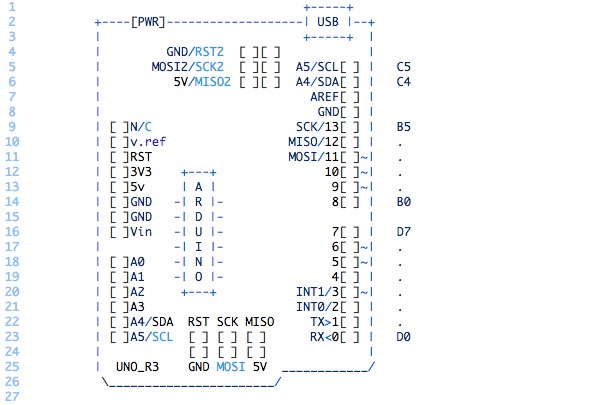
System Design Document for

EGR101 Boe Bot Simulation Software  
  
Emily Connearney  
Luke Crump  
Vivian Dang  
Keely Mashburn  
Daniel Khalil



|  |  |
| --- | --- |
| Version | Date |
| 1.0 | 9/22/2021 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Contents**

[**1 Introduction** **2**](#_Toc18762)

[1.1 Purpose and Scope 2](#_Toc18763)

[1.2 Project Executive Summary 2](#_Toc18764)

[1.2.1 System Overview 2](#_Toc18765)

[1.2.2 Design Constraints 4](#_Toc18766)

[1.2.3 Future Contingencies 5](#_Toc18767)

[1.3 Document Organization 5](#_Toc18768)

[1.4 Project References 5](#_Toc18769)

[1.5 Glossary 5](#_Toc18770)

[**2 System Architecture** **6**](#_Toc18771)

[2.1 System Hardware Architecture 6](#_Toc18772)

[2.2 System Software Architecture 9](#_Toc18773)

[2.3 Internal Communications Architecture 10](#_Toc18774)

[**3 Human-Machine Interface** **10**](#_Toc18775)

[3.1 Inputs 10](#_Toc18776)

[3.2 Outputs 11](#_Toc18777)

[**4 Detailed Design** **12**](#_Toc18778)

[4.1 Hardware Detailed Design 12](#_Toc18779)

[4.2 Software Detailed Design 15](#_Toc18780)

[4.2.1 Base Station Subsystem 16](#_Toc18781)

[4.2.2 Communication Subsystem 20](#_Toc18782)

[4.2.3 Demodulation System 20](#_Toc18783)

[4.2.4 Telemetry Subsection 22](#_Toc18784)

[4.3 Internal Communications Detailed Design 22](#_Toc18785)

[**5 External Interfaces** **25**](#_Toc18786)

[5.1 Interface Architecture 25](#_Toc18787)

[5.2 Interface Detailed Design 26](#_Toc18788)

[**6 System Integrity Controls** **27**](#_Toc18789)

# Introduction

## Purpose and Scope

This document describes the system requirements, operating environment, system and sub-system architecture, file and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces for the EGR101 Simulation System project.

## Project Executive Summary

This section provides an overview of the EGR101 Simulation System project from a management perspective, showing the framework with which, the system design was conceived.

### System Overview

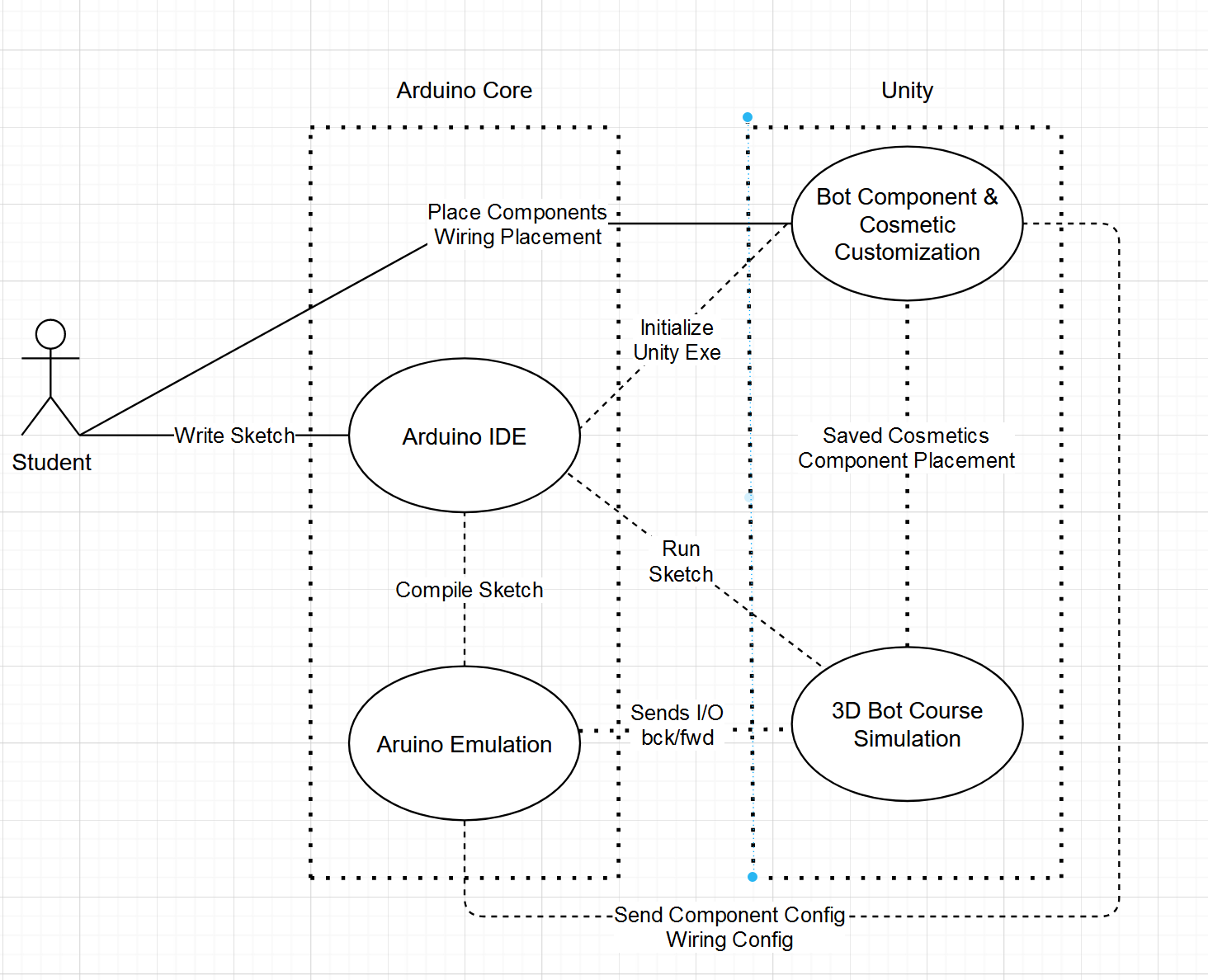


Figure 1: Use Case Diagram for EGR101 Simulation System.

### Design Constraints

This project design has two main constraints. The first constraint to limit product scope would be to only contain components needed from the EGR101 Arduino Kit, and there will be a limit on functionality of components such that components cannot perform behavior that the robot cannot.

### Future Contingencies

## Document Organization

## Project Reference

## Glossary

# System Architecture

Figure 3: PIGEONS Hardware Overview

## System Software Architecture

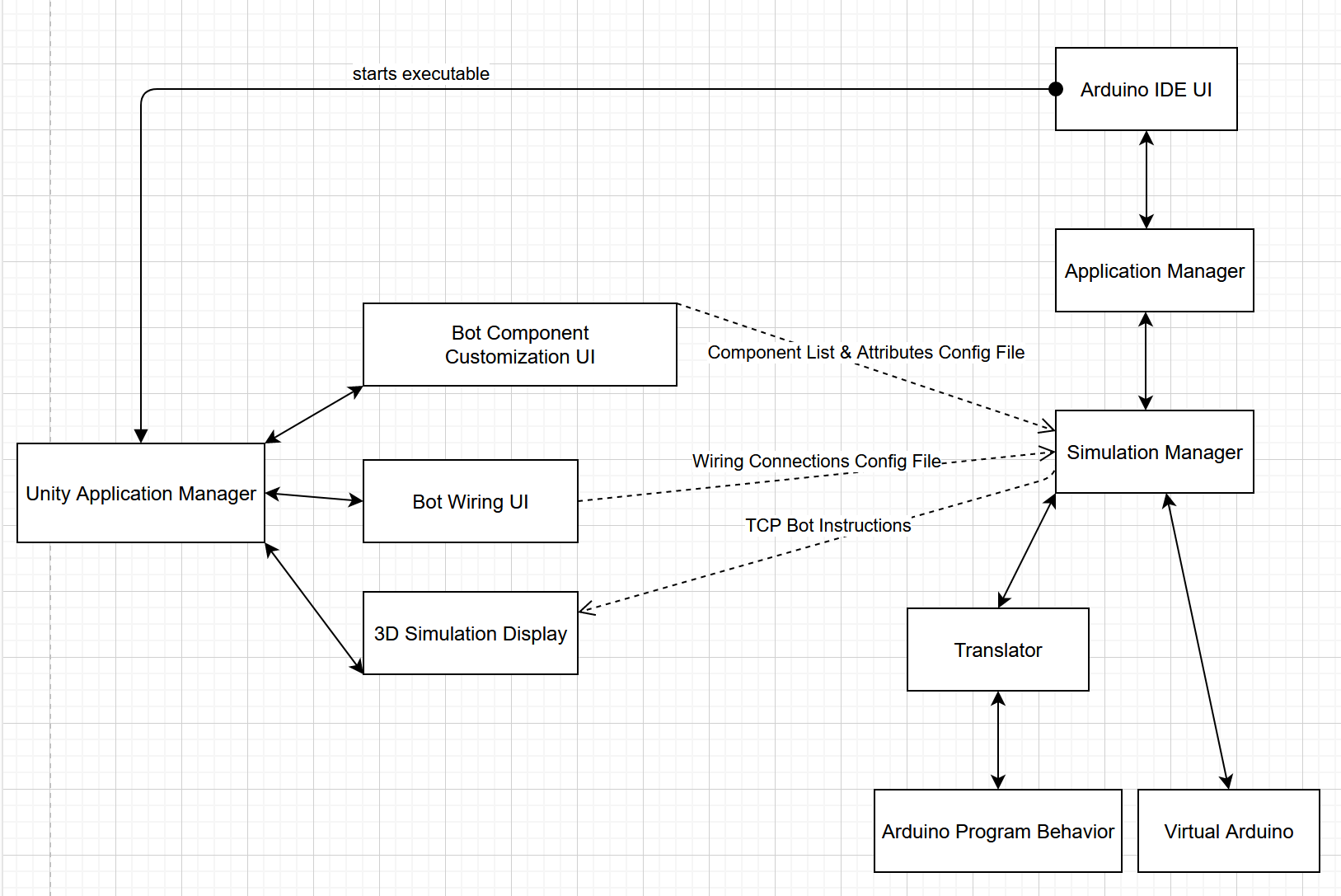


Figure 4: Software Architecture Overview

## Internal Communications Architecture

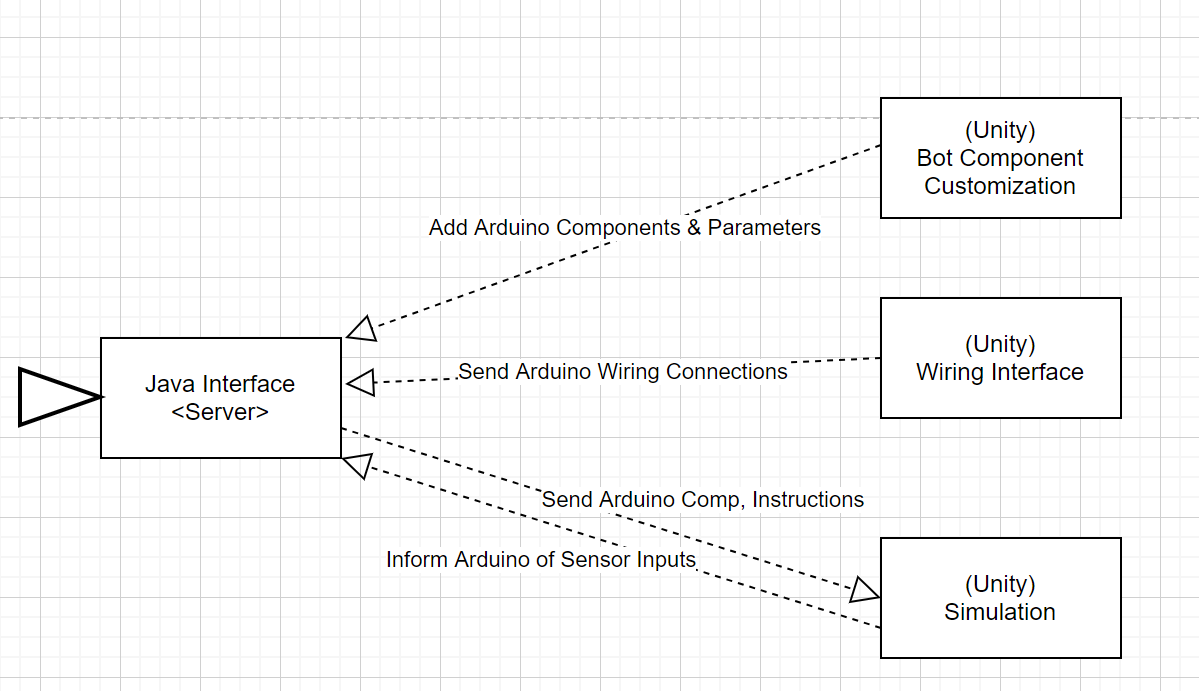


Figure 5: Internal Communications Architecture Overview

PUT SOMETHING ABOUT UNITY TCP CONNECTION WITH JAVA HERE

# Human-Machine Interface

## Inputs

MOCKUP GOES HERE

Figure 6: User Inputs

## Outputs

## Software Detailed Design

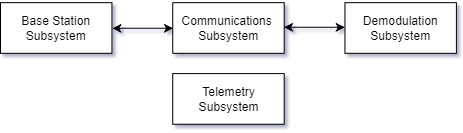


Figure 10: Software Architecture

MORE DESIGN SOFTWARE

### Simulated Arduino Engine

LIST CORE ARDUINO CLASSES AND THEIR DES RIPTIONS

CORE UML

### 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.

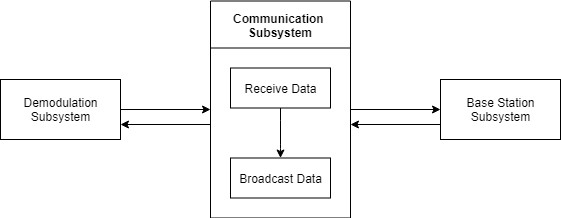


Figure 12: Communication Subsystem

### 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.

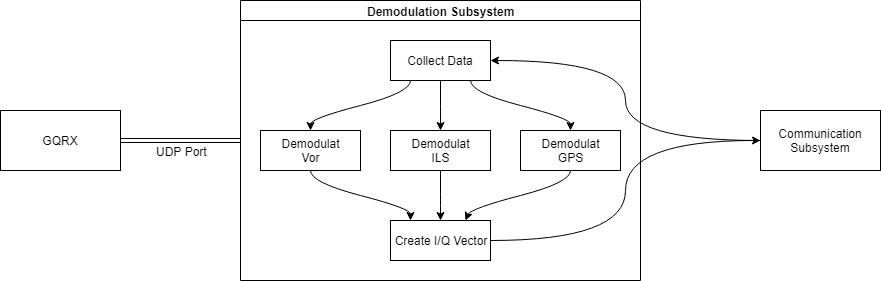


Figure 13: Demodulation Subsystem

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

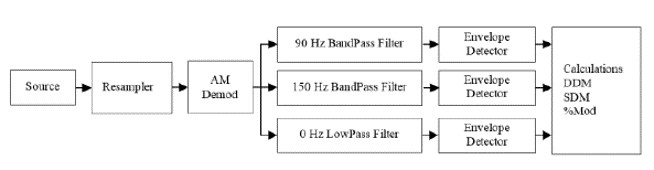


Figure 14: ILS Demodulation Schema

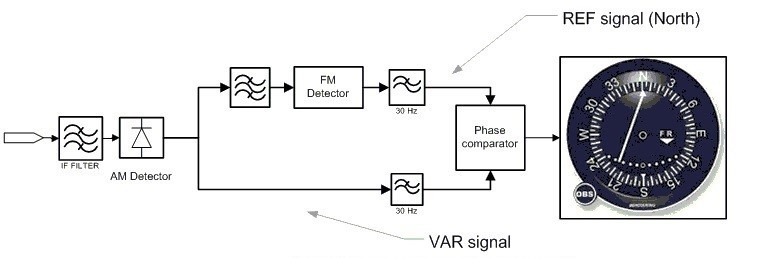


Figure 15: VOR Demodulation Schema

(Source: polaris-gnss.com/Alpha RTK Receiver User Guide.pdf)

### 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.

## 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:

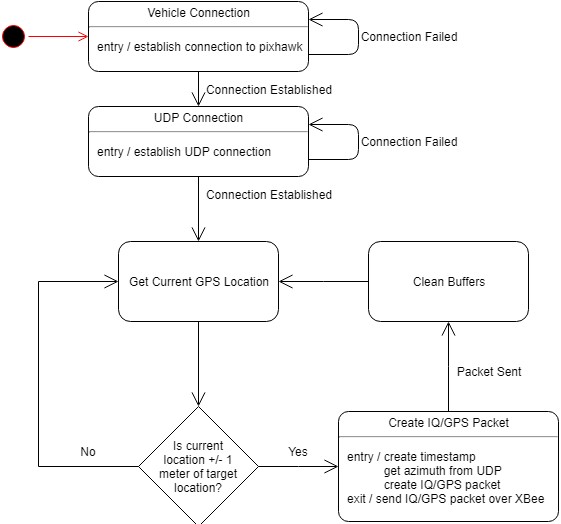


Figure 16: State Transition Diagram for Raspberry Pi Control Flow

# External Interfaces

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

## 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.

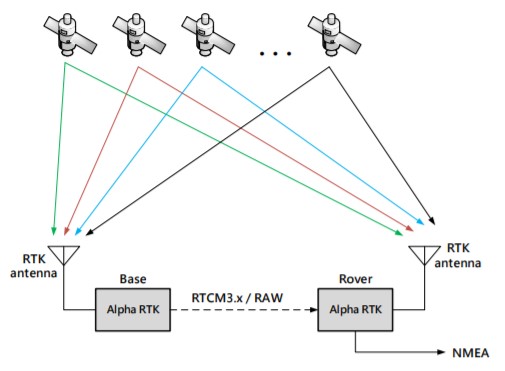


Figure 17: RTK Functionality

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.

## 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.

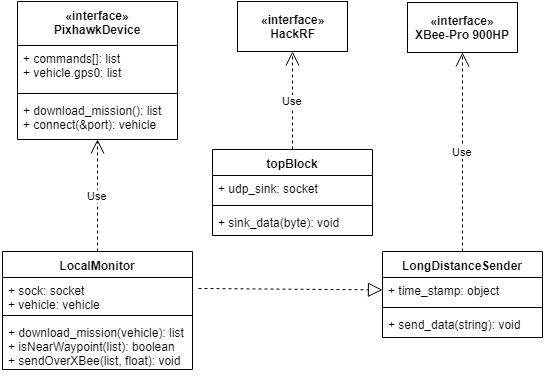


Figure 18: Internal Pi Class Diagram

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.

# 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.