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

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 which does not move with the Earth's rotation. The purpose of the Antenna is to capture satellite data from a satellite which will be moving with respect to the antenna. We aim to implement a robotic arm which can best match the rotation of the Earth, and therefore extend the capture time frame of data from the satellite, helping with debugging of the received signal.

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

1.2.1. System Overview

The system will comprise of three main entities. The first is the "Earth" platform which will have somewhere between 2-6 servos controlling its movement and simulating the Earth's movement.
On this platform sits the second device, our simulated receiver antenna station. This will also have a range of servos, likely to be 2, that will be what we are trying to control to make the antenna stay within range for the longest amount of time while the earth moves. This will additionally have a receiving antenna on it that will be reading radio frequency and interpreting it though a secondary device that is yet to be determined.
The final portion of this device is the simulated "satellite" which will have software defined radio (SDR) attached to it. This will simulate signals that will then be captured by the ground station. This device will be stationary in the simulation.

1.2.2. Design Constraints

The project has a few constraints. The first relates to the robotic arm. For a starting prototype, the product owner has requested we

use an off the shelf robotic arm, with a possible expansion to a custom build later. Size is another constraint, with the model needing to be generally able to fit on a table top. Hardware is a minor constraint, with three options for cheap, mid-range, and more expensive Software Defined Radios being presented to us. Finally is a 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, 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

Document Organization will be done using Github. Using Githubs issue feature, a backlog will be created and will be assigned to each sprint that is being worked on. The remaining documents will be in google drive and

1.4. Project References

The main reference will be the MIT Haystack Observatory Small Radio Telescope Project page, found here. This provides baseline information on what we are looking to build for the project.

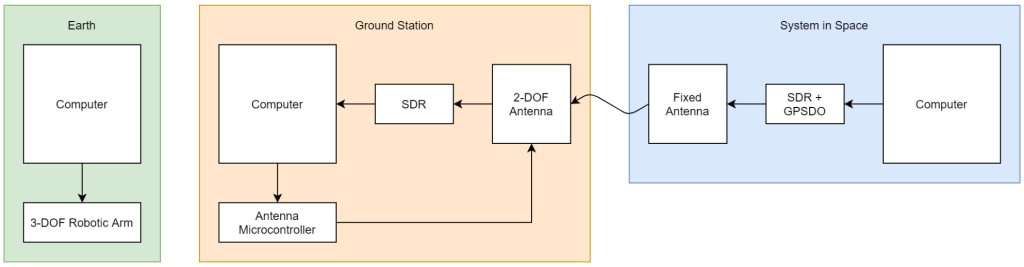
1.5. Glossary

SRT - small radio telescope
SDR - software defined radio
GPSDO - gps disciplined oscillator
DOF - degrees of freedom
SRS - System Requirements Specification

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.



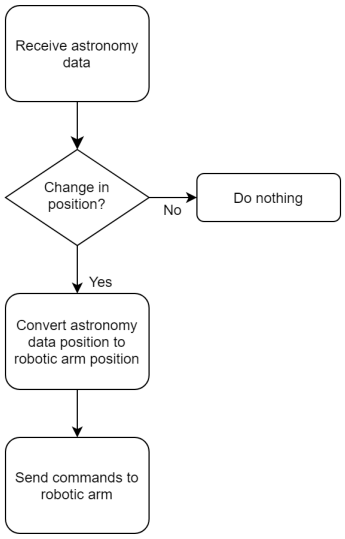
The Earth subsystem is conceptually independent from the other two subsystems and completely isolated from communicating with them. In reality, the Earth subsystem will serve as a platform for the ground station, so the two subsystems will move in unison. The Earth subsystem is intended to emulate the 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 which 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

The software in each subsystem will operate completely independent of each other. 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 a software-defined radio and control the position of the antenna. Finally, the system in space must transmit radio signals using a software-defined radio.



The Earth subsystem has a computer which 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 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.

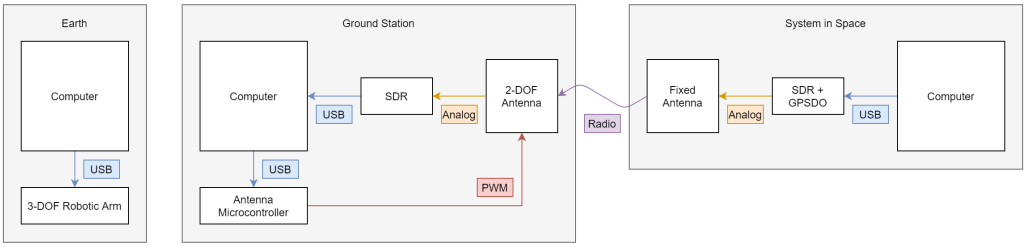
The system in space only has software for the SDR. The computer must send commands to the software-defined radio. This will be accomplished

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.



The Earth subsystem will be fully controlled by a computer. This computer will communicate with a 3-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 will also 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 which 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 to it: 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

Inputs for this project would be astronomy data for the simulated receiver antenna station. The GUI of that would be made in python using Dash. The input for the Earth would be the output of the simulated receiver antenna station and use it to move the arm into the correct position.

3.2. Outputs

Outputs for this project would be the data received from the satellite to the simulated receiver antenna station and the Earth moving to correct the position of the simulated receiver antenna station. Another output would be the rotation of the arm and rotator.

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 a myCobot 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, moving back to the Raspberry Pi 4 is also connected a hack RF 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.

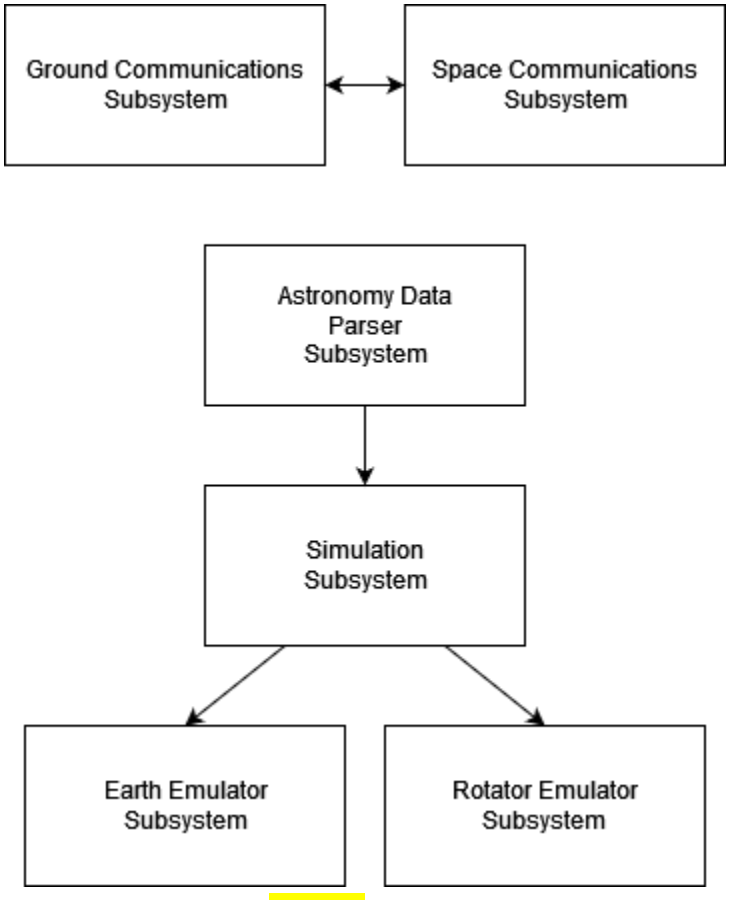
The Arduino Nano is the microcontroller used to control the rotator emulator. It does this by first receiving commands from the Raspberry Pi 4, which it then sends signals to two different servos a 180-degree servo used for moving up and down and a 360 for moving around the base of the rotator. This 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 myCobot is the robotic arm platform 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 six distinct subsystems; Ground Communications, Space Communications, Astronomy Data Parser, Simulation, Earth Emulator, and Rotator 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 Astronomy Data Parser 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. For the Simulation Subsystem, it will take imputed data from that of the Astronomy Data Parser and create commands to then send 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.



Software Architecture

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.

{Software Module Name: Followed by its description}

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.

{Software Module Name: Followed by its description}

4.2.3 Astronomy Data Parser Subsystem

The Astronomy Data Parser subsystem is responsible for taking imputed astronomy data and converting it into a form the Simulation subsystem understands. First, the test engineer 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. Second, it then passes this data to the simulation for further data processing.

{Software Module Name: Followed by its description}

4.2.4 Simulation Subsystem

The Simulation subsystem takes imputed data from the astronomy data parser and converts it into commands the Earth Emulator and Rotator Emulator understand. Once the data is received it is converted into a set of commands which relate to the position in which a specific system needs to be in for emulation.

{Software Module Name: Followed by its description}

4.2.5 Earth Emulator Subsystem

The earth emulator subsystem takes inputted commands from the simulation system and converts them into a series of signals that will be sent to the robotic arm myCobot over USB.

{Software Module Name: Followed by its description}

4.2.6 Rotator Emulator Subsystem

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

{Software Module Name: Followed by its description}

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 myCobot 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, between the Arduino Nano and the Raspberry Pi4 uses UART for its 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.

5.2. Interface Detailed Design

Need more information and work with the systems to finish this portion.

6. SYSTEM INTEGRITY CONTROLS

GitHub has built in safety features like security policy and code scanning which we may be able to use.
The network will most likely be local, not requiring an internet connection once the proper tools are installed.
We can probably validate those installs once they are done to verify that they are secure and do not pose a threat to our design.
Using GitHub revisioning protects the project using the power of open-sourcing.