2016 AUVSI SUAS Competition Journal Paper

Christopher Newport University / IMPRINT

Unmanned Aerial Systems Team

May 2016

Abstract

Team CNU/IMPRINT from Christopher Newport University in Newport News, Virginia is proud to present the team's entry to the 2016 AUVSI SUAS competition. The modified RMRC Anaconda was developed by employing a true systems engineering approach. The team analyzed the requirements of the primary and secondary missions to derive a set of tasks. During the design phase, tasks were translated into technical specifications while subsystems were researched and tested. The system was developed by incremental integration of components, testing flight capability at each stage. To increase efficiency, the team chose to use commercial hardware, adding additional components as needed. Software is based on open source software modified to fit our tasks. The airframe is based on the Ready Made RC Anaconda Expanded PolyOlefin Double Boom Delta Tail aircraft, with a wingspan of 81 inches. The system includes: the 3DRobotics Pixhawk board, a 2.4 GHz RC system, a 915 MHz telemetry radio, a Raspberry Pi SoC computer, a Canon G15 camera, and LiPo batteries. The ground station uses laptops, and a 5.8 GHz Ubiquiti Bullet M5. The UAS demonstrated a flight time of more than 30 minutes at a cruise speed of 31 knots, with a gross takeoff weight of 10 lbs in more than 30 flight tests. This document outlines the development process, final systems design, mission execution factors and safety factors were considered by the team.

Contents

1 Systems Engineering and Design

1.1 Team Organization

The UAS team at CNU is divided into four sub-teams, each assigned to one major subsystem: Flight, Payload, Ground Control Station (GCS), and Software. The subteams were created to focus on various tasks. To insure organization, everyone has a job to do on the team; the tasks are independent of each other . For the purpose of efficiency, teams are spread so a variety of tasks can be completed in a timely manner. The formation of groups permits successful completion of tasks within the given time frame.

1.2 Planned Tasks

Primary and secondary mission tasks provided by the AUVSI Student Unmanned Aerial Systems (SUAS) competition is extensive. To make the mission tasks easily completable within the given time frame, we selected specific tasks for our team to accomplish. The team chose to attempt the tasks listed in the *Expected Task Performance List*, Table 1.2.

Task	**Threshold**	**Objective**	**Expected Performance**
Autonomous Flight	- Controlled Takeoff - Controlled Landing - Autonomous Flight - Capture Waypoint	- Autonomous Takeoff - Autonomous Flight - Autonomous Landing - Capture Waypoint	Objective - Tested Successfully Objective - Tested Successfully Objective - Tested Successfully Objective - Tested Successfully
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	Objective - Testing in Progress Objective - Tested Successfully Threshold - Tested Successfully Objective - Tested Successfully Objective - Tested Successfully Objective - Testing in Progress
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	Threshold - Tested 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	Objective - Tested Successfully Objective - Tested Successfully Objective - Tested Successfully
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	Objective - Tested Successfully Objective - Tested Successfully Objective - Tested Successfully
SRIC	Download - n/a Upload n/a Autonomous SRIC - n/a	path: /team/X/download.txt path: /team/X/upload.txt Automatically detect SRIC and	Objective - Tested Successfully Objective - Tested Successfully Objective - Testing in Progress

		download/upload autonomously	
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	Objective - Tested Successfully Threshold- Tested Successfully Objective - Tested Successfully

Table 1.2: *Expected Task Performance Descriptions*

1.3 Design Rationale

1.3.1 Aircraft

The UAS is based on a heavily modified ReadyMade RC Anaconda airframe (see figure 1.3.1 below). The airframe features a robust carbon fiber reinforced structure with a spacious center fuselage pod. A spacious center fuselage, large, narrow wings featuring a thick airfoil, and carbon fiber reinforcement let the anaconda carry heavier payloads than previous airframes. With a twin boom design and large wings, the Anaconda provides many opportunities for external payload mounting.

The Anaconda features fixed landing gear capable of withstanding the stress of rough grass runways. The landing gear provides a simpler, easier takeoff and landing procedure than the previous airframe, and is one of the main focal points of transitioning to the Anaconda. Landing gear is also the main enabling design feature that provides the necessary tools for consistent autonomous takeoff and landing procedures.

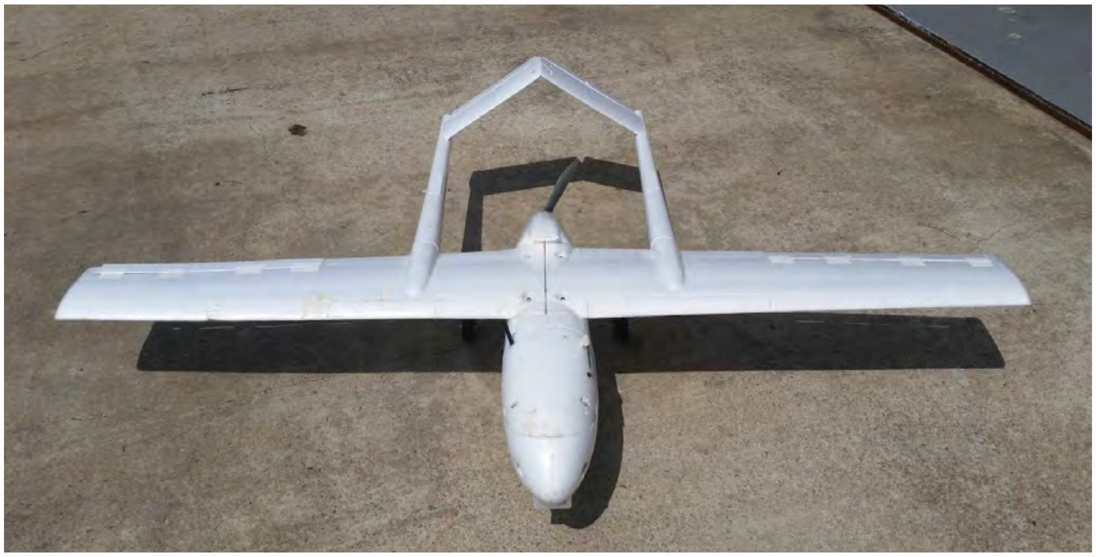


Figure 1.3.1: RC Anaconda Airframe

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

1.3.2 Flight Control

Through conducting an alternative analysis, the team concluded that continuing to use the pixhawk Flight Management Unit (FMU) is ideal. Table 1.3.2 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. Additionally, each of the three systems are priced at under five-hundred US dollars. Our team has experience with all three autopilots.

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 1.3.2: *Autopilot hardware comparison*

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. First, the paparazzi software has very little documentation and online support, and is difficult to operate. Second, the hardware requires connectors that are both hard to find and expensive. Finally, 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.

With 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 stood out as the best autopilot to suit our mission requirements.

1.3.3 Payload

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.

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.

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.

1.3.4 SRIC Task

We chose to implement a python program to download a file from a network on the airfield to the aircraft. From the aircraft, the file is sent to the ground control station. A script on the payload computer automatically grabs the file from the SRIC server and saves the file to the payload computer. From there, FTP is used to download to the ground station. MAVLink FTP was used as the specific FTP implementation because it runs over the much faster telemetry links.

An implementation of the MAVLink FTP server exists within the PX4 flight stack, and the code was ported to run as a MAVProxy module. An existing MAVLink FTP client implementation in QGroundControl was used to receive the files.

1.3.5 Interoperability

Our team chose to use the JavaScript Node.js framework to implement the interoperability task. We built a javascript back-end application that receives, interprets, and relays data from the AUVSI SUAS competition server to the ground station. The Node.js server receives telemetry from the unmanned aerial vehicle at an average rate of 10Hz, translating each packet from MAVLink to JavaScript Object Notation (JSON), and posting it using HTTP requests to the AUVSI SUAS server. Data sent to the competition server includes the vehicle's latitude, longitude, altitude above mean sea level (in feet), and the aircraft's heading. All requests between the Node.js server and the AUVSI SUAS server are done through HTTP requests. Communication between the unmanned aerial vehicle, the ground station, and the Node.js server is done using MAVLink on top of the User Datagram Protocol (UDP). The server makes use of JavaScript's event-driven, concurrent, non-blocking structure. This allows the Node.js application to send and receive only the data that changes, increasing efficiency and reducing overhead.. Any data sent and received by the AUVSI SUAS competition server is already in a format native to JavaScript, making it the ideal choice as the programming language for the interoperability task. Our team used the Node.js Javascript framework as it provides robust networking tools and libraries, and fast implementation of non-blocking, event-driven applications. Our team saw Node.js as an efficient and logical choice for the interoperability task.

1.4 Programmatic Risks and Mitigation Methods

Signal loss, hard-to-detect program logic errors, efficient pre-flight setup,and hardware malfunctions during flight, provided the most risk to mission success. In last year's competition we struggled mainly with pre-flight setup. There were some faults with hardware and a few issues we didn't expect. Some software loaded improperly during pre-flight setup last year and caused us to fly without image capture or SRIC set up properly. After the flight last year, it was determined that the problem occurred during preflight set-up and was caused by an error from a payload computer not booting properly into the correct state. The fault in our Raspberry Pi caused software to boot improperly and not connect to the camera. The lack of camera caused the SRIC code to hang and prevent both tasks from occurring.

This year we took several steps to mitigate risks involving both hardware and software utilized on the plane during flight. For the software portion, this year we developed a plan that details the need to prepare pre-flight setup procedures, in an effort to avoid mishaps. In previous years, one of the mishaps included peripherals to the flight computer failing to initialize properly. We have also ran a series of tests on all of our software. The purpose of the software tests was to mitigate the risk of different scripts and applications crashing in the middle of a flight. We have also detailed a list of procedures that shall be carried out in the event of software components malfunctioning. The main components in the recovery procedures are identifying the error thrown, implementing a fix if possible, and restarting the application to restore communication with the plane.

To minimize risks associated with hardware damages, we took steps to ensure the structural integrity of the hardware on the plane. As mentioned in section 2.4.3, the camera's gimbal mount allows for structural integrity. In the event of sudden movement on the plane, the gimbal will keep the camera stable for pictures yielding the accomplishment of the search area task.

Other mitigation methods were taken into consideration as well. The payload sub-team looked into methods of remote fixes on the plane in the event that the remote software should crash unexpectedly. These remote fix methods were not pursued however, due to hardware limitations and the risk that remote connections would impair existing software, already functioning normally. Our team hopes to achieve this in future missions.

2 System Design Description

2.1 Aircraft

2.1.1 Airframe

The airframe is a Ready Made RC Anaconda Expanded PolyOlefin Double Boom Delta Tail aircraft. The Anaconda provides a wingspan of 81 inches, a thick aerofoil, large payload capacity, and a rugged undercarriage suitable for operating from rough landing strips. Fully loaded, the plane weighs 11 pounds, with performance overhead for 4 additional pounds of payload.

The basic airframe has been modified extensively to meet payload and flight performance requirements. The lower rear section of the main fuselage was cut open to provide a designated slot for the modular camera system described in section 2.4.2.(Figure 2.1.1.1)

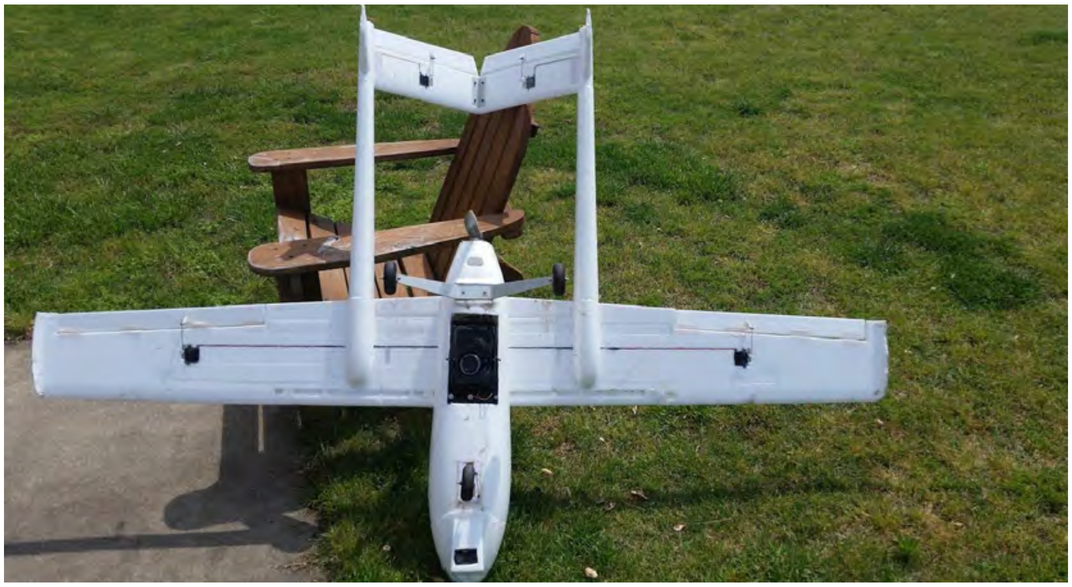


Figure 2.1.1.1: Camera Gimbal Slot (Center)

To maximize the internal payload space, an autopilot mounting tray was placed inside unused space within the battery hatch (Figure 2.1.1.2). The tray is held in the battery hatch by six screws that bolt into spacers. To ensure consistent alignment of the battery hatch, and the proper alignment of the autopilot in reference to the plane, two ⅝ inch bolts are mounted into the sides of the battery hatch. The bolts align with two stock aluminum right angle mounting brackets on the airframe, ensuring the battery hatch can only be secured in one orientation.

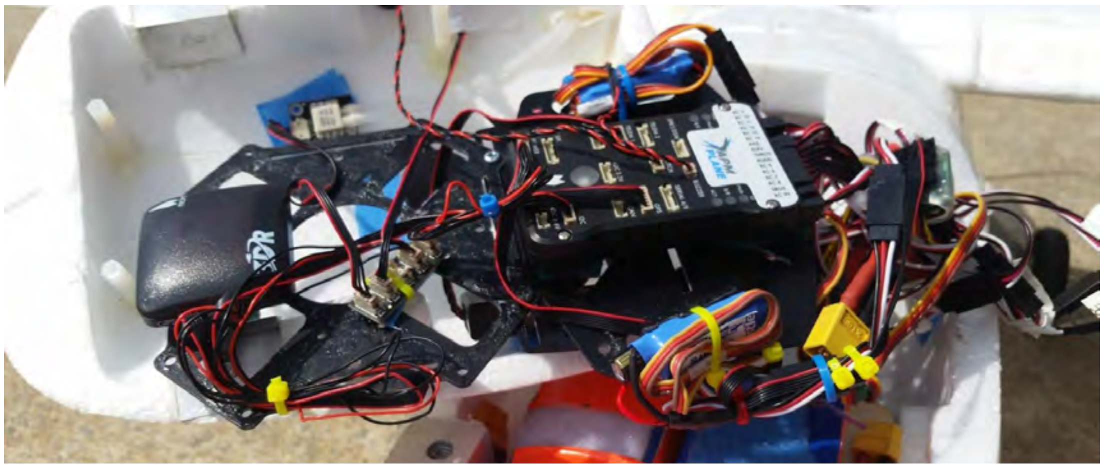


Figure 2.1.1.2: Autopilot mounting tray and battery hatch

The stock nose landing gear on the Anaconda was not capable of supporting the large amount of forward weight within the main fuselage pod. In order to support the weight of the fuselage, we designed and created a forge to carbonize and temper the main nose gear through the use of oil quenching and temperature specific heating. The resulting landing gear is harder, stiffer, and more resilient than the starting metal.

2.1.2 Power and 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.

2.2 Flight Control and Autonomy

Manual flight control is achieved through a 2.4 GHz RC radio link. The manual flight controls are passed into a 3DRobotics Pixhawk advanced FMU system, which provides the plane with Autonomous flight, including takeoff, landing, and waypoint navigation, in addition to the option for in-flight re-tasking.

The Pixhawk is configured with a barometric altimeter, GPS, magnetometer, dual inertial measurement units (IMU), differential pressure airspeed sensor, safety buzzer, and safety switch. Dual IMUs provide redundant measurements of linear acceleration, angular acceleration and reduces errors due to vibrations. The barometric altimeter provides the aircraft's altitude above ground level (AGL). The GPS provides 3-D position and velocity vector to the autopilot. The compass/magnetometer provides heading, and the differential pressure sensor provides airspeed. A voltage and current sensor provide battery monitoring to ensure safe operation of the LiPo batteries. The team tuned the system to optimize the flight parameters of the autopilot. In order to optimize the flight parameters, we had to fly the vehicle between extremes of roll and pitch, as defined within the autopilot, and adjusting the control gains until the autopilot exhibited the desired flight characteristic. We repeated the process utilizing ascents, descents, and repeated turns, to tune both the total energy control system (TECS) and navigation system to the specific flight envelope of the airframe.

We determined that it was necessary to replace the analog airspeed sensor with a digital sensor, due to the intense electromagnetic interference from the main flight power system. Reducing data inconsistency due to interference provided more stable and accurate airspeed data, improving flight performance of the plane in autonomous flight modes. We also added a Lidar-Lite V2 pulsed light range finder to the autopilot system, which uses pulsed light to find the distance between the aircraft and the ground below. Use of the Lidar-Lite V2 provides accurate altitude data below 40.0m, and allows for precision autonomous landings.

2.3 Datalink

The datalink consists of two Bullet-M5s from Ubiquiti Networks. We chose the 5.8GHz frequency range since 2.4GHz would conflict with the manual RC control. The airframe contains a single Bullet with an omnidirectional whip antenna specifically selected for its durability, and high gain. One drawback of the bullet is the large, heavy N-type connector. The team overcame the weight issue by using a blowtorch to remove the N-type connector and replace it with a small rp-sma connector. The ground station utilizes a Bullet, with a omnidirectional high-gain whip antenna.

2.4 Payload

2.4.1 Payload Computer

A Raspberry Pi 2 Model B single-board computer (SBC) acts as the inflight payload controller. The Raspberry Pi runs a combination of Python and BASH scripts to control the payload. A custom Python script, utilizing CHDK and MAVProxy, triggers the camera when the autopilot reaches a camera waypoint. The current GPS coordinates and attitude data are stored into a file for use in processing the images. The photos are saved onto the Raspberry Pi for subsequent movement to the ground station over a datalink by file transfer protocol (FTP).

2.4.2 Imaging Camera

A Canon G15 camera provides primary imaging for the search area task. The camera is triggered from a python script, which runs on the Raspberry Pi, via a CHDK extension of the photo transfer protocol (PTP), and utilizes the mavproxy interface to detect new waypoints that are received. The camera can also be manually triggered from the ground station. In the event of a payload computer failure, the camera will execute a script which runs the camera in intervalometer mode. In intervalometer mode, the camera automatically takes pictures at a predetermined interval. The intervalometer does not allow control or recording of where the plane takes pictures, but does provide imagery. Even in component failure, the imagery system can still perform the main task. In order to protect the imaging camera on takeoff and landing, the script does not open the camera's lense until the camera reaches a predetermined minimum altitude above ground level and closes the lens below the altitude.

2.4.3 Two-Axis Gimbal Mount

To keep the imaging camera parallel to the ground and improve image quality, we included a gimbal mount actuated by two HITEC HS-85MG micro servos. The gimbal mount went through 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 2.4.3 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 2.1.1.1.

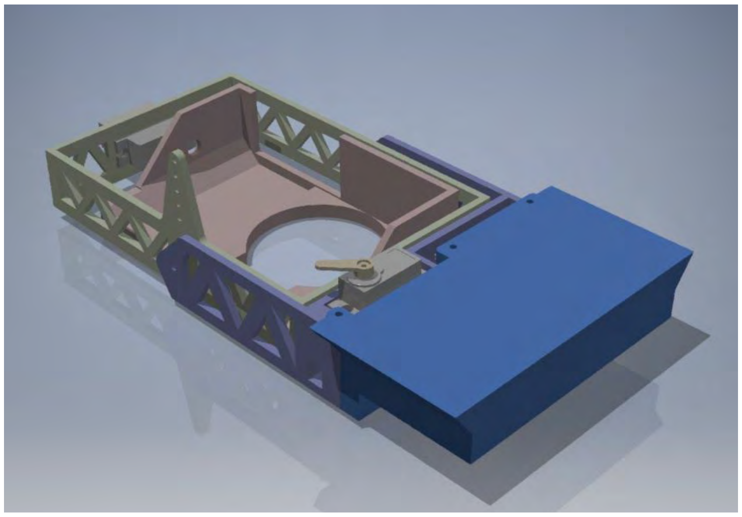


Figure 2.4.3: Camera Gimbal 3D Model

2.5 Ground Control 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.

2.5.1 Data Links

The primary radio link is a 900 MHz omnidirectional serial link. The link provides monitoring and control of the aircraft with the capability of in-flight re-tasking and is directly connected to the primary mission task computer. The links are detailed in section 2.3.

The backup flight control link is through a 2.4 GHz frequency-hopping spread spectrum (FHSS) RC held by the safety pilot in case of the need for manual flight.

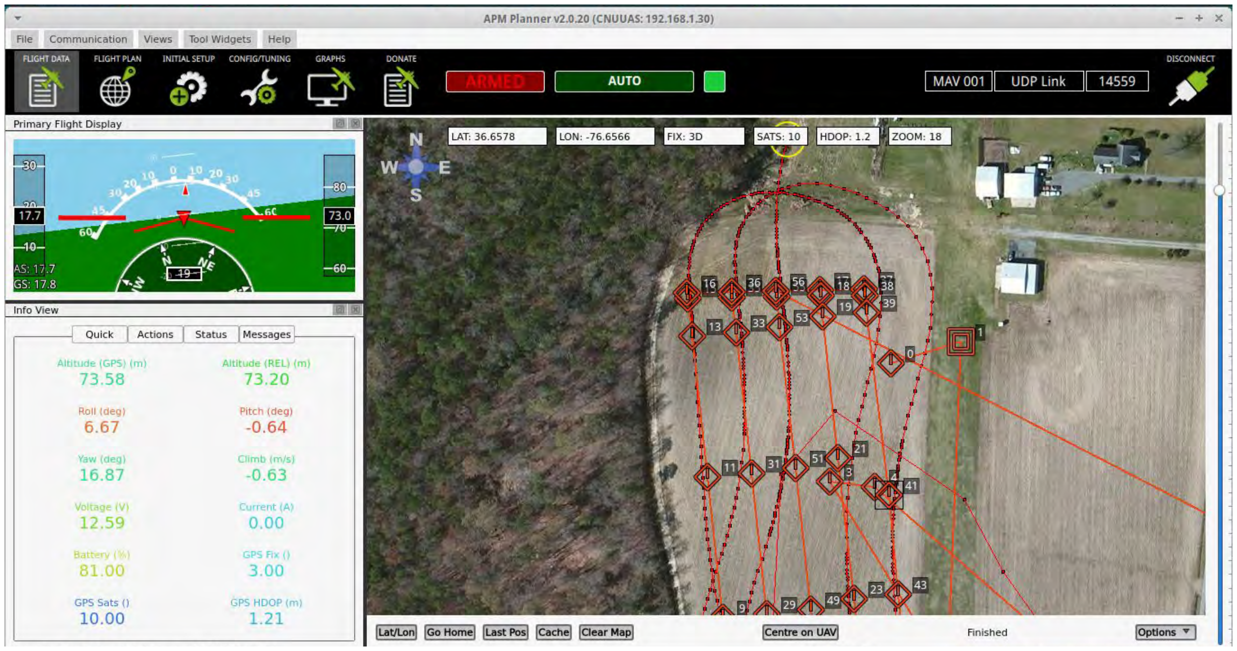
The GCS contains a high powered 5.8 GHz Bullet data link for communication with an identical data link on the aircraft. The link is utilized by the secondary GCS computer to complete tasks through controlling the onboard payload computer.

The GCS is equipped with a DHCP router. The switch allows simplified communication between the computers and the Bullet data link.

2.5.2 Flight Control Computer

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 2.5.2.



AS: 17.7
GS: 17.8

LAT: 36.6578 LON: -76.6566 FIX: 3D SATS: 10 HDOP: 1.2 ZOOM: 18

Altitude (GPS) (m) 73.58	Altitude (REL) (m) 73.20
Roll (deg) 6.67	Pitch (deg) -0.64
Yaw (deg) 16.87	Climb (m/s) -0.63
Voltage (V) 12.59	Current (A) 0.00
Battery (%) 81.00	GPS Fix () 3.00
GPS Sats () 10.00	GPS HDOP (m) 1.21

Figure 2.5.2: APM Planner

APM Planner, open source software, has predefined functionality to allow 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.

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. The interoperability display runs in a web-based interface 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.

2.5.3 Task Management Computer

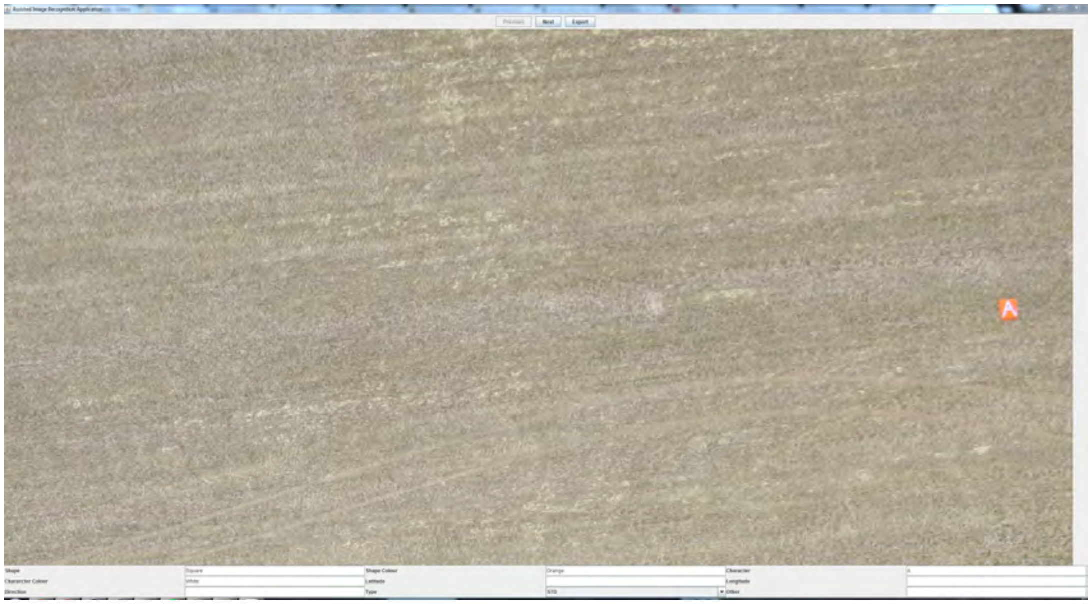
The task management computer handles the secondary systems of the GCS. The computer directly monitors that the on-board systems are booted up properly before flight through the datalink. This computer also utilizes the datalink to accomplish the SRIC task and the actionable intelligence task.

Image processing occurs on a task management computer containing custom software as defined in section 2.6. 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.

2.6 Image Processing

Image recognition is split between two programs; an image ranking script known as Priority List Image Recognition (PLIR), and an interface for manual target classification and submission called Assisted Image Recognition Application (AIRA).

PLIR is a Python program that uses the OpenCV library to detect which images are likely to contain targets. This is achieved by comparing the area of the potential shapes to a predefined configurable range along with weighting the images based on number of sides. The outputted priority list arranges the image names within a text file according to those that are most likely to contain targets. PLIR sorts pictures based on contours and color differentials that the algorithm is able to find.



Shape	Square	Shape Colour	Orange	Character	A
Character Colour	White	Latitude		Longitude	
Direction		Type	STD	Other	

Figure 2.6: AIRA interface

Once PLIR has sorted the images we execute the AIRA, shown in Figure 2.6. AIRA is a Java program that assists the operator with the identification of targets and management of the data for submission of target data through the interoperability network. In order to ensure reliability, AIRA also outputs target data into a text file that follows the format as defined in "Appendix E" of the competition rules. AIRA allows simple image manipulation such as zooming in on areas of the picture. AIRA also provides a series of fields to enter target data in for simple output to the interoperability network or text file.

2.7 Interoperability

Interoperability software between the unmanned aerial vehicle and AUVSI SUAS competition server implements a maintainable, modular design. The Java Application Manager (_JAM) handles communication between any tasks requiring access to the AUVSI SUAS server and the ground station. _JAM exposes any telemetry it receives from the vehicle through a web interface which can be accessed by any device connected to the same network. The web interface displays an OpenLayers map with real-time location of all obstacles, waypoints, boundaries, and plane location. The interface receives and updates all of its information as soon as new vehicle telemetry is received from the ground station. It communicates with the back-end portion of _JAM through the WebSocket JavaScript API.

_JAM converts all telemetry received by the vehicle into JavaScript Object Notation before posting it to the AUVSI SUAS server. Due to _JAM's modularity, any external system or software program can relay its information to any other system. Our team found a use-case for relaying information to other systems, by using the existing connection between _JAM and the AUVSI SUAS server to relay target data to the AUVSI SUAS server.

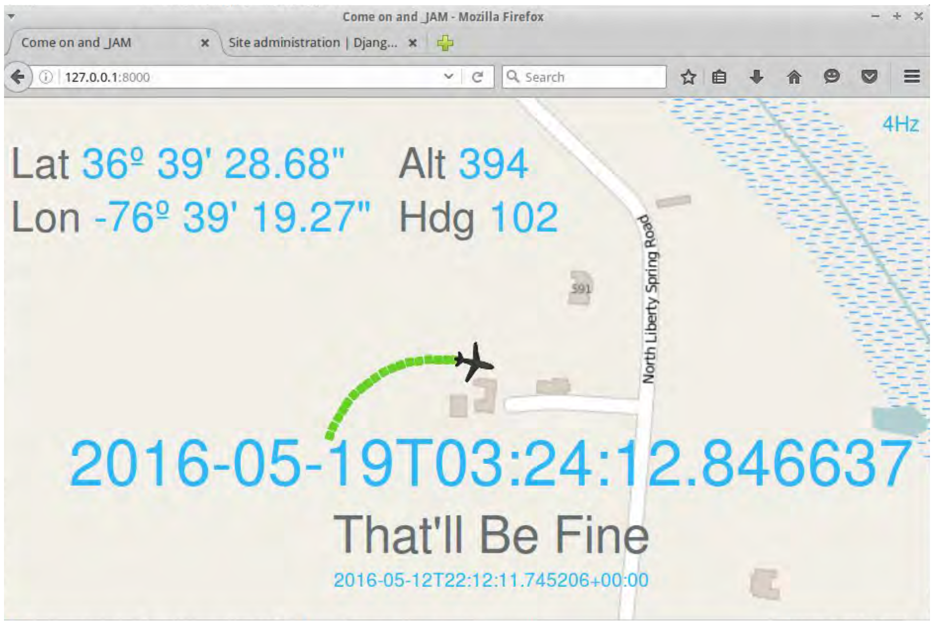

Come on and _JAM - Mozilla Firefox
Come on and _JAM
Site administration | Djang...
127.0.0.1:8000
Search
4Hz
North Liberty Spring Road
591
Lat 36º 39' 28.68" Alt 394
Lon -76º 39' 19.27" Hdg 102
2016-05-19T03:24:12.846637
That'll Be Fine
2016-05-12T22:12:11.745206+00:00

Figure 2.7: _JAM interface

3 Mission Planning and Operation

Mission planning assures team readiness, maximum efficiency, and contributes to safety involved in operations. Planning for missions occurs before takeoff. It is during the planning phase that we evaluate our safety procedures, and ensure that the plane is operating well before flight commences. Preflight procedures primarily occur to evaluate the condition of the plane's physical hardware and software readiness prior to takeoff. The preflight procedures also ensure that every member of the team is ready for the possibility of dynamic airborne retasking mid-flight.

Prior to each mission, the team conducts a pre-mission brief; the brief occurs in the pit area before the team approaches the flight line. During the brief, it is ensured that every team member comprehends the mission and their role. The mission commander reviews the primary and secondary tasks that the team will attempt. The commander will identify specific concerns with tasks and approve preliminary flight plans. Upon arriving at the flight line, the ground station manager will initiate the rest of the mission planning. The received mission data is incorporated into the mission and the mission commander confirms auto takeoff and landing patterns, adjustments are made as necessary. The ground station manager ensures that the mission path is correct.

Following a successful takeoff, the autopilot follows a predetermined flight path. Payload and imaging crews begin their analysis of images taken by the plane mid-flight. The plane needs to be continuously monitored, so the mission cannot require the aircraft to leave the line-of-sight. The flight plan can be modified in-flight to account for variances in safety, weather, and information.

At the end of a flight, a designated team member visually inspects the landing zone to determine if it is suitable for the aircraft to land. Upon successful inspection, the aircraft begins its approach. The aircraft uses a calculated fixed gliding rate to safely touch down. After landing, the mission commander reviews the mission objectives while designated team members retrieve the aircraft and clean up the flight line. Payload and imaging crews continue their analysis of the imagery data if necessary. Following completion of the mission, all team members are encouraged to offer feedback during the mission debrief. The mission commander ensures the mission is correctly logged and that necessary system repairs are noted.

4 Testing and Evaluation

4.1 Flight Task Performance

Flight systems testing began while other subsystems were still in development. Flight systems testing happened alongside development. As the flight systems were tuned, 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 also defined that the UAS need to be able to complete a U-turn with a 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 autonomous 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 necessary to complete a U-turn, however flight tests with well planned missions turning into the wind could meet the necessary requirements. Overall, the flight systems testing echoed a new focus on mission planning within the capabilities of the system to ensure mission success.

4.2 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 unit or 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 patched.

4.3 Interoperability Performance

Our team completed the interoperability task by successfully developing the JavaScript Application Manager (_JAM). _JAM is a system capable of relaying data between any other two systems. We used simulation tools, such as Software In The Loop, to replicate telemetry as would be received by our system during a real flight. While not all edge-cases were covered through simulation, significant error-checks and patches were developed during real test flights. _JAM exposed all of the telemetry it received through a web front-end, 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 _JAM. Added redundancy and edge-case handling are paramount features introduced to _JAM after previous flight-time experience.

4.4 Target Recognition Performance

The Priority List Image Recognition software (PLIR), our image sorting algorithm, 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 PLIR's reliability, we started testing with real images. Model targets were created and photographed so we could have realistic test images. PLIR was used to sort the images we took of targets against the images taken without any targets to further test the reliability of our program.

AIRA was tested separately as it needed to be tested on data transfer and thus did not need data to accurately that reflected the displayed image. We tested AIRA by outputting data to a text file and assuring it conformed to standards outlined in "Appendix E" of the competition rules. Upon confirming AIRA was outputting the text file correctly, we created a dummy interoperability server and tested AIRA's ability to output over the interoperability network.

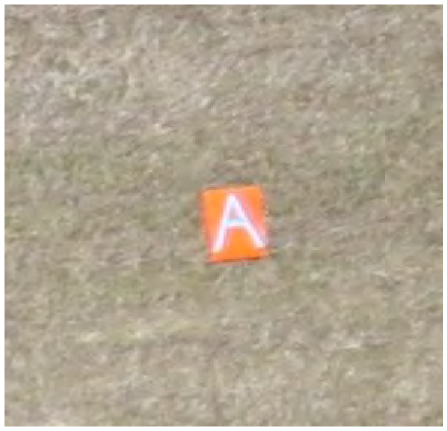


Figure 4.4: cropped target image

5 Operational and Design Safety

When it comes to safety, Team CNU/IMPRINT takes the subject very seriously and put it at the top of our priorities. Before takeoff, the team uses custom checklists to ensure that all equipment and every accessory that is part of the plane works optimally and most importantly works safely as well. The checklists are broken down into different categories that best describe the aircraft and the different portions associated and attached to the aircraft. The Safety Quality Gates are specific points on the checklists, which must be passed. At these points the team member reviews all the safety points they have passed, and reports these to the Mission Commander (MC) and Safety Officer (SO). These Safety Quality Gates help to ensure team communication and safe operation. Each Safety Quality gate helps to ensure that the subteams, as well as the plane itself, are ready for the mission before take-off procedures are initiated.

After the mission, team member meet to discuss lessons learned in the Stop, Start, Continue (SSC) format. We also couple our evaluation of the mission with feedback from the judges later on so that we know how to improve flight safety and procedural design in the following competitions.

6 Conclusions

The CNU/IMPRINT team is looking forward to successfully competing in the AUVSI SUAS 2016 competition. The team is fully equipped to meet the competition requirements and prepared to complete all planned tasks. Further testing will continue to take place to ensure that the CNU/IMPRINT UAS team can safely complete all expected tasks in a timely and efficient manner, however current testing shows the ability to complete all mission requirements. The systems engineering approach ensures a safe and reliable unmanned aerial system. Based on the level of engineering and completed testing, the CNU/IMPRINT team expects to performs well this year.

Appendix A - Cyber Security Considerations

A.1 Real World Threats

The major security threat to modern UAS systems is a hostile takeover. A malicious party can intercept data between the ground station and aircraft, as well as modify data and inject it into a data link. Malicious party interception can result in a breach of the three pillars of security: confidentiality, integrity, and accessibility. Without proper security mechanisms in place, an attacker can know the location and flight plan of the vehicle, construct malicious telemetry data, and gain complete control of the system.

Strong encryption can be used to mitigate data interception threats, but an additional security mechanism should be implemented to prevent data spoofing. University of Texas professor Austin Humphreys suggests that a sensor be included in the on-board payload that detects surplus radio signals and selectively ignores telemetry data coming from the spoofer.

A.2 Mitigation Implementation

To mitigate the interception of telemetry data travelling between the ground station and plane, we created a software encryption layer between the USB and serial interfaces. Plaintext data from the USB interface on the ground station is encrypted with AES and passed through a serial interface to the receiver on the plane. The serial data is decrypted prior to being passed through a USB interface on the plane for autopilot interpretation.

After testing our implementation, we ultimately chose not to include it in the mission-ready package. We assessed that the risks involved in the failure of the encryption layer would significantly outweigh the benefits. Should the encryption/decryption fail mid-flight, all telemetry data would be lost and the autopilot system would no longer function. The encryption implementation posed a threat of increased latency between the ground station and plane. With our relatively slow-maneuvering aircraft, we could not afford a larger gap in response time from the ground station.