EmberEye

Subcanopy Wildfire Monitoring Solution

Team B: Final Report

Kavin Ravie
Ishir Roongta
Shashwat Chawla
Jaskaran Singh Sodhi
Sai Gangadhar Nageswar

Sponsors: Sebastian Scherer, Andrew Jong

December 11th, 2024

Abstract

With a surge in wild-land fires across the United States, rising from approximately 50,477 fires in 2019 to as high as 68,899 fires in 2022, understanding and managing these disasters are imperative [5]. Front-line firefighters confront a hostile and chaotic environment characterized by dense hot air, obscured vision, and dynamic fire behavior. To ensure personnel safety and effectively mitigate fire spread, spatial information about the fires is essential for predicting their trajectory and planning safe entry and exit routes. However, existing high-altitude solutions cannot provide detailed insights into the ever-changing scene beneath the tree line. Addressing this gap, EmberEye offers an autonomous aerial solution equipped with a sensor suite designed to navigate through sub-canopy environments and provide crucial information to firefighters. By generating fire maps and serving as aerial scouts, EmberEye enhances situational awareness, enabling firefighters to make informed decisions and safely manage wildfire incidents.

Contents

1 Project Description

With the increase in wild-land fires and acres demolished over the years from approximately 50,477 fires in 2019 to up to 68,899 fires in 2022 in the United States, [5] it is vital to understand these catastrophes in order to control the devastation. The firefighters tasked with this control face a hostile and chaotic environment in which they have to safely maneuver and perform their duties. Thick hot air, towering trees with no light, and a dynamic blazing fire are what awaits them every time.

In such an environment to ensure the safety of personnel and efficiently control the spread the front-line responders need spatial information about the fires to predict the spread and plan entry-exit routes. With entrapment being a leading cause of fatalities, there is a dire need for granular situational awareness in such a chaotic environment. Existing high-altitude solutions provide some degree of information about the wildfires but cannot provide a view of the internal ever-changing scene beneath the tree line.

EmberEye is an autonomous aerial solution equipped with a sensor suite to navigate through this cluttered scene and provide meaningful information to firefighters. Using this information the fire-fighters can plan their routes, avoid entrapment, and safely control the spread. EmberEye can assist by providing a heat map of the area, and act as eyes in the sky by performing the dangerous task of monitoring and scouting.

Figure 1: Depiction of a Wildfire Scene

2 Use Case

It is a bright summer day. A man is driving out to the countryside, with lush green forests around him. Suddenly he notices some smoke coming from the canopy. He isn't sure about the source of the smoke, but it seems dense enough to be a fire. He rings the fire department to tip this potential fire.

On the other side, using the location of the caller, the fire department can extract recent (but temporally inaccurate) satellite imagery and get a coarse estimate of the GPS location of the fire source. This is something we expect from the results achieved by the MRSD Team Firefly [1] where they deployed a over-canopy solution.

Using this estimate, they can deploy their firefighting unit, equipped with a state-of-the-art sub-canopy wildfire monitoring solution "Phoenix". This drone is meant to act as the eye in the sky for the firefighters, giving them real-time feedback about their environment, and helping them plan their routes and action plans more efficiently.

The drone flies in the vicinity and transmits real-time Thermal, RGB, 3D Scene Structure, etc. back to the Ground Control Station (GCS), which helps the firefighters make informed decisions about the current fire situation as well as secondary hotspots. These real-time actionable insights from the UAS act as a game-changer for the firefighting unit.

The drone finally lands, and along with the GCS, is able to help mitigate the wildfire efficiently.

Figure 2: Depiction of the Use Case

3 System Level Requirements

The team derived their system-level requirements from the use case and objective tree, illustrated in Figure 30. Specifically, performance requirements were deduced from functional requirements, each categorized into mandatory and desirable segments. Stakeholder feedback and potential customer input were also integral in refining these performance requirements, which remain subject to evolution throughout the project's advancement.

The entirety of the system-level requirements has been documented within the subsequent tables. Table 1 provides an overview of the mandatory functional requirements of our system, complemented by detailed descriptions for each requirement. Correspondingly, Table 2 elaborates on the mandatory nonfunctional requirements. Additionally, a set of desirable requirements has been formulated, outlined in Table 3 for functional aspects and Table 4 for non-functional aspects.

Table 1: Mandatory Functional and Performance Requirements

Functional	Performance	Description
M.F.1 Sense Environment	**M.P.1** Localize itself at least 10Hz.	The aerial platform should be able to perceive its surrounding environment with its equipped sensors.
M.F.2 Localize itself	**M.P.2.1** Localize itself at least 10Hz. **M.P.2.2** Localize itself within a drift of 4%	
M.F.3 Map Environment	**M.P.3** Localise itself within a drift of 4%	Aerial platform should be able to map the environment of its operation.
M.F.4 Plan safe trajectories towards goal	**M.P.4** Navigate trees with separation >=5m	Aerial platform should be able to navigate through subcanopy environment
M.F.5 Be capable of completing the mission	**M.P.5** Have a flight time of more than 5 minutes	The aerial platform should be able to complete the mission under its flight time
M.F.6 Detect fire	**M.P.6** Detect fires within the accuracy of >=70%(fire vs non-fire frames)	The fire perception module should be able to accurately detect the fire during operation
M.F.7 Localize fire	**M.P.7** Localize fire positions up to 3m of distance in front of the drone, with a 2.5m error threshold	During the operation, the aerial platform should accurately localize the location of the fire
M.F.8 Communicate with user	**M.P.8** Have a communication range up to 150m	The user should be able to visualize the fire map and telemetry information on the GCS up to the set range.
M.F.9 Be teleoperable	**M.P.9** Have a communication range up to 150m	The user should be able to teleoperate the aerial platform when required.
M.F.10 Operate in GPS degraded forest environment	**M.P.10** Localize itself at 10 Hz	The aerial platform is capable of operating in an environment where we are not relying on the GPS.

Table 2: Mandatory Non Functional Requirements

Requirements	
M.N.1	Have appropriate dimensions/size
M.N.2	Have failsafe and redundancies
M.N.3	Be of Rugged Design
M.N.4	Rely on Passive Sensors Only
M.N.5	Be Easy to Use

Table 3: Desirable Functional Requirements

Functional	Performance
D.F.1 Plan Safe Trajectory Towards Goal	**D.P.1** Navigate autonomously between trees at minimum 1m/s
D.F.2 Operate in GPS degraded forest environment.	**D.P.2** Localize itself in GPS degraded environment with less than 0.05Hz of GPS connectivity

Table 4: Desirable Non-Functional Requirements

Requirements	
D.N.1	Be FAA Compliant

4 Functional Architecture

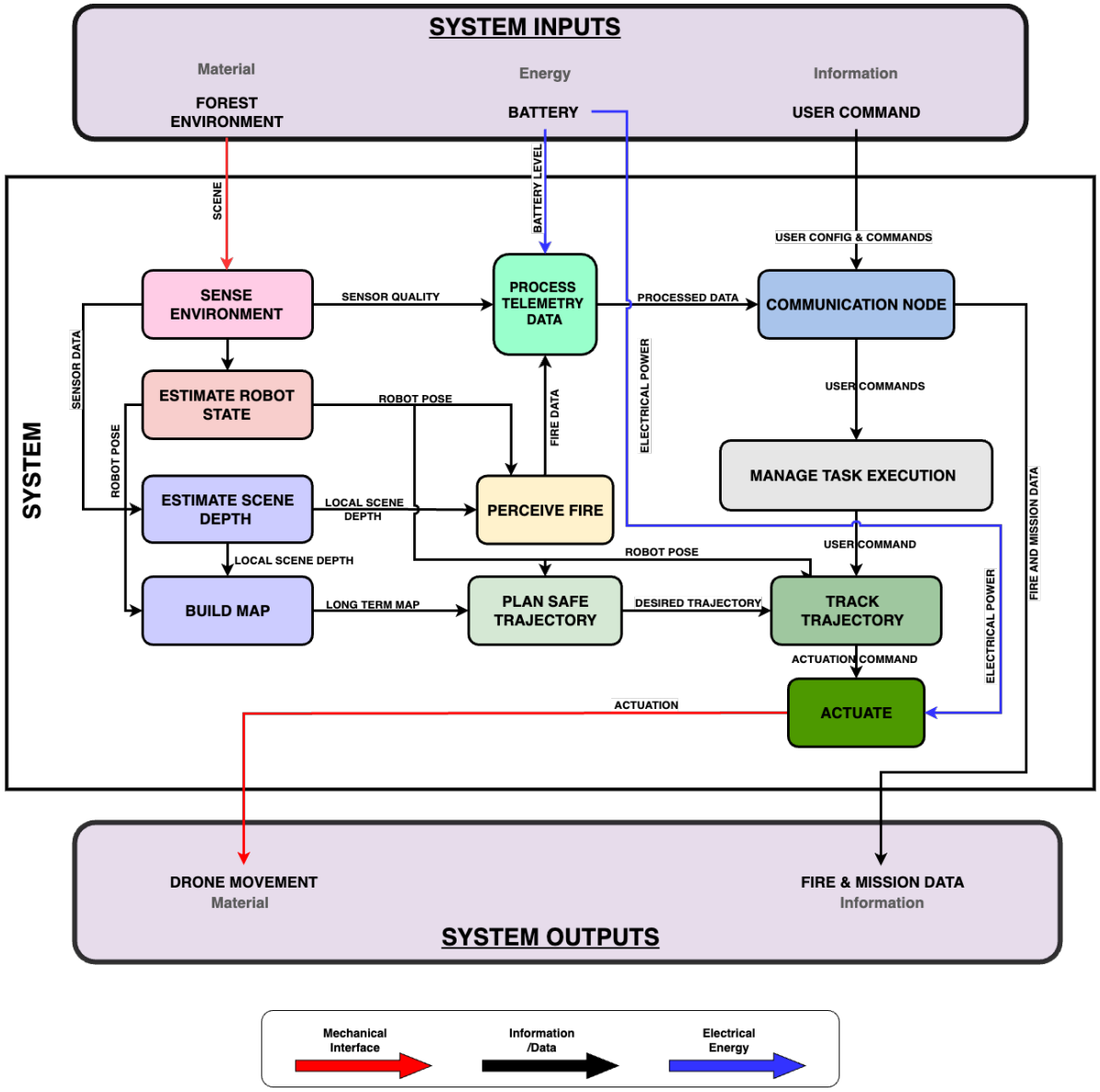


Figure 3: Functional Architecture

Figure 3 shows the functional architecture of our system. The inputs to the system are categorized broadly into Material/Mechanical input, energy into the system, and information input. The material input for our system is the wildfire environment the drone is traversing which is perceived by the on-board sensor suite. The electrical energy from the battery is utilized by the computation, communication, and actuation modules mounted on the drone's airframe. The only information input to the system is the GPS location of the estimated wildfire location. The system transmits mainly three data outputs, fire heat map, wildfire primary hotspot location, and the drone's telemetry data. Summing up, the system takes the GPS coordinates of the rough estimate of wildfire location and transmits the fire heatmap, primary hotspot location, and telemetry data.

5 Trade Studies

5.1 System Trade Study

Figure 3 shows the system-level trade study conducted between different mobility modes. The three main modalities of robots we have considered for our trade studies are legged, mobile, and aerial.

The objective of this trade study is to choose the best robotic platform for our application. The criteria for evaluation against which each modality is evaluated are Design/Manufacturing complexity, agility, recoverability, information gathering, size, payload, and endurance. The highest weightage is given information-gathering criterion which is the most crucial deliverable of our project. The second highest weightage is given to agility which is crucial for dynamic environments such as wildfires in forests, recoverability since these robotic systems are expensive and their retrieval is important for multiple missions, and design complexity which comes from the small time-frame of the MRSD project. Based on the analysis, we have concluded that the aerial robotic platform is the most suitable for our application. Although an aerial platform has lower endurance which is quite important for the operation, the other factors such as agility, information gathering, and design criteria were in favor.

Trade Study	Under-canopy Robot Mobility Modes			
		Legged	Wheeled	Aerial (Multi-rotors)
	Mobility Modes			
Value Ratings *				
1: Inadequete				
2: Tolerable				
3: Adequete				
4: Good				
5: Excellent				
Criteria	**Weight Factor (100%)**			
Design/MFG Complexity	15	5	3	4
Agility	15	5	3	5
Recoverability	15	2	2	5
Information Gathering	30	4	3	5
Size	10	4	3	5
Payload	5	5	5	3
Endurance	10	5	5	3
Weighted Average	5	**4.15**	**3.15**	**4.55**
**** Subjective Value Method***				

Figure 4: System-level Trade Study

5.2 Multi-Rotor Trade Study

The system-level trade studies have provided insights into the modality of the robotic platform to be used. To further refine our choice of mobility options, within aerial solutions, we conduct a trade study between different multi-rotor configurations as shown in Figure 3.

The criteria considered for the evaluation of the different configurations are manufacturing complexity, structural weight, perception suite flexibility, size, payload, aerodynamic efficiency, redundancies, and cost. The most important criteria among these are the all-up-weight which limits the weight of the components on board, the cost factor limited by the sponsor-provided budget and MRSD budget, perception suite flexibility which is essential for our highly multi-sensor platform, and design complexity. Performing trade study among these configurations has provided the quadcopter configuration to be the best. Although size and payload are in favor of coaxial quadcopter configuration, the all-up weight and design complexity criteria were in favor of the quadcopter design.

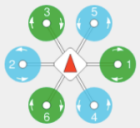
Trade Study	Multirotor AirFrame Configurations			
	Configurations	 Quadrotor	 Hexacopter	 Coaxial Octacopter
Value Ratings *				
1: Inadequete				
2: Tolerable				
3: Adequete				
4: Good				
5: Excellent				
Criteria	**Weight Factor (100%)**			
Design/MFG Complexity	30	5	4	3
Structural Weight	20	5	4	3
Perception Suite Flexibility	10	5	3	5
Size	10	3	4	5
Payload	5	2	4	5
Aerodynamic Efficiency	10	5	5	3
Redundancies	10	3	4	5
Cost	5	5	4	3
Weighted Average	5	**4.45**	**4.00**	**3.70**
**** Subjective Value Method***				

Figure 5: Multi-Rotor Trade Study

5.3 Propulsion Trade Study

We also conduct a trade study between different propulsion mechanisms as shown in Figure 3. The objective of this trade study is to decide the most optimal propulsion system hardware out of the most commonly used options. The different options are evaluated against availability since most of the desirable components may not be available, endurance which should meet our performance requirement MP3, all-up-weight which limits the weight of the components on-board, size of the drone restricted by the dimensions of the inter-tree distances, payload, and agility. Refer to Figure 3 for the components considered. The analysis has resulted in Config 2.2 being the best option for us, i.e. using KDE4215XF-465 BLDC motor, 6S 10000 mAh Li-Po Battery, and 15.5”X5.3” Bi-Blade CF propellers. Note that the choice of battery and propellers is limited by the choice of motor.

Trade Study	Quadrotor Propulsion Configurations				
	Configurations	**Quadrotor Config 1.1**	**Quadrotor Config 1.2**	**Quadrotor Config 2.1**	**Quadrotor Config 2.2**
	Specs				
	BLDC Motor	KDE XF2814 KV775	KDE XF2814 KV775	KDE4215XF-465	KDE4215XF-465
	Li-Po Battery	4S 9000 mAh	4S 12000 mAh	6S 7000 mAh	6S 10000 mAh
	Propeller	12" X 4" Bi-Blade CF	12" X 4" Bi-Blade CF	15.5" X 5.3" Bi-Blade CF	15.5" X 5.3" Bi-Blade CF
	Endurance	16 Min	20.5 Min	20 Min	25.5 Min
	AUW	3850g	3965g	4700g	5000g
	Airframe Span	570mm	570mm	650mm	650mm
	Payload	865g	750g	2500g	3000g
	Max. TWR	1.73	1.73	2.78	2.78
Value Ratings *					
1: Inadequete					
2: Tolerable					
3: Adequete					
4: Good					
5: Excellent					
Criteria	**Weight Factor (100%)**				
Availability	20	5	3	3	5
Endurance	30	3	4	4	5
All-up weight	20	5	5	4	4
Size	15	5	5	4	4
Payload	15	2	2	5	5
Agility (max. TWR)	10	2	2	4	4
Weighted Average	5	**3.77**	**3.68**	**3.95**	**4.59**
**** Subjective Value Method***					

Figure 6: Propulsion Trade Study

6 Cyberphysical Architecture

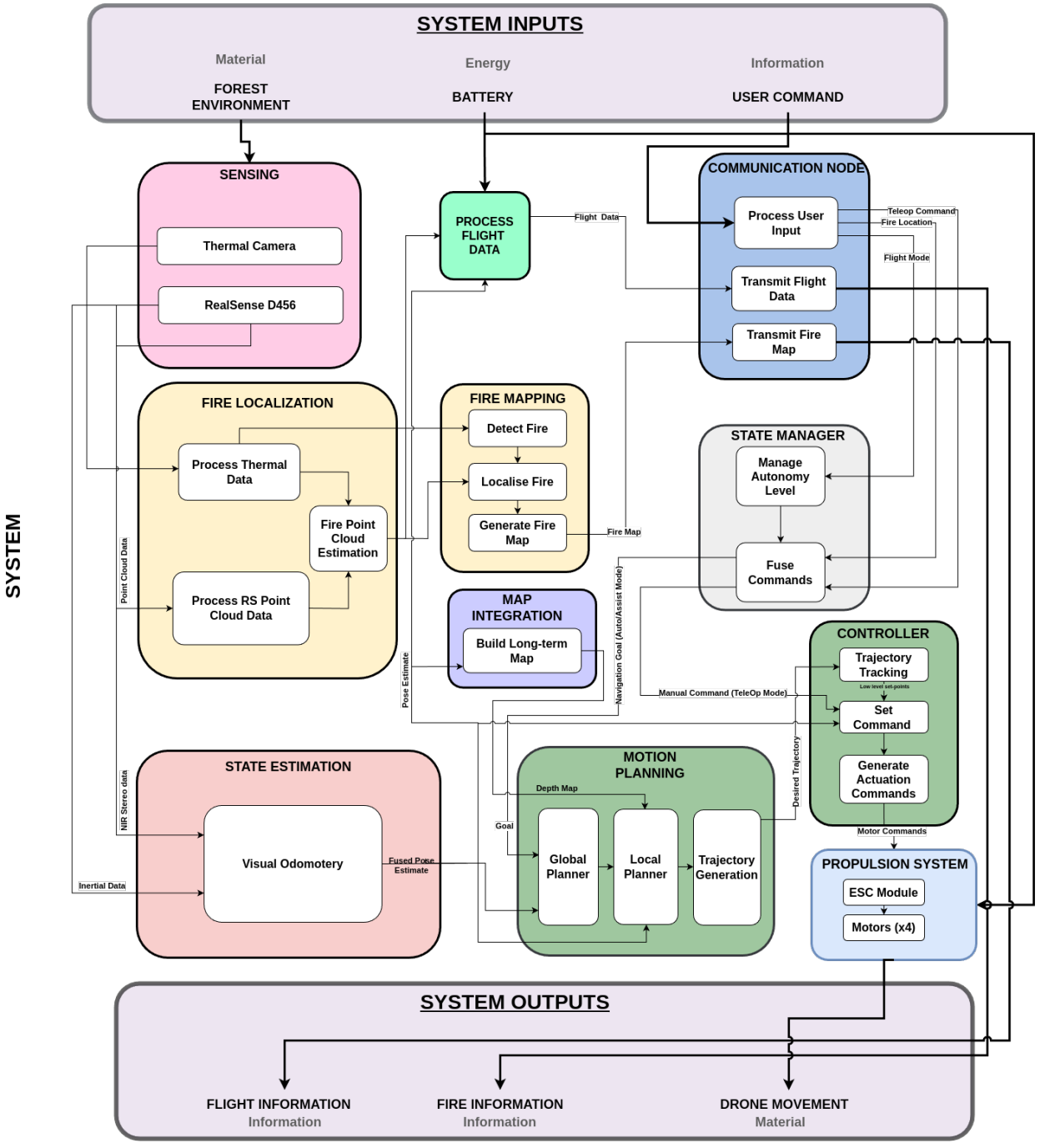


Figure 7: Cyberphysical architecture

The inputs to the system are the environment (material), electrical energy from the battery, and the user command (information). The desired output of the system is drone movement and fire information. The major subsystems present are Sensing, Depth Estimation, Fire Perception, Map Integration, State Estimation, Motion Planning, State Manager, Controller, Propulsion System, and Communication Node.

The flow of information between the modules is as follows. The sensor suite present on the aerial platform senses the environment and transmits the RGB and thermal image data to Depth Estimation and State Estimation modules. The State Estimation module takes in GPS location (degraded), thermal images, and RGB images and provides an accurate position of the robot. The local depth map is transmitted to the fire perception module for generating a heat map of fire locations, and to the Map Integration module. The output of the Map Integration module is a long-term global map that is used for motion planning. The planned trajectory is tracked using the Control and Propulsion subsystems. These are described in more detail in the subsystem descriptions.

One of the key inputs to the system is the **user command**. This command essentially consists of the rough GPS location of the fire (goal for navigation) and the autonomy mode command. The two modes of operation are assistive teleop mode and fully autonomous mode. State-manager module defines the state of modules, i.e. whether the particular subsystem is On/Off, and the flow of information.

We also have an external subsystem, namely the GCS (Ground Control Station), which is the means for information exchange between the Aerial System and the operator. As such a typical GCS consists of a field-rated touch screen for visualizing the flight data and telemetry, joysticks and switches for vehicle control, and a unified high bandwidth radio link for seamless low-latency wireless communication with the aerial system.

7 Current System Status

7.1 Overall System Depiction

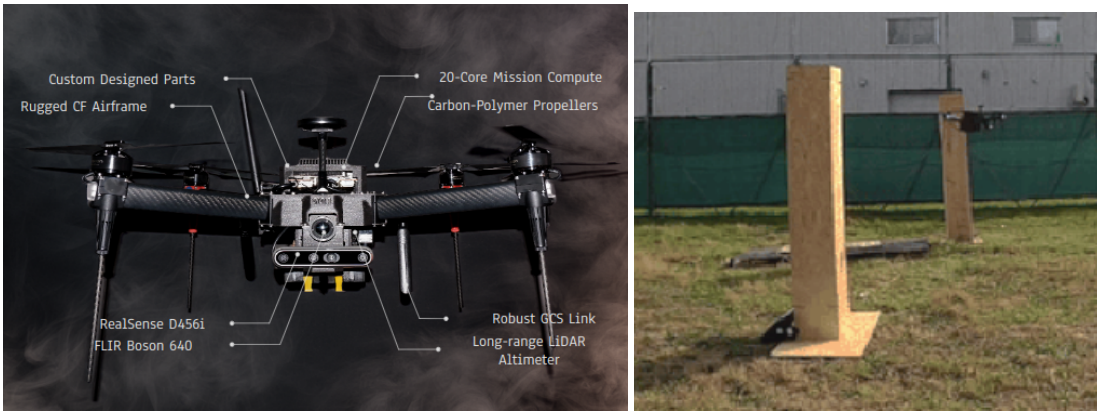


Figure 8: Overall System Depiction

The overall system comprises of the aerial platform: **Phoenix Pro** and the Ground Control Station, Fire Perception, State Estimation, and Autonomy subsystem.

The team custom-built the **Phoenix Pro** sUAS platform, which is equipped with a RealSense D456i camera (featuring 2 NIR cameras, 1 RGB camera, and an IMU sensor) and a Boson 640 thermal camera for environment sensing and depth prediction. The platform is powered by an onboard 20-core x86 compute unit and offers a reliable flight time of over 16 minutes. **The ground station** consists of mainly two systems: a laptop connected to the OTS GCS Herelink [3] which is connected to the onboard Mission computer via a wireless ethernet link to launch the relevant scripts and visualizers, and another laptop which monitors the QGC application to check for telemetry data and battery levels.

The software subsystems include the **Fire Perception** module, which serves a dual purpose: localizing fire hotspots (simulated using space heaters placed across a field) and appending these predictions to a map that is relayed back to the ground station in real-time. The **State Estimation** module leverages the left and right NIR cameras and IMU sensor of the RealSense camera to track features and estimate the pose (position and orientation) of the sUAS platform at a rate of 25Hz. Lastly, the **Autonomy** module functions in two modes: Point-to-Point mode, which is used to navigate between predefined goal points, and Exploration mode, which enables autonomous exploration and mapping of a designated region, both while effectively avoiding obstacles.

7.2 Subsystem Descriptions

7.2.1 Aerial Robotic Platform

Our previously developed aerial platform (ORDv1 aka Phoenix) was found to be inadequate for our FVD demo scenarios for the below reasons:

1. **Limited Endurance**: A Limited flight time of **3 minutes** for Phoenix would have severely limited the scope of the testing and demonstrations for FVD, since the exploration part of the demo by itself could take up to 3-4 minutes.

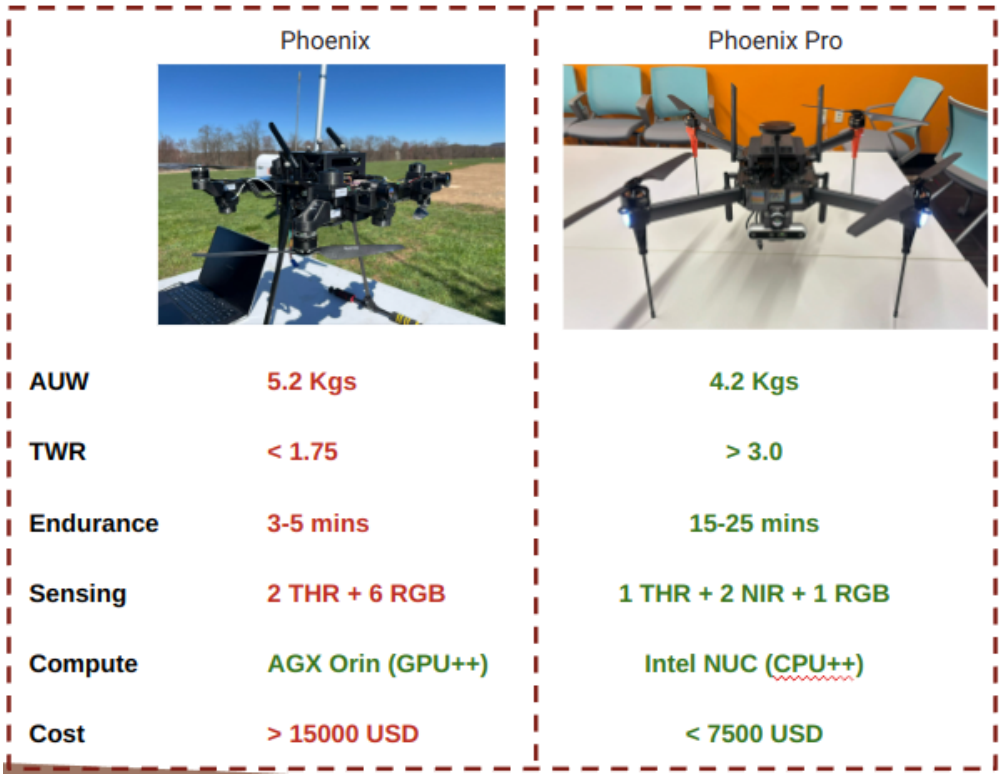


Figure 9: Aiframe Comparison

2. **Limited Thrust-Weight Ratio (TWR)**: Phoneix with its measly TWR of 1.50 places severe limitations on the type of aerial manoeuvres achievable in-flight. This is because agile flight would render its actuators saturated degrading the flight performance and as a result pose a significant hazard to life and/or property.

As a result, we developed a new aerial platform tailored to our overall use case. We call the new platform **Phoenix Pro**. Phoenix Pro was built using the Hexsoon EDU-650v2 airframe kit to accelerate development and improve modularity for future retrofits. Please see figure 9 for a comparison of the two.

The new airframe was developed in its entirety during the Fall 2024 break. The development workflow remained the same as before. The first successful flight test was performed at the RI SqH drone cage on 20 October 2024. Since then several such tests have been performed (manual and autonomous) and the platform was demonstrated to be reliable, rugged and stable.

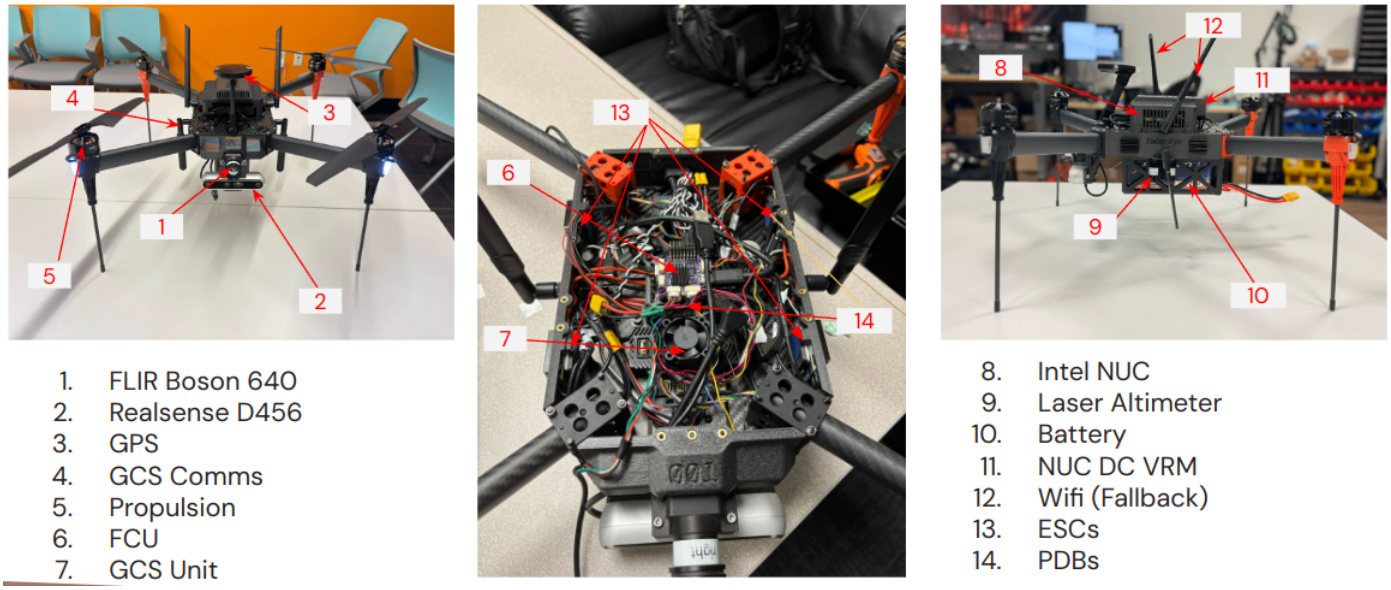


Figure 10: Phoenix Pro: Deconstructed

7.2.2 State Estimation

As our system operated autonomously in a GPS-denied environment, it was essential to develop a robust state-estimation pipeline capable of handling variable temperatures and sudden UAV maneuvers. Furthermore, adhering to our non-functional requirement **MN4**, which mandates the use of passive sensing modalities, we adopted a vision-IMU-based state-estimation approach.

After deciding to build a new UAV platform in the Fall, we revisited the decision to use Multi-Spectral Odometry (MSO), which was utilized during our Spring Demonstration. Given that MSO is not well-documented, tightly integrated within a Docker environment on ARM-based hardware (ORIN on ORDv1), and lacks complete code commits, combined with the departure of the key contributor from Airlab who implemented it on ORDv1, we decided to not use it. Instead, we opted for the off-the-shelf VINS-Fusion package, an optimization-based multi-sensor state-estimation framework. VINS-Fusion is widely adopted in the robotics community and is recognized for its robustness.

The key prerequisite for using the VINS package for state estimation with an IMU and camera was to provide accurate IMU-to-camera extrinsic, camera intrinsics, IMU gyroscope and accelerometer biases (noise), and the time offset between the IMU and camera topics. Figure 11 illustrates the calibration pipeline developed for these tasks. Initially, Kalibr was used to obtain the camera intrinsics for the RealSense left and right NIR cameras using a checkerboard pattern. The imu_variance_ros package was then used to estimate the IMU biases, which were subsequently used by Kalibr to perform the calibration and obtain the IMU-to-camera extrinsic (for both cameras) and time offset.

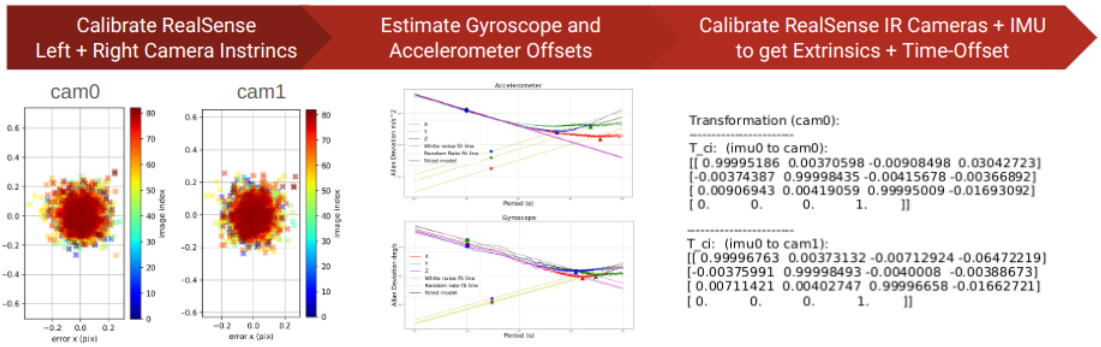


Figure 11: Calibration Pipeline

Figure 12 illustrates the overall flow of our state estimation module. The VINS node uses an IMU sensor and two NIR (Near-Infrared) images from the left and right cameras of the RealSense D456 to estimate the odometry. This data is then processed by the Transformer node, which publishes the odometry at a fixed rate of 25Hz, transformed with respect to the UAV platform's base-link frame. Additionally, this node is responsible for broadcasting the transforms between the map, IMU, odometry, base-link, and thermal camera to the RealSense depth frame.

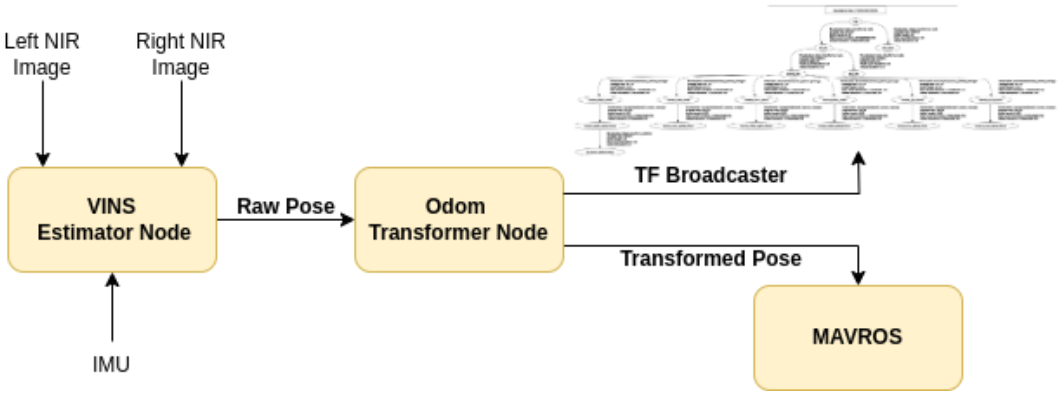


Figure 12: State-Estimation Architecture

7.2.3 Fire Localization Pipeline

Fire Localization in Phoenix: In our previous aerial system, Phoenix, the fire localization module processed thermal images to determine fire hotspot coordinates for mapping. Two methods were explored: a learning-based approach developed by AirLab and a classical method developed by our team.

The deployed method relied on classical stereo matching of synchronized thermal feeds, rectified using OpenCV's *stereoRectify* to correct distortions. Temperature-based masks were generated to segment hotspots, and ORB features were then detected and matched between the left and right frames. The matches were refined using epipolar constraints (Fig. 13) to estimate depth and compute 3D coordinates in the world frame.

Alternative approaches included generating disparity maps with StereoSGBM and using Fast-ACVNet for learning-based depth estimation. These methods showed promise but faced challenges such as information loss during image conversion and poor performance due to distribution mismatches (Fig. 14) and (Fig. 15.

While Phoenix Pro employs a different fire localization system, these efforts informed our understanding of thermal-based perception.

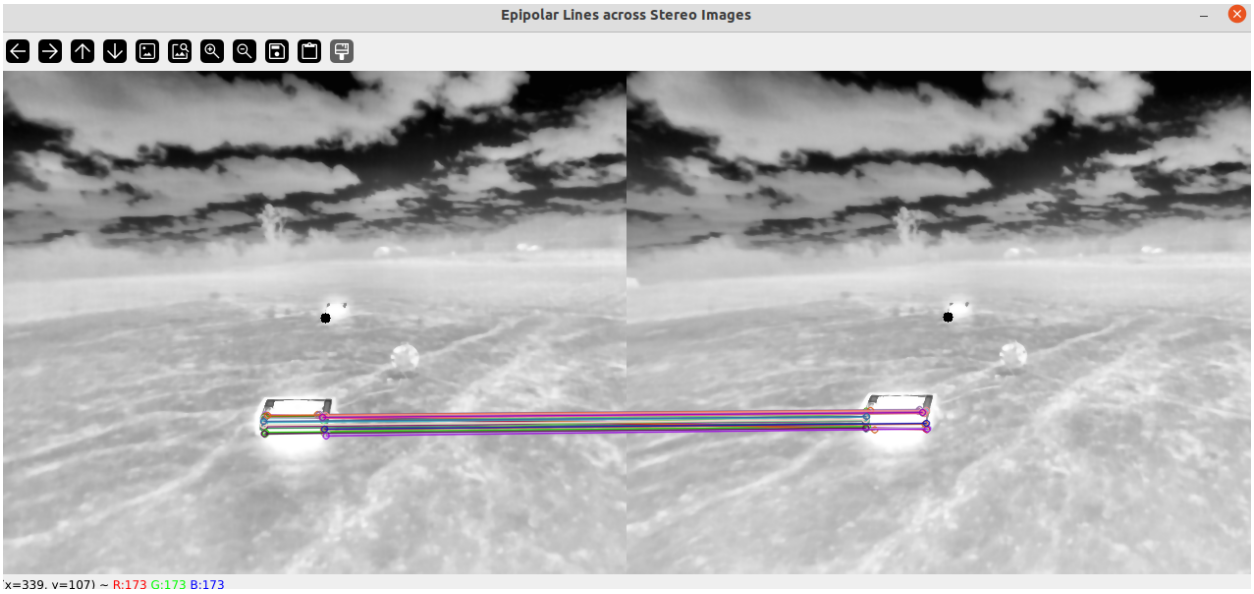


Figure 13: Epipolar lines joining the matched features

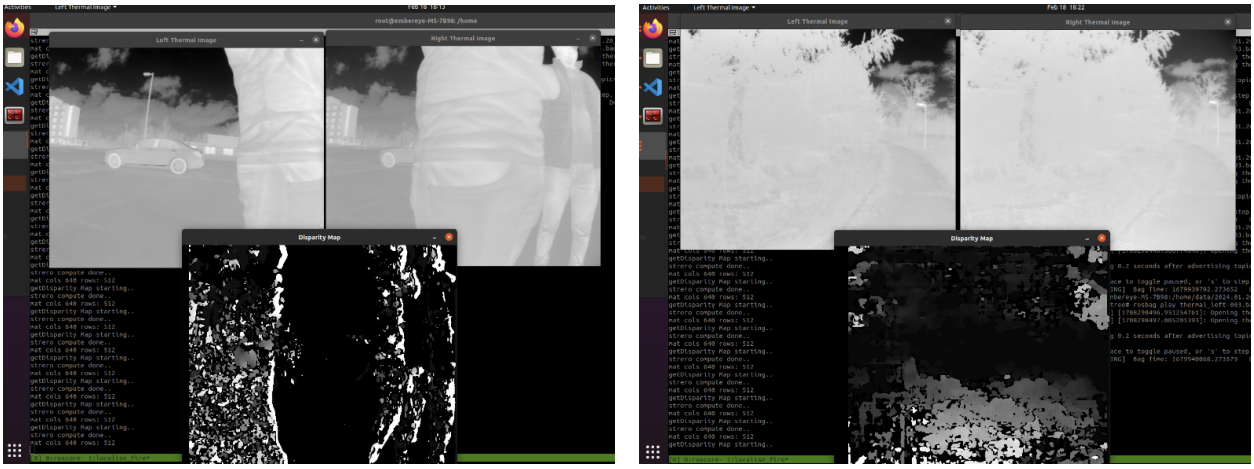


Figure 14: Disparity map from StereoSGBM algorithm

Fire Perception in *Phoenix Pro*

In the *Phoenix Pro* system, the fire perception module was designed to localize and map fire hotspots more accurately compared to the previous approach used in *Phoenix*. The previous system relied on a thermal stereo-based design, which exhibited a localization error of approximately 2 meters within a range of 5 meters. This approach faced several challenges, including the requirement for a

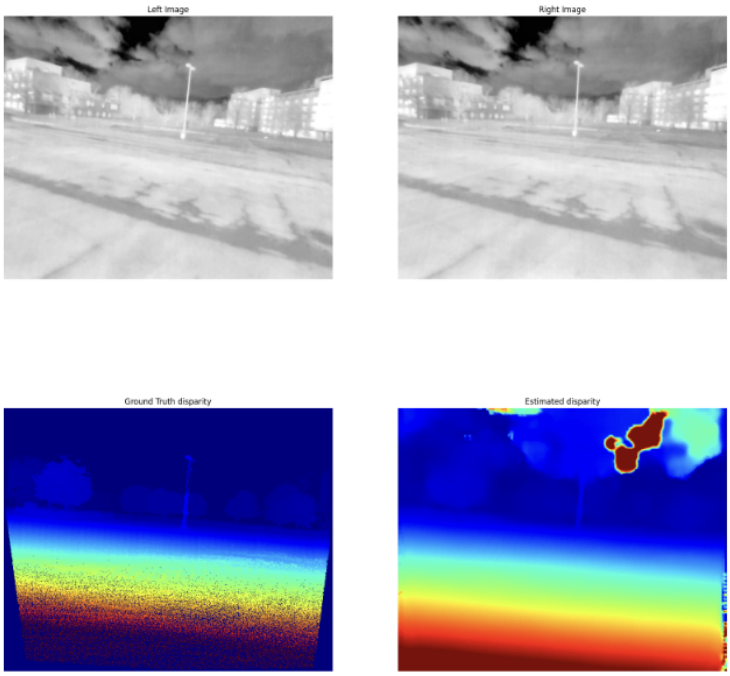


Figure 15: Disparity map from Learning-based Thermal Stereo

large baseline between thermal cameras and the difficulty of performing precise extrinsic calibration between them. These factors contributed to inaccuracies in fire localization, limiting the effectiveness of the system for real-world applications.

To overcome these challenges and improve localization accuracy, we adopted a new approach in *Phoenix Pro* that leveraged the RealSense point cloud, effectively bypassing the need for thermal stereo. The RealSense camera provided dense RGB-D data, enabling us to obtain accurate depth measurements without the need for large baseline configurations or complex thermal camera calibration. By integrating thermal information with the RealSense point cloud, we could project fire hotspots detected in the thermal images into 3D space with higher precision. This method significantly enhanced the accuracy, localizing fires now up to 50cm error for a 6m range.

The overall flow of information from the sensors (FLIR Boson + RealSense D456) to the fire localization and mapping is depicted in Fig. 16. We receive data from FLIR Boson at 30 fps, and point cloud from RS D456 at 30 fps. We first downsample the point cloud with a leaf size of 0.1m, and clip the range to 8m. This reduces the computational load and filters out bad points in the point cloud. With this configuration, we are able to achieve localization at 30 fps, i.e., in realtime.

We do a binary segmentation of fire hotspots, transform the point cloud into FLIR Boson's optical frame using RS-Thermal extrinsics, and then project the points onto the FLIR's frame using the intrinsic of FLIR. We reproject only the pixels corresponding to the hotspots using the mask, and then publish the centroid of the reprojected points. For tackling multiple hotspots, we cluster the segments in the binary mask generated from thermal image using OpenCV's connected_components. Clustering in 2D image space rather than 3D space reduces the computational overload significantly and enables to operate in real-time. The same has been depicted in the Fig. 17.

Note that the range of point cloud affects the fire perception capability during exploration. This parameter should be set based on the depth range assigned in the exploration. If the drone performs

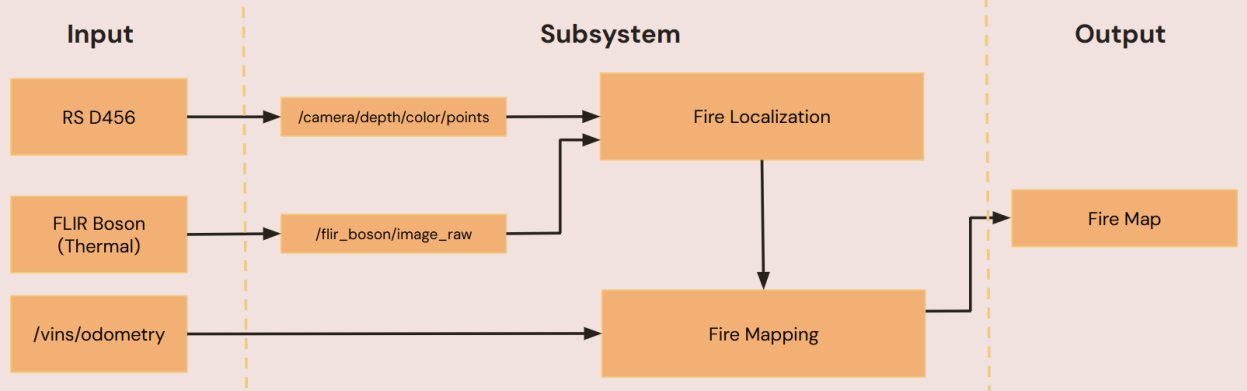


Figure 16: Fire Perception Pipeline

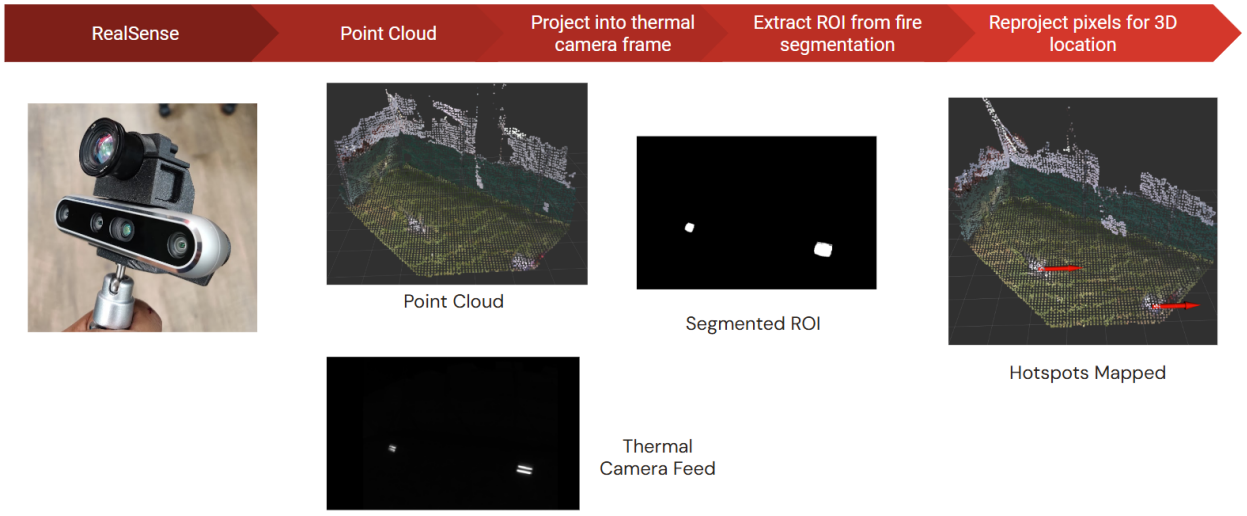


Figure 17: Fire Localization Pipeline

less exploration, then it is advised to set a longer range. However, this would affect the localization accuracy because of poor points from RS D456.

7.2.4 Global Fire Mapping

Our global mapping subsystem handles the temporal side of the fire perception subsystem. The fire perception module provides raw measurements of hotspot locations relative to the world frame. When the fire perception subsystem publishes a new set of measurements, this mapping subsystem is executed as described in Figure 18.

Filtering: [Deprecated in FVD] Once we receive a hotspot measurement, we first need to filter out the good readings, because most readings are quite noisy. The following are the 2 main filters we employ. Figure 19 shows the measurements from 2 successful flights (L) and the measurements after filtering (R). The red spots show the ground truth hotspot locations.

1. **Height Filter**: Filter out hotspot estimates above the UAS, since they are incorrect measurements.

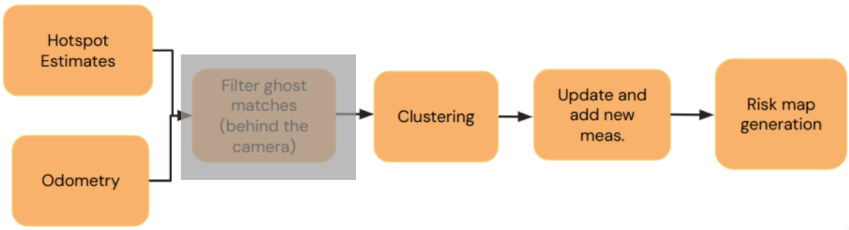


Figure 18: Overall architecture of Mapping Subsystem

2. **Distance Filter**: Filter out hotspot estimates too far from the UAS pose, since measurements which are too far are prone to parallax errors (as can be seen in Figure 19 (L)). The distance parameter of this filter can be tuned.

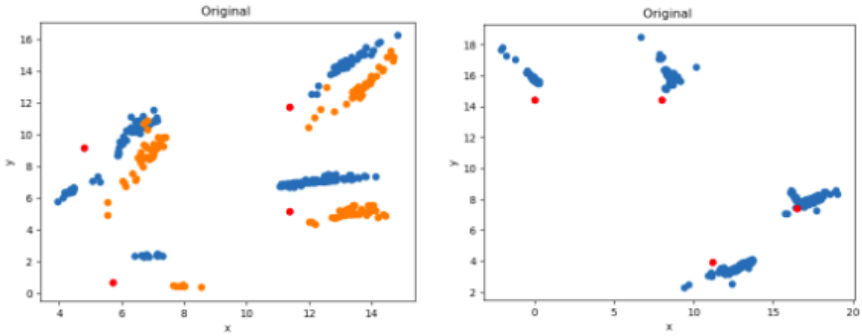


Figure 19: Measurements from 2 successful flights (L) and Filtered measurements from 1 flight (R)

Adding New Hotspot: We then perform nearest neighbor clustering for each hotspot to determine whether it is a measurement for a new or existing one. A new hotspot is created if it exceeds distances from all previously seen hotspots.

Update Existing Hotspot: If we instead find any measurement corresponding to an existing hotspot, we add it to the existing nearest-neighbor map, and use a mean search to find the updated position of its parent hotspot.

7.2.5 Ground Control System

This is a key part of our overall system as it is central to relaying the fire analytics processed onboard back to the first responders. This subsystem has remained unchanged since SVD.

After careful evaluation of different approaches to architect our GCS system, we ended up going for a development time optimal approach wherein most of the GCS hardware was outsourced as a fully integrated unit in the form of the HereLink v1.1 GCS Kit. We then proceeded to develop the required ROS wrappers and perform the necessary network configurations to enable a reliable and low-latency data stream link back to ground operations.

Figure 20: The HereLink GCS Kit

All non-image ROS topics were relayed as is over the Ethernet interface using distributed ROS over LAN. Image topics were processed using a custom ROS-GStreamer wrapper [4] to perform H.264 image encoding allowing for efficient low-latency image streams.

The same LAN was used to interface with the UAS via an SSH session to initiate the SW systems bring-up.

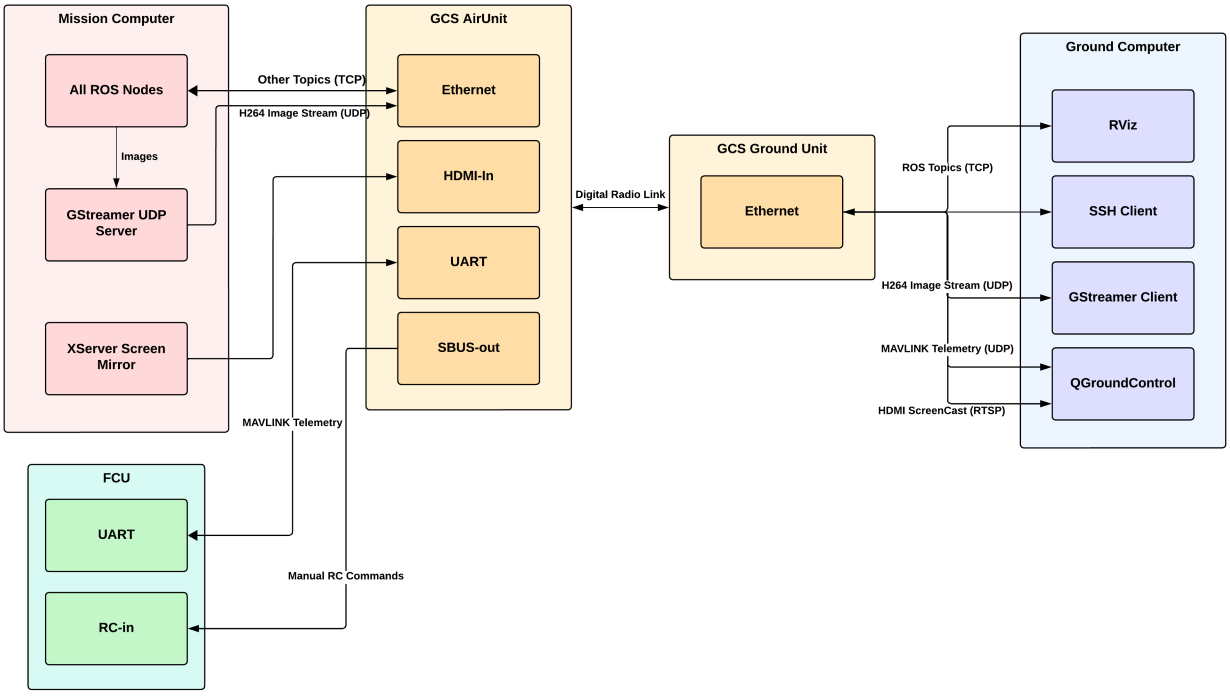


Figure 21: GCS Architecture

As seen in Fig. 21 we have a multitude of interfaces on the GCS AirUnit to pass a variety of data streams. All the streams are intercepted on the ground side via the Ethernet interface [2] and utilized by downstream processes for Visualization. Also, we should note that the link is bi-directional, enabling rich and dynamic user-centric workflows.

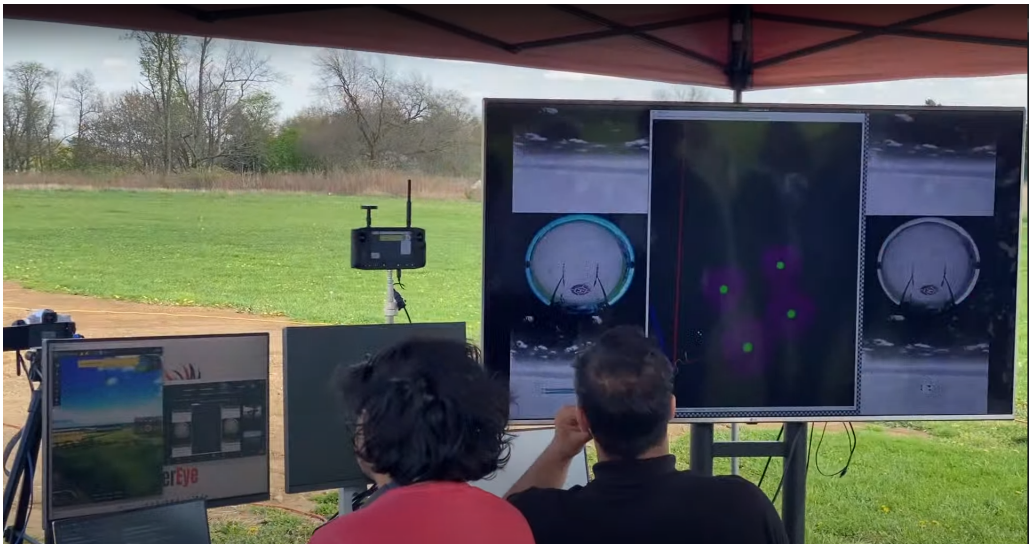


Figure 22: Overall GCS Setup

7.2.6 Autonomy

This subsystem handles the essential capability of our use case wherein the drone **autonomously** goes to a rough location while **avoiding obstacles** and then performs **exploration** inside a certain area to map out hotspots and relay this information back. To successfully demonstrate this capability, we divide our modules into a Point to Point Navigation module, and an Exploration module. At a higher level, these separate modules interface with a central Autonomy Manager which controls the state transitions, control outputs from the finite state machine to the MAVROS API which further sends the control commands to PX4's inbuilt controller. The overall architecture takes the shape as shown in Fig. 23.

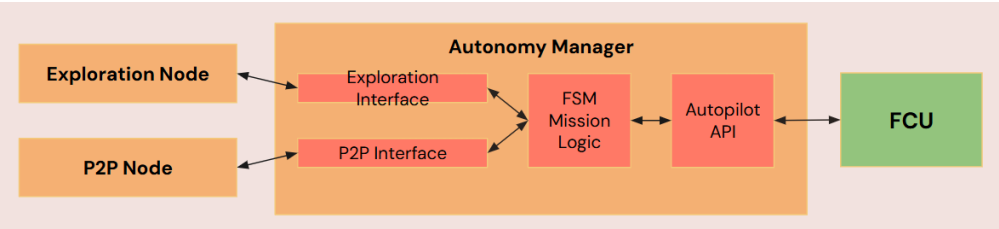


Figure 23: Autonomy architecture with FSM

Diving a bit deeper, we have internal and central states in our state machine architecture. To keep track of the overall state of the system, we have some central states which are the minimum essential for aerial systems and they help us implement logic for the entire mission. These are listed in Fig. 24.

- **GROUNDED:** System on the ground in unarmed state.
- **TAKEOFF:** System takes off and reaches desired height.
- **HOVER** System in offboard mode. holding positions, awaiting interface state change.
- **ACTIVE_P2P:** Active in P2P navigation mode
- **ACTIVE_EXP:** Active in exploration mode
- **LANDING:** System landing (in case of mission end or failure)
- **END:** Mission end, disarm, no further transitions.
- **FAILURE:** System level failures – asynchronous

Figure 24: Central States

Inside the Active P2P and EXP states we have further internal states inside the P2P and Exploration modules. These internal states handle the inner logic and help the central manager understand what's going on inside the individual packages. These internal states are listed in Fig. 25

- **PRE_INIT** : Pre-initialization phase to allow checking for inputs to either planner or exploration.
- **INIT_READY:** Above checks pass, system ready to receive goal
- **ACTIVE:** System is executing task (p2p navigation or exploring an area)
- **DONE:** Successful completion of task
- **FAILURE**: Can happen at any step during the execution of task.

Figure 25: Internal States for P2P and Exploration Modules

The essential components of the autonomy subsystems which use the above states and direct the mission flow are:

- **Central Autonomy Manager**: We have our mission logic inside the central manager. This handles the following tasks sequentially:
  - Arming the drone and takeoff commands using MAVROS API
  - Switch to OFFBOARD mode and checks
  - Once in OFFBOARD, and if overall system status in HOVER, switch to ACTIVE-P2P
  - Once a goal is given, P2P goes to ACTIVE
  - If there is no Failure, and P2P is done and system inside exploration region, switch to ACTIVE-EXP
  - Once exploration is done, and there is no failure, go to LANDING.

  The rough plan explained above has checks in between to ensure correct state transitions. Using these state transitions, when exploration is done, we can switch back to ACTIVE-P2P mode and give the goal as home base however this was not demoed at FVD due to lack of testing time, but this was tested successfully in simulation. These state transitions are logged in real time so that the operator knows the current state of the system. This was especially helpful to us during testing.

- **P2P Navigation Module**: We decided to go with CERLab's work for navigating in presence of static and dynamic obstacles (we were only interested in static obstacles). This package was

easy to setup and run, and our plan was to detach the controller from this package, and use the trajectory generation output and feed the way-points into our autonomy manager which can then further handle sending these way-points to PX4's inbuilt controller. The inner working include performing A-star search on the free/unoccupied pixels within the sensing range and generating waypoints on a fitted trajectory to minimize path length and avoid obstacles with a safety inflation radius.

- **Exploration Module**: We decided to go with HKUST's FUEL [6] package for this module due to previous familiarity with its workings. This module employs a frontier-based planner which directs trajectories within the sensing range towards un-explored regions of the map guided by frontiers. Over time it updates it's weights and improves it's planning in the same environment map. This can be seen in the map generated by this module and visible frontiers in Fig. 26

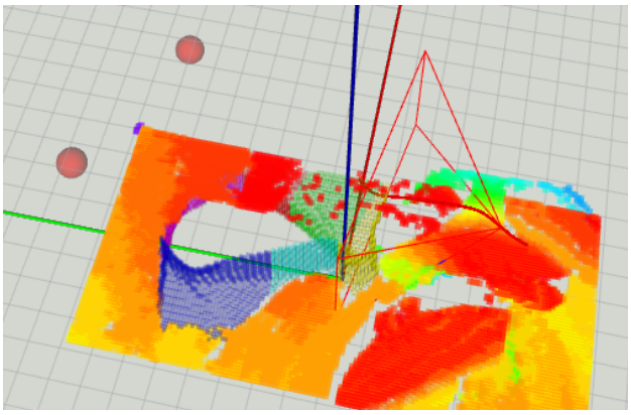


Figure 26: Exploration map returned in real time from field test

- **Interfaces**: These scripts handled the communication between the central manager and the individual modules. They handled communicating the control outputs from the modules to the manager, and the internal states which were used in the central manager to handle transition of the central states.

7.3 Modelling, Analysis and Testing

As detailed in our Fall Test Plan, we performed a continuous validation of our subsystems through several different tests, as shown in Table 7.

Table 5: Targeted Performance Requirements for Fall Semester

Test	Success Criteria	Result
Classical Thermal Stereo Pipeline Test on Dataset	- Qualitatively (visually) reasonable depth estimation results. - 99%-ile depth errors within 0.5m for maximum scene depth of 10m. - Minimum 3D Pointcloud output rate of 15Hz.	- Met all requirements using OpenCV SGBM and MS2 Dataset

Robotic Platform Structural Integrity Test	- Motor mounts rigidly affixed to the main airframe structure (no relative motion). - Sensor mounts (IMU, Thermal, and RGB Cameras) rigidly mounted to the airframe. - In-flight IMU vibration metrics should be lower than the prescribed limits (5 m/s/s). - No motion blur should be visible on all the camera image feeds.	- Motor and sensor mounts rigidly mounted and validated from in-flight sensor feeds. - Vibration metrics less than 5m/s^2 - Lack of motion blur verified through sensor streams.
Thermal Camera Calibration	- Calibration RMS reprojection error should be sub-pixel	- Achieved mean calibration projection error 0.000015±1.526674 pixels
Fire Hotspot Localization accuracy	- The module should localize fire upto a range of 5m. - The fires localized should be within a 2m radius.	- Achieved a localization accuracy upto 20cm error (for 5m range) - Achieved 30 fps publish rate - Achieved a min of 5m range.
P2P Autonomous Navigation SITL Test	- The planner should reach the user-specified goal safely while satisfying the enforced realistic constraints (velocity, acceleration, sensing distance) - The frames per second (fps) achieved from the segmentation model should be a minimum of 10 FPS.	- Planner was found to navigate to goal with realistic constraints in simulation.
Point-to-point and exploration pipelines validated in simulation	- Successful build and no dependency errors - P2P planner and Exploration pipelines run with correct state transitions.	- Built successfully and runs without any dependency issues - P2P and Exploration runs in simulation as desired for realism.
Fire Segmentation on Real-time Feed	- Detect fires with at least 70% accuracy. - The frames per second (FPS) achieved from the segmentation model should be a minimum of 10FPS.	- Detected fires with 100% accuracy - Achieved online frequency of 13 Hz of segmentation system.
Verification Tests for Modified Hardware	- Appropriate and stable regulated voltage is supplied to the FCU [5V]. - Handheld test by running VINS-Fusion with expected odometry rate of 30Hz and no latency. - Stable RealSense feeds streaming to verify proper wiring connections to NUC - Appropriate and stable voltage is supplied to the propulsion system [22.20-25.20V]. - Enough voltage is supplied to the on-board compute Intel NUC 13 [12V]. - In-flight IMU vibration metrics should be lower than the prescribed limits (5 m/s/s). - FCU log should indicate a nominally safe level of system resource utilization - RealSense camera mount rigidly mounted to the airframe and motion blur	- All the metrics were validated successfully after some initial hiccups with USB3 EMI issues and USB power delivery issues.
Visual-Inertial State Estimation Test	- Stable VIO estimates at the required update rate of 10Hz - Have a drift of < 4% over a track length of 100m	- Achieved 25Hz state estimation update rate (real-time) - Validated drift of 0.1-0.5% over a track length of 100 m
Ground Control Station Communications Test	- GCS unit connects to the Aerial Platform. - Able to configure drone mission parameters from the GCS unit. - Stable link connection with a communication range of at least 150 meters. - Stream processed visualization data at a minimum of 5 FPS.	- GCS unit successfully connects to system and can configure mission parameters. - Link achieves > 150m communications range. - Link achieves 30 FPS video streaming, 15-15 FPS map streaming, and 200 FPS VIO streaming.

Full System Reliability Stress Testing	- Same as FVD	- All desired requirements met with iterative system improvements leading up to an successful FVD demo

7.4 FVD Performance Evaluation

We demonstrated our system capabilities (shown in Table 6) during the FVD and FVD Encore runs. In the first FVD run, the pilot performed a manual landing during the autonomous exploration phase as a precautionary safety measure. Following this, two successful autonomous demonstrations were carried out. Overall, the system met all performance requirements, with the exception of a false positive hotspot detection during one of the FVD runs, which was addressed and corrected in the subsequent FVD Encore demonstration.

Table 6: FVD Performance Evaluation

Procedure	Success Criteria	Requirements
UI displays an updated fire map throughout duration of the flight.	UI successfully displays real-time feed throughout duration of the flight.	**M.P.8, M.P.9**
Communicate with drone up to 150 meters.	Successfully communicating with drone up to 150m.	**M.P.8, M.P.9**
Have a flight time of more than 5 minutes.	uUAS successfully had a flight time of more than 5 minutes.	**M.P.5**
Detect 70% of all hotspots with at least 2.5m accuracy.	87% of hotspots detected overall owing to positive detected in the runs with an accuracy of 1m	**M.P.1, M.P.2, M.P.3, M.P.6, M.P.7**
Navigate autonomously around obstacles with a separation of 5m and complete the mission	Succesfully navigated around obstacles within a separation of 5m and completed 2 out of 3 runs owing to a precautionary landing.	**M.P.1, M.P.2, M.P.3, M.P.4, M.P.5**

7.5 Strong/Weak Points

Our progress over both semesters culminated in successful demonstrations during SVD, SVD Encore, FVD, and FVD Encore. While our system demonstrated several strengths, there are still areas that require improvement. Below, we highlight both the strengths and weaknesses of our system.

Strengths:

- Robust Aerial Platform: Throughout extensive testing, our aerial platform experienced multiple crashes, yet remarkably, there was no significant damage to our system.
- Improved Flight Time: The newly developed platform, **Phoenix Pro**, achieved a flight time exceeding 16 minutes. This allowed us to successfully complete the entire FVD demonstration in a single run and significantly reduced downtime during testing, enabling faster development of functionalities.
- Robust State-estimation: The module demonstrated reliable performance, keeping the overall drift below the targeted 4%, even when tested under variable weather/lighting conditions and erratic sUAS movements.

- Accurate Fire Localization: Integration of RealSense camera for obtaining point cloud has drastically improved our fire localization accuracy. Additionally, the entire stack is written in C++, enabling us to achieve realtime localization performance.
- Reliable sUAS-Ground Communication: The GCS module was able to reliably stream information between the ground station laptop and the sUAS platform without packet drops, enabling us to showcase a live RGB camera feed during our runs.

Weaknesses:

- Fire localization: Our new sensor fusion-based approach for fire localization worked extremely well compared to our implementation demonstrated in SVD. But we still face the issue of false detections when the heater is partially occluded owing to imperfect extrinsic calibration between the D456 depth camera and the Boson-640 thermal camera.
- Erratic Flight Behaviour: Since we rely solely on visual-inertial sensing for state estimation, when operating the system in an outdoor setting, the low temperatures (0-5 C) affect the IMU characteristics significantly and result in erratic divergence of the UAS state estimate resulting in flight crashes.
- Autonomous Exploration: While the present level of performance was sufficient to meet our requirements for the overall system, it could be greatly improved by enabling more flexible kino-dynamic constraints when operating in a larger area.

8 Project Management

8.1 Schedule

We discuss our original plan for the fall semester as thought after our SVD, the major changes in the plan, and evaluation of our overall scheduling process. We also identify key success and failures which eventually helped us deliver a successful FVD and Encore.

8.1.1 Original Schedule for Fall Semester

Before moving to the discussions about our scheduling this fall semester, we would like to highlight some key points from our original schedule plan for fall semester as submitted in our SVD report. In the table 7 we can see we had projected a possiblity of having a new airframe but were relying on using thermal stereo cameras and RGB fisheye for sparse depth, and VIO using Airlab's modules.

Table 7: Original Plan for Fall - Submitted in SVD Report

Date	Milestones
15 September	Bi-weekly 1 (B1) - Reconsider new platform, Explore methods for Fisheye Depth Estimation
7 October	Milestone 1 (M1) - Have a new airframe, Achieve sparse depth from fisheye
8 October	Bi-weekly 2 (B2) - Integrate new airframe, setup VIO
4 November	Milestone 2 (M2) - Integrate mapping and motion planning
15 November	Bi-weekly 3 (B3) - Complete and integrate autonomy stack
22 November	**Fall Validation Demonstration (FVD)**
4 December	Bi-weekly 4 (B4) - Encore

Coming into the fall semester we discussed our motivating factors (endurance and autonomy being the two most prominent) for shifting to a new airframe and ditching the fisheye cameras and thermal stereo pair for a single thermal and RealSense RGBD camera. We also planned to use open source implementations for the autonomy stack once we converged on the elements of our fall demonstration which take us closer to our use case in terms of capability and realism. The modified plan was roughly sketched as shown in Fig. 27. The other subsystems were expedited by a few days to leave room for the new airframe or the backup plan of removing excess weight from ORDv1 (the airframe used in spring semester).

We also ditched our biweekly stand-up structure from last semester and enforced common work hours during the week wherein the entire team sat together for 3 hour periods twice a week. This helped us be more in sync with each other's work, added more accountability, and the subsystems supported each other from day 1.

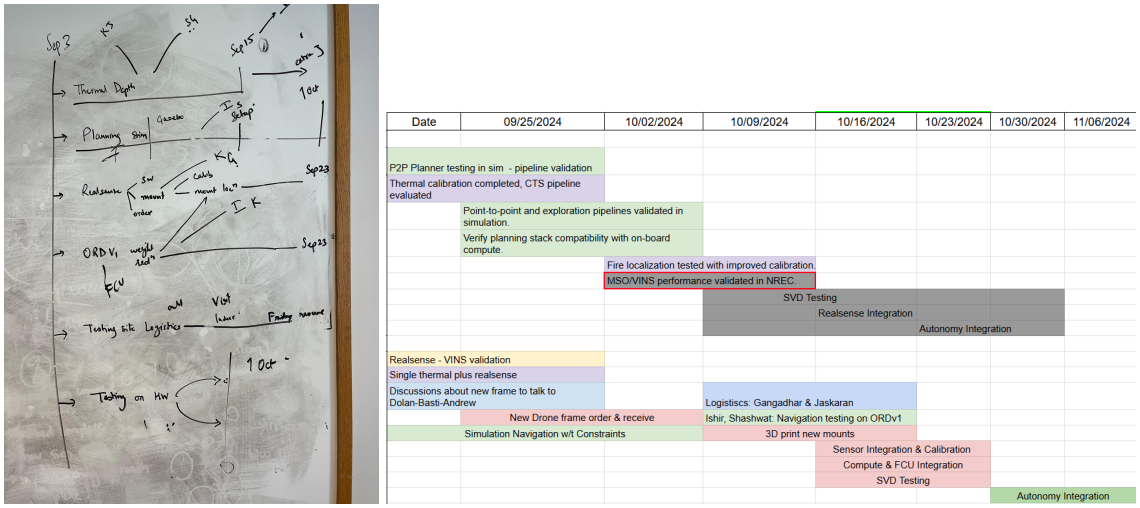


Figure 27: Modified Fall plan accounting for new airframe

8.1.2 Evaluating Scheduling for Fall

The team came together to take the radical decision of going through with a new airframe and managing the time pressure by parallelizing tasks as much as possible and expediting subsystems to leave room for preliminary hardware testing and full system integration. We introduced sufficient

padding in our plans and clubbed team members with relevant experience which resulted in successful completion of tasks with minimal delays. Simulation testing of autonomy was done in parallel and finished in sync with hardware development leaving no downtime for integration. Perception modules for hand tested in parallel as well. Spare parts were ordered and testing site was scoped out nicely. All of these tasks finished along with the new hardware development which helped us integrate and test on hardware immediately.

The team learned from the scheduling mistakes during spring and created a balance between exploring different methods and exploiting current knowledge to implement with confidence. In summary, the scheduling process of the team was highly efficient and helped us a lot during the semester.

8.1.3 Key points for success and failure

They key success points for us in our scheduling were:

- Syncing development of subsystems in simulation and the new hardware development.
- Parallel completion of handheld tests reducing iterative flights for perception subsystem.
- Deciding demonstration elements and logistics quite early helped us plan our last few weeks.
- Co-working sessions sped up our implementation process with quick knowledge transfer.

Some points of failure:

- Major changes in nearly every subsystem left little time to iteratively integrate and improve on the implementation.
- Sim 2 Real transfer time was underestimated and more time on that end would have helped us add more capabilities.

8.2 Budget

By the end of SVD Encore, we had spent a total of **USD 2993.30**.

This semester our spend increased significantly owing to the development of the new aerial platform. Aiming to foster a good UAS HW ecosystem for the project course, we with Prof. Dolan's approval ordered some parts which we could have otherwise sourced from the AirLab (LiPo batteries, LiPo charger, RC Link etc.).

As a result, our total expenditure totalling to **USD 7112.74** went above the nominally allocated limit of **USD 5000.0** by a significant margin. We however had informed Professor Dolan in advance and gained his approval before embarking on the hardware revamp work.

Our top 5 material expenses were:

1. Intel NUC13 - USD 953
2. Herelink GCS Kit - USD 900
3. Hexsoon Airframe Kit - USD 493
4. Intel Realsense D456 - USD 469

5. Pixracer Pro FCU - USD 350

For a complete look at the expenses please refer to this sheet (the sheet being quite large could not be added in-line inside the report). For a visual representation please see figure 28

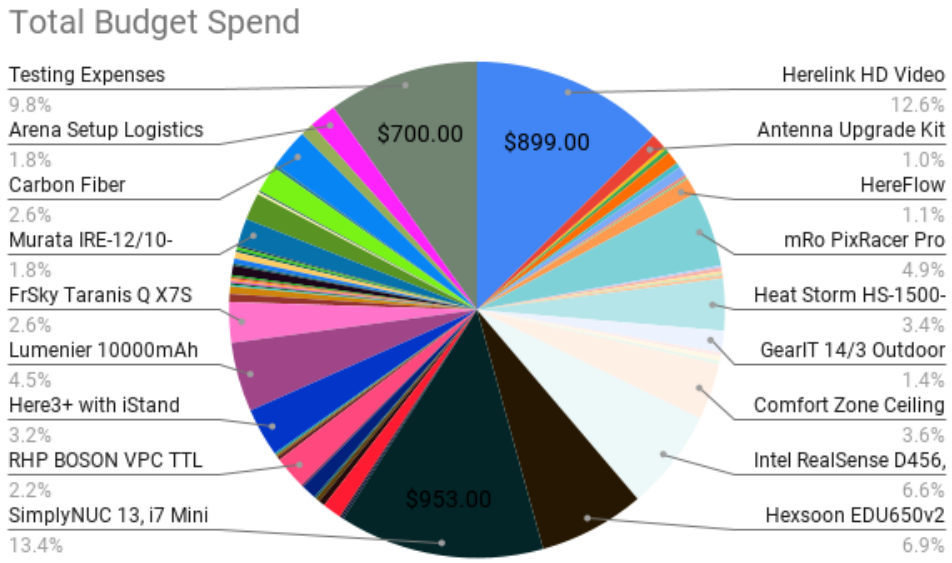


Figure 28: Spend Distribution

8.3 Risk Management

8.3.1 Keeping track of Risks

There has been no changes in this since last semester. We continued using Notion for our risk tracking and iteratively sought out to mitigate those risks since the start of this semester with the added internship experience in different fields. The full risk table can be found on this notion *page*.

8.3.2 Evaluating our Risk Management Process

Our risk management process helped us this fall semester during the following key development periods:

- **Finalizing Testing Space**: this was the biggest question mark for us moving into the fall semester as we went through 3-4 testing spaces during our spring semester which caused us a lot of delays and setbacks. To mitigate this, we sought out help from NREC and they allowed us to use their drone cage for iterative testing throughout the semester. This decision helped us structure our demo, plan our our tests with shorter advance notices, and give us more control over our testing environment.
- **New Hardware Frame**: Developing a new frame altogether in the fall semester was a risky move and hence we mitigated this by keeping Phoenix untouched. This allowed us the option to fall back in the case of integration issues with the new frame. Refer to Fig. 29

- **Stacks for Autonomy and VIO**: We continuously explored open source implementations rather than heavily relying on Airlab stacks. This allowed us to exploit our previous knowledge and implement modules we had previously worked with. This helped us modify open source implementation to our PR needs.
- **Drone Crashes**: Relying on the mitigation strategy of always ordering extra parts has helped us since the spring semester. In the fall semester we were able to quickly get our drone back flying after every crash using spare components. The team had mitigated this so well, that even after a crash during our encore demo, we could replace it and fix it within 3-4 minutes and then showcase two full successful runs.

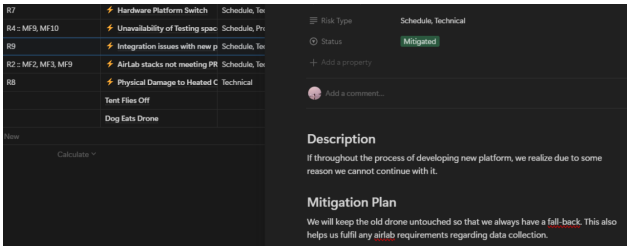


Figure 29: Phoenix kept intact as backup option

A single point of stress we faced with our process is we could have done a better job of including Sim2Real transfer risks in our timeline. Mitigation would have been limited as we already expedited our subsystem implementation and the new drone was also up and running in time. Weather conditions forced a reduction in our testing time and was the cause for some crashes. We hope the future teams doing outdoor projects with drones account for these risks as they were extremely helpful for us.

9 Conclusions

9.1 Lessons Learned

Minimizing Slack

Having learned valuable lessons from our progress until SVD and owing to the big new task of new UAS platform development, we had to parallelize our developmental efforts across all subsystems to efficiently handle the available manpower. This helped us in minimizing the slack in our timeline.

Hardware is Hard

Even after the successful completion of the airframe development on time, we still faced many downstream hardware issues which took significant time to analyse, find the root cause and then finally resolve the issues. This made us realize that even when adopting a stable hardware ecosystem new issues can still pop-up and we should always plan to account for these.

9.2 Future Work

In order to commercial the presently developed solution we have to add/improve our system significantly. These would be:

1. **System Generalizability**: Work towards making our overall system robust, reliable and stable for operation in select environment scenarios. One key improvement would be the development of thermal-only capable state estimation and depth perception pipelines for operation in smoke.

2. **Portability**: Work on making the overall system (UAS+GCS) man-portable to allow for quick deployments in remote locations.

3. **Improved Sensing**: Presently developed system only has depth/visual perception in the forward direction owing to the placement of the associated sensors. We should explore adding 360-deg depth/visual sensing for robust avoidance and safe operations.

4. **Standardized User Interface**: Our present means of interacting with the system is very low-level and not ideal for end-users like firefighters. A very useful development would be integrating **TAK** (Team Awareness Kit) with custom plugins to streamline the UAS operations on the field. TAK, which stands for Team Awareness Kit in civilian contexts and Tactical Assault Kit in military applications, is a suite of software designed to enhance situational awareness and geospatial collaboration. Developed initially by the U.S. Air Force Research Laboratory (AFRL), TAK has evolved into a versatile tool used across various sectors, including military, law enforcement, and emergency response.

5. **UAS Swarm:** Once all the above have been achieved, we could be a bit ambitious and explore a swarm-based solution for covering larger areas.

   We should note that some/all of the steps mentioned above may require a complete system re-design and hence we should plan for the additional overhead.

References

[1] Firefly mrsd team. https://mrsdprojects.ri.cmu.edu/2022teamd/team/.

[2] Gcs ethernet data transfer. https://discuss.cubepilot.org/t/ethernet-on-the-herelink-1-1/9766.

[3] Gcs herelink setup. https://docs.cubepilot.org/user-guides/herelink/herelink-overview.

[4] Gstreamer setup. https://gstreamer.freedesktop.org/documentation/tools/gst-launch.html?gi-language=c.

[5] Wildfire statistics. https://www.nifc.gov/fire-information/statistics/wildfires.

[6] Boyu Zhou, Yichen Zhang, Xinyi Chen, and Shaojie Shen. Fuel: Fast uav exploration using incremental frontier structure and hierarchical planning. *IEEE Robotics and Automation Letters*, 6(2):779–786, 2021.

A Appendix

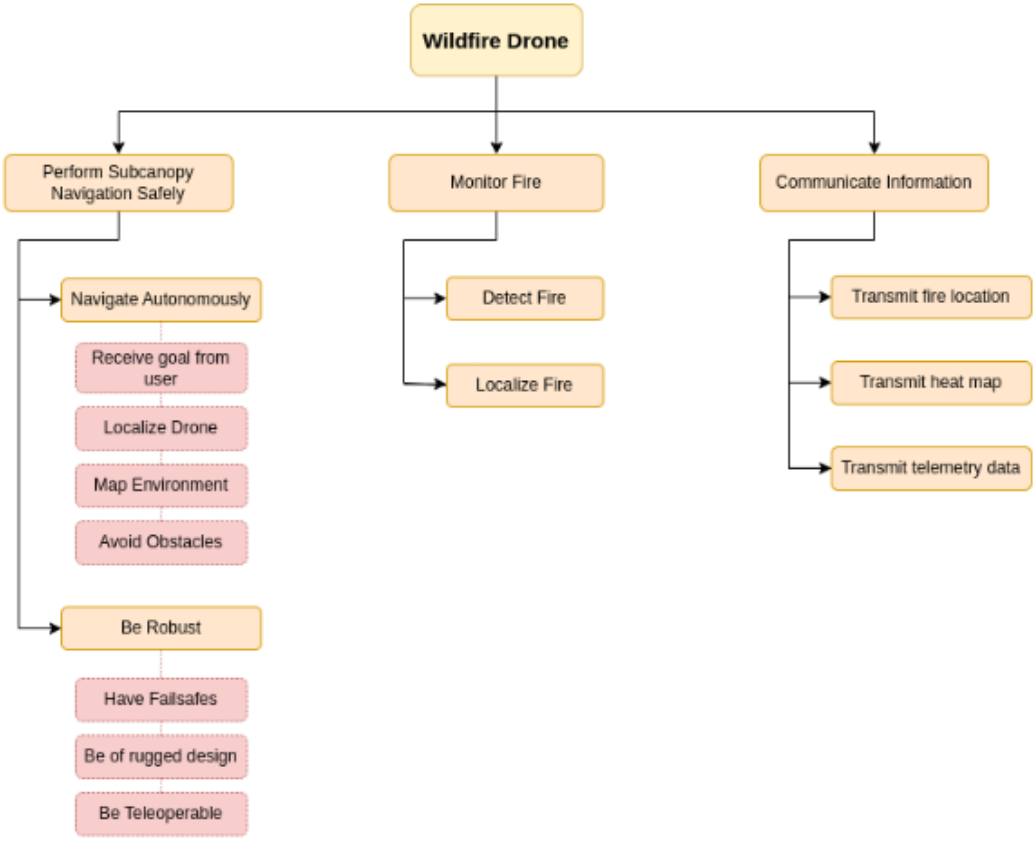


Figure 30: Objectives tree

Table 8: Risk ID 1: Delay in Hardware Development

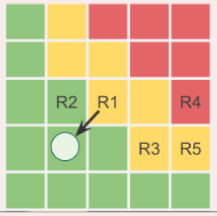
Risk Type	**Risk Owner**	**Date Submitted**	**Date Updated**
Schedule, Cost	Owner Name	12/01/2023	12/01/2023
Description	**Cause**	**Consequence vs Likelihood**	
Delay in Hardware Development at any stage in the testing phase	Can be due to shared inventory and delays in the outsourced manufacturing of certain parts		
Consequence			
If the hardware is not functional, there would be delay in the systems integration phase and in field testing.			
Mitigation Plan		**Expected Outcome**	
1. Parallely start working on setting up simulation to perform simulation testing 2. Use maximum off-the-shelf tested components to avoid manufacturing delays.		1. Simulation testing reduces the consequence of halt in software subsystems integration 2. OTS components reduce the likelihood of manufacturing delays.	

Table 9: Risk ID 2: AirLab stacks not meeting PR

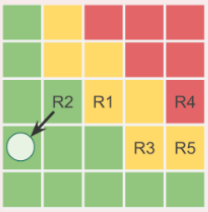
Risk Type	**Risk Owner**	**Date Submitted**	**Date Updated**
Schedule, Technical	Owner Name	12/01/2023	12/01/2023
Description	**Cause**	**Consequence vs Likelihood**	
AirLab stacks fail to meet the PRs which we expected.	While integrating the dependent stacks, their PRs get affected.		
Consequence			
This directly contradicts our promised PRs as part of the MRSD requirements. It also adds a schedule delay to find alternate options or work around negotiating PRs			
Mitigation Plan		**Expected Outcome**	
1. Parallely explore and document SOTA and possible open source solutions to estimate PR cushions. 2. Parallely integrate the dependent stacks as soon as individual stacks are tested.		Alternate options beforehand would helps us estimate the cushion our PRs would need and parallely integrating the stacks would let us know of any possible issue ahead in time.	

Table 10: Risk ID 3: Hardware Failures

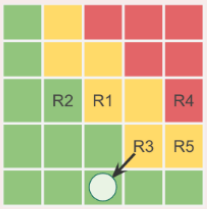
Risk Type		**Risk Owner**	**Date Submitted**	**Date Updated**
Schedule, Cost, Technical		Owner Name	12/01/2023	12/01/2023
Description	**Cause**		**Consequence vs Likelihood**	
Unexpected crashes or external parts might break if used over limit.	Replaceable parts like propellers, motors, prop guards get broken during random testing.			
Consequence				
An unexpected crash could have serious effect on the schedule and cost it would take to rebuild the hardware depending on the damage				
Mitigation Plan			**Expected Outcome**	
1. Perform stress testing of the hardware components which we keep in mind while testing. 2. Store replaceable parts.			Having an estimate of the hardware limits to reduce the likelihood and having spare parts can reduce the consequence.	

Table 11: Risk ID 4: Unavailability of Testing Space

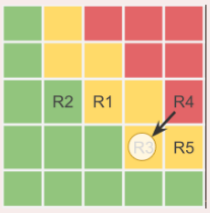
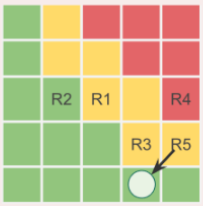
Risk Type		**Risk Owner**	**Date Submitted**	**Date Updated**
Schedule, Programmatic		Owner Name	12/01/2023	12/01/2023
Description	**Cause**		**Consequence vs Likelihood**	
Unavailability of appropriate testing space and issues in creating the testing env.	Permits are not obtained for the testing sites to turn off fire alarms and launch drones			
Consequence				
This will cause delays in continuous testing and might cause problems in the final FVD plan.				
Mitigation Plan			**Expected Outcome**	
1. Keep accurate schedule of the availability of these spaces and prescribed burns. 2. Perform unit testing to not get stuck when a space becomes unavailable 3. Work towards a safe option at the MRSD allotted space for testing and showcase.			Unit testing reinforces our system integration process and accurate scheduling keeps us on track to perform tests. A safer MRSD space option is to reduce the consequence.	

Table 12: Risk ID 5: Supply Chain Issues/Lags

<table>
<tr><th>Risk Type</th><th colspan="2">Risk Owner</th><th>Date Submitted</th><th>Date Updated</th></tr>
<tr><td>Schedule, Cost, Technical, Programmatic</td><td colspan="2">Owner Name</td><td>12/01/2023</td><td>12/01/2023</td></tr>
<tr><th>Description</th><th colspan="2">Cause</th><th colspan="2">Consequence vs Likelihood</th></tr>
<tr><td>Supply chain issues while ordering parts.</td><td colspan="2">Parts which need to be shipped from overseas but cannot due to permissions/ or some other reason</td><td colspan="2" rowspan="2"> 
</td></tr>
<tr><th colspan="3">Consequence</th></tr>
<tr><td colspan="3">If it's a integral part, it could cause us to not meeting our PRs and add more delay in system testing.</td></tr>
<tr><th colspan="3">Mitigation Plan</th><th colspan="2">Expected Outcome</th></tr>
<tr><td colspan="3">1. Freeze hardware design ahead of time as hardware is an essential component of our system
2. Explore alternate hardware options beforehand
3. Order essential parts beforehand</td><td colspan="2">Gives us an estimate of what all hardware components we need if the designed is frozen early, and what parts can be replaced with other options.</td></tr>
</table>