UAVND Technical Design Paper

Unmanned Aerial Vehicles of Notre Dame
College of Engineering
University of Notre Dame
Notre Dame, IN
UAVND@nd.edu

Abstract

This report details the progress that was made over the course of the early fall and late spring in developing a clean-sheet design for UAVND's first AUVSI SUAS competition vehicle. A detailed description of the requirements, acceptance criteria, and testing plan are also included below.

Index Terms

Signal Temporal Logic (STL), Robot Operating System (ROS), Simultaneous Localization and Mapping Algorithm (SLAM), Light Detection and Ranging Sensor (LIDAR), Flight Controller Unit (FCU)

I. Introduction

This technical design paper outlines the Unmanned Aerial Vehicles Club of University of Notre Dame's (UAVND) system for the 2021 AUVSI SUAS competition. This system offers a distributed approach to autonomous missions. Through the use of multiple computing devices distributed across each component of the competition and connected through the Robot Operating System paradigm, the UAV system is capable of autonomous route planning, object avoidance, aerial mapping, and object identification and classification. By distributing the computing over multiple devices a focus on redundancy and adaptability is used to create a versatile aerial platform. Which is then tailored specifically to the requirements of the AUVSI SUAS competition. The following sections outline the specific requirements for meeting the competition goals, an overview of the ideas behind the system, a detailed review of hardware and software choices, as well as a safety and testing plan for the UAV system.

Seeing that this is our first year participating in the AUVSI SUAS competition, we were tasked with constructing a clean-sheet design. The design process follows the systems engineering "V" model which is shown in Figure 1. As such, the design process began with every team member thoroughly reading the 2021 competition rules document and constructing a list of explicit requirements. After doing so, we came together as a team and created a comprehensive description of the overall system requirements and acceptance criteria. This description is presented in the following section.

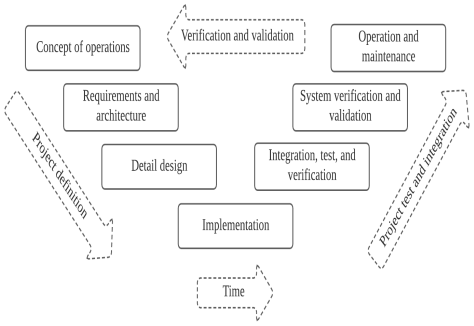

Fig. 1. This Figure illustrates the approach taken to designing the system

II. Requirements and Acceptance Criteria

System	Competition Criteria	Derived Requirements
Aircraft	-Fly the waypoint, mapping, and alphanumeric objectives in 30 minutes -Operate in 110°F and 15 knot winds with up to 20 knot gusts -A max take off weight under 24.9 kg -No exotic fuels	-Sustain a cruising speed of 19.5 m/s for 40 minutes -Carry a payload consisting of alphanumeric camera, mapping camera, and UGV -Electric propulsion system
Autopilot	-Must be capable of autonomous flight -Accurately reach waypoints -Manual takeover penalty	-Autonomous mission capability -Waypoint accuracy of <6ft -Reliable and robust in high ambient temperatures and other adverse conditions
Obstacle Avoidance	-Telemetry uploaded to the interoperability server at a rate of at least 1Hz -Avoid stationery obstacles -Avoid dynamic objects (i.e. other airborne teams)	-Strong telemetry link between the aircraft and the groundstation to ensure 1Hz update rate can be met -Ground station capable of creating a flight plath through the waypoints and around the obstacles -Airtraffic control using the telemetry data of the other teams through the interoperability server to take action if a collision is going to occur -Onboard 2D LiDAR used to detect other dynamic objects
Imaging System	-Mapping capable camera -Alphanumeric recognition capable camera	-Clean interface with Ardupilot to take pictures at specific points in a flight plan -Stream live video for computer vision system to identify objects and then take a high resolution image for further processing -Onboard computer capable of detecting objects in a live video feed -Resolve competition alphanumeric symbols from a cruise altitude of 250 ft.
Object Detection, Classification, Localization	-Detect objects in and around the competition area (up to 250 ft. outside area and within obstacles) -Classify the alphanumeric characters detected -Determine the GPS coordinates of the object -Must not detect non alphanumeric objects	-Use a live video feed with computer vision running on an onboard single board computer to identify where potential objects are located -Higher resolution images are taken of the objects for further image processing to determine the character -Through the use of the altitude of the plane, a fixed camera position, the pitch and roll angles, and the location of the object in the image, the location of the object can be triangulated -Need to train the machine learning specifically for the competitions' alphanumerics
Mapping	-Create a map conforming to Web Mercator Projection	-Must be able to take high resolution images and offload them to a computer to be stitched together -Images must be able to be taken at specific times and locations
Communication	-Communicate telemetry, radio, VTX, and pictures from the aircraft to the ground vehicle -Capable of robust connection throughout the entire competition area in the worst conditions	-Set up an LTE network and VPN system so the systems all appear to be on the same network -All communication links must have at least 10km of range with antenna tracking
Air Drop	-Capable of delivering a 8 oz water bottle to a target location once dropped -No thrusters or propellers can be used when dropping the ground vehicle	-Total weight of the vehicle will be less than 1.4 kg in order to meet the aircraft weight requirement -An appropriately sized parachute to ensure a secure drop without damaging vehicle hardware

III. System Design

A. ROS

1) What is ROS?: ROS is an open-source robotics middle ware. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. ROS is a distributed framework of processes (aka Nodes) that enables executables to be individually designed and then coupled at run time. These processes can be grouped into packages, which can be easily shared and distributed. Additionally, the ROS community is very active, ensuring strong support for current and future packages. In terms of functionality, ROS supports a publish-subscribe model where a node publishes data to a topic from which an arbitrary number of nodes can receive (i.e. subscribe to) that data. ROS also supports one-to-one connections called services. In this form of interaction, a node is able to elicit an action from another node via a service call. The result of that action is received in a service response.

2) Why ROS?: ROS was chosen for this project due to its wide ranging community support for functionality required within this project. Namely, MavROS, the ROS package for creating MAVLINK messages (i.e. to communicate with the flight controller), and ROSBridge, a package for interacting with non-ROS programs (i.e. the interoperability server). Also, ROS provides support for easy sensor integration. This is essential for our use case due to the fact that we will require a rangefinder for the alphanumeric recognition and a 2D LiDAR for dynamic obstacle avoidance. The ROS paradigm of isolating specific tasks to their own nodes allows for an asynchronous development cycle where various nodes can be developed at different speeds and times. Lastly, ROS nodes can be developed in both Python and C++. Having this capability means that we can utilize the quick development cycles that Python provides while utilizing C++ for those nodes that require optimal performance. Each ROS distribution corresponds to an Ubuntu LTS distribution. We will be using ROS Noetic with Ubuntu 20.04. Allowing us to make use of Python3 and avoid the deprecated Python2 that is used in ROS Melodic (Ubuntu 18.04) and ROS Kinetic (Ubuntu 16.04).

B. Aircraft

1) Design Rationale and Process: For the fabrication and design of the aircraft, our team decided to use an iterative process based loosely around the system prescribed in Keane (Small Unmanned Fixed-wing Aircraft Design). We began our initial design process through the use of OpenVSP, a vehicle sketchpad. We chose to avoid VTOL and multi copter aircraft for the reasons described in the section regarding alternatives we considered. It was decided to use a twin tractor, twin tail boom configuration. The twin tail boom helps with material constraints improving the cost efficiency of the overall design. A twin tractor design adds redundancy in the case that one of the motors were to fail during the mission. A rounded rectangular cross section is used for the aircraft fuselage. The rounded rectangle design as opposed to the ellipsoid design allows for increased stability for the items inside the actual fuselage, and an easy way to drop the ground vehicle and payload once necessary. The spacious geometry of the fuselage allows us to comfortably house the mission payload (i.e. ground vehicle, mapping mechanism, and alphanumeric recognition system), avionics, and batteries. After settling on the aforementioned geometry, we compiled a comprehensive design brief. This design brief was then used to create a constraint diagram from which we could determine the necessary wing loading (kg/m^2)and thrust (kW) required of our vehicle. Using these values, we were able to size the initial iteration of the aircraft. These values will be used to construct a model in Solidworks that can be more extensively analyzed and modified accordingly.

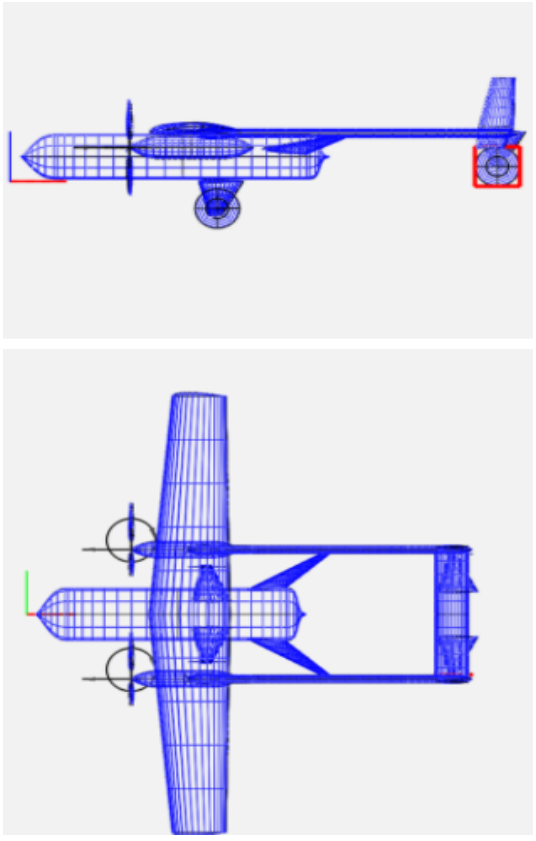

Fig. 2. This Figure illustrates the first iteration vehicle constructed

2) Manufacturing: From the aircraft CAD files the aircraft will be 3D printed by an external company. This method was found to be the most time efficient and cost effective. Molding aircraft parts was considered; however the tools, cost, and skills required for this were not feasible. For smaller parts (e.g. ribs) of the actual air frame, we will be using nylon. Carbon fiber spars and stainless steel will also be used to fortify parts that carry large portions of the load, such as engine bearers and

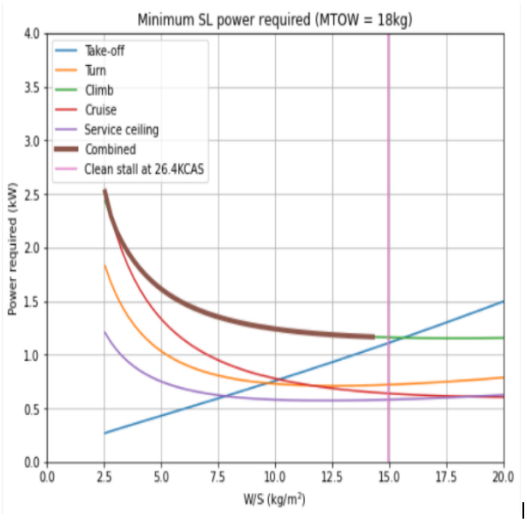

Fig. 3. This Figure illustrates the constraint diagram we constructed with the help of ADRpy.

the spars supporting the lifting surfaces. Once we have a CAD model, the design will be outsourced and manufactured and sent back to us. Our largest task is creating the comprehensive CAD model After the parts have arrived, we will:

- Use epoxy resin or bolted-in screws in order to form the overall air frame structure.
- Parts that may be removed should be attached with screws for easy removal.
- Parts that must be removed for storage should only have a couple screws to facilitate easy removal.

Parameter	Value	Units	Notes
Gross Weight	18	kg	
Power Produced	1.3	kW	1.3kW is the amount of power needed to produce the amount of thrust required to fly. Two Hacker A40 motors will provide this amount of thrust.
Wing Loading	14	kg/m^2	
Aspect Ratio	7	Unitless	
Wing Area	1.286	m^2	
Span	2.30	m	
Mean Chord	0.56	m	
Cruise Speed	19.5	m/s	

Fig. 4. This Figure illustrates the important design parameters characterizing our air frame (this will be more fleshed out in final paper)

3) Discussion of Airfoils Selected: Extensive research was done to figure out the appropriate airfoil for the twin tail boom design. In the end, it was decided to use the NACA 4415 which has the right amount of camber necessary for a great L/D ratio, so the airfoil itself allows for great efficiency. The NACA 4415 has been used on multiple twin tail boom designs including the AAI RQ-2, and AAI Shadow 200 and AAI Shadow 400. We also went through multiple tests for the best angle of attack for the main wing (will attach alpha vs Cl/Cd graph in final paper), and the best option hovered around 2 degrees. This design was used for both our main lifting surface and our horizontal stabilizer. As far as the vertical stabilizer, we went with the general fin design with no camber and symmetrical, as the main reason the vertical stabilizer is there is to prevent rollover. With this configuration we had a maximum L/D ratio of approximately 34.

Component	Airfoil Name	Alpha	Max Cl/Cd	Image
Main Lifting Surface	NACA 4415	2	34	
Horizontal Stabilizer	NACA 4415	0	34	
Vertical Stabilizers	NACA 0010	0	34	

Fig. 5. This Figure illustrates the airfoils used on various surfaces throughout our air frame

C. Autopilot

Autonomous flight is an essential aspect of the AUVSI SUAS competition. For this reason choosing an capable autopilot is crucial. For an autopilot to be capable of both meeting the competition requirements and fit into our team design plan it must meet the following criteria:

- Fly an autonomous mission reliably, safely, and accurately
- Capable of using return to launch as a fail safe
- Compatible with GPS units and radio telemetry necessary to communicate with the Interop server
- Communicate with a small single board computer on the UAV for active obstacle avoidance
- Interface with Robot Operating System paradigm used by our team
- Open source software with community support

Our team derived these requirements from the rules of the competition and funding limitations. From these requirements we were able to narrow our choices to PX4 and Ardupilot. Between these two options Ardupilot supported the features we required, included more documentation, and has a larger code base. Once an autopilot software was chosen a compatible flight control unit (FCU) could be found. From the list of supported FCUs in the Ardupilot documentation a few candidates were chosen based on functionality beyond Ardupilot support, redundancy, and cost. The following figure compares the final FCU selection.

The Pixhawk 2.1 (Cube) was chosen after reviewing these features as well as factors like prior experience and the accessibility of the units. The Pixhawk 2.1 was selected for the wide range of features, strong community

Name	Cost	Functionality/Interfaces	Sensors	Redundancy
Pixhawk 2.1 (Cube)	$250	Mixed autonomous manual backup mode Vibration isolation for 2 IMU sensors UART, CAN, DSM, I2C, SPI, PPM, RSSI input, microUSB port Versions at different price points with different sensors Carrier Board	Accelerometer, Gyroscope, Magnetometer, Barometer	3x IMU 2x Barometer 2x Power Supply Input
CUAV v5 Plus	$350	Mixed autonomous manual backup mode Vibration isolation for all sensors UART, CAN, DSM, I2C, SPI, PPM, RSSI input, microUSB port Carrier Board	Accelerometer, Gyroscope, Magnetometer, Barometer,	3x IMU 3x Power Input
PixHawk mRo	$190	Mixed autonomous manual backup mode UART, CAN, DSM, I2C, SPI, PPM, RSSI input, microUSB port	Accelerometer, Gyroscope, Compass, Barometer	2x IMU 2x Power Supply Input
BeagleBone Blue	$85	Runs Debian Linux Bluetooth Wifi UART, CAN, DSM, I2C, SPI, RSSI input, microUSB port	Accelerometer, Gyroscope, Barometer	None

Fig. 6. This Figure illustrates the various flight controllers considered for use in the project

support, redundancy in critical components, and team member experience. Multiple versions of the cube have been created allowing for the autopilot hardware specific to aerial and terrestrial applications. By using slightly different versions of the internal processing unit on different Pixhawk 2.1 models cost, energy, and weight can be saved. For example a terrestrial targeted version could be used for the ground vehicle that has less redundancy in sensors specific to flight. Additionally, the Pixhawk 2.1 has a recommended ecosystem which includes supported GPS modules, additional sensors, and telemetry links to a ground station ensuring compatibility.

The other autopilot systems were eliminated mainly due to limited redundancy or availability. Although the CUAV v5 plus contained many of the same features as the Pixhawk 2.1 some team members had prior experience with the pixhawk 2.1 as well as immediate access to units for SITL testing. The mRo Pixhawk also contained all of the necessary features but lacked the redundancy featured in the pixhawk 2.1. Additionally, the IMU sensors in the mRo version of the pixhawk are not isolated from vibration resulting in less accurate readings. The ability to run debian linux as well as built in wifi on the BeagleBone Blue allows for easier development but the lack of redundancy, community support, and failsafe features are the reasons it was eliminated as a viable option.

The Pixhawk 2.1 in the UAV will communicate with a ground control station through a telemetry radio connection. At the ground station an instance of APM Planner 2.0 will be used to interact with the UAV. This software was chosen as it offers cross compatibility between operating systems and is open source. By using an open source mission planning software our team will be able to build on tested features. Using the MavROS ROS package, our custom software can externally control the UAV through MAVProxy while benefiting from the reliability of a tested ground station software.

D. Obstacle Avoidance

1) General System Layout: The obstacle avoidance capabilities of the system can be divided into two sub-capabilities: dynamic obstacle avoidance and stationary obstacle avoidance. Both of these sub-capabilities will be implemented within their own ROS node. The StationaryObstacleAvoidance node will reside on the ground station computer and subscribe to topics from which it can receive the 12 mission waypoints and stationary obstacles. It will be responsible for computing an initial trajectory which avoids all stationary obstacles and satisfies all mission waypoints. The DynamicObstacleAvoidance node will reside onboard the aircraft and subscribe to topics from which it can receive data generated by the onboard 2D LiDAR sensor, the locations of other airborne teams, and the initial trajectory through the region produced by the StationaryObstacleAvoidance node. The StationaryObstacleAvoidance node will optimize a signal temporal logic (STL) specification in order to generate the trajectory through the region. The DynamicObstacleAvoidance node will also utilize STL in order to generate a sub-trajectory around the nearby obstacle. The set of waypoints going around the local dynamic obstacle are inserted into the master trajectory generated from the StationaryObstacleAvoidance and uploaded to the flight controller via MAVROS functionality (i.e. a service call).

2) STL Overview: STL can be described as a type of time dependent Boolean logic. Unlike in Boolean logic, STL formulas are qualified using a time domain. That said, STL is defined over a signal y(t) which is continuous over the entire time domain. In addition to the Boolean logical operators of NOT, AND, and OR, STL possesses time dependent ALWAYS and EVENTUALLY operators. These various operators can be used to define predicates. One can then combine predicates to create specifications that can be further combined to create one global specification describing the desired behavior of a system. This is particularly useful because one can define this global specification in terms of Robust Semantics. In doing so, a given signal no longer discretely satisfies or does not satisfy the specification. Now, there is a quantitative indication of how well a signal satisfies the given specification. And so, an optimization problem can be defined where the goal is to find the signal that best satisfies the robustness function. In the context of the StationaryObstacleNode, this is particularly useful as one can create a cost function which can be minimized to find the optimal route through the region. STL formulations have several advantages over other approaches. A particularly notable example includes the ability to specify complex combinations of time dependent actions that would be otherwise difficult to implement. Just as architects construct complex designs out of simple building blocks, STL allows for the construction of complex specifications from simple predicates. For the sake of simplicity and the ability to generate a globally optimal solution using mixed integer programming (MIP), we will be using rectangular predicates (i.e. made up of linear functions). This means that all of our waypoints and obstacles will take the form of rectangles in our STL specification. This specification will then be optimized through the use of GurobiPY.

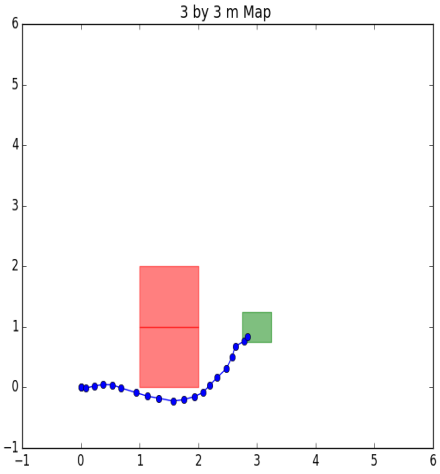

Fig. 7. This Figure illustrates the trajectory generated by optimizing a STL specification consisting of a single obstacle and goal region

E. Imaging System

In order to automate the process of imaging for both the mapping and alphanumeric recognition a method of controlling the cameras through the UAV's onboard companion computer needed to be determined. Due to the differences in camera requirements between the mapping portion of the competition and the alphanumeric identification the imaging system will comprise two cameras. A Canon PowerShot SX620 HS was chosen for the mapping system and a NDI camera capable of both video streaming through a wired connection and taking photos for the alphanumeric recognition system.

Necessary features ALPHA:

- Offload pictures (NDI) for AI image processing
- Take pictures remotely and precisely for mapping
- Scripting (Could be base on gps, obstacles, or others)
- Ability to shoot in RAW and other locked formats (Better for AI sometimes)

Necessary features Mapping:

- Take pictures remotely and precisely for mapping
- Scripting (Could be base on gps, obstacles, or others)
- Interface with Ardupilot (CHDK)

The industry standard for airplane tail numbers is one foot lettering. Using this standard for alphanumeric recognition 12 pixels per foot are needed to recognize the characters. In ideal conditions the goal of the alpha numeric systems is to have 12 pixels per foot of ground area at max competition elevation.

F. Object Detection, Localization, Classification

Alphanumerics are identified as separate from the ground area while towards the edge of the field of view. Using the angles to the character, altitude of the UAV, pitch, roll, and GPS coordinates a triangulated coordinate for the potential alphanumeric will be recorded. After the way points have been traversed the located objects will be used in an STL instance. An efficient path over the previously identified objects will be developed and high resolution images taken for further processing. This trajectory will allow the UAV to detect characters with higher confidence without compromising the waypoint mission. Using an optimized path to re image previously identified objects will help prevent missed alphanumerics. This solution allows for more of the competition area to be effectively imaged all without deviating from the waypoint mission.

1) Object Detection: Due to the processing time of large format images the object detection will take part in two steps. A live video from the alphanumeric camera will be reviewed by a computer vision program. From this video the characters can be separated from the ground. Once a character is found a high resolution image will be taken allowing for more advanced processing. Additionally, the live video processing will be trained to see anomalies that may be characters but are at too steep of an angle to currently identify. These object locations will be saved and an STL instance will be used to determine an efficient path between them for re imaging. The computer vision used to identify characters and anomalies will use the open source OpenCV python library to develop the program.

2) Classification: An AI program will give an alphanumeric character along with a confidence score. Characters with low confidence scores will be either marked for another flyover (given a weight and STL used) or if that is not feasible within the mission timeframe an image will be transmitted to the ground station for Human review. The classification portion of the object detection will also use the OpenCV library.

3) Localization: Using a LiDAR sensor with a fixed mount, and the pitch data from the UAV an accurate altitude can be calculated. Then using this altitude and the derived angle between the camera and the object a distance vector between them can be calculated. This distance vector along with the GPS coordinates of the UAV allow for a GPS location of the object to be determined.

G. Mapping

In order to generate a high-quality map, the system demands several key features. The ideal solution would begin with a versatile camera system that allows for dynamic adjustments to the aperture, ISO, and shutter speed to properly expose the images collected. A fast focal ratio as well as a global shutter would be preferred to preserve the fine details within the image. That being said, there are several constraints that need to be considered. The system must be lightweight, able to interface with Ardupilot, and affordable. While there are a myriad of high quality imaging systems out there suitable for the job, most come with a hefty price tag. With this in mind, the Canon Powershot SX620 HS was chosen as it is able to

satisfy all the constraints and provide reasonably high-quality data. The camera is able to interface with Ardupilot via the Canon Hacker Development Kit (CHDK).

The CHDK platform allows for autonomous control of the camera, allowing for in flight adjustments of the focal length, exposure, and intervalometer if needed. The captured images can be offloaded via the CHDK Photo Transfer Protocol extension (PTP). While the PTP extension is functional, it may prove to be a bottleneck in the system as its file transfer times are rather slow for the large raw files. Alternatives are currently being explored.

The next choice to be made concerns that of the map making software. OpenDroneMap (ODM) was chosen due to its affordability without a loss in performance. Our team is considering multiple options regarding where the stitching of the orthophoto will occur.

- The most streamline solution would be to run ODM on-board the aircraft, however it may prove too resource intensive to execute given the baseline system requirements call for 4 GB RAM and 20 GB storage minimum to process around 100 images. Given we anticipate upwards of 350 images, a fast CPU, 16 GB RAM and 100 GB Storage would be an ideal setup for our needs. One system that shows promise is the UDOO single-board style computers with quad core AMD processor, and extendable storage and memory that would adequately fit our needs.

- The next option would be to process our data in the cloud. ODM has an application CloudODM that allows us to run ODM in the cloud via the NodeODM API. This is currently an attractive solution if we can confirm it has a reasonable run time while maintaining the files' integrity.

- The last option would be to offload the files after landing and process them on the ground through a vanilla instance of ODM. While this is a simple, straightforward solution, our team, by competition rules, would only have 10 minutes to assemble and send off the map for assessment.

The current course of action is to get well acquainted with running ODM on the ground, as discussed in our last solution, and experiment with different CPU, memory, and storage combinations to see if on-board stitching is viable.

H. Communication

1) Communication Links: For our manual receiver link, we elected to use the TBS (Team Black Sheep) Crossfire Mini. The TBS Crossfire Mini has a range of up to 40 kilometers, more than enough for 10 kilometer range our team determined was necessary. As well, the TBS crossfire Mini interfaces well with ArduPilot, our flight software. The TBS crossfire also has frequency hopping built in, and communicates on the frequency band 915 MHz. This allows not only for us to not worry about interference on the ground, but anywhere close by. Additionally, immunity to onboard noise as well as ultra low latency and a 150 Hz update rate. It also has bandwidth control and range optimization. As far as the links between the actual UAV and the ground station, we have the TBS crossfire mini for manual control, and a wireless controller linked. As previously stated, this is approximately 915 MHz. The second connection to the UAV is the laptop running the ground communication software is the RFD 900+ Telemetry radio. This connects the ground station laptop to the actual drone. Its communication frequency is around 915 MHz as well. Its range is about 902 MHz to 928 MHz. The third link is an LTE link. Our whole network runs on one VPN, so we can link the aircraft IP address to the ground station IP address. This also allows all the ROS topics to communicate with each other because they will see each other on the same VPN. The VPN server will be run in the cloud. We will presumably use AWS as our cloud provider and Wireguard as our VPN due to its easy accessibility on Ubuntu. The frequency of this LTE link hovers around 700 MHz. We chose to use LTE as opposed to wifi so we do not have to take on the complexity involved with setting up directional antennas. In addition, we checked the competition map location, which is in a very good service area, negating the necessity of devoting time and effort to constructing a Wifi signal.

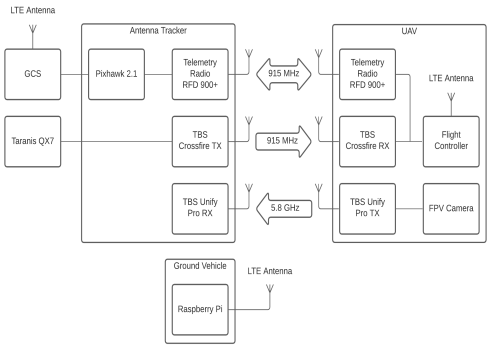

Fig. 8. This Figure illustrates the communication links to be implemented between each component within our system

2) Antenna Tracking Rationale: To maximize the range of the wireless communication channels, each transmitter and receiver connected to the ground station will use directional antennas. The antennas will be mounted on an automatic antenna tracking device which will continuously keep the antennas aligned with the UAV. One servo below the tracker will turn to point it in the right compass direction and a second servo on the side will control its elevation angle. Because the location of the ground station is in the center of the flight zone, the first servo must have 360 degrees of rotational freedom. A continuous rotation servo will be used to meet this constraint. The elevation servo requires 90 degrees of rotation to allow the antennas to point anywhere between horizontal and vertical. A standard high-torque servo will be used to adjust the elevation. Both servos and the telemetry radio transceiver connect to a Pixhawk flight controller on board the antenna tracker, which will be loaded with Ardupilot's AntennaTracker software.

Using its own GPS and orientation data, as well as the GPS data from the Pixhawk on board the UAV, it will calculate the relative position of the UAV and move the tracker to point in its direction.

I. Air Drop

1) Unmanned Ground Vehicle: The following components will make up the ground vehicle:

- The Ground Vehicle consists of various electronic components. For the controller, the UGV will use a Pixhawk 2 which will relay signals back and forth from the ground stations which will allow for a direct and stable connection to the Ground Vehicle. The Pixhawk 2 will also allow for the automation of the UGV and will direct it to destination to drop off the payload.
- To power the DC motors for the wheels, electric speed controllers (ESC) will be used. The ESC's will be connected from the Pixhawk 2 to the DC motors to power the motors and allow the UGV to move. This will also keep the UGV from exceeding ten miles per hour, or the maximum speed allowed for the UGV.
- A LiPo battery will be used in the UGV to power up the entire system by allowing power to go to the Pixhawk 2.
- Companion Computer. The Companion Computer will allow for our systems to interface and communicate with ArduPilot on a flight controller. Using MAVLINK, the UGV will be able to move autonomously. Thus, the UGV will be able to drive to the payload drop off destination by communicating between the hardware and the software.
- Alongside the Companion Computer is the ROS node that the UGV will incorporate. This will allow the UGV to communicate with the UAV as well as the ground control from a software perspective.
- Lastly, an LTE dongle will also be used to allow for communication between the different parts of the competition. It will allow for communication similar to the ROS node; however, it is the hardware aspect of the communication between all aspects.

The following describes the iterative design for the ground vehicle:

- Initially, the design for the Ground vehicle was going to consists of only two wheels to optimize the size of the UGV; however, once drafting began, an issue arose with the center of gravity of the vehicle. With the components that the UGV consists of, the two-wheel iteration would prove to be a difficult task especially with the given constraints. Although this design allowed for the UGV to be lighter, the overall difficulties resulted in approaching the UGV a different way.
- The second iteration consisted of using two different wheel sizes and making the UGV a two-wheel drive. This design would allow for the body of the UGV to be at an angle which would allow for the payload to slide of the frame. Some issues with this design is the practicality of using two different tire sizes which results in the UGV being a two-wheel drive instead of an all-wheel drive. Also, some problems could arise with two different tire size and could cause different issues throughout the process.
- The final iteration, shown above or below, is a all-wheel drive vehicle with an elevated portion on the back en of the UGV. Like the second iteration, an angled surface is ideal in order to allow the payload to easily slide off the frame. This iteration will allow for an all-wheel drive which can allow for the payload to reach its destination quicker while still being able to maintain its speed below ten miles per hour.

The following describes how the water bottle payload will interface with the UGV:

- By having the UGV frame at an angle, it will allow for the payload to slide off when it reaches the destination. A servo will be used to secure the payload along with a frame that will allow the payload to be secured throughout the process. When the UGV reaches the destination, the servo will then change positions which will open the frame and allow the payload to slide from the frame. The UGV will reverse once the servo opens and will allow the payload to completely drop from the frame.

2) Drop Mechanism: The air drop system consists of two parts: the UGV (Unmanned Ground Vehicle) and the release mechanism.

The release mechanism consists of the cargo bay area, the payload release module, and the trap door.

The cargo bay design fits the shape of the UGV to prevent excess movement while the plane is in the air. The payload release module will be located at the top of the cargo bay and the trap door will be located underneath the UGV. The payload release module will carry most of the weight of the UGV, while the trap door will not be under normal force.

The payload release module we decided to go with is the servoless payload release from E-flite. This release module is a lightweight and compact mechanism, which will be easy to install inside the cargo bay.

The trap door will consist of a solid plank of [insert material] which will be attached to the main cargo bay by a torsion spring. The spring torque will be just enough to hold the trap door of the plane shut, but weak enough to let the UGV push through when the release module is activated.

Operation of release mechanism: To simplify the calculation of the ODT (Optimal Drop Time) and ODD (Optimal Drop Distance), from the ground target, we will keep the UAV at a

constant speed of [insert] and at a constant cruise altitude of 250 ft. Given the assumptions of no drag, no wind, and UGV in freefall, the drop time is:

$$ODT = \sqrt{\frac{2h}{g}}$$

Where h is the cruise altitude and g is the gravitational acceleration. The drop distance is:

$$ODD = v\sqrt{\frac{2h}{g}}$$

Where v is the constant speed of the UAV. Once we test our UAV dropping mechanism, we will modify the ODT and the ODD to account for the drag and the slow descent of the UGV.

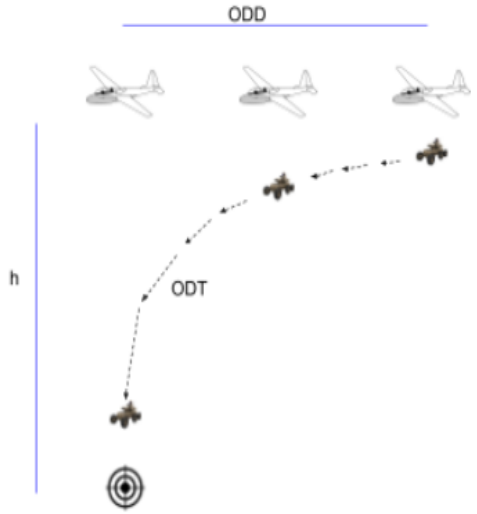

Fig. 9. This Figure illustrates the ODT and ODD of the UGV

IV. Alternatives Considered

Initially, our design team considered a few different options. These included VTOL and multicopter aircraft. A VTOL design was not selected due to the potential mechanical and firmware complexity. Likewise, the maximum takeoff weight would be impacted substantially due to the vertical takeoff and landing. In addition, the multicopter was not selected due to endurance concerns regarding an inability to complete the 6 mile waypoint string and then continue on to successfully attempt the alphanumeric recognition, UGV and mapping tasks. Likewise, multicopters are prone to crashes when experiencing motor problems. As such, it was decided that a properly design fixed wing would better satisfy the mission requirements. Another point of contention was that of the propulsion. Initially, we were considering using either petrol or electric motors. The petrol motor's increased complexity was justified through the substantial increase in range due to petrol's much higher energy density than LiPo batteries. That said, after examining the mission requirements once more, it became apparent that we would not be required to fly for any longer than 30 minutes. Keeping that in mind, the relatively high efficiency and simplicity of brushless motors justified their use in the place of petrol motors. (in the final paper we will also discuss PX4 vs ArduPilot here–we will also discuss the various frames we debated using in the initial brainstorming we did using OpenVSP)

V. Testing and Evaluation Plan

A. Developmental Testing

In regards to testing the individual nodes within our network, ROS makes doing so very convenient due to its support of the rostest package. This package makes writing and launching test nodes within a system very easy to integrate. Generally speaking, we will be pushing spoofed sensor data into the nodes within our network that we would like to test and observing the output. Observing the results of this fudged sensor data will be safe and effective due to the extensive and well integrated support for Gazebo within ROS. Within Gazebo, we will be able to construct a custom object that will mimic the dynamics of our airframe. Likewise, we will be able to model a sample mission grid with a search area, series of waypoints, drop zone, and mapping region. Nodes requiring imaging sensor input will also be easily tested seeing that Gazebo allows us to pipe image output from a defined camera view into an image topic to which we can subscribe. After we have verified the correct functioning of the ROS nodes under spoofed data, we will be able to move onto the simulator data from Gazebo discussed above. This will be done by wiring up a MAVROS node to a Software-In-The-Loop (SITL) instance of the flight controller unit (FCU) running on the test computer. This will be followed up by a Hardware-In-The-Loop (HITL) test using a real Pixhawk 2.1 Cube to interface with the MAVROS node.

The focus on distribution of computing processes and the system integration that ROS allows for means that each portion of the system can be tested independently. As each process is developed in parallel with only the overarching requirements predefined each system can be individually optimized. For example the dynamics of the air frame can be tested before the pathfinding algorithm is ready to be used in flight. As development progresses each system can independently be tested in a SITL situation, then HITL, and with further physical system integration until the complete system is tested together. By testing the parts of the system in this order each can be evaluated to meet the derived requirements of that system in the specific version testing implemented. Once capable of meeting these requirements in simulation the system can progress to the next phase in testing.

1) Mission Testing: In order to fully test for mission accuracy, the team is going to test a few key features on the fully integrated system within the airframe. In order to test the durability of the ground vehicle, what we are going to do is have a constant speed test and make sure that the payload lands correctly. After that, we will spoof waypoints around a large uninhabited area, in order to see if the UAV can autonomously follow a set path. In order to accomplish this, the team will

System	Components	Testing Plan
Aircraft	1. Airframe 2. Propulsion System 3. Onboard Control System	1. Load testing of structural components will be done to ensure airworthiness. 2. A static thrust test will be done to ensure that the thrust values used in design of the airframe correspond realistically. 3. Gazebo simulator tests mirroring the operational environment to determine whether outputted trajectories and controls are as expected.
Communication Links	1. Verizon Cellular (LTE Sim Card) 2. Telemetry Radio (RFD 900x) 3. Manual RC Link (TBS Crossfire Mini) 4. Antenna Tracking Device	1. A connection aerial test will be done in the Indiana countryside to ensure reasonable latency and test environment similar to that of the competition location (2 Mile) 2. WNDU to Hesburgh Library Communication Test (1 mile) 3. WNDU to Hesburgh Library Communication Test (1 Mile) 4. Tracking test done with subscale airframe and proper autopilot (2 miles).
Airdrop	1. E-Flite Drop Mechanism and associated nodes 2. UGV, UGV controller nodes, and CDS (controlled descent system)	1. Manual release test with size and weight analogue standin for UGV module. 2. Durability drop test from 50, 100, 250 ft. from static altitude (i.e. using multicopter in discreet location).
Mapping	1. Collection Node and Processing Node. (OpenCV)	1. Flight test with software integrated on the plane. Test regional mapping.
Alphanumeric Recognition System	1. Imaging System 2. OCDL Nodes	1. Flight test with camera and integrated software. 2. Flight test with camera and integrated software.
Autopilot	1. ArduPilot Software	1. Software Simulation Test with manually selected waypoints to determine UAV ability to accurately and autonomously determine continuous path.

Fig. 10. This Figure illustrates first iteration design that was considered in OpenVSP

be using a large farm in Michigan that has enough space for waypoints that are far apart from each other. This will give us the option of making sure the UAV has enough waypoints which allow us to measure mission accuracy. As well, we will use the airspeed sensors to make sure that the speed stays below 25 m/s during cruise and within a reasonable range (12-18 m/s) for both takeoff and landing.

VI. Safety Risks and Mitigation

Risk	Mitigation
Use of Power Tools when building the aircraft	•Make sure everyone wears proper eyewear and follow correct lab protocols
	•Make sure that anyone who uses a power tool has the correct clearance through the university's student fabrication lab
Battery Storage	•Safely store LiPo battery in a fireproof bag in room temperature
	•Never charge over 4.2 Volts per cell or discharge below 3.0 V
Testing Motors before the flight	•Make sure all hands are away from propellor blades when testing
	•Remove battery after every test and charge battery accordingly

Fig. 11. This figure outlines the risks and mitigation of UAV development in the lab

Risks	Description	Likelihood	Mitigation
Mid-air collisions with another UAV or a manned aircraft	•In AMA certified areas, collisions between aircraft is possible	•Medium	•Ensure a Visual Line of Sight (VLOS) during the entire flight
			•If the UAV is in autonomous mode, ensure a clear path through the waypoints
			•If possible, test in a private airfield to avoid any other aircrafts
Loss of Control of UAV	•Signal between ground station and UAV is lost	•Low	•Enable Flight Termination System to minimize any damages
			•Test connections before flight and ensure that there is a stable connection before flying
	•Auto-pilot fails to maintain correct waypoint path	•Medium	•Revert to manual control in order to safely land the UAV
Weather	•Weather is a prevalent issue in northern Indiana	•Low	•Check METAR and TAF reports before each flight and identify any potential hazards
			•If precipitation begins to fall, safely and quickly land UAV and avoid any damages to the battery
			•Follow all guidelines regarding weather under FAA's part 107
Loss of power in any component	•Loss of Power in the UAV	•Low	•Ensure that Li-Po battery is fully charged and in good condition before every flight to avoid any loss of power within the UAV
			•Take into consideration the cold weather when flying which could cause voltage drops in the battery
			•If power is lost, ensure that UAV enters FTS and Fail-Safe routine system
	• Loss of Power in Ground Control	•Low	•Follow similar protocols regarding battery maintenance to ensure that the battery is in good condition
			•If power is lost in the ground control, ensure that the UAV enters FTS and Fail-safe routine

Fig. 12. This figure outlines the risks and mitigation of UAV flight

References

[1] András Sóbester Andrew J. Keane and James P. Scanlan. *Small Unmanned Fixed-wing Aircraft Design: A Practical Approach*. John Wiley Sons Ltd, 2017. ISBN: 9781119406297.

[2] Calin Belta and Sadra Sadraddini. "Formal Methods for Control Synthesis: An Optimization Perspective". In: *Annual Review of Control, Robotics, and Autonomous Systems* 2.1 (2019), pp. 115–140. DOI: 10.1146/annurev-control-053018-023717. eprint: https://doi.org/10.1146/annurev-control-053018-023717. URL: https://doi.org/10.1146/annurev-control-053018-023717.