**Team 9 - CubeSAT**

**Sponsor:** Dr. Sklivanitis

**Date:** 11/4/2023

**Module 8 – Requirements Specifications and System Level Design**

**Functionality Requirements:**

1. Determine the frequency band
   1. Currently, we will use UHF band. Will need to explore different bands when project is ready for the space deployment.
2. Encode data at proper frequency
   1. Encoding bits onto pulses to travel over a long distance.
3. Transmit data from satellite to ground station.
4. Decode on the ground station to convert data to readable information
   1. Based upon received signal, decode pulses into usable binary bits representing information.
5. Adequately respond to noise / disruptions in the signal
   1. Develop techniques for handling noisy signals, radiation, obscurity, or unknown factors.
6. Clean the data to turn analog signal into digital correctly
   1. Error correction, digital signal processing
7. Store information in database
   1. Database will store information after processing/cleaning
8. Sufficient Data rate:
   1. Speed of transmission (bits / second); affected by frequency selection
9. Antenna Gain:
   1. Based upon the size of the antenna and the available power, we need to determine the minimum amount of gain provided for the antenna mathematically to be able to receive a clear signal. Noise could factor into this calculation.
10. Modulation Scheme:
    1. Need to select a modulation protocol suited for the task (BPSK, GMSK, QPSK)
       1. Factors for selection include data rate, bandwidth, signal fidelity, interference, and noise.

**Usability**

1. Preserve data integrity
   1. Make sure that the user has clean data and can make informed decisions based upon that data.
2. Power Requirements / Constraints
   1. 12V battery pack available; will change to solar panels when ready for launch.
3. Size Constraints
   1. 1U available, rough estimation (10x10x10cm).
4. Consistent Frequency
   1. Ensure transmissions remain in the specified band for user to interpret data successfully from the ground station.
5. Compatibility Requirements
   1. Communication protocols are in line with the receiving ground station for decoding.
6. Operating Modes
   1. State diagram of operation modes: emergency, deploy antenna, low power, low battery, data transmission (low, high, none).
7. Correct deployment
   1. Verify satellite was deployed correctly, send ‘pass’ signal via Morse Code

**Safety**

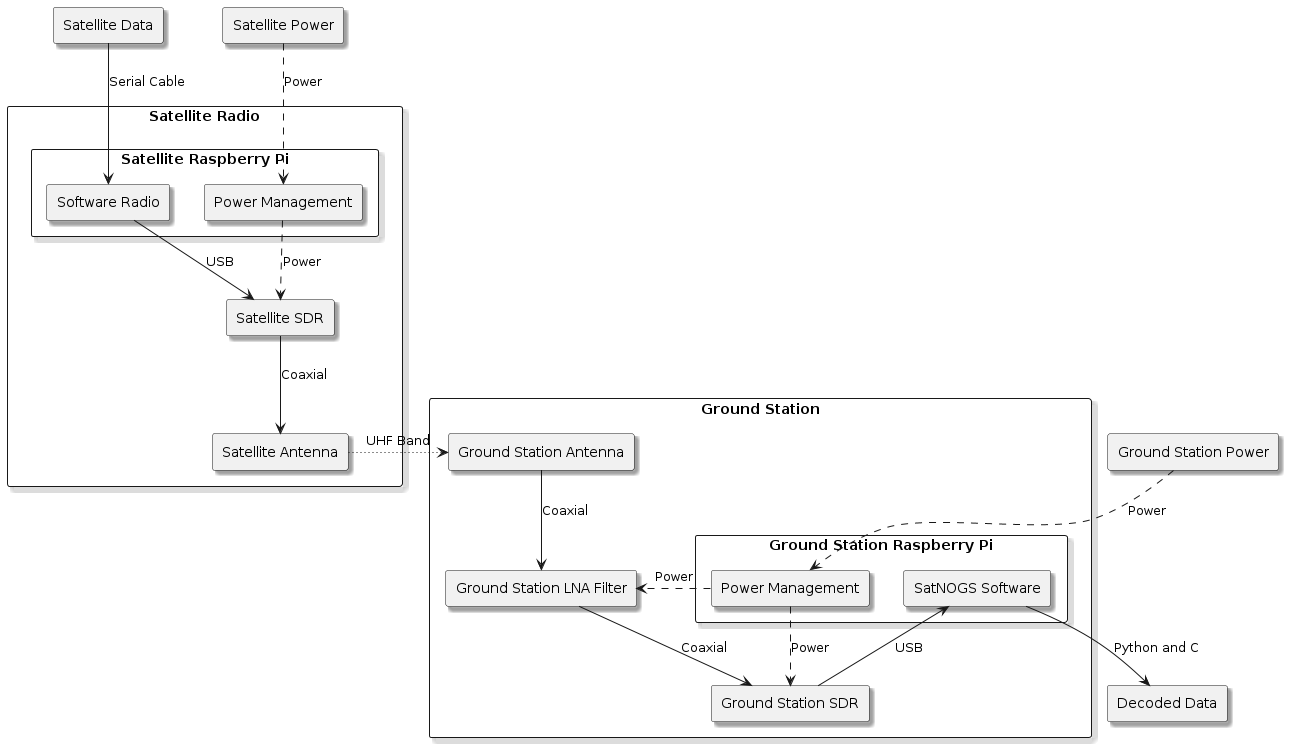
1. Emergency Shutdown
   1. If there is a power error or fault, turn the system off to ensure safety.
2. Backup Systems
   1. Develop low power alternative to only send position/coordinates and limited data if power reserves are low.
3. Ensure antenna is deployed safely
   1. Must make sure that a proper amount of time has passed before deploying antenna for safety of spacecraft / orbit.
4. Ensure ground station and satellite are operating on the same frequency
   1. If transmissions are not on the correct frequency, they cannot be received by the ground station.
5. Follow all FCC guidelines and regulations for radio transmissions.
   1. We are only able to transmit at a specific frequency. These transmissions are regulated and must follow the FCC guidelines for transmission.
6. Licensing requirements
   1. To transmit in the UHF band, an amateur HAM radio license is required.

**Block Diagrams**

**System Level Design**

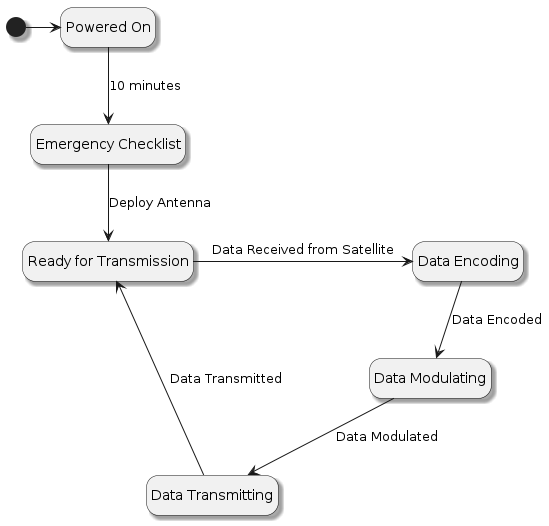
Our CubeSat system operates as a well-coordinated information relay, starting with a microcontroller that accepts data from the satellite. This collected data is then transformed into a modulated signal through our Software-Defined Radio (SDR) before being transmitted via an antenna. At the receiving end, our ground station is equipped with a specialized antenna tuned to our satellite's frequency. To ensure the purity of the incoming signal, we employ a Low-Noise Amplifier (LNA) to filter out unwanted interference. The SDR at the ground station then demodulates the received signal, making the original data accessible. This decoded information is subsequently processed by locally hosted software, ultimately restoring it to its original format. This streamlined process ensures a reliable flow of information between the satellite and ground station.

**Member(s) Responsible: All**



**Subsystems Diagrams**

**Satellite**



**Data Collection and Transmission Subsystem**

We’ve selected the Raspberry Pi as our microcontroller onboard the satellite for its adequate processing power and compatibility with our software-defined radio (SDR). The SDR allows for more flexibility and adaptability in handling various radio communication protocols and frequencies. Data will first be collected from the satellite by a serial connection from our Raspberry Pi to hardware being prepared by another team.

To ensure the data’s successful transmission, there will be a dedicated program running on the Raspberry Pi responsible for processing the incoming data. As power constraints are an obstacle, we may need to perform operations to see if the data has changed enough from the satellite’s last transmission to warrant spending the necessary power needed to send information. After these operations, the data will then be encoded. Once encoded, the data will be passed by USB connection from the Raspberry Pi to the SDR to undergo modulation and transmission.

**Team Member(s) Responsible: Morgan Benavidez**

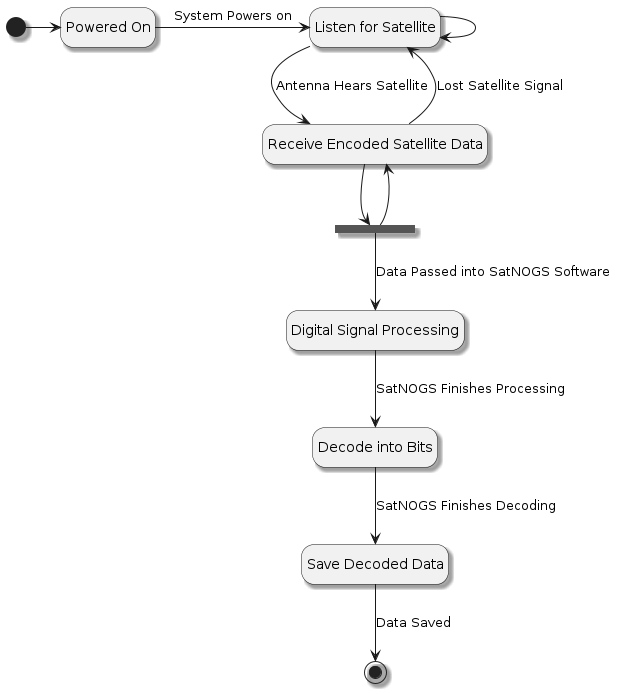
**Satellite Power Management Subsystem**

Power will be routed from the Satellite into the Raspberry Pi's main power. Then, the Raspberry Pi will switch power to the SDR when ready to broadcast. The Power Management Subsystem will monitor power drawn from the Raspberry Pi and the SDR. Each component will be powered will standard connection for that device. The Raspberry Pi will be connected to power via its USB-C port. The SDR will receive power from the Raspberry Pi via its own USB connection.

The Goal of this system is to ensure both the proper functionality of the radio equipment, and the safety of all satellite components. Functionality is ensured by activating power to the antenna when broadcasting is ready. Power to the Raspberry Pi must also remain consistent, and part of this system's role is to manage that. Safety is also a necessity. The design of this subsystem will ensure that during the operation of the Satellite Radio, no shorting or disconnects will occur. Components will also receive power at the rate and level that they are designed to support.

**Team Member(s) Responsible: Zee Fisher**

**Ground Station**



**Ground Station Subsystem:**

The ground station subsystem is responsible for precisely capturing signals from our satellite. Its antenna is carefully aimed at the satellite, enabling it to pick up radio signals accurately. To ensure data purity, the Low-Noise Amplifier (LNA) acts as a filter, eliminating unwanted signals like nearby FM radio stations and other radio traffic that isn't relevant to our mission. The filtered signal is then passed to the Software-Defined Radio (SDR), which is configured to the specific frequency we used for data transmission from the cubesat. This configuration ensures a proper match between the captured signal and our satellite's transmission. Next, the signal is transmitted to the Raspberry Pi, which hosts the SatNOGS ground station software. This software is instrumental in handling and processing the data, making it accessible and facilitating the essential communication between the ground station and the satellite.

**Team Member(s) Responsible: Carson Van Buren**

**Ground Station Raspberry Pi Subsystem:**

The Raspberry Pi subsystem for the CubeSat ground station is a critical component, designed to provide power management and integrate SatNOGS software to facilitate the ground station's various functions. At its core, the Raspberry Pi Compute Module serves as a versatile platform, offering computational power, memory, and an array of interfaces that can be tailored to the CubeSat project's specific needs. For power management, this subsystem will monitor power usage and optimization.

In terms of communication, the Raspberry Pi subsystem interfaces with the SatNOGS software, a widely used open-source solution for satellite ground station operations. This integration allows for scheduling satellite passes, telemetry data reception, and efficient management of the CubeSat's communication needs. The subsystem interfaces with the ground station's radio equipment to facilitate uplink and downlink communication with the CubeSat, while also handling data decoding and storage. It offers a user-friendly web interface for pass scheduling, satellite tracking, and real-time telemetry data visualization.

**Team Member(s) Responsible: David Miloseski**

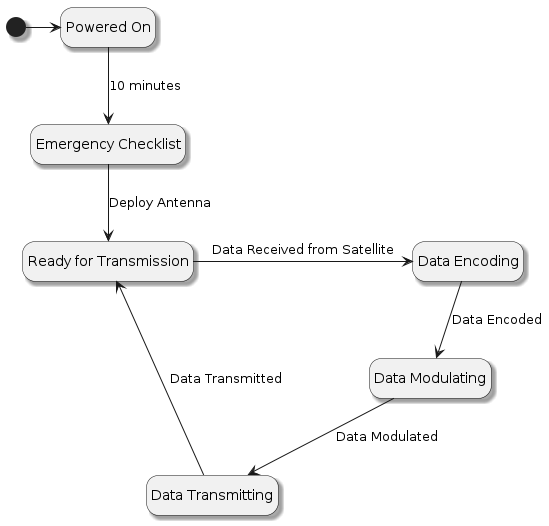
**SDR (Software Defined Radio) Subsystem:**

Software defined radio is a physical hardware component that is capable of handling analog signals inside of software instead of requiring specific analog electrical components. Modulation, demodulation, digital signal processing, filtering, and frequency selection are all tasks that would have required specific analog components. However, with SDR, all of this can be done with software. This greatly increases the reconfigurability of radio transmissions based upon the needs of the user. Internally, the SDR contains an Analog-to-Digital Converter, a sound card, a microphone, an antenna, and a general-purpose processor. Changes to the software change how the SDR can interact with signals, what type of signals, and how digital signal processing is achieved.

We will have two SDRs: one for the satellite and one for the ground station. While we will not be transmitting data from the ground station to the satellite at this point, in the future we will be able to send control signals to the satellite, if required. The satellite SDR will be responsible for modulating a signal at a specific carrier frequency that is amplified by the antenna on the satellite to be received by the ground station.

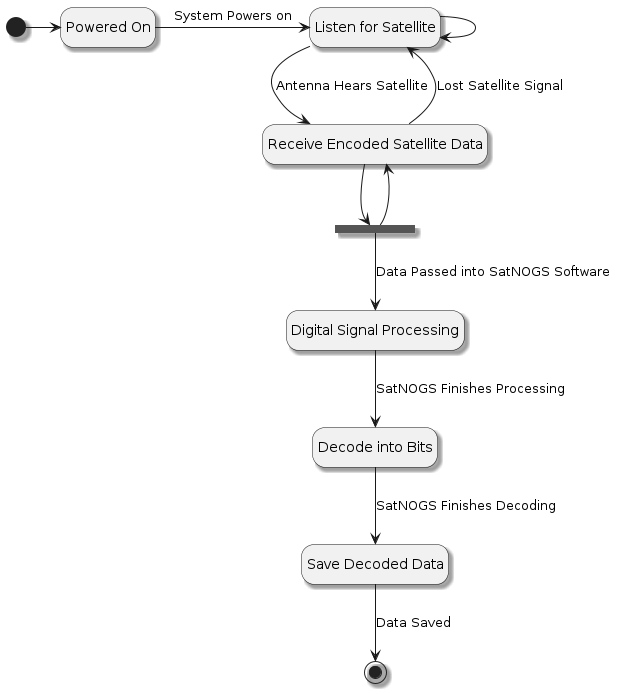
**Team Member(s) Responsible: Jonathan Mazurkiewicz**

**Satellite Subsystem State Scenario**



|  |  |
| --- | --- |
| **Transition** | **Description** |
| Powered On to Emergency Checklist | After 10 minutes of going online, the emergency checklist is started. |
| Emergency Checklist to Ready for Transmission | After the emergency checklist is completed, the antenna is deployed. |
| Read for Transmission to Data Encoding | As data is received from the satellite, the software radio will begin encoding the data. |
| Data Encoding to Data Modulating | As data is encoded, the software radio will begin modulating the data. |
| Data Modulating to Data Transmitting. | As data is modulated, it will be transmitted via the SDR and antenna. |
| Data Transmitting to Read for Transmission | After data is transmitted, the Software Radio will enter its ready state. |

**Ground Station Subsystem Scenario**



|  |  |
| --- | --- |
| **Transition** | **Description** |
| Powered On to Listen for Satellite | When the system is powered on, it will begin listening for the satellite. |
| Listen for Satellite to Listen for Satellite | The system will keep checking for the satellite if it doesn’t hear it. |
| Listen for Satellite to Receive Encoded Satellite Data | Once the system hears the satellite, it will begin retrieving encoded data. |
| Receive Encoded Satellite Data to Fork | Once data is being received, the process forks, as data can be recorded and decoded at the same time. |
| Receive Encoded Satellite Data to Listen for Satellite | When the signal is lost, the ground station will stop receiving data. |
| Fork to Receive Encoded Satellite Data | The system will continue to listen for data as it decodes what it has. |
| Fork to Digital Signal Processing | Data that needs to be decoded will be sent to the SatNOGS software for processing. |
| Digital Signal Processing to Decode into Bits | Once SatNOGS finishes processing data, it will begin decoding. |
| Decode into Bits to Save Decoded Data | Once SatNOGS finishes decoding, data is saved locally. |