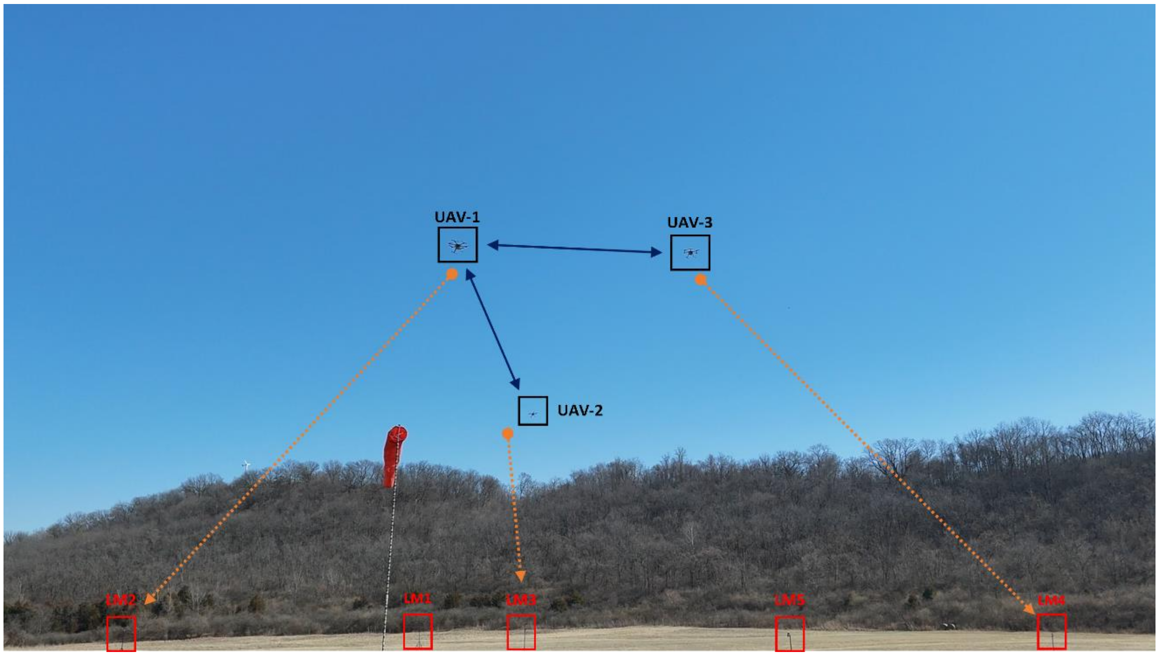


Co-Nav

Establishing a Multi-Vehicle Cooperative Localization Test Bed - Benchmarking for the Development and Deployment of Cooperative Navigation Algorithms

Table of Contents

Introduction 2
Multi-rotor platform 3
  Mounts and Base Plate Designs 4
Ranging Radio 6
  First Time Setup 6
Onboard Computer: 9
  First Time Setup 9
    Operating System – Ubuntu 20.04 LTS 9
    ROS Operating System – ROS noetic 9
    Mavros package for Pixhawk 10
Communication Network 12
  Hardware setup 12
  Fist Time Setup 13
    System Network Settings 13
    Ground Station Setup: (Rocket AC/M5) 14
    Onboard Computer (Bullet AC/M5): 16
Challenges 18
  Weight balance of UAVs with payload 18

Introduction

This document provides comprehensive instructions to develop the test bed used in associated paper. The test bed is designed for experimental validation of cooperative localization. The necessary aspects of cooperative localization testbed are:

- **Platform**: A multi-rotor UAV with stable flight characteristics and a payload capacity exceeding 10 lbs.
- **Intrinsic Measurements:** Incorporate a reliable Inertial Measurement Unit (IMU) to measure linear accelerations and angular velocities of the UAV.
- **Ground Truth:** Equip the UAV with a Global Navigation Satellite System (GNSS) module to provide accurate ground truth position information.
- **Extrinsic Measurements:** A ranging sensor that can measure distances up to 300 meters for outdoor operations.
- **Onboard computer:** Install an onboard computer capable of running Ubuntu 20.04 for processing and control tasks.
- **Communication network:** Establish a communication network capable of transmitting ROS messages between UAVs and ground stations at a frequency of 15 Hz or higher over a range of 300 meters.

The following sections will provide detailed steps to select and integrate these modules into the test bed.

Multi-rotor platform

To accomplish the first three tasks outlined in Introduction of the associated journal, selection of an appropriate Flight Control Unit (FCU) is critical. The chosen FCU must support the integration of various sensors and efficiently execute control tasks on the UAV. For this platform, a Pixhawk-based FCU was selected due to its extensive community support and hardware compatibility. Specifically, the Pixhawk Cube Orange Plus FCU is utilized, offering built-in Inertial Measurement Units (IMUs) with redundancy, control firmware for diverse UAV platforms, and extensive ports for connecting additional sensors and telemetry radios. Another significant advantage of Pixhawk FCUs is the availability of the MAVROS package, which converts internal MAVLink messages in the Pixhawk FCU to ROS messages, facilitating seamless integration into the platform.

For the base platform, the Aurelia X6 Standard hexacopter has been chosen due to its excellent performance in fulfilling the initial three tasks. The Aurelia X6 is a pre-built hexacopter that offers stable flight, a payload capacity of 11 lbs., and the ability to accommodate additional sensors and communication modules necessary for the test bed. It comes equipped with a Here2+ GNSS module, which provides the required positional sensor information, ensuring accurate and reliable ground truth data. Various other prebuilt platforms can be adopted for the testbed based on users' requirements.

Figure 1Aurelia X6 standard hexacopter.

Mounts and Base Plate Designs

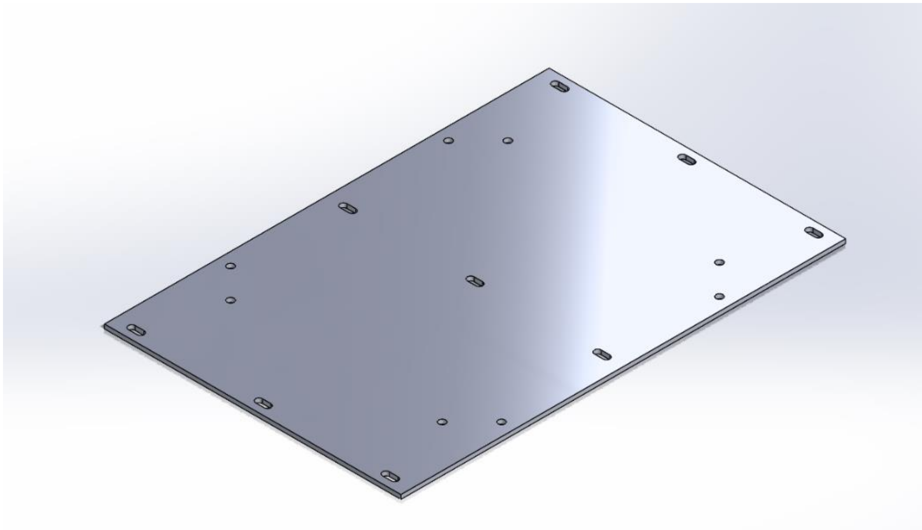


Figure 2 Stackable base plate design for Aurelia X6

This plate attaches to the bottom of the battery compartment using the three rails that come with the drone. It is also used in subsequent layers to attach additional hardware. The base plate is connected to the rails which are connected to the bottom of the battery compartment.sss

Required Hardware for level 1(directly attached to battery compartment):

- 6- M3x12mm grub screw to fix standoffs to the rails on the other side of base plate
- 6 - M3X70mm aluminum/ stainless steel standoffs to connect level 1 base plate to level 2 base plate (to accommodate onboard computer in level1)
- 2 - M3x8mm screws to connect middle rail to base plate

The outer hole pattern is fixed to attach base plate to the rails, and the inner hole patterns of the base plate can be modified to accommodate the hardware that will be attached to this.

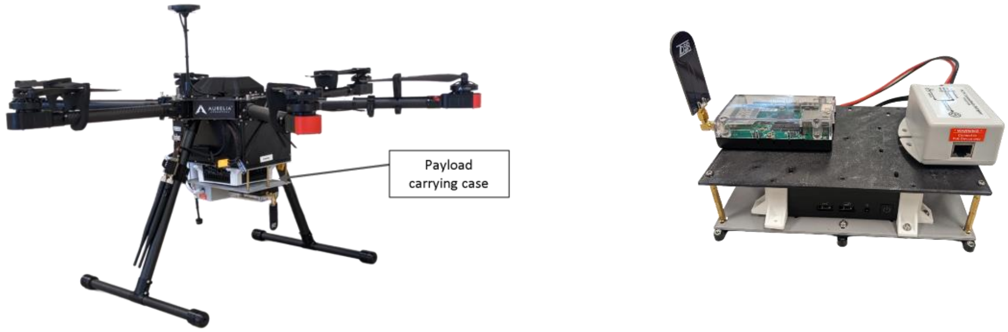


Figure 3 Example of a 2-level use of base plate separated by standoffs.

Ranging Radio

The TDSR P440 radios are employed for extrinsic sensor measurements in this test bed. These radios operate based on two-way time of flight (TW-TOF) technology, which allows them to measure the range between two units. The radios can be categorized into two types: Tags and Anchors. In this test bed, Tags are configured to measure distances to all nearby Anchors and other Tags.

Each radio is powered by a power bank that provides a 5V power supply with a maximum current of 3 amps via a micro-USB port. Tags must be connected to a PC to run the necessary node and collect range measurements.

For the experiments conducted with this test bed, a total of eight radios are used. Three of these are mounted on multi-rotor UAVs (Tags), which are connected to onboard computers. The remaining five radios are placed on the ground as landmarks to act as Anchors.

First Time Setup

The radios can be configured using RangeNet software provided by TDSR along with the radios. RangeNet GUI uses Serial, USB, or Ethernet communication to control and monitor P440 radios. Key features include:

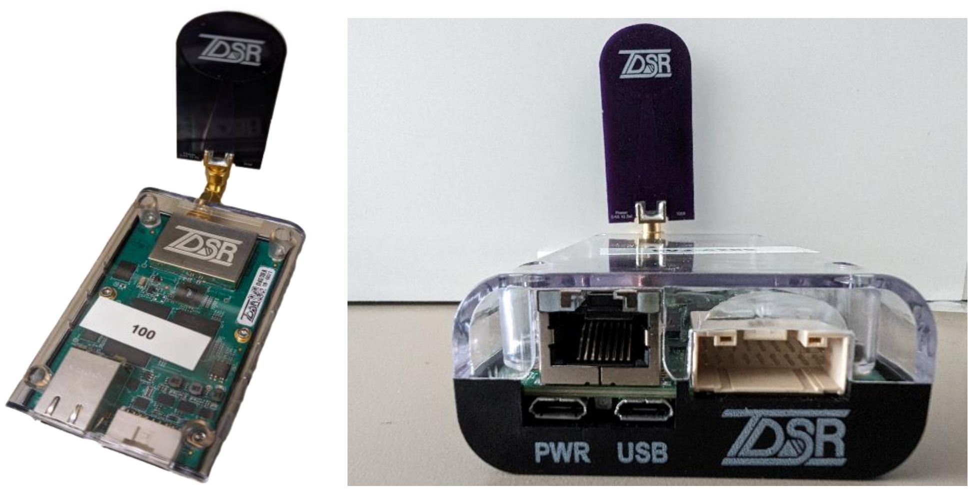


Figure 4 TDSR P440 radios

- Network status includes range measurements, range quality metrics, and system ranging rates.
- Control the radios connected to the network and mode of the network.

The radios need to be assigned a node ID by connecting to the RangeNet GUI. The process using a USB connection is described as follows:

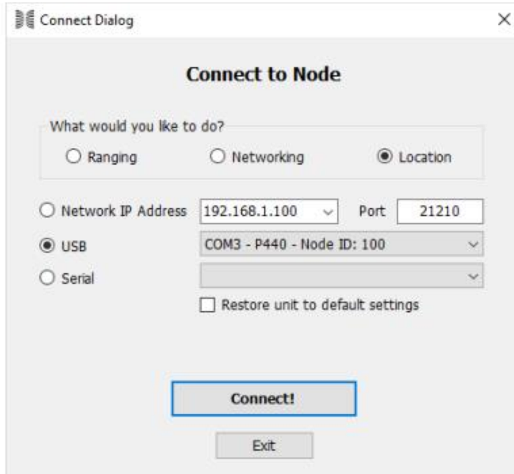


Figure 5 RangeNet Connection page

1. **Connecting via USB**: Connect the USB cable to the P4xx and the host computer. Click on the function you are interested in (for this setup, Ranging, Network, or Location). Click the USB button and select the COM port/unit serial number from the drop-down window. Refer to Figure 5 (RangeNet Connection page) for an example. If more than one P4xx is connected via USB to the host, select the unit of your choice from the drop-down menu. Note that it takes about 10 seconds for a P440 to boot, so if you power up a unit connected to the host, you will not see a COM port until the unit has booted and the driver has been identified.
2. **Assigning Node IDs**: After connecting, assign a unique node ID to each radio through the RangeNet GUI. See Figure 6 (RangeNet layout) for guidance on navigating the layout.

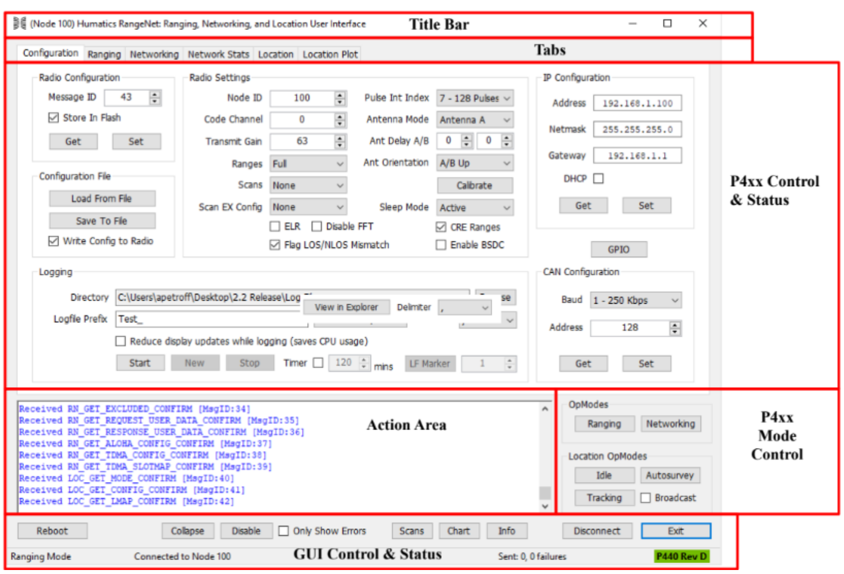


Figure 6 RangeNet layout.

3. **Testing Ranging Performance**: Once all node IDs have been assigned, test the ranging performance by using the Ranging tab. Change the responder ID to the desired radio ID and click on "Send" to send a request to the respective radio to test the signal. Refer to Figure 7 (RangeNet Range tab layout) for an example of the Ranging tab.

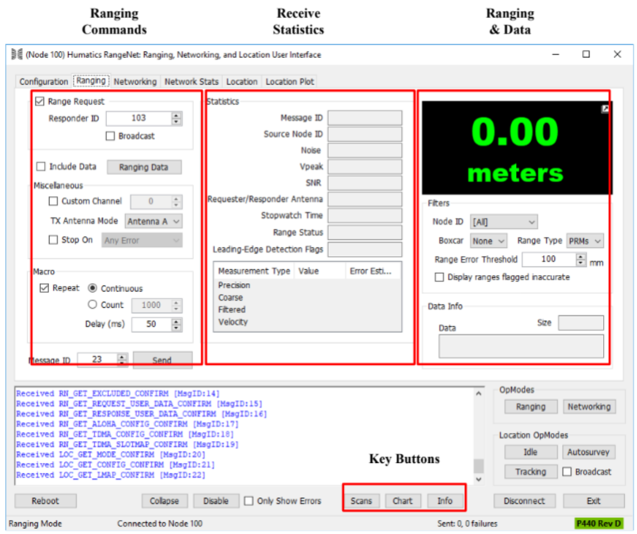


Figure 7 RangeNet Range tab layout.

With these steps completed, the radios are set up for use in the test bed. Any tag connected to a host PC will be able to request range information from all other radios in the vicinity.

Onboard Computer:

An Intel NUC 11 WSKi5 is used as the onboard computer in this test bed. This mini-PC is equipped with a 12-core Intel processor, 16 GB of DDR4 memory, and 256 GB of storage. These specifications make it suitable for running onboard localization algorithms, collecting data, and handling data transmission.

First Time Setup

With the onboard computer installed, the supporting OS and software packages need to be set up. The instructions for these installations are as follows.

Operating System – Ubuntu 20.04 LTS

This operating system is used for both the onboard computer and the ground station. Ubuntu 20.04 was chosen because it is the latest OS that supports ROS1, which is essential for this test bed. The instruction to install latest version of Ubuntu 20.04 can be found here.

ROS Operating System – ROS noetic

ROS Noetic is the ROS1 version compatible with Ubuntu 20.04. After installing the Ubuntu 20.04 OS, follow these steps to install ROS noetic:

1. Setup your source list

```
>> sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

2. Setu your keys

```
>> sudo apt install curl # if you haven't already installed curl

>> curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
```

3. Update Debian packages

```
>> sudo apt update
```

4. Install ROS version based on system specifications
    a. ROS-Base, for ROS packaging, build, and communication library, No GUI tools

    ```
    >> sudo apt install ros-noetic-ros-base
    ```

    b. ROS-Desktop, for everything in ROS-Base plus tools like rqt and rviz

    ```
    >> sudo apt install ros-noetic-desktop
    ```

c. ROS-Desktop_Full, for everything in Desktop plus 2D/3D simulators and 2D/3D perception packages.

```
>> sudo apt install ros-noetic-desktop-full
```

5. Environmental setup, to source setup.bash file in for ROS directory.

```
>> source /opt/ros/noetic/setup.bash
```

6. For convinence to automatically source this scrip every time a new shell is launched, ~/.bashrc must only source the setup.bash for the version you are currently using.

```
>> echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
>> source ~/.bashrc
```

7. Download required dependencies for using ROS.

```
>> sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator py
thon3-wstool build-essential
```

8. Initialize rosdep to easily install system dependencies for source you want to compile and is required to run some core ROS components.

```
>> sudo apt install python3-rosdep
>> sudo rosdep init
>> rosdep update
```

Mavros package for Pixhawk

MAVROS is a ROS package that facilitates communication with the Pixhawk FCU and the ground station. Instructions for installing and using the MAVROS package are as follows.

1. Install ROS noetic version of packages for mavros, mavros-extras

```
>> sudo apt-get install ros-noetic-mavros ros-noetic-mavros-extras
```

2. Download dependencies file for mavros package.

```
>> wget https://raw.githubusercontent.com/mavlink/mavros/master/mavros/scripts/in
stall_geographiclib_datasets.sh
```

3. Make this file executable using chmod command and execute install_geographiclib_datasets.sh to install GeographicLib library, which is used for geospatial computations.

```
>> chmod a+x install_geographiclib_datasets.sh
>> ./install_geographiclib_datasets.sh
```

By following the steps outlined in the subsections for Operating System – Ubuntu 20.04 LTS, ROS Operating System – ROS Noetic, and MAVROS package for Pixhawk, the environment for the test bed is successfully established. Once these steps are completed, the code for the localization and ranging radio packages can be executed to achieve full functionality of the test bed. The algorithm utilized in the localization package is detailed in the associated journal.

Communication Network

The communication network is established using Ubiquity rocket-AC/M5 and bullet-AC/M5. This setup establishes a long range 5-GHz Wi-Fi network, with rocket-AC working as an access point (AP) and bullet-AC working as station. The following subsection gives details on how to set up this network and use it in the testbed.

Hardware setup

As mentioned previously, the ground station is connected to the Rocket-AC/M5, which functions as a Wi-Fi access point. The Rocket-AC is connected to an airMAX 5 Ghz. 13dBi Omni antenna to enhance the range and bandwidth of the communication network. On the multirotor, the Bullet-AC/M5 is equipped with a Lumenier AXII 5.8 GHz antenna.

At the ground station, the Wi-Fi access point is elevated to increase the communication range and reduce interference caused by ground effects, as illustrated in Figure 8 Ground Station. On the UAV, the Bullet-AC/M5 is mounted on the back with its antenna facing downward to ensure a direct line of sight during flight.

The Bullet-AC/M5 receives both power and data via a Power over Ethernet (PoE) connection, which is provided by a PoE injector. The injector connects to a 22.2V auxiliary power connector on the multirotor and to the Ethernet port on the onboard computer, as depicted in Figure 9 Wi-Fi Station on the UAV.

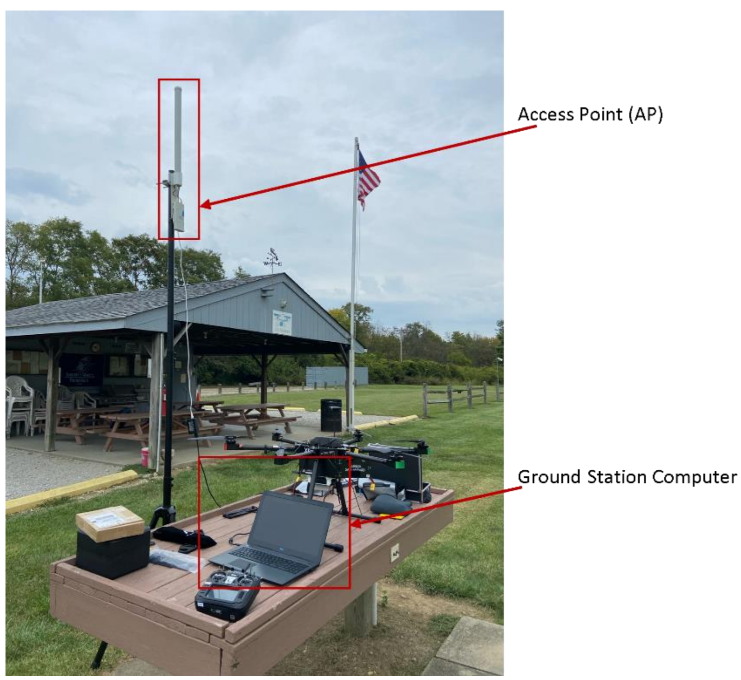


Figure 8 Ground Station

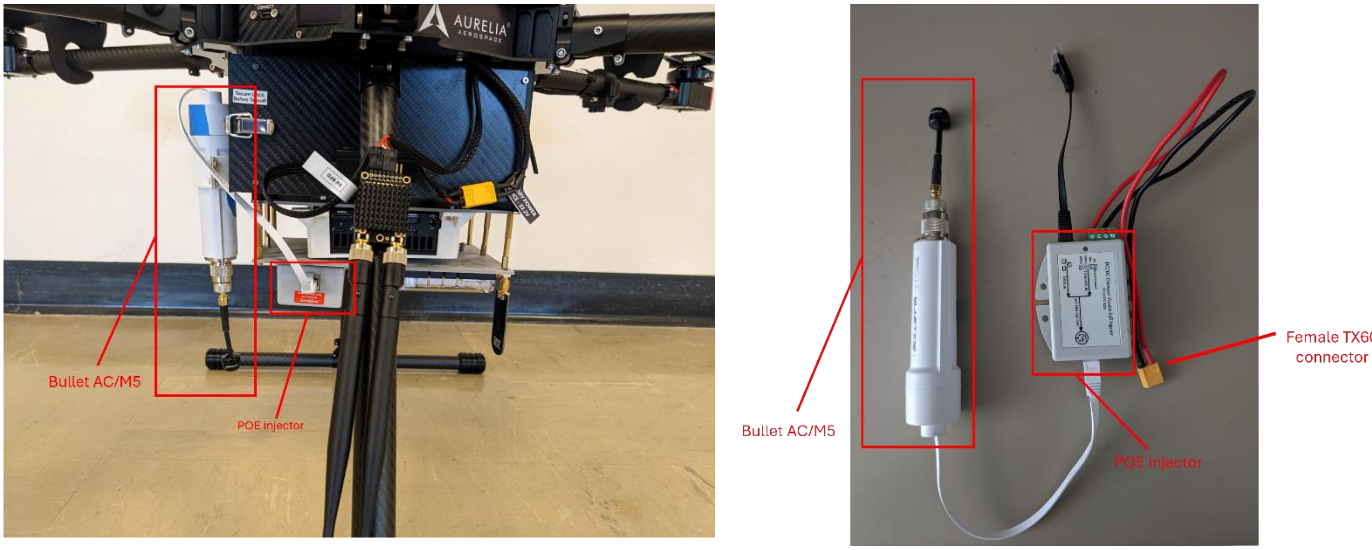
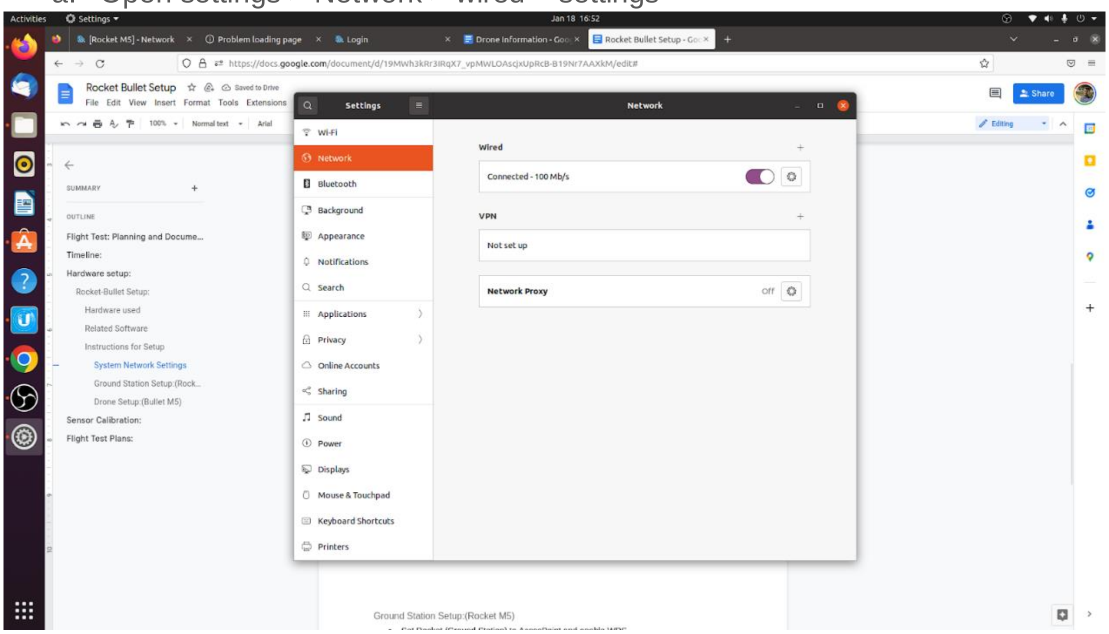


Figure 9 Wi-Fi Station on the UAV

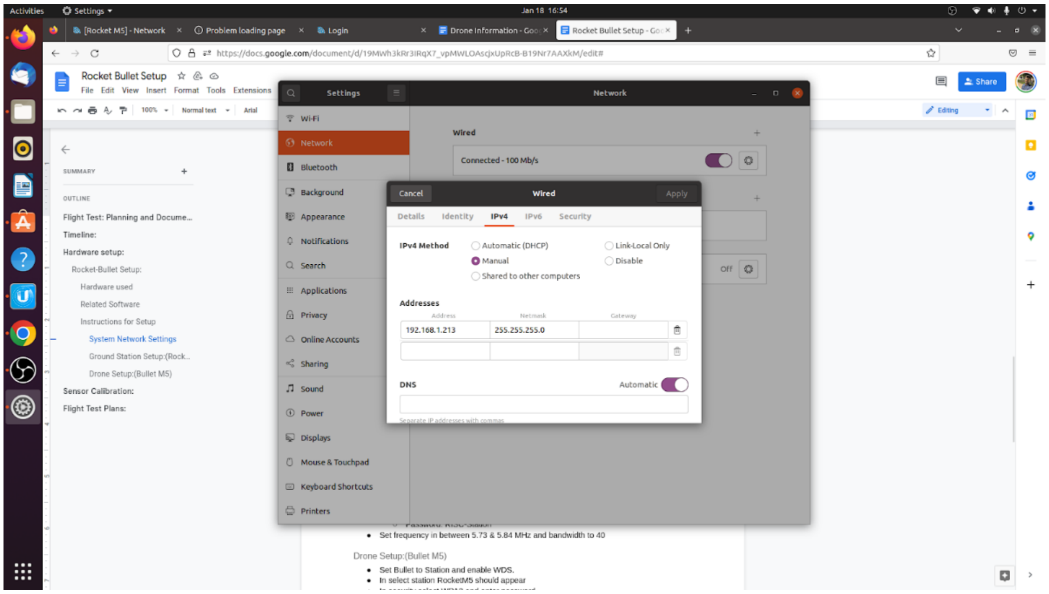
Fist Time Setup

System Network Settings

1. In an onboard computer with Ubuntu OS, set device IPv4 IP address to static and change these settings
    a. Open settings > Network > wired – settings

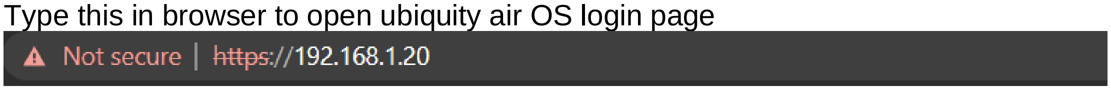


2. Set IPv4 IP address as **192.168.1.xxx** (should be unique for each device in network) and Netmask as **255.255.255.0**.



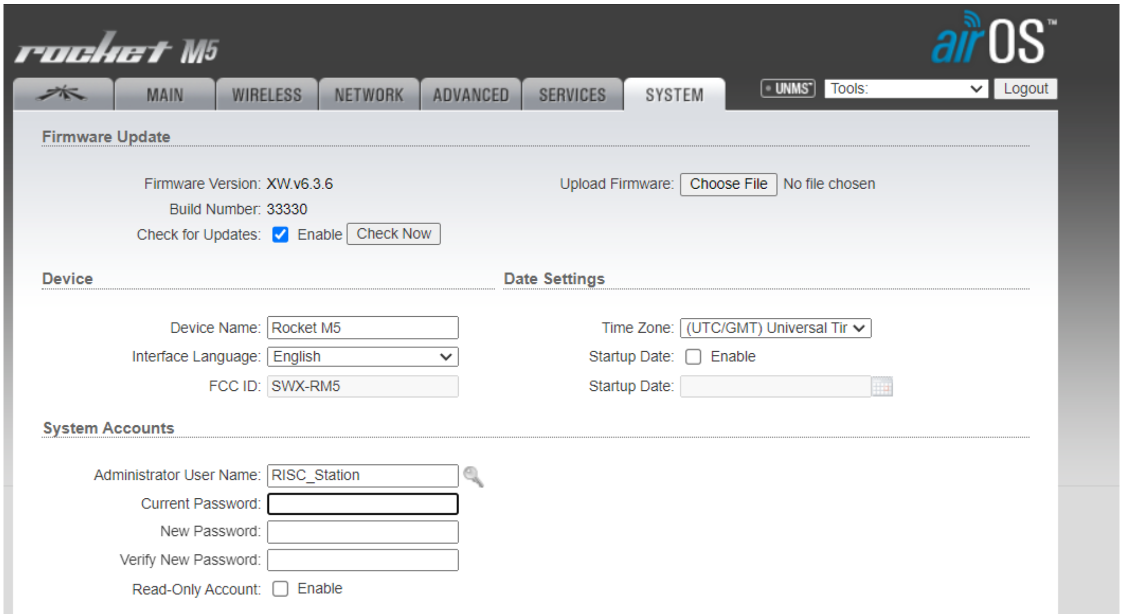
Ground Station Setup: (Rocket AC/M5)

1) Do a factory reset of the device. Reset switch can be found under bottom panel of the device
2) Now rocket will have its default IP as **192.168.1.20**
3) Type this in browser to open ubiquity air OS login page

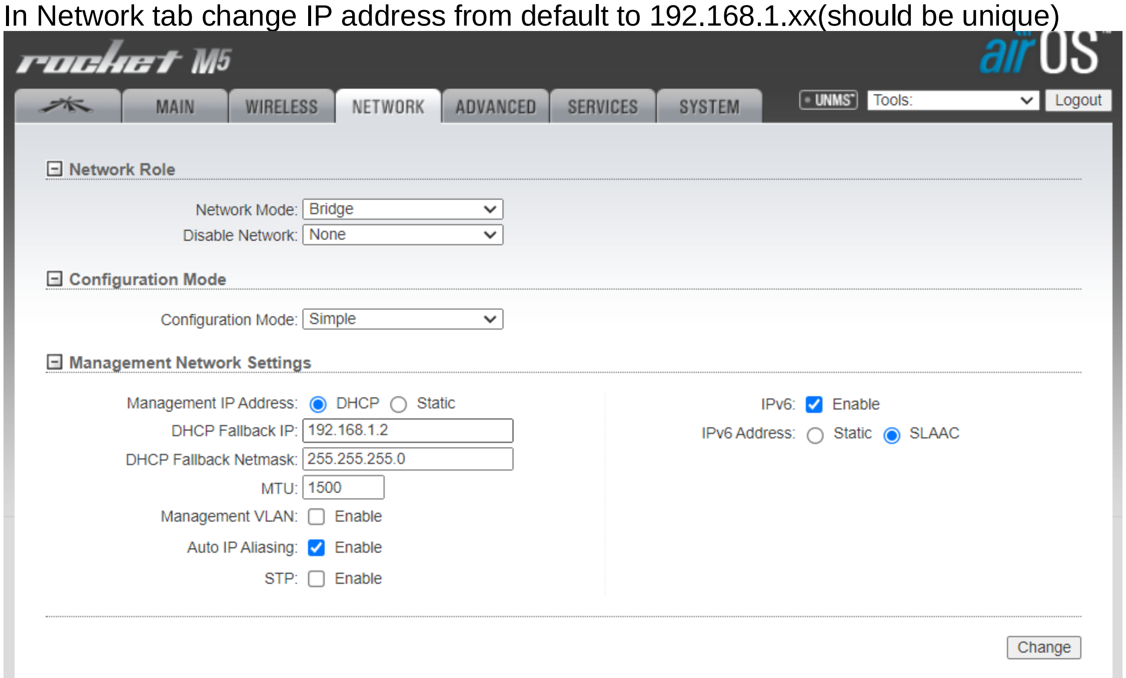


4) Default username and password is **ubnt**

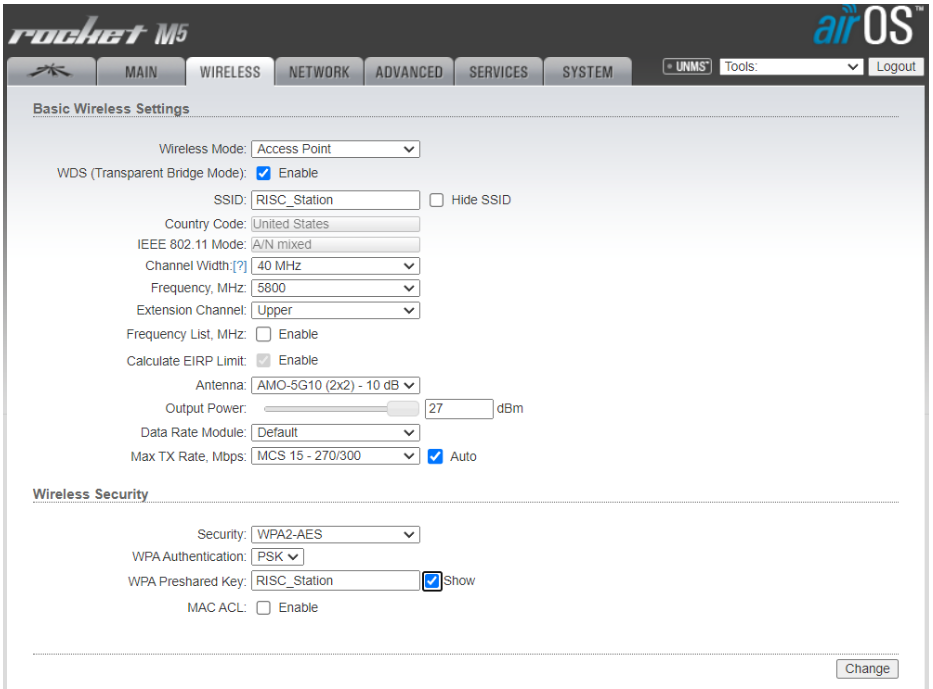
5) In System tab change Administrator username and password (first change password and then change username) then press change>Apply



6) In Network tab change IP address from default to 192.168.1.xx(should be unique)



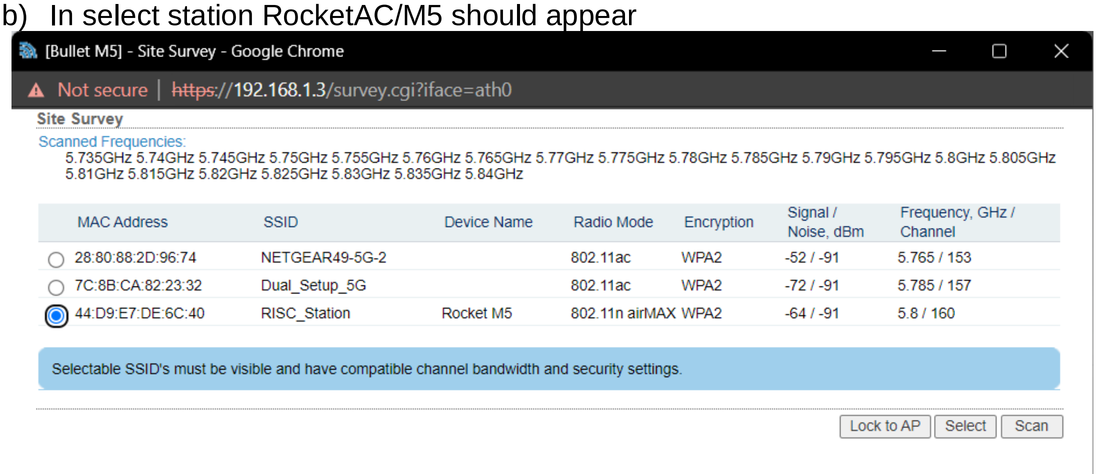
7) In Wireless tab
   a) Set Rocket (Ground Station) to Access Point and enable WDS.
   b) Set security to WPA2 and setup a new password
   c) Set frequency in between 5800MHz and bandwidth to 40MHz and apply these changes



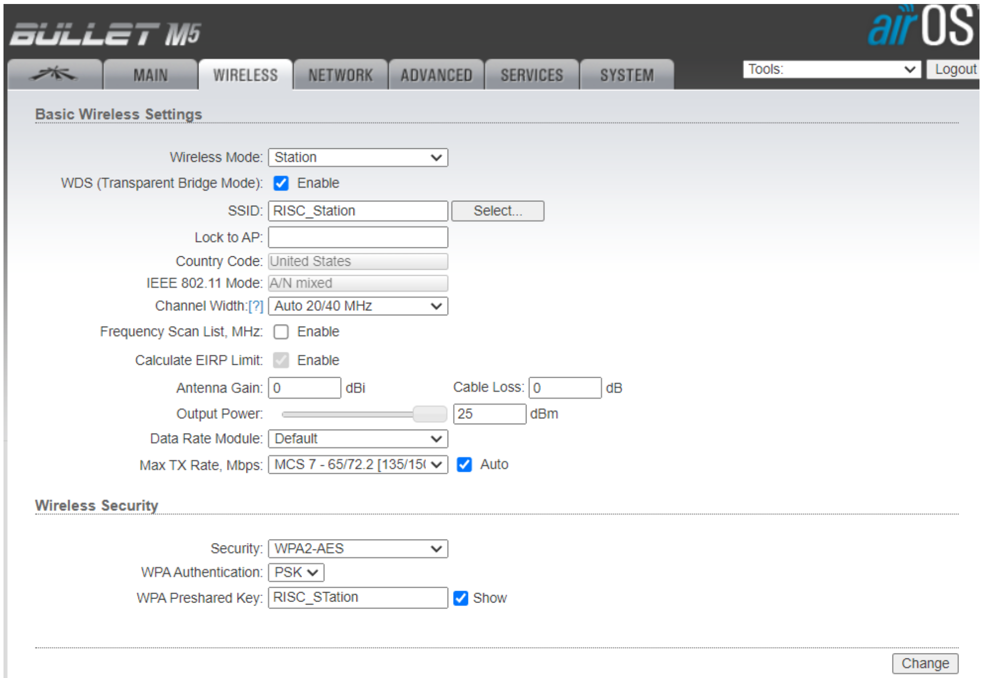
Onboard Computer (Bullet AC/M5):

1) Follow step 1 to step 6 in Ground Station Setup: (Rocket AC/M5)
2) In wireless tab

   a) Set Bullet to Station and enable WDS.
   b) In select station RocketAC/M5 should appear



   c) In security select WPA2 and enter password.



3) Repeat the above steps for all drone stations and connect to ground station.

Note: To use Bullet AC/M5, AC devices need to be set as Access point and M devices as stations.

Challenges

Weight balance of UAVs with payload

In the initial design phases of the platform, several crashes of the multi-rotor were observed. These incidents were primarily attributed to a high center of gravity caused by the placement of the payload on top of the drone, as illustrated in Figure 10 Initial design with higher center of . This configuration resulted in instability and required excessive control input to maintain stable flight.

To address this issue, subsequent designs relocated the payload to the bottom of the drone, as shown in Figure 11 Final design for additional payload with lower center of . The updated designs also ensured that heavy payloads were centered, thereby lowering the center of gravity and reducing the control input needed to stabilize the drone during flight.

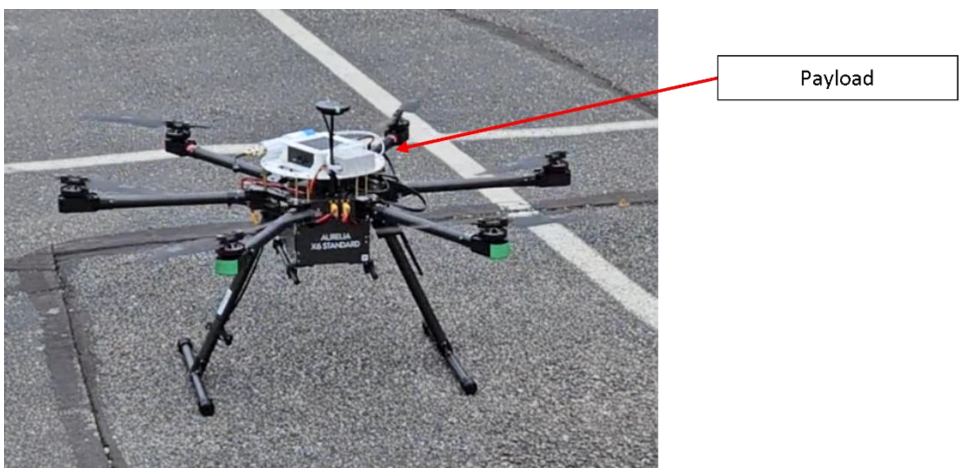


Figure 10 Initial design with higher center of gravity.

Figure 11 Final design for additional payload with lower center of gravity.