Christopher Newport University / IMPRINT

2017 AUVSI SUAS Competition Journal Paper

Unmanned Aerial Systems Team

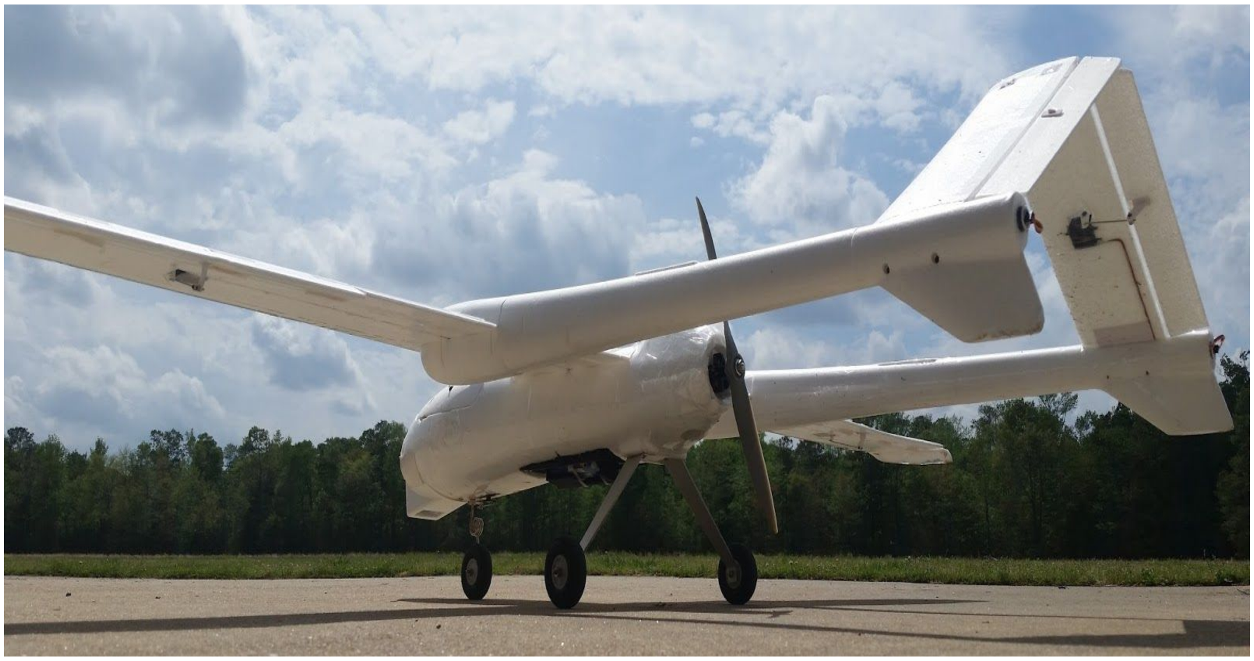

Figure 1: Anaconda

Abstract

This journal demonstrates the design and development of the modified RMRC Anaconda built to compete in the 2017 AUVSI SUAS competition. The team CNU/IMPRINT from Christopher Newport University in Newport News consists primarily of undergraduate students majoring in Computer Engineering, Computer Science, and Electrical Engineering. The team's focus is to surpass last year's performance utilizing an ever expanding team and growing resources to drive our ambitions. This unmanned vehicle is the product of rigorous design, analysis, computer and flight simulations.

Table of Contents

1. Systems Engineering Approach

1.1 Mission Requirements Analysis

In order to ensure the completion of the primary tasks required for the competition, we prioritized the development of our solutions such that we could efficiently utilize the efforts of our team members. We focused on a multi-staged development process, which allowed us to simultaneously work on several tasks. We began our development with split efforts between the Interoperability task and the Waypoint task. This gave us the greatest amount of time to work out any design flaws with either system. We chose Interoperability and the Waypoint task as our initial development efforts solely because each of these tasks provide the basis for the completion of every other task. For example, the completion of the Interoperability task is required before we can develop the necessary software to submit identified targets to the interoperability server. In the same way that Interoperability is the base for the software development effort, the development of a flight system capable of waypoint guided autonomous flight is the base for the remaining hardware challenges. This includes the final primary task for the competition, the Search Area task, which began development immediately following the completion of the Waypoint task. From there, we focused on secondary tasks that were easier to complete based on the strengths of our individual team members and the relative ease of development based on the previous tasks we had completed.

Our system designs constantly evolved during their development as our system requirements became more refined. This required us to make many difficult decisions which ultimately both increased and limited the capabilities of our software and aircraft. One such decision is our continuing choice of aircraft for this years' effort. We decided to continue using the ReadyMadeRC Anaconda this year, because it provides a sturdy airframe with landing gear. When compared to the previous airframe we used two years prior, which had a bungee launch system, the landing gear provides a significantly easier and smoother takeoff process, allowing us to take off very quickly and easily. However, the downside is that the previous aircraft is significantly more efficient in the air and has significantly more internal storage capacity. Not having this storage space has required us to mount several systems outside of the fuselage, further reducing our efficiency. However, we determined that the flight time of the Anaconda is more than sufficient to complete all attempted tasks, and continued our development of the Anaconda system.

Another key design decision revolved around the completion of the Off Axis Target and Search Area tasks. From previous experience, we learned that having a downward facing fixed camera was ineffective at taking pictures during maneuvers, especially when we are flying a grid pattern and expect to be forced to capture images during a high bank turn. This lead to us being unable to effectively complete the Search Area task in previous years. As a result, a small team of engineer within our team designed a 3D printed gimbal specifically to be mounted within our airframe. This also had the benefit of providing the capability to complete the Off Axis Target task. However, the cost of adding this system is an extra 0.5lbs of weight added to the aircraft, in addition to the weight of the main camera. More importantly, however, is the large volume of space required to house a large gimbal in the fuselage of the Anaconda. This reduced the number of additional systems we could include in the aircraft significantly, thus reducing the number of task we could attempt. However, the completion of Off Axis Target and Area Search tasks were deemed more important than the cost in weight and space, as those tasks are worth a considerable amount of points.

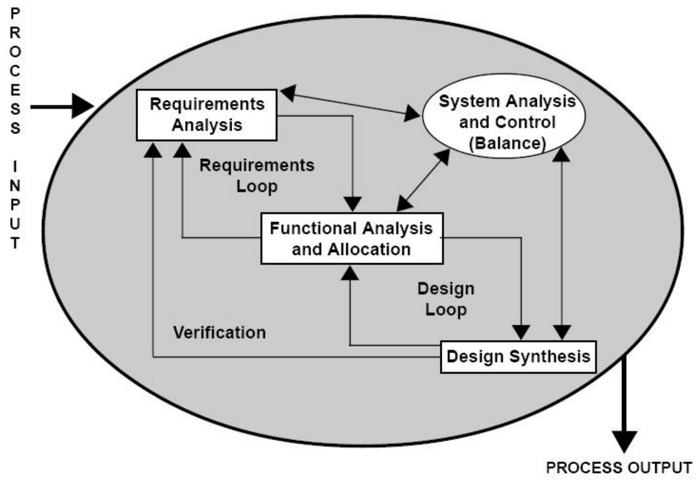

Figure 2: Our Design Analysis and Task Requirements Planning Mechanism.

1.2 Programmatic Risks and Mitigations

In approaching this year's design, the team carefully analyzed the judge's critiques from the previous year. The goal of this analysis is the improvement of areas in which the team was lacking in the previous years. The factors which posed the most significant risk to the success of the mission were issues like signal loss, hard-to-detect program logic errors, inefficient pre-flight setup,and hardware malfunctions during flight. In the previous year's competition the main issue we struggled with was the pre-flight setup. There were also some faults with hardware, as well as unexpected issues with mission components. One of these issues was the improper loading of software during the pre-flight setup which in turn resulted in a chain reaction of issues, making it impossible to capture images or properly configure the SRIC.

Upon later analysis, it was determined that this problems faced were caused by the payload computer not booting into the correct state during preflight setup as well as faulty tools that were used in the software designed for the Interoperability task. Additionally, it was later determined that an error in our Raspberry Pi's power on process caused software to boot improperly. This prevented it from connecting to the camera. The lack of camera functionality prevented the SRIC code from working properly which kept other tasks from occurring. Several steps were implemented this year in order to mitigate the damage caused to both hardware and software.

For the software portion, we developed a plan that details the need to prepare pre-flight setup procedures, in an effort to avoid mistakes. As part of the plan to prevent problems, we ran a series of tests on all of our software. These tests were intended to minimize the risk of different scripts and applications crashing in the middle of a flight. Also, in an attempt to decrease the vehicle's unpredictability, we have detailed a list of procedures that shall be carried out should the software components malfunction. The main components of the recovery procedures are identifying the error thrown, implementing a fix if possible, and restarting the application to restore communication with the plane. After heavy analysis of NodeJS and the dependencies that are designed to handle Mav Packet information, it was discovered that unreliable code that was designed for HTTP connections was disposing of 48% of the MavPackets received over UDP. It is still unknown why this happened, but the team suspected it was due to broken JSON that wasn't being processed correctly. This year the team has written it's own MavPacket handler along with a dedicated integration of the tools required to fulfill the interoperability task in a way that fulfills the mission requirements.

Another immense source of possible uncertainty is the aircraft's hardware. During flight procedures, it is common for a vehicle to become damaged. We took steps to ensure each component's structural integrity as well as the physical stability of the plane. For example, we have reinforced the airframe of the aircraft with carbon fiber rods and covered the plane in clear adhesive tape. The carbon fiber provides structural rigidity to the airframe, making the aircraft less susceptible to the effects of turbulence and more capable of absorbing the stresses of flight. The clear tape also works to distribute the stresses of flying to a larger part of the airframe, while also acting as a protective coating that prevents minor wear on the foam body of the plane. Improvements like these, and many others, ensure that we reduce the risk of hardware failures to our aircraft.

2. Systems Design

2.1 Aircraft

Airframe

The UAS is based on a heavily modified ReadyMade RC Anaconda airframe *(see Figure 3)*. The airframe features a robust carbon fiber reinforced structure with a spacious center fuselage pod, large, narrow wings featuring a thick airfoil, and carbon fiber reinforcement. The build of the airframe lets the anaconda carry heavier payloads than models previously used by the team. The twin boom design and large wings, provides several opportunities for external payload mounting.

The Anaconda features fixed landing gear capable of withstanding the stress of rough grass runways. Landing gear permits a simpler takeoff and landing procedure than the previous airframe, and is the primary reason for the team's transition to The Anaconda. The gear also allows the capability to smoothly land in rugged terrain is the primary design feature enabling consistent autonomous takeoff and landing procedures.

The wing profile provides no inherent aerodynamic self-righting ability, thus permitting easier autopilot tuning and more precise autonomous flight. The wing features a flat-bottomed design with a thick airfoil and no dihedral. Combined with an extended tail section, this airframe offers exceptional handling and maneuvering.

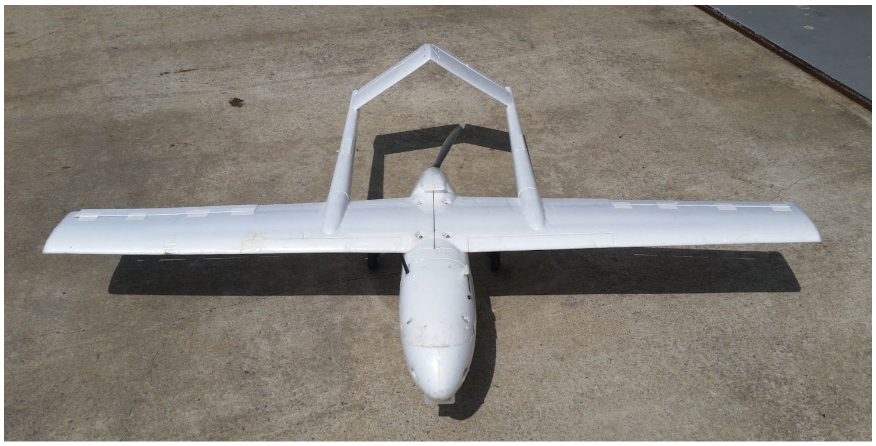
Figure 3: ReadyMade RC Anaconda Airframe

Propulsion

The aircraft is powered through a combination of two of lithium polymer (LiPo) batteries. The main flight and payload subsystem batteries are comprised of four cell LiPo batteries each of which have a nominal capacity of 4000 milliampere-hour (mAh) and nominal voltage of 14.8 volts. We used two four cell LiPo batteries positioned in the nose of the aircraft to maintain a proper center of gravity of the aircraft. The primary source of propulsion is provided by a 800kv Tiger Brushless Outrunner AT3520-5 Motor combined with a 15x6 static propeller and Tiger Motor 80A Electronic Speed Controller. Conclusive testing demonstrated that the system has power suitable for sustainable 60 degree climbs.The aircraft is able to maintain a cruising speed of 30 meters per second (m/s), or 60 Knots-Indicated Airspeed (KIAS) and a normal cruising speed of 16 m/s (31 KIAS). During normal cruise, the power system draws only 6A. This gives us a theoretical flight time of 1 hour.

2.2 Autopilot

Through conducting an alternative analysis, the team concluded that continuing to use the Pixhawk Flight Management Unit (FMU) is ideal. *(See Table 1)* The table below shows a comparison between various autopilots considered for usage in the unmanned system. The three flight control systems provide the same level of mission capability and flight functionality. Each of the three systems are priced at under five-hundred US dollars. Our team has experience with all three autopilots.

The APM 2.6 hardware is deprecated and no longer supported by Ardupilot. As such, the APM 2.6 hardware is no longer a viable option for safe flight. While the Lisa M V2.0, based on paparazzi autopilot software, is a capable autopilot, the system has several drawbacks. The paparazzi software has very little documentation, insufficient online support, and is difficult to operate. The hardware requires connectors that are hard to find and expensive. The input/output logic voltage for the hardware is 3.3v, whereas the majority of our onboard systems operate on 5v. This discrepancy creates a need for additional hardware to convert the logic voltage, adding weight and power usage.

Due to easy-to-use, configure, and operate software, coupled with open source hardware and peripheral support for many types of sensors, the Pixhawk autopilot using ArduPlane firmware was seen as the best autopilot to suit our mission requirements.

Hardware	**APM 2.6**	**Pixhawk**	**Lisa/M v2.0**
CPU	ATMega 2560 Standard Arduino	STM32F427 ARM Cortex M4	STM32F105RCT6 ARM Cortex M3
RAM	8KB - SDRAM	256KB	64KB
Flash	128KB (124KB usable)	2MB (1MB Usable)	256KB
Redundancy	None	- STM32F103 failsafe co-processor - Redundant power supply inputs - Backup mixing systems	Multiple R/C Receivers
Software	Arudpilot (Deprecated Hardware)	Arudpilot	Paparazzi
Cost	$0 (owned)	$0 (owned)	$0 (owned)

Table 1: Distinction between available autopilots.

2.3 Payload System

The key components of payload are the on-board computer, primary imaging camera, and a gimbal that holds the camera in place. The computer and camera facilitate the primary search area task in addition to the off-axis target task.

Camera System

We selected the Canon G15 camera as our main camera due to several important features. The camera is based on the DIGIC 5 processor, which has faster image saving and transferring speeds than most other Canon PowerShot processors.The camera also has a 12 megapixel image sensor. The sensor and lense configuration lets us identify features as small as ¾ inch at an altitude of 200 feet. The camera is capable of shooting continuously, 3 frames per second, or one image every 4 seconds over USB. This resolution and speed is required to fully cover the ground when flying roughly 32 knots at an altitude of 200 feet. An important decision criterion element is the camera’s compatibility with the Canon Hacker Developer Kit (CHDK). The CHDK software framework facilitates the interface between the camera and payload computer. In particular, CHDK permits operation and control of the camera through an extended Picture Transfer Protocol (PTP), used to transfer pictures from the camera to the computer. CHDK only supports point-and-shoot cameras, this was not considered disadvantageous as it corresponds to the team’s goals of maintaining a minimal payload weight.

Gimbal

To keep the imaging camera parallel to the ground and improve image quality, we designed a two-axis gimbal mount actuated by two HITEC HS-85MG micro servos. The gimbal mount went through several iterations of design. The first iteration was made of large 3D printed blocks of plastic. Later iterations improved tolerances, cut weight, fixed balance, and increased space utilization. The final design of the gimbal is shown in Figure 5 with the plane mounting block, the two brackets, the camera plate and the servos. A picture of the mounted gimbal can be found in Figure 5.

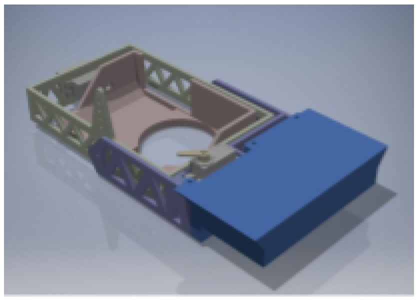
Figure 5: Camera Gimbal 3D Model

Raspberry Pi

We chose the Raspberry Pi 2 Model B to control the payload subsystem due to three factors: cost, size, and performance. The Linux-based Raspberry Pi is commercially available for thirty-five dollars, which is lower than most single board computers. Due to its high processing capability, large support base, and open source origins, the Raspberry Pi 2 Model B offers greater flexibility in usage, a higher feature count out of the box, and a greater ease of use than a majority of similar, competing system on a chip (SoC) computers. Slightly larger than a credit card, the Raspberry Pi operates a quad-core ARM Cortex-A7 CPU at 900MHz with a current draw of less than 500mA. Compared to a microcontroller, the Raspberry Pi offers more standard interfaces, including USB ports and Ethernet. Other system on a chip(SoC) computers do not offer the low price point or small size of the Raspberry Pi. Given the team's budget and size constraints, the Raspberry Pi was the best choice.

2.5 Object Detection, Classification, Localization

The interoperability task controls all of the Object detection required to run the Obstacle avoidance system, as well as the visual display for the Ground Station Control. Localization is controlled through a program used within the interoperability system in order to detect for objects that either intersect the plane on its designated path, or objects that are close by. This is a requirement set by the team. It ensures that there is real-time obstacle avoidance in the system. In the past, recursive algorithms used to situate paths around an obstacle proved to be too costly to process possible paths. This year, we will be employing a different type of algorithm involving dynamic programming, that takes past known solutions into account. In order to figure this path based on the plane's last known location, the Interoperability system, as well as the obstacle detection system, must work simultaneously. The interoperability system specializes on telemetry-based communications with other remote servers looking to process data.

2.6 Communications

Radio Communications

The controller held by the safety pilot operates via 2.4Ghz RC. With the intent of full autonomous flight, the link will not be utilized under normal circumstances except for use of failsafe settings for the aircraft in flight. The controller is capable of supporting six channels to achieve complete control of the control surfaces. Due to the safety nature of this link, it operates fully independent of other systems and links onboard.

Telemetry Communications

A 900Mhz telemetry radio provides a real time data link for transmission of MavLink flight data to and from the mission control station. The data is utilized for the interoperability task and submission to the competition server. To achieve a greater transmission rate, we subsequently decreased transmission range. To account for this, we switched to using RFD900+ telemetry modules in place of the standard use 3DR radio modules. The RFD900+ module is capable of 1.0 watt transmit power or 30dBm whereas the previous used modules are rated for 0.1 watts of power or 20dBm. With the new setup, the radio utilizes dual polarization to prevent loss of communication. Due to enhanced features, the aircraft is capable of greater transmission rates than previous years and minimal packet loss.

Payload Communications System

A data link consisting of two Ubiquiti Bullet-M5's is utilized for communication to the payload systems onboard. The use of 5.8GHz instead of 2.4GHz ensures it does not create interference with the RC safety link. The Bullet is modified to have an omnidirectional whip antenna for high gain. We are able to reduce the weight of the Bullet data link by a third. The telemetry data is also routed through this data link to provide a greater transmission rate than achievable with the 900MHz radios. This ensures quick in flight retasking.

2.7 Ground Station

The Ground Control Station (GCS) consists of all equipment not part of the aircraft. The main components include the flight control computer, task management computer, radio links, and backup systems.

Mission Control

Mission control for this competition is divided amongst several components that are used to track telemetry of the plane, but also allow it to avoid virtual obstacles, as well, as physical ones, including the flight boundaries, and making sure that the physical systems on board are within the specifications that allow the plane to fly safely. These specification include those such as battery efficiency, fuselage stability, as well as redundancy and internal component assessments, which is specified within the payload section. *(See Section 2.3)*

Flight And Guidance

The flight control computer is the main control system in the GCS. It runs essential software such as MAVProxy, APM Planner, and Interoperability. MAVProxy is a lightweight software used to manage inputs and outputs for telemetry data. The software allows the distribution of telemetry data across multiple systems. The link from the plane is fed into MAVProxy which then routes the data to appropriate destinations. (*see Figure 6)* APM Planner is a piece of open source software. Having predefined functionality allows for waypoint mission planning and monitoring. Prior to being on the flight line, a base mission is created and all future tasks are added as waypoints; points received on the flight line are added while in flight. To permit maximization of versatility on flight day, four autonomous takeoff and landing patterns are made in advance. The patterns are made for each direction on both runways.

Planning and Execution

The flight plan is split between two mission files. The first file handles takeoff, waypoint path, and search area grid; the file ends with a loiter unlimited command. The second file is modified in-flight during the search area task to include the emergent target task; the file is uploaded during the loiter unlimited command. The use of two files effectively creates a smaller file that can be uploaded more quickly while maintaining a fully autonomous flight. The custom version of the interoperability task makes use of the telemetry link from MAVProxy and posts it to the competition server directly. The software runs on the flight control computer allowing efficient routing of packets.

Ground Station Computing

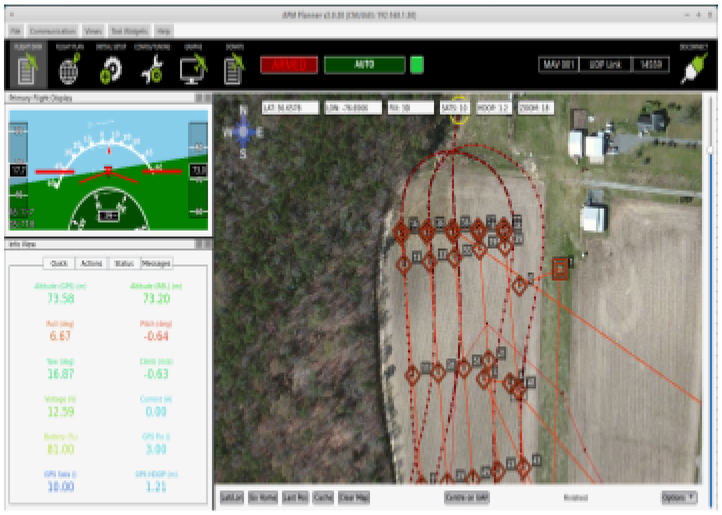
Figure 6: APM Mission Planner on display at the flying field

The interoperability display runs in a web-based interface through electron open on a second monitor to provide confirmation of the APM Planner display. The design of a web server for the web interface allows the webpage to be opened by the secondary GCS computer as well. The task management computer handles the secondary systems of the GCS. The computer directly monitors that the on-board systems are booted up properly before flight through the datalink. Image processing occurs on a task management computer containing custom software. The computer is able to download the picture in flight via the Bullet Data Link and begin processing the images before the flight is completed.

Interoperability

The Interoperability for the competition was handled in a multi-layered, and modular base of multithreaded programs working simultaneously. The software, which allowed us to complete the Interoperability task, was mostly written in Python. All data handling was done through the API provided by the competition's AUVSI's client to the competition server. This was the first year using any sort of Python library to handle interoperability. Our main application ran multiple modules concurrently. Utilizing multithreading, the application demonstrated its ability to simultaneously retrieve telemetry from the plane, and post telemetry to the competition server. This was demonstrated in both simulated testing as well as in-flight testing done on the flight line. The software uses several custom built utility applications to alert the ground station of events such as mission successes, in-flight errors, and even mission failures.

The Interoperability application is designed to handle telemetry data from the plane, as well as data coming to and from the competition server. The application also handles the photos that the plane takes, by running them through an Artificial Intelligence Engine, Nvidia Digits. The program sorts photos, and posts pictures relevant to competition tasks.

In testing, the software demonstrated its capabilities to post telemetry data to the competition server at a minimum of 10 Hz. Tests were completed to provide concurrency of the plane's position to the judges, as well as to ensure that the plane doesn't leave the flight area. Initial tests were done through simulation software operating MAVProxy. MAVProxy is a proxy server for Mavlink Packets that routes those packets through a UDP server to our software for data handling. To reference the diagram in Figure 7, all requests were done either through the mav, mission or main modules. These are the interfaces that allow us to communicate with outside servers and make data comprehensible to the system for processing.

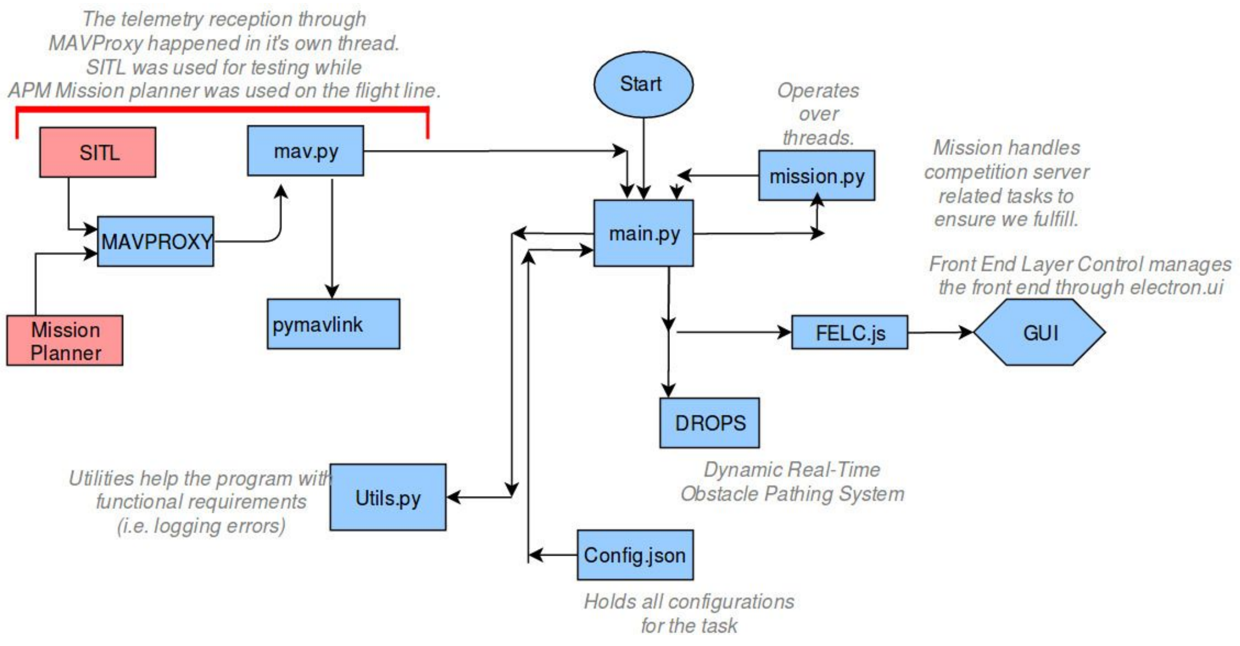
The telemetry reception through MAVProxy happened in it's own thread. SITL was used for testing while APM Mission planner was used on the flight line.
SITL
mav.py
Start
Operates over threads.
mission.py
Mission handles competition server related tasks to ensure we fulfill.
MAVPROXY
main.py
Front End Layer Control manages the front end through electron.ui
Mission Planner
pymavlink
FELC.js
GUI
DROPS
Dynamic Real-Time Obstacle Pathing System
Utilities help the program with functional requirements (i.e. logging errors)
Utils.py
Config.json
Holds all configurations for the task

Figure 7: Interoperability mission specifications and components.

Image Processing

The plane, as described in the payload systems specification, contains a camera that is used to take pictures. For the competition, we use the Canon G15 camera to identify off-axis targets, as well as various other targets specified within the rules. In order to do this we have attempted a multi staged system that designed to ensure accuracy and completeness of image recognition.

The first stage is an experimental system using Nvidia Digits. Digits is software for a neural network using automated image processing with an artificial learning engine. With Digits the images are analyzed which provides a certainty level for the characteristics of the target. A list of cropped images are populated by the results generated from running the program. The characteristics might have corrections made as we expect minor inconsistencies with this method.

The second stage is manual recognition. While Digits computes, the images are manually sifted through to find targets. Custom software known as Image Manipulation Genie (IMG) provides an interface for cropping images quickly with defined keyboard and mouse shortcuts. The software then may be used to enter characteristics of cropped targets. Only targets not recognized by Digits have characteristics inputted. At the end of this stage, all of the targets captured should be ready.

The third stage is validation. The user selects the targets to be sent. IMG collects the data for each image and performs a validation check to ensure the data is complete. If the data entered is incomplete, the software displays a warning message. In the event of failure, the user has the ability to override the warning message. An easily readable view of the images is presented to the user as the final barrier for submission. data is transferred through the interoperability system following the submission.

2.8 Cyber Security

To complete a thorough analysis of the cyber security specifications involved with this year's competition, we modeled potential threats that are present at the competition. The new regulation stating that , we can broadcast telemetry over any frequency that we want within a specified range poses several security implications. For example, any team that intending on using a specific frequency is capable of using that frequency to connect to other planes that are flying. This newly presented risk is an example of an accidental anomaly. A intentional anomaly would occur if, a team were to intentionally use a specific frequency another team's plane is on,they would be able to control the other team's plane by passing new waypoints, and other instructions to it. The anomalies might prove fatal to mission success, and the structural integrity of the plane's ability to fly.

The software component of the analysis relies on an extensive understanding of the program. In the case of our UAS, the interface with remote stations is accomplished through the interoperability program. Due to communication with the competition server as well as other components that are needed in the entirety of the flight mission, the interoperability program is essential. Lack of proper security measures might result in impersonation by another team to make it appear as though we are logged into the competition server when, in actuality,we are not. To establish a more secure mission, a login and password has been created .

3. Test and Evaluation Plan

3.1 Developmental Testing

Flight Task Performance

Flight systems testing began while other subsystems were still in development. As the flight systems were enhanced, testing was conducted to ensure the tuning would meet the mission requirements.

The primary requirements for the flight system are capturing waypoints within 50ft and maintaining stable level flight for the imagery task. Based on issues during the competition last year, the team defined an extra specification that the UAS needs to be able to complete a U-turn within 25m (80ft) to successfully complete the search area task without crossing the “no-fly” boundary. The flight team set secondary goals for autonomous take-off and landing.

During flight tests with appropriately planned missions, the flight vehicle was able to maintain 50ft accuracy on waypoints. After some tuning, the payload team determined the flight was stable enough for the imagery task. In circle mode, the UAS was not able to maintain the 25m radius desired for the U-turn specification. However, flight tests with well planned missions turning into the wind could meet the necessary requirements. Overall, the flight systems testing resulted in a new focus on mission planning within the capabilities of the system to ensure mission success.

3.2 Individual Component Testing

Payload System Performance

The payload team employed a three-level testing scheme: the first level was the unit level test, the second level was the unit-level integration test, and the third level was a full systems test. All three levels of testing were conducted on the payload system. The first testing step can be a unit test of the payload code or individual tests of the networking systems. Unit level testing was applied to the SRIC and imaging programs. Unit tests assisted with the development of the payload system and prevented regression errors. With individual tests, we were able to ensure that the programs were producing the correct output based on the input we applied. The second step was a unit-level integration test. An integration test checks the interoperability of unit-level systems with other subsystems. The integration tests ensure the system will perform properly prior to attempting mock missions. The final testing step was a full systems test. The system was placed in a mock mission environment and expected to perform as it would at the competition. Mock missions confirmed proper operation of the tasks and disclosed a few problems that were subsequently fixed.

Interoperability Performance

Our team completed the interoperability task by successfully developing the Python On Computers for Enhanced Management Over Networks (POKEMON). POKEMON is a system capable of relaying data between any other two systems. We used simulation tools, such as Software In The Loop (SITL), to replicate telemetry data that would be received by our system during a real flight. POKEMON exposed all of the telemetry it received through a web front-end using Electron as a Desktop Container, which successfully overlaid the plane's location, obstacles, and waypoints on an interactive map of the current area.

To test stationary and moving obstacle displays, we created a sample mission on the AUVSI SUAS server using several coordinates that aligned with waypoint locations on the current test site. Our team used feedback and data collected throughout the last competition to increase the reliability, performance, and overhead of POKEMON. Added redundancy and edge-case handling are features introduced to POKEMON after previous flight-time experience that the team found paramount to success.

Target Recognition Performance

The Nvidia Digits neural network, was originally tested through the use of computer generated images to make sure the program was able to reliably detect contours and large differences in colour. Once we increased the network's reliability, we started testing with real images. Model targets were created and photographed so we could have realistic test images. The network was used to identify targets with a given certainty. The results were then manually reviewed.

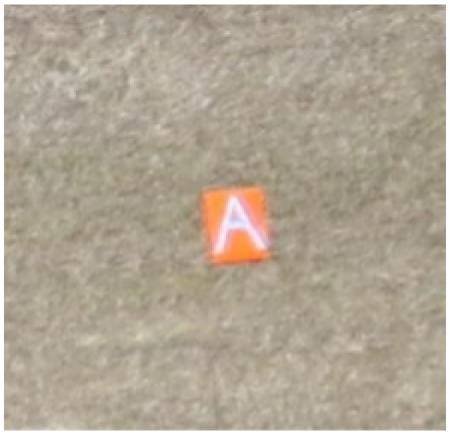

Figure 8: cropped target image

Image Manipulation Genie (IMG) was tested separately as it needed to be tested based on data transfer and ease of use. We tested IMG by outputting data to a text file and assuring it conformed to standards outlined in the appendix of the competition rules. Upon confirming IMG was outputting the text file correctly, we created a testing interoperability server and tested IMG's ability to output over the interoperability network.

3.3 Mission Testing Plan

Mission Tests

Multiple tests have been conducted to test our system's ability to accomplish all mission goals set in place. These tests consisted of possible challenges we would face at competition to ensure that our systems would function as expected when the time came to compete. These tests were also designed to ensure that we operated with efficiency and safety. These mock missions confirmed our system's ability to operate effectively and displayed some shortcomings that we had previously missed.

Predicted Results

The team chose to attempt the tasks listed in the expected task performance list *(See Table 2).*

Task	**Threshold**	**Objective**	**Expected Performance**
Autonomous Flight	- Controlled Takeoff - Controlled Landing - Autonomous Flight - Capture Waypoint	- Autonomous Takeoff - Autonomous Flight - Autonomous Landing - Capture Waypoint	Fully achieve
Search Area	- Localization - within 150ft - Classification - provide 2 characteristics - Classification - Detection - Imagery - n/a - Autonomous Search - n/a - Secret Message - n/a	- within 75ft - provide all 5 characteristics - Decode Message - provide cropped image. $\geq$ 25% - in auto during search - Decipher message anagram	Successful search Achieve manual recognition of targets
Actionable Intelligence	Provide a target location within 150 ft and 3 characteristics within the same flight	-Provide a target location within 75ft and 5 characteristics within the same flight	Download image in flight successfully
Emergent Target	- In-flight re-tasking - n/a - Autonomous search - n/a - Target - provide image	- add target position as waypoint - autopilot control during search - provide image and location within 75ft and description	Capture image of specific point in high detail
Interoperability	- Download & Display Server Info and - Time at 1 Hz -Download and Display Obstacles at 1Hz -Upload target details - n/a	-Download and display at 10 Hz -Download and display at 10 Hz -Upload all submitted targets and their details	Publish information at 10 Hz Post target information through system
Off-Axis Target	- Imagery - n/a - Classification - Provide 2 characteristics - Payload autonomy - n/a	- Provide an image of target - Provide 5 characteristics - Automatic tracking of target	Take picture at given side angle

Table 2: Mission Tasks and planned objective fulfilments

4. Safety, Risks, and Mitigations

4.1 Developmental Risks and Mitigations

When creating our system for the competition, we made specific choices with safety in mind to ensure that no one was harmed during the development process. We reinforced our plane's wings with steel and carbon fiber rods to ensure that gusts of wind would not damage the integrity of our plane, causing us to lose control over a large aircraft in an area occupied by our team members and others. All of our software that we wrote was tested in simulations run in SITL to make sure that our own code would not jeopardize our system mid flight, possibly causing damage to anyone beneath it.

4.2 Mission Risks and Mitigations

Signal loss and pre-flight setup are some of the largests risks to mission completion and mission success. If we lose signal or if our pre-flight set up is not correct, then we could face mission failure. With signal loss, if we lose connection to our system, we can no longer monitor our autopilot to ensure that it is still following its predetermined route. To minimize risks associated with connection loss we have the ability to take over the plane during flight and either land it where possible or bring it back in range to regain connection. To minimize risks associated with lack of a complete and correct pre-flight set up, we have created a series of checklists to ensure that everything is functioning how it should and providing us with the data it should be.

4.3 Operational Risks and Mitigations

The propeller on the plane becomes dangerous as soon as the plane is powered on, so in order to prevent any accidents related to the propellers, the pilot is the only one allowed in front of the propellers once the batteries in the plane are plugged in. Because the plane itself can be a danger during takeoff and landing, everyone on the landing strip needs to be alert and monitoring the takeoff process so that if something were to go wrong, everyone would be as ready to react as possible. The batteries used inside of the plane are Lithium-Polymer (LiPo) batteries, which provides us with a powerful and reliable source of energy for the plane, but also puts us at risk for fires caused by the batteries. Due to the danger presented, we transport the batteries in LiPo safe bags and store them in a fireproof container when not in use to minimize the possibility of damage.