System Design Document

emStart: A Satellite Transceiver and Small Radio Telescope Emulator

Ivan Borra, Matthew Gasper, Matthew Grabasch, TJ Scherer, and Matthew Selph

Author	**Date**	**Version**
All	9/28/21	1.0
All	10/26/21	1.1
All	11/30/21	1.2
All	2/4/22	1.3

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Purpose and Scope

This document aims to lay out an overview of what the emStart project will consist of, and clearly define what a finished product looks like. By the end, an understanding of what hardware will be needed, what objectives the software will look to accomplish, and what human interfaces will be.

1.2. Project Executive Summary

The goal of this project is to first create a model of a real-world antenna for data collection and then to move that implementation to an already existing full-scale antenna. The existing implementation of the antenna is a fixed arm that does not move with the Earth's rotation. The purpose of the Antenna is to capture satellite data from a satellite that will be moving with respect to the antenna. We plan on using a pan tilt configuration for microservers that can best match the rotation of the earth, and therefore extend the capture time frame of data from the satellite, helping with the debugging of the received signal.

This emulator will include the construction of a mockup satellite that will transmit and auintuate data, and a model ground station that will receive that data, and attempt to move with the expected movement of the satellite.

1.2.1. System Overview

The system will have three main subsystems. Firstly, the "Earth" platform which will be a two degree of freedom pan tilt configuration in order to simulate the Earth's movement. On top of this will sit the second platform which is the ground station which also is a two degree of freedom pan tilt configuration using micro servos. Its objective being to control the position of the antenna which will be mounted on top of the last extending servo. Finally the last system being a stationary transmitting "satellite" which will have an auntinuator in order to mimic the effects of an off-positioned ground station.

1.2.2. Design Constraints

The project has a few constraints. The first relates to the man tilt configuration for the arm. Size is another constraint, with the model needing to be generally able to fit on a tabletop. Hardware is a minor constraint, with three options for cheap, mid-range, and more expensive Software Defined Radios being presented to us. Another constraint is the minimum PC requirements needed to run the software, which is recommended at having at least Windows 7, a 64 bit processor on either Intel or AMD’s architecture, 4 GB of RAM, and about 5 GB of open storage. The final constraint is the budget, which has been set at $3400.

1.2.3. Future Contingencies

With most of the hardware and software we are using being open source, we have a good amount of flexibility to work with along the way. The plan is to use easily programmed devices such as Arduino Nano, raspberry pi, and Hack RF boards. This should make any changes to the mission easy as the hardware should be flexible, and the change will likely just be a software change.

1.3. Document Organization

This document is organized in the following fashion:

1. Introduction and project overview
2. System Architecture Overview
3. Human Machine Interfaces
4. Detailed Design
5. External Interfaces
6. System Integrity Controls

1.4. Project References

Arduino. (2015, April 23). *Servo*. Retrieved from Arduino: https://www.arduino.cc/reference/en/libraries/servo/

Gadgets, G. S. (2017, March 11). *SDR with HackRF One, Lesson 1 - Welcome - 720p*. Retrieved from YouTube: https://www.youtube.com/watch?v=zNUCiGVJQo0&ab_channel=PE1PID

Kazuaki, H. (2019, August 27). *Parallax FeedBack 360 Servo Control Library 4 Arduino*. Retrieved from Github: https://github.com/HyodaKazuaki/Parallax-FeedBack-360-Servo-Control-Library-4-Arduino

Massachusetts Institute of Technology. (n.d.). *SRT: The Small Radio Telescope*. Retrieved from MIT Haystack Observatory: https://www.haystack.mit.edu/haystack-public-outreach/srt-the-small-radio-telescope-for-education/

Parallax LLC. (2017, August 12). *Parallax Feedback 360° High-Speed Servo.* Retrieved from Parallax: https://www.pololu.com/file/0J1395/900-00360-Feedback-360-HS-Servo-v1.2.pdf

The Robotics Back-End. (2021). Retrieved from Raspberry Pi Arduino Serial Communication – Everything You Need To Know: https://roboticsbackend.com/raspberry-pi-arduino-serial-communication/

1.5. Glossary

DOF - Degrees of freedom
GPSDO - GPS disciplined oscillator
PWM- Pulse width modulation
RF- Radio frequency
SDR - Software-defined radio
SMA - SubMiniature version A
SRS - System Requirements Specification
SRT - Small radio telescope

2. SYSTEM ARCHITECTURE

2.1. System Hardware Architecture

There are three main subsystems in the design. These subsystems are the Earth, the ground station, and the system in space. Each subsystem serves a unique purpose in the emulation, allowing rapid prototyping and testing different angles between the ground station and the system in space.

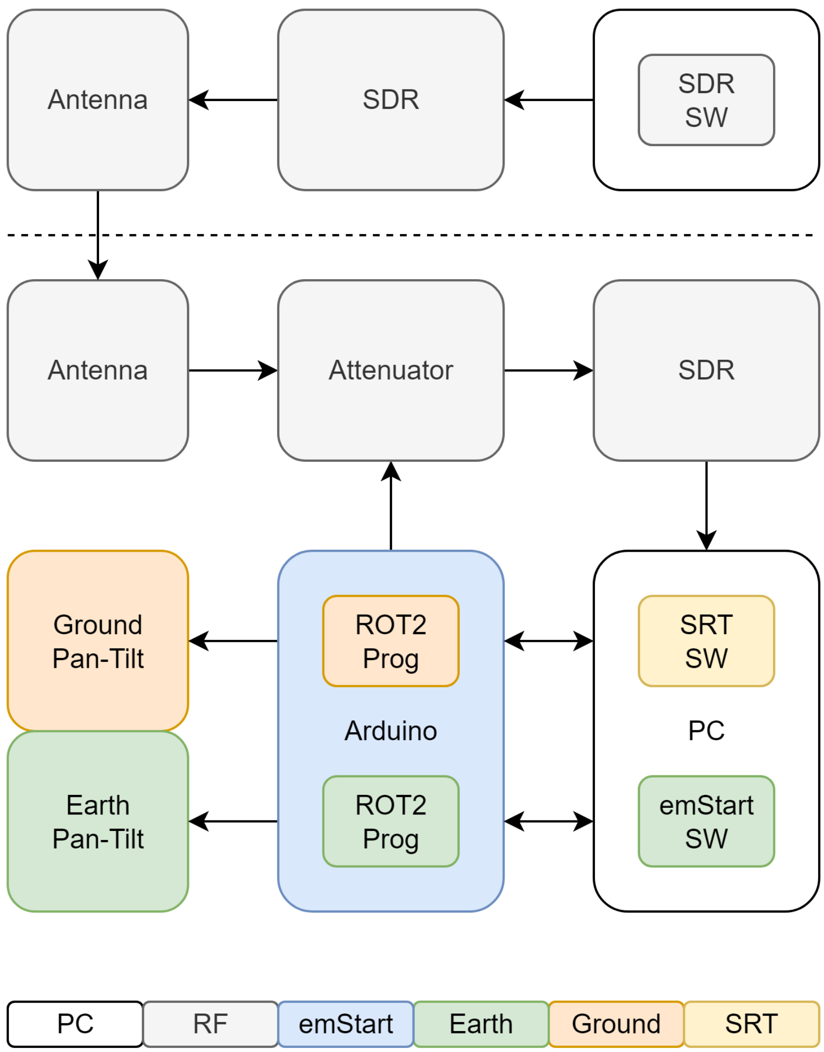

Figure 1: emStart Hardware Architecture

The Earth subsystem is conceptually independent of the other two subsystems and makes minimal communication with them. The only shared information is the current date and time of the emulation in order to synchronize the ground station, as well as some handshaking to set up the emulation from the start. The Earth subsystem will serve as a platform for the ground station, so the two subsystems will move in unison. The Earth subsystem is intended to emulate the rotation of the Earth relative to the system in space. It consists of a robotic arm with three degrees of freedom, which is directly controlled by a computer. This will allow the arm to reposition a platform to which the ground station is fastened.

The centerpiece of the ground station is the communications system. Using a software-defined radio (SDR) and an antenna that can rotate with two directions of freedom, the communications system can receive transmissions from the system in space. The computer in the design serves two purposes. The first purpose is to interpret the received transmissions from the software-defined radio. The second purpose is to send commands to a microcontroller to alter the rotation of the antenna. These two operations are performed independent of each other on two separate ports of the computer.

The system in space generates the transmissions which the ground station receives. This subsystem is driven by a computer that sends commands to the software-defined radio, which then sends its signal out through the fixed antenna. This subsystem is entirely stationary with no moving parts.

2.2. System Software Architecture

For the Earth subsystem, the software must control the position of the emulated Earth. In the ground station, the software must perform two independent operations: receive radio signals from space on software-defined radio and control the position of the antenna. Finally, the system in space must transmit radio signals using a software-defined radio.

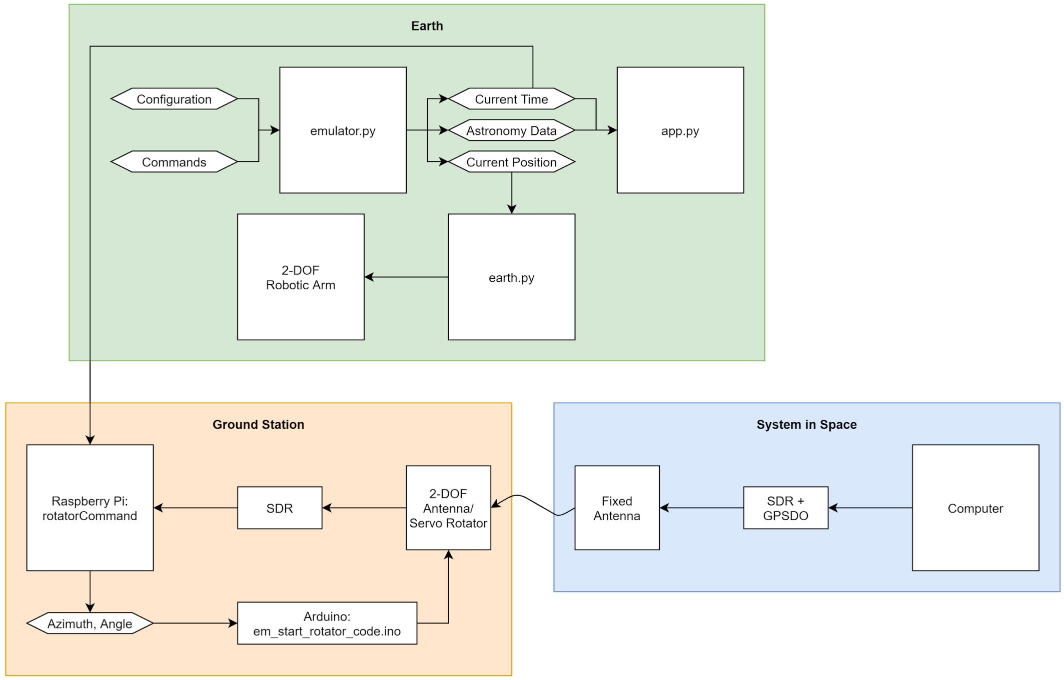

Figure 2: emStart Software Architecture

The Earth subsystem has a computer that will be running all of the software we require. The purpose of this software is to translate a stream of astronomy data into commands for the robotic arm, allowing the robotic arm to emulate the relative rotation of the Earth so that the system in space can remain stationary. Our software must accept the astronomy data as an input to understand the relative position of the Earth, then translate that relative position to the robotic arm to properly emulate the Earth's rotation and orbit. Both ends of this process will have their own respective APIs which we must adapt to properly pass information through our design. The Earth will also need to transmit the current time in order to synchronize the other subsystems.

The ground station computer will use a python script to send commands to the antenna microcontroller, which will then interpret those commands and rotate the servos accordingly. The computer must also send commands to the software-defined radio. This will be accomplished using GNU Radio which has a GUI interface where blocks can be placed to create certain functionalities in the SDR.

The system in space only has software for the SDR. The computer must send commands to the software-defined radio. This will be accomplished using GNU Radio as well.

All of the programming will be done using Python as a universal language on two different platforms. Some of the Python code will be interpreted by a laptop or similar device, and some of it will be interpreted by a Raspberry Pi 4 and communicate with a host computer over a serial port. This will reduce the amount of software required for each machine in the system, making it easier to change them out as needed.

2.3. Internal Communications Architecture

Most communication in the design is constrained within each individual subsystem. The only instance of the subsystem boundary being crossed is the radio signals being transmitted from the system in space to the ground station. Other than that, standard communications such as USB and PWM are used for sending commands and controlling servos respectively.

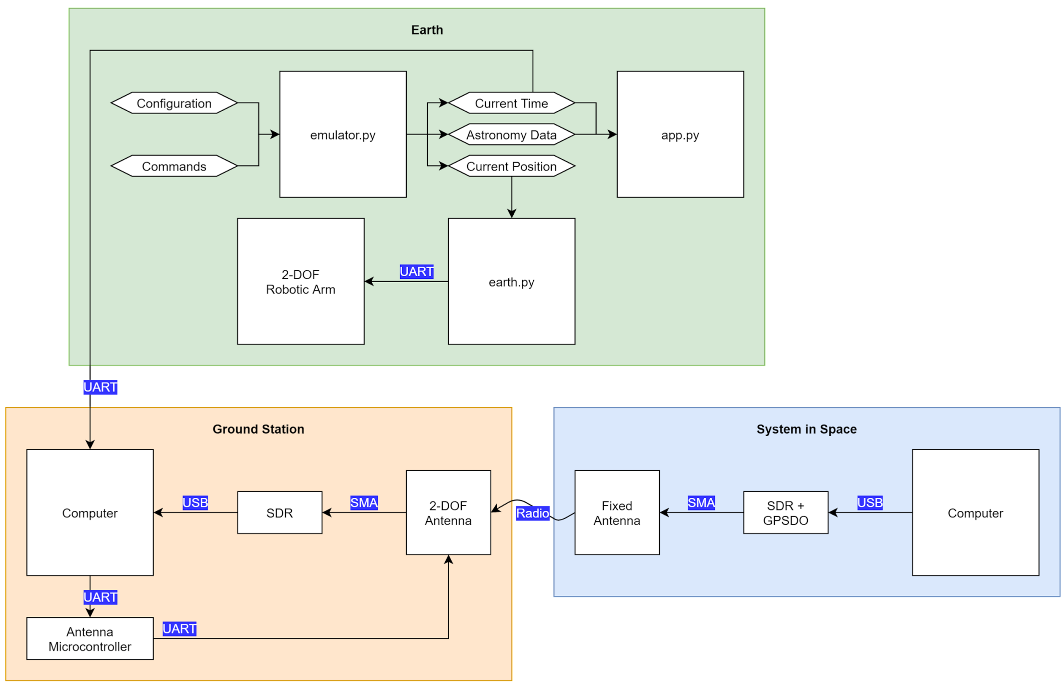

Figure 3: emStart Internal Communications Architecture

The Earth subsystem will be fully controlled by a computer. This computer will communicate with a 2-DOF robotic arm over USB to adjust the tilt and rotation of the Earth platform.

The ground station will have an antenna with 2-DOF rotation. The antenna will receive radio signals transmitted by the system in space. This antenna will then connect to a software-defined radio with analog signals using an SMA connector. The software-defined radio then communicates with the computer over USB. The antenna mount will be controlled by a microcontroller using PWM signals. This microcontroller will be connected to the computer over a separate USB port and communicate using a message queue to send commands from the computer to the microcontroller.

The system in space will have a stationary antenna that communicates with another software-defined radio using analog signals. This software-defined radio will then interpret this input and make the information available to the computer over USB.

3. HUMAN-MACHINE INTERFACE

EmStart has 3 main parts: the system in space, the simulated satellite, the system on the ground, the simulated receiver antenna station, and the robotic arm which is simulating Earth. The system in space is meant to transmit space signals, the system on the ground that controls the antenna which moves toward the space signal, and the emulation station which moves the system on the ground to mimic earth.

3.1. Inputs

Once initial setup procedures have been completed and the system is at its default main screen. The User creates a standard python virtual environment which handles all project files in use (Background Tasks). With the virtual environment created it opens a command prompt allowing the user to input the location data which consists of latitude, longitude and date.

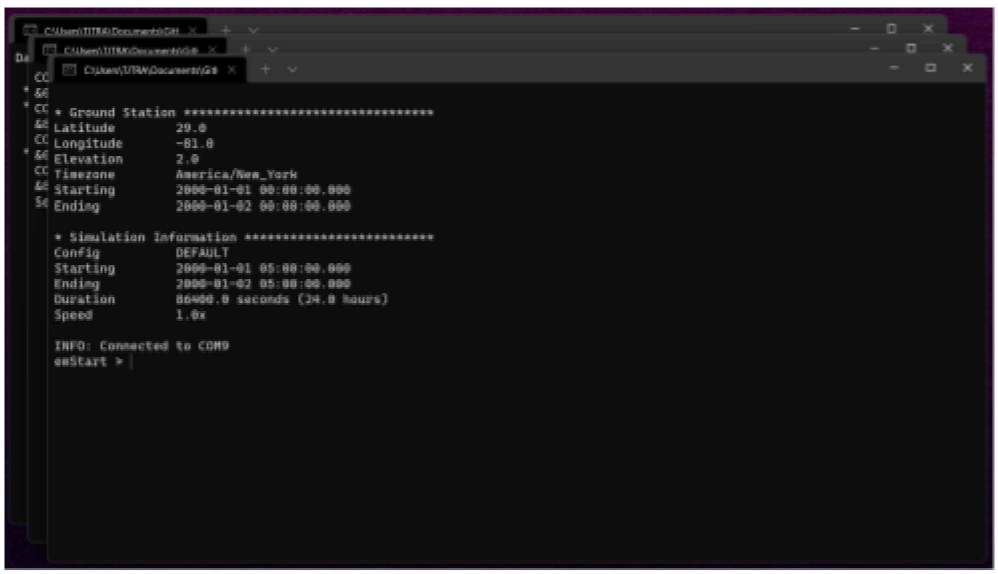

Figure 4: Emstart.bat Command prompt.

3.2. Outputs

When the system is initialized with the proper positional values of the tracking target i.e. the position you are on earth and the location of your target. The robotic arm “Earth” will orient itself in its starting position to mimic that of the initial conditions of a real system emulating it. Once the system is set to begin “Earth” will move according to the imputed time-frame for both objects in question. The same can be said of the “Rotator” mounted on top of the “Earth.” At the same time the azimuth and altitude are shown on a Plotly GUI on an HTML website.

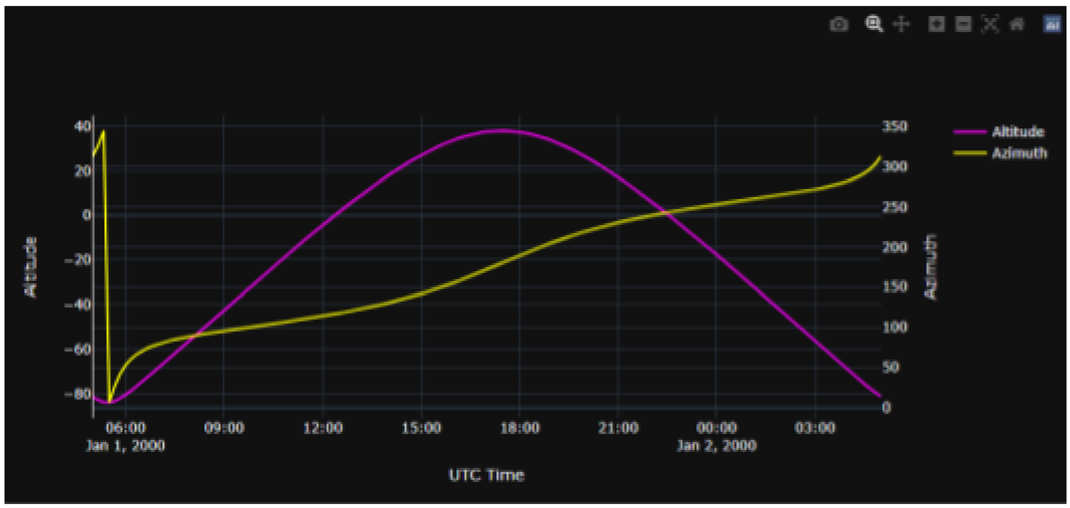

Figure 5: Plotly Graph.

4. DETAILED DESIGN

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

4.1. Hardware Detailed Design

The design of the hardware system can be viewed through the lens of two distinct systems. The first is the “Earth” emulation system and the second, being the rotator antenna system.

The hardware components are connected as follows to perform the overarching goals of the system. First with the Earth Emulation system which its main control is stemmed from a laptop or PC connected to the robotic-arm via a USB-A to USB-C connector. The laptop/PC is also connected to a Raspberry Pi 4 via a USB-C to USB-A connector which controls our second system, the rotator antenna system. This raspberry pi 4 is connected also to an Arduino nano via a USB mini-A to USB-C connector. Which is then connected to the electromechanical controlling points of two individual servos controlling azimuth and elevation for the patch antenna. Furthermore, the Raspberry Pi 4 is also connected to a HackRF via a USB connector which handles all communications received by a said patch antenna.

The Raspberry Pi 4 is the processor used to control the radio communications and the rotator emulator commands. It boots its operating system off an SD card which is inserted into the Pi 4, on this card is also stored all program data for its operation in the system. It will be used in such a way to receive communications from the Hack RF and perform operations relating to the data received. It will also be used in order to control the Rotator emulator via inputted astronomy data positions relative to its position on the emulated “Earth”. These commands it sends will be received by an Arduino Nano.

The Arduino Nano is the microcontroller used to control the rotator emulator. It does this by first receiving commands from the Raspberry Pi 4, which it then sends signals to two different servos: a 180-degree servo used for moving up and down and a 360 for moving around the base of the rotator. This allows them to adjust the patch antenna fixed to the end of the rotator to the correct orientation to receive incoming signals.

The Hack RF is a software-defined radio (SDR) that handles receiving of communication signals. It is connected to a patch antenna via an SMA connector on the Hack RF, it is also attached to the Raspberry Pi 4 via a USB connection to communicate its data. The antenna used for this system consists of a metal plane connected to the SMA connector wire.

The robotic arm platform is used to emulate earth. It has 6 available axes of movement providing superior movement control to the system. It is connected to a laptop or personal computer (PC) via a USB connection which can then be controlled by the code programmed on the computer.

4.2. Software Detailed Design

The emStart software systems can be divided into four distinct subsystems; Ground Communications, Space Communications, Earth Emulator, and Ground Emulator. Both communications are responsible for handling the emulation of two communicating bodies, ground communications being the receiver and space communications being the transceiver. The Earth Emulator has the responsibility of taking in requested astronomy data and breaking it down to its core directional units to then be sent to the simulation subsystem, then send the data to the Earth Emulator and the Rotator Emulator respectively. For both emulators, they will then take the data they receive and translate it into their respective movements on the physical hardware system.

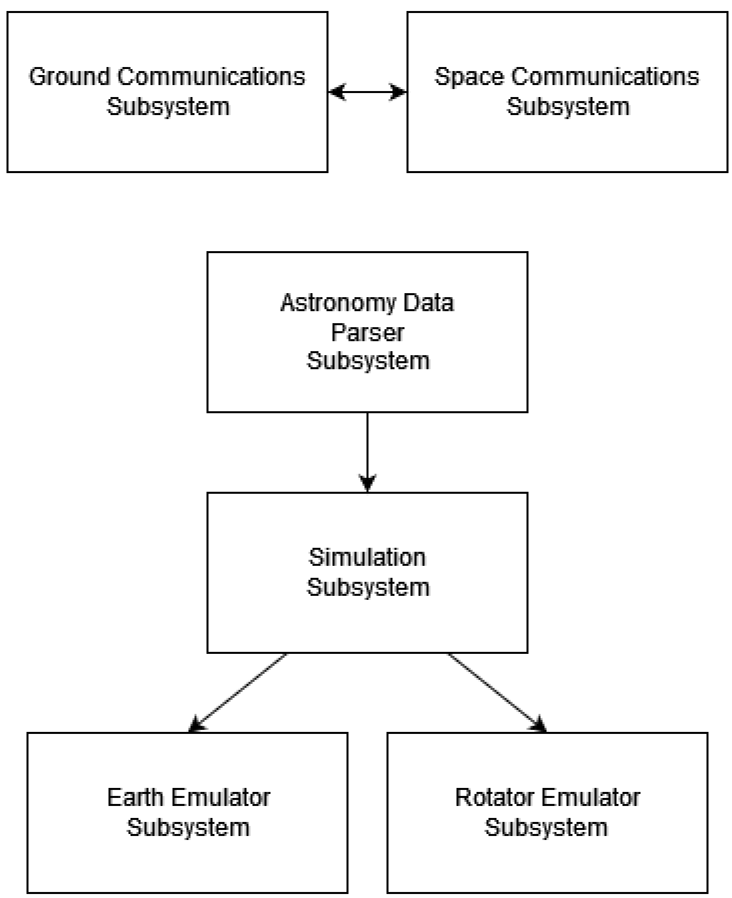

Figure 6: Detailed Software Design

4.2.1. Ground Communications Subsystem

The Ground Communications subsystem is responsible for receiving incoming data from the Space Communications subsystem. The data is then stored within a file for test engineers to inspect for accuracy during the emulation. This data can also be used to perform some other verification task if deemed necessary.

4.2.2. Space Communications Subsystem

The Space Communications subsystem is responsible for transmitting data from an emulated source, to the Ground Communications subsystem. The data it transmitted is inputted via a file transfer for a given test to then be received on the other end by the ground communications and decoded.

> **Hack RF SDR (GNU Radio Companion Transmitter):** Through the use of software blocks on the GNU companion software the specified frequency is able to be selected and transmitted from the SDR when connected to the system. It automatically detects the SDR's presence as long as its libraries are installed and are able to be activated with the click of the start button the companion.

> **RTL RF SDR (GNU Radio Companion Receiver):** Through the use of software blocks on the GNU companion software the specific frequency is able to be selected and received from the SDR. It also provides the ability for an output block to visually show Signal inputs from the antenna. It automatically detects the SDR's presence as long as its libraries are installed and are able to be activated with the click of the start button the companion.

4.2.3. Earth Emulator Subsystem

This subsystem is responsible for collecting astronomy data and converting it into altitude and azimuth angles, then sends commands to the hardware. First, the test engineer configures the simulation or inputs a file including all necessary astronomy data; it is then inspected by the parser for positional data relating to the position of the earth relative to the astronomical body; this can be either a satellite or a far-off entity. Once this data is collected, it is displayed on the user interface and begins sending instructions to the hardware. Once the data is received it is converted into a set of commands that tell the robotic arm which angles to position the joints to properly emulate the current state of the simulation.

> **Parameters:** The parameters module reads the configuration data from a file and retrieves the corresponding astronomy data. The retrieved data contains an altitude and azimuth angle, describing where the system in space is located relative to an observer on Earth.

> **Simulator:** The simulator module manages the progress of the emulation, keeping track of the current time. This is how the Earth and ground subsystems know how to position themselves. Each subsystem is

configured based on the same parameters, and once the data is obtained the only requirement to remain synchronous is that each subsystem knows what time it is.

Emulator: The emulator module is the main controller for the robotic arm. This subsystem takes the current time and determines the position of the robotic arm. Then it sends commands which will move the arm to the appropriate position.

App: The app is the user interface, which displays the altitude and azimuth relative to the time. It also shows the current time so the operator can easily understand the state of the emulation.

Sockets: The sockets module coordinates the communications between the simulator, emulator, and app. Each of these interfaces uses ZeroMQ to communicate, where the simulator acts as the server and the emulator and app act as clients, making requests to the server to get new data.

4.2.4. Rotator Emulator Subsystem

The rotator emulator subsystem takes commands from the simulation system and converts them into a series of signals to the rotator's electromechanical components.

4.3. Internal Communications Detailed Design

There are several protocols for communication used for the emStart system providing various levels of complexity for each. For communication between the robotic arm and the PC USB protocol is used; this is also used for communication between the Raspberry Pi4 and Hack RF. Furthermore, the communication between the Raspberry Pi4 and the PC uses ethernet for its communication. Lastly, the Arduino Nano and the Raspberry Pi4 use UART for their communication.

5. EXTERNAL INTERFACES

This section examines the external systems, which is any system that is not within the scope of the overall main project, that interact with the Ground Base.

5.1. Interface Architecture

The primary communication between our systems will be the analog signals sent from the System in Space, which will be used to convert astrological data into movements for the antenna attached to the Ground Station.

The Earth System shall work independently, with no communication from either the Ground Station or System in Space other than time which is used to emulate the Earth's rotation.

5.2. Interface Detailed Design

The following class diagrams show which classes use which interfaces. The earth rotators, ground station, and fixed antenna are all off the shelf hardware that the emStart system requires in order to operate.

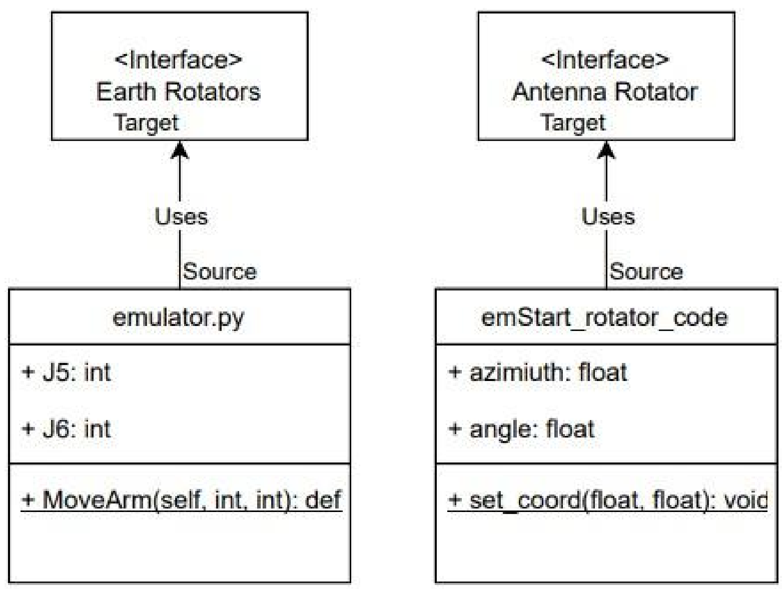

Figure 7: Class Diagram for Earth Rotators and Antenna Rotator

Besides these interfaces, the rotator antenna can receive signals sent through the satellite system through wireless transmission.

6. SYSTEM INTEGRITY CONTROLS

For the system all of the code for EmStart is stored on Github which has its own built-in safety features to ensure no vulnerabilities in the code. The system obtains the astropy data from a server that then stores it onto the Raspberry pi and computer. The arm has to be calibrated whenever it loses power to ensure that it is properly calibrated; the joints must match up with its respected notch and the wooden board has a mark on it indicating north. The base of the arm has suction cups on it that won’t suction on certain services. When running the virtual environment the system verifies that all the dependencies are properly installed.