AUVSI SUAS 2018
Technical Design Paper

Rutgers University
School of Engineering

May 5, 2018

Abstract

The AUVSI SUAS competition presents a number of challenges for participating teams to accomplish, including autonomous waypoint navigation, image capture, object detection, classification and localization, interoperability, and air-delivery. From these, RUAutonomous, representing Rutgers University, has developed a system called Rtemis to accomplish the tasks deemed feasible to complete given the time frame and past experience of the team.

Contents

1 Systems Engineering Approach

1.1 Mission Requirements Analysis

The first step of any large scale engineering project is to lay out exactly what is to be achieved by the system to be built. These goals will then drive each of the individual design decisions made later and will provide a framework for how to approach the problems. RUAutonomous conducted a preliminary assessment to address potential challenges before development. The team started with looking at each of the competition tasks and deciding what would be required of the UAS to complete each task. This allowed RUAutonomous to look into how each task influenced the requirements of software and hardware components.

Task Name	Description	System Requirement
Timeline (10%)	• Team is provided 20 minutes of setup time • Total mission time of 45 minutes • Mission is graded as max(0, 255 - 5*flightTime-postProcessingTime) • Timeouts are allowed costing 20% of points	• Make sure the team has multiple flight practices to ensure team setups in time to hear the mission briefing • Reduce timesouts by making sure all components are setup properly before flight • Reduce post-processing time by having a fast image processing pipeline
Autonomous Flight (30%)	• UAS must fly for at least 3 minutes • Lose 10% of points for each manual takeover • UAS must navigate to specified waypoints • Waypoint accuracy is graded as max(0, 100ft - distance)/ 100ft	• Rtemis system will utilize an autopilot capable of navigating to specified waypoints • Autopilot must support autonomous takeoff and landing • Extra sensors are necessary for precise landing
Obstacle Avoidance (20%)	• Aircraft must avoid stationary obstacles • Aircraft must avoid moving obstacles	• Utilize some form of path-planning • Path-planning fits into waypoint paradigm • Autopilot must support ability to alter aircraft paths in-flight
Object Detection, Classification, Localization (20%)	• Aircraft must take pictures of Objects scattered on the ground • UAS must identify shape, color, alphanumeric, orientation and location of Object • UAS gets extra points for completing processing in-flight	• Fast image processing system to identify all Objects before UAS lands • UAS must be able to get timing of image for localization • Ideally one algorithm to identify all characteristics
Air Delivery (10%)	• UAS must drop an 8 oz water bottle over specified location • Water bottle must disperse all water on impact	• UAS must contain a dropping mechanism capable of autonomous release • The team must do the necessary calculations to ensure all water is dispersed and target is hit
Operational Excellence (10%)	• Subjective measurement of team performance • Examples include: operational professionalism, team communication, reaction to failure, and attention to system.	• Ensure the team has a checklist to ensure all members are on the same page • Come up with a plan to deal with failures • Ensure all practice fly days are executed safely.

Table 1: Mission Requirements Analysis

The team chose the software and hardware of the Rtemis system to meet the requirements of Table 1. The design of the system to complete each task is describe in the sections that follow.

1.2 Design Rationale

RUAutonomous began its design with an analysis of the shortcomings of previous year's systems. The team first focused on improving the existing system before attempting to pursue other competition tasks. Next, the team analyzed the environmental factors affecting such improvements. Finally, RUAutonomous looked into which new tasks the team would attempt. These three components drove this year's design.

1.2.1 Redundancy and Failure

This past competition, the team's aircraft crashed due a vital sensor failure. The previous year's system did not take redundancy and system failure into account. This year it was considered the most important component of the system design. The sensor that failed was the airspeed indicator. The Rtemis system utilizes three airspeed sensors for redundancy. In addition, there are three on-board batteries: one for the receiver, autopilot and servos, one for the imaging system and one for the motors and ESCs. RUAutonomous also did research to find other high quality components such as a better GPS. A better checklist was also developed to make sure all equipment is secure and only the features of the autopilot the team wants during flight are enabled. Lastly, a parameter in ArduPilot was utilized to have the aircraft to switch over to using GPS speed if the airspeed indicator fails.

1.2.2 Improved Camera

One of the team's areas of needed improvement has always been the imaging system. RUAutonomous has struggled to find the ideal camera and on-board computer system to succeed in the imaging system task while keeping it lightweight. The team believes that Rtemis has been designed with what they consider to be the optimal system for years to come. Throughout past competitions, the team has tested many cameras. Each one was found to have an issue in one of the following areas: weight, reliability, image quality and camera coverage, all of which the team believes are essential for the ideal camera. This past year RUAutonomous tested four cameras and in the past has tested a total of eight cameras. Only one met all of these requirements. The team chose the Blackfly S, which weighs only 36 g. The Blackfly S is an industrial grade Machine Vision camera that puts reliability first. The camera, having a 1.1" sensor with 12.3 MP of resolution, provides the required image quality to resolve even the smallest of Objects. In addition, with a 25 mm lens it provides the required coverage to complete the search-area task in time. The Blackfly S has exceeded all of the team's expectations for the system's camera. The only downfall of the Blackfly S was its price. The camera costs a total of $2600 with the high quality lens. The team believes this camera was worth the purchase due to RUAutonomous' past issues with cameras.

1.2.3 Improved Imaging Processing

The third area of needed improvement was the on-board computer. Last year, the team used the heavy Intel NUC. The NUC, due to the team not completing the autonomous ODLC task because of a weak image processing system, was underused and unnecessary weight. The team chose this computer last year due to the camera's requirement of working with an Intel X86 based system. However, the Blackfly S supports ARM64 systems, including the Jetson TX2s. Last year the team attempted to use a weak ODLC system. The team has struggled to find a system that is able to identify and classify Objects with high accuracy until this year. The team decided to switch over to Deep Learning; Rtemis uses solely Deep Learning for all of the ODLC tasks in addition to plane telemetry. This system now identifies Objects with very high accuracy while maintaining speed. Since the system uses Neural Networks, the team utilizes the Nvidia Jetson TX2. Rtemis uses two of these devices, which together weigh less than the Intel NUC.

1.2.4 Environmental Factors Analysis

After determining what changes needed to be made to the current system the team assessed the environmental factors affecting the design. The allocated budget is one of the most limiting factors for the design. The team this year was able to get more allocations than in year's past. However, decisions still had to be made to decide which components were worth purchasing. In addition, this expensive equipment is at risk during flight and duplicates cannot be afforded. This was accounted for by adding sensor redundancy as mentioned in the previous section. In addition, the proper pre-flight checks are done to reduce the risk of failure in-flight.

1.2.5 Attempting new tasks

In previous years, the team was able to complete waypoint navigation via the Pixhawk flight controller, autonomous takeoff and landing via LiDAR, Object imaging and manual ODLC. Finally, RUAutonomous considered which new tasks the team would attempt this year. These consisted of Obstacle Avoidance and Air Delivery. The team's main two focuses were Autonomous ODLC and Obstacle Avoidance due to the considerable amount of time and effort required to achieve both. For Obstacle Avoidance the team utilized a path-finding algorithm called Rapidly-exploring Randomizing Trees due to its ease of integration into the waypoint navigation paradigm. Overall this year's design consisted of improving on previous year's and attempting new tasks not yet achieved. The following sections present RUAutonomous' Rtemis system which achieves all of these tasks.

2 System Design

2.1 Aircraft

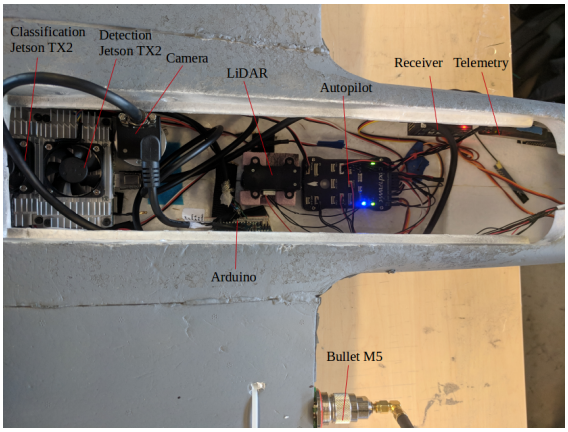

Figure 1: Top facing view

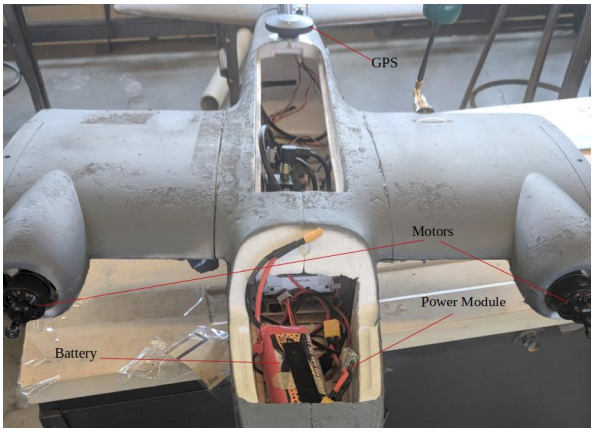

Figure 2: Forward facing view

RUAutonomous has chosen to use the My Twin Dream (MTD) airframe. The MTD uses dual electric motors and has a 1.8 m wingspan and a 1.2 m length. This airframe was chosen due to its large payload bay, long range capabilities, and the team's past experience with the airframe. A second airframe, the Skywalker Titan, was considered, but ultimately rejected due to its higher cost and heavier weight. In addition, a hexacopter was considered because of its ease to setup, launch and land, but was ultimately rejected due to its lower flight time.

Aircraft Specifications and Performance					
Length (mm)	1230	**Motor**	Cobra 2814-850kv	**Cruise Speed (m/s)**	20.28
Wingspan (mm)	1800	**Prop Size**	9x6	**Stall speed (m/s)**	10.83
Height (mm)	350	**ESC**	30A	**Thrust/weight**	.68
Weight (kg)	3.30	**Battery**	LiPo 4s 10000mAh	**Pitch speed (m/s)**	22.22
Wing Chord Length (mm)	280	**Mixed flight time (min)**	21	**Max Current (A)**	30.5

Table 2: Aircraft Metrics

After an airframe was chosen, a combination of batteries, Electronic Speed Controllers (ESC), motors, and propellers was then selected. A 4-cell 10-Ah Lithium-Polymer (Li-Po) Battery was used to power two Cobra 2814-10 850KV brushless motors with 30-amp ESCs. A 10A BEC was also connected to the battery to supply power to the Pixhawk. A separate 4-cell 2.2-Ah Li-Po battery was used to power the two on-board Nvidia Jetson TX2s. The ideal propeller configuration was a 2-blade, 9 to 10 inch diameter, 6 inch pitch to generate a thrust to weight ratio of .78 and a pitch speed of 89 km/h. The MTD airframe was also modified to support the imaging system. A hole was created on the bottom of the fuselage to be aligned with the camera and its mountings, and a dedicated external mounting point was created to mount the imaging antenna on the port side of the wing. Table 2 shows the aircraft metrics.

2.2 Autopilot

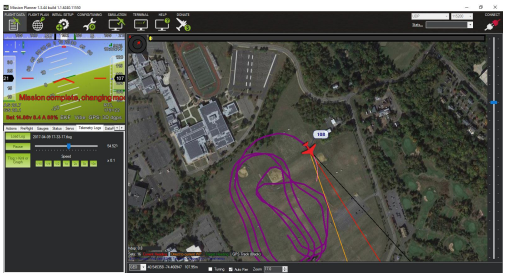

Figure 3: Mission Planner

The team uses the 3DR Pixhawk Autopilot running the open source software ArduPilot due to its capability to support many sensors, reliability, and extensive documentation. Since ArduPlane is opensource, this allowed the team to make modifications, including support for injecting alternative routes for the Obstacle Avoidance task and the required fail-safe systems for the competition. The system runs the plane firmware and is connected to a ground station running Mission Planner via RFD 900+ transceivers operating in the 900MHz range.

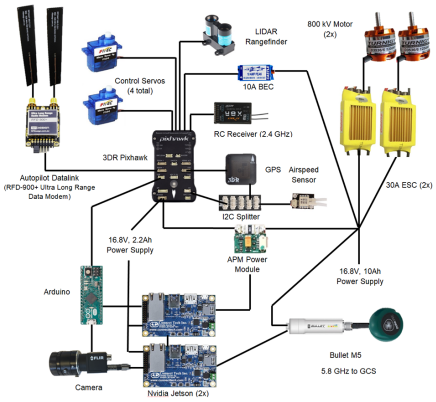

Figure 4: On-board Connections

The Pixhawk is also connected to a Taranis RC receiver to maintain contact with a Taranis transmitter over 2.4GHz and allow for manual control of the aircraft. Other externally mounted sensors include a Here M8N GPS unit providing geolocation and heading, three Pixhawk airspeed sensors, LiDAR for precise landing, and a power module providing current and voltage monitoring. Figure 3 shows the Autopilot GCS known as Mission Planner. Mission Planner receives telemetry through MAVProxy which is directly connected to the UAS via the RFD900+. Figure 4 shows the on-board connections to the autopilot.

2.3 Obstacle Avoidance

Last year, RUAutonomous developed an obstacle avoidance algorithm using basic geometry and algebra. However, due to its high failure rate in certain scenarios, the task was not attempted. This year, the team took an interest in the path-finding algorithm called Rapidly-exploring Random Trees (RRT). RRT is capable of quickly generating a feasible path, but fails to optimize the path. Therefore, constraints were added to the sample space in which the path waypoints are generated to maximize the odds of creating appropriate paths.

Modifications to the ArduPilot firmware were made to allow for efficient uploading of new waypoints mid-flight. A new MAVLink mission command was created to bypass the check if ArduPlane is accepting a mission and to guide the plane before reaching the next mission waypoint.

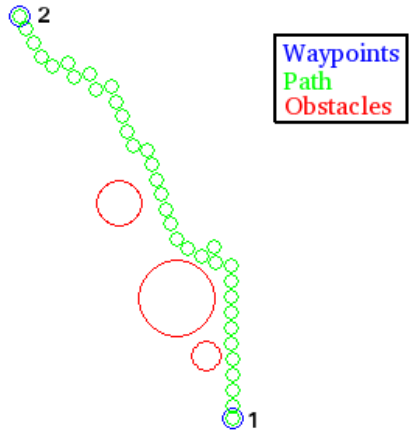

Figure 5: A path connecting two points generated by RRT

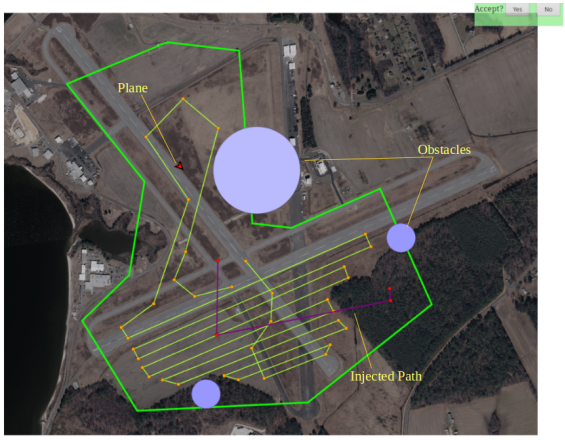

Figure 6: Obstacle Avoidance GCS

To inject operator control into the obstacle avoidance process, a custom path viewer was built into the team's Central Control Server (CCS), including features for viewing the current aircraft location, stationary and moving obstacles, and current mission path. When the Obstacle Avoidance system creates a new path, the path is presented to the operator for approval before finally being committed to the aircraft. An "operator-in-the-loop" system like the one RUAutonomous developed protects against the risks associated with a fully autonomous obstacle avoidance system, like potential damage to the aircraft or inadvertent deviations from the Object search path.

The Obstacle Avoidance algorithm runs on the ground in the team's Obstacle Avoidance Engine, in its own dedicated process that can also be distributed to a separate computer The engine communicates with the CCS to receive telemetry and allow the operator to view and deny paths.

2.4 Imaging System

Rtemis utilizes the Blackfly S 1.1" 12.3MP Machine Vision camera fitted with a Computar 12MP 25mm lens (Figure 7). RU Autonomous tested four cameras this past year with the focus on the ability to resolve the smallest Objects, coverage and mission time, ease of integration, and reliability.

Figure 7: Blackfly S

The Blackfly was chosen because of its ability to produce high quality images, programmable via an easy to use API, and lightweight. The Blackfly S weighing only 36 grams is controlled via a Nvidia Jetson TX2. The TX2 communicates with the ground allowing the user to control a wide variety of camera functionalities in-flight via the camera's Spinnacker SDK, such as Analog Gain and frame rate. The following shows that the camera is able to resolve even the smallest of objects at the desired search altitude of 61 m. The team set the required resolution for 1 foot Objects to be 36 pixels in the horizontal direction. The team found that this resolution was sufficient for the Autonomous ODLC system to detect and classify Objects. Equations (1) - (3) show that the camera and lens meet this requirement.

$$HorizontalFOV = Distance * SensorWidth/FocalLength \tag{1}$$

$$HorizontalFOV = 61m * (14.19mm/25mm) = 34.62m \tag{2}$$

$$ResolutionOfObject = .30m * (4112pixels/34.62m) = 35.63pixels \tag{3}$$

RUAutonomous verified these calculations with real world tests. Along with the resolution requirement, RUAutonomous utilized Mission Planner's survey tool to figure out the required frame rate of the camera and speed of the aircraft given the coverage, required overlap and mission time requirements. It was found that the aircraft going at a speed of 18 m/s taking pictures at 1 frame per second provides enough overlap and completes the search-area in 12 minutes.

2.5 Autonomous Object Detection, Classification & Localization

Due to the incredible advances in the field of Deep Learning over the past few years, the team decided to design Rtemis's ODLC system to consist solely of Convolutional Neural Networks (CNN's) with Multi-Task Machine Learning. CNN's are non-linear function approximators that adjust model parameters via

Gradient Descent to minimize a non-convex loss function. All CNN's were implemented using Google's Tensorflow API. However, due to the computational and memory requirements of Deep Neural Networks, the system uses two Nvidia Jetson TX2s. Multi-Task Learning means that the approximator loss consists of multiple components or "tasks." The system is split into two components, as seen in Figure 8: Detection and Classification.

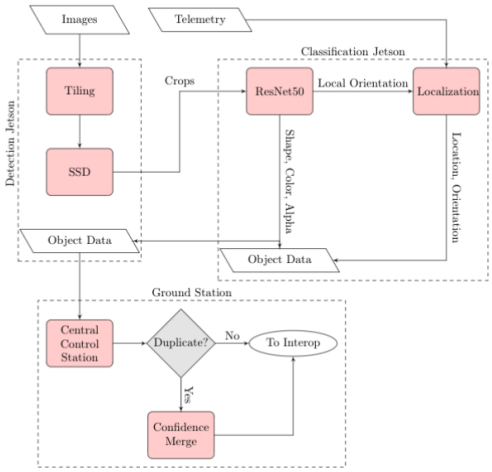

Figure 8: The Autonomous ODLC Pipeline

2.5.1 Object Detection

For Object detection, Rtemis utilizes the Single Shot Multi-box Detector (SSD). SSD learns via multi-task learning on the location of the bounding boxes of Objects and their associated classes. The model was trained on fifty thousand 1000 × 1000 computer generated fly day images containing around 15 Objects per image. It learned to find the location of the bounding box and classify the shape of the Object. The team also designed a custom version of SSD that learned to classify multiple characteristics of a single Object. However, it was found that learning multiple characteristics affected the accuracy of the bounding box localization, so the system was designed to split detection and classification into two components. Since SSD works on a resolution that is lower than that of the input image, the system first extracts 1000 × 1000 crops, or "tiles", from the input image. The detector then processes them in parallel on a GPU. SSD consists of a CNN that simultaneously predicts multiple bounding boxes called "default boxes" of various scales and aspect ratios. The network then outputs a correction for the location of the box and a class label for it. SSD on average produces no false-positives and is able to identify all Objects in a picture with high confidence. The bounding boxes of two classified objects in a single tile are shown in Figure 9.

Figure 9: SSD output on a single tile

2.5.2 Object Classification

The second component is Object Classification. The system utilizes ResNet50 which was trained to classify, simultaneously, the shape, shape color, alphanumeric color, alphanumeric, and local alphanumeric orientation. The local orientation is then combined with telemetry information to calculate the global cardinal direction of the Object. ResNet50 was trained on over two million 150×150 computer generated Objects (Figure 10). If false-positives are detected by SSD, ResNet50 removes them by setting a confidence threshold on all network outputs; if the class confidences are below a certain percentage the Object is marked as a false-positive.

(a) Computer Generated Object

(b) Real Object

Figure 10: Computer Generated and Real Objects

2.5.3 Object Localization

Localization, could not be performed with Deep Learning, as it is dependent on too many external factors that use the plane's location. The following algorithm is used to find the GPS location of objects. Equations 6 and 7 return the Object's Latitude and Longitude, respectively.

Variable	Quantity
α_h, α_v	Horizontal & Vertical View Angle
$pixBelow, pixToLeft$	Pixels below/left of image center
$pixWidth, pixHeight$	Image Width & Height, pixels
β	Aircraft Bank Angle from Vertical
h	Aircraft Altitude, meters
lat, lon	Aircraft latitude & longitude, pixels
R	Average Earth Radius, meters

Table 3: Air Drop Parameters

$$c_x = 90^\circ - \beta - \alpha_h\left(\frac{pixBelow}{pixWidth} - 0.5\right) \tag{4}$$

$$x = h\left[\tan\left(\beta - \frac{\alpha_h}{2}\right) + \frac{\cos\left(\beta - \frac{\alpha_h}{2}\right)\sin\left(\alpha_h \frac{pixBelow}{pixWidth}\right)}{\sin(c_x)}\right] \tag{5}$$

The equations for the y - direction are completely symmetric to the x - direction.

$$Latitude = \frac{lat}{R}\left[\sqrt{x^2+y^2}\cos\left(\arctan\left(\frac{x}{y}\right)\right)\right] \tag{6}$$

$$Longitude = \frac{lon}{R\cos(lat)}\left[\sqrt{x^2+y^2}\sin\left(\arctan\left(\frac{x}{y}\right)\right)\right] \tag{7}$$

Telemetry information is sent to the Classification Jetson from an Arduino Micro. The Arduino Micro receives a signal via the Blackfly S's GPIO port when a picture is taken. The Arduino then takes a snapshot of the telemetry it is viewing from the Pixhawk's telemetry port and sends it over USB to the Classification Jetson.

2.5.4 Autonomous Duplicate Handling

Rtemis also has an automatic Object merging component implemented on the ground station that detects duplicate Objects. If two Objects are within a certain threshold distance of each other, they are considered duplicates. Duplicates are then merged probabilistically to form a new Object; the new Object will have higher confidences for classes that match and lower confidences for classes that mismatch.

2.6 Manual Object Detection, Classification & Localization

2.6.1 Central Control Server

The Central Control Server (CCS) is a RESTful web application that communicates with the on-board computer, client image viewers, obstacle avoidance viewer, obstacle avoidance engine, and the Interoperability server. In addition, it performs tasks such as submitting Objects, posting telemetry, fetching obstacle locations and retrieving the current mission. Thus all systems are fully integrated into the web server. The CCS is implemented using the Django web framework and it's RESTful extension. The following sections describe the different aspects of the CCS for manual ODLC.

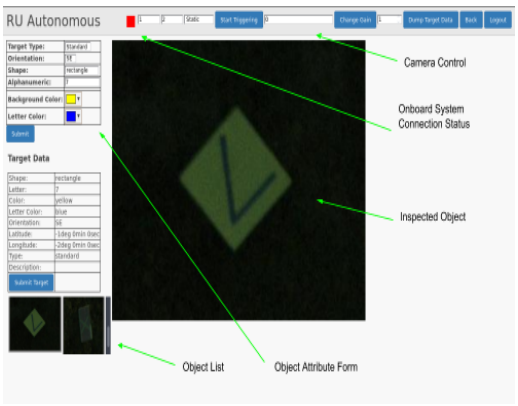

Figure 11: Interface for image viewer

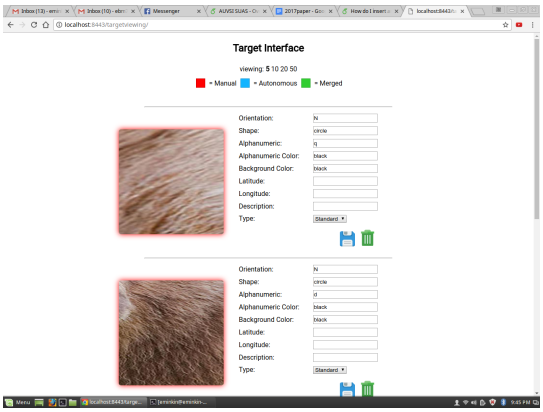

Figure 12: Target viewing page

2.6.2 Image Viewer

Each viewer (Figure 11) functions as a slideshow for the images and requests them at a fixed rate, which can be paused when the user finds an Object. The user can then zoom in on the image, crop out the area with the Object, and enter information such as Object shape, alphanumeric, color, alphanumeric color and an additional description in the case of the Emergent Object. To save bandwidth, the user views a decimated version of the image. The cropped coordinates are then sent to UAS which has the original image. The on-board Jetson TX2 then maps the coordinates to the original image and sends down the resulting crop. After the user submits the Object's information, it is updated in the database. The CCS supports multiple image viewers and provides different images for each one by keeping a central queue of unevaluated images. Information such as Objects found is synchronized across the viewers so that users can check and ensure that no duplicates are created. The viewer also allows users to modify Objects, dump Object data to a text file, and submit Objects to the Interoperability server.

2.6.3 Object Viewer

The CCS also supports an Object viewing page (Figure 12) in which a user can view all submitted cropped images with their characteristics. Through this interface, the user can re-crop images or edit characteristics of the Object itself such as alphanumeric, object shape, color, etc. These edits can be propagated through both the CCS and also sent to the Interoperability Server should the user wish to do so. The user is also provided the option to delete certain images and Objects from both the CCS and the Interoperability Server.

2.7 Communications

2.7.1 UAS Communications

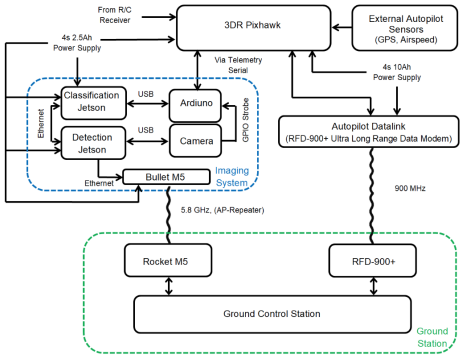

Figure 13: UAS Communications

Two separate connections between the UAS and ground (Figure 13) are established during the aircraft's flight. The on-board imaging system consists of two Nvidia Jetson TX2s which communicate over a Local Area Network via Ethernet over USB. The Detection Jetson sends images and classified Objects to the ground over the Ubiquiti Bullet M5 via the 5.8 GHz bandwidth. The Classification Jetson receives image telemetry information via an Arduino Micro which connects to the Pixhawk to receive telemetry over serial. A second connection between the Pixhawk autopilot and the Autopilot Control Station is maintained through the RFD900+ modem for long range communications within the 900 MHz bandwidth.

2.7.2 Ground Communications

The ground station (Figure 14) consists of two main components: the Central Control Server (CCS) and Autopilot Control Station (ACS). The CCS integrates all ground components, acting as the ground imaging server, obstacle avoidance ground control station, and the interoperability client. It contains endpoints for the UAS to post images and classified Objects. In addition, it has endpoints for the obstacle avoidance engine to request obstacle updates, telemetry and post paths. The Ubiquiti Rocket M5 is an access point for the UAS to join the ground network and contact the CCS.

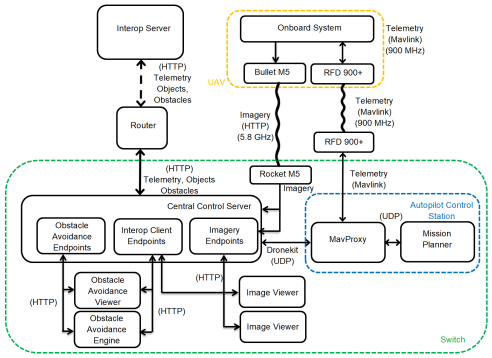

Figure 14: Ground Communications

2.8 Air Delivery

The dynamics of the air delivery are influenced by two major components: quadratic air resistance as a function of velocity, and gravity. The equation of motion is therefore:

$$\vec{F} = m\vec{a} = -Dv^2\hat{v} + m\vec{g} \tag{8}$$

Where D, the drag parameter, is given by $D = \frac{\rho C_d A}{2}$.

Breaking up into x and y components:

$$F_x = m\ddot{x} = -Dvv_x = -\frac{\rho C_d A}{2}\dot{x}\sqrt{\dot{x}^2 + \dot{y}^2} \tag{9}$$

$$F_y = m\ddot{y} = -Dvv_y - mg = -\frac{\rho C_d A}{2}\dot{y}\sqrt{\dot{x}^2 + \dot{y}^2} - mg \tag{10}$$

The equations of motion are a highly nonlinear coupled set of linear differential equations, which is not possible to solve analytically. Therefore, a numerical solution was found using MATLAB. For a given altitude $y(0)$ and a aircraft velocity $\dot{x}(0)$ and $\dot{y}(0)$, the horizontal distance the projectile would travel before reaching the ground $y(t) = 0$ can be calculated using this method. The values for the parameters used in Equations (9) and (10) are given in Table 15.

Variable	Quantity	Value	Units
ρ	Air Density	1.2	$\frac{kg}{m^3}$
C_d	Drag Coefficient	0.85	None
A	Cross Sectional Area	0.00257	m^2
m	Projectile Mass	0.25	kg
g	Gravitational Acceleration	9.81	$\frac{m}{s^2}$

Figure 15: Air Drop Parameters

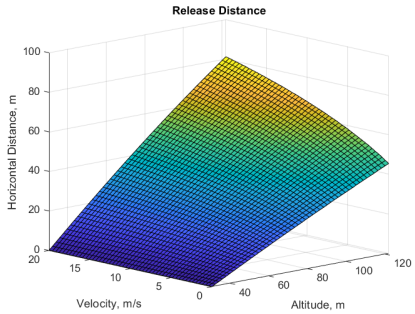

Figure 16: Plot showing drop distance

Assuming the aircraft's position relative to the target is known, the distance at which the aircraft must drop the projectile for a given speed and altitude can be computed. Figure 16 shows this distance as a function of the velocity and altitude. During flight, the distance to the drop zone can be calculated by interpolating between data points on this graph. This, in turn, will determine the precise drop time.

2.9 Cyber Security

Security is an important component of system design. RU Autonomous designed Rtemis from the start to take security into account. The first step when performing a security analysis is to define the Threat Model. RUAutonomous then performed a full threat analysis of the system and determine the security controls to put in place to help prevent such attacks. In addition, a fallback action is determined, if possible, when such systems fail. The following (Table 4) presents the threats to the system, the security controls in place and the plan for dealing with when such security controls are broken. The team follows the STRIDE threat analysis. Each attack is ranked from 1 to 3, where 3 is the most detrimental.

Rank	Asset	Attack	Description	Security Control	Fallback
1	**ODLC Object Local Edit/ Deletion**	Spoofing/Tampering	Attacker gains ability to edit/delete Objects from local database.	TLS/SSL: Adds non-repudiation, authentication and integrity. Utilize JSON Web Tokens for authentication.	• Objects also submitted to interop server • Attacker can't gain other abilities due to permission isolation
2	**ODLC Object Interop Submit/Edit/Delete**	Spoofing/Tampering	Attacker gains ability to Submit Objects to and/or edit/delete Objects on the Interop server.	TLS/SSL and JWTs	• If attacker doesn't have access to local database, submit objects with USB thumb drive • Attacker can't gain other abilities due to permission isolation
3	**Obstacle Avoidance GCS**	Spoofing/Tampering	Attacker gains ability to accept/reject injected avoidance paths	TLS/SSL and JWTs	• Pilot takes over to avoid dangerous injected paths • Attacker can't gain other abilities due to permission isolation
3	**MAVLink Commands**	Spoofing	Attacker finds vehicle ID, and injects MAVLink commands	Safety Pilot attempts to put aircraft into manual and land	• Terminate flight if aircraft cannot go into manual via Safety Pilot kill-switch
3	**MAVLink Commands**	Denial of Service	Attacker sends invalid MAVLink commands to deny GCS communication with aircraft	Safety Pilot attempts to put aircraft into manual and land	• Terminate flight if aircraft cannot go into manual via Safety Pilot kill-switch
3	**CCS Availability**	Denial of Service	Attacker performs DoS on CCS	No DoS protection is in place	• Attempt to whitelist IPs
3	**RC Jamming**	Denial of Service	Attacker jams Safety Pilot RC signal	Attempt to still land aircraft via GCS command	• If all connection is lost, flight will automatically terminate
3	**Telemetry Jamming**	Denial of Service	Attacker jams GCS connection to UAS	Safety Pilot attempts to land aircrafts	• If all connection is lost, flight will automatically terminate

Table 4: Threat Model and STRIDE Threat Analysis

3 Safety Risks and Mitigation

3.1 Developmental Risks and Mitigations

Risk	Outcome	Mitigation
Low Budget	Can't afford new components or backups	Purchase items careful. Perform cost analysis through design matrices.
Injury	Team member or spectator is hurt during testing.	Providing checks, following guidelines and safety procedures. Performing multiple simulations before actual tests.
Component failure	Component fails during testing.	Purchase or make backups. Test carefully and make sure proper checks are done before testing.
New UAV Policies	A new UAV policy or rule is broken.	Make sure to always check for updates in FAA regulations. Ensure it is legal to fly at the field.

Table 5: Developmental Risks and Mitigations

There are a multitude of risks that can occur during developmental testing. UAV technology is a constantly evolving field and the regulations and safety procedures need to be followed carefully. Table 5 outlines the risks, outcomes of such risks and the mitigations to avoid them.

3.2 Mission Risks and Mitigations

Risk	Outcome	Mitigation
Compass not calibrated	Inaccurate heading	Calibrate compass before flight
Airspeed Sensor Clogged	Inaccurate airspeed	Redundant airspeed sensor, inspect for clogs and calibrate before flight
No GPS Lock	Unable to navigate to waypoints	Calibrate GPS before flight. Ensure GPS has position lock before takeoff.
Unexpected Propeller Movement	Injuries to hand or other body parts in contact with propeller	Safety switch installed on plane to cut power quickly. Throttle arming feature on Pixhawk enabled.

Table 6: Pre-Flight Mission Risks and Mitigations

Risk	Outcome	Mitigation
Loss of RC Signal	Unable to control plane manually	Plane switches to return to home mode to attempt to regain signal.
Loss of telemetry signal	Unable to update telemetry or add waypoints	Plane switches to return to home mode to attempt to regain signal.
Loss of all signals	Unable to control aircraft	Continue with mission. After 30 seconds of no signal, switch to return to home immediately. After 3 minutes of no signal, enter flight termination mode
Battery Below Critical Voltage	Insufficient power to aircraft, damage to battery	Voltage sensor installed to monitor battery and go into return to land mode if below a critical voltage.
Interference between radio components	Loss of signal or GPS	Keep components apart from each other and from motors. Ensure no components transmit on similar frequencies.
Battery Failure	Loss of power or loss of control of aircraft	Both batteries are connected to the Pixhawk Servo rail to provide a redundant power supply to the controls.
Overheating	Components fry or stop working	Airframe modified to provide airflow to critical components

Table 7: In-Flight Mission Risks and Mitigations

The design and programming of the plane and autopilot system incorporated numerous safety features. When modifying the frame of the My Twin Dream aircraft, weak spots on the airframe were noted and reinforced with fiberglass coating to avoid structural failure in flight. The ESCs, motors, and batteries were also chosen to provide a thrust to weight ratio that would allow for maneuverability during potential evasive maneuvers. In addition, the airframe of the plane was modified to provide airflow around the Ubiquiti Bullet M5 and the Jetson computers to avoid overheating. The electrical system of the plane was also designed with safety as a priority. The GPS, telemetry antennas, and receiver antennas are positioned apart from potential sources of interference including the motors, Ubiquiti Bullet M5, and the Nvidai Jetson TX2s to reduce the chances of signal loss. In addition, dual antennas are positioned perpendicular to each other to minimize dead spots in their radiation pattern. Furthermore, the plane includes an accessible hardware safety switch which provides an emergency cut off for all power. Finally, wires are kept organized to allow ease of inspection. Table 6 and Table 7 note the pre-flight and in-flight mission risk and mitigations.