System Design Document For ILS/VOR Data Logger (PIGEONS)

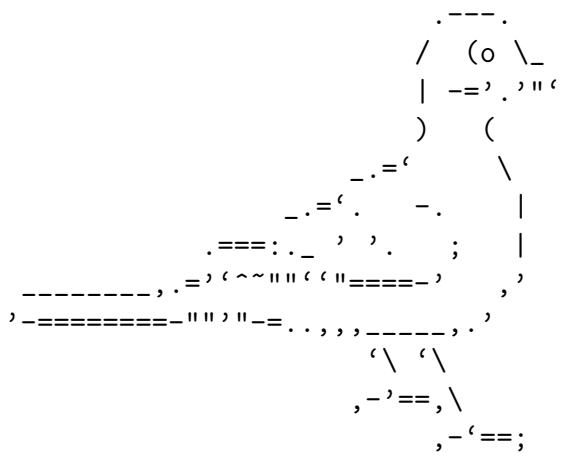
Michael Cain

Brian Jaury

Andrew Newlon

Daniel Sommer

James Trafny



Version	Date
118.85	12/05/18
122.95	03/05/19
124.05	04/16/19

Contents

1 Introduction

1.1 Purpose and Scope

This document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces for the PIGEONS project.

1.2 Project Executive Summary

This section provides an overview of the PIGEONS project from a management perspective, showing the framework with which the system design was conceived.

1.2.1 System Overview

The goal of the PIGEONS system is to collect VOR/ILS signal data via an autonomous drone. Under this, there will be two systems working together: the Data Collection System, and the Drone Flight System. The Data Collection System will take raw signal data, process it, and sent the processed data to a ground station. The ground station will be able to view the results of collection and processing in real-time where results will be overlaid over a geographic map, based on Mission Planner. The position of this will be determined though a GPS unit shared with the Drone Flight System. The processed signal data will be transferred via a ZigBee communication module.

The following use case diagram (Figure 1) gives an overview of the whole system:

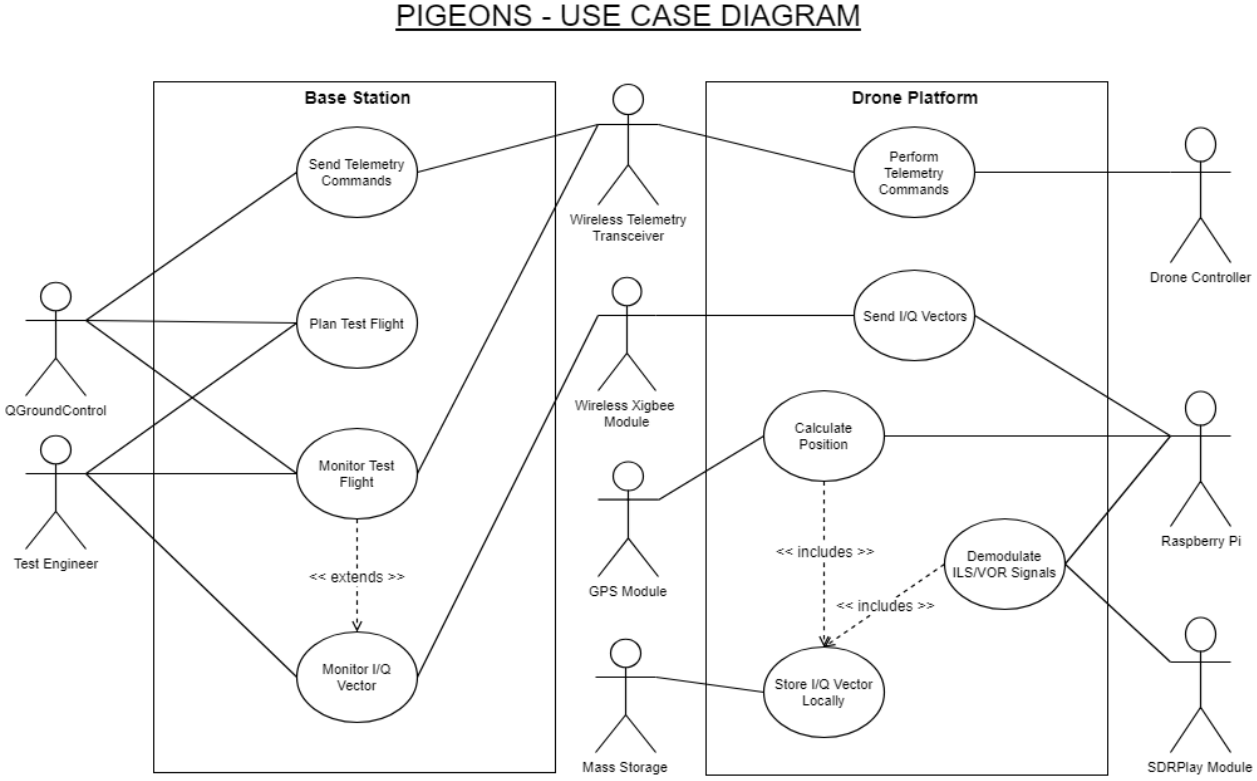


Figure 1: Use Case Diagram for PIGEONS system.

The following figure (figure 2) is a basic overview of the PIGEONS system including all incoming/outgoing signals and a top level view of the subsystems. The Drone Flight System will be responsible for the autonomous drone flight. This will consist of a Pixhawk, GPS unit, telemetry unit, and the motors. The Pixhawk will use the telemetry modules to communicate with the ground station, giving the coordinates and determining the flight path. The GPS module will be shared with the Data Collection System.

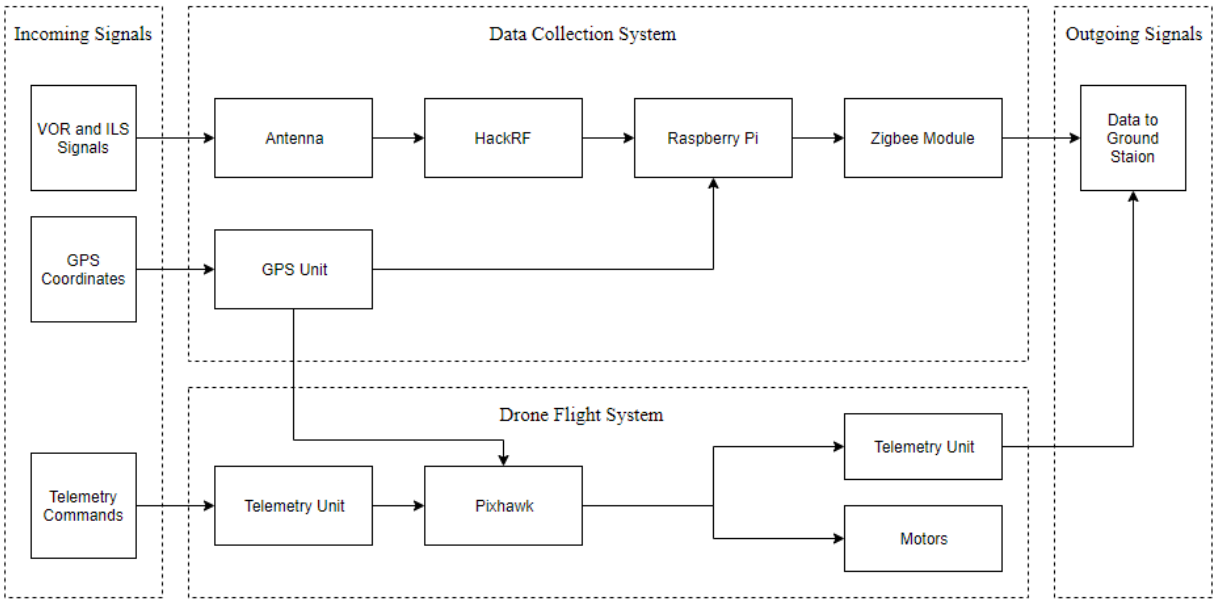


Figure 2: Basic parts used in the PIGEONS system, and where they communicate.

1.2.2 Design Constraints

This project has three major limitations. These are weight, time, and cost. The system is design to be mounted on a drone to fly around an airport or VOR station and pick up terrestrial landing aid signals. The first concern, weight, is due to the drone. A drone only has a limited carrying capacity. The hexcopter we are using will have a carrying capacity of about 1-2 kg. The second concern, time, is also due to the drone. The drone will be running on a Lithium-Ion battery. If weight permits, a secondary or tertiary battery can be affixed to the drone, increasing the flight time. The last is the cost. This project is intended to be a much more cost efficient solution to verify navigation aid transmitters. The project will be carried out with as little costs as possible. The antenna, software defined radio, and communication modules alone take up a large portion of the budget. This will determine the remaining purchases made, potentially forcing a lesser quality part.

Other than physical limitations to the drone, there are regulatory constraints it must conform to. These are mainly from the Federal Aviation Authority (FAA). There are several sections of the Federal Aviation Regulations (FAR) that are pertinent to this project, namely FAR Part 47, Part 48, Part 101, and Part 107. These are all regulations for the person flying the drone and any necessary registration associated with this. Part 107 will most likely be the way this is taken care of. Part 107 is the easiest regulatory section to get through and deals with the certification of a drone pilot. This person is then responsible for knowing where and how to fly the drone and to stay within compliance of FAR Part 107. FAR Part 101 deals

with hobby drones, and does not require registration. FAR Part 47 and Part 48 deal with non-hobby drones not regulated under Parts 47 or 48. Along with this, the drone cannot fly within certain areas. These include near stadiums and sporting events, near airports, near security sensitive areas, restricted airspace or Washington D.C. It is assumed under FAR Part 107 that the drone pilot is aware of these.

1.2.3 Future Contingencies

The current design of the project will include a 3D printed case and a Raspberry Pi as the main processor. The case will be used to keep the unit in a self-contained, modular package. The antenna shall be mounted to the drone in a way that maintain the center of mass on the drone. Currently, the Raspberry Pi is being used to demodulate the ILS and VOR incoming data. While a more powerful microprocessor is preferred, the Raspberry Pi contains enough processing power to perform demodulation. The Intel NUC is a suggested microprocessor for this system.

1.3 Document Organization

This document is designed to give the reader an idea of the system design. The following sections will provide information on what the product does, limitations, interactions, interfaces, hardware and software designs, and security.

1.4 Project References

1.4.1 Manual of Testing of Radio Navigation Aids, ICAO Doc 8071, Vol 1. Fourth Edition, 2000.
1.4.2 Mission Planner Quick Start Guide, from http://ardupilot.org/copter/docs/quick-start-guide.html
1.4.3 Digi ZigBee RF Modules User Guide, from https://www.digi.com/resources
1.4.4 Pixhawk 4 User Manual, from docs.px4.io

1.5 Glossary

DDM - Difference in depth of modulation. The percentage modulation depth of the larger signal minus the percentage modulation depth of the smaller signal, divided by 100.

ICAO - International Civil Aviation Authority

ILS - Instrument landing system

I/Q Vector - A custom data type consisting of a VOR azimuth, ILS glideslope correction, and GPS coordinates

MP - Mission Planner, the software used for drone autopilot feature.

PMV - PIGEONS Mission Viewer, ground station software controlling data acquisition functions of PIGEONS payload, and receiving/displaying the data for analysis.

RPAS - Remotely Piloted Autonomous System

RPi - Raspberry Pi 3 model B

SDR - Software Defined Radio

UDP - User Datagram Protocol

VHF - Very High Frequency

VOR - VHF omnidirectional radio range

2 System Architecture

This section describes an overview of the hardware and software architecture for the PIGEONS system and subsystems.

2.1 System Hardware Architecture

The PIGEONS system consists of several connected devices that must work properly for reliable data to be obtained. This resulted in the decision to use the following hardware for the PIGEONS system. While the system is configured using the following devices, each of the devices mentioned can be interchanged for an equivalent or more capable device.

MP Computer - Ground station computer running Mission Planner PMV Computer - Ground station computer running PIGEONS Mission Viewer Raspberry Pi - A microcontroller used to demodulate the incoming signals.
XBee HP900 Pro - A ZigBee based module that will transmit the data to a ground station.
RFD 900+ - Long range telemetry module for RPAS communications with QGC
RAMI AV-525 - A Piper VOR/ILS antenna used to pick up the signals.
Polaris GNSS RTK GPS - A device used to find the position of the drone.
HackRF - An SDR that receives the VOR/ILS signals.
USB Drive - Internal storage for the received VOR/ILS signal data.
SD Memory Card - Boot drive for the RPi.
Power Source - 14.7V Lithium-Ion Battery to power the drone and PIGEONS system.

Ground Station Computer - Computer running Windows 7 or later.

Figure 3 demonstrates the architecture of the PIGEONS system. The VOR and ILS signals are received by the Raspberry Pi using the HackRF connected to the RAMI AV-525 antenna. The VOR/ILS signals as well as the GPS data from the RTK GPS are then processed by the Raspberry Pi where a copy is saved to the SD card before being sent to the PIGEONS Mission Viewer ground station using the XBee module. PIGEONS Mission Viewer will display the ILS and VOR values received from the XBee connected to the Raspberry Pi. RPAS operations will be controlled using the QGroundControl ground station. Connections between the RPAS and ground station will be facilitated with the RFD 900+ telemetry module.

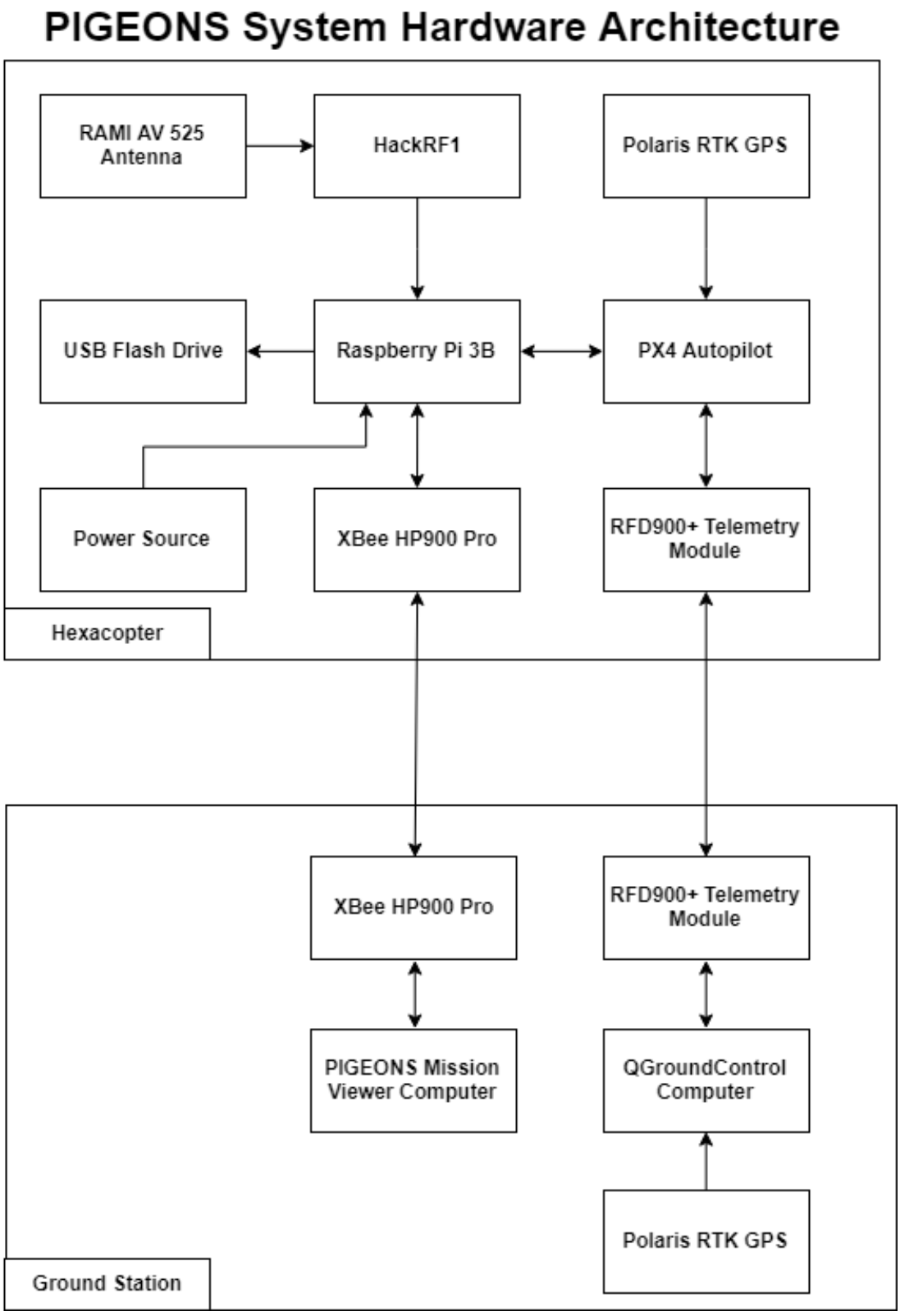


Figure 3: PIGEONS Hardware Overview

2.2 System Software Architecture

The PIGEONS system consists of several distributed pieces of software running on physically separated hardware. The base station, developed in C++ using QT Creator, allows the test engineer to configure and view test data. The base station contains a User Interface subsystem for the test engineer to enter test parameters, such as viewing a live or previously recorded mission profile. The User Interface subsystem interfaces with an IO subsystem. The IO subsystem receives data from the communications subsystem and stores it locally in an SQLite database. The NAVAid data stored locally is then displayed on a map view using ARCGIS. The communication subsystem bridges the base station subsystem with the demodulation subsystem. The demodulation subsystem uses GNU Radio Companion (GRC) to receive NAVAid data from the software defined radio and pipes the data over UDP for a python script in the communication subsystem to package and send to the XBee transceivers. Mission Planner is used for the flight management subsystem to control autonomous flight for the UAV. Figure 4 shows how the data will be passed from the drone to Mission Planner to be displayed to the user.

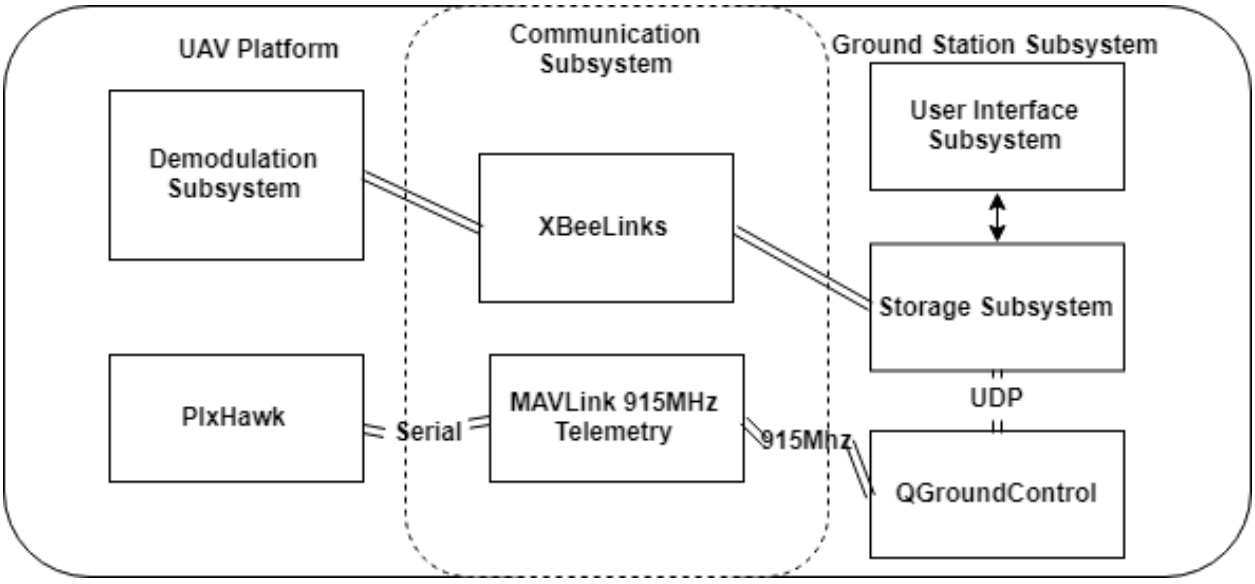


Figure 4: Software Architecture Overview

2.3 Internal Communications Architecture

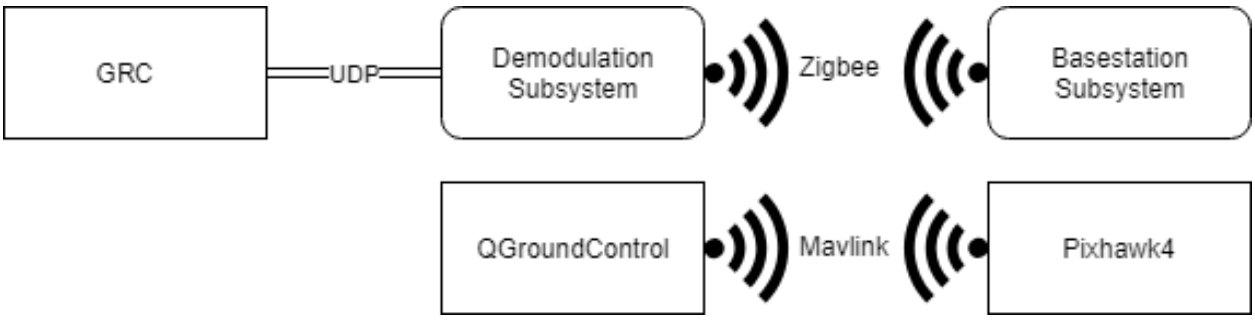


Figure 5: Internal Communications Architecture Overview

In the PIGEONS system, there are three main channels of communication; data over UDP, data over ZigBee, and data over Mavlink. Communications over UDP are handled on a port chosen by the user, the default UDP connection is on localhost (127.0.0.1) over port 5005. The ZigBee/IEEE 802.15.4 protocol is a specification created for wireless networking. Our ZigBee module is the XBee-900HP, which has data rates up to 200 kbps. Its transmitting power is 24 dbm and weights approximately 0.7 oz. The ZigBee network carries String data types across the network to be parsed by our base station and drone platform. This unit was picked due to its large data rate, and the ability to transmit, albeit at lower data rates, up to 25 miles. The Mavlink protocol is used to carry telemetry data and command back and forth between the drone and control its physical location. The messages sent over Mavlink are controlled only by Mission Planner and was not modified by the PIGEONS team.

3 Human-Machine Interface

This section provides details of the inputs and outputs as they relate to the test engineer operating the device.

3.1 Inputs

The operator will have to provide several key pieces of information to the PIGEONS system for it to carry out a mission. Flight paths will be designed in Mission Planner using the MP interface. MP is commercial, off the shelf software and is not modified for this project. The next set of inputs are the tuning parameters for the software defined radio. Figure 6 demonstrates the user interface of PIGEONS Mission Viewer used to input mission parameters and configurations. In PIGEONS Mission Viewer, the user will be able to select a Live Mission, or replay a previous Mission, XBee configurations, and Mission configurations.

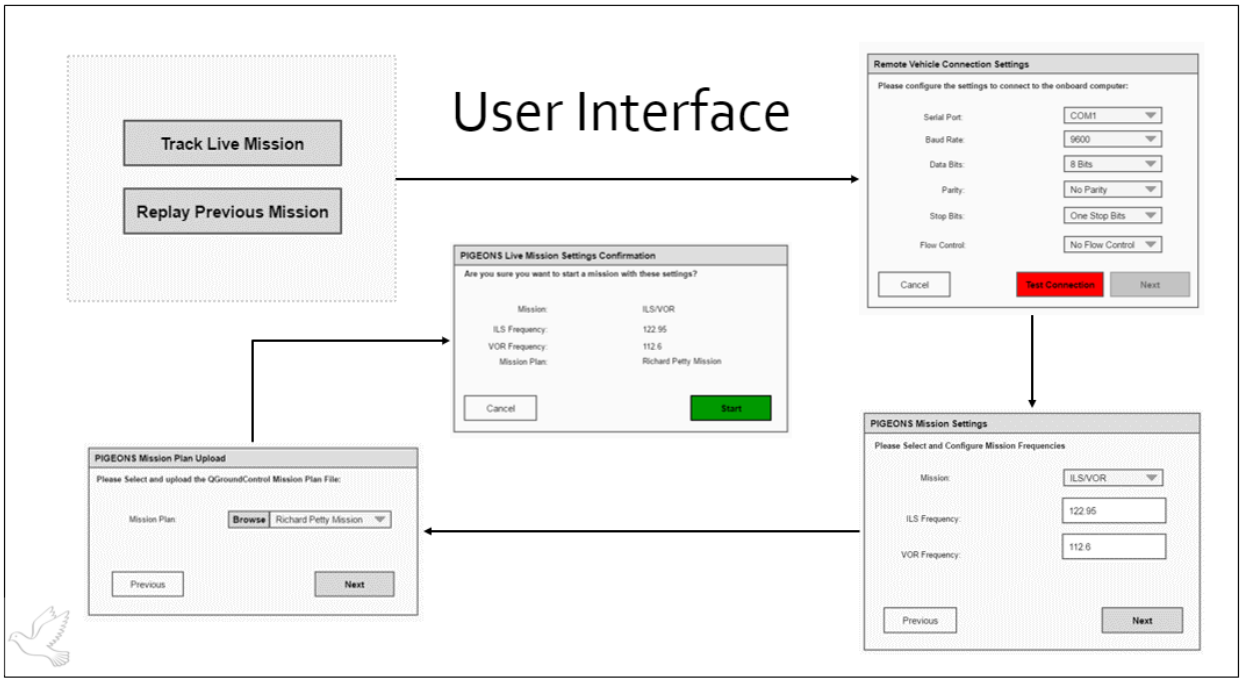


Figure 6: User Inputs

3.2 Outputs

During a mission, the user will be presented with PIGEONS Mission Viewer. PIGEONS Mission Viewer provides a constant output of useful information to the user in a graphical and tabulated way. The graphical output (shown in Figure 6) shows where the data points were collected on a map of the area. Clicking on individual points will detail the location, altitude, measurement type, signal strength, and whether the test passed or not at that location. At the end of the mission, a summary table will be presented to the user listing the details of the mission.

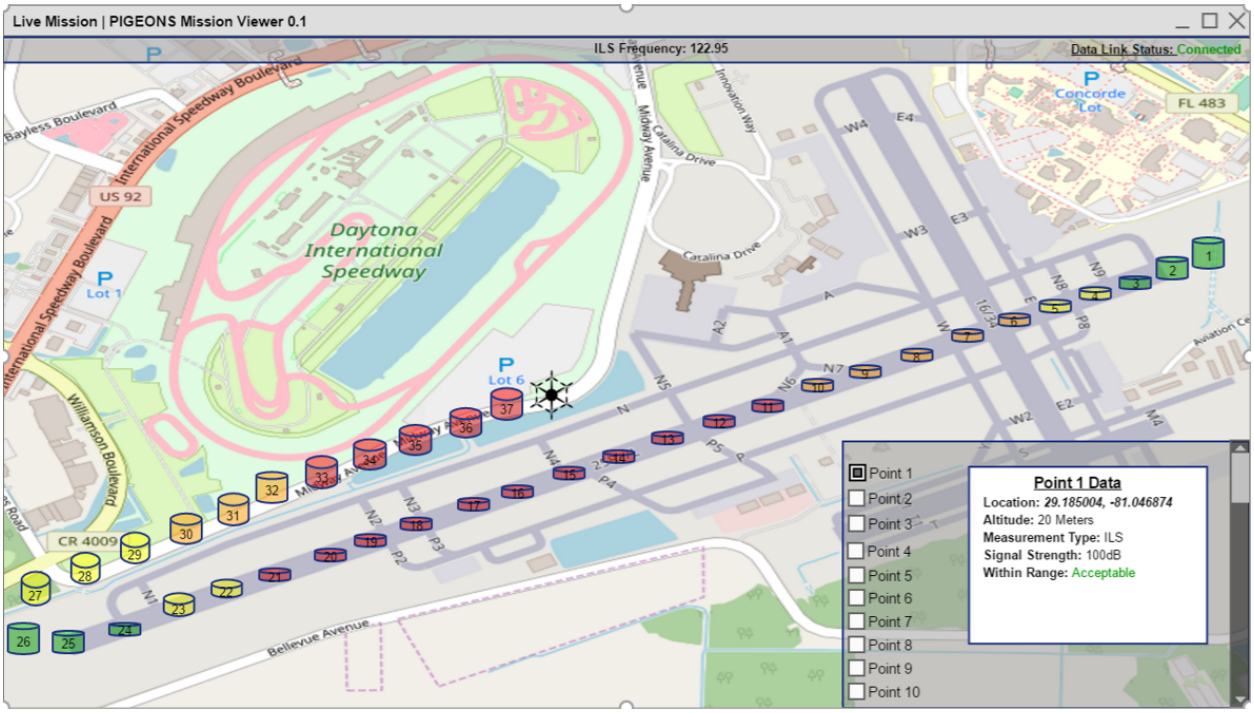


Figure 7: User Outputs

4 Detailed Design

This section contains detailed information about the hardware and software design of the system.

4.1 Hardware Detailed Design

The system is designed to where there are two hardware subsystems being used on the drone. The first is used to collect, process, and send data to the user while the other is used to fly the drone. The two will work in tandem to obtain the data the user desires.

The hardware components used to collect the data are the RAMI AV-525 antenna, HackRF SDR, Raspberry Pi, Alpha Polaris RTK GPS module, and an Xbee HP900 Pro module. The antenna will be mounted to the outside of the box and connected to the SDR. The Raspberry Pi is connected to the SDR, the Xbee module, and the Pixhawk. The Pixhawk will take in data from the GPS and forward it to the Raspberry Pi. The components used to fly the drone are the Pixhawk, RFD 900+ telemetry units, the Alpha Polaris RTK GPS module, and the motors. The Pixhawk is connected to the motors, through electronic speed controllers. The telemetry unit is connected using a TTL connection to the Pixhawk and is

used to communicate commands between the base station and the drone. The GPS unit is the same as above.

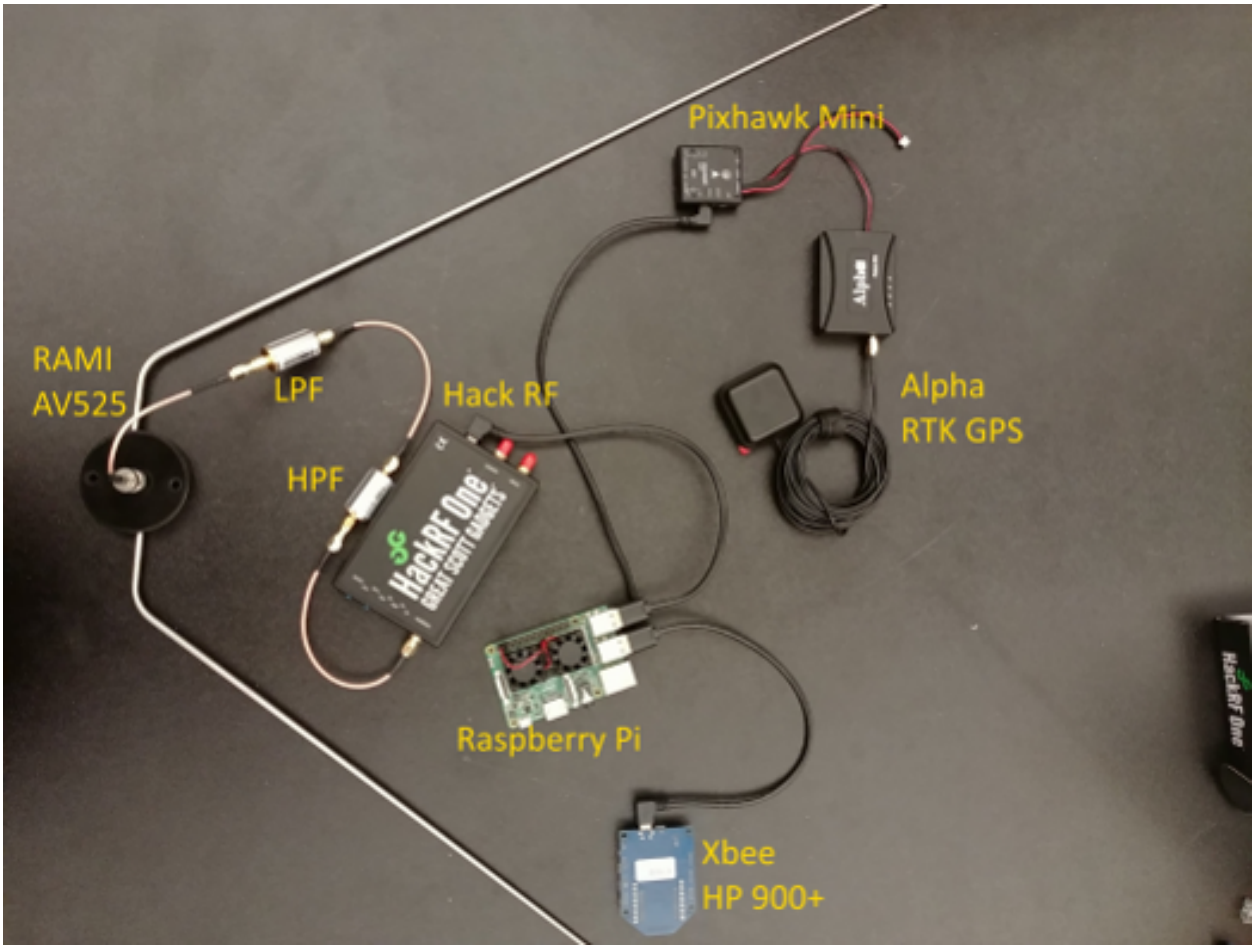


Figure 8: Expanded components

The antenna picked is a RAMI AV-525 antenna. This antenna is utilized on Piper and Cessna aircrafts and will be utilized on the drone to provide the most real-world scenario for testing. This has a female BNC connector on it and gets connected to the HackRF SDR, which has a female SMA connector. To connect these together a 50Ω BNC-male to SMA-male line is used. The SDR and Raspberry Pi are connected using USB. The Xbee modules are attached to a development board, with a micro-USB port. Again, this is connected with a USB cable. Lastly is the Alpha RTK GPS module. This is connected to a receiver through the device's SMA port, while a TTL connection will be used to connect it to a Pixhawk.

The GPS data used by the system comes from the onboad Pixhawk module. The GPS data from the Pixhawk is forwarded to the Raspberry Pi through the Mavlink protocol. While the

payload is connected to the drone, a 6 pin connector will be used to connect the RTK unit in the payload to the Pixhawk used to control the drone. This allows both the measuring equipment and flying equipment to have the RTK level of accuracy. While not on the drone, the user will have to utilize a separate GPS module connected via Serial USB.

The Raspberry Pi 3 is the current processor used in the system, while the operating system and all required software for processing is on a micro SD card. This is subject to change if the budget allows and if it cannot process all of the data coming in. The Raspberry Pi will take in data from the HackRF SDR, and do two things with it. First, it will demodulate the VOR and ILS signals. This data will be sent to the HP900 Pro Xbee module to be sent to the ground station. The second thing it will do is store the raw data on a USB drive in-case the user wants the raw data. Sending both the raw data and demodulated data may cause the Xbee module to bottleneck, so storing the data locally will help solve this issue.

The Xbee HP900 Pro was picked due to it being lightweight low power, and have the versatility for high data rates and long range. The module weighs approximately 0.7 oz, which is important in a system designed to be as light as possible. Second is the power consumption. It has a transmitting power of 24 dbm, with a transmitting current of 215 mA. Low power consumption here is crucial for a system which relies solely on battery power. The last benefit here is the ability to transmit high data rates. At short distances of a few hundred feet, the module has the ability to transmit up to 200 kbps. If need be, the distance can potentially be pushed up to a range of 25 miles, if weather permits and if there is line of sight. At this range though, the data rate drops to 10 kbps.

Because the whole unit is either attached to a drone or controls the drone, everything needs to be powered by battery. To accomplish this, the drone will be powered using a 5000 or greater mAh battery. Connecting two of these together will result in an extended flight time for the drone. It will be possible to get between 10 and 30 minutes of flight time out of this, depending upon if one or two batteries are used, and the weight of the payload. All of the electronics used to collect, process, and send the data will be enclosed in an easily removable container. The final weight of this container and contents will have a large impact on the battery life.

The last part of the hardware design will be the base station. This will be a laptop the user has, with the required software downloaded. Connected to this will be the other RTK module, as well as its antenna, and the other RFD 900+ telemetry module.

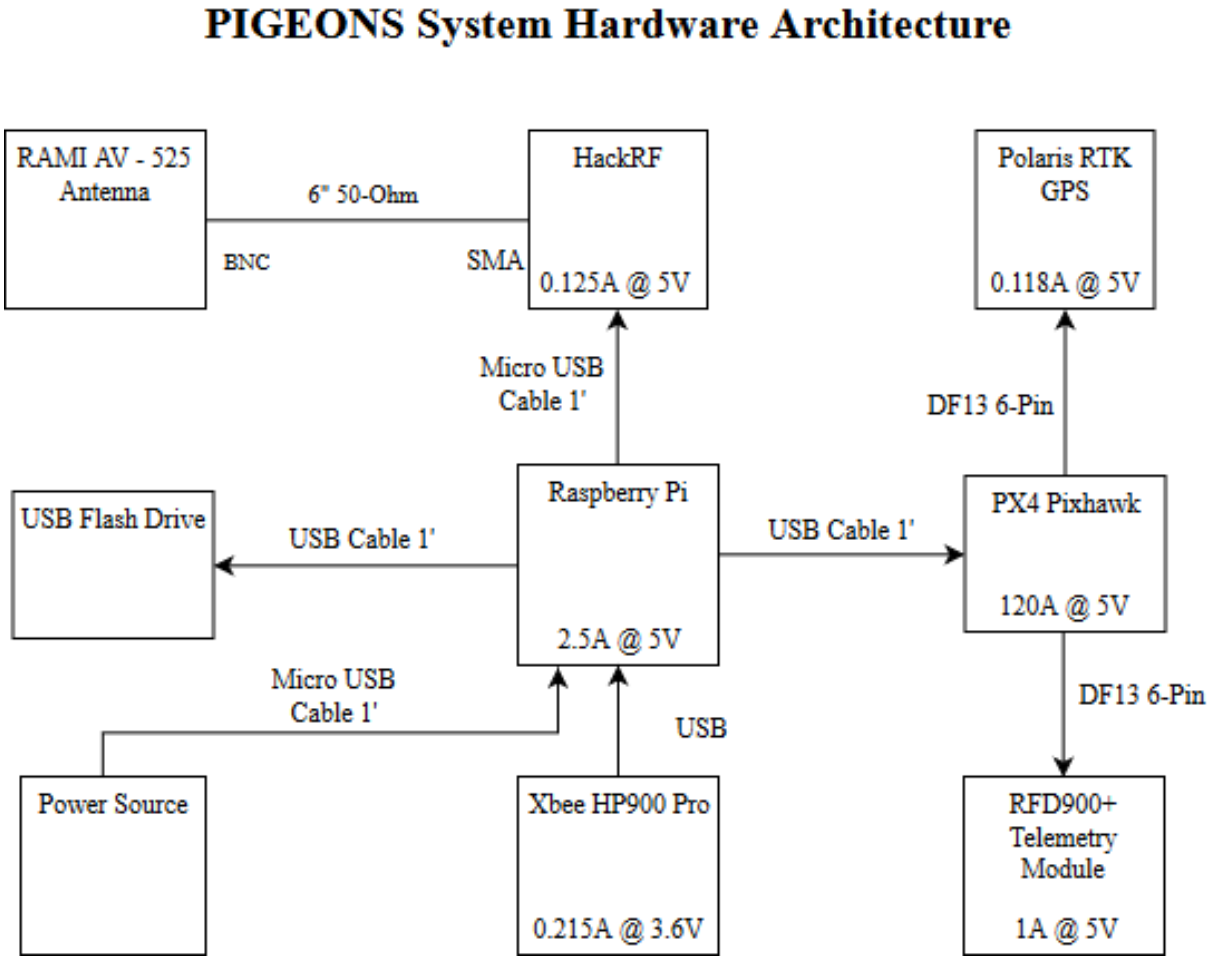


Figure 9: Hardware Architecture

4.2 Software Detailed Design

The PIGEONS software systems can largely be broken up into four subsystems; the base station subsystem, the demodulation subsystem, the communications subsystem and the telemetry subsystem. The communications subsystem is responsible for handling communications between the demodulation subsystem and base station subsystem. The base station subsystem has the responsibility of displaying information received from the communications system as well as relaying mission parameters to the communications subsystem for use in the demodulation subsystem. The demodulation subsystem handles capturing raw data from the SDR, decoding the signals into usable data, and packaging the data into useful vectors to be sent over the communications network to the base station. The final subsystem is the telemetry subsystem, which is responsible for handling the command and control of the physical hex-copter.

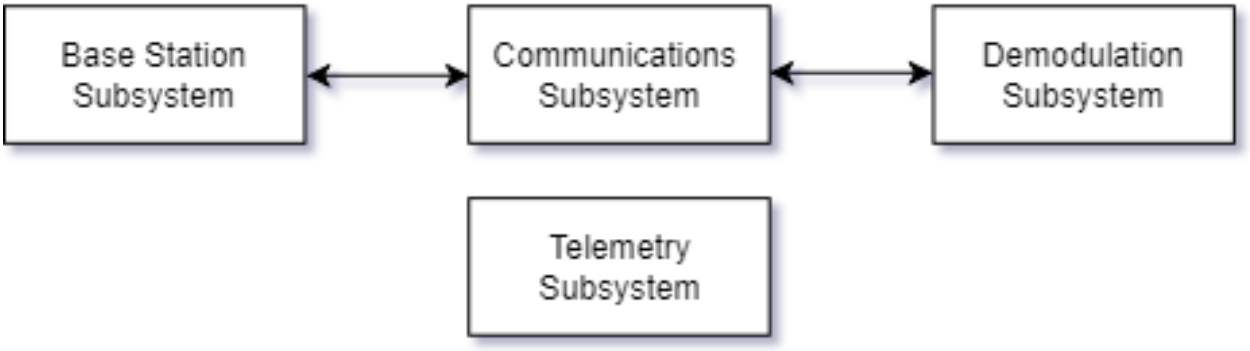


Figure 10: Software Architecture

4.2.1 Base Station Subsystem

The base station subsystem is responsible for passing mission parameters, receiving mission NAVAid data, and displaying the NAVAid data recorded for the test engineer to understand. The base system is divided into three subsystems: user interface subsystem, IO subsystem, and mission subsystem. Figure 10 (below) illustrates the relationships and interactions among the software objects in the base station subsystem.

User Interface Subystem:ViewDisplayManager: This module is used to serve and display the user interface. It is executed when the user starts the base station. Upon execution, the **ViewDisplayManager** serves the **MissionTypeSelectionView**.

MissionTypeSelectionView: This view displays options for the operation of the base station. It is displayed when the **ViewDisplayManager** is first executed. The view displays options to either view a live or previously recorded mission. Depending on the option chosen, the UI will display its respective option and log the option in the **StorageIO** module. If a live mission is chosen, then the **ViewDisplayManager** displays the **ZigBeeSettingsView**. If viewing a previous mission is selected, then the **ViewDisplayManager** will display the **PreviousMissionSettingsView**.

ZigbeeSettingsView: This view displayed by the **ViewDisplayManager** when the user selects to view a live mission. In this view, the user configures the settings that will allow the zigbee communication module to operate. The user will configure the serial port, baud rate, data bits, parity, stop bits, and flow control that the zigbee board will utilize. A button will be displayed to allow the user to test the connection to ensure that the connection can be established. If a connection can be established, the user will be able to progress to the next view, the **MissionProfileSettingsView**.

MissionView: This view is a superclass of **LiveMissionView** and **ReplayPreviousMissionView**. **MissionView** is instantiated by **LiveMissionView** and **ReplayPreviousMissionView**. **LiveMissionView** utilizes an instance of **ARCGISMapUtils** to display data in a 3D interactive map from the NAVAid data that is received from the zigbee and is stored in **StorageIO** module. Basic map controls will be provided by **MissionView**.

LiveMissionView: This view is a child class of **MissionView** superclass. **LiveMissionView** will read data in **StorageIO** provided by **ZigbeeIO**. This will be the IQ data of various test coordinates along test paths for NAVAids. **ARCGISMapUtils** will be utilized to draw coloured cylinders indicating the accuracy and magnitude of the data collected.

ReplayPreviousMissionView: This view is a child class of **MissionView** superclass. **ReplayPreviousMissionView** will read data in **StorageIO** provided by **PreviousMissionSettingsView**. This will be the previously recorded IQ data of various test coordinates along test paths for NAVAids. **ARCGISMapUtils** will be utilized to draw coloured cylinders indicating the accuracy and magnitude of the data collected.

ARCGISMapUtils: This module is a COTS SDK that will be utilized for drawing interactive maps that display the data acquired. All calls to its functions will be done by **MissionView**.

MissionProfileSettingsView: This view will be displayed by the **ViewDisplayManager** when the user has confirmed settings in **ZigbeeSettingsView**. The user will select which mission to be performed, ILS, VOR, or both, and the frequencies to tune in to for each respective test. Once the data is entered, it is sent to **StorageIO** for storage.

PreviousMissionSettingsView: This view will be displayed by the **ViewDisplayManager** when the user selects to view a previously recorded mission. This selection is recorded in **StorageIO**. This view will allow the user to select the mission files to be replayed in **ReplayPreviousMissionView**. Once the user has finalized their selection, then **ReplayPreviousMissionView** will be executed.

IO Subsystem: StorageIO: This module will be utilized by **MissionTypeSelectionView**, **ZigbeeIO**, **MissionProfileSettingsView**, **LiveMissionView**, **PreviousMissionSettingsView**, and **ReplayPreviousMissionView** to store and retrieve data and parameters. **StorageIO** will be implemented using a SQLite DB to store the data locally for the base station. Each storage key will have a corresponding storage value.

ZigbeeIO: This module will be utilized by **StorageIO** to both transmit and receive data

from the Demodulation Subsystem. On base station execution and data submitted in **ZigbeeSettingsView**, **ZigbeeIO** will established communication with the UAV ZigBee through the Communication Subsystem.

SerialPortIO: This module will be utilized by **ZigbeeIO** to communicate and establish connection with the zigbee module that will be plugged into the base station. Communication is done locally from the base station machine to the zigbee module in the communications subsystem through a serial port. **SerialPortIO** will utilize settings stored in **StorageIO** for zigbee connection.

DroneKit: This module is a COTS SDK that will be used to tap into the telemetry stream of the UAV and QGroundControl, done through a TCP connection. The data pulled from the telemetry stream will be stored in **StorageIO**. Upon application execution, the base station will establish a connection to the MAVLink stream through DroneKit.

Mission Subsystem: MissionProfile: This module is a domain data super class for **VORMissionProfile**, **ILSMissionProfile**. **MissionProfile** defines the datapoints that must be recorded for a mission. In the future, more mission profiles can be added to increase the testing functionality of the tool.

VORMissionProfile: This module is a subclass of **MissionProfile**. **VORMissionProfile** defines the profile of a VOR mission. Specifically, which datapoints must be recorded during a VOR mission and stored in **StorageIO**.

ILSMissionProfile: This module is a subclass of **MissionProfile**. **ILSMissionProfile** defines the profile of an ILS mission. Specifically, which datapoints must be recorded during an ILS mission and stored in **StorageIO**.

MissionPlannerlMission: This module a domain data structure class that will be used to define Mission Planner mission path data that will be stored in **StorageIO**. Data from Mission Planner pulled using **DroneKit** will be stored in **StorageIO** using the data structure of **MissionPlannerMission**.

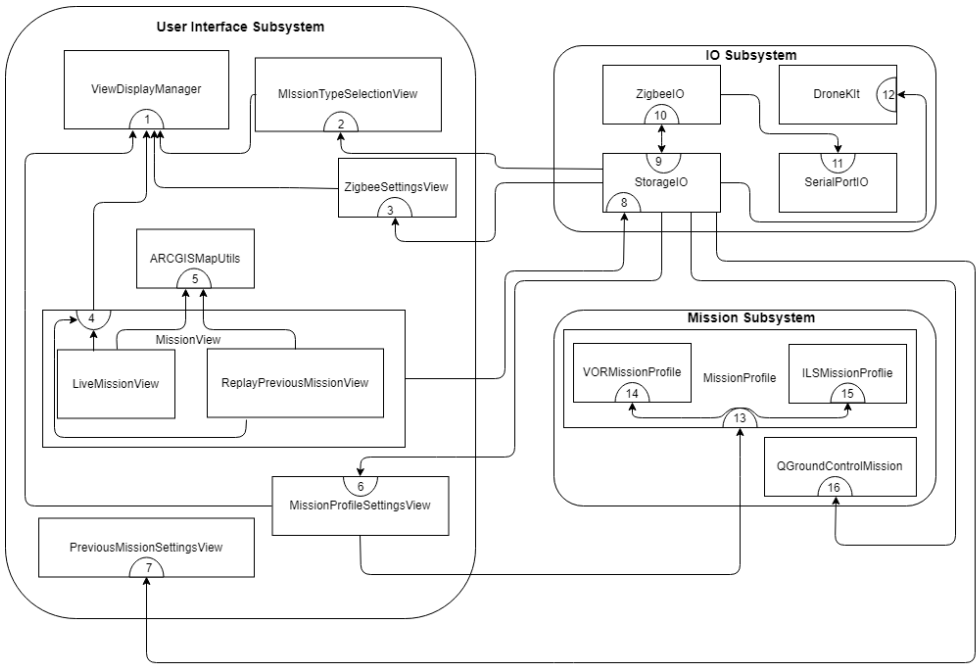


Figure 11: Collaboration Graph of Base Station Subsystem

4.2.2 Communication Subsystem

The communications subsystem is responsible for moving data from the demodulation subsystem to the base station and vice versa. Data is transferred over the ZigBee network described in section 4.1. Data coming from the demodulation subsystem consists of I/Q vectors, and the data going from the base station to the drone platform contains mission parameters for use in mission setup. Note that data used for telemetry is self-contained in the telemetry subsystem handled with Mission Planner.

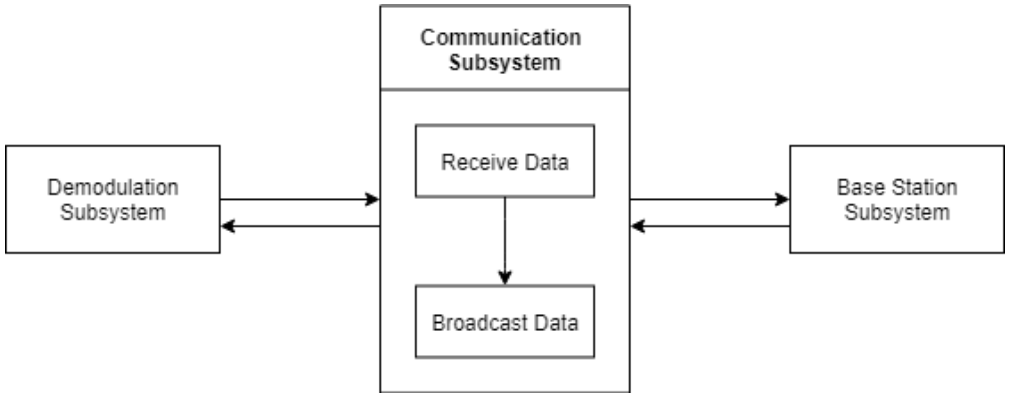


Figure 12: Communication Subsystem

4.2.3 Demodulation System

The demodulation subsystem has the responsibility of transforming the raw data collected by the SDR and GPS module and converting it to a useful piece of data that can be logged and transferred to the base station for use. We call this data packet an I/Q vector. The demodulation subsystem gets its input data from a user defined UDP port, which is producing raw signal data from the SDR by GQRX. When the demodulation subsystem collects the raw data from the UDP port, it processes the signals based on if it is a GPS, VOR, or ILS signal type. The subsystem then passes this data to the communications subsystem to be sent over the ZigBee network to the base station.

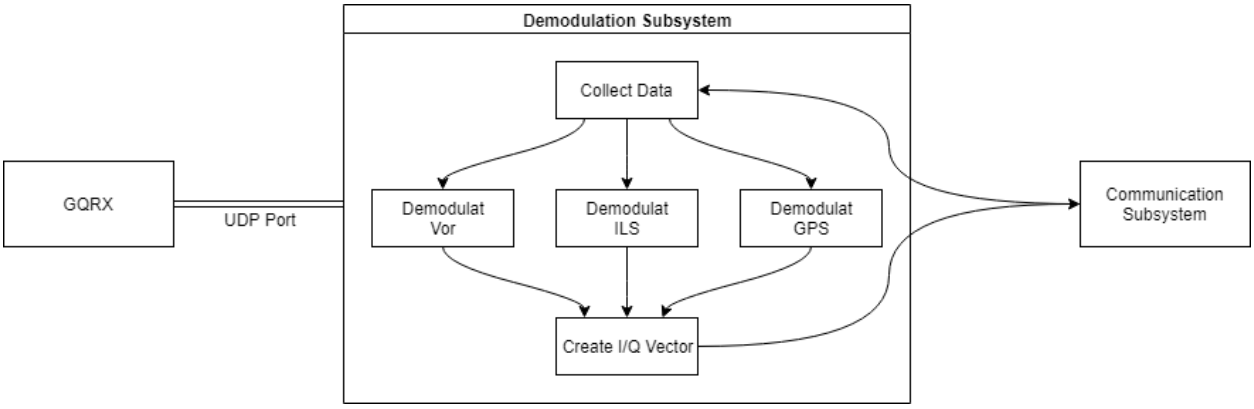


Figure 13: Demodulation Subsystem

The process logic for demodulating the ILS and Vor signals are shown below:

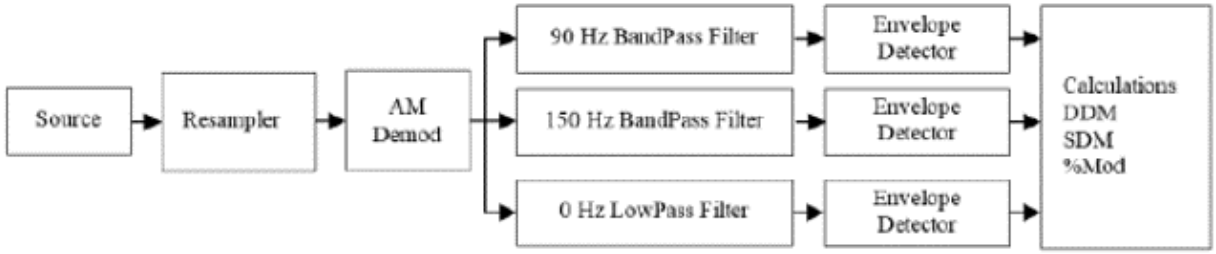


Figure 14: ILS Demodulation Schema

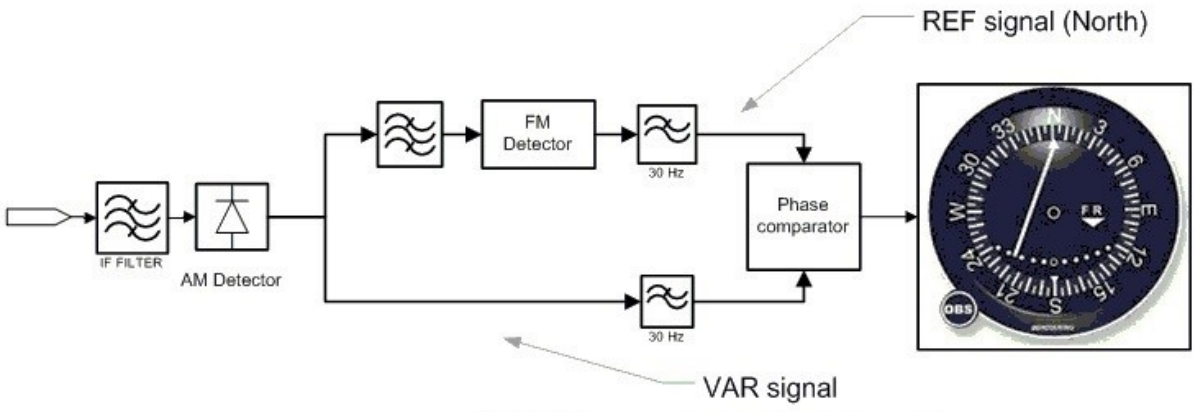


Figure 15: VOR Demodulation Schema
(Source: polaris-gnss.com/Alpha_RTK_Receiver_User_Guide.pdf)

4.2.4 Telemetry Subsection

Telemetry in the PIGEONS system is handled by QGroundControl (QGC) on the software side, and the Pixhawk4 on the hardware end. QGroundControl uses a Mavlink protocol to communicate telemetry commands to the drone. QGC is utilized to plan out flight paths and issue telemetry commands to the drone. In future work, we hope to utilize QGCs custom plug-in capabilities to display mission data within QGC; but for now, we are utilizing a separate piece of software in the base station subsystem to show mission details.

4.3 Internal Communications Detailed Design

As mentioned, a UDP protocol is used to transfer data from GQRX to our Python script demodulating the data is set up on localhost over port 5005. These are the default setting that can be adjusted by the end user. The data sent over the UDP server is the raw audio data from the SDR collected at 128kbps at a bit depth of 12 bits. The data sent over the XBee network consists of String data type packets transmitted at 150kbps over the 2.4Ghz frequency. Data packets consist of the following structure to be formatted in JSON:

Key (Command)	Size	Value
Packet_ID	Int	677978 for command
Mission_Type	Int	737683 for ILS, 867982 for VOR, 66111 for both.
Frequency_1_Type	Int	737683 for ILS, 867982 for VOR
Frequency_1_Value	float	Frequency value or -1 for null.
Frequency_2_Type	Int	737683 for ILS, 867982 for VOR, or null
Frequency_2_Value	float	Frequency value or -1 for null.

Table 1: Mission configuration packet sent from base station to demodulation subsystem.

Key	Size	Value
Time_UTC	Long	Time in UTC of sample point collected.
GPS_Lat	Long	GPS Latitude associated with sample point collected.
GPS_Long	Long	GPS Longitude associated with sample point collected.
GPS_Alt	Long	GPS Altitude associated with sample point collected.
Data_Sample_Type	Int	Sample type identifier. 737683 for ILS, 867982 for VOR.
Data_Sample_IQ	Long	Numerical representation of data value found.

Table 2: Data sample packet to be sent from demodulation subsystem to Base Station for storage and processing.

The following state transition diagram gives an overview of the control flow for the internal communications on the drone platform:

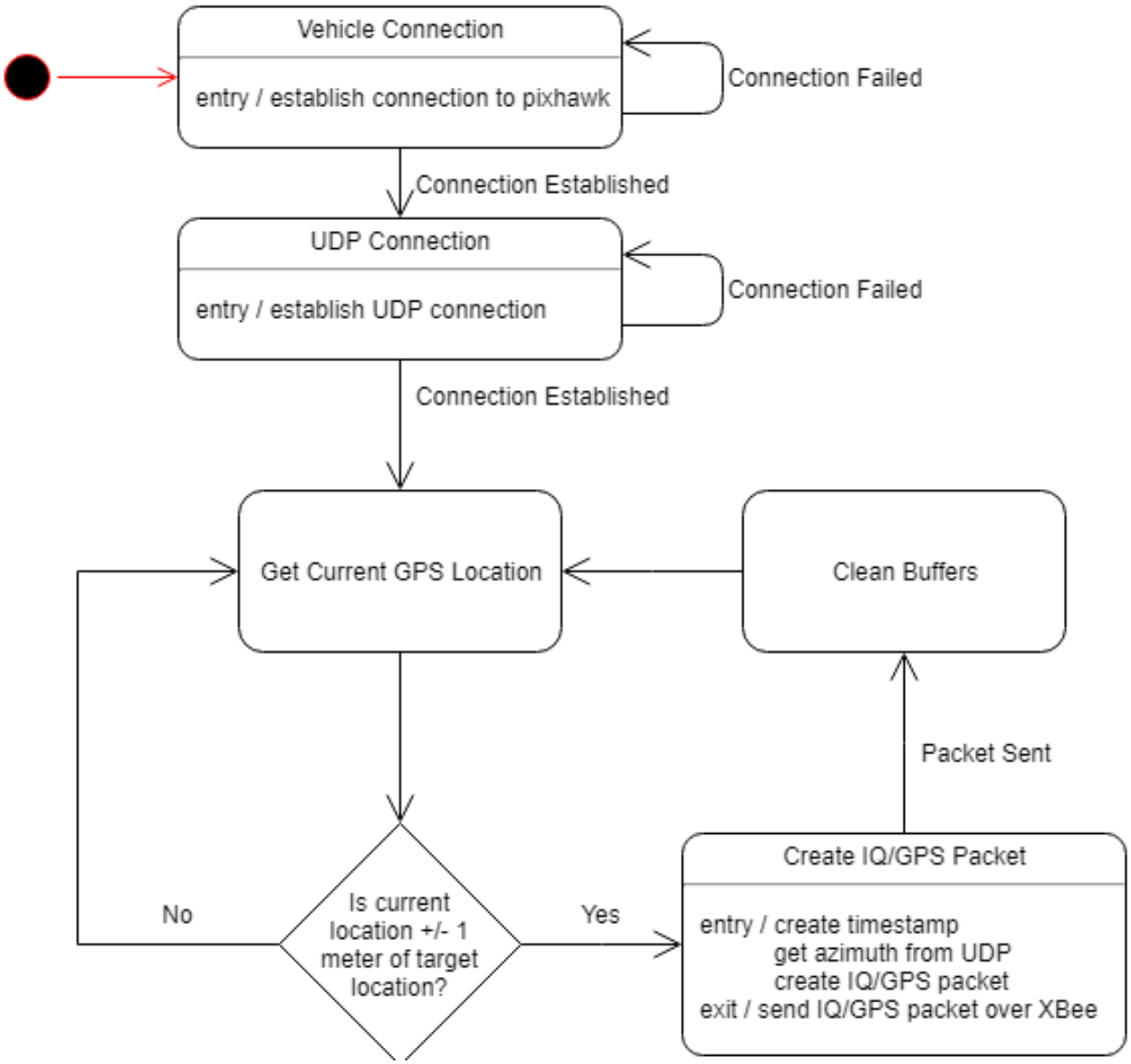


Figure 16: State Transition Diagram for Raspberry Pi Control Flow

5 External Interfaces

This section will examine the external systems that will interact with the PIGEONS system.

5.1 Interface Architecture

The primary external communication source for the PIGEONS system will be the ILS/VOR transmitters. The ILS transmitter will provide the PIGEONS with carrier wave the encompasses two modulated signals. Based on where the system is, the system will collect various strengths in signals. The transmitters have a constant directional output. This means that as the system moves farther away from the focus of the modulated signal, the system will receive a weaker signal. VOR will provide the system with two AM signals. Once signal will be used as a reference. The other signal will be a directional signal that rotates 360 degrees around the transmitter. The system will be able to determine the difference in phase between the variable signal and the reference signal to acquire the bearing of the system.

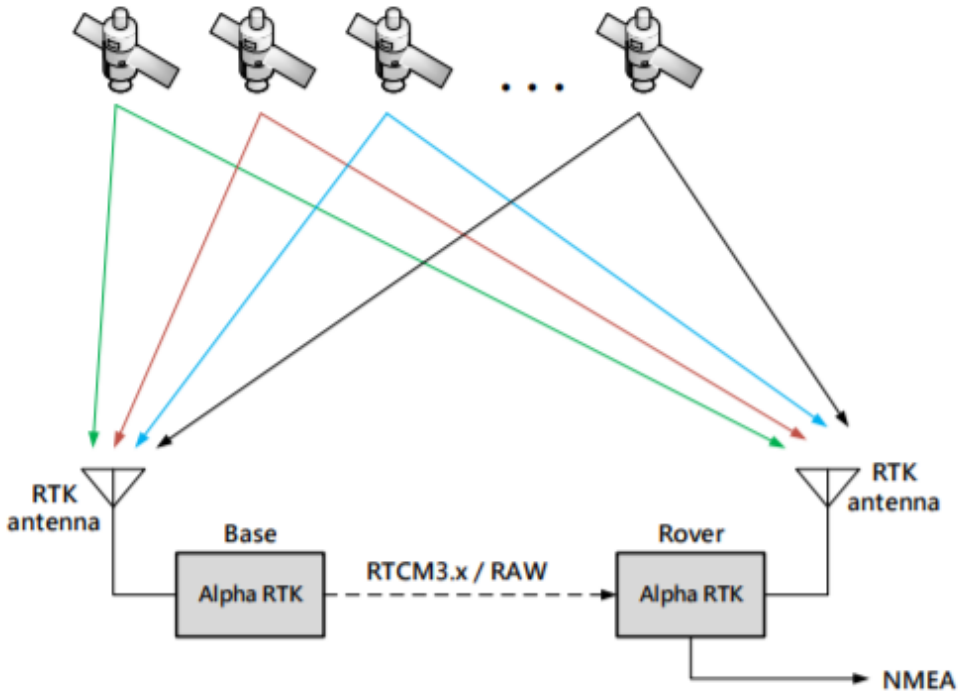


Figure 17: RTK Functionality
Source: polaris-gnss.com/Alpha_RTK_Receiver_User_Guide.pdf

The other external communication for the system will be provided via GPS satellites which

be acquired using the RTK GPS receivers. A figure of the communication between the ground station, the PIGEONS system and the GPS satellites is shown above. There will be two GPS receivers; one on the ground station and one on the system. The ground station will acquire a signal from at least 3 antennas which will allow for the calculation of the ground stations coordinates. The ground station will be at a known location as it will be acquired using previous professional survey data. This will allow for the calculation of the error in measurement provided by the GPS receiver and satellite. The PIGEONS system will then receive the error in measurement and correct the measurement that it acquires from the satellites to provide an accurate reading.

5.2 Interface Detailed Design

The following class diagram shows which classes use which interfaces. The Pixhawk, SDR, and XBee-Pro 900HP are all off the shelf hardware that the PIGEONS system requires to function.

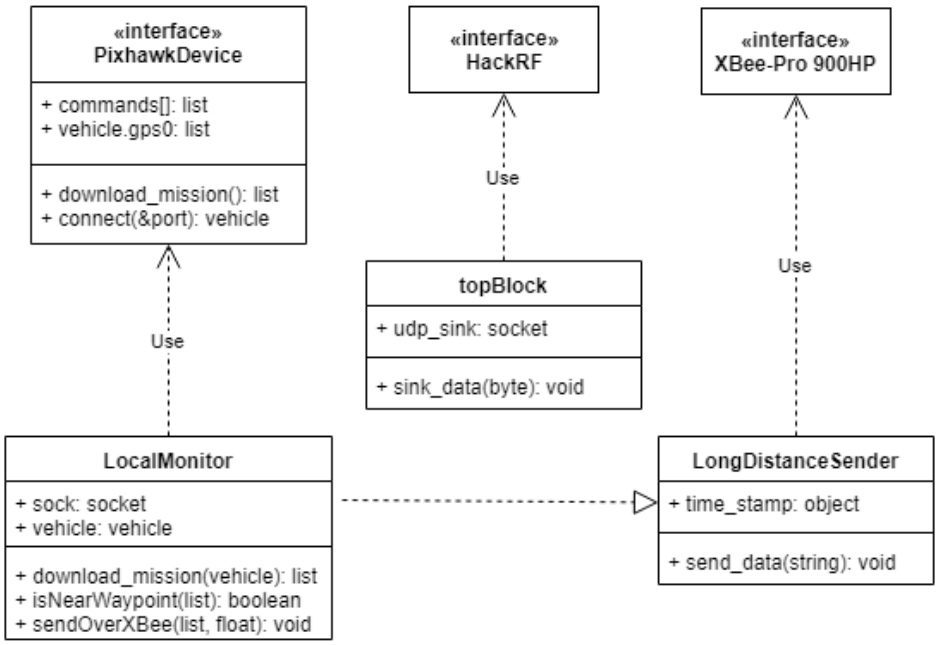


Figure 18: Internal Pi Class Diagram

Besides the Pixhawk, SDR, and XBee mentioned above, the last external interfaces are the ILS and VORTAC signals coming from the navigational antennas. Physically collecting these signals is achieved with an off the shelf SDR module. In order to demodulate the raw signals, please refer back to Figure 11 and Figure 12. Demodulation is achieved by utilizing GNU Radio Companion.

6 System Integrity Controls

The designed system will be used to pick up ILS and VOR signals. These signals can be picked up by anyone, assuming they have an antenna and a way to demodulate the signal, such as with a spectrum analyzer. With this in mind, the data being sent between drone and ground station is not a secret, but should not be intercepted to ensure integrity of the data. This will be taken care of when the Xbee modules are programmed. The Xbees must be setup on the same network, requiring a certain code to be known to enter the network, acting as a password.

Along with this, the raw data will be stored on a local USB drive. This will give the user the original, unmodified data if needed.

The DX6 controller will have switches A and B used to control the state of the drone. Switch A is the kill switch and switch B switches between autonomous mode, manual mode, and return to land mode. By activating switch A, power will be cut to the motors, causing the drone to fall. Switch B will be used if manual mode needs to override the autonomous mode. The switches are shown below in Figure 19.

Figure 19: Controller with switches A and B labeled.

To ensure that the drone in operated in a safe manor, prior to using it to collect signals, it will go through several tests. These three tests are to test 1) stability and yaw, 2) safety features, and 3) autonomous flight mode. These were picked in this order to make sure the drone flies, and is safe to fly, before using it. To test the stability and yaw, the drone will be operated manually, flying in several directions for a short distance then being held in place. This will be used to ensure the drone is operating correctly. Next will be to test the safety functions. To do this, the drone will hover about a foot off of the ground, and then the power to the motors will be killed, causing the drone to fall. This would only be used in emergencies to stop the drone from flying away. Next, the drone will hover a few feet in the air due to a program being placed in Mission Planner. The pilot will then take over manual control. This will be used to ensure the drone can be taken over in the middle of the mission, and if this does not go according to plan, the emergency motor cut can be used. Lastly, having a mode that causes the drone to return to launch will be implemented. The third test will be setting up a small course in Mission Planner to test the autopilot functions of the program. If this works, the drone should be ready for autopilot usage.