ANAV Qualification Round – Design Report

Team: *AetherX*

Date: *April 20th, 2025*

1. Description of ANAV

Overview of ANAV:

The Aerial Navigation Autonomous Vehicle (ANAV) is a quadrotor-based unmanned aerial system engineered explicitly for autonomous GPS-denied operations. Built on the robust Cube Pilot Orange flight controller, ANAV is designed to achieve stable and precise vertical take-off, sustained hovering, and pinpoint landing within a bounded area.

The system integrates high-performance EMAX 935KV brushless motors and carbon fiber propellers powered by a 5200mAh 4S LiPo battery, delivering efficient lift and control. The ANAV is equipped with the CubePilot HereFlow Optical Flow sensor and TFMini LiDAR for precise height estimation and horizontal motion tracking to enable accurate localization and motion control without reliance on GPS. These sensors feed high-fidelity motion data to the onboard Raspberry Pi 5, which acts as the companion computer for autonomous decision-making and vision-based processing.

Flight stability and position control are achieved through sensor fusion using data from the IMU, barometer, and CubePilot HereFlow Optical Flow sensor. The system supports both manual and autonomous modes, allowing seamless transition and mission execution upon a single auto-start command from the base station.

Safety is a top priority, with real-time monitoring of battery levels, positional drift, and system health. In the event of any abnormal condition—such as low battery, communication loss, or sensor error—the ANAV triggers safe landing protocols without requiring manual intervention.

Designed to meet the qualification round requirements, ANAV ensures high reliability, precision, and autonomy, even in challenging environments where GPS is unavailable.

System Architecture:

ANAV's architecture consists of multiple subsystems working together seamlessly. These include:

1. **Flight Control System:**
    - CubePilot Orange: Ensures flight stabilization, attitude control, and sensor integration.
    - CubePilot HereFlow Optical Flow Sensor: Provides accurate horizontal motion tracking and velocity estimation in GPS-denied environments.
    - TFMini LiDAR: For accurate height estimation and landing support.
    - Raspberry Pi 5: Handles high-level processing, mission logic, and AI-based decision-making.
    - Electronic Speed Controllers (ESCs): Enable precise and efficient control of brushless motors.
2. **Sensing and Perception:**
    - LiDAR for depth sensing and obstacle avoidance.
    - HereFlow Optical Flow (with integrated IR-based LiDAR) for hover stabilization.
    - TFMini LiDAR to assist in maintaining altitude and smooth descent.
3. **Navigation and Planning:**
    - Predefined command sequence for:
      • Vertical takeoff
      • Hovering at 3–10 meters
      • Controlled landing at the takeoff point
    - Sensor fusion for real-time position hold using IMU, barometer, and HereFlow data.
    - Failsafe mechanisms for link-loss or low battery-induced auto-landing.
4. **Communication and Data Handling:**
    - Base station telemetry for real-time monitoring.

- Wireless communication link to issue a single auto-start command.
- Live data logging (if required) for post-flight analysis.

5. **Power and Propulsion:**
    - Emax motors with electronic speed controllers (ESCs) for stable flight.
    - Electronic Speed Controllers (ESCs) for efficient motor handling.
    - Lithium-polymer (LiPo) battery pack optimized for a ~20-minute flight duration.
    - Power distribution board for efficient energy management.

Functional Flow:

The ANAV system follows these key operational steps:

1. **Takeoff:**
    - Initiated by a single auto-start command from the base station.
    - Controlled vertical ascent using CubePilot Orange with data from IMU and barometer.
    - Height estimation aided by TFMini LiDAR to ensure smooth and accurate takeoff to 3–10 meters.
    - Flight control algorithms maintain thrust balance and attitude during ascent.
2. **Hovering :**
    - Maintains stable position at target altitude for a minimum of 30 seconds.
    - HereFlow Optical Flow Sensor ensures accurate horizontal hold in a GPS-denied environment.
    - IMU and barometer data fused for precise control and drift correction.
3. **Landing :**
    - Initiates automatic descent after the hover phase.

- Uses TFMini LiDAR for accurate ground distance estimation.
- Lands smoothly within the defined 1.2 m × 1.2 m home position boundary.
- Includes safety checks during descent to handle low battery or signal loss.

Subsystem Integration and Workflow:

To ensure smooth coordination among various subsystems, the ANAV follows a streamlined and modular integration approach focused on stable autonomous flight. The workflow includes:

- **Sensor Data Processing:** IMU, barometer, CubePilot HereFlow Optical Flow, and TFMini LiDAR data are processed in real-time to estimate orientation, position, and altitude.
- **Altitude and Position Hold :** The flight controller fuses data from the barometer and TFMini LiDAR to maintain accurate altitude. Simultaneously, the HereFlow sensor supports steady horizontal positioning.
- **Autonomous Flight Execution:** Upon a single auto-start command, the ANAV initiates vertical takeoff, maintains a 3–10 meter hover for 30 seconds, and executes a smooth landing—all autonomously using pre-programmed control logic.
- **Failsafe Mechanisms:** Battery level monitoring and communication link checks are continuously performed. If low voltage or link loss is detected, the ANAV triggers an autonomous safe landing procedure to prevent damage.

Environmental Adaptability:

Since ANAV operates in a simulated Martian-like terrain, it is designed to remain stable and functional in challenging, GPS-denied environments. The current prototype integrates key adaptability features relevant to autonomous takeoff, hovering, and landing:

- **Altitude Stability on Uneven Surfaces:** Using the TFMini LiDAR, the system measures accurate distance from the ground to maintain safe and smooth vertical ascent and descent—even if the surface is mildly uneven.

- **Position Hold in Low-Contrast Environments:** The CubePilot HereFlow Optical Flow Sensor, which includes infrared-based tracking, allows for stable hovering even over low-texture surfaces or in low-light conditions, mimicking Martian dust-laden environments.
- **Sensor Fusion for Real-Time Corrections:** Data from the IMU, barometer, LiDAR, and optical flow are fused to maintain a stable hover and responsive control loop in the absence of external navigation aids like GPS.
- **Emergency Handling and Fail-Safe Protocols:** In the event of link loss, low battery, or unexpected instability, the ANAV initiates a pre-programmed safe landing routine to minimize risk and ensure system recovery.

2. *Components with Their Specifications*

1. F450 / Q450 Quadcopter Frame (PCB Version with Integrated PCB) + Plastic Landing Gear Combo Kit

Fig 1. F450 / Q450 QUADCOPTER FRAME

Key Specifications:

- **Wheelbase:** 450 mm
- **Weight:** Approximately 330 grams
- **Arm Dimensions:** 220 x 40 mm (Length x Width)

- **Motor Mounting Hole Diameter:** 3 mm
- **Landing Gear Material:** ABS Plastic
- **LandingGearHeight:**200mm

2. EMAX MT2213 935KV Brushless DC Motor Black Cap (CW Motor Rotation) With Prop1045 Combo (Original)

Key Specifications:

- **Model:** MT2213
- **KV:** 935 KV
- **ESC:** 18A
- **Thrust:** 860G
- **Propeller:** 8″-10″
- **Diameter:** 28 mm
- **Length:** 39.5 mm
- **Weight:** 53 gm

Fig 2. BRUSHLESS DC MOTOR

Voltage (V)	Throttle (%)	Thrust (g)
11.1V (3S)	50%	~290-300g
11.1V (3S)	100%	~760-800g
14.8V (4S)	50%	~400-450g
14.8V (4S)	100%	~950-1000g

3. Orange HD Propellers 1045(10X4.5) Carbon Fiber Black 1CW+1CCW-1pair

Key Specifications:

- Length: 10″.
- Pitch: 4.5″.
- Weight: 30 gm.
- Shaft Diameter: 7.7 mm.
- Total Length: 10 inch / 250 mm
- Material: Carbon Fiber

Fig 3. PROPELLERS

4. Raspberry Pi 5 (8GB)

The **Raspberry Pi 5 (8GB)** is the latest iteration in the Raspberry Pi series, offering significant enhancements over its predecessors. Here's an overview of its key features:

Key Specifications:

- **Processor:** 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU
- **Memory:** 8GB LPDDR4X-4267 SDRAM
- **Graphics:** 800MHz VideoCore VII GPU
- **Display Support:** Dual 4Kp60 HDMI output
- **Connectivity:** Gigabit Ethernet, Wi-Fi 802.11ac, Bluetooth 5.0
- **Storage:** MicroSD card slot
- **USB Ports:** Two USB 3.0 and two USB 2.0 ports
- **Power Supply:** USB-C 27W PD

Fig 4. RASPBERRY PI 5 (8GB)

5. Hex Pixhawk Cube+ Flight Controller Autopilot

Key Specifications:

- Model: HX4-06222
- Input voltage: 3.3 V / 5 V
- Sensors: Accelerometer, Gyroscope, Compass, Barometric Pressure Sensor

Fig 5. CUBE PILOT

6. Orange 14.8V 5200mAh 40C 4S Lithium Polymer Battery Pack

Key Specifications:

- Model No: ORANGE 42999/4S-40C
- Weight:488 gm.
- Voltage: 14.8V.
- Constant Discharge: 40C.
- Max discharge: 50C (10 sec).
- Balance Plug: JST-XH.
- Discharge Plug: XT60.

Fig 6. BATTERY

7. ReadytoSky 40A 2-4S ESC for Drone

Key Specifications:

- 40A OPTO 2-4S Brushless ESC.
- The applicable number of battery cells: 2-4S.
- BEC output: 5V 3A.
- Use for F450 450mm S500 ZD550 RC Helicopter Quadcopter.
- Highly intelligent, has adaptive ability, extremely easy to use.

Fig 7. ESC (Electronic Speed Controller)

8. Connectors and Wires (XT60 and 16 AWG)

- **Role:** Ensure reliable electrical connections.
- **Key Specifications:** XT60 connectors for secure power connections, 16 AWG wires for power delivery.
- **Working:** The XT60 connectors securely link the battery to the power distribution board, while the 16 AWG wires carry current to the motors, Raspberry Pi, and other components, ensuring efficient power transmission.

Fig 8. CONNECTORS AND WIRES (XT60 AND 16 AWG)

9. APM/Pixhawk Power Module Output BEC 3A XT60 Connector 28V 90A

Key Specifications:

1. **Power Supply:**
    - Provides regulated 5.3V (3A BEC output) to power the flight controller.
    - Supports up to 6S LiPo batteries (28V max input).
2. **Current and Voltage Sensing:**
    - Measures battery voltage (0-28V) and current (up to 90A).
    - Data is sent to the flight controller for monitoring in Mission Planner (ArduPilot) or QGroundControl (PX4).
3. **XT60 Connectors:**
    - Pre-soldered XT60 male/female connectors for easy battery and ESC connections.
4. **High Current Handling:**
    - Rated for 90A continuous current, making it suitable for high-power drones and UAVs.
5. **Compatibility:**
    - Works with APM 2.5 / 2.6 / Pixhawk 1 / Pixhawk 2 (Cube) / PX4 flight controllers.
    - Compatible with ArduPilot and PX4 firmware.

Fig 9: Power Module

10. TFMini-S Micro LiDAR Distance Sensor

Key Specifications:

- Small size
- Lightweight
- Low power consumption
- High frame rate(up to 1000Hz)

Fig 10.TFMini-S MICRO LiDAR

11. Hereflow Optical Flow Sensor

Fig 11. CUBEPILOT HEREFLOW OPTICAL FLOW SENSOR

Specifications of CubPilot HereFlow Optical Flow Sensor:

Optical Flow Sensor:

- **Sensor Type:** PMW3901
- **Effective Range:** 80mm to infinity
- **Field of View:** 42°
- **Maximum Movement Speed:** 7.4 radians/second
- **Minimum Illumination:** 60 lux
- **Infrared Emitter:** 940nm invisible light emission (Class 1)

LiDAR Module:

- **Measuring Frequency:** Up to 50 Hz
- **Field of View:** 27°
- **Maximum Distance:** 2 meters
- **Accuracy:** ±3%
- **Infrared Emitter:** 940nm invisible light emission (Class 1)

Total Estimated Weight Calculation:

1. Drone Frame (DJI F450): 380g
2. Motors (MN2212): 560g (for 4 motors)
3. Propellers: 56g (for 4 propellers)
4. Raspberry Pi 5: 46g
5. CubePilot: 45g
6. Battery (5200mAh 4S LiPo): 350g
7. ESC: 120g (for 4 ESCs)
8. LiDAR: 170g
9. Camera Module :25g
10. Power Distribution Board: 40g
11. Connectors and Wires (XT60 and 16 AWG): 30g

12. TFMini-S Micro LiDAR: 5g
13. CubePilot HereFlow Optical Flow Sensor: 1.2g

These weights give a total of approximately 1,828g (1.828 kg) for the drone components.

3. Flow chart of Algorithms and sensing methods:

Autonomous Mode – ANAV System

This document provides a detailed breakdown of the ANAV system's operation in Autonomous Mode. The flowchart below captures each phase in the sequence, highlighting the onboard sensors used for safe and efficient navigation in a GPS-denied environment.

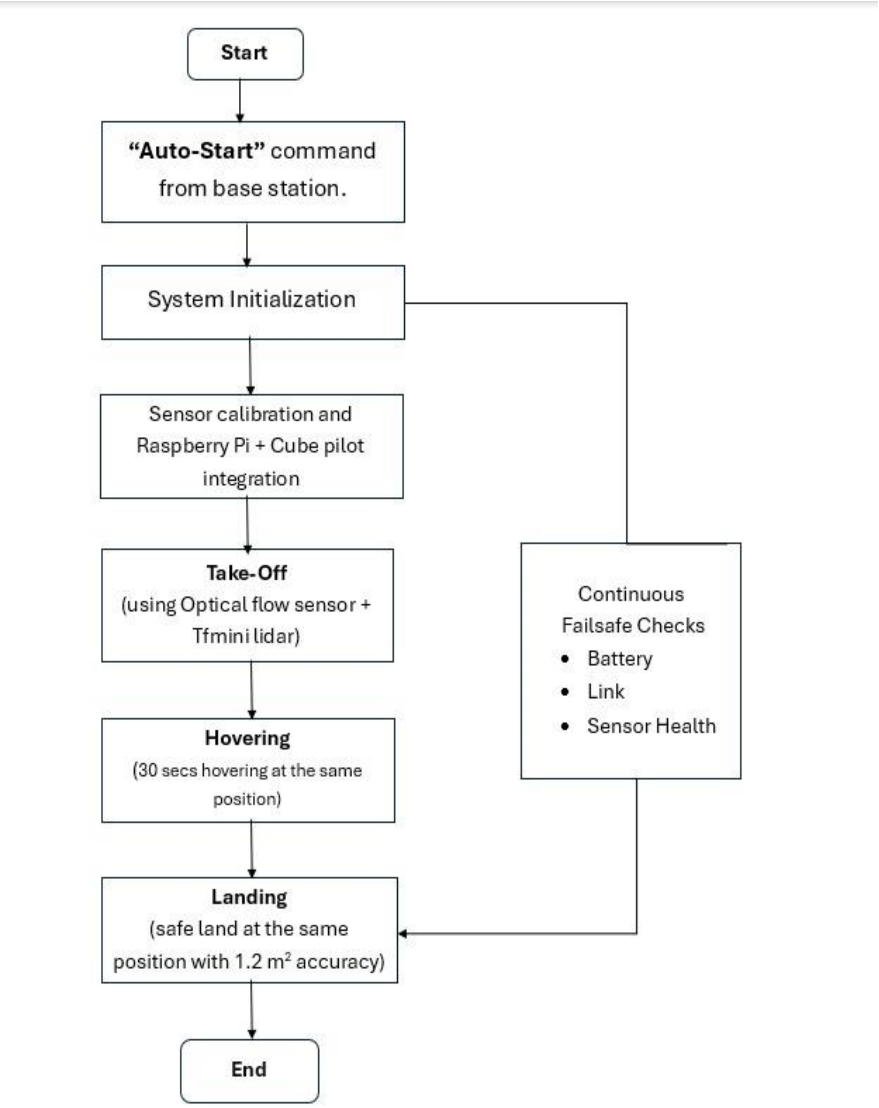

Figure: Autonomous Mode Flowchart with Sensor Integration

1. Start

This is the beginning of the drone operation sequence. It indicates that the system is ready to accept commands and perform autonomous tasks. At this stage, the drone is either powered on or already on standby.

2. "Auto-Start" Command from Base Station

- The Base Station (a computer or ground control system) sends a command to initiate the drone's automatic flight process.
- This command tells the drone to begin its pre-defined mission, which includes taking off, hovering, and landing autonomously.
- It eliminates the need for manual piloting.

3. System Initialization

- All the necessary systems on the drone start booting up.
- This step includes initializing:
  - Flight controllers (like CubePilot)
  - Companion computer (like Raspberry Pi)
  - Communication modules
  - Power distribution systems
- It ensures all components are online and ready for flight.

4. **Sensor Calibration and Raspberry Pi + CubePilot Integration**
   - The sensors (especially the ones used for flight stability and navigation) are calibrated.
   - The Raspberry Pi (used for higher-level tasks like image processing, AI, or scripting) is integrated with the CubePilot (flight controller that handles real-time flight).
   - This integration ensures data is shared between the high-level and low-level systems, allowing better control and monitoring.
   - The Raspberry Pi (used for higher-level tasks like image processing, AI, or scripting) is integrated with the CubePilot (flight controller that handles real-time flight).
   - This integration ensures data is shared between the high-level and low-level systems, allowing better control and monitoring.
   - The drone takes off autonomously using:
     - CubePilot HereFlow Optical Flow Sensor: Tracks horizontal movement using PMW3901 along with integrated IR-based

LiDAR for ground velocity estimation and distance measurement.
    - TFMini Lidar: Measures the distance to the ground with high precision. Crucial for altitude control during take-off.
    - These sensors ensure a stable and controlled ascent even in indoor or GPS-denied environments.

After take-off, the drone hovers at the same position for 30 seconds

Purpose:

- To check positional accuracy and stability.
- To validate sensor feedback in real-time.
- To simulate conditions of a stationary task (like inspection or surveillance).

Hovering at the same spot requires consistent altitude and position control, showcasing how well the drone can maintain stability.

5. Landing (Safe Land at the Same Position with 1.2 m² Accuracy)

- After hovering, the drone attempts a precise landing at the same position where it took off.
- The landing process aims for an accuracy of 1.2 square meters, meaning it should land within a 1.2m x 1.2m area.
- Sensors guide the descent and ensure the drone touches down gently without damage.
- Parallel Process: Continuous Failsafe Checks While all the above steps are happening, the drone continuously monitors key parameters to ensure safety. If any issue arises, the mission can be aborted, or an emergency landing can be triggered.
- Battery: Ensures there is enough power to complete the mission and land safely.
- Link: Checks if the communication link with the base station is stable.
- Sensor Health: Ensures all essential sensors are working correctly. If any fails (e.g., lidar or optical flow), it may trigger a safety procedure.

6. End

- Marks the successful completion of the mission.

- The drone has taken off, hovered, and landed safely, and all systems have remained healthy throughout. System Architecture Overview The figure below illustrates the hardware architecture of the ANAV system used during Autonomous Mode. It includes all major components such as the flight controller (Cube Pilot), onboard computer (Raspberry Pi), and various sensors (Camera Module, Optical Flow Sensor, RPLIDAR, TF Mini Lidar), as well as power management units. This block diagram represents the hardware architecture of an autonomous drone system, showing how various components are interconnected. Let's break it down component by component and explain each part in detail:
    1. Battery
        - Primary power source for the entire drone system.
        - Powers both the motors (through ESCs and power distribution board) and onboard electronics (via UBEC and power module).
        - Likely a LiPo battery, commonly used in drones due to high energy density.
    2. Power Distribution System
        - Power Distribution Board (PDB): o Distributes electrical power from the battery to the Electronic Speed Controllers (ESCs) that drive the motors.
            - Also supplies power to the Cube Pilot and other systems via the Power Module.
        - UBEC (Universal Battery Elimination Circuit, 5A):
            - Provides a regulated 5A power supply to sensitive electronics (e.g., Raspberry Pi, Cube Pilot).
            - Converts high battery voltage to a safe level.
        - Power Module:
            - Supplies regulated power to Cube Pilot.
            - Provides voltage and current sensing feedback to the flight controller.
    3. Cube Pilot (Flight Controller)
        - The central flight control system of the drone.

- Responsible for:
    - Stabilization and orientation control (pitch, roll, yaw)
    - Reading sensor inputs
    - Commanding motor speeds via ESCs
    - Executing autonomous missions
    - Safety and failsafe routines
- Receives data from sensors like:
    - TFmini lidar (altitude)
    - Optical Flow sensor (position tracking)
- Communicates with Raspberry Pi for mission planning, computer vision, or advanced decision-making.

4. Raspberry Pi (Companion Computer)
    - Acts as a companion computer for advanced processing tasks.
    - Typically used for:
        - Running high-level algorithms (e.g., AI, image processing)
        - Communicating with the base station via 4G/Wi-Fi
        - Handling camera input and external sensors like RPLidar
    - Communicates with the Cube Pilot via serial or MAVLink.
5. Camera Module
    - Connected to the Raspberry Pi.
    - Used for:
        - Visual tracking
        - Navigation (e.g., object detection, line following)
        - Image or video recording
    - Could also be used in real-time decision making if paired with AI models.
6. Hereflow Optical Flow Sensor
    - Measures the drone's movement relative to the ground by tracking surface patterns.
    - Useful in GPS-denied environments.

- Helps maintain position during hover and slow movement.

7. TFmini Lidar
    - A compact range-finding sensor for altitude measurement.
    - Provides precise vertical distance from the ground (up to a few meters).
    - Ensures accurate takeoff, hovering, and landing.
8. ESCs (Electronic Speed Controllers)
    - ESCs control the speed and direction of each motor.
    - Receive PWM signals from the Cube Pilot.
    - One ESC for each motor (4 in total for a quadcopter). 10. Motors (Motor 1 to Motor 4)
    - Brushless DC motors responsible for generating lift and thrust.
    - Controlled via ESCs based on commands from the Cube Pilot.
    - Work in coordination to maintain stability and maneuver the drone.
    - How Everything Works Together

      1. Battery powers everything.
      2. Cube Pilot manages real-time flight based on input from lidar, optical flow, and other sensors.
      3. Raspberry Pi handles advanced processing and may give high-level commands to Cube Pilot.
      4. Camera & RPLidar provide environmental awareness.
      5. ESCs and Motors receive flight control commands and drive the drone's motion.
      6. Failsafe features are typically handled in Cube Pilot with feedback from all sensors and the power module.

4. ANAV realization

The Autonomous Navigation Aerial Vehicle (ANAV) has been realized with a refined set of components to ensure optimal performance in non-GPS environments. The upgrades from the proposed components were made to enhance processing power, stability, and accurate localization. Below is a detailed specification of the current setup used in the qualification round:

Hardware Realization:

S. No.	Component (Proposal)	Component (Final Implementation)	Remarks
1	**T-Motors (MN2212)**	Emax 935kV motors	Changed due to availability and thrust efficiency
2	**DJI F450 Drone Frame**	DJI F450 Drone Frame	No change
3	**Carbon Fiber Propellers**	Carbon Fiber Propellers	No change
4	**Raspberry Pi 4**	Raspberry Pi 5	Upgraded for improved processing power
5	**8000mAh 3S LiPo Battery**	5200mAh 4S LiPo Battery	Changed for weight and voltage efficiency
6	**Hobbywing ESC (40A)**	Hobbywing ESC (40A)	No change
7	**TFmini-S LiDAR**	TFmini-S LiDAR	No change
8	**Camera Module 2**	Camera Module 3	Upgraded for better image processing
9	—	CubePilot Hereflow Optical Flow Sensor	Newly added for GPS-denied navigation

10	—	CubePilot Orange Flight Controller	Newly added for high-end autopilot capabilities
11	**RPLiDAR A1M8 360°**	RPLiDAR A1M8 360°	No change

Summary of Changes from Proposal

- **Motors:** Switched to Emax 935KV for better thrust-to-weight performance.
- **Processor:** Upgraded to Raspberry Pi 5 for enhanced computational power, aiding real-time vision and control.
- **Battery:** Optimized to 4S 5200mAh to match current draw and reduce weight.
- **Camera:** Camera Module 3 provides better image processing for vision-based tasks.
- **Added Sensors:** CubePilot Hereflow was added to improve localization in GPS-denied environments.
- **Raspberry Pi 5:** The Raspberry Pi 5 features an ARM Cortex-A76 quad-core processor, significantly improving computational performance for real-time processing tasks such as path planning, obstacle avoidance, and sensor data fusion, Enhanced RAM and Connectivity, and Improved Power Efficiency.
- **CubePilot Orange:** Advanced Features and Customization, Enhanced Autopilot Capabilities, Wider Compatibility.
- **LIDAR Integration:** A 360° LIDAR was chosen to ensure precise mapping and obstacle detection, crucial for GPS-denied environments.

Software Realization:

ANAV's software architecture is designed around modularity and autonomy. Each component plays a specific role in sensing, processing, or actuation.

1. System Architecture

The software stack runs on **Raspberry Pi 5** and includes:

- **Operating System:** Raspberry Pi OS 64-bit
- **Development Tools:** Python and C++
- **Middleware:** ROS Noetic and MAVROS for communication with CubePilot
- **Libraries/Protocols:** MAVLink for communication with Pixhawk via MAVROS

2. ROS Nodes and Functionality

The software is divided into specialized ROS nodes:

- **Sensor Node:** Handles data from TFMini LiDAR, IMU, and optical flow sensor
- **Flight Control Node:** Publishes **takeoff, hover, and landing setpoints** using MAVROS
- **Mission Manager Node:** Executes a finite-state sequence to complete the Qualification Round tasks autonomously

3. Navigation and Control

For the Qualification Round, the movement is predefined and limited to vertical commands:

- **Takeoff Logic:** Controlled ascent to 3 meters using setpoint_position/local via MAVROS
- **Hovering:** Maintains position for 30 seconds using a timer and position hold
- **Landing:** Smooth descent initiated based on time or altitude threshold
- **Failsafe Handling:** Timeout detection, battery monitoring, and link-loss triggers safe landing

4. ROS Integration with Pixhawk via MAVROS

- **Setpoints:** Published to /mavros/setpoint_position/local for altitude control
- **Flight Modes:** Managed through ROS services (arming and switching to OFFBOARD mode)
- **Monitoring:** Real-time telemetry data from Pixhawk monitored via MAVROS topics
- **Failsafes:** System monitors MAVLink heartbeat and triggers landing if anomalies are detected

5. Mission Sequencing and State Management

A simple finite-state machine governs the mission:

- **Framework:** SMACH-based (ROS finite-state machine)
- **Sequence:** IDLE → TAKEOFF → HOVER → LAND → COMPLETED
- **Triggering:** A single ROS service command initiates the entire mission flow
- **Monitoring:** Real-time status via ROS logs and rqt_graph for debugging

6. Qualification Round Integration

The following sequence is designed for the Qualification Round demo:

- **Launch Sequence:** One ROS launch file activates all required nodes and sets OFFBOARD mode
- **Takeoff:** Ascends to the target altitude (3–10 meters)
- **Hover:** Holds position for a minimum of 30 seconds
- **Landing:** Returns to the original takeoff position and lands smoothly
- **Shutdown:** Automatically disarms upon successful landing

This ROS-based, resource-efficient software stack ensures that ANAV autonomously and reliably performs takeoff, hovering, and landing with **a single command**, fully meeting the IRoC-U 2025 Qualification Round requirements.

5. *Realized ANAV Specification*

Parameter	Specification
Frame	DJI F450 Drone Frame
Motors	Emax 935kV Brushless Motors (Quad configuration)
ESCs	Hobbywing 40A
Propellers	Carbon Fiber
Battery	5200mAh 4S LiPo
Flight Controller	CubePilot Orange
Companion Computer	Raspberry Pi 5
Camera	Raspberry Pi Camera Module 3
Altitude Sensing	TFmini-S LiDAR
Velocity Estimation	HereFlow Optical Flow Sensor
SLAM & Mapping	RPLidar A1M8 (360° Scanning Lidar)
Navigation Type	GPS-denied indoor navigation
Control Communication	MAVLink between Pixhawk and Raspberry Pi
Navigation Algorithms	Visual Odometry, Optical Flow, SLAM, Obstacle Avoidance
Flight Time	Approx. 10–12 minutes (depending on payload and maneuvering)
Failsafe Mechanisms	Obstacle detection, emergency landing, out-of-bound handling

6. Tests and outcomes:

Test 1: Calibration & Stability Test

Objective:

To verify if all components (motors, ESCs, flight controller, and transmitter) are correctly calibrated and to assess the stability of the drone without GPS.

Test Setup:

- **Controller Used:** FlySky
- **Hardware:** DJI F450 Drone, Raspberry Pi 5, ESCs, Motors
- **Software:** Mission Planner
- **Test Conditions:** Indoor/Outdoor, Wind conditions, etc.

Procedure:

1. Powered on the drone and performed pre-flight checks.
2. Tested the throttle response and motor synchronization.
3. Attempted to hover at a fixed altitude manually.
4. Observed drone stability in different throttle conditions.

Results & Observations:

- Initially, the drone was unable to maintain a steady altitude.
- Stability issues were observed due to the lack of GPS-assisted flight.
- Manual control required continuous adjustments.

Analysis & Fixes:

- Issue Identified: Instability due to the absence of GPS hold.
- Solution Implemented:
    - Adjusted flight control parameters to operate without GPS.
    - Achieved altitude hold by changing flight modes in Mission Planner.
    - Stability was improved through PID tuning and mode selection.

Test 2: Fixing Stability & Achieving Altitude Hold

Objective:

To apply the fixes from Test 1 and verify that the drone can maintain stability and altitude without manual intervention.

Test Setup:

- Controller Used: FlySky
- Hardware: DJI F450 Drone, Raspberry Pi 5, ESCs, Motors
- Software: Mission Planner

Procedure:

1. Applied new tuning parameters to the flight controller, ensuring proper calibration.
2. Conducted a test flight under controlled conditions, observing for signs of instability.
3. Activated Altitude Hold (ALTHOLD) mode and attempted to maintain 3 meters altitude.
4. Observed the drone's ability to maintain a steady hover over time.
5. Checked motor and ESC responses under sustained hover conditions.

Results & Observations:

- The drone successfully maintained altitude without major stability issues.
- The previous instability issues were resolved after tuning the flight parameters.
- The system was ready to proceed with autonomous hovering tests.
  Minor issue noted: Some drifting was observed, likely due to air currents.

Analysis & Fixes:

- Solutions Implemented:
  - Adjusted mode settings to enhance stability.
  - Fine-tuned PID values to ensure smoother altitude hold.
  - Conducted multiple tests to verify hover reliability.

Test 3: Initial Autonomous Hovering Test

Objective:

To test autonomous takeoff and hovering using Raspberry Pi 5 as the main processor, with communication to the Cube flight controller via a telemetry-to-USB connection.

Test Setup:

- Processor Used: Raspberry Pi 5
- Flight Controller: Cube (connected via Telem-USB)
- Software & Libraries Installed: MAVSDK, DroneKit
- Planned Flight:
    - Autonomous takeoff to 3 meters
    - Maintain hover for 10 seconds (initial test)

Procedure:

1. Installed required libraries and set up Raspberry Pi 5 for flight control.
2. Connected the Cube flight controller to the Raspberry Pi via Telem-USB.
3. Executed the autonomous takeoff command to reach 3 meters.
4. Observed drone behavior after takeoff.

Results & Observations:

- The drone successfully took off to 3 meters.
- However, instead of hovering, it immediately fell down after reaching the target altitude.
- Issue identified: The Altitude Hold (ALTHOLD) mode was not set in the flight script.

Analysis & Fixes:

- Root Cause: The script lacked an altitude-hold command, causing the drone to descend after reaching the target height.
- Solution Implemented:
    - Updated the script to include ALTHOLD mode after takeoff.

- Adjusted flight parameters to ensure stable hover.

Test 4: Implementing Altitude Hold & Stability Test

Objective:

To fix the issue from the previous test by enabling Altitude Hold (ALTHOLD) mode in the autonomous flight script and verifying stable hovering for 10 seconds.

Test Setup:

- Processor Used: Raspberry Pi 5
- Flight Controller: Cube (connected via Telem-USB)
- Flight Mode: Takeoff → Altitude Hold (3m) → Hover for 10s → Land

Results & Observations:

- The drone successfully hovered for 10 seconds.
- However, during landing, the throttle suddenly dropped, causing the drone to flip upon touchdown.
- A motor was damaged due to the hard landing.

Analysis & Fixes:

- Issue Identified: Sudden throttle drop during landing, leading to an unstable descent.
- Solution Implemented:
  - Modify the landing sequence to gradually reduce the throttle instead of an abrupt cut.
  - Implement a controlled descent algorithm to ensure a smooth landing.
  - Replace the damaged motor before the next test.

Test 5: Gradual Safe Landing & 30-Second Hover Test

Objective:

To achieve:

1. Stable hover for 30 seconds at a fixed altitude.
2. Smooth and controlled landing without sudden throttle drops.

Results & Observations:

- The drone successfully hovered for 30 seconds.
- The landing was smooth, preventing any damage.
  Minor Issue noted: The drone drifted with air currents instead of holding position.

Next Steps:

- Implement Position Hold (POSHOLD) or another stabilization method.
- Conduct another test to evaluate position stability.

Test 6: Final Autonomous Hover and Safe Landing (Loiter Mode + Optical Flow)

Objective:
To demonstrate autonomous takeoff, 30-second stable hover, and smooth landing using **Loiter mode**, supported by **HereFlow Optical Flow Sensor** and **TFMini LiDAR**, without GPS.

Test Setup:

- **Flight Controller:** CubePilot Orange
- **Mode:** Loiter (GPS disabled)
- **Sensors:** CubePilot HereFlow + TFMini LiDAR
- **Processor:** Raspberry Pi 5
- **Platform:** DJI F450 Frame
- **Software:** Mission Planner + MAVROS
- **Environment:** Indoor with low contrast flooring and light air currents

Procedure:

1. Initiated flight with a single auto-start command.
2. Drone autonomously took off to a height of 3 meters.
3. Engaged **Loiter mode** for positional hold using optical flow.
4. Maintained stable hover for 30 seconds.
5. Executed a controlled descent and safe landing.

Results & Observations:

- Takeoff and hover were smooth and precise.
- The drone maintained position with minimal drift.
- Landing was soft and stable within the 1.2 × 1.2 m home boundary.
- Optical flow ensured accurate horizontal hold in a GPS-denied indoor setup.
- Minor drift occurred due to indoor airflow, but remained within safe limits.

Analysis & Fixes:

- **Improvement:** Switched from ALTHOLD to **Loiter mode** for enhanced stability.
- **Stability Source:** Optical flow + IR sensing from HereFlow enabled steady position hold.
- **Altitude Accuracy:** Achieved using LiDAR feedback instead of barometer alone.
- **Failsafe Ready:** Battery and link health monitored during entire mission.

6. Tasks and outcomes

Throughout the development and testing phase of ANAV, several tests were conducted to evaluate the functionality and performance of key components, particularly the motors, ESCs, and power systems. This section outlines the key issues encountered, the solutions implemented, and the outcomes after each test.

Test 1: Motor Thrust Balance and Calibration

Objective:

Verify motor calibration and ensure that each motor generates balanced thrust during flight for stable hovering and maneuverability.

Problem Encountered:

One of the motors was spinning faster than the others, leading to yaw instability and unwanted rotation during hover and low-speed maneuvers. The drone was unable to maintain a stable hover despite a steady throttle input.

Cause:

The issue was traced to improper ESC calibration, causing an imbalance in power distribution among the motors. Additionally, slight differences in motor characteristics contributed to the problem.

Solution Implemented:

1. ESC Recalibration: A full recalibration was performed using CubePilot software to synchronize power distribution.

2. Motor Matching: Each ESC was paired with the correct motor and tested for uniform rotation speed.

3. Wiring Check: All wiring connections were inspected to eliminate inconsistencies in power delivery.

Outcome:

After recalibration, the yaw instability was resolved, and the drone achieved stable hovering with balanced thrust across all motors.

Test 2: ESC Performance Under High Throttle Conditions

Objective:

Evaluate ESC response to high-current demands during rapid throttle changes and high-speed maneuvers.

Problem Encountered:

During flight, the drone experienced motor stalling when throttle was rapidly increased or during sharp turns, leading to temporary power loss and flight instability.

Cause:

ESC settings were not optimized for handling high current draw, causing power delivery issues during aggressive maneuvers.

Solution Implemented:

1. ESC Firmware Update: Updated the firmware to improve high-current handling.

2. Throttle Curve Adjustment: Fine-tuned ESC parameters to ensure smooth power transitions.

3. Dynamic Testing: Conducted additional flight tests to verify stability under high-speed conditions.

Outcome:

The motor stalling issue was resolved, and the drone executed high-speed maneuvers smoothly with improved flight stability.

Test 3: Propeller Balance and Thrust Consistency

Objective:

Ensure propellers are balanced to generate consistent thrust and prevent instability during flight.

Problem Encountered:

The drone exhibited tilting and yawing during hover, indicating thrust imbalance.

Cause:

Unbalanced propellers caused uneven thrust, leading to stability issues.

Solution Implemented:

1. Propeller Balancing: Used a balancing tool to correct weight distribution.

2. Propeller Alignment Check: Ensured proper mounting and alignment with motors.

Outcome:

The tilting and yawing issues were resolved, resulting in smoother and more controlled flight.

Test 4: Power Supply and Battery Performance

Objective:

Evaluate battery performance under high-load conditions and ensure consistent power delivery.

Problem Encountered:

The drone experienced voltage sag during high-speed maneuvers, causing momentary power loss.

Cause:

The battery could not handle the high current draw, leading to voltage dips and power interruptions.

Solution Implemented:

1. Battery Upgrade: Tested a higher-capacity 6000mAh battery for improved stability.

2. ESC Power Optimization: Adjusted ESC settings to regulate power usage and prevent excessive draw.

Outcome:

The voltage sag issue was significantly reduced, and the drone maintained consistent power even during aggressive maneuvers.

Conclusion

The testing phase identified and resolved several key challenges, leading to significant improvements in performance:

1. ESC Calibration: Balanced motor thrust, eliminating yaw instability and ensuring stable hovering.

2. ESC Performance Tuning: Optimized ESC response, preventing motor stalling and improving dynamic control.

3. Propeller Balancing: Eliminated thrust imbalance, enhancing flight stability.
4. Battery Performance Optimization: Upgraded the power system to ensure consistent power delivery during high-load conditions. These optimizations resulted in a reliable, stable, and responsive drone capable of executing mission objectives with improved control and efficiency.

8. Realized Emergency Response System

Autonomous drones, particularly those operating without GPS, are prone to unpredictable failures such as hardware malfunctions, battery drops, signal loss, or control drift. To ensure the safety of the drone, its environment, and nearby personnel, we implemented a comprehensive Emergency Response System (ERS). This system is designed to detect anomalies in real-time and immediately trigger predefined safety actions like stopping, landing, or returning to a safe zone.

Hardware Components of ERS

Component	Function in Emergency	Example Scenario
Flight Controller (Pixhawk)	Executes emergency protocols like auto-land or disarm	Auto-landing triggered on low battery or geofence breach

RC Transmitter	Manual disarm using kill switch	The operator stops drone manually during hover instability
Power Module	Monitors voltage/current, feeds to companion computer	Battery voltage monitoring for auto-land decision
LiDAR Sensor	Measures altitude accurately for safe hovering/landing	Maintain hover before safe descent
Companion Computer (Raspberry Pi)	Executes ROS nodes and custom emergency logic	Triggers autonomous land on communication timeout
Telemetry Module / WiFi	Maintains connection to GCS; detects link failure	Starts failsafe if heartbeat not received in 3 sec
Buzzer/LED Indicators	Provides audible and visual alerts for emergency conditions	Warns user of low battery or communication loss

Software Components of ERS

Software Component	Role in Emergency	Example Feature
PX4 Firmware	Built-in failsafe handlers for critical flight states	Auto-disarm on kill switch, auto-land on low battery
ROS (Robot Operating System)	Middleware to coordinate sensor data and emergency logic	Geofencing, battery monitor, event logger

MAVROS	Bridge between ROS and Pixhawk flight controller	Allows ROS to send /land, /disarm, and status read
Custom ROS Nodes	Implements specific safety behaviors and logic	battery_monitor.py, geofence_checker.py
Python/C++ Scripts	Trigger emergency commands based on logic or sensor thresholds	Command override to /land or log safety event
Ground Control Station (GCS)	Manual interface to send kill switch, land, and monitor telemetry	Used to stop drone or observe safety warnings
Logging Tools (rosbag, ulog)	Record flight data for debugging and post-mission analysis	Detect cause of emergency land or battery failure

Safety Features and Demonstrations

Feature	Problem Solved	How It Works	Demonstration Result
Manual Kill Switch	Unpredictable behavior during flight	RC/GCS sends disarm command to flight controller → immediate motor shutdown	Used during a slight mid-air tilt; the drone stopped instantly

Low Battery Auto-Land	Drone crash due to sudden battery drop	Voltage < 10.5V → triggers /land via ROS or PX4 → controlled descent	Drone landed before battery was depleted during intentional drain test
Signal Loss Handling	Loss of communication with the ground station	Heartbeat timeout > 3s → drone switches to auto-land state	The ground station turned off mid-flight. → The drone initiated a safe descent
Geofencing	Drone drifting outside safe area	Virtual square boundary (±1.5m) defined in software → If breached, drone commands landing	During trial, drone drifted out of zone → auto-corrected and landed
Emergency Event Logging	Difficult to trace cause of emergency events	ROS + rosbag logs battery, altitude, motor status, and event triggers	Used logs to identify a false landing caused by voltage misconfiguration → Threshold adjusted accordingly

System Flow (Example Scenario)

1. Takeoff in autonomous mode
2. Battery voltage drops → battery_monitor.py triggers /land
3. The drone begins its controlled landing
4. The operator observes erratic motion → uses the kill switch
5. The drone instantly disarms and stops mid-flight
6. After the mission, logs are reviewed via rosbag to analyze trigger conditions

Summary

The Realized Emergency Response System offers a layered approach to drone safety, integrating:

- Hardware-level kill switches and power monitors
- Software-based decision-making using ROS and custom logic
- Real-time telemetry and manual override via GCS
- Detailed logging for post-event analysis

This system ensures safety even in unpredictable or GPS-denied environments, fulfilling all the critical emergency requirements of the IRoC-U 2025 challenge.

9. Project Management: Responsibilities of Team Members

1. Project Manager

Responsibilities:

- Overall planning, execution, and delivery of the ANAV project.
- Setting timelines, assigning tasks, and tracking progress.
- Coordinating between departments (hardware, software, testing, etc.).
- Risk management and resource allocation.
- Communicating with stakeholders and preparing reports.

2. Hardware Lead / Mechanical Engineer

Responsibilities:

- Designing drone structure considering aerodynamics and payload.
- Selecting materials for durability and weight optimization.
- Integrating mechanical components like rotors, landing gear, and frame.
- Ensuring mechanical stability during flight.

3. Power Systems Engineer

Responsibilities:

- Selection and integration of battery systems and power distribution.
- Ensuring sufficient flight time with safety margins.
- Monitoring power usage and efficiency.
- Developing battery charging and management protocols.

4. AI/Navigation Engineer

Responsibilities:

- Developing and training AI models for obstacle detection and path planning.
- Implementing SLAM (Simultaneous Localization and Mapping) algorithms.
- Integrating computer vision with navigation logic.
- Testing autonomous response under real-time conditions.

5. Communication & IoT Systems Engineer

Responsibilities:

- Developing V2X (Vehicle-to-Everything) communication protocols.
- Ensuring stable and secure wireless communication (e.g., 4G/5G, LoRa).
- Integrating sensors and cloud-based data transmission.
- Real-time telemetry and command handling.

6. Embedded Systems / Software Developer

Responsibilities:

- Programming flight controller (e.g., PX4, ArduPilot).
- Writing low-level firmware for sensors and actuators.
- Debugging and maintaining onboard code.

- Ensuring safety features like auto-landing and fail-safes.

7. Testing & Validation Engineer

Responsibilities:

- Planning and conducting ground and aerial tests.
- Recording flight data and analyzing performance.
- Validating navigation accuracy and obstacle avoidance.
- Suggesting design improvements based on test outcomes.

8. Documentation & Presentation Lead

Responsibilities:

- Preparing technical documentation and reports.
- Creating project presentations, posters, and user manuals.
- Managing logs of design changes, test results, and configurations.
- Supporting in project review and submission stages.

Here's the responsibilities table for team members involved in the realization of ANAV (Autonomous Navigation Aerial Vehicle):

Role	Responsibilities
Project Manager	- Overall planning and execution - Task assignment and tracking - Risk and resource management - Stakeholder communication
Hardware Lead	- Design of drone structure and frame - Material selection - Integration of mechanical components - Ensuring structural integrity
Power Systems Engineer	- Battery selection and integration - Power distribution and management

	- Efficiency optimization - Safety protocols
AI/Navigation Engineer	- Path planning and obstacle avoidance using AI - SLAM implementation - Computer vision integration - Real-time decision making
Communication & IoT Engineer	- V2X communication setup - Sensor and cloud integration - Telemetry and command protocols - Wireless network configuration
Embedded Systems/Software Dev	- Flight controller programming (PX4, ArduPilot) - Sensor and actuator interfacing - Code debugging - Safety feature implementation
Testing & Validation Engineer	- Planning test scenarios - Conducting flight and ground tests - Data analysis - Feedback for design improvements
Documentation & Presentation	- Technical report preparation - Project documentation and user manuals - Visuals for presentation - Change logs and records management

10. Novelty in the realized ANAV

The realized ANAV system designed for the IRoC-U 2025 Qualification Round showcases several novel elements in both its implementation and architecture. One of the most impactful innovations is the ability to execute the entire flight mission—takeoff, hover, and landing—using a single command issued from the base station. This design enables a completely autonomous flight cycle with no need for manual intervention, which is crucial for remote or planetary missions where real-time human control is limited or entirely unavailable due to communication delays or signal loss.

The system is uniquely tailored to operate in GPS-denied environments, such as the surface of Mars, where traditional satellite-based positioning is not possible. To overcome this, the ANAV uses a combination of the CubePilot HereFlow Optical Flow Sensor and TFMini LiDAR to achieve precise localization and altitude control. The optical flow sensor estimates horizontal position and movement using infrared-assisted ground tracking, while the LiDAR provides real-time vertical distance measurements. These are fused with IMU and barometric data to deliver stable and accurate flight control even in low-contrast or featureless terrains, mimicking the unpredictable surface of extraterrestrial landscapes.

Rather than relying on complex and computationally intensive systems like SLAM or external markers, the system uses lightweight control strategies to achieve autonomous hovering and precise landing. The final mission script is optimized to perform takeoff to a height of three meters, maintain a stable hover for at least thirty seconds, and land smoothly within a defined home boundary—all without GPS or visual mapping..

To ensure mission safety and reliability, the ANAV incorporates robust fail-safe mechanisms. These include autonomous landing protocols triggered by low battery, link loss, or positional anomalies. The system is continuously monitored through telemetry and onboard logic, ensuring that any deviation from expected behavior is corrected immediately. Each stage of flight—idle, takeoff, hover, and land—is modularly defined and executed in a way that makes the system both fault-tolerant and easy to modify or extend for future phases of the competition.

11. Review report by first mentor:

The team has made significant progress in realizing the ANAV system, effectively moving from proposal to implementation with a clear understanding of system-level integration. The decision to eliminate GPS and rely solely on onboard vision and distance sensing shows a mature grasp of real-world autonomy challenges. The combination of the CubePilot Orange, HereFlow Optical Flow Sensor, and TFmini-S LiDAR is a robust foundation for indoor navigation, while the upgrade to Raspberry Pi 5 provides enhanced processing capabilities for computer vision and sensor fusion.

I commend the team for selecting well-matched components and ensuring compatibility across systems. However, I suggest deeper testing under variable lighting and surface texture conditions, as optical flow sensors are known to be sensitive in such environments. In future iterations, real-time adaptive filtering techniques or additional inertial data fusion may improve system reliability. This team shows excellent technical potential, and with continued focus on stability and sensor calibration, they are well positioned for success in the next stages.

— Dr T Tirupal, HOD & Dean-Skill Development (ECE)

12. Review report by second mentor:

I am pleased to see how far the team has come since the preliminary proposal stage. The students have shown not just technical growth, but a strong sense of innovation and adaptability. Replacing proposed components with more efficient alternatives, like upgrading to **Camera Module 3** and **Emax 935kv motors**, highlights their ability to make strategic decisions under practical constraints. Their choice to implement a **GPS-denied ANAV system** showcases bold thinking, pushing the boundaries of conventional autonomous navigation.

I encourage the team to continue exploring novel sensing strategies, such as visual landmarks or neural network-based object detection, for further autonomy. More importantly, their teamwork, documentation discipline, and hands-on experimentation are setting a solid example for future project batches. This project reflects both academic rigor and creative engineering. I look forward to seeing their progress in the final rounds.

— *Seemakurthy Hupesh naga ketan, general manager, RMJ IT Solution*