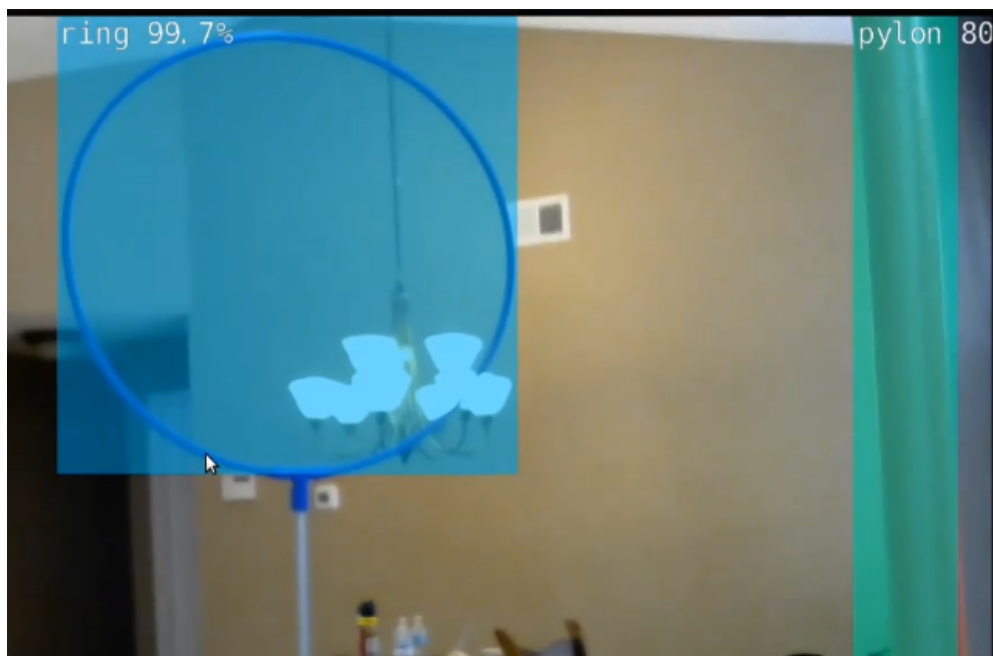


Figure 46: Custom Dataset To Detect Rings and Pylons

Our original intention was to use ROS as a basis for all of the software. While it worked well to extract image information and manage other sensors, we had difficulty using ROS to control the flight controller using the MAVLINK protocol. In our attempts to debug our problem, we stumbled upon a framework called DroneKit, which we used to send commands to our flight controller. DroneKit python library allowed us to easily write programs to direct the drone based on inputs we defined. Therefore, our project uses both ROS and DroneKit, rather than just the former.

10.3 Overall Drone Functionality

Our completed drone is shown below in Figure 47.



Figure 47: Fully assembled drone

We were able to implement object detection and tracking, navigation using sound triangulation, rudimentary autonomous flight, and a live video datafeed.

Please refer to the video demo to view the drone in operation.

10.3 Final Cost

Due to the changes outlined above, the final assembled cost of our drone had changed. The bill of materials was cheaper for the final build was cheaper than our prototype build, primarily due to the cheaper ESCs and HereFlow sensor. Table 14 displays the components and cost of our final product that we would present if competing at Lockheed Martin. However in reality, some components such as the Pixhawk were free to us.

Table 14: Cost of Final Drone Build

Component	Name	Unit Cost	Quantity	Cost
Jetson Power Cable	Adafruit 5V 4A Supply	\$14.95	1	\$14.95
Battery	Venom 4s 30c 3200mah14.8V LiPo battery	\$59.99	1	\$59.99
Drone Wifi Module	Geekworm NVIDIA Jetson Nano Wi-Fi Adapter	\$16.79	1	\$16.79
Companion Computer	NVIDIA Jetson Nano Developer Kit	\$99.00	1	\$99.00
Flight Controller	ReadyToSky Pixhawk	\$74.99	1	\$74.99
Optical Flow/Lidar	HereFlow Optical Flow/Lidar Sensor	\$49.99	1	\$49.99
Power Distribution Board	PDB XT60 Matek Power Distribution Board	\$8.49	1	\$8.49
ESCs	Spedix GS30 32bit DShot 1200 30A ESC	\$12.95	4	\$51.80
Propellers	HQ Prop 5x3 Propellor (Black) (4)	\$0.49	4	\$1.96
PCB	Custom Microcontroller BOM	\$9.40	1	\$9.40
Ultrasonic Sensors	HC-SR04	\$3.95	4	\$15.80
Motors	CM-2206/17-V2 MULTIROTOR MOTOR KV=2400	\$22.99	4	\$91.96
Microphone	ReSpeaker Mic Array v2.0	\$64.00	1	\$64.00
Mounting	Frame, Double Sided Tape, M3 Nuts/Bolts/Standoffs, Cable Ties, Battery Clip	\$26.71	1	\$26.71
SD Card	MicoSDXC	\$19.49	1	\$19.49
Depth Camera	Intel RealSense D435	\$177	1	\$177.00
			Total:	\$782.32

Table 15 outlines the cost spent strictly on prototyping, in other words, parts we received but did not include in the final build.

Table 15: Final Prototyping Costs

Component	Name	Unit Cost	Quantity	Cost
Optical flow camera	PX4 Flow (Donated)	\$0	1	\$0
Electronic Speed Controller	Cobra MR30	\$27.99	4	\$111.96
Miscellaneous PCB Components	PCB Prototype and Mouser Components	\$27.78	1	\$27.78
PPM Encoder	ShareGoo 8CH PPM Encoder & I2C Splitter	\$13.99	1	\$13.99
			Total:	\$153.73

11.0 Summary

We were able to demonstrate a competent design for an autonomous aerial drone. Although we consider some of the performance of our drone to be “unpolished” compared to our original expectations, we were severely impacted by the COVID19 pandemic in our ability to order parts due to the university closure, congregate as a team to debug issues due to the stay-at-home orders, test parts due to the closure of the ECE lab, and to test fly our drone in an appropriate environment as Lockheed Martin’s drone facility became unexpectedly unavailable to us. Nevertheless, we were able to work in a multidisciplinary team in order to deliver a functional autonomous drone which serves as a proof of concept for our design methodology. We developed a greater understanding for the importance of research, collaboration and project management in an ambitious project such as this.

12.0 Appendices

12.1 Bibliography

- [1] A Short History of Unmanned Aerial Vehicles
<https://consortiq.com/media-centre/blog/short-history-unmanned-aerial-vehicles-uavs>

- [2] ArduPilot TX1-PixHawk Interface
<http://ardupilot.org/dev/docs/companion-computer-nvidia-tx1.html#companion-computer-nvidia-tx1>
Used info licensed under Creative Commons [Attribution-ShareAlike 3.0 Unported](https://creativecommons.org/licenses/by-sa/3.0/)

- [3] California RoHS
<https://dtsc.ca.gov/restrictions-on-the-use-of-certain-hazardous-substances-rohs-in-electronic-devices/>

- [4] Choosing A Ground Station
<http://ardupilot.org/copter/docs/common-choosing-a-ground-station.html>

- [5] Coding Standards and Guidelines
<https://www.geeksforgeeks.org/coding-standards-and-guidelines/>

- [6] CPU Performance Comparison of OpenCV and other Deep Learning frameworks by Satya Mallick
<https://www.learnopencv.com/cpu-performance-comparison-of-opencv-and-other-deep-learning-frameworks/>

- [7] CUI INC Power Supply Safety Standards, Agencies, and Marks
<https://www.cui.com/catalog/resource/power-supply-safety-standards-agencies-and-marks.pdf>

- [8] Drones and Unmanned Aerial Vehicles (UAV) Certification (UL3030)
<https://www.ul.com/offerings/drones-and-unmanned-aerial-vehicle-uav-certification>

- [9] Electronic Speed Controller (ESC) Calibration
<http://ardupilot.org/copter/docs/esc-calibration.html>

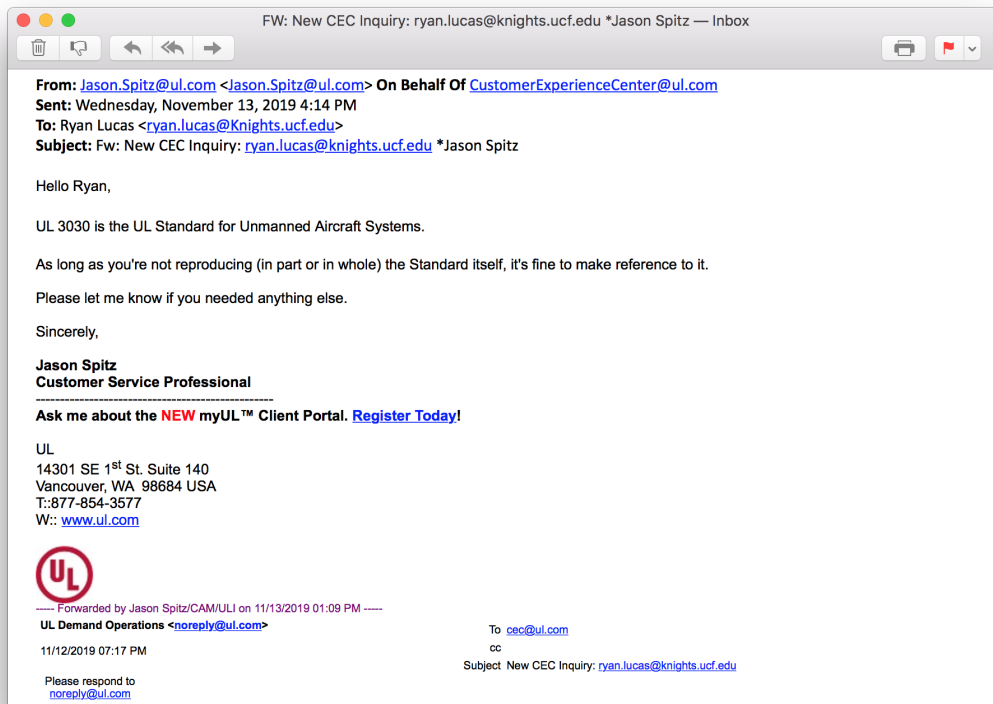
- [10] Ethical Issues With Use of Drone Aircraft
<https://csiflabs.cs.ucdavis.edu/~ssdavis/188/Ethical%20Issues%20with%20Use%20of%20Drone%20Aircraft.pdf>

- [11] How to connect the power system (ESC, BEC and power board) in your multirotor/quadcopter by Painless360
<https://www.youtube.com/watch?v=AIQOzFILfJg>
- [12] How to install RealSense ROS node on Jetson Nano (recent as October 2019)
<https://github.com/JetsonHacksNano/installRealSenseROS>
- [13] IEEE Standards Association
https://standards.ieee.org/project/2025_1.html
- [14] Intel D400 Datasheet
https://www.mouser.com/pdfdocs/Intel_D400_Series_Datasheet.pdf
- [15] Intel RealSense D400 Series Cameras Calibration
<https://dev.intelrealsense.com/docs/intel-realsense-tm-d400-series-calibration-tools-user-guide>
- [16] MicLoc Sound Detection
<http://ruralhacker.blogspot.com/p/micloc.html>
- [17] NVIDIA Trail Drone
<https://arxiv.org/pdf/1705.02550.pdf>
- [18] Raspberry Pi 4 vs Jetson Nano Machine Learning Benchmarking:
<https://www.hackster.io/news/benchmarking-machine-learning-on-the-new-raspberry-pi-4-model-b-88db9304ce4>
- [19] RealSense ROS node
<https://github.com/IntelRealSense/realsense-ros#step-3-install-intel-realsense-ros-from-sources>
Licensed under Apache License 2.0
- [20] Recreational Flyers FAA
https://www.faa.gov/uas/recreational_fliers/
- [21] Response from Intel confirming most users are able to use RealSense ROS Node with ROS Melodic
https://forums.intel.com/s/question/0D50P00004FD93E/can-i-use-my-intel-realsense-d435-with-ros-melodic-for-ubuntu-18042?language=en_US
- [22] ROS Concepts (includes figure)
<http://wiki.ros.org/ROS/Concepts>
Wiki is licensed under Creative Commons Attribution 3.0

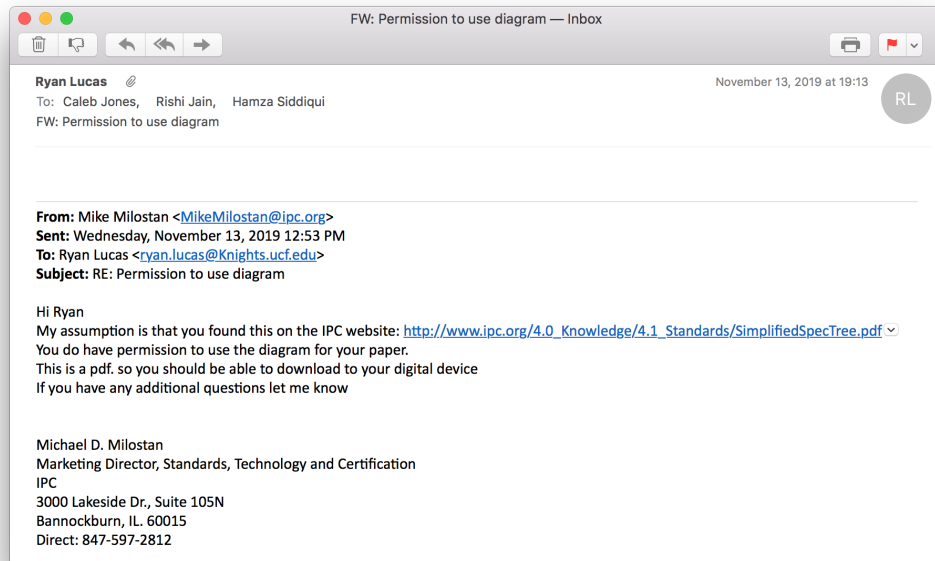
- [23] Sensor Setup (Ardupilot)
https://docs.qgroundcontrol.com/en/SetupView/sensors_ardupilot.html
- [24] The History and Basics of IPC Standards
[https://www.allaboutcircuits.com/news/ipc-standards-the-official-standards-for-pcbs/###targetText=IPC%20standards%20are%20the%20electronics,assembly%20\(see%20Figure%201\).](https://www.allaboutcircuits.com/news/ipc-standards-the-official-standards-for-pcbs/###targetText=IPC%20standards%20are%20the%20electronics,assembly%20(see%20Figure%201).)
- [25] The History of Drones
<https://www.dronethusiast.com/history-of-drones/>

12.2 Copyright Permissions

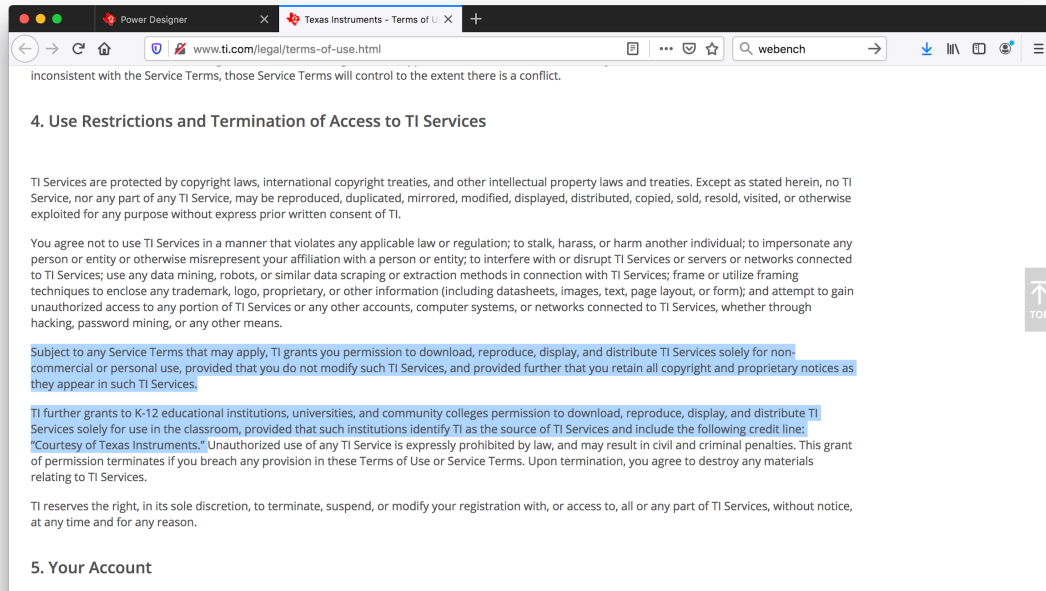
[A] UL Permission



[B] IPC Permission



[C] Texas Instruments Permission



[D] Painless360 Permission

From: Lee Schofield <lee@painless360.com>
Sent: Tuesday, November 26, 2019 1:40:41 AM
To: Ryan Lucas <ryan.lucas@unf.edu>
Subject: Re: Permission to use a diagram from your video

Hi Ryan,

That's fine. Let me know if you need anything else.

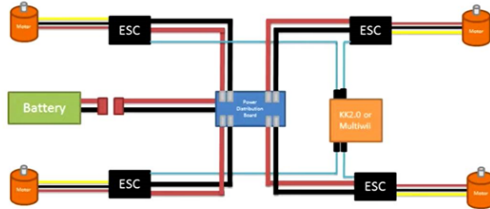
Best of luck with the paper!

Lee

From: Ryan Lucas
Sent: Monday, November 25, 2019 1:48 AM
To: lee@painless360.com
Subject: Permission to use a diagram from your video

Dear Painless 360 Personnel,

I am a student at the University of Central Florida and am currently doing my Senior Design project paper. I came across a diagram on your YouTube video, "How to connect the power system (ESC, BEC and power board) in your multibrotos/quadcopter", that I would like to use on my paper. I am writing to you today to ask for your permission to use the diagram which will be properly cited on my paper, and all credits for the diagram will be given to you, your group, or your organization. Please let me know if you would allow me to use the diagram; a flowchart of power distribution to ESC from PDB and battery. I have attached a copy of the diagram I am interested in below:




Looking forward to hearing from you soon.

Thank you,

[E] Intel Permission

← → ↻ 🏠 <https://www.intel.com/content/www/us/en/legal/terms-of-use.html> 📄 ⋮ 🌟

Products Solutions Support 

FURTHER REPRODUCTION OR REDISTRIBUTION IS EXPRESSLY PROHIBITED, UNLESS SUCH REPRODUCTION OR REDISTRIBUTION IS EXPRESSLY PERMITTED BY THE LICENSE AGREEMENT ACCOMPANYING SUCH SOFTWARE. THE SOFTWARE IS WARRANTED, IF AT ALL, ONLY ACCORDING TO THE TERMS OF THE LICENSE AGREEMENT. EXCEPT AS WARRANTED IN THE LICENSE AGREEMENT, INTEL HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE SOFTWARE, INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT.

FOR YOUR CONVENIENCE, INTEL MAY MAKE AVAILABLE AS PART OF THE WEB SITE OR IN ITS SOFTWARE PRODUCTS, TOOLS AND UTILITIES FOR USE AND/OR DOWNLOAD. INTEL DOES NOT MAKE ANY ASSURANCES WITH REGARD TO THE ACCURACY OF THE RESULTS OR OUTPUT THAT DERIVES FROM SUCH USE OF ANY SUCH TOOLS AND UTILITIES. PLEASE RESPECT THE INTELLECTUAL PROPERTY RIGHTS OF OTHERS WHEN USING THE TOOLS AND UTILITIES MADE AVAILABLE ON THE SERVICES OR IN INTEL SOFTWARE PRODUCTS.

Notice Specific to Intel Documents

Permission to use Intel owned white papers, press releases, datasheets, specification documents, FAQs etc. ("Documents") from the Web Sites is granted, provided that (1) the below copyright notice appears in all copies and that both the copyright notice and this permission notice appear, (2) use of such Documents from the Services is for informational and non-commercial or personal use only and will not be copied or posted on any network computer or broadcast in any media, and (3) no modifications of any Documents are made. **Accredited educational institutions, such as K-12, universities, private/public colleges, and state community colleges, may download and reproduce the Documents for distribution in the classroom.** Distribution outside the classroom requires express written permission. Use for any other purpose is expressly prohibited by law, and may result in severe civil and criminal penalties. Violators will be prosecuted to the maximum extent possible.

Documents specified above do not include the design or layout of the Web Site or any other Intel owned, operated, licensed or controlled site. Elements of the Web Sites are protected by trade dress, trademark, unfair competition, and other laws and may not be copied or imitated in whole or in part. No logo, graphic, sound or image from any Web Site may be copied or retransmitted unless expressly permitted by Intel.

INTEL MAKES NO REPRESENTATIONS ABOUT THE SUITABILITY OF THE INFORMATION CONTAINED IN THE DOCUMENTS AND RELATED GRAPHICS PUBLISHED ON THE WEB SITE FOR ANY PURPOSE. ALL SUCH DOCUMENTS AND RELATED GRAPHICS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. INTEL HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS INFORMATION, INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL INTEL AND/OR ITS RESPECTIVE SUPPLIERS BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF INFORMATION AVAILABLE FROM THE SERVICES.

THE DOCUMENTS AND RELATED GRAPHICS PUBLISHED ON THE WEB SITE COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN. INTEL MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED AT ANY TIME.