System Requirements Specification

emStart: A Satellite Transceiver and Small Radio Telescope Emulator

Version 1.3 approved

Prepared by Ivan Borra, Matthew Gasper, Matthew Grabasch, TJ, Scherer, Matthew Selph

EECS Senior Design

3/6/2022

Table of Contents

Revision History

Name	Date	Changes	Version
All	9/28/21	Document creation	1.0
All	10/26/21	Document revision	1.1
All	11/30/21	Document revision	1.2
All	2/4/22	Document revision	1.3
All	3/6/22	Document revision	1.4

1. Introduction

1.1. Purpose

The goal of this project is to first create a model of a real-world SRT antenna for data verification and then to move that implementation to an already existing full-scale antenna, operating at 1.42ghz. The existing implementation of the antenna is on a fixed arm that does not move with the Earth's rotation. The purpose of the Antenna is to capture astronomy data from an object in space that will be moving with respect to the antenna. We plan on using a pan-tilt configuration for micro servos 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 attenuate data (by making the object in space signal weaker when the SRT isn't pointed at it), and a model ground station that will receive that data, and attempt to move with the expected movement of the satellite. Overall this project should establish a foundation for handling emulation of SRT operation.

1.2. Document Conventions

Requirement Hierarchy:
3.x.x - denotes an external interface requirement
- 3.1.x - denotes a user interface requirement
- 3.2.x - denotes a hardware interface requirement
- 3.3.x - denotes a software interface requirement
- 3.4.x - denotes a communication interface requirement

4.x.x - denotes a system feature requirement
- 4.1.3.x - denotes a Functional requirement of Earth Emulation System
- 4.2.3.x - denotes a Functional requirement of Rotator Emulation System
- 4.3.3.x - denotes a Functional requirement of Radio Communications System
- 4.4.3.x - denotes a Functional requirement of Mission Viewer (GUI)

1.3. Intended Audience and Reading Suggestions

This document is meant for 3 entities all listed in the SCRUM Framework, as well as the senior design faculty. These include the product owner, which in this case is Dr. Liu, the scrum master, which was selected to be Matthew Gasper, and the remaining senior design team. Additionally, the senior design faculty may use this document to track the progress of the team.

1.4. Product Scope

There are 4 pieces of software being developed/utilized for this project. The first is the transmitter software. This aims to emulate a "satellite" and send out signals to be captured by the receiver. The second is the receiver software, which aims to interpret the data being

sent to it to verify the accuracy of the tilt function that is described next. The third is the tilt function for the ground station, which has a goal of making the receiver always point in the direction of the transmitting “satellite”. The final piece of software is the earth rotation emulation which will be installed on a robotic arm to simulate the movement of the ground station on Earth. This also entails the creation of two individual 2-DOF systems which control respectively the earth system modeling earth. Along with the ground station which models the SRT and its operations.

1.5. 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/

2. Overall Description

2.1. Product Perspective

emStart will be the first in an eventual series of products that will be open-sourced for others to recreate this project themselves. The goal of the project is to create a functioning prototype that can be refined and simplified to make it accessible as a science fair project. Overall, it will be a self-contained product with multiple variants, first variant using a premade arm and the second variant using an arm we build ourselves, depending upon the users’ resources, ranging from very simple and cost-effective to more complex, more precise, but also more expensive. Providing these different tiers for the same product will make it much more accessible and adaptable to a variety of experimentation, research, development, and debugging efforts.

2.2. Product Functions

- The product must use astronomy data to determine the position of the Earth
- The product must use astronomy data to determine the position of the ground station antenna
- The product must be able to receive radio signals at the ground station
- The product must be able to send radio signals from the system in space

2.3. User Classes and Characteristics

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 pan-tilt servos. Then it sends commands which will move the servos 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.

This product will have a variety of user classes dependent upon the application. Keeping in mind the goal of open-sourcing the project and simplifying it to make it more accessible, there will be many different levels of expertise required for each tier. At the top, there will be the researchers and developers using the product as a test bench for their own designs. There will be a tier in between for those who do not require the same level of precision but still have a useful application for the design. At the lowest level, students and tinkerers will be able to use the design on a smaller scale which will not require the same levels of precision and accuracy.

Beginning at the highest level, with the most complex design, researchers and developers will use this design to its fullest. It will be used frequently to run tests on hardware and software in the user design, likely utilizing the majority of the functions the product has to offer. This application will require high precision, so technical expertise and experience will be critical to understanding the intricacies of the product to achieve the most desirable

results. More experienced users will get more use from the functions provided, while less experienced users will likely use default settings.

On the lowest level, this product should be accessible to students creating a science fair project and tinkerers looking for something to do. With this will come significantly less frequent use, less access to complex functions, and less required expertise. In this scenario, the product will serve as a proof of concept rather than a precise tool for conducting research.

2.4. Operating Environment

The hardware platform will be custom-built for this project but will consist of the following hardware: the pan-tilt servos, Raspberry Pi 4, Arduino Nano, and HackRF One. The software for each of these will use manufacturer programming methods. However, in the case of the antenna movement, an Arduino will be used to directly control the servos. The software must coexist with the astronomy data which will be streamed into the software to determine the Earth and antenna positions. It will also need to interface to the API to allow control of the pan tilt servos for the Earth.

2.5. Design and Implementation Constraints

The options available to the developers will be limited by the hardware and by the external software required to run the entire system. Hardware limitations will mostly limit the level of precision that the system is able to achieve when translating the astronomy data into a physical emulation. The software will be able to achieve much higher precision than the hardware itself due to the inherent limitations of the servos selected for the project. As far as software limitations are concerned, our software will only be as capable as the interfaces to astronomy data and the robotic arm.

2.6. User Documentation

Resource	Version
System Design Document	1.1
Raspberry Pi 4 Documentation	N/A
Elephant Robotics GitHub Repository	N/A
emStart GitHub Repository	N/A
Serial - Arduino Reference	N/A

2.7. Assumptions and Dependencies

It is assumed that we will be able to achieve within one degree of precision in the hardware, and if that assumption is incorrect that may affect our software design, as it may require more mitigating factors to prevent uncertainties. This will help ensure that the

emulator will be able to handle tests of longer duration which will make it a much more useful tool for research, development, and debugging.

3. External Interface Requirements

3.1. User Interfaces

The user shall have access to an interface that allows for basic control and the current status of different aspects of the antenna and signals being received.

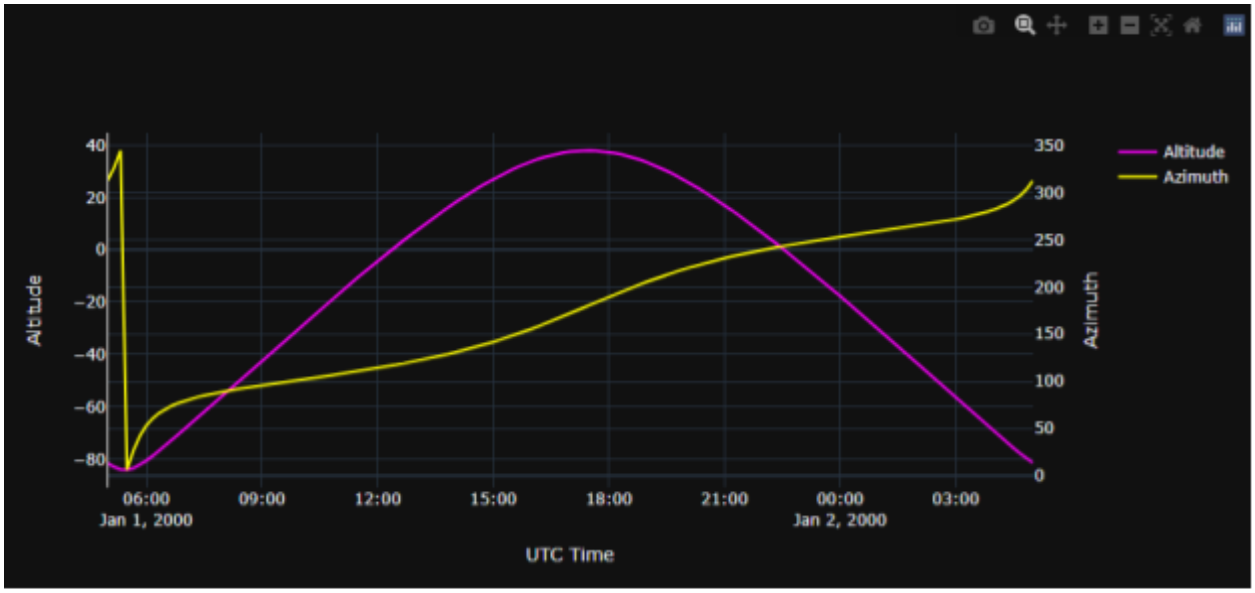

Figure 1: Plotly Graph.

3.1.1. Diagram showing current azimuth of the antenna.

3.1.2. Diagram showing current altitude of the antenna.

3.1.3. Button that allows the user to stop the movement of the antenna.

3.1.4. Button that allows manual control of the antenna for testing.

3.1.5. Diagram showing the current status of alignment for the antenna, depicting whether it is tracking the signal or not.

3.2. Hardware Interfaces

The hardware for the emulator will consist of three different base systems working in tandem, these systems being the Ground Base, Earth, and the Satellite.

3.2.1. The Earth System shall consist of a computer connected to a 2-DOF robotic arm via a USB cable.

3.2.2. The Ground Station shall consist of a computer connected to the antenna microcontroller via a USB, which is used to control the 2-DOF antenna.

3.2.3. The Ground Station shall work independently from the Earth System, meaning that the antenna must compensate for the movement of the “Earth”.

3.2.4. The System in Space shall work using a computer connected via USB to an SDR and GPSDO with a fixed antenna, sending the data towards the Ground Base.

3.2.5. The antenna shall be controlled using a microcontroller connected to a computer via USB.

3.3. Software Interfaces

The Ground station will consist of systems that move the antenna and work with the given astronomy data. This data collected will be used to convert into which position the antenna will be facing to properly receive the signal from the system in space. The system in space will only be sending signals, which will then be received by the antenna.The Earth system shall use the 2-DOF arm to move the “Earth” in different positions, thus causing the Ground Base to be utilized to keep the antenna aligned with the space object.

3.3.1. The Ground Station shall control a 2-DOF antenna through the altering of PWM signals sent to servos.

3.3.2. The microcontroller shall be given astrological data, which will be converted into operations to command the movement of the antenna and the earth emulation.

3.3.3. The “Earth” shall be fully controlled by a raspberry pi 4, working without reference to the other systems via a 2-DOF servo system connected to a computer via a USB cable.

3.4. Communications Interfaces

3.4.1. The Ground Base shall track and only receive the analog signal that is being transmitted (and not receiving) by the System In Space at 1.42ghz.

3.4.2. The Ground Base shall use the astronomical program to aid in tracking the signal by converting the data into movements for the 2-DOF Antenna.

3.4.3. The Earth System shall have no communication with the Ground Station or System in Space in order to isolate its operation.

3.4.4. The Space System shall transmit a 1.42ghz signal at 30mW to the Ground base.

4. System Features

Section 4 details a more specific list as put forward in section 2.2. This goes along with the functions associated with Earth Emulation in 4.1, Rotator Emulation in 4.2, Radio Communications in 4.3, and Mission Viewer in 4.4.

4.1. Earth Emulation

4.1.1. Description and Priority

The user shall be able to define a specified astronomical situation/path for the earth emulation to copy. The user will import a set of astronomical data that they wish to emulate, this data when imported will be used for emulation once the rotator emulation and communications of radio receiver and transceiver are also ready to execute.

4.1.2. Stimulus/Response Sequences

Once the user imports the correct astronomical data for emulation and selects “ready” the system will be on a waiting period until the other systems confirm their ready status. After emulation, the system will then return to a starting position for further emulations.

4.1.3. Functional Requirements

4.1.3.1. The system shall use a six-degree of freedom robotic arm to emulate “Earth.” via software commands.
4.1.3.2. The system shall have an attachment point to mount a flat plane to mount a rotator emulator system
4.1.3.3. The system shall import data from astropy in order to keep up-to-date astronomy data peruse of the software.
4.1.3.4. The system shall use a personal computer to control the system in its entirety.
4.1.3.5. The system shall have a mechanism in which to sync with the rotator emulation to ensure accurate emulation.
4.1.3.6. The system shall have a mechanism to ensure the calibration of the internal stepper motors.
4.1.3.7. The system shall send its current status to the GUI in order for the user to analyze the operation of the system.

4.2. Rotator Emulation

4.2.1. Description and Priority

The user shall be able to define a specified astronomical source for which the rotator will track, this information with a known earth positioning will be used to emulate a rotator’s movement when tracking.

4.2.2. Stimulus/Response Sequences

Once the user imports their specified astronomical source via initial azimuth and elevation followed by selecting “ready” the system will be on a waiting period until the other systems confirm their ready status. Once the other systems confirm their ready status emulation will begin. After emulation, the system will then return to a starting position for further emulations, while producing data for verification of the emulation.

4.2.3. Functional Requirements

4.2.3.1. The system shall have an attachment point to mount a patch antenna.
4.2.3.2. The system shall have a mounting point for a raspberry pi 4 in order for it to be then routed to an RTL SDR.
4.2.3.3. The system shall have a mounting point for an RTL SDR in order to support the system.
4.2.3.4. The system shall have a movement ability of two degrees of freedom in order to emulate the actual system.
4.2.3.5. The system shall have a mounting point in which to attach itself onto a wooden plane.
4.2.3.6. The system shall take in azimuth data to then position the azimuth servo accordingly for emulation.
4.2.3.7. The system shall take in elevation data to then position elevation positional servo according to emulation.
4.2.3.8. The system shall use an Arduino nano in order to send commands to the control servos.
4.2.3.9. The Arduino nano micro controller shall receive commands over serial from the raspberry pi 4 in order to control azimuth and elevation.
4.2.3.10. The Raspberry Pi4 shall send commands over serial to the Arduino nano based on astropy data.
4.2.3.11. The system shall take in astropy data in order to know the position of a desired celestial object to track.
4.2.3.12. The system shall take time data over WiFi in order to sync with the earth emulation station platform.

4.3. Radio Communications

4.3.1. Description and Priority

The system shall communicate between a patch antenna mounted on the rotator emulator's endpoint, from a transmitting free form radio transmitter consisting of a simple wire antenna connected to an SDR and mini computer.

4.3.2. Stimulus/Response Sequences

Before setup of the emulation GNU companion must be started and the receive and transmit files must be loaded respectively to their platforms. Once the files have been loaded a run command can be sent in order for the RF transmission and

reception to begin. Once emulation has ended the GNU companion can be stopped or left on depending on if the user wishes to continue emulation.

4.3.3. Functional Requirements

4.3.3.1. The system shall use a patch antenna in order to receive electromagnetic signals.
4.3.3.2. The system shall use a software-defined radio in order to allow for easier control over incoming frequencies.
4.3.3.3. The system shall use a personal computer to take in signals coming from a software-defined radio in order for them to be analyzed by a technician.
4.3.3.4. The system shall be mounted atop a rotator emulator in order to perform its operations.
4.3.3.5. The system shall transmit at a frequency of 910MHz in order to emulate that of an actual received frequency of a ground station.
4.3.3.6. The system shall receive a frequency of 910MHz in order to emulate that of the actual frequency transmitted from a satellite.
4.3.3.7. The system shall be controlled via GNU Companion radio in order to construct software for both the transmission and receive radios.
4.3.3.8. The system shall display received RF signals to the monitor in which it is connected in order to verify the correct operation of both the receive and transmit radios.
4.3.3.9. The system shall transmit a constant sine-wave signal in order for it to be picked up on the receiving end.
4.3.3.10. The system shall transmit sufficient power in order for the signal to be received over the distance of a wide 30ft room.

4.4. Mission Viewer (GUI)

4.4.1. Description and Priority

The Mission Viewer will be the front end of the entire system allowing for a simplified view of the entire system. It will provide the means to import astronomy data and other such information for emulation. This includes the emulation of the earth in 4.1 and of the rotator in 4.2 furthermore, a readout of radio communications will also be displayed within the mission viewer.

4.4.2. Stimulus/Response Sequences

Once the user clicks start emulation on the GUI the following occurs. During the operation of the emulation station, a readout of current operation being completed will be displayed. Along with data relating to the azimuth and elevation of the celestial body in question for emulation. This will be displayed via a time vs measurement graph.

4.4.3. Functional Requirements

4.4.3.1 The system shall visually represent the physical disposition relating to the rotator emulator.
4.4.3.2 The system shall visually represent the physical disposition relating to the earth emulator.
4.4.3.3 The system shall provide a visual for the current state of the overarching platforms attached to it being the earth and rotator emulator along with the communications system.
4.4.3.4 The system shall provide an ability to import astronomy data for it to be then sent to the earth and rotator emulator for emulation.
4.4.3.5 The system shall display graphs relating to the azimuth vs time of the system in order to verify operation.
4.4.3.6 The system shall display graphs relating to the elevation vs time of the system in order to verify operation.
4.4.3.7 The system shall be accessed via a HTML webpage on a local machine for easy programming of the GUI and future alterations.

5. Other Nonfunctional Requirements

5.1. Performance Requirements

5.1.1. The system shall be able to carry the antenna system on the arm.
5.1.2. The system shall receive data into the antenna system.
5.1.3. The system shall rotate the joints so the antenna is pointing to the satellite.
5.1.4. The antenna shall not exceed 250 grams.
5.1.5. The antenna shall act in real-time.

5.2. Safety Requirements

5.2.1. The system base should be secured to a table
5.2.2. While the system is operational the users shall maintain a safe operating range of 10 feet minimum distance from the system.

5.3. Security Requirements

5.3.1. The system shall not be left connected unsupervised to the internet.
5.3.2. The system shall be monitored while connected to the internet.
5.3.3. The system shall be set to a secure state when an error state or security threat is detected.
5.3.3 The system shall be protected through the use of a password system to ensure proper users.

5.4. Software Quality Attributes

5.4.1. The ground station software shall display the correct overlay selected by the user.

5.4.2. The system shall be able to emulate the earth's rotation.

5.5. Business Rules

5.5.1. The system shall use astronomical data from astroPy
5.5.1.1. The system shall use the astronomical data to adjust the arm

6. Other Requirements

Appendix A: Glossary

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

Appendix B: Analysis Models

N/A

Appendix C: To Be Determined List

The position of wiring for the SRT.