Systems Engineering Approach

Mission Requirement Analysis

To begin the systems design process, we began by looking at the specifications. From the detailed specifications that were provided, we were able to define both functional requirements and several use cases.

Input-Output Functional Requirements (IORP)

1.0 The system shall (TSS) be able to alter the Unmanned Aerial System (UAS) altitude (Turn up and down)
2.0 TSS be able to alter the UAS speed
3.0 TSS be able to turn the UAS left and right
4.0 TSS capture photographs at a set rate
5.0 TSS capture photographs at a set quality
6.0 TSS aim its camera at a GPS marked location
7.0 TSS drop an 8oz water bottle on a GPS marked target
8.0 TSS direct a UAS to a GPS marked location
9.0 TSS communicate the UAS location to a server over ethernet
10.0 TSS match search-grid object to a library autonomously
11.0 TSS drop a water bottle on a target
12.0 TSS operate in up to 15 knot winds with 20 knot gusts
13.0 TSS operate in temperatures of up to 100 degrees sustained and 110 F peak
14.0 TSS operate in 2+ miles of visibility
15.0 TSS receive manually inputted coordinates
16.0 TSS report UAS location at a rate of under 1 HZ
17.0 TSS plan a flight plan for the UAS
18.0 TSS be able to be set up in a set amount of minutes

For the use cases of the system, we thought through the potential processes that would be requested of the plane, and walked through the actions that would be necessary to complete these processes.

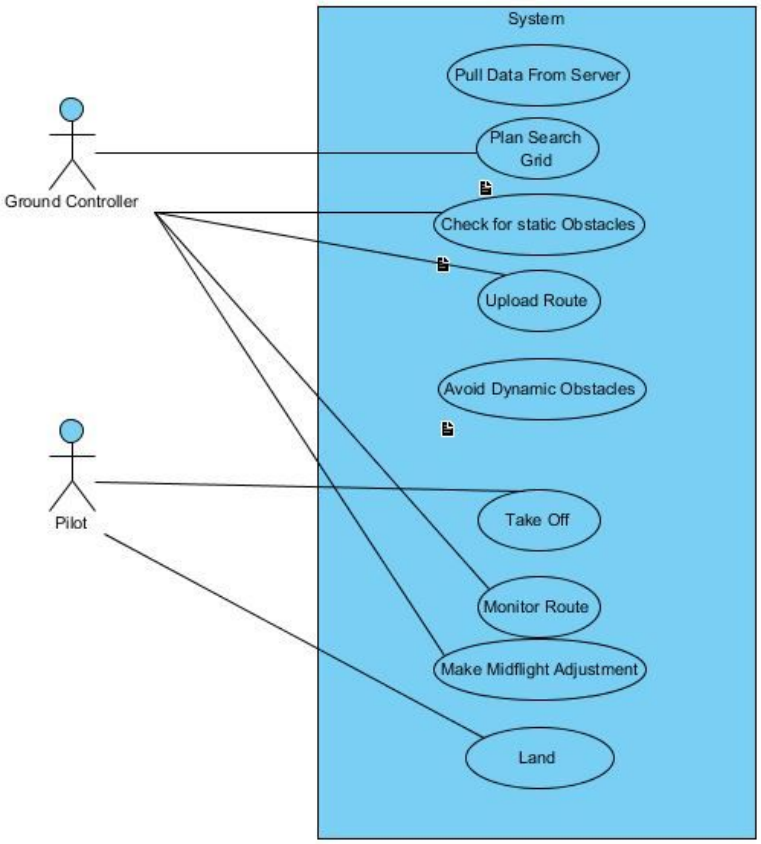

Figure 1 Human Interface connections

Having defined the use cases for the plane and solidified the requirements, we next turned to the functional architecture of the UAS. In both the physical and software designs, we established the parts needed to accomplish the necessary purposes.

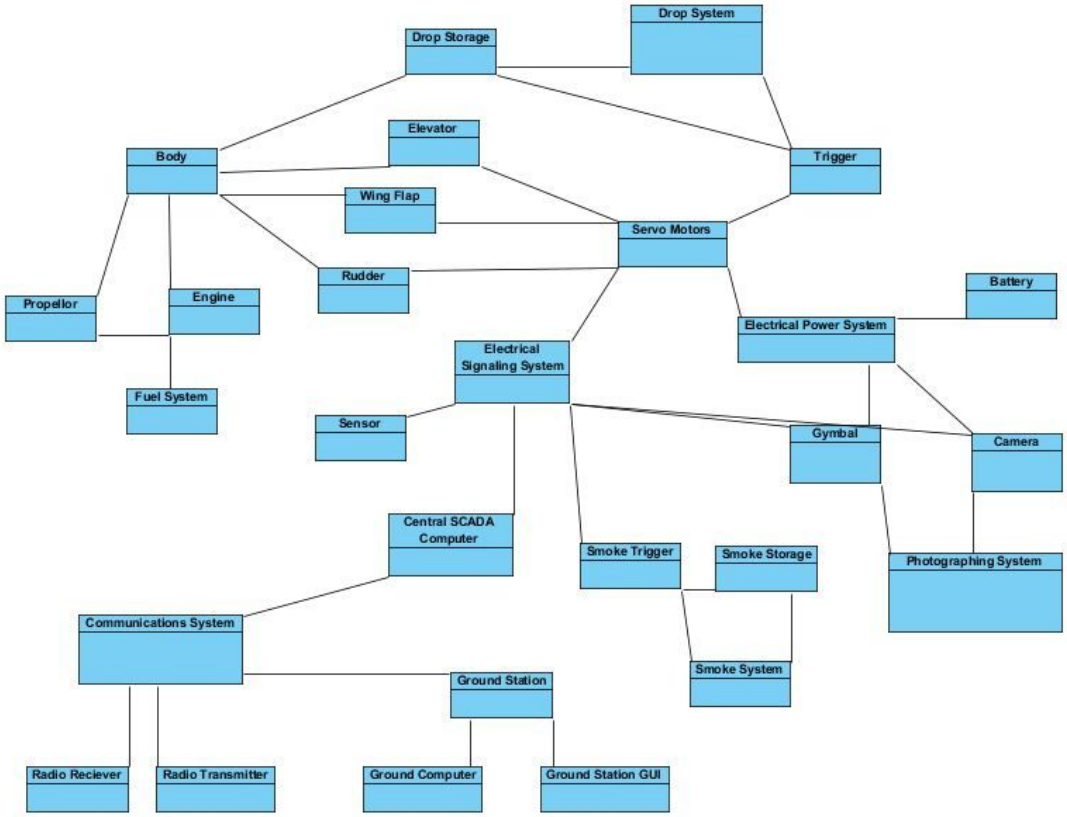

Figure 2 Plane Functional Diagram

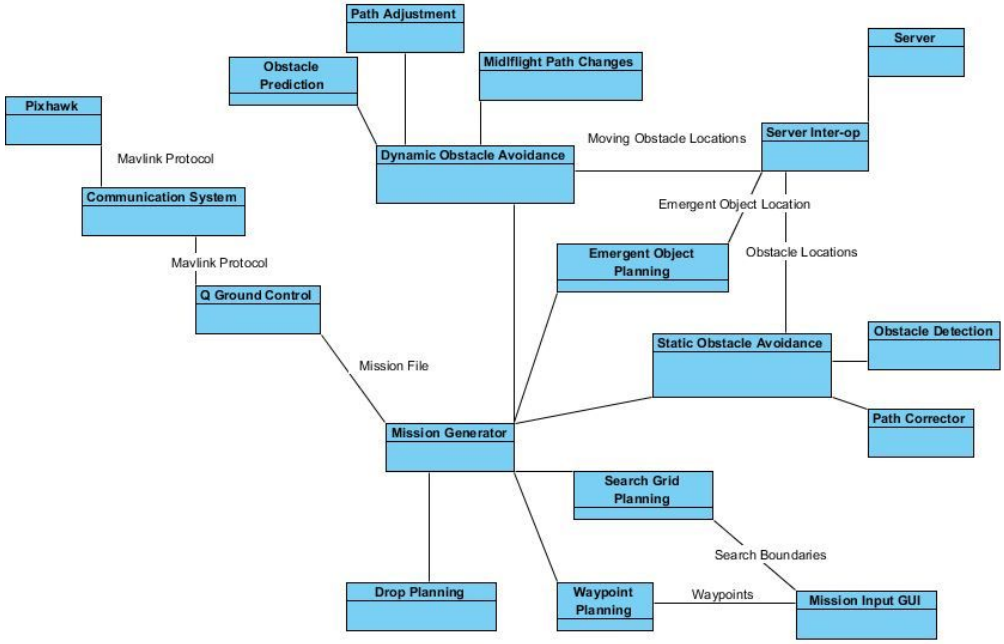

Figure 3 Control System Software Functional Diagram

After creating a functional design for the system, we next moved on to making the design choices to realize that design in the real world.

Design rationale

Arizona Autonomous consists of primarily Electrical and Computer Engineering and Computer Science students, as well as an Optical Engineer (President), Aerospace Engineer, and Systems Engineer. These students are almost all Juniors and Seniors, with the exception of a couple of Freshman on the hardware team. The budget for this year's aircraft was ~$12,000. The club this year had the challenge of creating a stable platform for future competitive teams to build on, as well as compete in this year's competition and collect data.

Hardware Selection

The tasks of this competition, as outlined in the 2018 rules, is to complete waypoint navigation, object detection, and payload delivery via autonomous flight. The points awarded are proportional to how well each task is completed, and how timely each task is completed. In order to meet the competition requirements, we needed to select a platform that was capable of comfortably carrying the payload which included the camera system, on board processing computer, flight controller and bomb drop. Based on last years success with a fixed wing, we decided to use a fixed wing again. The primary reason for this is because of the long flight times. This allows us to attempt everything we could want to within the allotted mission time, and instead of rushing due to battery life, we are able to complete the mission methodically. The next step was to decide on a platform. One of the limitations of last year's aircraft was its takeoff and landing stability. We therefore decided to get a larger plane. Since we originally planned on using a large G-Shot camera, it was also beneficial to go larger.

The next selection process involved determining what camera to use. We were originally going to use the G-Shot, however due to this cameras limitations we decided to look into other systems. We ultimately decided on a small machine vision camera because it had a very large sensor. It also has great software capabilities allowing for us to connect it to a Linux based computer for processing and settings. The lens was selected to fill the aperture, but have the right magnification to allow for 2 pixels per inch.

The propulsion system was relatively easy for us to decide on. There are only a couple engine manufacturers that make aircraft engines for this style of aircraft. We have a very good relationship with the owner of Desert Aircraft, so we decided on their engines. In terms of power, we were in between 150cc and 200cc. We chose 150cc because we are not looking for high acceleration or power; we are primarily interested in fuel efficiency and steady power. Finally we needed to select between the in line or side by side. Classically, in line are more efficient, however it doesn't fit in the cowl of the selected aircraft, therefore we selected the side by side.

The final decision required was the autopilot. We have been using Pixhawk for years and are very pleased with its performance, so we decided to stick with the Pixhawk systems. After research, and due to a GPS failure we experienced last year, we decided to try the Pixhawk 2.1. The Pixhawk 2.1 has many redundant systems which was the most attractive functionality for us. Since one of our members is also rather familiar with GPS systems, and has used the GPS chip used by the Pixhawk 2.1, it was strongly advised to use the GPS system included with the Pixhawk 2.1.

System Design

Aircraft

The aircraft we selected to use this year was the Hangar 9 3.1m Sukhoi ARF. The reason we selected this aircraft is because its payload and flight characteristics have been heavily tested by our safety pilot. He flies the same platform competitively in acrobatics competitions countrywide. Due to this experience on the platform, he strongly suggested this aircraft to be our flight platform. Heavy modifications were required to make this platform work in the competition. Using a CNC router, multiple mounting tabs have been designed and cut to host the power supply, as well as the flight controller. These were mounted on replaceable platforms to allow for replacement of failed components, and future upgradeability. The aircraft has a wingspan of 3.1 meters, a length of 3 meters, and a height of 0.8 meters. There is a total wing surface area of 17.9 square meters. This provides for a large lifting body, and is capable of, during experimentation, taking off with 40 pounds of weight with no notable issues. There is also a very large open fuselage allowing for the easy installation of our components. The propulsion system is a Desert Aircraft 150cc gasoline engine with a Mejzlik 32x10 Evo propeller. This propulsion system generates more than enough power to get the aircraft flying at a steady 70 miles per hour, well above our required airspeed.

Figure 4 This years aircraft

Autopilot

The aircraft uses the Pixhawk 2 hardware running the px4 stack autopilot software to run the mission flight. The Pixhawk 2 is able to run the onboard px4 stack software and also comes packed with a reasonably accurate GPS unit, along with more than enough parts needed for any of the plane's external hardware. The px4 stack was chosen because it allowed a mission to be planned in preflight using the QGround Control software as well as its ability to push and pull new coordinates along its current mission path, allowing the modification of flight as to avoid obstacles. The ground station used for this system uses the QGround Control software and is ran on a laptop along with a Pelican holding the high end hardware.

Figure 5: Last years ground station (will be used again this year)

Obstacle avoidance

Our dynamic avoidance algorithm predicts collisions with dynamic obstacles and determines an optimal avoidance path that minimizes risk. The algorithm works in three main stages; collision prediction, generation of avoidance paths, and selection of path. In the first stage, the code checks whether any of the dynamic obstacles will cross the intended path of the UAV by projecting vectors at angles from the direction of travel of each obstacle. If these vectors cross the plane's path of travel, a collision is defined.

At each possible collision, the code establishes several dodge points, representing various degrees of left and right dodging. For each of these dodge points, the code generates 20 feasible alternative paths passing through this point. Finally, to select the optimal path, the code uses obstacle prediction equations to evaluate the relative riskiness of each generated path. The path that gets selected is the lowest risk path running through the dodging point with the lowest average risk level.

Imaging system

For an imaging system to properly identify targets on the ground, an identification minimum must be met in terms of pixels per inch. For proper identification, at least 2 pixels per inch are required. Calculations were performed by finding the required magnification to project 2 pixels per inch on the ground. We ended up using a JAI GO RGB camera for a few reasons. Its pixel count is 2560 by 2048. When paired with a 25mm focal length lens, this gets us our 2 pixels per inch from 200 feet altitude. We also have an MTF of 100 lp/mm which will allow us to identify the difference between letters with no issue. Finally our ground image size on the sensor is 106 feet by 84 feet. Our shutter speed is 30 frames per second as well, which allows us to fly at much higher speeds than we are capable of flying at. A final design consideration was to find a camera with a global shutter. Since we are flying at fast speeds, rolling shutter will cause separation between pixels as the information comes off of the sensor onto the computer. If we had this separation, it would cause our identification capabilities to be severely reduced. Due to this, we want a global shutter so all information is saved per pixel locally before it is offloaded onto the cameras processor. Finally this camera has a Linux support structure, and connects to the onboard processing computer with a USB 3.0 connection.

Object detection, classification, localization

The imaging system based upon the JAI GO includes five separate components that work together to take a captured image and perform the necessary processing. Once an image is captured by the JAI GO, it is transferred to the onboard computer, and preprocessing and ROI detection are performed. This data is then transmitted to a base station computer via a 2.4 GHz link established with two Ubiquiti Bullet M5s. On the base station computer the image is run through a series of convolutional neural networks (CNNs). If a a target is detected, a JSON packet is created that is sent for manual review and then passed to the interop server for judging.

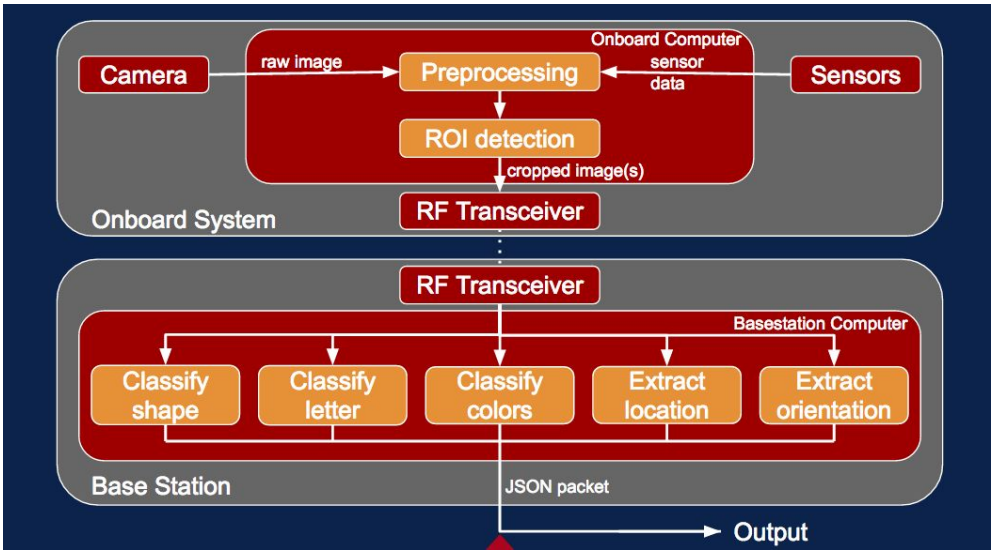

Figure 6. Image processing system overview

After an image is received, the onboard preprocessing stage consists of packaging the image with relevant metadata from the aircraft's sensors such as heading and position. The image was then processed to determine if any regions of interest (ROIs) existed within the frame. In order to detect ROIs, OpenCV first blurs an image then transforms it into various different color spaces such as HSV. Edge detection is then performed on the images by thresholding operations and the Sobel operator. Contours that were extraneously detected were removed using various sanity checks such as size of contour and frequency. Remaining contours are packaged with extrapolated geolocations and the cropped image and its metadata is sent to the base station using the onboard Bullet M5.

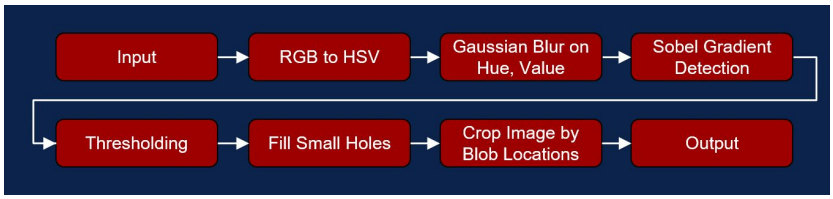

Figure 7. ROI detection algorithm overview

At the base station, the cropped ROIs are passed through a series of CNNs to classify the alphanumeric, alphanumeric color, shape, and shape color of the target, if a target exists in the frame. In order to increase the accuracy of the CNNs being used, training of the neural networks was completed using synthetic images that we had created along with real images captured from the 2017 AUVSI SUAS competition and test flights. To label the training images, a GUI program named Image Labeler was created using Qt Creator, which allowed an image to easily be classified with the necessary characteristics. Image characters and the images were then packaged into a JSON packet that could easily be used by the other components in our system. After being processing by the CNNs, the images were then passed to a modified version of Image Labeler which was capable of real-time processing allowing us to verify the accuracy of target information before submission. Once the data was verified, it is then passed to the interop server for judging.

Communications

We are using two frequencies for communication from the plane to the ground. The first system we are using is a long range telemetry radio operating at 900mhz. This band carries all of the telemetry data between the ground station and the aircraft. This allows us to receive GPS information, autopilot health, and aircraft speed. We are also able to send emergency commands to the aircraft should a hazardous situation arise. The next frequency we are using is 2.4ghz for transmitting image data to the ground. It acts as a wireless ethernet link for us to transmit photos to the ground station for processing at high speeds.

Air delivery

The payload will be delivered using an E-flite Servoless Payload Release mounted on the underside of one of the wings. The water bottle attachment was designed in Solidworks to fit the water bottle and reduce movement during flight. This part was 3D printed to enable more testing runs and easier replacement. The new part was tested both manually and through servo signal with the manufactured release mechanism to ensure that it worked properly. The drop timing was calculated assuming a terminal velocity of 70 mph. At 200 ft, the timing is ~4s.

Cyber security

When considering any cybersecurity scenario, one must begin by characterizing the goals of the process or system. In this case, these goals are derived from the functionality needed to complete the provided mission. The functionality for a UAS can be divided into two rough categories. The first category is those functions necessary to keep the plane in the air. These include the steering systems, propeller and fuel systems, and a subset of the flight control capabilities. The second category of functionality is those functions designed to accomplish the secondary tasks of the mission. These include computer vision, the water bottle drop mechanism, obstacle detection, and the more complex autonomous flight functions.

Having divided the UAS capabilities into critical and secondary categories, we next pursue a rough characterization of the system at hand to identify areas of vulnerability. As can be seen from the functional systems diagrams, the UAS is divided roughly into two main subsystems. The first is the ground control. In our case, this is an on the ground computer with a radio. The second is the in the air component. This subsystem contains the plane, the onboard computer/controller, and a radio.

From a cybersecurity perspective, the ground control computer is relatively safe. It is air gapped from the internet, which means that any threats would come through pre-existing malware on the ground control or the judging computer. For similar reasons, the airborne subsystem is also relatively safe. Any potentially damaging cybersecurity breach would come through a pre-flight attack, which would require physical access to the plane.

By far, the weakest link in the system is the radio link between the two subsystems. Assuming an unsecured channel, a radio link is vulnerable to an extensive list of attacks, including jamming, denial of service, data injection, and any hacks on either subsystem possible through the attack vector of the radio ports.

We did not conduct an extensive cybersecurity analysis and mediation effort for this project. We did however, take basic precautions with the radio link.

Safety, Risks, and Mitigations

Developmental Risks and Mitigations

With such a large build and a small team, time constraints were one of our biggest risks. In order to mitigate this, we carefully divided up the team into different sections. Some working on software, some on hardware, and finally some on the autopilot functionality. Since software was mostly done, the hardware team had the most work ahead of them. Not only did a fuel system need to be designed, the plane needed to be assembled and heavily modified to handle all the new components. To reduce manufacturing time, we acquired a 3D printer and a CNC machine. This allowed us to model things in Solidworks, and have them made with little input. This gave the hardware team time to focus on assembly. Due to this, there are about 20 different custom parts in the plane, including a custom gimbal and multiple modular mounting systems.

Another major risk was object detection. It was the first time we designed an object detection software, so we were unsure of its behavior. In order to mitigate this, we gained access to a private field near our campus where we simulated digital targets, and tested our obstacle avoidance.

Mission Risks and Mitigations

Safety Risks Posed by Competition

The Hangar 9 3.1m Sukhoi ARF is large and heavy aircraft. This makes it hazardous to bystanders and the safety pilot when it takes off, flies, and lands. To avoid the potential risks, we never fly the plane over or near people. If the aircraft loses radio for 15 seconds, the plane will implement a “suicide function” that immediately crashes the plane into the ground. This prevents the risk that the plane will continue flying and potential fly over or crash into people, buildings, or obstacles. Additionally, we included redundant IMU, GPS, and accelerometer to maintain stability of the aircraft during flight.
The propeller on the plane rotates at extremely high speeds. This poses a safety risk to people within a close range of the propeller. To prevent a safety risk, we enforce a 15 foot safety radius around the aircraft while the propeller is in motion. Additionally, if the plane is set to take off, there will be no one in the front of the plane.

Safety risks Posed by Autonomous Flight

A potential safety risk is that the autonomous flight may not be modulally coded correctly and could crash. To avoid this, again we included redundant IMU, GPS, and accelerometer to maintain stability of the aircraft during flight. Additionally, we have the option for manual takeover to allow the safety pilot to take over the aircraft. Another safety risk is a complete loss of communication. In this event, the system recognizes its total system loss, and will give a maximum input command to all control surfaces, and turn off the engine. During this failure mode, the results will be catastrophic, causing the aircraft to crash near its current position. In the event of a total system failure, there is a bypass switch that allows the pilot to shut off the engine. The plane will eventually crash but with the engine off, it will allow for minimal damage and minimal crash speed.