Northrop Grumman Collaboration Project Report

Team Members:

Kellen Rivest
Nelson Chong
Julio Ortiz
John Hartman

Faculty Advisors:
Dr. Zekeriya Aliyazicioglu

Table of Contents

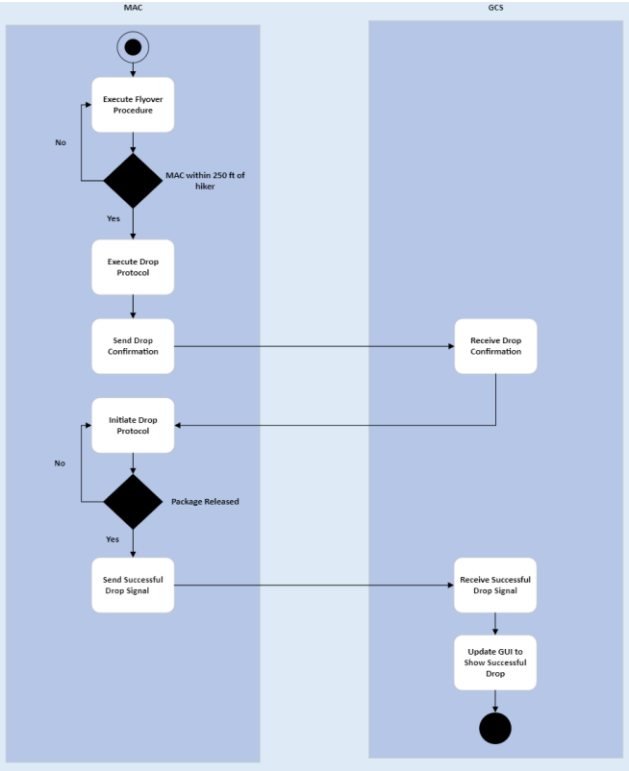
1. Introduction:

The Objective of the project is to Design, Build and Test a group of systems to fulfill a specified Request of Proposal (RFP) assigned by Northrop Grumman to demonstrate the ability to complete a program in a smaller scale of a industry program, this shows readiness in the students of the project to work in a professional adaptation of NGCP. This specific mission from Northrop Grumman was the rescue of a stranded hiker and the putting out of a simulated wildfire. The mission as a whole involved an unmanned air vehicle to locate the hiker and fire, and to drop a relief package near the hiker to aid them. The location of the hiker will trigger the response of an unmanned ground vehicle that will pick up the relief package and hiker then travel to a specified location to await a second unmanned air vehicle. This second unmanned air vehicle is called once the fire is located where it will travel to the location of the fire simulate putting out the fire and then travel to the ground vehicle's location to pick up the hiker and take the hiker to a separate location representing a hospital to drop off the hiker and then return to the vehicles point of origin. The first air vehicle will return to its point of origin after locating the fire and hiker and dropping off the package, and finally the ground vehicle will return to its point of origin after the hiker was picked up by the second air vehicle. The scale of this mission has been reduced to have achievable builds in the time and budget constraints given. The hiker is represented by a figure with sizing specified in the RFP.

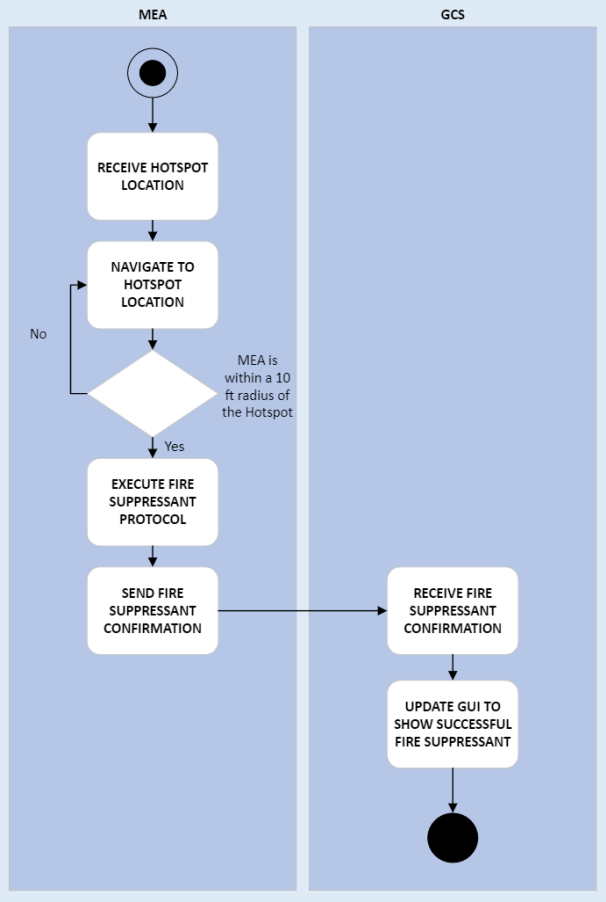
2. Systems Department

To complete the project system or vehicle has a technical team staffed by a Chief Engineer with leads for each subsystem of the vehicle such as electrical, mechanical, software, and more depending on the vehicles function and design. Additionally, there is a control station with a user interface to monitor and manually control each of the vehicles during different parts of the overall mission profile. To manage the build, facilitate intersystem discussions, develop documents like a requirements list, operational and functional architectures, use case and testing documentation a system engineering department was also put in place with a system chief and one lead per vehicle. The systems team ensures the systems created are fulfilling the requirements set by the Northrop Grumman and the derived requirements written to show how the requirements will be achieved on a technical level. Use case and Operational Diagrams shows the customer the exact actions of the vehicles will take during the mission profile. Personally, my position in the project was to lead the systems process for the Multi-role Aircraft (MAC) vehicle and later adopt responsibilities for the Medical Evacuation Aircraft (MEA) as well. I lead a team of five assigning each systems personnel to one or two technical teams to manage their draft of the requirements, sit in on key meetings, ensure the teams adherence to the overall schedule of the program, verify testing has taken place and what requirements are fulfilled by the designs made and verified by the testing done. I was also responsible for the electrical and software tech teams. Later, our team would ensure the requirements, operational and functional diagrams and data be done for the MEA and MAC simultaneously. I was also tasked with leading interface meetings necessary for relations and interactions between systems, these included the design and building of the package in coordination with the MAC

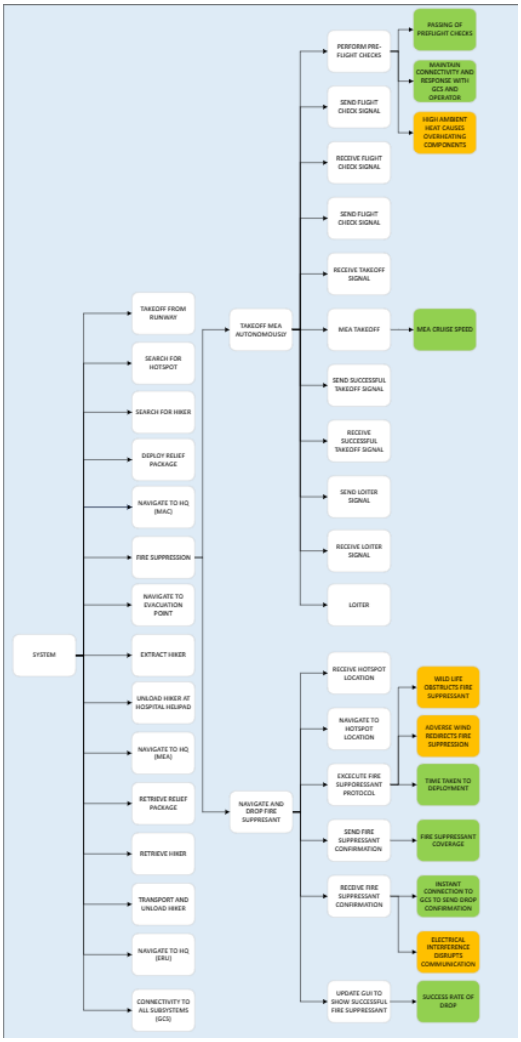
and Emergency Response Unit (ERU). Another interface need was between the Ground Control Station (GCS) and MAC to establish when the vehicle would send updates to the station and the nature of communication with the vehicle, getting the MAC software team and GCS Leads to agree and coordinate the signals and updates sent between the vehicles and the station verified the necessary steps were met during the mission we planned to fulfill all mandated requirements. The requirements were also crucial during the design process of the MAC's architecture. The architecture chosen had to be the optimal design for the given tasks of the MAC's mission contrasting priorities of system capabilities to aerodynamics and payload space to weight just to name a few, we were tasked with verifying the architecture chosen would be able to meet all of the program's needs. Later as we turned to our functional work the mission is further verified by outlining the procedure of the vehicles functions and building these diagrams off the operational architectures done the semester before. Both serve as outlines for the technical teams to fulfill the RFP given once it is time for the demonstration.



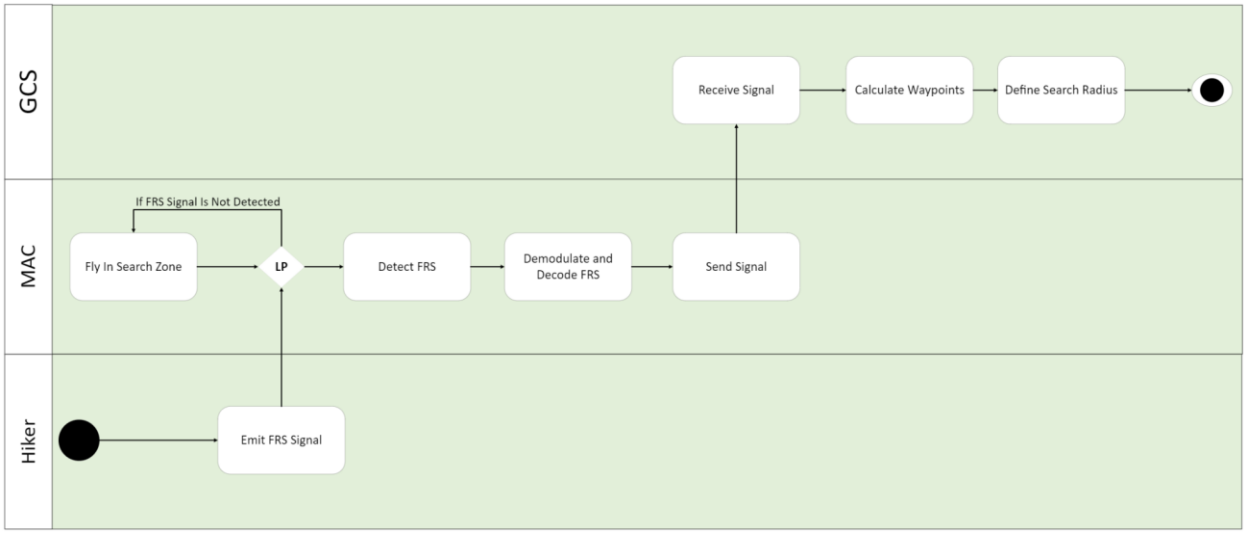
Systems Figure a) Operational Architecture 4.2 Dropping Relief Package



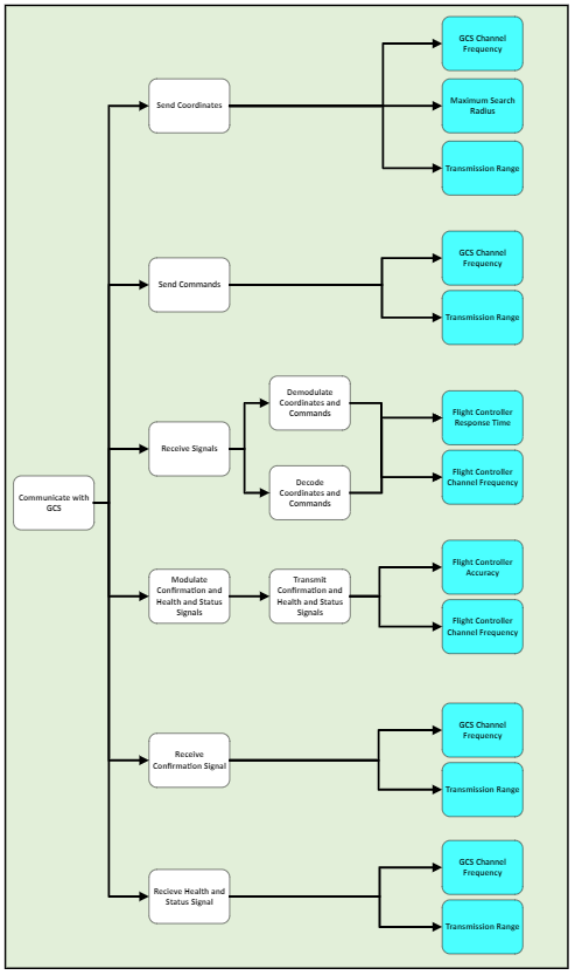
Systems Figure b) Operational Architecture 2.2 Navigate and Drop Fire Suppressant



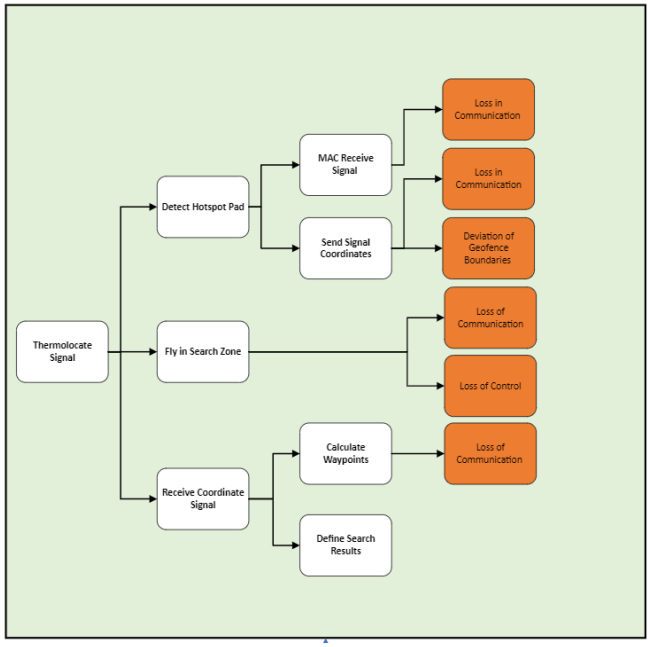
Systems Figure c) MOE Diagram 6.0 Fire Suppression



Systems Figure d) Functional Architecture 4.0 Radiolocate Signal



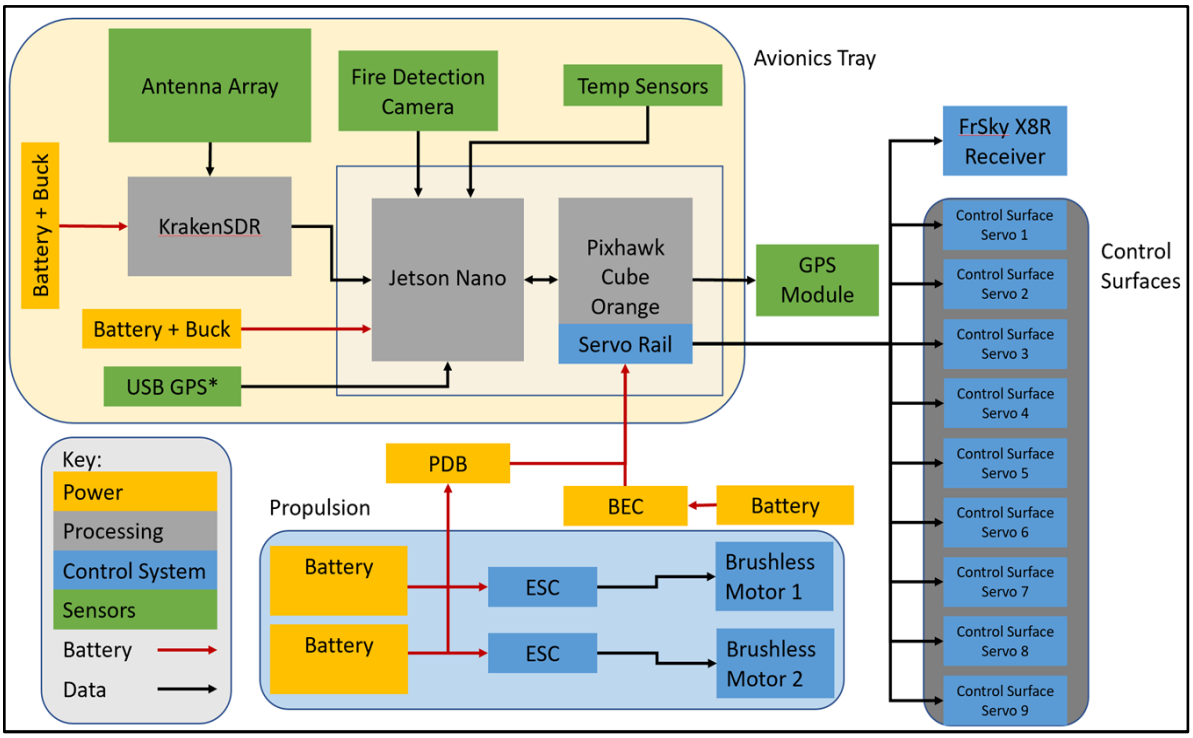
Systems Figure e) KPP Diagram 1.0 Communication with GCS



Systems Figure f) Malfunction Diagram 3.0 Thermo-locate Signal

3. Electrical

The electrical portion of the project focused on four main electrical systems aboard the MAC vehicle which we deemed necessary to complete the mission objectives, those being power, processing, control surfaces, and sensors. A wide range of components were used to allow us to fulfill the objectives of autonomous flight, radio location finding, image detection, and communications with GCS. Aside from the motors, control surfaces and some sensors, a dedicated avionics tray houses all the electronics necessary to complete the mission. The components on the avionics tray include: a GPS, a flight controller for all flight controls, including manual and autonomous flight, a software defined radio (SDR) connected to an antenna array to locate the hiker based on their radio signal, a computer that will process all incoming and outgoing data, a webcam that will be used for fire detection, temperature sensors to monitor the ambient temperature in the tray, and batteries to power everything.

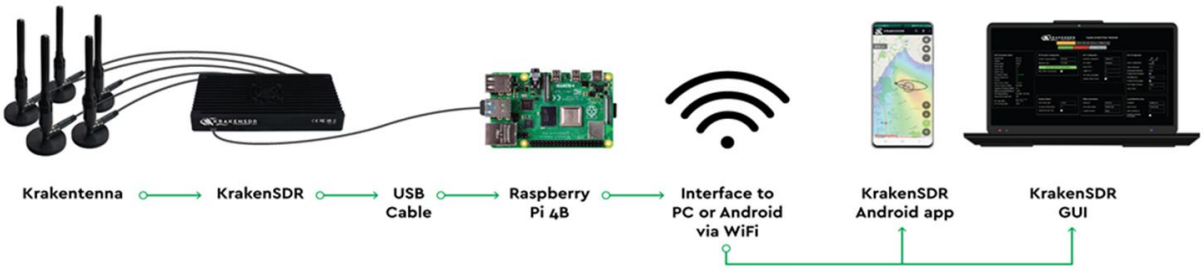


Electrical Figure a) Block Diagram of Electronics

For the electrical team, the tasks were separated into three different categories: Radio Location Finding, Avionics Tray, and Circuitry and Wiring. Tasks falling under Radio Location Finding focused on researching possible methods to use, ensuring the chosen method would be compatible with all the necessary electronics, and testing to meet the accuracy requirements asked for by the RFP. Tasks for the Avionics Tray included modeling the tray as well as the components it houses given the sizing limitations of the vehicle's design, mapping the connections of the components on the tray, mounting the components to the tray, and ensuring the weight of all the components falls within an acceptable range given by the structures team. The Circuitry and Wiring category of tasks deals with finding the appropriate batteries for the

electrical components to last for the entirety of the mission, creating the electrical wiring schematic detailing all the components' connections, creating the connections needed for all the components, deciding which components would be needed, and confirming that all components are compatible and will not fail while carrying out the mission through testing. As the electrical team lead, I delegated and oversaw the progression of all the tasks, but I focused primarily on Radio Location Finding and Circuitry and Wiring.

To meet the objective of locating the hiker, research had to be done on what methods of radio location finding would be the most feasible given the chosen design of our vehicle. Through a combination of analysis of past year's radio location finding method and creating a trade study of possible methods, we came to choose a method of radio location finding known as Correlative Interferometry. This method of radio location finding had the most ideal amount of accuracy, range and compatibility, the field of merits (FOMs) which we deemed had the most weight in achieving our objectives. The piece of hardware we chose that will allow us to perform this method is a relatively new one and is the first of its kind. Seeing as this is a relatively new method, not many commercial off the shelf products exist as do for some of the other methods, however, we managed to find a commercially available device that would be compatible with the other parts that it would connect to. The device is a software defined radio (SDR) called the KrakenSDR and it functions in cohesion with a five antenna array that must be placed in specific positions based on the desired frequency for which we needed to search. By having each antenna channel synchronized on one internal clock, the KrakenSDR takes the difference in phase angles received and measures them against each other to come up with a location of the signal for which we are searching. Once the location is pinpointed, a microcontroller, at least as strong as a Raspberry Pi 4, is used to transmit the data to GCS via WiFi.

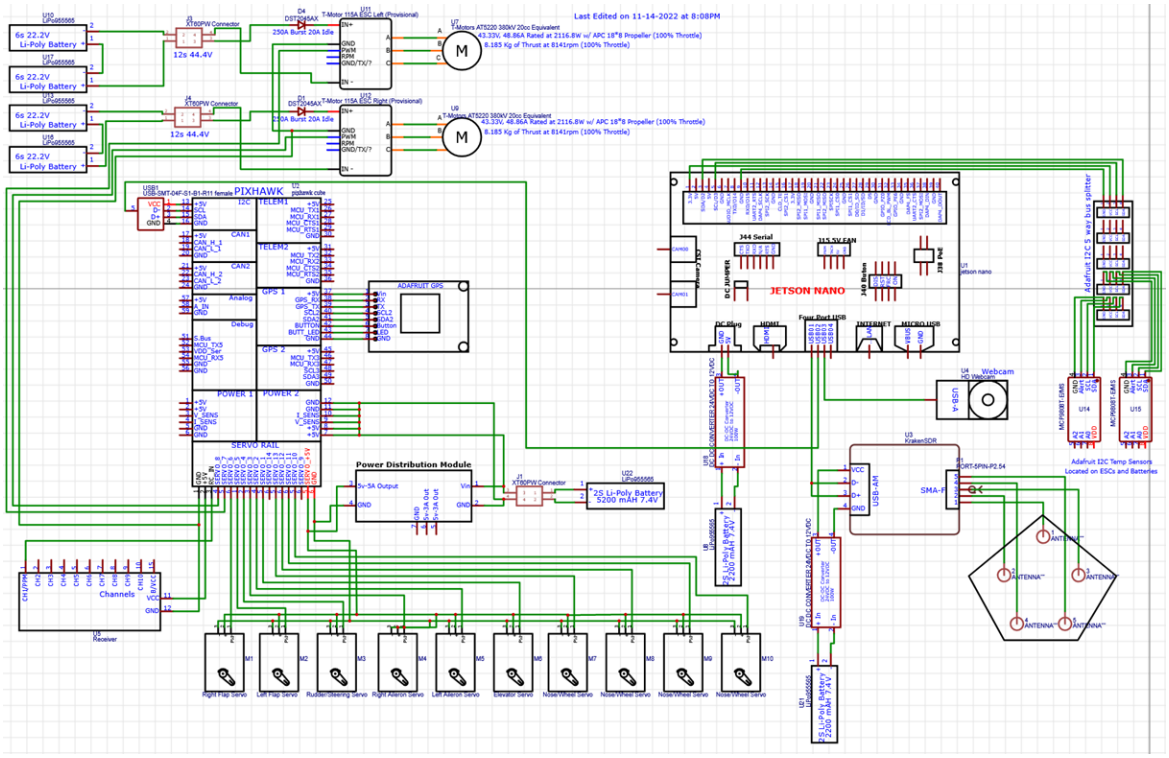


Electrical Figure b) KrakenSDR Flow Diagram

Our onboard computer we have chosen that will handle the majority of the data processing and communications with GCS is the Jetson Nano 4gb. This small yet powerful computer was chosen over other choices given its low power consumption, the high amount of port availability outside of the 40 pin GPIO (general purpose input/output), and a Quad-core ARM A57 CPU that will enable the high levels of data processing involved in performing communications with GCS, radio location finding, and image detection aboard the UAV. Due to the Jetson Nano running Linux as its operating system, we had to choose electronics that would be compatible with Linux. To confirm our sensors and other components would be compatible with the Jetson and the batteries would be sufficient to power all our components, we tested

each component individually before testing the entire system together. Through these tests, we were able to confirm that all necessary devices, those being the KrakenSDR, the flight controller, and the camera for fire detection, could effectively communicate with the Jetson Nano.

After we had confirmed that all the components could effectively communicate and the Jetson Nano could perform all the processing required, our focus shifted to circuitry and wiring. We used an online PCB design tool to demonstrate all the electrical component's connections, including the systems outside of the Avionics Tray, such as the motors, control surface servos, and payload system. Each connection was analyzed to ensure safe power draw from the power supply. Once we finalized the routing and connections, we were able to test the entire system and confidently conclude that the Jetson Nano would fulfill our requirements for the project.



Electrical Figure c) Electrical Wiring Schematic

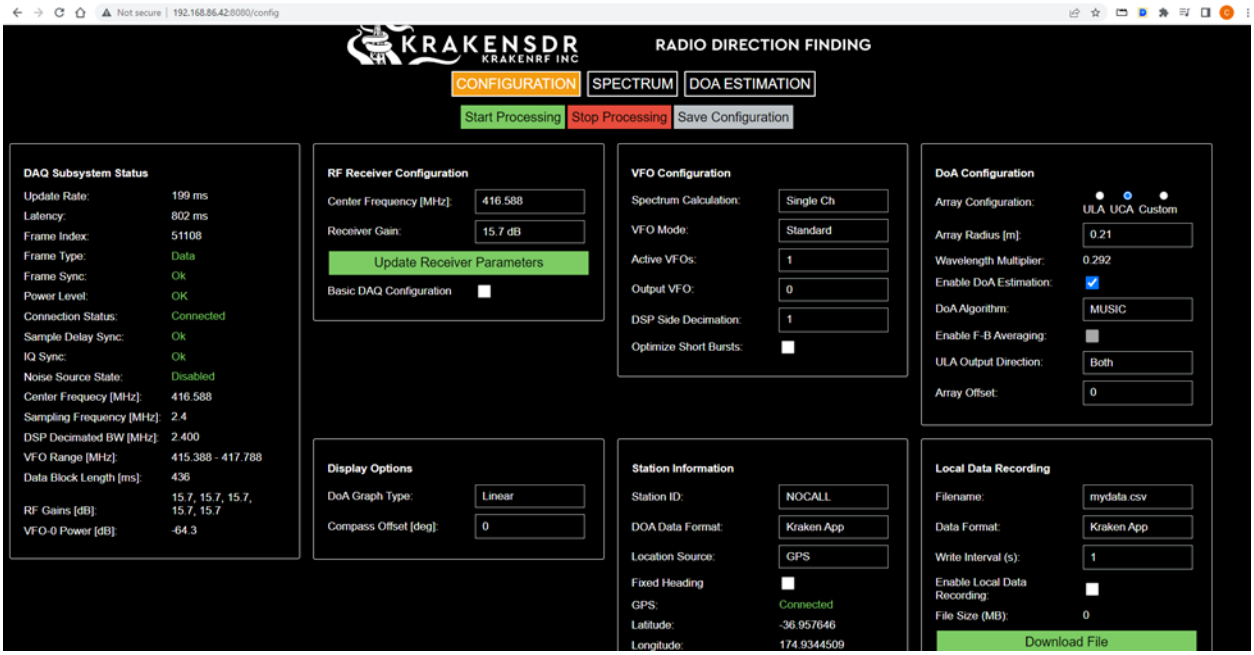
4. Software

The software used starts off with the use of Mission Planner. This was a simulation that we can use to test our flight controller. The flight controller we used is called the PX4 Cube Orange Flight Controller. The flight controller also came with a GPS that we used for both flight simulation testing and actual flight tests with a model plane. This flight controller also provides the ability to perform autonomous flight. For our UAV to be able to process, send and receive data, we also had to purchase an onboard computer for the UAV. For that, we decided to use the Jetson Nano. The Jetson Nano is a small and powerful computer that can run many applications that will help us, including image classification and object detection.

Within the software team, the work had to be divided into a few sections: Communications, Radio Location, and Fire Detection. In my case, I worked in the Communications side of things.

The Communication side required the use of radio hardware and software. For that, we decided to use the KrakenSDR. The KrakenSDR is an SDR that can track GPS coordinates depending on the location of the physical device. The data includes a CSV file that gives the user the latitude, longitude and the accuracy of the coordinates recorded. Since we were in the software team, we had to work closely with the electrical team due to their experience in the power and electronics portion of the KrakenSDR. In fact, one of the main issues that both of our teams had was figuring out what kind of software-defined radio we needed to use. Last year's NGCP teams worked with GNURadio, but the issue with that software is that while GNURadio could track GPS coordinates just like how the KrakenSDR can, we could not figure out how to set up the GPS data collection in real time.

Compared to GNU Radio, the KrakenSDR comes with a GitHub that has all the documentation needed to effectively test the software and the data collection. In addition to documentation, GitHub also includes downloadable software in both file format and in the form of an Ubuntu Virtual Machine. The KrakenSDR software also includes its own GUI (graphics user interface) that can be found after running the proper script from the downloaded software. The GUI has many settings that can be configured so both the software and the electrical team can manipulate whatever is needed. However, the implementation of real-time GPS data collection still had to be solved on our own. As of now, the KrakenSDR software is only able to collect data and place it in a file somewhere in the system. Therefore, we had to write up Python code to implement the real-time data collection.



Software Figure a) KrakenSDR GUI

The first thing we did was to test the ability to parse the data from the CSV file. This was done using two different Python scripts. The first was called KrakenSim.py and its whole purpose was to collect data from a custom CSV file we created and place all that data into another CSV file. This is to simulate pulling data from the KrakenSDR's csv file. The second Python script is called Parser.py and how this script works is that it checks for the lowest possible value in the 'array' of numbers being added to the new CSV file. Once it finds a value it will print it out. Parser.py was created so we can output the latitude, longitude, and confidence values on the terminal so we can read the coordinates.

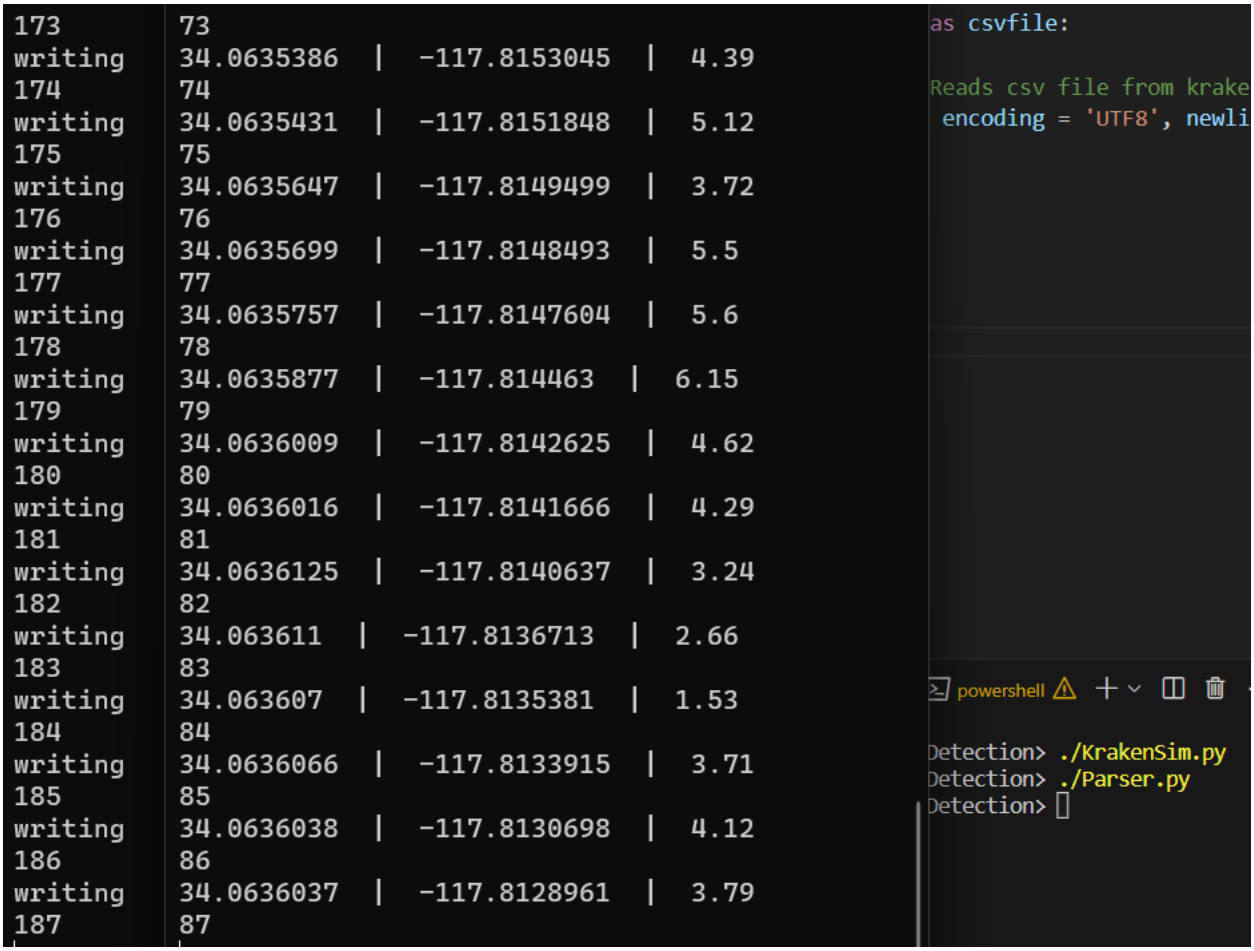
```python
import csv
import time

rowNumber = 0
with open('KrakenOutputSim.csv', 'r') as csvfile:
    # Create a CSV reader object
    reader = csv.reader(csvfile)    # Reads csv file from krakensdr
    with open('Krakenoutput.csv', 'w', encoding = 'UTF8', newline = '') as f:
        writer = csv.writer(f)
        for row in reader:
            print('writing')
            writer.writerow(row)
            print(rowNumber)
            rowNumber = rowNumber + 1
            time.sleep(1/4)
```

Software Figure b) Code for KrakenSim.py

```python
time.sleep(1/2) # See Update 1.4
while confidence < 8.3:
    # Open the CSV file
    with open('krakenOutput.csv', 'r') as csvfile: #('/home/krakenrf/krakensdr_doa/krakensdr_doa/poop.csv', 'r') as csvfile:
        time.sleep(0.5) # See Update 1.5
        final_line = csvfile.readlines()[nextRow]    # Pulls data from the last line of the csv file (most updated)
        lastRow = final_line.split(',') # Converts string of data into elements in a list seperated by a ,
        longitude = lastRow[8]  # Longitude coordinates
        latitude = lastRow[9]   # Latitude coordinates
        confidence = float(lastRow[2])   # Confidence values
        print(longitude, ' | ' , latitude, ' | ', confidence)
        print(nextRow)
        nextRow = nextRow + 1
```

Software Figure c) Code snippet of Parser.py



Software Figure d) Simulation testing of the KrakenSim.py and Parser.py scripts. From left to right: latitude | longitude | confidence values.

From figure d, the left terminal output indicates that the pulling of data from the data simulated csv file is successful and is recorded onto an empty csv file. The right terminal output shows latitude, longitude, and confidence values from left to right.

As of now, the confidence values tested were low, and still needed to be fine-tuned. However, once we found out that the simulation works and is possible, we tested this on the actual CSV file given to us from the KrakenSDR. In the case of testing with the KrakenSDR CSV file instead of a simulated csv file, it also works, but there is a problem.

The problem with our current technique of gathering data from the CSV files is that because the Parser.py script is supposed to grab the very last line of GPS data being collected, there is a chance that the script may fail. This failure is caused by the Parser.py script attempting to output the next last line of GPS data when the KrakenSDR has not updated the CSV file yet. In order to fix this problem, we added sleep functions in an attempt to synchronize the Parser.py data collection and the updating CSV file from the KrakenSDR. This idea of adding sleep functions also has its own issues as well. We must hardcode the sleep function, but to get a working Parser.py script, the sleep value needs to be adjusted according to the

network connection of the KrakenSDR software. If KrakenSDR's data collection is slow the Parser.py sleep function needs to be increased so that the Parser.py script outputs the data after KrakenSDR records the data into the CSV file. As of now, the best way to continue with this is to work with the hard coding of the sleep function, as there is currently not enough time to make any more adjustments.

5. Testing and Results

The various electrical systems that have been tested in-flight include the Pixhawk 4 with Cube Orange microcontroller, the FrSky X8R flight receiver, a 40A electronic speed controller, a 3300mAh four-cell lithium-polymer propulsion battery, the Jetson Nano microcomputer with a 5V buck converter power supply to power it from a 7.4V 2200mAh 2 cell lithium-polymer battery, a logitech 1080p webcam, and four micro servos controlling the left aileron, right aileron, elevator and rudder of an E-flite Apprentice STS 1.5m wingspan R/C airplane which served as a stand-in for the 50% scale model (which had a very restricted total payload capacity) until the 100% scale model can be built by the manufacturing and safety team, which will be large enough to house the additional electronics for complete integrated in-flight systems tests. In-flight electrical testing on the 100% scale model will consist of the antenna array, an additional seven servos for a total of eleven (eight for flight/ground steering, three for payload), and the Kraken software-defined radio-direction finder. The initial flight tests of the Apprentice were to tune the parameters of the Ardupilot settings for the microcontroller such as the pitch, roll, and yaw axis rates, which were initially too fast and would likely cause damage to the 100% scale model due to the increased mass and inertia if not reduced. The next flight-test phase for the Apprentice included running the software team's navigation code for autonomous flight at specific altitudes of 50, 100, and 200 feet AGL (above ground level) and GPS waypoints at Cal Poly Pomona's Spadra Farm. Next, we tested the Apprentice with the Jetson Nano onboard and were able to link it during flight with an Ayrmesh long-range 2.4GHz Wi-Fi transceiver and antenna system on the ground and were able to sustain a strong connection beyond 1,500 feet of lateral distance. During this test, the software team was able to upload new commands to the Jetson Nano and update the waypoints in the Pixhawk 4 during mid-flight.

Flight-testing for the 50% model had mixed results that eventually led to major structural changes with the 100% design. The only electronics used on the 50% were the flight electronics used in the initial Apprentice tests only sufficient for manual flight. The first iteration of the 50% scale model had two major flaws resulting from a miscalculation for the center of gravity so that it was unable to maintain straight and level flight under any conditions as well as a high aspect ratio wing design that increased high-speed cruise efficiency at the expense of total loss of lift in even low degree turns at speeds much lower than what was estimated to be 40mph. Normal flight at that scale calls for stall speeds closer to 25-30mph. These design errors resulted in two separate crashes of the 50% scale model. The center of gravity issue was addressed by removing almost all of the extraneous material from the aft portion of the fuselage and moving the tailplane forward several inches. The low aspect ratio wing issue was partially addressed by using the roof of a moving car as a platform to launch the airplane on takeoff.

This third flight was successful with high-speed, slow rate-of-turn several circuits around the field that culminated in an expected stall and crash on final approach.

For the 100% model, the aft portion of the fuselage behind the main wing was completely removed and replaced with lightweight carbon fiber longeron booms and foam covering to address the center-of-gravity issue and the high aspect ratio of the main wing was also removed in addition to more area added for slower flight characteristics upon landing. Upon testing the 35kg torque servos for the 100% scale model, it was found that each of servos purchased by the manufacturing and safety team drew about 7A at 5V during stall torque which would draw 280W of power from just the 8 flight control surfaces alone and 385W with the payload, steering, and flight control servos operating simultaneously, which was an unacceptably high power requirement. The servos were exchanged for 25kg high-speed servos which draw 1.5A each for a total maximum power consumption of 82W with all servos operating simultaneously during the payload drop sequence. There were also issues with unreliable power supplies used for testing both the Jetson and Kraken. The 100% model will use an industrial buck converter made by Bel Power Systems Inc. with verified current and voltage specifications to power the Kraken SDR, which is our most difficult hardware to source.

Testing and Results Figure a) Successful 50% Scale Model Flight Test During Attempt 3

When it came to testing the KrakenSDR, we ran into a few issues early in testing. The first was a problem that we, initially, believed was due to the nature of the hardware itself in that it would take a while for the device to connect to the KrakenSDR server through the web GUI and begin processing data. We carried on while taking this issue into consideration as we moved forward. Then, in later testing, we ran into an issue where one of the antenna's channels stopped working and we believed it to be due to a power surge, which could have happened sometime during testing with a different battery than we had first used in testing. However, after replacing the device for a new one, we found that the new KrakenSDR device connected to the

server and began processing data much quicker than the KrakenSDR we had originally received. This fact led us to believe that we initially had a defective device as opposed to a power surge. Our next issue occurred during subsequent testing where all the antenna channels on the KrakenSDR stopped working due to a short in the buck converter, a variable voltage regulator, we had been using. This issue led us to explore other options to replace the buck converter, upon which we decided on a Bel DC/DC Converter.

Output Specifications

Parameter	Min	Typ	Max	Notes
Output Voltage Set Point	-2%Vo,set	-	2%Vo,set	Vin=12 V, full load
Output Voltage Set Point [1]	-2.5%Vo,set	-	3.5%Vo,set	
Load Regulation	-0.7%Vo,set	0.4%Vo,set	0.7%Vo,set	Io=Io, min to Io, max
Line Regulation	-0.7%Vo,set	0.3%Vo,set	0.7%Vo,set	Vin=Vin, min to Vin, max
Regulation Over Temperature (-40 °C to +85 °C)	-	0.5%Vo,set	-	Tref=Ta, min to Ta, max
Output Current	0 A	-	6 A	
Current Limit Threshold	6.8 A	-	15 A	
Short Circuit Surge Transient	-	0.25 A^2s	-	

Bel Fuse Inc. 206 Van Vorst Street, Jersey City, NJ 07302 • Tel 201-432-0463 • Fax 201-432-9542 • www.belfuse.com

NON-ISOLATED DC/DC CONVERTERS
4.5 Vdc -14 Vdc Input 0.75 Vdc - 5.0 Vdc/6 A Output

Jan. 25, 2013
Bel Power, Inc. , a subsidiary of Bel Fuse, Inc.

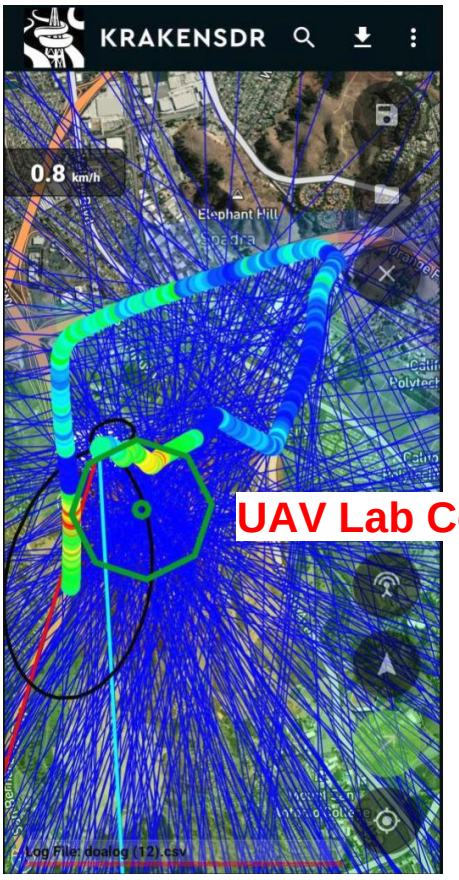
Output Specifications(continued)

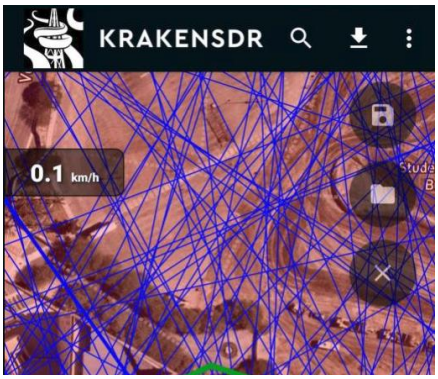
Parameter		Min	Typ	Max	Notes
Ripple and Noise (pk-pk)					Tested with 0-20 MHz, with 10 uF/10 V tantalum capacitor & 1 uF/10 V ceramic capacitor at the output
	Vo=5.0 V	-	100 mV	140 mV	
	Vo=3.3 V	-	80 mV	120 mV	
	Vo=0.75 V	-	35 mV	7 0 mV	
Ripple and Noise (rms)					
	Vo=5.0 V	-	35 mV	50 mV	
	Vo=3.3 V	-	25 mV	40 mV	
	Vo=0.75 V	-	10 mV	15 mV	
Turn on Time		-	6 mS	12 mS	
Overshoot at Turn on		-	0%	3%	
Output Capacitance					
	ESR ≥ 1mohm	0 uF	-	1000 uF	
	ESR ≥ 10mohm	0 uF	-	2200 uF	
Transient Response					
50% ~ 100% Max Load	All outputs	-	200 mV	350 mV	di/dt=2.5 A/uS; Vin=12 V; and with 10 uF/10 V tantalum capacitor & 1uF/10 V ceramic capacitor at the output.
Settling Time		-	25 uS	50 uS	
100% ~ 50% Max Load		-	200 mV	350 mV	
Settling Time		-	25 uS	50 uS	

Testing and Results Figure b) Data Sheet of Bel DC/DC Converter to Power Kraken SDR

The simulated hiker detection (Kraken SDR) was tested via passenger vehicle in order to serve as an analogue to the multi-role-aircraft (MAC) that will ultimately implement it. The trajectory of the can be seen in figure d) and varies in radius from the detected target between 1600 feet to Interstate-10 and one statute mile to the Temple Blvd exit of Interstate-15. Color shading of the route indicates the signal strength of the target at any given point with red as the weakest to green as the strongest. Results of this test revealed that obstructions such as buildings and hills reduced the overall accuracy of the hiker detection, which should be greatly mitigated in continued tests at at least 100 feet of additional altitude above the target with the MAC during the detection phase.

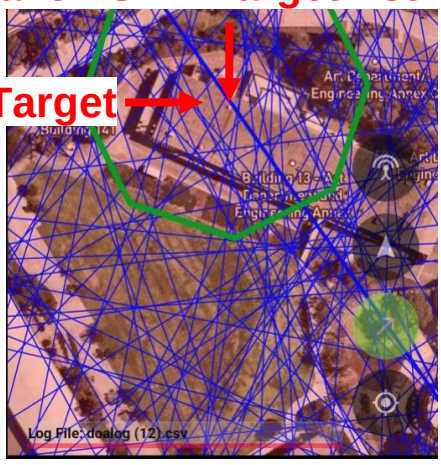
Testing and Results Figure c) Kraken SDR (left) and Antenna Array (right) Mounted on Vehicle





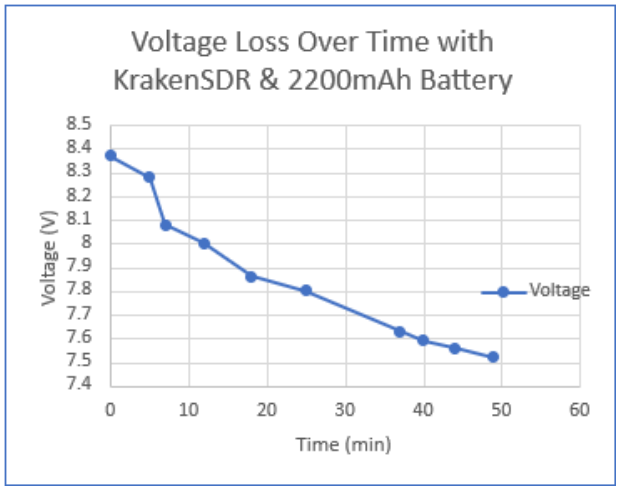
Kraken SDR Target Estimate

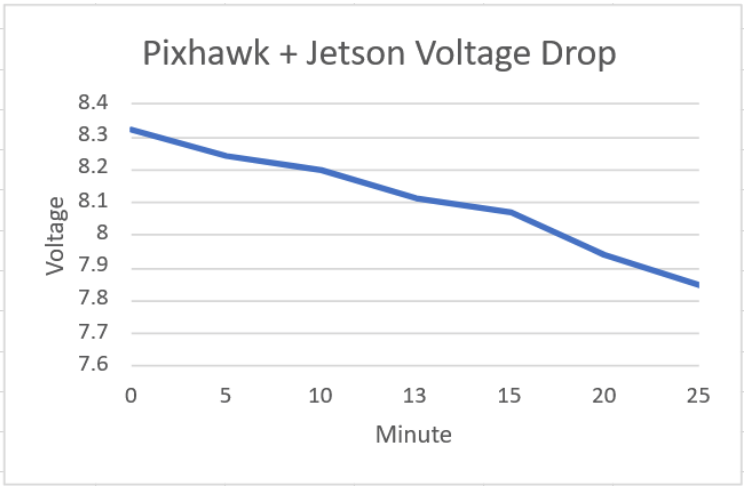
UAV Lab Containing Radio Target



Testing and Results Figure d) Two-Dimensional Trajectory of Vehicle on 10 and 57 Freeways, and Kellogg Dr Carrying Kraken SDR (Left)

Testing and Results Figure e) Location of Family Radio Service Band Transmitter in the UAV Lab at the Engineering Annex. (Right)





Testing and Results Figure f) Battery Endurance on Separate Kraken & Jetson/Pixhawk Circuits (2S Li-Po Storage Voltage: 7.4V)

Fire detection testing was undertaken using the Apprentice STS as the supporting platform using a 1080p Logitech webcam pointed downward under the midsection of the fuselage and recording into the SD card of the Jetson Nano in-flight. Issues with compass calibration on this test day prohibited accurate autonomous traversal of the fire target GPS waypoint at the desired altitude in order to calibrate the image detection software for distance. The Apprentice was hand-flown over the target to generate the images in figure g), so further testing onboard the 100% in autonomous mode is still required to calibrate the distance to ground.

Testing and Results Figure g) Masked Image of Simulated Fire (left) and Simulated Fire (right)

6. Conclusion

The architecture chosen has undergone sufficient testing, verification, coordination, and review, to the best of our ability, to verify that the enterprise or system of systems has the technological capability to complete the mission given by Northop Grumman's RFP. The MAC has undergone subsystem testing using a prebuilt remote-controlled aircraft to fly embedded systems needed for the mission such as sensors and flight controllers proving they can function independently. We have also constructed a half scale model which could be constructed more quickly along with wind tunnel and propulsion testing to show the aircraft's aerodynamics and ability to fly for the duration and speed needed to complete its part of the mission. We had two review presentations with Northrop Grumman: a System Requirements Review (SRR) and a Preliminary Design Review (PDR). Both noted different pros and cons of the aircraft architecture as well as the derived requirements presented. While these critiques have been mitigated or corrected, the main elements were as follows: The requirements gave no priority ranking to the finding of the hiker and hotspot, given the named objective of the mission is the rescue of the hiker, an emphasis should be able to be seen by the customer to ensure the decisive hierarchy of action taken in the mission. The aerodynamics of the aircraft and its overall design was heavily scrutinized during the SRR, the fuselage and chassis were not aerodynamic enough to comfortably ensure flight for the needed duration but was firstly chosen for ease of manufacturing in order to test sooner in the program's lifecycle. The existing architecture was adapted to be faster but requires high speeds to stay stable in flight. This still has a concern of ease of autonomous flight response as well as the feasibility of manual flight, takeoff, and landing. Leading up to the date of demonstration, more flight testing and manual flight rehearsals are scheduled to learn from any likely failure points in the mission to have a successful run, in front of customer personnel.