Flexible Digital Radio Interface for CubeSat Communications using GNURadio

Timothy Lucas Briggs[1], Mary Knapp[2], John Swoboda[2], Ryan Volz[2], Frank Lind[2], Philip Erickson[2]

[1] REU Intern, MIT Haystack Observatory; Undergraduate, Northeastern University [2] Mentors, MIT Haystack Observatory

ABSTRACT

AERO and VISTA are twin 6U small satellite missions funded through NASA and are both being led by MIT Haystack Observatory. The main goal of the missions is to study the radio emissions from the aurora and will require the downlink of numerous types of data. As part of these missions a testable digital radio interface for CubeSat communications with a ground station has been developed. This radio interface is designed for maximum flexibility and versatility using open-source software tools. The signal chains for both transmitter and receiver allow reliable "bursty" transmission of definite-length framed packets using GFSK modulation/demodulation between two USRP B210 SDRs using GNURadio. Frame input/output utilizes ZMQ message-passing blocks such that frames can be built/parsed using more straightforward Python class implementations. Radio and framing parameters are fully configurable through YAML files. These parameters include center frequency, samples per symbol for GFSK, packet length, syncword, CRC, and whitening. A REDIS server is used to stream data to and from the radio communication interface to other portions of a larger satellite ground data pipeline. All software is designed with proper SOLID principles and object composition/inheritance patterns in mind to create a package that is as usable and extendable as possible.

MATERIALS & METHODS

Several hardware devices, software packages, and certain features of those software packages have been used to generate a solution for this project. In summary:

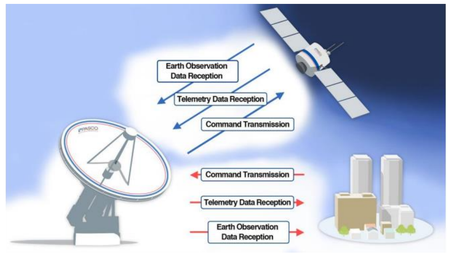
- USRP B210 Software Defined Radios (x2) provide a quickly and widely configurable radio transceiver setup for rapid software testing.
- Python scripting language (version 3.8.8) provides a dynamic and interpreted Object-Oriented language for fast development cycles, a flexible software implementation, and a collection of useful libraries.
- GNURadio package (version 3.9.1.0) provides a library of signal processing “blocks” and a “companion” program for connecting those blocks. This program generates Python code that is used to operate the SDR transceiver. Newest versions have updated signal processing blocks for digital radio communications and message passing blocks for software interfacing
- gr-satellites package (version 4.2.0) provides additional signal processing blocks to GNURadio that are particularly useful for CubeSat applications.
- REDIS Server (version 5.0.3) provides in-memory key-value databases for asynchronous local storage. Newest version implements “streams” that enable storage of time-series data that is easily ranged and polled.
- YAML and associated package py-yaml: provides an easy translation between a human-readable configuration file and usable parameters for software configuration, as well as the ability to string-ify nested key-value data.

A change was implemented in the gr-satellites project to enable runtime modification of the “packet length” parameter of the “Fixed Length Packet Tagger” block. This enables variable packet-length for the ground station uplink and downlink. The enhancement was submitted as pull-request #292 on the gr-satellites GitHub repo.

REQUIREMENTS

A proper implementation of a ground station will enable asynchronous and remote usage of a digital communication channel (using radio) between mission operations and the spacecraft.

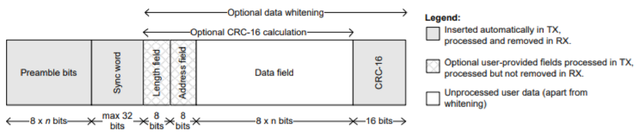
The ground station should provide an interface of the radio transceiver that will handle the reliable transmission of data over that channel by properly maintaining the half-duplex uplink and downlink. This data can be in the form of a:



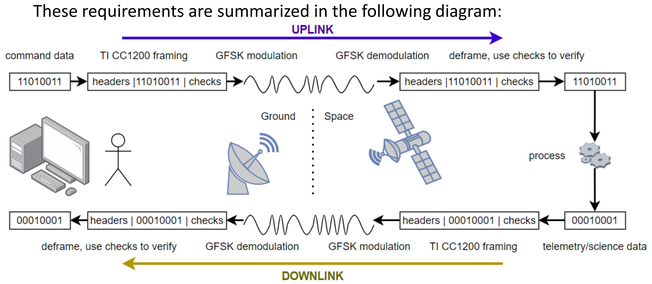
- Command: (uplink) information for the execution of an action on the spacecraft; **relatively small** packs of bytes
- Transmission Acknowledgement: (uplink/downlink) help to manage the channel by flagging success on receive; **very small** packs of bytes
- Telemetry Data: (downlink) information on the current state of the spacecraft and its subsystems; **relatively large** packs of bytes
- Scientific Data: (downlink) measurements of auroral emissions spectra and auxiliary sensors; **very large** packs of bytes

In addition, carrier wave modulation and data packet framing are required for reliable digital radio communications and defined for the AERO/VISTA project as:

- Gaussian Frequency Shift Keying (GFSK) Modulation: binary data represented in minimal shifts above/below center frequency, with transitions smoothed by a Gaussian FIR filter to limit sideband power.
- Framing as implemented by the TI CC1200 Chip Transceiver: utilizes a preamble for symbol recognition, syncword for packet detection, packet length and address fields, a Cyclical-Redundancy-Check (CRC-16) for reliability, and data whitening to avoid long sequences of 1s or 0s.



These requirements are summarized in the following diagram:



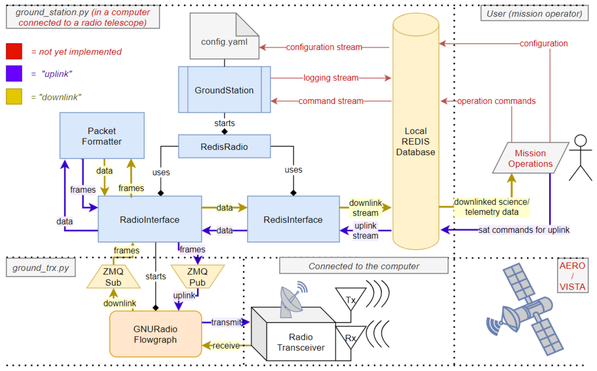
It is also desired that the ground station software package be a flexible “drop-in” solution. In other words:

- Applies to many different use cases and radio transceiver configurations
- Is easy to configure and run as a ground station operator
- Is easy to use at any time (asynchronous) from any location connected to the internet (remote) as a mission operator

These requirements come from the fact that the AERO/VISTA project will utilize several sites in several locations as ground stations. Radio telescopes in NASA’s Near-Earth Network (NEN), at Morehead State University, and at Haystack Observatory itself will be used, and each has a different set of radio setups and frequency bands it is capable of utilizing. The ground station software should fit cleanly into all these sites.

RESULTS

The current implementation of the ground station software and radio communication interface for AERO/VISTA is a flexible and extendable package that enables asynchronous queueing of commands on the station uplink and asynchronous querying of data received on the station downlink. The diagram below presents the current infrastructure for ground station configuration, operation, uplink, and downlink, as well as certain aspects of the program that are crucial but not yet implemented.

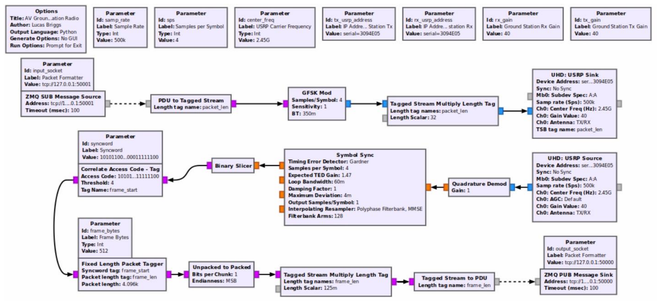


A “starts” relationship indicates that the child object is running constantly in a background process (initialized by the parent) such that iterative code can be executed within the parent. Other notable implementation details:

- Division between Python classes (blue blocks) according to **single responsibilities**. This promotes software versatility and extendibility.
- Use of a REDIS database for asynchronous queuing of operative tasks/uplink data and querying of downlink data. This will act as the link between the ground station and the larger ground data pipeline.
- Use of ZMQ Pub/Sub sockets (part of the GNURadio Messaging API) for asynchronous access to the flowgraph input and output.

The GNURadio Flowgraph implementation is presented below. The signal chain can be explained as follows:

- Messages input at the ZMQ Sub socket are converted to a tagged stream and modulated with GFSK, which will unpack each byte.
- The length tag of the stream is scaled by 8 bits-per-byte; and then scaled by the samples-per-symbol of the GFSK Mod.
- The USRP Sink block utilizes bursty transmission on the packet_len tag.
- Received signals are input to the Quadrature Demod, which outputs the phase velocity of the signal and thus demodulates FSK.
- The Symbol Sync block conducts timing-recovery on the symbol. The parameters of this block were copied from gr-satellites FSK demod.
- Symbols are then sliced into bytes, the syncword is tagged to indicate the start of a frame, then a fixed-length tag is placed at that location.
- These bytes are then packed and converted to a PDU Message that is output at the ZMQ Pub socket
- Important constants for each block are brought out as parameters for the larger Python program to configure as needed.



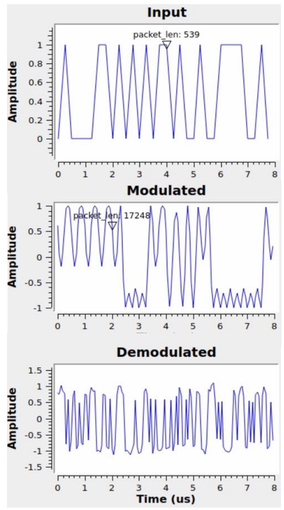
FUTURE WORK

A solid portion of the full software diagram is marked “not implemented”. Namely, configuring and operating the GroundStation can not yet be done asynchronously through the database, and the GroundStation does not log its operations, warnings, and errors. Unexpected behavior results in program-ending exceptions, when instead such errors should be logged and dealt with by the ground station so that it can continue running. Remote access to the REDIS database also needs to be implemented.

There has not yet been any tests involving a real AERO/VISTA engineering or flight model. Instead, the radio link has been demonstrated through a pair of USRP B210 SDRs directly connected with an RF cable. This can be imagined as a “bent-pipe” between the Tx and Rx antennas in the diagram above. This situation is close to ideal in terms of channel integrity, and it does not properly demonstrate the “call and response” nature of the interactions between the ground stations and satellites. Therefore, there is much future work to be done in satellite mocking, channel modeling, and signal processing.

DEMONSTRATION

These plots show QT GUI outputs from the GNURadio flowgraph during the transmission of a file through the “bent-pipe” setup that has been described. These blocks are at the input of the GFSK Mod, the output of the GFSK Mod, and the output of the Symbol Sync. Sampling rates and numbers of points are not properly aligned, so these plots do not show a direct relationship between the displayed signals. They do, however, show signals that make sense for raw binary that is then modulated and then demodulated with GFSK. Packet-length tags are also shown.



A full video demonstration of file transfer of a “.png” image using this software is available on YouTube at the following link.
https://youtu.be/5SMxow1ln5Q

REFERENCES

Lind, Frank D., et al. USU Digital Commons, 2019, *AERO & VISTA: Demonstrating HF Radio Interferometry with Vector Sensors*. https://digitalcommons.usu.edu/smallsat/2019/all2019/96/

“Frequency-Shift Keying.” *Wikipedia*, Wikimedia Foundation, 28 July 2021, https://en.wikipedia.org/wiki/Frequency-shift_keying

“Cc1200.” *CC1200 User Guide*, | *TI.com*, https://www.ti.com/lit/ug/swru346b/swru346b.pdf?ts=1629316684524&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FCC1200.

“PASCO Provides Rental Service for Satellite Ground Station Facilities.” *GIM International*, 19 May 2021, www.gim-international.com/content/news/pasco-provides-rental-service-for-satellite-ground-station-facilities.

Estévez, Daniel. “Daniestevez/Gr-Satellites: GNU Radio Decoders for Several Amateur Satellites.” *GitHub*, 2021, github.com/daniestevez/gr-satellites. https://github.com/daniestevez/gr-satellites

Briggs, Timothy. *Haystack Demo. YouTube*, 11 Aug. 2021, youtu.be/5SMxow1ln5Q. https://youtu.be/5SMxow1ln5Q

CONTACT

Timothy Lucas Briggs, Electrical and Computer Engineering
Northeastern University
briggs.tim@northeastern.edu
415-300-7760