AUVSI Student Unmanned Aerial Systems (SUAS) Competition 2018

May 4th, 2018

Rowan University, Rowan RAS

Julia Konstantinos, Breanna Higgins, Scott Hood, Tim Hollabaugh,
Andrew Galloway, John McAvoy, Jeff Stransky, Aaron Guidarelli, Nate Dolnack

Table of Contents

Introduction

This technical design paper was written to describe the design and the rationale behind design choices of the Rowan RAS team entry into the AUVSI Student Unmanned Aerial Systems (SUAS) Competition for the 2018 year. This report aims to show the team's overall coordination and systems engineering process, design tradeoffs, and final design solution. The mission scenario given in the AUVSI SUAS 2018 competition is as follows: 'emergency services are handling a forest fire and have tasked an Unmanned Aerial System (UAS) to locate a person needing rescue and to suppress the fire by releasing water on it.'

Systems Engineering Approach

Mission Requirement Analysis

As per the systems engineering approach used by the Rowan RAS team, the first step was to determine the requirements of the given mission scenario. The tasks which were necessary for the UAS to complete during the mission were analyzed, and are listed with their points in **Table 1** below:

Table 1: Mission Tasks and Point System

Mission Tasks	Point System	Description
Autonomous Flight	**30%**	
➔ Autonomous Flight	40%	The team receives points if the UAS flies autonomously for at least 3 minutes.
➔ Waypoint Capture	10%	The teams will be graded on whether they can fly the entire waypoint sequence and get within 100ft of each waypoint.
➔ Waypoint Accuracy	50%	Teams will be graded on how close they can get to the waypoints in a sequence.
Obstacle Avoidance	**20%**	
➔ Stationary Obstacle Avoidance	50%	Each stationary obstacle will be a cylinder, with height axis perpendicular to the ground, and bottom face on ground. The cylinders will have a radius between 30ft and 300ft, and height between 30ft and 750ft. There can be up to 10 stationary obstacles.
➔ Moving Obstacle Avoidance	50%	Each moving obstacle will be a sphere. The spheres will have a radius between 30ft and 300ft, and move between 0KIAS and 40KIAS. There can be up to 10 moving obstacles.
Object Detection, Classification, Localization	**20%**	
➔ Characteristics	20%	For standard objects there are 5 characteristics: shape, shape color, alphanumeric, alphanumeric color, and alphanumeric orientation.
➔ Geolocation	30%	Teams are awarded points for accurately providing the GPS location of objects.

➔ Actionable	30%	Objects which are submitted during the team's first flight will be considered actionable.
➔ Autonomy	20%	Teams may submit objects manually and autonomously.
Air Delivery	**10%**	UAS should be able to airdrop an object to a specified position.

By analyzing **Table 1** above, the Rowan RAS team was able to determine what systems needed to be built in order to complete these above required tasks. As seen above, the point system used by the competition must also be taken into account, as mission tasks that provided the team with a larger point gain was favored versus mission tasks that may have been easier to achieve, but were not worth attempting to attain.

Table 2: Mission Tasks and the Systems Build or Chosen to Complete Them; Design Tradeoffs

Mission Tasks	Design Tradeoffs	Systems Built/Chosen
Autonomous Flight	The UAS must fly autonomously *for at least 3 minutes*. The UAS must be get *within 100ft of each waypoint*	Pixhawk flight controller, GPS, heavy-duty batteries.
Obstacle Avoidance	The UAS must be capable of *avoiding both stationary and moving obstacles.*	Auto-pilot, Raspberry Pi Camera.
Object Detection, Classification, Localization	The UAS must be able to *detect objects*, accurately provide the *GPS location of objects*, and *automatically and manually submit objects detected during fligh*t.	Auto-pilot, Raspberry Pi Camera, Pixhawk Flight Controller.
Air Delivery	UAS should be able to *airdrop* an object to a *specified position.*	Grasping Mechanism.

Table 2 above shows the systems which were built and or chosen by the Rowan RAS team in order to complete the aforementioned mission tasks. They are accompanied by the subsequent design tradeoffs for those requirements.

Design Rationale

Following the complete and thorough analysis of the mission requirements, the Rowan RAS team was left with a reasonable understanding of the systems which must be built in order to complete mission tasks. However, before the team was able to come to a final design solution for these systems, there were a myriad of environmental factors which the Rowan RAS team had to consider throughout the design of the UAS. The most important of these factors consist of the budget of the team, the number of team members working on this competition, the qualifications of each team member, and finally, the amount of time dedicated to the competition by all members of the developmental team. The design of the UAS was severely limited by the budget allocated to the Rowan RAS team. As this is the first time the team will be competing in a competition as prestigious as AUVSI SUAS, the team was not accustomed to the initial cost of entering the competition. In addition to this, with only nine team members working on the development of the UAS, it was difficult to make significant progress throughout the school year. Time available to work on the UAS was limited for each team member, which caused some design decisions to be made based on time that could be saved by the purchase. Finally, although there were many different design choices available that would satisfy the requirements of this competition, the Rowan RAS team was limited by the qualifications of the Rowan RAS team members. Though some design choices may have appeared more attractive at some points during the design process, the design choices made had to be something achievable by our members skillsets. The culmination of the above environmental factors were taken into consideration as the team reached a final solution for the UAS design.

System Design

The following subsections describes the design of the UAS system. For each system, a description will be found as to what was chosen and or built, why it was chosen, and the implications that decision will have on task performance.

Aircraft

The aircraft which was chosen to act as the UAS in this competition was a large quadcopter. As derived by its prefix “quad” meaning four, and its suffix “copter” meaning vertically mounted propeller, a quadcopter is a sub-group of multirotor which consists of a x-shaped frame in which four motors are attached to each of the far points of the frame. These motors are used to provide the main source of thrust which both propels the aircraft as well as stabilizes it. This aircraft was chosen because it offered both an efficient thrust to weight ratio, while minimizing cost and overall aircraft complexity and increased the feasibility of construction. During initial planning stages, the Rowan RAS Team researched three different multirotor sub-classes: Tricopters, quadcopters, and hexacopters. Early on in the planning stages, the Rowan RAS Team decided against a tricopter design in favor of a hexacopter or a quadcopter mainly due to the availability of open source flight controller softwares currently available. Currently, most flight controller softwares trend heavily towards controlling quadcopter and hexacopter systems, therefore, the team made the decision to decide on one of these two systems as the time cost of using a tricopter based firmware would be too costly. A quadcopter system was chosen over a hexacopter system because of the cost factors associated with system. In general, a hexacopter system is more economically costly than a quadcopter system because of the addition of two additional motors as well as two additional electronic speed controllers. Because of this, the team made the decision to design and build a quadcopter system as it would comply most with the project budget. During the initial research into the three sub-groups of multirotors, the team dedicated time in the research of thrust to weight ratios of each system as well as aspects such as stability for each system. One aspect that the Rowan RAS Team decided to focus on researching was the stability of the different sub-groups as the team felt that the system needed to demonstrate adequate stability to be able to accomplish the assigned task with reasonable efficiency. From the research phase, the team found that as the number of motors increase for the system, the more stable the system is while in the air, but the lower the ratio between thrust and weight. Because of this, the decision to design a quadcopter system was then further justified as the quadcopter system showed stability performances between the other two systems, as well as showing the same trend for thrust to weight ratio. The quadcopter design was found to be the best candidate in which stability, was balanced with thrust to weight ratio.

Referenced below in Figure 1 is the final frame assembly. The final assembly consists of fourteen(14) total members in addition to twenty-four(24) required fasteners. The design centralizes around the Top Plate, in which all other member mates are related to. The attachment of the Top Plate to the arms is achieved using quarter(¼) inch by twenty(20) SAE style bolts. The attachment of the Top Plate to the Main Stand-Offs and then to the Bottom Plate is achieved using the same style and size fastener. The Supports are fastened to the Arms using ten-thirty second(10/32) by 20 SAE style bolts. The Top Plate, Bottom Plate, and Arms were fabricated from quarter(¼) inch 6061 Aluminium Alloy flat stock. The Supports were fabricated from eighth(⅛) inch 6061 Aluminium Alloy. The Main Stand-Offs were fabricated from half(½) inch aluminium round stock of unknow grade and alloy.

Fig. 1 *Rowan RAS UAS Frame Assembly*

Table 3: Frame Assembly Components and Relevant Metrics

ITEM NO.	PART NUMBER	MATERIAL	QTY.
1	TOP PLATE	1/4in FLAT ALUMINIUM 6061 ALLOY	1
2	BOTTOM PLATE	1/4in FLAT ALUMINIUM 6061 ALLOY	1
3	ARM	1/4in FLAT ALUMINIUM 6061 ALLOY	4
4	SUPPORT	1/8in FLAT ALUMINIUM 6061 ALLOY	4
5	MAIN STAND-OFF	1/2in ROUND ALUMINIUM 6061 ALLOY	4

Autopilot

The selection of the autopilot system was determined by two main factors: compatibility with available hardware and interface with competition interoperability server. As such, the system used is a PixHawk PX4 running ArduPilot through APM Mission Planner with additional computations performed on a Raspberry Pi 3 Model B microcontroller connected to the PixHawk through MAVLink communication protocols. This implementation makes full utilization of the computing power available on the native PixHawk controller with great access to autonomous controls through programming of the Raspberry Pi.

Due to the system implemented, the PixHawk is able to receive mission data in a timely manner. In this configuration, the PixHawk is able to communicate real-time flight information to the Raspberry Pi to make smart decisions for autonomous control. With this data, such as waypoint locations, speed, attitude, and GPS readings, the Raspberry Pi aggregates the data to design the flight path the UAS should continue using. This is done through the open source DroneKit library installed on the Raspberry Pi. This system enables high task performance by using the library to make calculations autonomous that the flight controller cannot compute itself. These flight instructions are then relayed to the flight controller to navigate the physical UAS to its objectives.

Testing of these systems exemplify the capabilities of the autopilot system. Testing methodology focused on modular tests of UAS subsystems to establish reliable connections between all systems deployed on the UAS. Initial tests conducted had autopilot disabled for manual control of receiving real time flight data. Fig. XXX shows an example run of the UAS under manual control for collecting GPS data during flight. Tests like this for GPS data help create the basic framework for data collection as needed by the autopilot system. Further manual flight tests were conducted to confirm ability of the UAS to maintain safe and reproducible flight patterns.

Fig. 2 *Example screenshot of mission control software reading real-time GPS data of UAS under manual control*

Further testing focused on the software systems employed by the autopilot system. Another benefit of the selected system was the ability to test software implementations in a controlled environment. The DroneKit API installed on the UAS's Raspberry Pi was also installed on development computers used in designing the autopilot software architectures. Through this work, the local installation of DroneKit was able to simulate the UAS in a paradigm called Software in the Loop (SITL). The SITL simulations allowed for development of systems to communicate flight directions such as motor speeds to the flight controller from the base computer (the Raspberry Pi on the UAS) according to received flight data. Reasonable flight data was simulated based on data collected in manual testing.

This development pattern also enabled early implementation of further intelligent systems for the Raspberry Pi such as the system for Object Detection, Classification, and Localization. This allows for through debugging of the systems prior to deployment on the final UAS design.

With modular tests complete and successful, final systems integration enabled all capabilities of the chosen autopilot system. The system is fully capable of the desired tasks as identified in section "System Engineering Approach: Mission Requirement Analysis."

Obstacle Avoidance

The obstacle avoidance functionality of the quadcopter is achieved by a combination of pre-planned flight paths created in ArduPilot MissionPlanner and on-the-fly avoidance instructions sent from the Raspberry Pi to the Pixhawk. Before flight begins, the Pixhawk's autonomous flight plans are set in MissionPlanner. One of the features of MissionPlanner is the ability to designate boundaries for the drone's flight. The boundaries set in MissionPlanner are GPS coordinates that the Pixhawk will not allow the drone to cross during flight. This preflight boundary planning is able to set avoidance zones for static regions which correlate into the drone avoiding static obstacles.
A second, more versatile obstacle avoidance mechanism is also implemented with the use of the Raspberry Pi to Pixhawk MAVLink protocol. Flight instructions are able to be sent from the Raspberry Pi to the Pixhawk and telemetry data is able to be read from the Pixhawk by the RaspberryPi via MAVLink protocol. This enables the RaspberryPi to be used as an on-the-fly obstacle avoidance module. A Python script that tracks both static and dynamic obstacle coordinates runs on the Raspberry Pi. The same script also reads telemetry data from the Pixhawk via MAVLink protocol to determine the drone's current GPS coordinates and if the drone is approaching an obstacle, the Raspberry Pi is able to send navigation signals to the Pixhawk to avoid crossing the obstacle boundary. This obstacle avoidance solution was chosen because avoiding both static and dynamically changing obstacles were aspects of this competition. While MissionPlanner supports static boundary avoidance for autonomous flight, there is no way to account for moving obstacles. The Raspberry Pi MAVLink solution was chosen due its ability to avoid dynamic obstacles and using nearly the same code and protocol as the Imaging System to interface between the Raspberry Pi and the Pixhawk.

The utilization of two obstacle avoidance methods increases the reliability of the drone to succeed at obstacle avoidance. Both the pre-flight and on-the-fly obstacle avoidance methods account for static boundaries which provides redundancy if one fails to navigate the drone away from obstacles during flight.
The RAS team successfully tested the MissionPlanner aspect of the static obstacle avoidance by testing the GPS module and Pixhawk telemetry during flight to ensure the Pixhawk is able to accurately determine its location and avoid pre-defined boundaries.

Imaging System

The imaging system on the UAS is built into the Raspberry Pi microcontroller. This entails the main Raspberry Pi connected to the PixHawk flight controller over MAVLink protocols with a Raspberry Pi Camera v2 connected to the camera input of the microcontroller. The Raspberry Pi camera interfaces with the microcontroller using both native Raspberry Pi imaging software and the open source OpenCV library for advanced image manipulation. This system was selected due to its versatility in multiple applications, community standards, and familiarity to the engineers working on this system. Rigorous testing tests yields a high probability of successful completion of the mission tasks given this system.

In accordance with the system testing methodology used on the UAS as a whole, several tests were performed on the imaging system in an isolated environment prior to deployment on the UAS. Tests of the imaging system were performed by connecting the Raspberry Pi and its camera to a local development machine to review of its image capture abilities and performance in image manipulation. This involved the creation of simple "test targets" to be shown the system for processing. As the final deployed system is mounted facing the world in the perspective of the UAS, tests had the camera facing a wall at various distances to simulate the height of the UAS during flight. The system then collects images through the Raspberry Pi Camera and performs real-time image manipulation on the microcontroller. Fig. XXX shows an example output of the system where it received image data from the camera alongside simulate telemetry data to perform image manipulation calculations as would be required of the target mission tasks.

Fig. 3 *Example imaged produced by imaging system on capturing test target information and processing region bounding in real time*

These tests demonstrate the success of the designed imaging system. The success of these tests allow for systems integration to incorporate the functionality on the UAS.

Object Detection, Classification, Localization

The Object Detection, Classification, and Localization system chosen for the UAS utilizes the systems described in section “System Design: Imaging System” to perform the computer vision tasks required for fulfilling mission requirements. It is designed to perform calculations on the Raspberry Pi to relay to the PixHawk flight controller over MAVLink protocols which enables telemetry to update the interoperability about located targets in real time autonomously.

Fig. 4 *Diagram of desired system architecture for implementing Object Detection, Classification, and Localization*

Using the information received by the camera's image data stream as well as the flight controller telemetry data, a three part system is designed as shown in Fig. 4. After the target image data is captured, the image manipulation software uses two processes for the describing the target: an Optical Character Recognition (OCR) model for identifying the letter and a feature detector for identifying target shape and color. Parallel to these processes, the computer vision system is able to "region bound" the target for further processing. This involves using the feature detector to locate where the target is positioned within the captured image and to augment this data with a set of points that describe the boundaries of the target against its background. After this is completed, the system performs the algorithms described below to use the image data and telemetry data to interpret the location and orientation of the bounded image in relation to real-world coordinate systems. This collects all the data needed to packetize information into JSON format for real-time updating of the interoperability system of collected object data. This enables for automatic processing of images without manual intervention, although the pilot still maintains normal access to the interoperability server.

As mentioned previously, one of the main components of this system is the feature detector used for the target information. The first step of this system after receiving the image capture is to convert the image into a matrix of Hue-Saturation-Value (HSV) information. With this known, a threshold is set to find Hue information which deviates from the statistically defined average Hue of the image, thus revealing the approximate location of the target. The image is further processed to find the exact target by filtering the image with common computer vision techniques and separating the specific shape from its background. This shape then undergoes feature matching to classify the shape as one of the known shapes application to the mission targets. This process yields the shape and color information needed for the object data, and further outputs information used by the other two processes for data collection. This process is implemented on the Raspberry Pi using the OpenCV libraries.

The algorithm for locating the target object is performed again to locate the letter within the target object. This data is then feed into a custom-designed OCR for identifying the letter contained as well as its orientation. This custom OCR is implemented as a Convolution Neural Network (CNN). To design this system, first a dataset must be created which creates example images to be shown to the network. This dataset was created with the 26 letters at each orientation (North, South, East, West) under various forms of noise (Gaussian blur, Gaussian Noise, and inverted contrast) of 7800 grayscale images. These images are split into two labeled datasets: "test" and "train." A network is designed with "convolutional layers" (layers for extracting information from images) and "fully connected layers" to develop the model for the letters. A backpropagation algorithm is performed to optimize the ability of the system to correctly output the known label of the training data as it is supplied to the CNN over many iterations. At each iteration, test data is used to evaluate the accuracy of the system in identifying the letters. The final "trained" CNN takes the letter image processed by the Raspberry Pi and outputs the expected letter and orientation of the target information needed for the object data. This CNN model and its process is deployed on the Raspberry Pi using the Keras Deep Learning libraries with a Tensorflow machine learning library backend.

The image data found by the feature detector is further used to interpret the image information into real-world coordinates. This is done by using the computer vision algorithms described by Sun et al: "A Camera-Based Target Detection and Positioning UAV System for Search and Rescue (SAR) Purposes" [1]. The specification document for the Raspberry Pi yields the intrictic information of the camera itself [2]. Information for homography transformation is found by the GPS position, compass heading, and attitude information received from the telemetry data. This information is applied to the image data to extract the location information needed for the object data. The compass heading is used to perform a coordinate system transformation needed to calculate the real-world position of the target within the image. The UAS attitude describes the "distance to the camera" which is used to calculate the distance in meters described by each pixel in the Raspberry Pi camera capture. This transformations are used to calculate the offset of the center of the target in relation to the UAS itself. This offset is converted into GPS coordinates relative the GPS reading of the flight controller for the longitude and latitude location data of the target.

These three processes are successfully capable of producing object data from the targets. This data is compiled on the Raspberry Pi and transferred to the PixHawk over the MAVLink protocol. The PixHawk is then able to autonomously upload the object data to the interoperability server using its established communication systems.

Communications

Our main communications system was chosen to be compatible with the PixHawk and Ardupilot. We decided on a 915 MHz serial radio specialized for the MAVLink protocol. The MAVLink protocol was is used because it is natively understood by Ardupilot, and can be used without any hardware or software modifications. There are also existing libraries to run on the base station that implement MAVLink, simplifying things further. MAVLink is also designed to handle communications to UAS, and it can handle all of the data we need to send and receive. The data being sent from the UAV includes the gps location, sensor data, and the current status of the UAS. It can also send data from the coprocessor, such as targets found. The ground station will also send data to the UAS. This can include commands such as Return To Launch, Land, to start a mission. It is also used to upload new waypoints fly to, scan coordinates, and locations of obstacles to avoid. In addition to telemetry, the safety pilot's transmitter runs at 2.4GHz. It transmits the current flight mode, arm/disarm, manual takeover, as well as stick controls for manual takeover. This was designed to be seperate from the main communications for safety and reliability. This way, there are two seperate ways to activate the return to launch and land features if needed. Also, the pilot transmitter will always work in the case of a manual takeover. The last communications involved is a video transmitter. This transmits the current video as seen from the camera on the UAS, allowing the safety pilot to take over even if the UAS is out of visual range. This runs at 2.4GHz.

Fig. 5 *Rowan RAS Communications Block Diagram*

Air Delivery

Fig. 6 *Rendered CAD output of desired drop mechanism on UAS frame design*

Fig. 7 *Rendered CAD output of payload carrier, consisting of a custom built release mechanism*

For the payload carrier, we custom built a bottle release mechanism with a simple latching pin design that will release the payload on command. The servo controls a Scotch-Yoke style mechanism that pulls out a pin releasing the water bottle. There are 4 components to this entire design: a servo motor, retaining pin, base-plate, and the actuating yoke which converts the rotational motion of the servo into linear motion to pull the pin. The baseplate also contains the retaining arms that will hold the bottle.

We went through a number of designs created in solidworks and focused building a design that was simple and contained as little moving parts as possible to minimize complications as well as ensure swift troubleshooting in the event of a latch component failure or a failure to release. This also simplified the build process as their are only 4 components to the entire release mechanism. Our final design should allow for minimal disruptions as long as friction between the pin and retaining holes is kept to a minimum and the weight of the bottle/payload does not exceed the amount of torque produced by the actuating mechanism. The base-plate, yoke, and retaining pin were all 3-D printed using PLA-type plastic.

Bench testing was done prior to flight testing and good actuation and clearances between the latch pin and pin holes was noted. During weighted payload testing it was discovered that the servo could not produce the torque required to actuate the pin for payload release. The base-plate was modified and reprinted to place the servo higher and more in-line with the direction of travel and a thicker yoke mechanism was also reprinted. All testing completed after these modifications showed successful bottle release and no further issues.

Cyber Security

The Rowan RAS team elected to not heavily invest time into the cyber security aspect of the UAS. This decision was made after assessing potential cyber security threats, and noting that the threats posed were insignificant. The Rowan RAS team still took steps to protect the aircraft, payload, and data. This was achieved mainly by limiting access to the UAS and its associated payload and data. The Rowan RAS UAS has only one access point, the test server. In this way, outside communication and interference was limited. In future competitions, the Rowan RAS team will make cyber security a greater priority.

Safety, Risks and Mitigations

The Rowan RAS team takes the safety of all members very seriously. Safety is a top priority for the SUAS competition, as well as for the Rowan RAS team. The subsections below describe the potential safety risks inherent to the competition and the steps taken by the Rowan RAS team to mitigate them.

Developmental Risks and Mitigation

There were many risks posed throughout the developmental stage of the UAS design process. These include risks posed during construction and testing of the UAS, as well as human error which could affect the performance of the UAS. In order to mitigate the risks associated with construction and testing, personal safety equipment was required to be worn at all times. Safety goggles were worn when building the UAS and working with power tools necessary for the completion of the UAS, as well as when testing any individual component of the UAS, as well as its flying ability. All team members were required to take a short safety course offered by Rowan University, which taught team members how to behave in a lab setting, as well as how to deal with equipment. Checklists were also used during testing to ensure no dangerous mistakes were ever committed. This mitigated any human error which could have occured in the developmental stage.

Mission Risks and Mitigation

Safety risks posed by the competition mission and autonomous flight were similar to the risks posed during the developmental stage of the UAS. However, due to the nature of the competition, the UAS will be around a greater number of people who may not be aware of safe practices around an aerial vehicle. A greater number of steps were taken to mitigate the new risks brought on by an additional number of people present around the UAS. Additionally, as the UAS will be autonomous throughout the competition mission, safety risks are prevalent as the UAS is no longer under manual control. To mitigate the risks described above, a number of safety features were built into the UAS. Personal protection equipment will be worn by all those nearby the UAS. The UAS itself has safety features built in, such as a ‘kill switch’, RTL functions, and safety features inherent to the auto-pilot software used by the UAS.

References

[1] Jingxuan Sun, Boyang Li, Yifan Jiang, Chih-yung Wen, "A Camera-Based Target Detection and Positioning UAS System for Search and Rescue (SAR) Purposes", *Sensors*, vol. 16. 2016

[2] Raspberry Pi Camera V2 Specifications, https://www.raspberrypi.org/documentation/hardware/camera/README.md