Team 4 - Rescue Drone
Final Report

Members

Alexandra Borgesen	alb13m@my.fsu.edu
Peter Burchell	prb08@my.fsu.edu
Cody Campbell	cjc13j@my.fsu.edu
Shawn Cho	hc11c@my.fsu.edu
Sarah Hood	sah13h@my.fsu.edu
Halil Yonter	hy14c@my.fsu.edu

Sponsor
Mr. David F. Merrick

Faculty Advisor
Dr. Rodney Roberts

Instructor
Dr. Jerris Hooker

Reviewers
Dr. Bruce Harvey
Dr. Simon Foo

04/21/2017

Executive Summary

UAVs used by Florida State University's Emergency Management and Homeland Security Program can autonomously scan an area, but will provide no feedback regarding image contents, nor do they have a user-friendly interface for interprocess communication. The multidisciplinary ECE Senior Design Team #4 was tasked with creating a new, unique UAV capable of scanning disaster zones and identifying unique objects of interest.

Careful research and planning has led to an innovative flight control architecture. The final product features a powerful onboard computer capable of live image processing for object detection, with distinct algorithms for color filtering and pedestrian tracking. A conversion algorithm was also implemented for converting the UAV's latitude and longitude data, which is read from the flight control hardware, into USNG format. An IP network governs all communication between the ground station and the UAV.

In the pursuit of increasing autonomy and implementing computer vision, a reliable and consistent object detection remains integral to accomplishing this task. By analyzing an image in search of HSV values that satisfy a predetermined range of color, any region of pixels that comply with the given range is highlighted using the color filtering algorithm. The range is preprocessed in HTML by selecting a value in an interactive color spectrum. This value auto populates the necessary variables on the image processing server, which results in a fluid transition between frames on the Logitech C920 as it detects the color of interest as it corresponds with the input variable data.

In order to accomplish the desired 18-minute flight time and address the sponsor's request for a portable design, the mechanical structure of the UAV consists of a foldable hex frame with onboard electronics powered by a four-cell LiPo battery. Furthermore, a lightweight hexacopter frame was assembled with brushless direct current (BLDC) motors and slow-fly (SF) propellers to maximize the UAV's efficiency, with the added benefit of an increased flight time. A complementary landing gear was also designed to reduce the weight and folded dimensions of the airframe. The UAV is equipped with an RC transmitter which allows manual missions and acts as a fail-safe during autonomous missions. The user can plan the autonomous mission and deploy the route to the UAV using any preferred mission planning software.

The resulting product, named by ECE Team 4 as *Saurus*, increases the ability to conduct reliable and efficient search and rescue missions by eliminating manual processing in favor of increased autonomy for the UAV's mechanical structure, and capable computer vision for real-time image processing.

Table of Contents

Table of Figures

Table of Tables

Table of Equations

1. Introduction

a. Problem Statement

UAVs play an essential role in the immediate recovery efforts of natural disasters. In addition to being a major threat to human life and development, the destructive tendencies of natural disasters impose a haunting reality that endangers human safety and mobility in regions suffering from widespread destruction. The rescue drone is a senior design project sponsored by the Florida State University's Emergency Management and Homeland Security Program.

The UAV is capable of completing search and rescue missions with features intended to not only enhance the success rate of search and rescue missions, but also automate the most time-consuming steps in what is often a time sensitive process, including image processing.

The problem statement may be summarized as: "Current methods employed by the project sponsor do not sport the desired quality of flight or transmission efficiency. The shortcomings are particularly apparent when handling photographic images transmitted by UAVs currently in use, as these images need to be manually evaluated one by one and important details risk being overlooked. A more advanced and capable UAV is necessary to simplify the process."

The objective was to create an autonomous, multirotor aircraft that could scan a designated area for objects. The identified objects should then be reported to a ground station with information that also pertains to the object's location. This project was specifically requested and sponsored by the FSU EMHS Program. The request was to provide an aircraft that could be reproducible, easily repairable, and user friendly. FSU EMHS seeks to deploy such a vehicle in contexts ranging from local to state needs.

A formal needs statement was to "build an innovative UAV capable of completing search and rescue missions in unsafe regions following natural disasters."

b. Operating Environment

This UAV is intended to fly during daylight hours, in good weather, and in compliance with local, state, and federal regulations.

c. Intended Use

Sponsor, David Merrick, hopes to see this technology available and used by Emergency Management and Homeland Security departments in all Florida counties. This UAV's technology would reduce the risk of sending out a human based search committee as well as increase the efficiency in a search and rescue mission.

d. Assumptions and Limitations

The desired specifications of the autonomous UAV were divided into needs and wants. The listed “needs” were the main objective for the design of the product, while the “wants” dictated our goals for further development. Ultimately, the primary goal was to have a deliverable search and rescue UAV by April 2017 that met or exceeded project expectations. All project objectives were ultimately accomplished.

i. Objectives

- Multirotor Aircraft
- Autonomous flight based on user designated path
- Flight time of minimum 18 minutes
- Identify particular object
- Carry photometrics; sensors
- Able to communicate with a ground station
- Output location data using USNG coordinates
- Reproducible vehicle design based on construction documentation
- Includes concise user manual
- Two axis gimbal for camera

ii. Goals

- Flight time closer to 30 minutes
- Use of IP network for the communication of data
- Autonomous collection of stand-out data
- Autonomous location logging of stand-out-data
- Fits in the sponsor’s backpack (50 L Weekend Pack)

This project was completed with consideration of and compliance to local, state, and federal regulations, codes of safety, conduct, and ethics as prescribed by FAMU-FSU College of Engineering, FSU department of EMHS, as well as the code of conduct agreed upon, signed, and submitted by all project participants.

e. End Product and Other Deliverables

The final product is a fully equipped UAV capable of flight and color recognition. NAVSTIK Labs in India released confidential software for the team’s exclusive application, which made the UAV flight control architecture compatible with the Nvidia TX1. FlytOS comes preloaded with a native mission control interface, however UDP capabilities of the network

allows the user to interact with preferred mission planner, including QGroundControl and Mission Planner.

A user manual that contains instructions to operate the vehicle and to reproduce in the future can be found in Appendix C.

f. Acknowledgements

Electrical/Computer Engineering Senior Design Team 4 would like to acknowledge the Florida State University Emergency Management and Homeland Security Program for its generous contribution to this project's development.

Additionally, the guidance of Dr. Hooker, Dr. Harvey, and Dr. Roberts has been influential in bringing this project to fruition.

2. System Design

This section covers aircraft design and performance specification. These shall be considered separately.

a. Design Specification

i. Mechanical Specification

The sponsor required the drone have a multi-rotor design that is easily collapsible for transport by backpack. The vehicle must carry the necessary power sources, actuators, sensors, processors, and communication equipment for autonomous flight and satisfaction of any computing specifications.

Use of a multirotor platform for this UAV should deliver a vehicle of predictable flight characteristics in a variety of environmental conditions. The drone should also be easy to operate, and require only a small clearing for launch and recovery.

Flight stability for this type of UAV is handled by an onboard flight controller. With the control mechanism placed on the vehicle it becomes easy to control, even in unstable wind conditions. Automated deployment and recovery are also possible using the flight controller, but local conditions often make use of these features inadvisable.

The dimensions requested by the sponsor is capable of fitting in a pelican hard case. These are of various sizes and ideally, EMHS department would prefer to have the folded vehicle dimensions capable of fitting inside a "backpack" which has been further defined as a 50 L weekend pack.

Some system components were provided by our sponsor. The supplied brushless direct current (BLDC) motors are capable of producing 14 lbs. of thrust collectively allowing for an all-up vehicle weight of 8 lbs. Weight savings were a primary goal in design for the purpose of maximizing flight time.

Propulsion is the largest consumer of power on this vehicle. 250 W for static flight is likely. Onboard computing is also power intensive. Based on power requirements of computing resources under consideration, 50 W were budgeted for onboard processing. Lithium Polymer batteries were utilized due to their high-energy density and availability.

Materials for use in this vehicle needed to be light in weight and durable. Large use of prefabricated carbon fiber reinforced polymer (CFRP), aluminum tubing, and stock fasteners made production and reproduction of this UAV much more feasible, while keeping costs down.

ii. Computer Specification

The sponsor required the aircraft have autonomous flight capability and the ability to navigate through user-determined waypoints. He specified two controllers for this purpose; the Pixhawk, and the ArduPilot APM. These controllers were provided by the sponsor.

The multi-rotor aircraft must be able to process images and video from an onboard camera. The processing of the images involves detection of unique objects in various environments. When a unique object is detected, it is indicated with a red box drawn around the object in the live video feed, the location of the vehicle is displayed in United States National Grid (USNG) coordinates. This image processing is executed by a processor capable of the necessary computations, while also maintaining power consumption under the maximum 50 W budgeted.

b. Performance Specifications

i. Mechanical Specifications

A significant quantity of the components for this vehicle were provided by the sponsor. Thus, performance specifications are based on the provided components and the sponsor's needs.

- Maximum Total Weight: 8 lbs. (3.076 kg)
- Cruise Speed: 20 kts (10 m/s)
- Flight Time: 18 to 20 min
- Maximum Power for Onboard Computing: 50 W

ii. Computer Specifications

The performance of the onboard electronics. the communication systems and the power source constitute the non-mechanical performance of the aircraft and are specified below:

- Min IP communication range: 0.5 km
- Onboard computer max power consumption: 50 W
- Real-time image processing with frame rate of 1 frame per second at a quality of 1280 x 720 pixels, using the JPEG compression algorithm.

c. Early Approaches and Initial Considerations

i. Mechanical Aspects

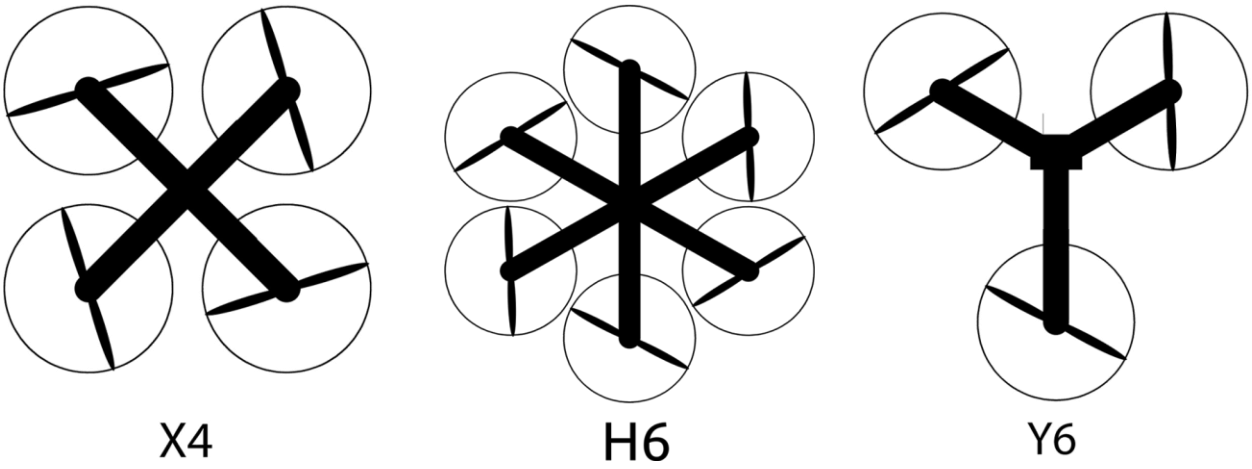


Figure 1 - Vehicle Configurations

Airframe dimensions were based on the storage and transport defined by the project's sponsor. Vehicle layouts considered were as Figure 1: a 4-rotor (X6), Hexacopter configuration (H6), Y configuration (Y6). The final configuration would be determined by vehicle weight. Ideally the armatures of the craft would detach easily for storage, transport, and serviceability, as shown in Figure 2. Materials considered for airframe construction were CFRP, fiberglass, aluminum, and plastic. Larger components would be made of prefabricated CFRP shapes; small or unique structures would be made of aluminum. Where electrical isolation was required fiberglass or plastic would be used.

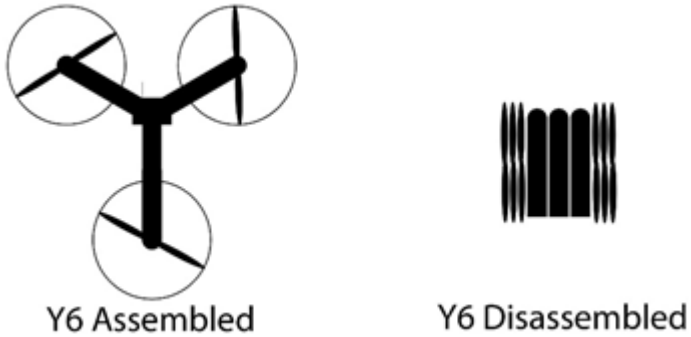


Figure 2 - Vehicle comparison of assembled and disassembled state

Components determined to be integral to the UAV were motor, electronic speed controller (ESC), flight controller, global positioning system (GPS), 3-axis gimbal, radio control (RC) receiver, ultimate battery eliminator circuit (UBEC), camera, image processing system (IPS), internet protocol communication link (IPCL), and battery. Anticipated interconnectivity is shown in Figure 3, and concept of general layout in Figure 4.

Figure 3 - Component interconnectivity

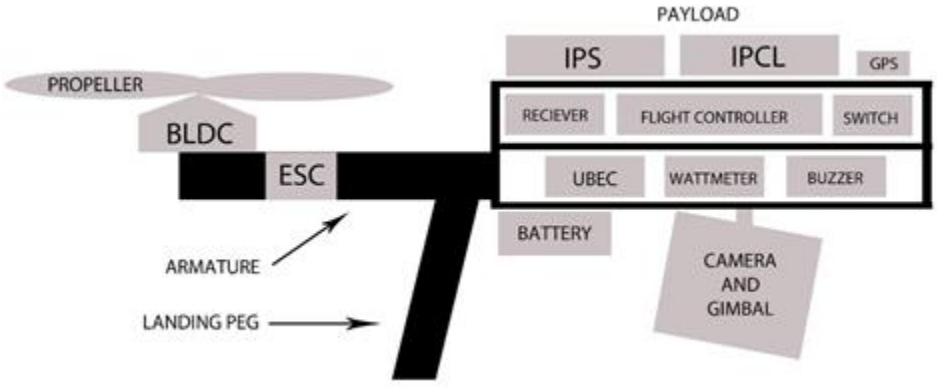


Figure 4 - General Layout with Duplicate Components Removed for Clarity

The power system consisting of a lithium polymer (LiPo) primary battery for chassis voltage, A UBEC providing low voltage (5 V or 12 V) to the flight controller and its satellites, RC receiver, gimbal, camera, IPS, and IPCL.

i. Computer Aspects

- Hardware

The majority of the flight hardware, including the flight controller and motor speed controllers, were provided by the sponsor with the request that they should be used due to their flexibility and the sponsor's familiarity with the equipment. Given these components, additional processing was applied towards the image processing. The flexibility of the given

components meant almost any chosen processor will easily communicate with the remainder of the components.

Image processing is a highly computationally intensive task. While airborne, the UAV requires a fair amount of processing power to complete the mission. The primary constraints when picking a processor for use on the UAV were size, weight, power consumption, and processing power. Along with these constraints, it was also necessary to prioritize a processor that is compatible with the available computer vision software. This permits the freedom of applying modifications whenever necessary. Given these constraints, NVIDIA TX1, NVIDIA TK1, Odroid XU4, and the Raspberry Pi 3 were under consideration as the onboard computer.

- Software

After the initial research, the team decided to further investigate the following three options for use in the final product.

The Robot Operation System (ROS) is a collection of software libraries and tools that can be used in the development of robot applications [1]. The entire framework is open source and is maintained by the Open Source Robotics Foundation (OSRF). Even though ROS wasn't developed with UAVs in mind, recent developments ensure it can now interact with the flight controllers or run on the companion computer as a standalone unit. It requires a Linux based single board computer for deployment and offers object identification and avoidance solutions. Figure 5 provides the basic workflow of image processing procedure using ROS.

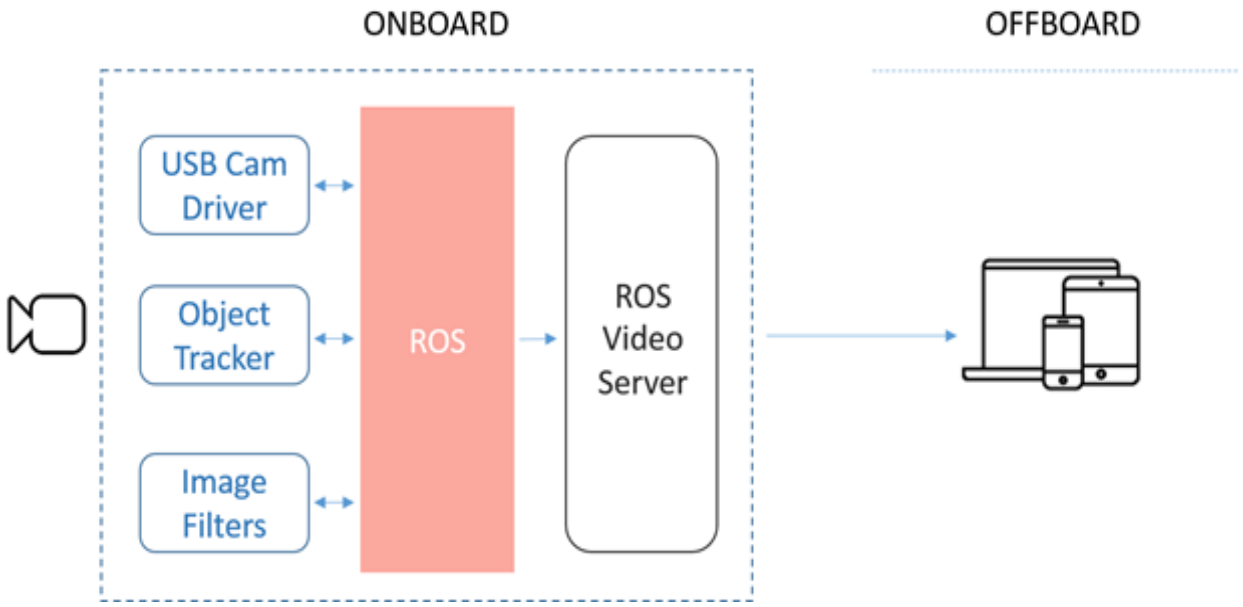


Figure 5 - ROS basic workflow of image processing procedure

The framework was developed in 2007 and has grown significantly with the continued feedback provided by active users and the framework community. The available

documentation is a great benefit for using the ROS, however the adaptation of the framework for UAVs is still an ongoing process, which is of some concern considering the scope of this project.

FlytOS is a framework built on ROS and Linux for developing high level UAV applications. It is developed by Navstik Labs in India and the first version was just released in 2016. The design efforts were particularly aimed to exploit the power of Linux based single board computers on UAVs which resulted in a final product that offers APIs for computer vision, navigation and wireless communication.

The framework is fairly new and currently it doesn't have a large support community, however the initial research has proven that what FlytOS promises to deliver is greatly aligned with the objectives of this project. It can be deployed on any of the companion computers currently under consideration, and the onboard apps are scripts that can be written using onboard APIs either in C++ or Python.

Open Source Computer Vision (OpenCV) is the last option considered for use in the final product. It utilizes several popular interfaces such as C, C++, Python and Java, and it was specifically designed for computational efficiency with real time image processing applications. It can be deployed on any of the Linux-based single board computers that are under consideration, and out of all three options, it has the biggest user community thus allowing a magnitude of available resources. However, unlike the previous two, it is only a programming library and requires the development of additional interfaces to be used separately alongside of the mission control interface on the ground station, shown in Figure 6.

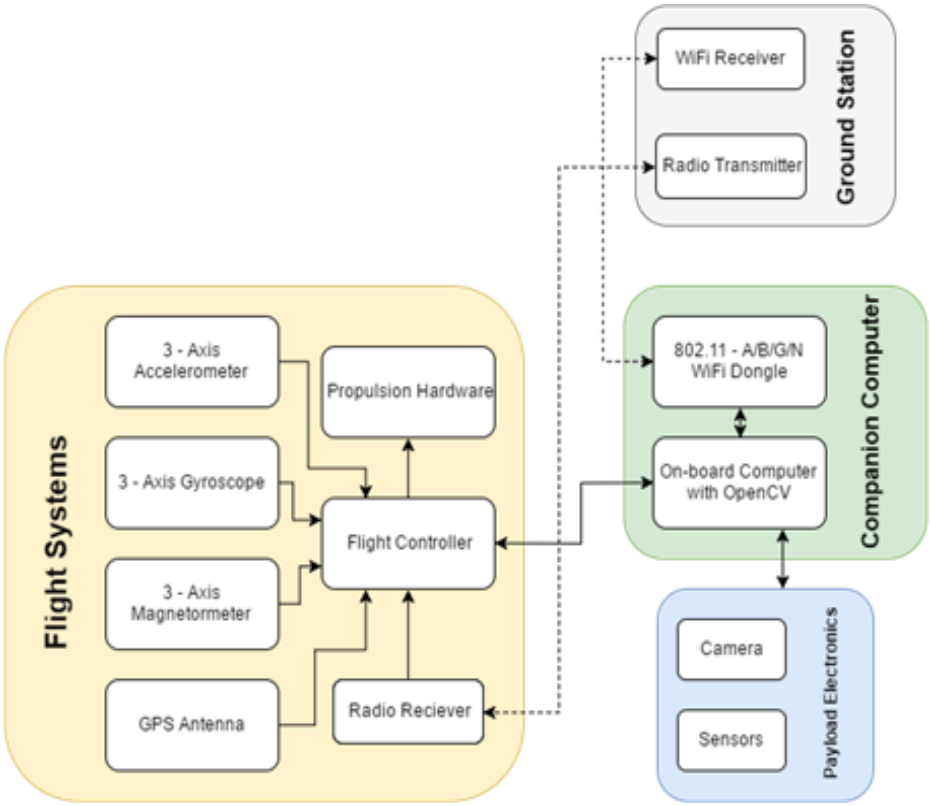


Figure 6 - Preliminary system design

d. Final Design Selection

i. Mechanical Aspects

Peak thrusts, combined with estimated dry weight of the aircraft and a power allowance for on-board computing, allowed calculation of the flight time for two different airframe types under consideration. Vehicle total weight was considered to be half of peak thrust. Battery weight was considered to be vehicle total weight minus vehicle dry weight. Available energy was considered to be LiPo energy density multiplied by battery weight. This information is displayed in Figure 7.

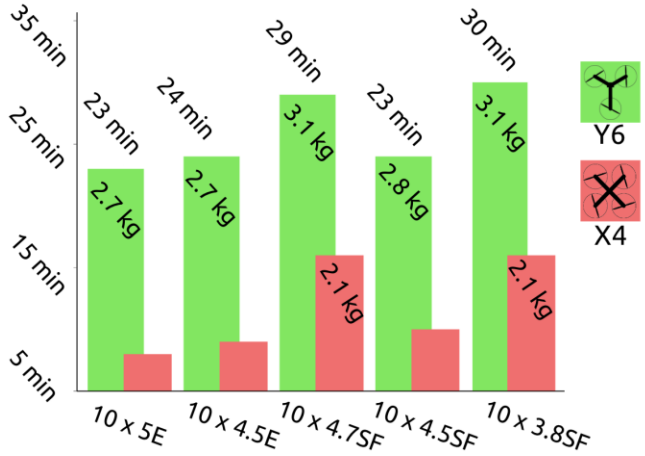


Figure 7 - Performance estimates of top 5 propellers

The goal, a flight time of 30 minutes, is achievable with the use of a 10 x 3.8 SF propeller and a Y6 airframe. The airframe was modeled using these results, resulting in a 6 rotor airframe using 10 x 3.8 SF props. After further testing, and implementation of all onboard electronics, 12 in props were additionally bought to increase the thrust that the aircraft is capable of generating. The drone still flies with the 10 in props, and were used in demonstration flights.

H6 was chosen as the airframe layout for increased payload capability and compatibility with FlytConsole. The main battery was chosen to be 4-cell LiPo, necessitating 2 UBECs. One UBEC would provide 5 volts to the flight controller, and the other to provide 12 V to the companion computer and gimbal.

ii. Computer Aspects

When deciding on a companion computer, multiple options were considered, including the Raspberry Pi, the Odroid XU4, the Nvidia TK1, and the Nvidia TX1. After reviewing these processors, the best option was determined to be the Nvidia TX1. This computer stood out because of its low power consumption, weight, and high processing ability. To interface the TX1 with the flight controller, the operating system FlytOS will be used. The team at FlytOS has successfully uploaded their operating system onto the TX1 to utilize its processing

power in a familiar way via their operating system. Ultimately, the Nvidia TX1 was chosen because of its superior processing power, and low power consumption. The TX1 consumes less power than the Odroid XU4, at only 10 Watts, but provides much more processing power which will be needed for image processing. The image processing software implemented was OpenCV, or Open Computer Vision. The benefit of this processing software is that it can directly access the GPU, or graphics processing unit. This allows the software to directly utilize the hardware capability of the board without being slowed down by multiple layers of abstraction.

FlytOS is hardware specific, which means it requires specific files to be installed on different platforms. These files however, are not publicly available for the onboard computer that was mentioned in the previous section.

In an attempt to get access to the installation files on TX1, the team reached out to the developers of FlytOS: Navstik Labs. Through a conference call, the details and scope of this project were discussed, which ultimately led to receiving a confidential copy of FlytOS along with classified documentation after several weeks of consideration. Once the availability problem was resolved, the team evaluated both options once again and decided to establish FlytOS as the top level controlling entity of the aircraft.

Keeping the sponsor's request of utilizing parts that were familiar to EMHS team and reusing the already existing inventory, the team considered only two flight controller hardware: Pixhawk Autopilot and ArduPilot Mega 2.5. The research indicated that both controllers deliver similar flight performance and are capable of satisfying the stable and reliable autonomous flight requirement[2]. However, although ArduPilot is a fine and proven performer, it has reached the limits of its capabilities and new firmware has already moved beyond its memory and speed capabilities.

Figure 8 - Flight control hardware - Pixhawk [2]

Conversely, Pixhawk shown in Figure 8 provides a significantly improved hardware: 32 bit architecture, faster processor and more memory[2]. It is a high performance autopilot module suitable for many types of robotics applications, including multi-rotor aircrafts. It brings many necessary sensors together on one board, such a gyroscope, accelerometer, compass, barometric pressure sensor and voltage and current sensor. It runs NuttX real-time operating system that offers flexibility through a Unix/Linux-like programming environment to accommodate any specific need. Its integrated multithreading and autopilot functions such as scripting of missions and flight behavior provide powerful development capabilities. Subsequently, the team decided to move forward with the Pixhawk Autopilot as the flight controller hardware. Figure 9 demonstrates the flight stack running on the finalized aircraft.

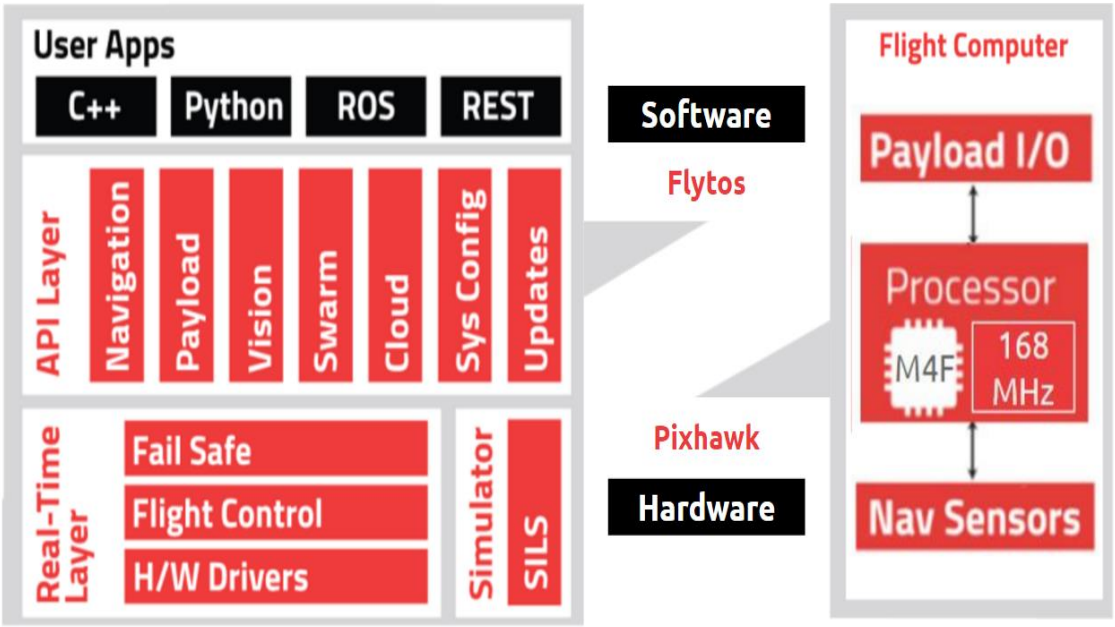


Figure 9 - Flight stack of the aircraft

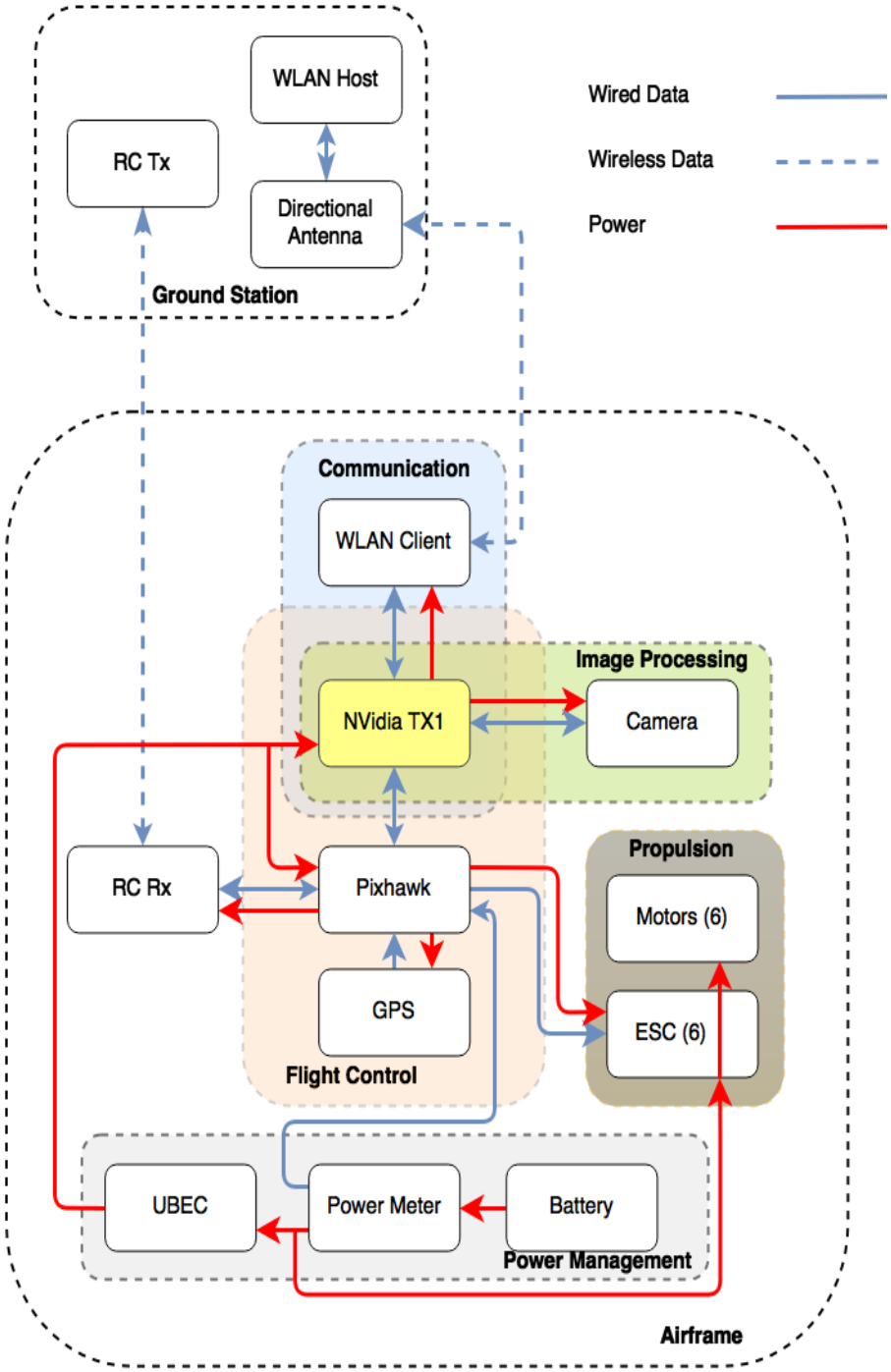
The main control to the aircraft is provided from a WLAN capable device located at the ground station. This interaction is executed through the web interface, provided by FlytOS. However, the aircraft will feature another communication channel for manual operation as a fail-safe. Should the primary communication fail or the operator desire to assume manual control of the aircraft, the secondary communication method, radio control, is available on standby.

3. Design of Major Components

To deliver the novel functionality of autonomously detecting unique objects, the aircraft must be able to process large amounts of data, while simultaneously interfacing with the flight controller. Accomplishing this requires an onboard computer that is fast enough to handle these tasks while also being light and efficient enough to be integrated into the UAV without excess strain.

a. System Level Block Diagram

Figure 10 shows the detailed system diagram that encapsulates both the UAV and the ground station. Although multiple ground station operators are supported, the diagram only displays one for simplicity.



ESC	Electronic speed control
FlytOS	Operating system for companion computer
GPS	Global positioning system
Nvidia TX1	Companion computer
Pixhawk	Flight controller
RC Rx	Radio control receiver
RC Tx	Radio control transmitter
UBEC	Universal battery eliminator circuit
WLAN	Wireless local area network

Figure 10 - Diagram of system components

b. Companion Computer

The Nvidia TX1's GPU has 256 CUDA cores, which are parallel pipelines for computing. Parallel pipelines allow commands to be sent down multiple pipelines so they can all be completed at once, together, without having to wait. The TX1 can also process and train neural nets while taking direct advantage of the CUDA cores, which could allow neural networks to aid in the detection of objects through training rather than strict programming. To process images, the TX1 must be able to grab image data directly from the on-board camera. The UAV is currently interfacing the TX1 with the Logitech C920 camera.

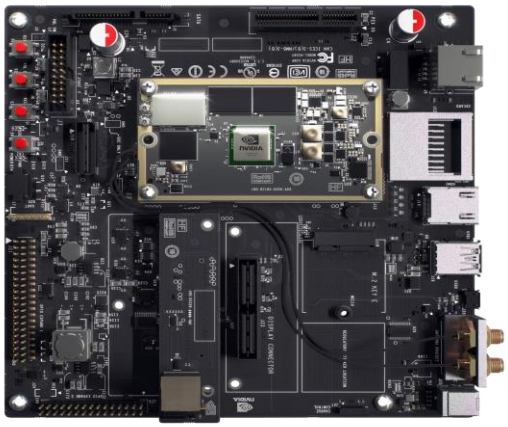


Figure 11 - NVidia TX1 on included Development Board

Above in Figure 11 [3] is the TX1 along with its development board. The development board includes all needed ports and protocols to communicate with the TX1, and thus has additional abilities not needed for this project. Initially, to interface with the TX1 the ports and power management provided by this board are needed for testing and establishing connections. After the software has been tested it has been determined that the needed ports are the USB 3.0 port for the camera, the UART port for communication with onboard electronics, and the Micro SD port for file storage. This lead to the choice of the Orbitty carrier or our carrier board.

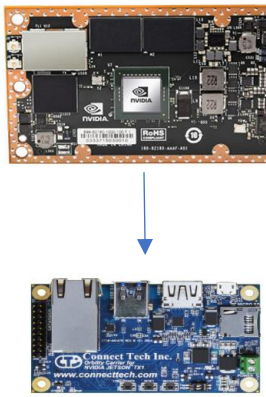


Figure 12 - TX1 Computer Module and Companion Board

Figure 12 shows the companion board for the TX1, the orbitty carrier. This board contains the ports needed to maintain all functionality at a much smaller form factor then the development board. This carrier board not only decreased the size, but greatly decreased the power consumption and weight, leaving more power available for the motors while reducing their load, allowing for longer and more stable flights.

c. Flight Control

The finalized aircraft has two major electronic subsystems: the onboard computer and the flight controller. For the system to function as intended, it is crucial that both subsystems work concurrently. Furthermore, a proper communication was established between the subsystems as they rely on the data provided by one another.

i. FlytOS

Soon after the initial research, it was determined that a controlling entity was needed to assume such tasks; overseeing the operation of the subsystems and handling the data transfer between them. Further research indicated that this functionality is traditionally provided by Robot Operating System (ROS) which is a collection of open source libraries and tools. It is a generic OS to be used in a variety of robotics applications and provides the basic control and intersystem communication services[1]. The drawback associated with ROS is that it doesn't provide any specialized functionality or APIs to be used in a UAV application.

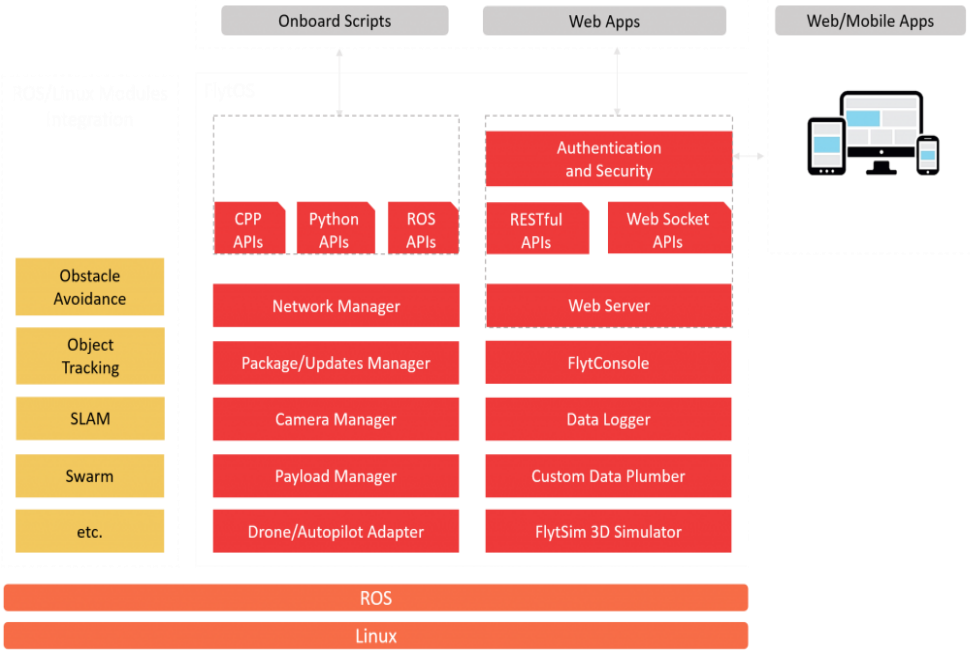


Figure 13 - General architecture of FlytOS

During the research process, another possible solution to the control and intersystem communication issue was identified as FlytOS, a framework to develop high level UAV applications. It is built on ROS thus it inherits the entire pre-existing robotics functionality while offering additional APIs to operate the UAV and to access telemetry data [1]. Vision APIs offered by FlytOS development of image processing applications. It is Wi-Fi network capable, and provides a convenient web based ground control station. Figure 13 demonstrates the general architecture of FlytOS.

ii. PixHawk, GPS Module and RC Controller

For the proper operation of Pixhawk, and the aircraft in general, the flight controller is equipped with a GPS module and a radio-control (RC) receiver. This equipment was provided by the sponsor, thus the team did not look for alternative options after researching to ensure that the provided equipment would deliver the desired performance and work satisfactorily with the rest of the UAV's systems.

The GPS module is Ublox GPS by 3DR shown in Figure 14. It is the recommended GPS for both Pixhawk and ArduPilot due to its accuracy[4]. It features active circuitry for the ceramic patch antenna, rechargeable backup battery for warm starts, and I2C EEPROM for configuration storage. In addition, it ships preconfigured for use with Pixhawk, which reduces the time spent for preliminary tasks.

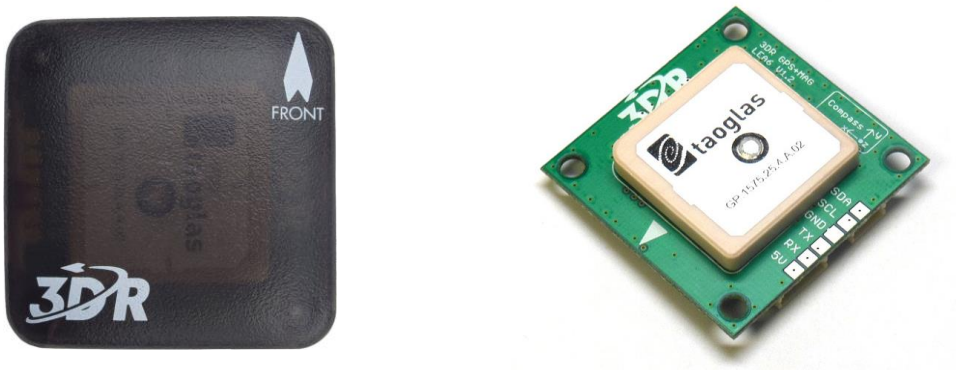


Figure 14 - Ublox GPS by 3DR

According the Pixhawk documentation, any transmitter that has an available receiver which outputs a CPPM / PPM sum signal, S-BUS or Spektrum Satellite is supported [5]. The provided radio transmitter, FrSky Taranis Plus, is one of the recommended systems by Pixhawk. It pairs with the FrSky X8R radio receiver which is directly attached to the Pixhawk. Shown in Figure 15, this controller takes advantage of the entire 2.4 GHz band, resulting in excellent range and reliability which are necessary for failsafe operation.

Figure 15 - Taranis RC transmitter and X8R receiver by FrSky

d. Image Processing

i. Software and Object Detection

OpenCV, or Open Source Computer Vision—paired with the programming language Python—is the dominating software component. Implementing object detection required extensive use of training databases and machine learning to filter through the image processing stage. Being pinnacle to the project's standard of success, the UAV was expected to identify and return unique objects to the ground station for further evaluation. While object detection is traditionally performed at close to medium range, the UAV remains airborne at an altitude of approximately 200 ft. This factor imposes numerous limitations upon the detection process. While thermal sensing would be ideal for human detection, the climate in Florida makes thermal sensing a poor candidate. A secondary option would be facial recognition or pedestrian tracking, but neither approach is practical at 200 ft.

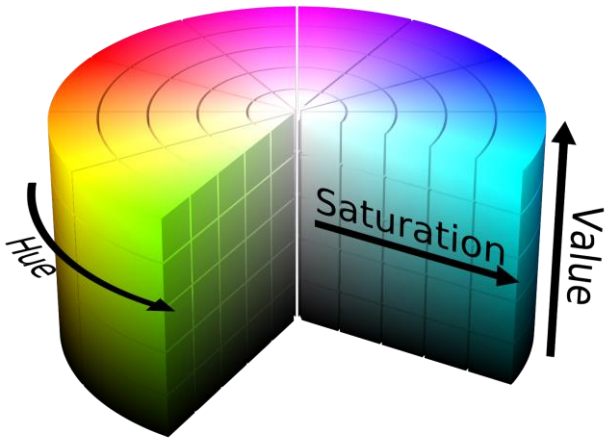


Figure 16 - 3D Visual of HSV

While the initial approach to the image processing entailed relying on a segmentation and elimination process—in which an image is divided into a grid of smaller parts, with each cell undergoing elimination to eliminate the known environment—the complexity resulted in a switch

to a more simple and convenient color filtering algorithm. In the color filtering process, the software is instructed to highlight portions of an image native to specific colors. These colors are determined using three values which individually represent the hue, saturation, and value as assigned to each pixel of an image. Hue refers to the color in its balanced state, saturation is a measure of its intensity, and the value is governed by brightness. Hue, saturation, and value (HSV) visualized in Figure 16. These numbers are checked against a predetermined range, as seen in the sample code below. If the combined values land outside the given range, the color is filtered out in the masking stage (see Figure 17).

```
# Select HSV range for color red
lower_red = np.array([30, 150, 50])
upper_red = np.array([255, 255, 255])
# Converts frames to HSV,
# ... if the frame’s HSV is within the given range
# ... the mask will return true for that frame
mask = cv2.inRange(hsv, lower_red, upper_red)
# Restores image frames only where mask is true
res = cv2.bitwise_and(frame, frame, mask = mask)
```

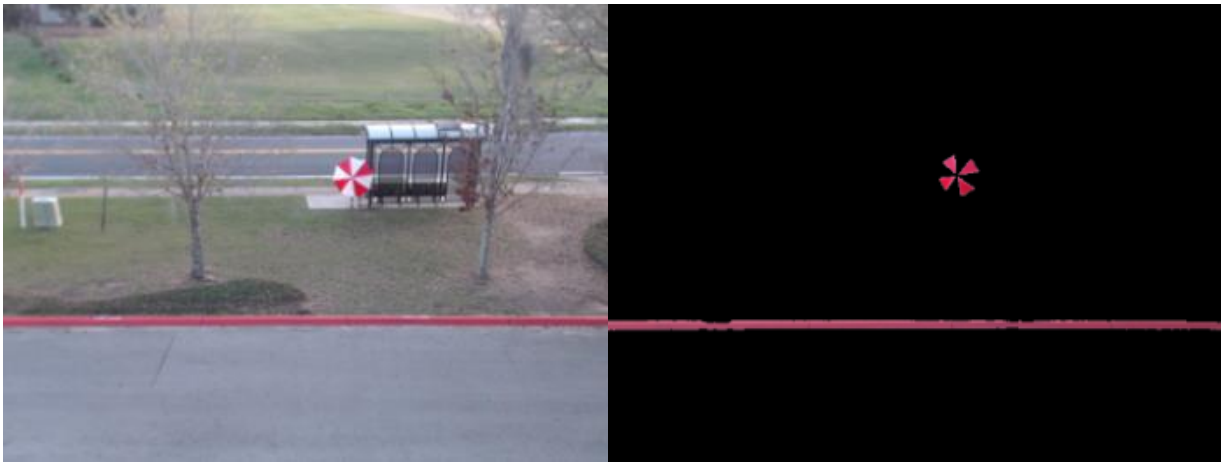


Figure 17 - Masked object detection output

ii. Camera

The designated camera for the object detection process needed to provide consistent and reliable high-definition images of immediate surroundings to attain the highest detection accuracy. The Logitech C920 shown in Figure 18 assumed this role. Chosen for its practicality and live USB streaming capability, the camera features 1080p high definition imaging.

Figure 18 - Logitech C920

e. Communication

i. Network Type

The current ground station equipment consists of several devices to perform different mission tasks such as flight control, image processing, and mission flight planning. The consolidation of these devices enables simultaneous execution of the multiple mission tasks, and reduces the need for frequent maintenance.

A Wireless Local Area Network (WLAN) based on Internet Protocol (IP) is implemented in the UAV to achieve the objective of multi-tasking. The type of WLAN implemented is commonly known as WiFi or Wi-Fi. WiFi is defined as a technology for WLAN that uses devices conforming to the IEEE 802.11 standards, and thus widely available and easily accessible [6]. Conveniently, the main flight control architecture FlytOS supports WiFi communication and requires a WLAN to access the web application for the ground station. The web application is accessed through WLAN from the ground station and has the graphic user interface (GUI) a pilot can use to control all systems of the UAV.

The ground station consists of a laptop capable of creating a WiFi network and an extended wireless range router. The router is equipped with two 9 dBi external antennas, and the UAV is equipped with two 5 dBi external antennas that are connected to the native WiFi chipset onboard TX1. MOFI-4500 4G/LTE is the router and the primary device for the ground station due to its high transmission power of 21 dBm, 4G LTE capability, and four SubMiniature Version A (SMA) connectors that allow flexible combinations of external antenna attachments [7]. FlytOS lacks the ability to cache the mission map profile, and therefore requires a constant internet connection which the extended range wireless router provides. There are two methods of providing the internet connection to the wireless router: one method is using a wired ethernet connection from a traditional internet service provider (ISP) and the second method is to utilize a subscriber identity module (SIM) card from a telecommunication to provide 4G LTE internet service. In the event that the traditional ISP is not available, the user can quickly switch the router to 4G LTE mode to provide the constant internet connection to FlytOS. However, in the absence of all internet connection the UAV can opt to use other flight control architecture such as Mission Planner to complete missions.

ii. Operating Range

The IEEE 802.11 is a set of media access control (MAC) and physical layer specifications for devices that implement WLAN communication in 900 MHz, 2.4 GHz, 3.6 GHz, and 5 GHz. Currently the most used frequencies in WiFi are 2.4 GHz and 5 GHz [8]. 5 GHz is the newest implementation in WiFi standards and provides the fastest data transfer rate. However, 2.4 GHz provides longer range and stability due to its relatively longer wavelength and lower frequency. Using a combination of the link budget equation and free-space loss equation,

the theoretical range was calculated from the ground station to the UAV, and vice versa. The calculation is shown below:

$$\text{Received Power}\,(dB) = \text{Transmitted Power}\,(dB) + \text{Gains}\,(dB) - \text{Losses}(dB) \quad \text{Eq. 1}$$

Expanding the above equation becomes:

$$P_{RX} = P_{TX} + G_{TX} - L_{TX} - L_{FS} - L_{LM} + G_{RX} - L_{RX} \quad \text{Eq. 2}$$

P_{RX} = received power (dBm); P_{TX} = transmitter output power (dBm)
G_{TX} = transmitter antenna gain (dBi); L_{TX} = transmitter losses (dB)
L_{FS} = free space loss (dB); L_{LM} = fade margin (dB)
G_{RX} = receiver antenna gain (dBi); L_{RX} = receiver losses (dB)

Free-space path loss equation in standard form:

$$\text{FSPL} = \left(\frac{4\pi df}{c}\right)^2 \quad \text{Eq. 3}$$

f = signal frequency (Hz)
d = distance from the transmitter (m)
c = the speed of light in a vacuum (m/s)

By combining equation 2 and 3 and rearranging them, the operation range, *d*, becomes:

$$d = 10^{((FSPL - K - 20\,log_{10}(f))/20)} \quad \text{Eq. 4}$$

Using the equation above, the theoretical operation range from the ground station to the drone was calculated to be approximately 1.5 km, and 0.954 km from the drone to the ground station [9]. The difference in transmit power and receiver sensitivity of the router and the WiFi chipset onboard the drone.

f. Location Conversion

UAVs used by the Department of Emergency Management and Homeland Security program (EMHS) report the coordinates in latitude and longitude system which is expressed in degrees, minutes, and seconds to account for the curvature of the earth. Without a specific map projection, specifying a location in two-dimensional space requires complex calculations that include elevation in a specified region.

An improvement to this location system was to implement the United States National Grid (USNG) system. The USNG is a point reference of grid systems commonly used in the United States and is constantly updated and maintained by the Federal Geographic Data

committee, a government committee which promotes the coordinated development, use, sharing, and dissemination of geospatial data on a national basis. The USNG system resembles a common two-dimensional cartesian coordinate that allows the user to read “right and then up.” Therefore, it allows easy calculations of distance and location of the object of interest, reducing time and increasing efficiency. An example of reading a USNG coordinate is shown below in Figure 19.

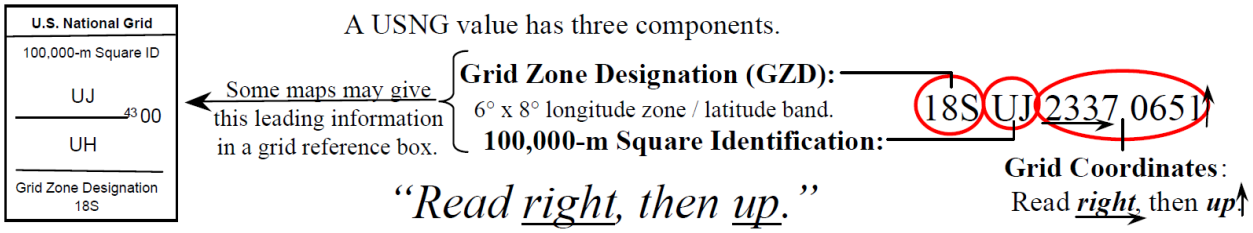


Figure 19 - Components of a USNG value [10]

The USNG provides algorithm for converting latitude and longitude system to USNG system. The algorithm is implemented in the web application GUI to perform conversion and display of USNG system coordinates.

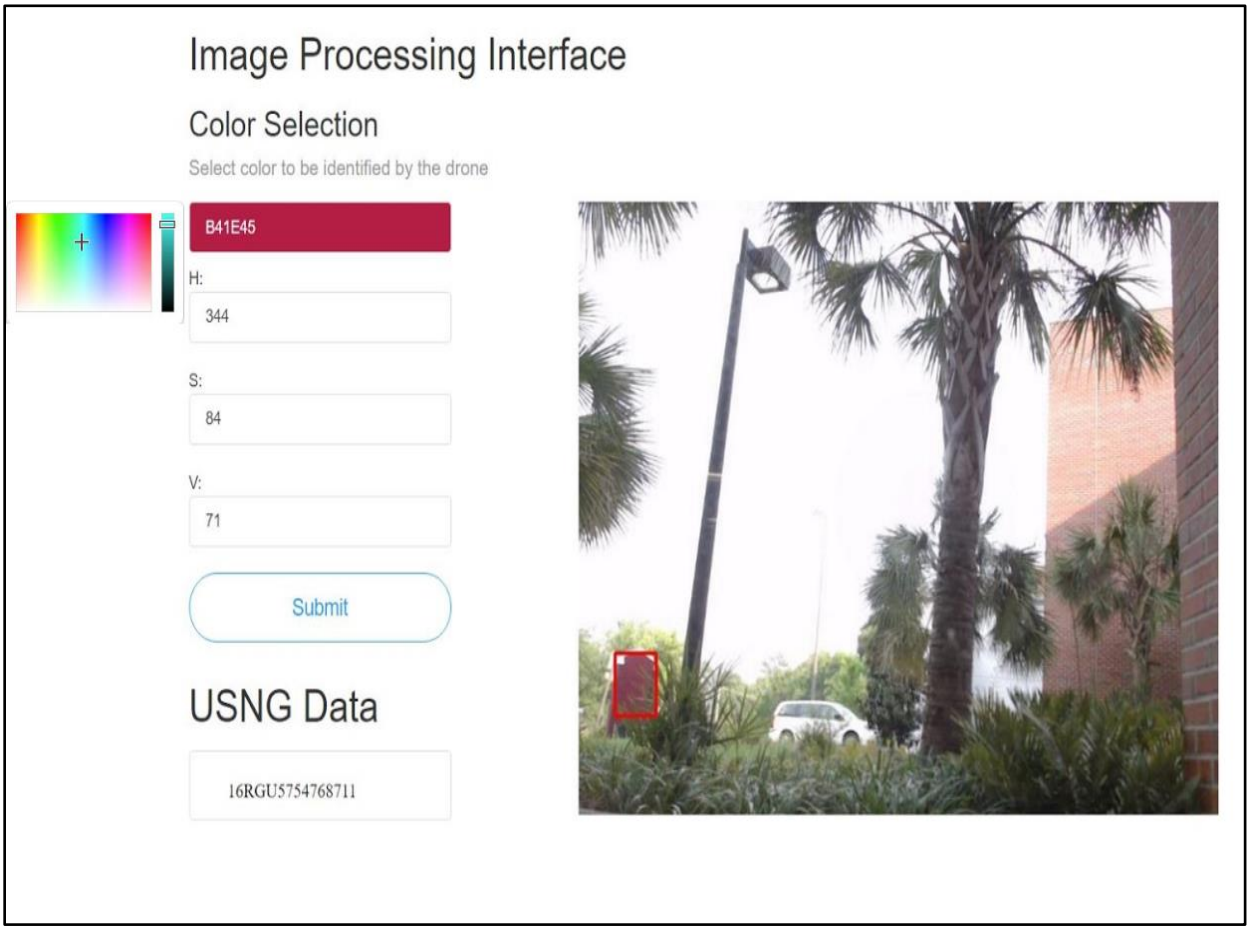


Figure 20 - User interface

g. User Interface

The user interface, shown in Figure 20, is the main way to view the processed video feed from the UAV, the current color being detected in the video feed, and the live USNG coordinates of the UAV. The user interface also provides an intuitive way to select a new color to be detected by clicking on the color and its corresponding six-digit hex value. This will prompt the user to select a color from the color grid shown below. Once the color has been selected, the hue, saturation, and value fields will be automatically updated with the values of the selected color. The hue, saturation, and value variables can also be manually input for greater precision. The USNG coordinates will automatically update every ten seconds, and will appear under the header "USNG data".

h. Power management

Power for the vehicle will be supplied by a lithium-polymer (LiPo) battery pack. This chemistry was chosen for its high energy density [11], availability, and the sponsor's familiarity with use of this battery type. Voltage and current will be sensed by a power meter at the main battery terminals giving the flight control system best information about battery condition and rate of discharge. Full pack voltage is split in a parallel form delivering power to the propulsion system and to a UBEC. The UBEC in play was supplied by the sponsor and is capable of continuously delivering over 5 volts, 15 amps (75 watts) to the vehicle's flight control, image processing, and communication systems.

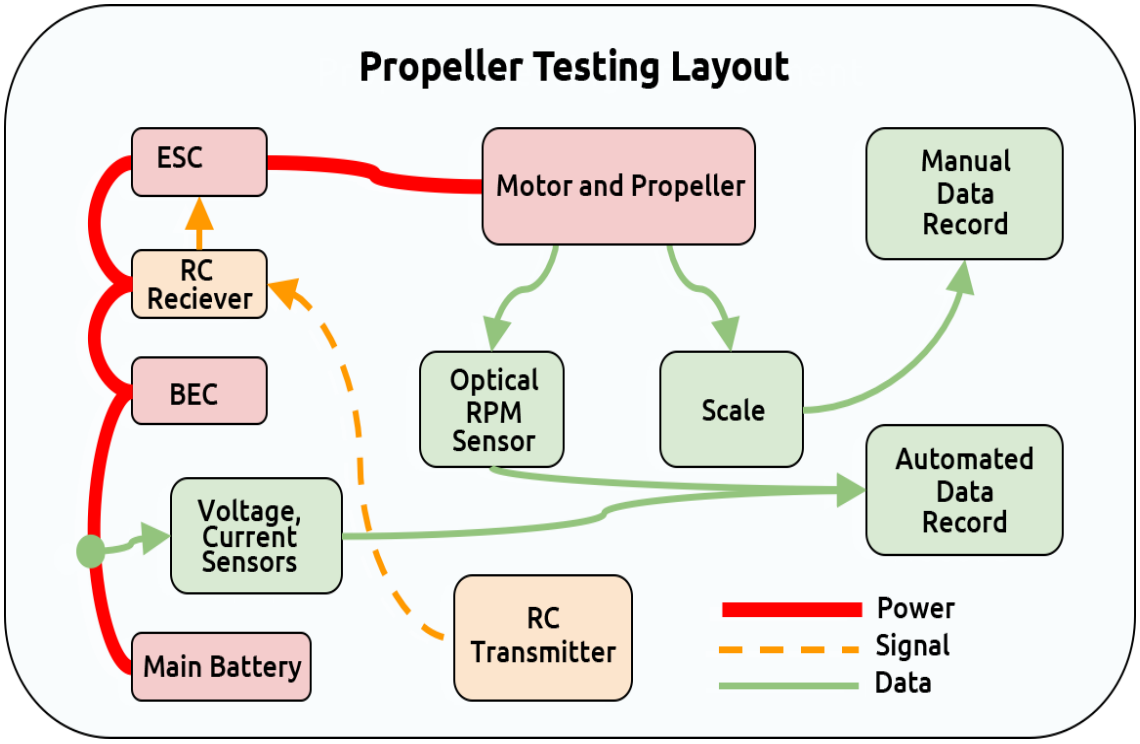


Figure 21 - Propeller Testing Layout

i. Propulsion

"A propulsion system is a machine that produces thrust to push an object forward [12]." Key to maximizing the vehicle's flight duration entailed optimization of the propulsion system. The supplied ESCs and BLDC motors were tested using 10 propellers and 2 battery voltages with respect to limitations specified by the component manufacturers, at 5 throttle positions. Volts, amps, rotational speed, and thrust were measured. Figure 21 shows the general lay-out of the test equipment. Those data were expressed in terms of peak thrust, and power expenditure per unit thrust, and are available in Appendix E.

The aircraft weight was determined by the maximum thrust available, and its flight time was limited by the efficiency of the propulsion system. A multi rotor aircraft use "brute-force" to lift and stabilize themselves [13], so under normal flight conditions the UAV will be operating short of its peak thrust allowing for stabilization and easy maneuvering. With this in mind peak thrust and mid-range throttle efficiency were given equal weight and combined to select the best 5 propeller/battery combinations for consideration. Table 1 displays relevant information about these, and were used for predictions about the airframe and its behavior.

Table 1 - Propeller Rankings, Size, and Peak Thrust

Ranking	Propeller Size (cell-count)	Peak Thrust	Midrange Efficiency
1	10 x 3.8 SF/4 (4 cell)	1,037 g	8.59 g/W
2	10 x 4.7 SF (4 cell)	1,031 g	8.37 g/W
3	10 x 4.5 SF (4 cell)	922 g	8.65 g/W
4	10 x 4.5 E (4 cell)	910 g	8.64 g/W
5	10 x 5.0 E (4 cell)	895 g	8.53 g/W

j. Airframe

The dimensions of the Quanum 680UC in its assembled state, without propellers, are 26.8 inches long by (680 mm) in diameter and 10.2 inches (260 mm) high due to landing gear. In its folded state, the UAV becomes 11.0 inches (280 mm) high by 10.2 inches (260 mm) wide [14], so the craft will fit into an oversized (50 L) backpack as required by the needs assessment. The

dimensions can be seen in Figure 22. Propellers add up to 12 inches to these dimensions, but they can be easily assembled and disassembled for storage position.

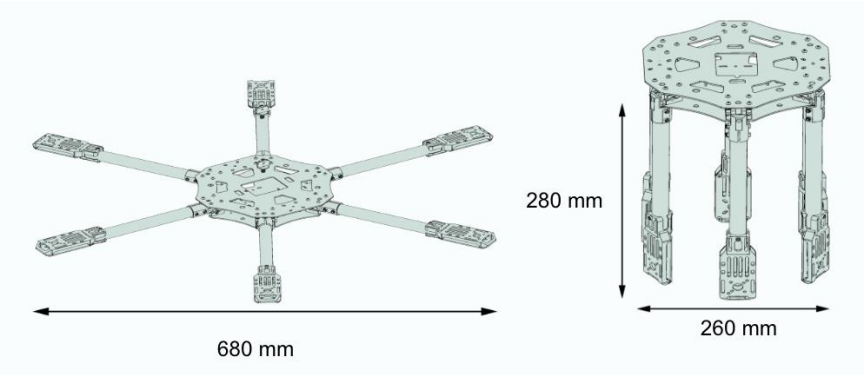


Figure 22 - Dimensions of the Quanum 680UC

k. Completed Vehicle

The finalized vehicle is a result of careful planning and construction that will meet the goals of this project. Table 2 shows the overall weight of the UAV and the individual weight of the components.

Table 2 - Overall weight

Pixhawk	1	38 g
Pixhawk accessories	1	60 g
Motor/Prop/ESC each	6	118 g
Controller receiver	1	22 g
Gimbal	1	214 g
Camera	1	150 g
Nvidia TX1	1	144 g
Airframe	1	740 g
Battery	2	500 g
Total weight	**3076 g**	

4. Test Plan

The comprehensive test plan consisted of two parts; component level and system level testing. Each component was individually tested, verifying their functionality and operation. The system level testing evaluated the progress of the system as a whole, with all components installed on the UAV. Component level testing included both hardware and software components. The testing was completed as components were rendered available and their dependencies acquired and tested. Components like the gimbal was tested without any other hardware or software in place, but components like the carrier board could only be tested once the TX1 has been adequately tested.

Components and corresponding qualities to be tested:

TX1

- Operating system
- FlytOS software
- Wifi Connection
- Power Consumption

Gimbal

- Motion in all directions - automatic stabilization
- Responsive to ground controller controller

External Router

- Long range connection
- 4G wireless connection
- Base station connection

Power System

- Peak Thrust of motors
- Motor draw in relation to battery configuration
- Motor configuration tests

USNG Software

- Ensure that USNG coordinates are accurately created from latitude/longitude coordinates
- Produce USNG coordinates on the fly, updating every 10 seconds

Pedestrian Tracking software

- Verify reliable distance of detection software

Image recognition software

- Evaluate viability of color based detection

Once the component level testing was completed, the system as a whole was assembled and tested collectively. The testing resulted in a fully operational onboard computer and deemed ready for flight.

5. Project Schedule

Fall 2016 accomplished a lot of the background research behind choosing the components and the directions with which to head. The Spring 2017 allowed time to accomplish construction and further coding implementation. Seen in Figure 23 and 24, each task was allotted time to keep the team on track and moving forward. The rescue drone was successfully able to follow all deadlines.

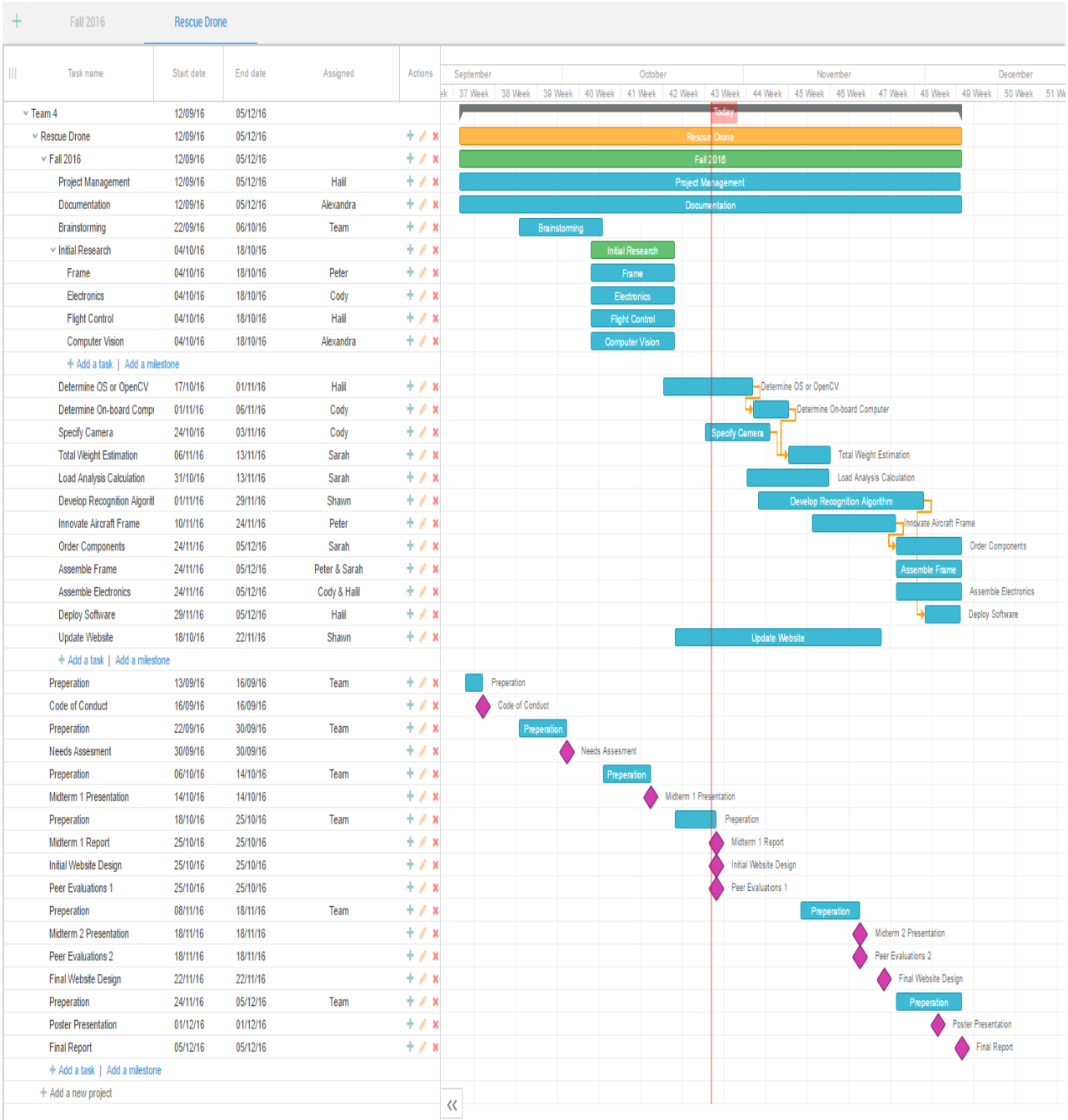


Figure 23 - Gantt Chart, Fall 2016

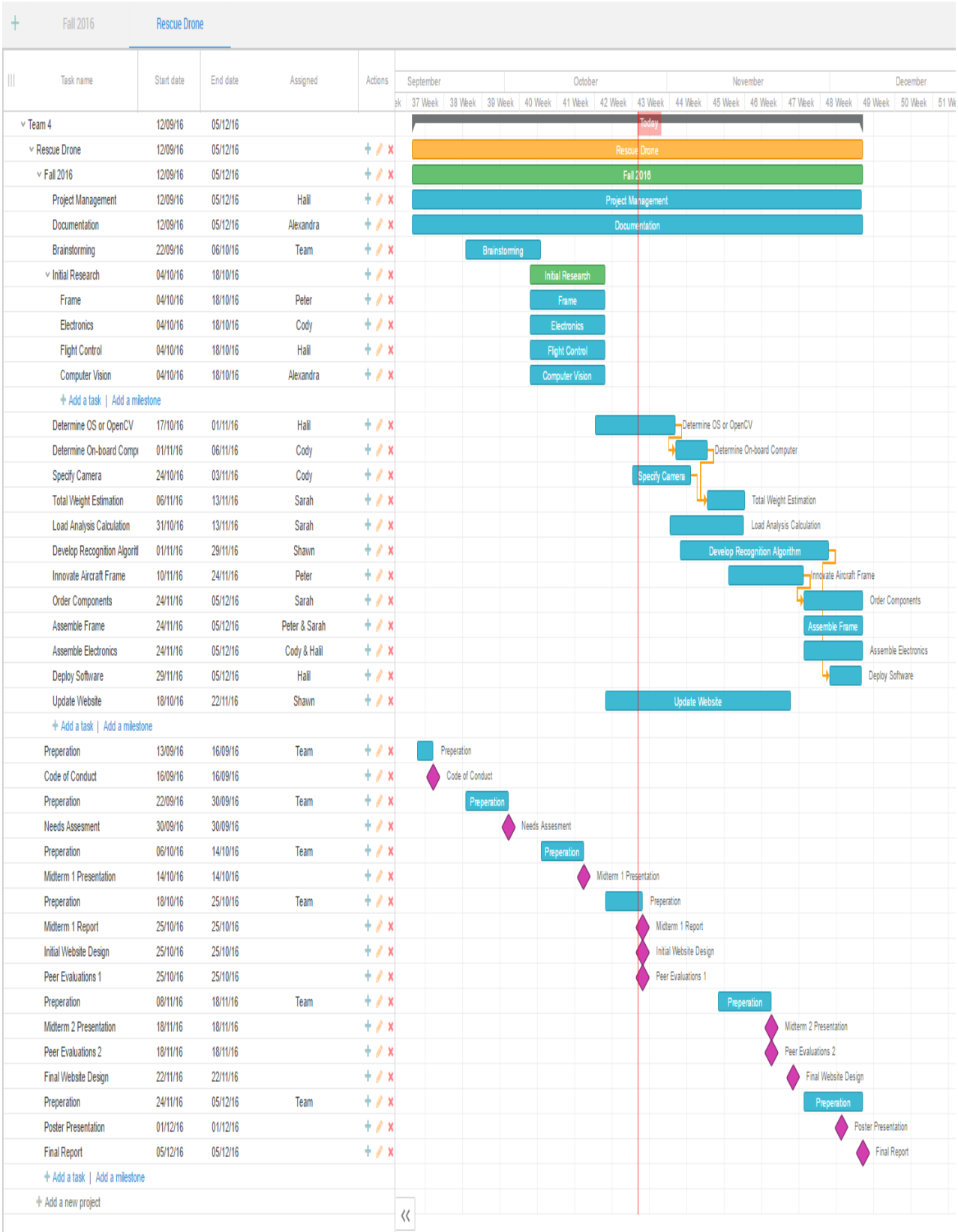


Figure 24 - Gantt Chart, Spring 2017

6. Budget Analysis

The project was sponsored with $3,000 toward components, construction and any additional costs. As seen in Table 3, approximately 50% of the budget was spent in the creation of this UAV.

Table 3 - Cost breakdown for project

Ground Station Router	$340.00
Jetson NVIDIA TX1	$325.00
Gimbal	$208.00
Carrier Board for TX1	$200.00
Hexacopter Frame	$175.00
Camera	$65.00
Carbon Fiber Sheets	$55.00
Carbon Fiber Tubes	$47.00
UAV Antenna	$40.00
Prop 12 x 3.8 SF	$47.00
Prop 10 x 3.8 SF	$35.00
Prop 12 x 4.5 SF	$15.00
Ground Station Antenna	$13.00
UBEC	$4.00
Total	**$1569.00**

The pie chart in Figure 25 shows how much was spent in each of the different areas of the design process. Airframe cost a total of $378, and the onboard hardware cost $525. The modem and Wi-Fi technology for communication between the UAV and the ground station is $393. The gimbal and the camera make up the image hardware components and cost $273.

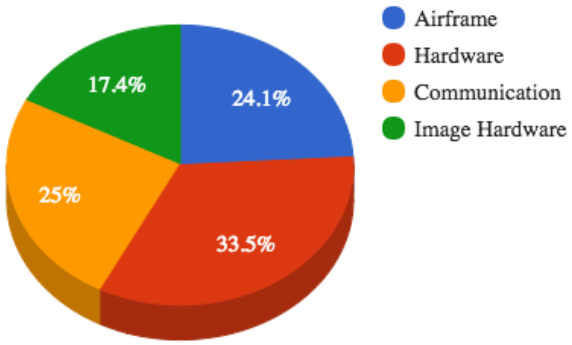


Figure 25 - Cost percentage distribution into 4 major categories

7. Conclusion

Due to their small size and high maneuverability, unmanned aerial vehicles provide a fast and reliable way to incorporate aerial assistance in search and rescue operations for a fraction of the cost and resources. In a situation where every second matters, they provide quick response as they can be easily deployed. Florida State University's Emergency Management and Homeland Security Program has been taking advantage of drones in their search and rescue efforts. Through extensive utilization, they realized the insufficient aspects of their current inventory and provided ECE Team 4 with the opportunity to improve upon them.

To reduce the time spent by the operators to manually process the images captured by a search and rescue drone, an image processing algorithm that could identify colors was implemented. Image processing is a computationally heavy task, and making it readily available on a small aerial vehicle was one of the primary challenges throughout the project. The required processing power to run the algorithm onboard was provided by NVidia TX1 single chip computer. To reduce the size, weight, and power consumption, TX1 was removed from its developer board and placed on an Orbitty Carrier board. To better accommodate the ever-advancing technology, the onboard camera was connected via USB connection and provides the option to be easily upgradeable as more capable models become available.

Upon EMHS department's request, another algorithm was written to return the location of the drone to the ground station in the United States National Grid format, as it is more widely adopted by the emergency management agencies. This data was made available through a web interface that supports multiple operators to interface with the UAV and provides an environment to input new target values to the image processing algorithm. Another action taken to enable multiple operators was to implement a WiFi network between the ground stations and the drone. This also removed the need for designated telemetry equipment previously used to operate UAVs.

Through a rigorous calculation and testing process, an hexacopter frame was found the most suitable for the UAV. This configuration combined with 650kv brushless motors, 12x3.8 slow fly propellers and a 4 cell lipo battery allowed the UAV to deliver 20 minutes of flight duration. The custom landing gear designed and manufactured by Team 4, which replaced the original landing gear, was another contributor to the flight time. The flight capability of the drone was tested and verified both in manual and autonomous flight modes.

ECE Team 4's efforts primarily focused on delivering a reliable UAV with an intuitive user experience, while keeping the expenses as little as possible. During flight operations, the team observed safety precautions that are laid out in the FAA's small unmanned aircraft regulations. Utilizing Pixhawk flight controller and Navstik's FlytOS, the effort required towards developing time-consuming background tasks were redirected towards improving and enhancing the drone's image processing and communication features. Implementing an UDP bridge between the UAV and the ground station provided the option to use well-established mission planning software.

After demonstrating the completed UAV to the EMHS staff, it has been verified that the vehicle satisfies the needs of the EMHS department. Team 4 is confident that, in the hands of EMHS pilots, *Saurus* will provide a lasting contribution to the application of search and rescue drones for recovery efforts following natural disasters.

Figure 26 - Saurus, The Completed Aircraft

8. References

[1] "Robot Operating System." [Online]. Available: http://www.ros.org/. [Accessed: 01-Dec- 2016].

[2] "Pixhawk Autopilot." [Online]. Available: https://pixhawk.org/modules/pixhawk. [Accessed: 01-Dec-2016].

[3] "Nvidia Jetson TX1 Dev. Board is a Mobile Supercomputer for Machine Learning." [Online]. Available: http://arstechnica.com/gadgets/2015/11/nvidias-jetson-tx1-dev-b oard-is-a- mobile-supercomputer-for-machine-learning/. [Accessed: 01-Dec-2016].

[4] "3DR UBlox GPS + Compass Module." [Online]. Available: http://ardupilot.org/copter/docs/common.html. [Accessed: 01-Dec-2016].

[5] "Compatible RC Transmitter and Receiver Systems (Pixhawk/PX4)." [Online]. Available: http://www.ardupilot.org/copter/docs/common-pixhawk-and-px4-compatible-rc- transmitter-and-receiver-systems.html. [Accessed: 01-Dec-2016].

[6] "An Overview of the Wi-Fi Alliance Approach to Certification." [Online]. Available: http://www.senzafiliconsulting.com/downloads [Accessed: 01-Dec-2016].

[7] "MOFI4500-4GXeLTE-SIM4 Spec Sheet." [Online]. Available: http://mofinetwork.com/files/MOFI4500_4GXeLTE_SIM4_Spec_Sheet.pdf. [Accessed: 20- Apr- 2017].

[8] "IEEE-SA Standards Board Operations Manual." [Online]. Available: https://standards.ieee.org/develop/policies/opman/sect8.html. [Accessed: 01-Dec-2016].

[9] Sklar, Bernard. Digital Communications Fundamentals and Applications. Upper Saddle River, NJ: Prentice Hall, 2001. Print.

[10] "Reading US National Grid (USNG) Coordinates." [Online]. Available: https://www.fgdc.gov/usng/educational-resources/USNGInstruct_No1v4_No2_r.pdf. [Accessed: 20- Apr- 2017].

[11] D. Gilson, "Li-Ion vs Li-Poly, plus how do Lithium batteries work anyway?," All About Symbian, 25-Sep-2012. [Online]. Available: http://www.allaboutsymbian.com/features/item/15775_How_do_Lithium_batteries_wo rk.php. [Accessed: 04-Dec-2016].

[12] "NASA Beginner's Guide to Propulsion." [Online.] Available: https://www.grc.nasa.gov/www/k-12 /bgp.html. [Accessed: 01-Dec-2016].

[13] M. J. Dougherty, Drones: an illustrated guide to the unmanned aircraft that are filling our skies. London: Amber Books, 2015.

[14] "Quanum 680UC Pro Hexa-Copter Umbrella Carbon (Kit)", *Hobbyking*, 2017. [Online]. Available: https://hobbyking.com/en_us/quanum-680uc-pro-hexa-copter-umbrella-carbon-kit.html. [Accessed: 20- Apr- 2017].

Appendix A - Design of Major Components

a. Coordinate Conversion Algorithm

```
//Senior Design Team 4 - Spring 2017

// Halil Yonter
// Cody Campbell
// Sarah Hood
// Alexandra Borgesen
// Shawn Cho
// Peter Burchell

/*Description: The following code is designed to convert latitude and longitude
 GPS input to a variable USNG data format. The location is output alongside the
 date and 12-hour format for reference.*/

#include <core_script_bridge/navigation_bridge.h>
#include <core_script_bridge/param_bridge.h>
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <chrono>
#include "lpos_gpos_convertor.h"

#define BUFFSIZE 50
#define DTTMFMT "%Y-%m-%d %H:%M:%S "
#define DTTMSZ 21

Navigation nav;
Param par;
sensor_msgs::NavSatFix gpos;
core_script_bridge::UserData rad_data;
int rad_data_pub;

float sigma = 0.0002;
float mean_x = 0;
float mean_y = 0;

static char *getDtTm (char *buff)
{
   time_t t = time (0);
   strftime (buff, DTTMSZ, DTTMFMT, localtime (&t));
   return buff;
}
```

```
void simulate_data(float lat, float lon, float &data)
{
     //data = (1/(2*3.14*sigma*sigma)) * exp(-((lat-mean_x)*(lat-mean_x) + (lon-mean_y)*(lon-
mean_y)(2*sigma*sigma));
   data = 10 * exp(-((lat-mean_x)*(lat-mean_x) + (lon-mean_y)*(lon-mean_y))/(2*sigma*sigma));
}
void gposCb(void *_gpos)
{
   gpos = * (sensor_msgs::NavSatFix*)(_gpos);
   float radiation_data;
   simulate_data(gpos.latitude,gpos.longitude,radiation_data);
   rad_data.data_double.clear();
   rad_data.data_double.push_back(gpos.latitude);
   rad_data.data_double.push_back(gpos.longitude);
   rad_data.data_double.push_back(radiation_data);
   nav.userPublish(rad_data_pub,rad_data);
}

int main(int argc, char *argv[])
{
   FILE *fp;
   char file_type[40];

   char argument1[BUFFSIZE];
   char argument2[BUFFSIZE];
   char argument3[BUFFSIZE];

   double lat = 0;
   double lng = 0;
     int loader = 0;

   //   The appropriate input for GeoConvert in order to convert from
   //   latitude-longitude to USNG should be in the following form:
   //
   //   GeoConvert -m --input-string "-25.47 -84.1");

   rad_data_pub = nav.userAdvertise("radiation_data");
   nav.sysSubscribe(Navigation::global_position,gposCb);

     if(ros::master::check())
     {

          lat = 0;
          lng = 0;
          sleep(1);

          lat = gpos.latitude;
          lng = gpos.longitude;

          strcpy(argument1, "GeoConvert -m --input-string \"");
```

```
        sprintf(argument2, "%.4f", lat);
        strcat(argument2, " ");
        sprintf(argument3, "%.4f", lng);
        strcat(argument3, "\"");

        strcat(argument1, argument2);
        strcat(argument1, argument3);

        fp = popen(argument1, "r");
        if (fp == NULL) {
            printf("Failed to run command\n" );
        }

        while (fgets(file_type, sizeof(file_type), fp) != NULL) {
            //printf("%s", file_type);
        }

        std::ofstream myfile;
        char buff[DTTMSZ];
        myfile.open ("/home/ubuntu/customApps/usng/setup_folder/location.txt");
        //myfile << getDtTm(buff) << file_type << std::endl;
        myfile << file_type << std::endl;
        myfile.close();

        pclose(fp);
    }
    //std::cout<<"\nKill signal received..\nExiting.."<<std::endl;
}
```

b. Image Processing Algorithm

```
'''
Senior Design Team 4 - Spring 2017

Halil Yonter
Cody Campbell
Sarah Hood
Alexandra Borgesen
Shawn Cho
Peter Burchell

 Description: This code is a simple mjpg stream http server, which works by
 broadcasting live camera output onto an online server. Before outputting,
 it executes a color filtering algorithm which checks the range of HSV values
 passing in through a color picker.
'''
import cv2
import numpy as np
import Image
import threading
from BaseHTTPServer import BaseHTTPRequestHandler,HTTPServer
from SocketServer import ThreadingMixIn
import StringIO
import time
import re

H = 11
S = 11
V = 22

class CamHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path.endswith('.mjpg'):
            self.send_response(200)
            self.send_header('Content-type','multipart/x-mixed-replace; boundary=--jpgboundary')
            self.end_headers()
            while True:
                try:
                    rc,img = capture.read()
                    if not rc:
                        continue

                    hsv=cv2.cvtColor(img,cv2.COLOR_BGR2HSV)

                            #scale HSV values (we get H[0-360],S[0-100],V[0-100]
                            # but need H[0-180],S[0-255],V[0-255]
                    #H = H * 0.5
                    #S = S * 2.55
                    #V = V * 2.55
```

```
        #ranges for HSV
    hueRange = 18
    satRange = 100
    valRange = 100

    #calculate uppper/lower
            upperH = H + hueRange
    upperS = S + satRange
    upperV = V + valRange

    lowerH = H - hueRange
    lowerS = S - satRange
    lowerV = V - valRange

    if upperH > 180:
        upperH = 180
    if upperS > 255:
        upperS = 255
    if upperV > 255:
        upperV = 255

    if lowerH < 0:
        lowerH = 0
    if lowerS < 50:
        lowerS = 50
    if lowerV < 25:
        lowerV = 25

    #defining the range
    red_lower=np.array([lowerH,lowerS,lowerV],np.uint8)
    red_upper=np.array([upperH,upperS,upperV],np.uint8)

    red=cv2.inRange(hsv, red_lower, red_upper)

    #Morphological transformation, Dilation
    kernal = np.ones((5 ,5), "uint8")

    red=cv2.dilate(red, kernal)
    res=cv2.bitwise_and(img, img, mask = red)

    #Tracking the Color

(_,contours,hierarchy)=cv2.findContours(red,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    for pic, contour in enumerate(contours):
        area = cv2.contourArea(contour)

        if(area>300):
            x,y,w,h = cv2.boundingRect(contour)
            img = cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
            cv2.putText(img,"",(x,y),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,255))
```

```
                    imgRGB=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
                    jpg = Image.fromarray(imgRGB)
                    tmpFile = StringIO.StringIO()
                    jpg.save(tmpFile,'JPEG')
                    self.wfile.write("--jpgboundary")
                    self.send_header('Content-type','image/jpeg')
                    self.send_header('Content-length',str(tmpFile.len))
                    self.end_headers()
                    jpg.save(self.wfile,'JPEG')
                    time.sleep(0.05)

                except KeyboardInterrupt:
                    break
            return

        if self.path.endswith('.html'):
            self.send_response(200)
            self.send_header('Content-type','text/html')
            self.end_headers()
            self.wfile.write('<img src="http://192.168.1.24:5050/cam.mjpg"/>')
            self.wfile.write('</body></html>')
            return

    def _set_headers(self):
        self.send_response(200)
        self.send_header('Content-type' + "kem", 'text/html')
        self.end_headers()

    def do_HEAD(self):
        self._set_headers()

    def do_POST(self):
        # Doesn't do anything with posted data
        content_length = int(self.headers['Content-Length']) # <--- Gets the size of data
        post_data = self.rfile.read(content_length) # <--- Gets the data itself
        self._set_headers()

        num = re.findall(r'\d+', post_data)
        global H
        H = int(num[0]) * 0.5
        global S
        S = int(num[1]) * 2.55
        global V
        V = int(num[2]) * 2.55


class ThreadedHTTPServer(ThreadingMixIn, HTTPServer):
    """Handle requests in a separate thread."""

def main():
    global capture
```

```
    capture = cv2.VideoCapture(0)
    capture.set(3, 640);
    capture.set(4, 480);
    capture.set(5, 1);
#   capture.set(cv2.cv.CV_CAP_PROP_SATURATION,0.2);
    global img
    try:
        server = ThreadedHTTPServer(('192.168.1.24', 5050), CamHandler)
        print "server started"
        server.serve_forever()
    except KeyboardInterrupt:
        capture.release()
        server.socket.close()

if __name__ == '__main__':
    main()
```

c. User Interface Code

```
/*
Senior Design Team 4 - Spring 2017

Halil Yonter
Cody Campbell
Sarah Hood
Alexandra Borgesen
Shawn Cho
Peter Burchell

 Description: This html code is designed to operate the image processing interface,
providing a color wheel for the user's ease in autopopulating the range of the
color filter with HSV values upon selecting a color of interest.
*/


<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8" />
   <link rel="apple-touch-icon" sizes="76x76" href="static/assets/img/apple-icon.png">
   <link rel="icon" type="image/png" href="static/assets/img/favicon.png">

   <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
   <title>Image Processing Interface</title>

   <meta content='width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0'
name='viewport' />
      <meta name="viewport" content="width=device-width" />

      <link href="static/bootstrap3/css/bootstrap.css" rel="stylesheet" />
   <link href="static/assets/css/gsdk.css" rel="stylesheet" />
      <link href="static/assets/css/demo.css" rel="stylesheet" />

      <!--     Font Awesome     -->
      <link href="static/bootstrap3/css/font-awesome.css" rel="stylesheet">
      <link href='http://fonts.googleapis.com/css?family=Grand+Hotel' rel='stylesheet' type='text/css'>
</head>
<body>


<script src="static/jscolor.js"></script>

<div class="main">
   <div class="container tim-container">
      <div class="tim-title">
         <h2>Image Processing Interface</h2>
         <h3>Color Selection<br><small> Select color to be identified by the drone </small> </h3>
   </div>
```

```
      <div id="inputs">
         <div class="row">
            <div class="col-sm-3">
          <div class="form-group">
            <input class ="form-control jscolor {onFineChange:'update(this)'}" value="ffcc00" >
               <IFRAME style="display:none" name="hidden-form"></IFRAME>
               <p>

               <form action="http://192.168.1.24:5050/cam.mjpg" method="post" target="hidden-form">
               H:<br>
               <input id = "p1" type="text" class="form-control" name="H*"/><br>
               S:<br>
               <input id = "p2" type="text" class="form-control" name="S"/><br>
               V:<br>
               <input id = "p3" type="text" class="form-control" name="V"/><br>

               <button type="submit" class="btn btn-block btn-lg btn-info btn-round">Submit</button>

               </form>

            <script>
            function update(picker)
            {
               document.getElementById("p1").value = Math.round(picker.hsv[0]);
               document.getElementById("p2").value = Math.round(picker.hsv[1]);
               document.getElementById("p3").value = Math.round(picker.hsv[2]);
            }
            </script>


      <h2> USNG Data</h2>
         <iframe name="usng" class = "form-control input-lg" scrolling="no"
src="http://192.168.1.24:1111/"></iframe>
         </div>
         </div>

   <div class="col-md-6">
      <img src="http://192.168.1.24:5050/cam.mjpg">
   </div>

   </div>
</div>
</body>
</html>
```

Appendix B - Test Plan Documentation

Test Type	Test #	Target Component	Purpose of Test	Expected Result	Actual Result	Actions Taken
Hardware	1	TX1 Development Board	Functionality of TX1 and Linux operating system	System will boot up and successfully access internet	The TX1 booted and ran Ubuntu without errors, with full internet connectivity	No further actions needed
	2	TX1 on the Orbitty Carrier Board	Ensure all functionality is maintained through transition	The TX1 will boot up, access internet, and successfully communicate with on board hardware	The TX1 maintained all functionality	No further actions needed
	3	TX1 WiFi + external router	TX1 wireless communication and reliable range	TX1 will be able to communicate to base state without internet, with hardwired internet, and with 4G sim card	Communication was successful, but not reliable at desired distance	Informed sponsor of antenna trackers and directional antennas
	4	Gimbal	Ensure proper operation and control of gimble	Gimbal should respond to controls and self-stabilize.	Gimbal failed all operational tests.	Replaced gimbal
	5	Gimbal	Ensure proper operation and control of gimble	Gimbal should respond to controls and self-stabilize.	Nominal gimbal movement and tracking	No further actions needed
	6	Props, Power System, Battery, ESC, Motors	Peak Thrust provided by each motor.	4 motors would be sufficient to fly expected weight.	4 motors does not provided sufficient thrust.	Moved forward with a 6 rotor hexa-copter.

Test Type	Test #	Target Component	Purpose of Test	Expected Result	Actual Result	Actions Taken
Software	1	USNG Code	USNG coordinate accuracy	USNG coordinates accurately generated from latitude longitude coordinates every 10 seconds	USNG coordinates were accurately generated	No further actions needed
	2	Pedestrian tracking algorithm	Reliable human detection	A box should be drawn around pedestrians at distances of over 200ft	Pedestrians were only reliably detected up to 30ft	Move forward with color detection
	3	Image recognition code	Reliable color detection	Designated HSV colors should be detected reliably in image	Didn't accurately detect colors with HSV values	Modified code to properly handle HSV values
	4	Image recognition code	Reliable color detection	Designated HSV colors should be detected reliably in image	Colors were accurately detected as long as they could be seen by the camera	No further actions needed
	5	FlytOS	FlytOS functionality and proper interaction with on board software	FlytOS running and updating all widgets with live information from various programs	Calibration interface responded as expected. All Widgets were functional except for the battery capacity	Informed the dev team of the inoperative battery widget

Test Type	Test #	Target Component	Purpose of Test	Expected Result	Actual Result	Actions Taken
Full System	1	Ground test (no propellers)	Verify motor direction, flight controller response, gimbal operation, web service, software functionality, network connection	Nominal operation of all systems.	All systems responded nominally.	No further actions needed
	2	Tethered flight test	Functionality of airframe.	Positive control response and neutral stability.	Control was positive and neutral stability was achieved	No further actions needed
	3	Un-tethered flight test	Functionality of vehicle and associated systems.	Nominal flight characteristics and nominal software performance.	Hover attainable at 80% to 90% throttle.	Obtained larger propellers.
	4	Un-tethered flight test	Functionality of vehicle and associated systems.	Nominal flight characteristics and nominal software performance.	Airframe behaves nominally. Live video feed extremely shaky. Sponsor-supplied battery is inoperable crash at t = 3 min, 2 propellers broken.	Informed sponsor of camera inadequacy.Ordered new propellers. Obtained better battery. Continued testing with smaller propellers and lower vehicle mass.
	5	Un-tethered flight test	Functionality of vehicle and associated systems with deference to battery health.	Nominal flight characteristics, nominal software performance, longer flight.	Vehicle operated nominally. Flight time was extended significantly.	No further action needed
	6	Un-tethered flight test	Functionality of vehicle and associated systems with deference to data connectivity.	Nominal flight characteristics, nominal software performance, 1 km range.		UAV is delivered.

Appendix C - User Manual

a. Components

i. Arm Assembly

Motor - *T Motor 3506 650Kv*

- Propeller - 12x4 Slow-Fly
- Propeller nut, washer

Electronic Speed Control (ESC) - *Castle Creations Multirotor 25*

- Power wire connecting to main power bus
- Signal wire connecting to autopilot

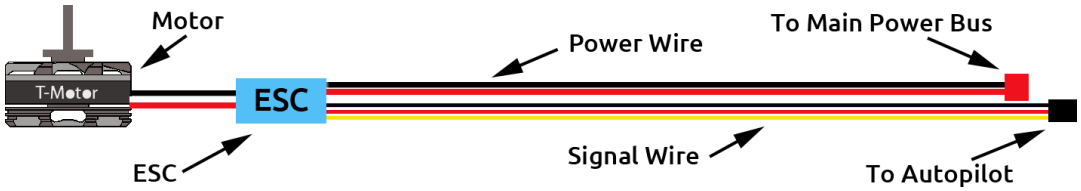


Figure 27 - Electronic Speed Control power wire and signal wire layout within the arm.

Figure 01 shows the layout within the arm of the connections between the motor, the ESC, the main power bus, and the autopilot.

Landing Gear Legs (arms 2, 4, and 5)

- The landing gear for functional purposes can be positioned on any non-consecutive arms, though it is important to position them on arms 2, 4, and 5 to avoid obstruction of the camera. See the fuselage assembly section to understand the location of arms 2, 4, and 5.

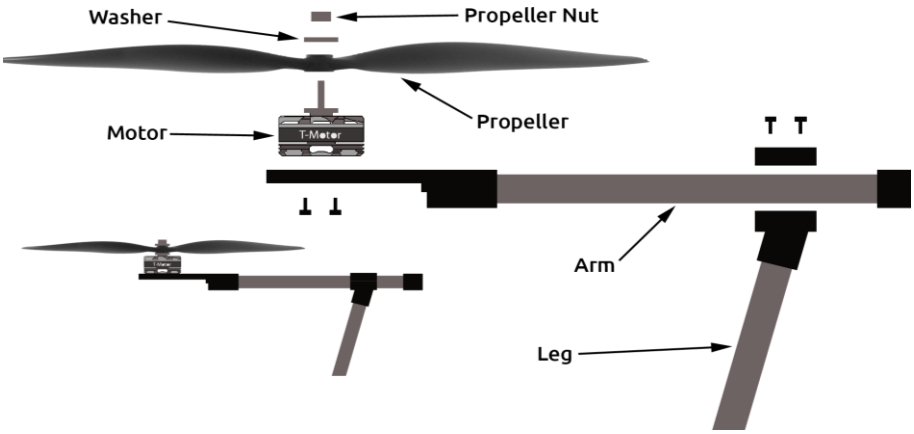


Figure 28 - Exploded view of the arm and landing gear

ii. Fuselage Assembly

Autopilot - *Pixhawk PX4*

- Power meter (PM)
- GPS/compass
- Switch
- Note: These components are imperative for the autopilot, but housed separately.

Radio Control Receiver - *FrSky X8R*

- Receives the signal from the radio controller.

5 Volt Battery Eliminator Circuit - *KINGKONG UBEC*

- Divides the voltage to necessary components requiring 5 Volts.

12 Volt BEC - *Castle Creations BEC Pro*

- Divides the voltage to necessary components requiring 12 Volts.

Main Power Bus

- Distributes battery power among necessary components.

Fuselage

- Carbon fiber plates that the components are secured to.

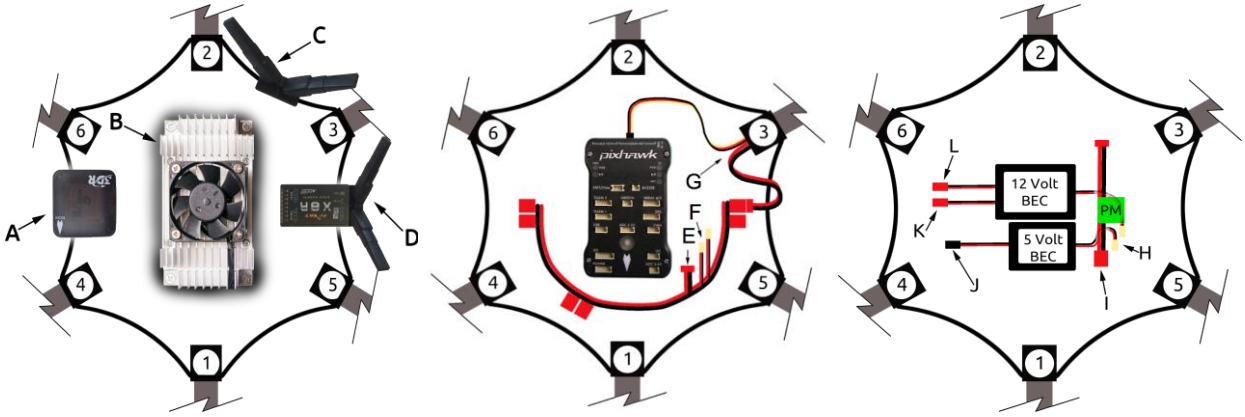


Figure 29 - From left to right are the views of the top, middle and bottom of the fuselage

In Figure 03 the top, the middle, and the bottom of the fuselage show the connections and locations of the components. A is the GPS/compass. B is Nvidia Jetson TX1 discussed in the following section. C are 2.4 GHz WiFi antennas for data transmission. D is the Radio Control Receiver which also contains an antenna. E is the input connector from the PM. F are the output connectors to BECs. G is the arm wiring coming in from the motor and the ESC. H are power inputs for the BECs. I is the output from the power meter. J is power supply for Pixhawk. K is power supply for the gimbal. L is power supply for the TX1. Wiring from the other 5 arms were omitted for clarity.

iii. Payload tray assembly

Battery - *4 cell Lithium-Ion Polymer (LiPo)*

Gimbal - *Feiyu mini 3D pro*

- Stabilizes the camera to ensure a smooth footage collection.

Camera - *Logitech HD pro webcam*

- Captures images of the desired objects and environment.

Payload tray

- This holds the gimbal, camera, and battery; note depiction in Figure 04. When the arms are folded down, the payload tray must be removed by unscrewing 4 thumbscrews.

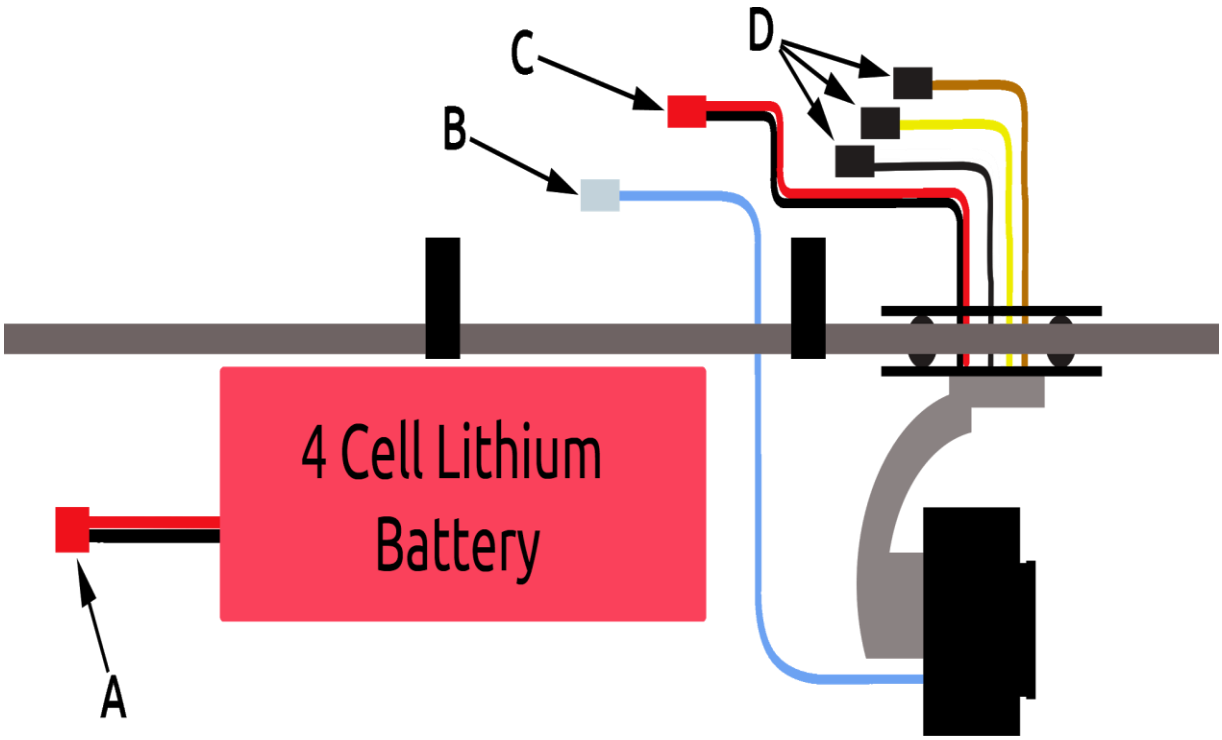


Figure 30 - Payload tray layout and connections

Figure 4 shows the layout of the payload tray: A is the power lead from the battery. B is the USB connector for the camera. C is the 12 V supply for the gimbal. D shows the gimbal's controller connections.

iv. Companion computer

If the Nvidia Jetson TX1 is purchased along with the development board, it must first be removed from the development board. To separate the companion computer from the development board, first ensure the board is powered off. Unplug the fan connector and remove the screws in each of the four corners (Figure 05). Once these connections are cleared, gently pull the TX1 from the development board to remove it.

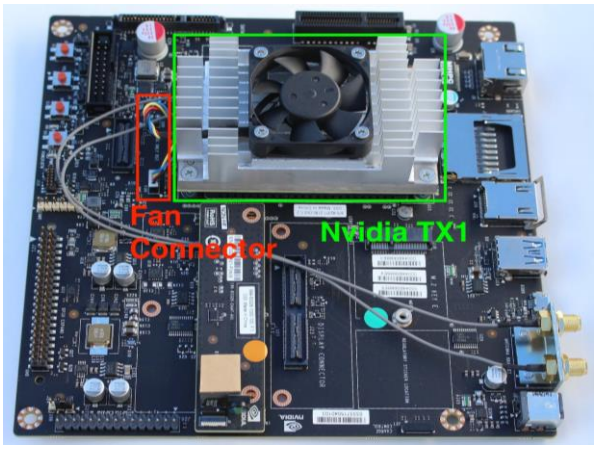


Figure 31 - TX1 and companion board

The Orbitty carrier board can then be attached through the same multi-pin connector that connects the TX1 to its development board. The carrier board and the TX1 will fit together as pictured in Figure 06. For more information, see reference 1.

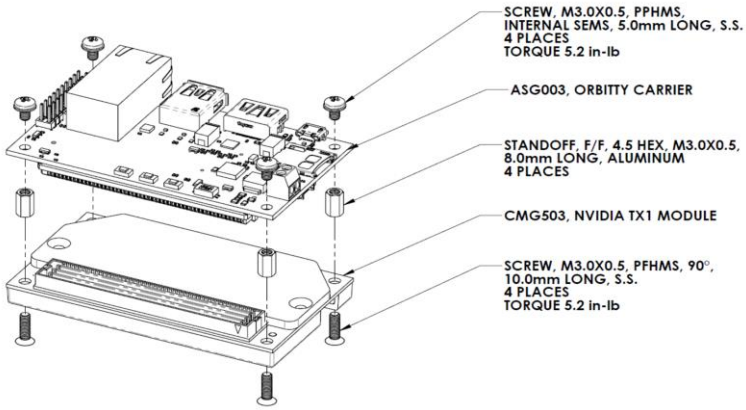


Figure 32 - Exploded view of the TX1, carrier board, and standoffs

The Orbitty carrier board connects to the Nvidia TX1 via its multi-pin connector in Figure 07.

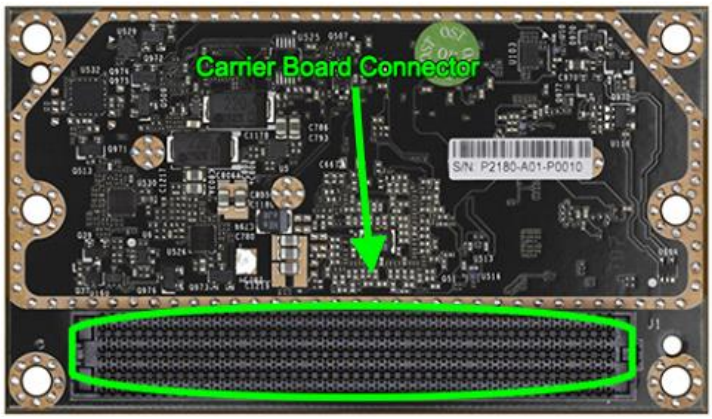


Figure 33 - Multi-pin connector of the TX1

Gently connect the TX1 and the carrier board as shown above. Install the included standoffs to secure the components and reduce the load on the electrical connection. The webcam can now be plugged into the USB port. Once the carrier board is installed, position the fan and power switches as in Figure 08.

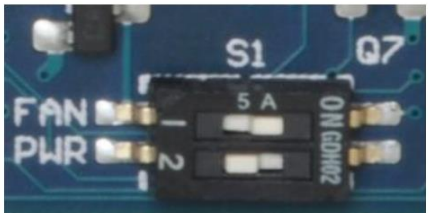


Figure 34 - Fan and power switch

Connect the positive and negative leads of the 12 V battery eliminator circuit (BEC) to the carrier board, as shown in Figure 09.

Once the power has been connected, press the power button in Figure 10 to power on the companion computer.

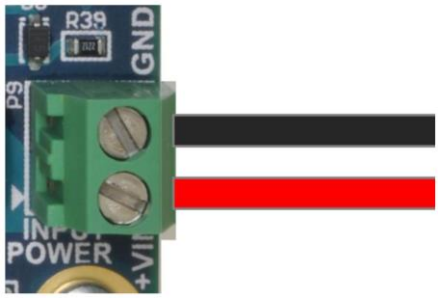


Figure 35 - Power connections to the companion computer

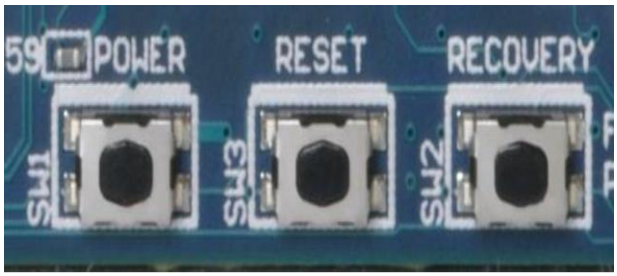


Figure 36 - Power, Reset, and Recovery buttons

The antennas located on the top of the fuselage connect to the carrier board here. These antennas are used to receive and transmit data to the ground station router. This is important in sending the processed images from the webcam.

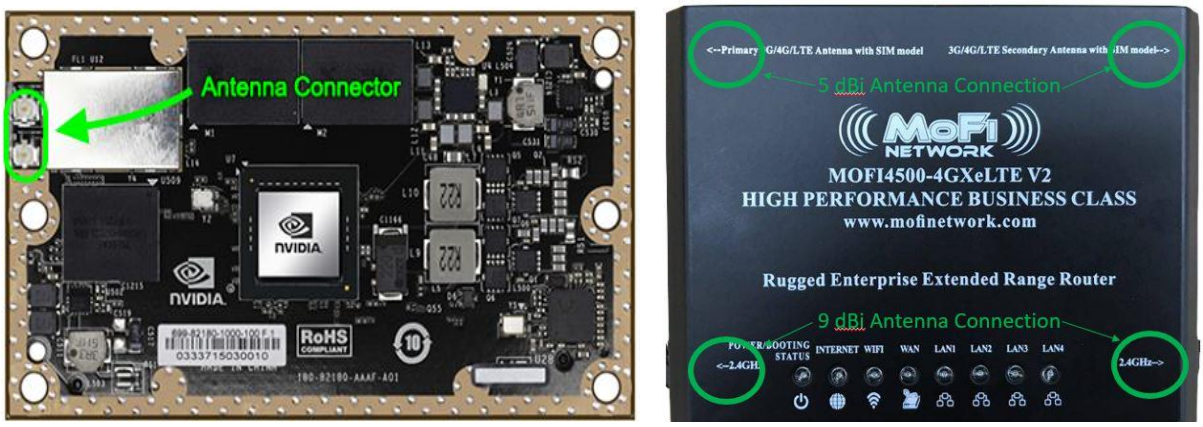


Figure 37 - Antenna connector (left) and the router for ground communication (right)

v. RC controller and wireless base station (WBS)

Extended Range WiFi Router - *MoFi 4500 4GXeLTE*

- Supports cellular data connection in addition to traditional ISP modem
- Provides 2.4 GHz signal for stable WiFi connection between drone and base station

Transmitter - *FrSky Taranis plus*

- Transmits flight control radio signal to the drone in 2.4 GHz
- Supports full telemetry with RSSI

External Antennas - *2x9 Reverse SMA Antennas*, *2x5 dBi SMA Antennas*

- 2x5 dBi Antennas extend cellular signal
- 2x9 dBi Antennas extend 2.4 GHz WiFi

b. Setup

i. Installation process

Flashing the companion computer and Installing FlytOS

JetPack, the Nvidia Jetson software development kit, is required for the initial configuration of the onboard computer. This process requires a Ubuntu x86_64 host machine for compatibility reasons. The latest version of the JetPack software (currently L4T 2.3.1) can be downloaded from Nvidia's website. The following instructions are required steps to install JetPack.

- Download most recent version of JetPack. Ensure host machine has 17 GB free space.
- Create a setup folder and put the installer inside.
- Give exec. rights to the installer by executing the following command in the setup folder:

```
chmod +x JetPack-${VERSION}.run
```

- Launch the installer, which will bring up the component manager:

```
JetPack-${VERSION}.run
```

- The component manager allows to customize which components are to be installed. Choose standard installation and wait for the installation to complete.

Once JetPack is installed on the host machine, it can flash the memory on the TX1. However, for this process both the host machine and the TX1 should be connected to the same router through ethernet ports. During the flashing process, the host machine will ask for the IP address of the TX1 on the network. Before continuing with the installation, obtain this information by going to the main page of the router and looking at the list of devices on the network.

To flash the OS on TX1, the device must be put into Force USB Recovery Mode. The following instructions explains the steps required to perform this action. Once the IP address is obtained, put the device in Force USB Recovery Mode (Figure 13).

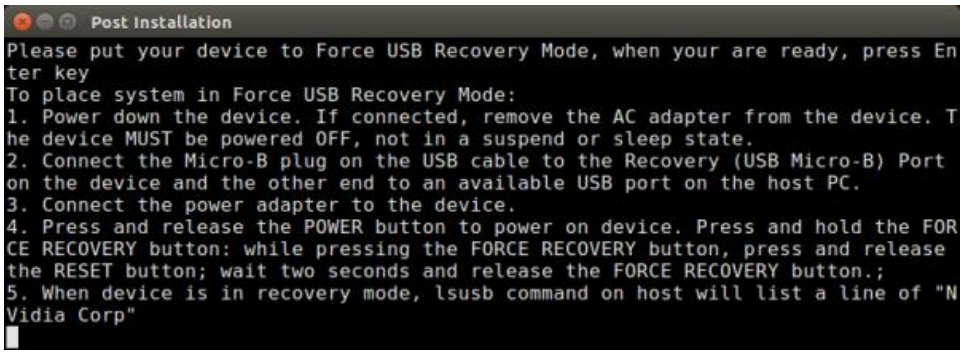


Figure 38 - Force USB Recovery Mode Instructions

- A prompt will appear to install components on the target machine.
- Wait for post-install to complete.
- Once installation tasks are completed, TX1 should be running L4T and Ubuntu 16.04.

- Connect the device to a WiFi network and install an internet browser, the terminal as the default will not contain an internet browser. The following will install Google Chrome browser on TX1:

```
sudo apt-get install chromium browser
```

FlytOS is built on Robot Operating System (ROS) thus for proper operation ROS needs to be installed on the TX1. The version of FlytOS that is used in this project is compatible with ROS Kinetic (desktop version). Do not update/upgrade the system before installing ROS-kinetic-desktop, otherwise unresolved dependencies won't allow you to install ROS later. A convenient script is available to install ROS and is used in this project. However, more detailed information regarding the Ubuntu install of ROS Kinetic can be found the ROS web page. In order the install ROS Kinetic using the script, create a new folder on TX1 desktop and issue the following commands in the terminal:

- `git clone https://github.com/jetsonhacks/installROSTX1.git`
- `cd installROSTX1`
- `./installROSTX1`

Configuring Pixhawk and Establishing Communication

Currently, FlytOS only works when Pixhawk is loaded with PX4 flight stack. The first step of configuring the Pixhawk is to connect it to a computer through USB and to flash the flight stack. This can be done through any mission planning software but in this manual

QGroundControl is used. With QGroundControl running on the computer, connect Pixhawk and it should be automatically detected. Click the firmware tab and choose PX4 flight stack. Follow the prompts until the installation is complete. The Pixhawk should now be running the PX4 flight stack.

For proper communication, there are two parameters that need to be changed on Pixhawk: `SYS_COMPANION` and `MAV_COMP_ID`. `SYS_COMPANION` which sets the baud rate of the TELEM2 port. This parameter should be set to `912600`. `MAV_COMP_ID` determines to component number and it should be set 50.

TX1 and Pixhawk communicate over UART port. To establish this communication, connect the Pixhawk's Telem2 port to TX1's UART 1 port. If using the TX1 development board, the UART1 can be found on the J17 connector. If using the carrier board, connect Pixhawk's TELEM2 port to the UART1 port on the expansion header of the Orbitty Carrier using pins in Figure 14 and 15.

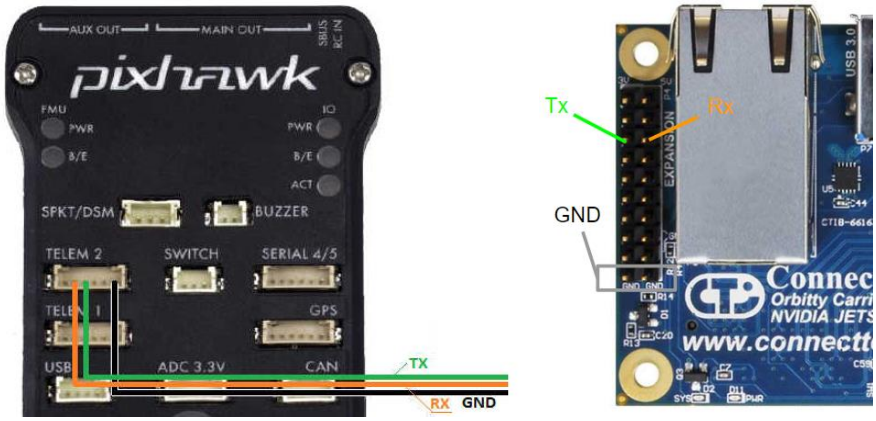


Figure 39 - Serial connection pins of Pixhawk (left) and Orbitty Carrier (right)

Installing FlytOS

The version of FlytOS that is used for this project will be provided in the designated folder on Google Drive. Before installation, it's dependencies must be installed. On TX1, open a terminal and issue the following commands in the given order:

i. `sudo apt-get install -y ros-kinetic-rosbridge-suite ros-kinetic-control-toolbox ros-kinetic-octomap-ros ros-kinetic-octomap-msgs ros-kinetic-image-proc ros-kinetic-image-transport-plugins ros-kinetic-image-transport ros-kinetic-eigen-conversions`

ii. `sudo apt-get install -y python-serial python-flask python-wtforms python-sqlalchemy python-concurrent.futures python-mock python-zmq python-twisted`

iii. `sudo apt-get install -y python-pip`

iv. `sudo -H pip install flask_cors flask_reverse_proxy flask_restful tblib webargs click flask_security httplib2`

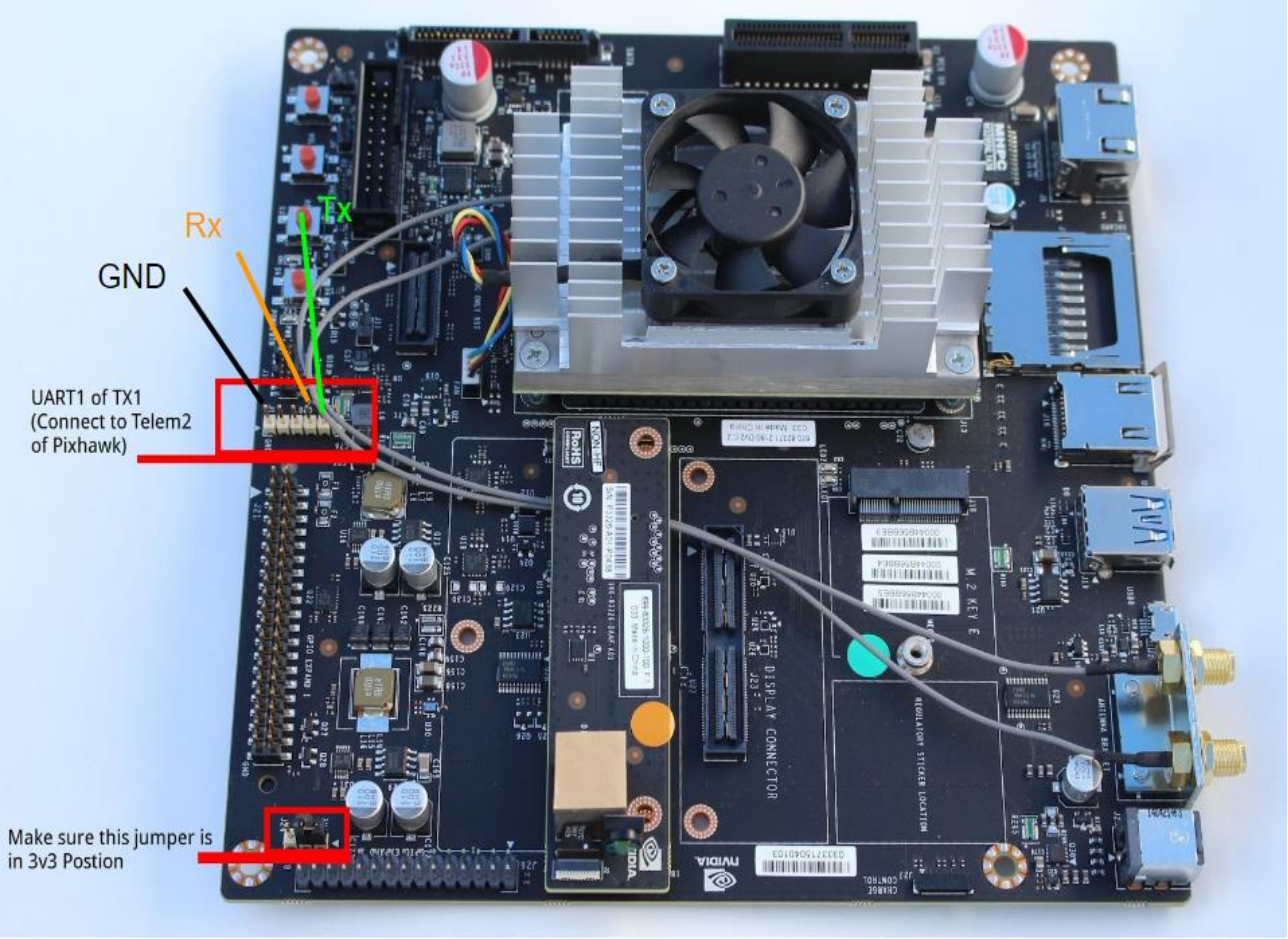


Figure 40 - Serial connection pins of TX1

Before proceeding, add the following at the end of `$HOME/.bashrc` file:

- `export PYTHONPATH=$PYTHONPATH:/flyt/flytapps`
- `source /flyt/flytos/flytcore/setup.bash`

Download FlytOS into a folder on the desktop. Issue the following command and check for the "`Congratulations! FlytOS installation completed`" message:

- `sudo dpkg -i <path to debian package location>/flytcore*.deb`

Should any dependency issues raises while installing FlytOS, run the following command and execute the previous install command again:

- `sudo apt -f -y install`

Installing GeographicLib

The location conversion services are provided using the geographic library and it is essential for proper operation. This library can be compiled and installed with the following:

i. Download `GeographicLib-1.47.tar.gz` and unpack the source by issuing the following command while in the folder that contains the tar file.

```
tar xfpz GeographicLib-1.47.tar.gz
```

ii. Change directories

```
cd GeographicLib-1.47
```

iii. Create an additional build directory and enter

```
mkdir BUILD
cd BUILD
```

iv. Compile and install the software by issuing the following two commands:

```
make
make install
```

Installing the Server Files

In addition to the web server provided by FlytOS, the aircraft hosts two more servers for location data and for the live video stream. The necessary files are located in the `customapps` folder which will be provided in the designated folder on Google Drive. To incorporate these files to the setup, paste the folder in the TX1 home directory.

Calibrating the flight computer

The flight computer needs to be calibrated before the first flight and FlytOS mission control interface provides an environment to perform this calibration. Access the interface through `192.168.1.24:9090/flytconsole`.First click on the *Frame Select* tab and inform the flight controller of the frame type of the aircraft by selecting Hexacopter + setting, shown in Figure 16.

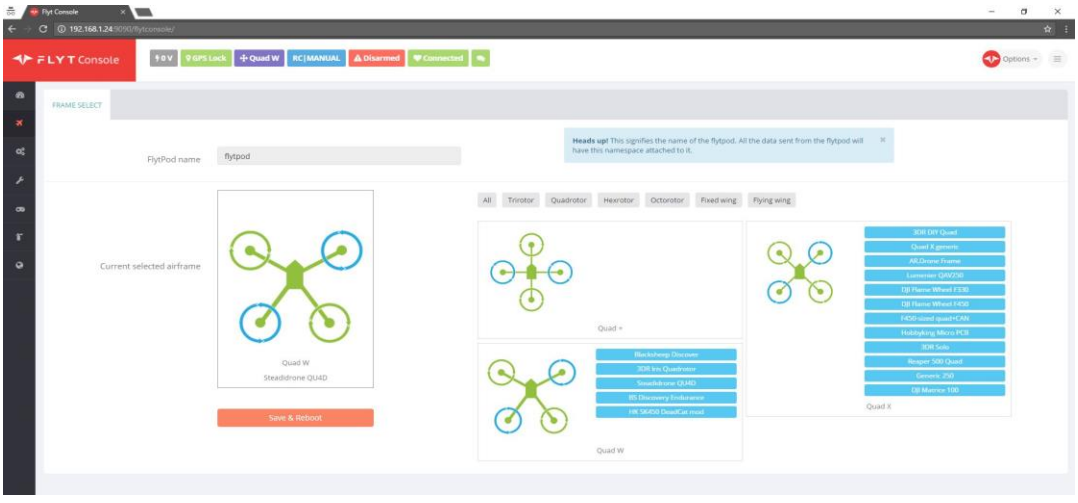


Figure 41 - FlytOS selection of frame type for calibration

The sensor calibration is consisted of four steps:

- Calibrating the accelerometer
- Calibrating the magnetometer
- Calibrating the gyroscope
- Level calibration

Click on the *Sensor Calibration* tab and complete each of the four steps, shown in Figure 17, by clicking on the respective buttons and following the prompts. Ensure that the settings are saved after completing each step.

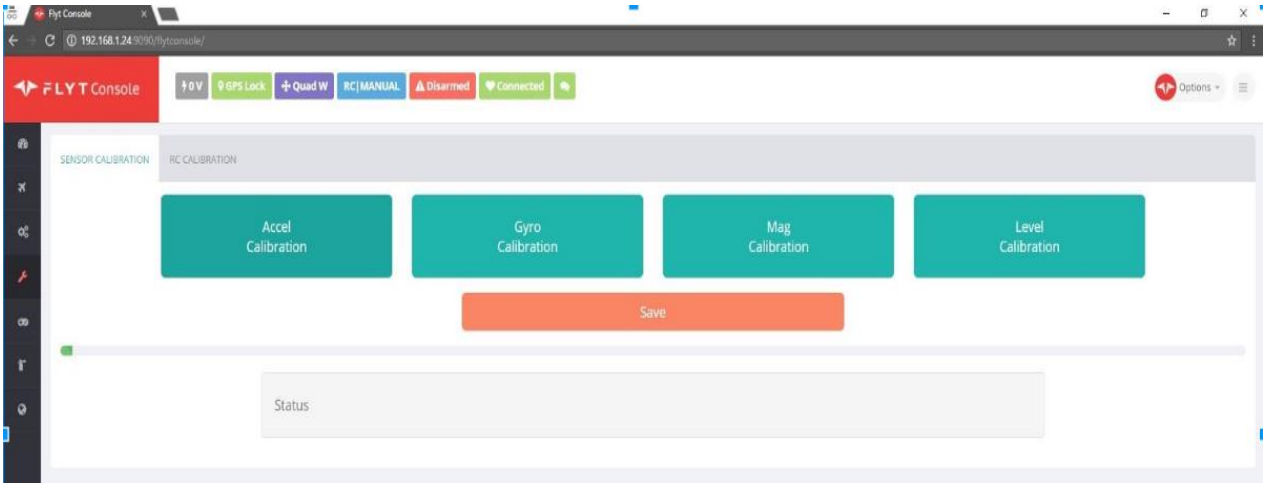


Figure 42 - FlytOS sensor calibrations

ii. Drone setup

Unfolding the arms

Assuming the drone is in folded position, the first step is to fold each of the arms and use the red aluminum screws to secure each arm in the upright position as in Figure 18. When arms are folded, the red thumbscrew screwed into the joint of the folded arm for storage.

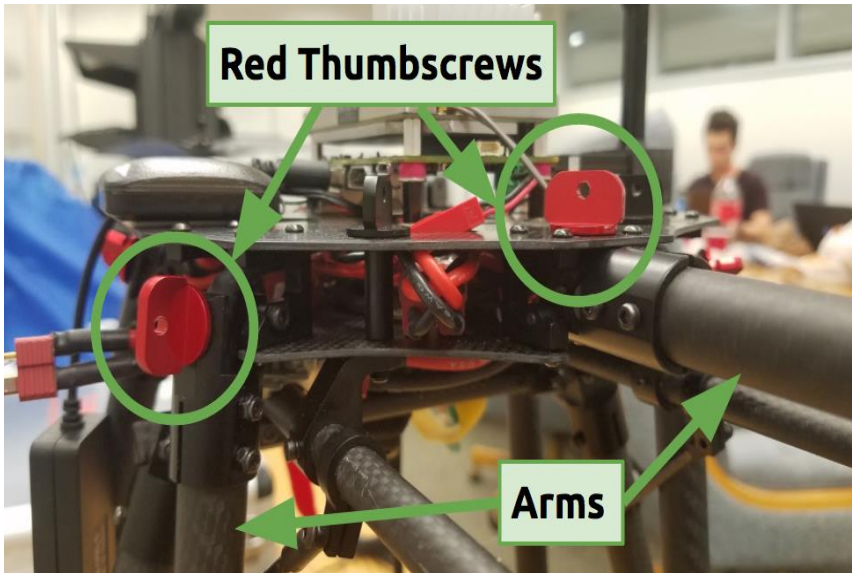


Figure 43 - Display of the red thumbscrews

Reorienting the landing gear

On arms 2, 4 and 5 the landing gear is attached on using two screws. While the function of the landing gear permits them to be placed on any non consecutive arms, the camera limits the location to the specific arms to ensure the landing gear is placed specifically out of the view of the camera. For the folded positions, the landing gear is disoriented to avoid collision into each other, thus the screws need to be loosened enough to rotate each leg to its proper position for supporting the drone. The arm and leg connection is outlined in Figure 19.

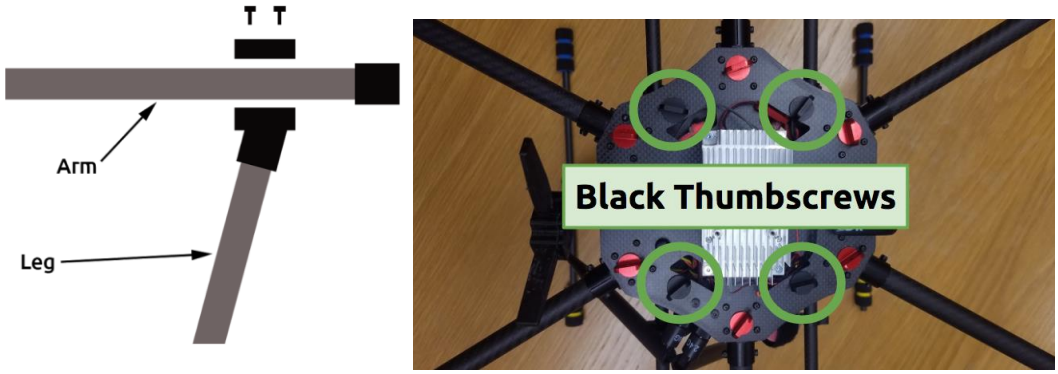


Figure 44 - Arm and leg connection (left) and Payload tray connection (right)

Connecting the payload tray

The payload tray will already have the gimbal attached. Secure the camera to the gimbal by tightening the thumbscrews on the gimbal. Connect the gimbal to the 12 BEC, and its control wires to the Pixhawk. Connect the USB wire to the camera. Strap the battery to the payload tray using the velcro straps. The payload tray is connected to the fuselage using the black thumbscrews as in Figure 20.

c. Operation

To operate the drone, connect the battery and press the marked button on TX1 to power up the companion computer. Pixhawk will power up upon battery connection. To connect a laptop or any Wifi enabled device to the drone, the device must first be connected to the wifi base station.

i. Acquire wifi access to the drone's website server. To link your base station the the drone, connect to the wifi router **FsuSR**, with the password **team04#1**.

ii. To view the flight console, enter `192.168.1.24:9090/flytconsole` into any standard web browser. The flight console will display telemetry data, vehicle operations, and other necessary flight information.

iii. To access the object detection and view the live video feed as well as the USNG coordinates, enter `192.168.1.24:9090/interface` into any standard web browser.

iv. Verify GPS health by noting BRIGHT green, flashing LED on Pixhawk.

v. Verify necessary flight plan settings at this point.
vi. Verify data transmission by noting change in attitude, heading on the ground station laptop.
vii. Have qualified pilot power up the RC transmitter, select the correct vehicle, and verify connection by noting the green LED on the X8R receiver.
viii. Press and hold the flashing red button until it stops flashing to pre-arm the vehicle.
ix. Make sure all persons are well clear of the multi-rotor from this point, until the drone is disarmed.
x. To fully arm the vehicle hold the left RC transmitter stick at its bottom, right position for more than 2 seconds. When the craft is fully armed the BRIGHT green LED will no longer be flashing.
xi. Test the motors to verify flight readiness by raising the throttle stick slightly, and returning it to low.
xii. Complete the flight mission.
xiii. After the vehicle is on the ground hold the left stick of the RC transmitter to the bottom left position until the BRIGHT green LED on the Pixhawk begins flashing. The vehicle is now disarmed and may be approached.
xiv. Power off vehicle by unplugging the battery.

Adjusting Color Filtering Parameters for Object Detection

The object detection relies on HTML input to adjust the criteria for the color filtering algorithm. While a default color is predetermined upon launching the server, adjusting the base color value for the filtering process requires the color of interest to be manually selected from a color picker provided on the server page. The code of the selected color will auto-populate the HSV fields. To adjust the filtering algorithm's boundaries, click submit. Note the HSV fields can also be populated manually.

d. Troubleshooting

The common problems and solutions are discussed here. For additional support, please consult the manufacturer's operation manual for respective components.

- FlytOS heartbeat not detected
  - Check connection between TX1 and Pixhawk.
  - Ensure Pixhawk parameters `SYS_COMPANION` and `MAV_COMP_ID` are set to `912600` and `50`, respectively.
- Drone cannot connect to the ground station WiFi
  - Check that both the TX1 and router are powered on.
  - Check the router interface to see if drone has been assigned a static IP address.
  - Remove saved WiFi connection settings from drone and reconnect to the network.

- Drone does not operate within the specified maximum range
    - Check for obstructions between drone and ground station
    - Place router on higher ground
- `192.168.1.24:9090/flytconsole` or `192.168.1.24:9090/interface` are unaccessible
    - Ensure ground computer is connected to **FsuSr** network
    - The drone should automatically join the network once TX1 is turned on. Make sure that TX1 is powered on and the power indicator LED is solid blue.

- Reading "16N" as the USNG coordinate
    - Ensure that the flight controller has GPS lock, which is indicated by the green flashing of the primary LED of Pixhawk. The LED is solid green if the vehicle is armed. The aircraft is designed to be operated outdoors only.
- Motors do not respond to the RC Controller
    - The aircraft is equipped with a two-step arming sequence. First, press and hold the arm button on the aircraft until the light inside the button starts blinking rapidly. Then move the throttle stick to the bottom right corner and hold until the light stops blinking and remains solid. This should enable the motors.
- No video feed available on `192.168.1.24:9090/interface`
    - Check if video feed is available on `192.168.1.24:5050/cam.mjpg`. If so, reset the FlytOS by restarting TX1. This should start the web server of FlytOS.
    - If `192.168.1.24:5050/cam.mjpg` doesn't have a video feed either, check the camera connection. After ensuring the camera is connected, restart TX1 which should relaunch the image processing server.
- USNG data not updating.
    - SSH into TX1 and check terminal for the error message: "`Waiting for master. Could not contact ROS master at [http://localhost:11311, retrying...`" If the error message is present, enter the following command or restart TX1 to automatically start ROS
        - `roslaunch home/ubuntu/flyt/flytos/flytcore/scripts/pr2.launch`
- FlytConsole does not display the battery status.
    - This is a known issue with the current version FlytOS. The battery widget is inoperative and currently the developer team does not have a solution. In order to track the battery status, employ an external battery meter that is directly connected to the balancing port of the battery or use an alternative mission planning software.

e. User Manual References

[1] "Orbitty carrier manual", Connecttech, 2015. [Online]. Available:http://www.connecttech.com/pdf/CTIM-ASG003_Manual.pdf. [Accessed: 4- April- 2017].

[2] “NVIDIA Jetson TX1 Developer Kit User Guide” [Online]. http://developer.download.nvidia.com/embedded/L4T/r23_Release_v1.0/NVIDIA_Jetson_TX1_Developer_Kit_User_Guide.pdf [Accessed: 4- April- 2017].

[3] “Pixhawk Quick Start Guide” [Online]. https://3dr.com/wp-content/uploads/2014/03/pixhawk-manual-rev7.pdf [Accessed: 4- April- 2017].

Appendix D - Operation Range Calculations

From the link budget equation, free space loss can be calculated by simply rearranging the equation such as below:

$$L_{FS} = P_{TX} + G_{TX} - L_{TX} - L_M + G_{RX} - L_{RX} - P_{RX}$$

Transmitter losses, L_{TX} , account for coax and cable loss which are negligible. The same applies to receiver losses, L_{RX} . Subsequently, the equation above is reduced to the one below.

$$L_{FS} = P_{TX} + G_{TX} - L_M + G_{RX} - P_{RX}$$

The specification sheet from the manufacturers provided the data below for both the router and the WiFi chipset onboard TX1. Fade margin of 30 dB is the widely accepted value among industries.

$$P_{TX} = 21\,dBm$$

$$G_{TX} = 2 \times 9 = 18\,dBi$$

$$L_M = 30\,dB$$

$$G_{RX} = 2 \times 5 = 10\,dBi$$

$$P_{RX} = -85\,dB$$

Substituting the values above for the variables, free-space loss is obtained:

$$L_{FS} = 21 + 18 - 30 + 10 - (-85) = 104\,dB$$

Free-space loss value obtained from above is equated to the standard form of the free-space loss equation. The standard form of the equation can be expressed in terms of *dB*, as shown below, for the convenience of calculating the distances:

$$\begin{aligned} \text{FSPL (dB)} &= 10\,log_{10}\left(\left(\frac{4\pi df}{c}\right)^2\right) \\ &= 20\,log_{10}\left(\frac{4\pi df}{c}\right) \end{aligned}$$

$$= 20\,log_{10}(d) + 20\,log_{10}(f) + 20\,log_{10}\left(\frac{4\pi}{c}\right)$$
$$= 20\,log_{10}(d) + 20\,log_{10}(f) + 32.45$$

Values of *d* and *f* are expressed in km and MHz, respectively, and therefore 20 $\log_{10}$ ($\frac{4\pi}{c}$) is simplified to 32.45 as the constant.

$$104 = 20\,log_{10}(d) + 20\,log_{10}(f) + 32.45$$

$$(104 - 32.45 - 20\,log_{10}(f))/20 = log_{10}(d)$$

$$log_{10}(d) = (104 - 32.45 - 20\,log_{10}(f))/20$$

$$d = 10^{((104-32.45-20\,log_{10}(f))/20)};\ f = 2450\,MHz$$

$$d = 10^{((104-32.45-67.78)/20)}$$

$$d = 1.55\,km$$

The same approach was used to calculate the operation range from the drone to the ground station with only the different receiver sensitivity, P_{TX} , and transmit power, P_{RX} .

$$P_{TX} = 12\,dB$$

$$P_{RX} = -90\,dB$$

Substituting the values above for the corresponding variable, the operation range is calculated to be:

$$d = 0.954\,km$$

Appendix E - Peak Thrust Calculations

Three Cell (11.1 V nominal) Test Results

Thrust/Unit-Power (g/W)										
Percent Throttle	**12x10 E**	**11x8.5E**	**10x10E**	**10x5E**	**10x4.5E**	**9x4.5E**	**10x4.7SF**	**10x4.5SF**	**10x3.8SF**	**9x4.7SF**
20%	7.53	7.49	5.38	8.91	9.33	7.87	10.37	8.62	9.96	8.59
40%	7.54	8.15	6.14	10.14	11.32	10.28	11.04	10.53	10.87	9.84
60%	6.00	7.08	5.40	10.17	10.59	10.34	10.75	10.00	11.03	10.03
80%	4.87	6.29	4.55	9.10	9.43	9.69	8.89	9.01	6.53	9.46
100%	4.11	5.29	3.90	7.91	8.25	8.68	7.56	7.91	8.23	8.45
Efficiency 40% - 80% Throttle (g/W)	6.14	7.18	5.36	9.80	10.45	10.10	10.23	9.85	9.48	9.78
Peak Thrust (g)	694.00	677.00	508.00	554.00	565.00	411.00	633.00	558.00	628.00	428.00

Four Cell (41.8 V nominal) Test Results

Thrust/Unit-Power (g/W)										
Percent Throttle	**12x10 E**	**11x8.5E**	**10x10E**	**10x5E**	**10x4.5E**	**9x4.5E**	**10x4.7SF**	**10x4.5SF**	**10x3.8SF**	**9x4.7SF**
20%	7.34	8.44	5.58	9.00	8.88	8.03	10.26	9.14	9.43	10.07
40%	6.15	7.03	5.30	9.67	9.80	9.46	9.66	9.71	9.63	9.95
60%	4.20	5.66	4.29	8.62	8.69	8.62	8.49	8.75	8.89	9.07
80%	3.62	4.46	3.57	7.29	7.42	7.76	6.95	7.48	7.26	7.56
100%	2.76	3.56	3.00	6.31	6.13	6.79	5.86	6.35	6.24	6.71
Efficiency 40% - 80% Throttle (g/W)	4.66	5.72	4.38	8.53	8.64	8.62	8.37	8.65	8.59	8.86
Peak Thrust (g)	900	968	780	895	910	688	1,031	922	1,037	708