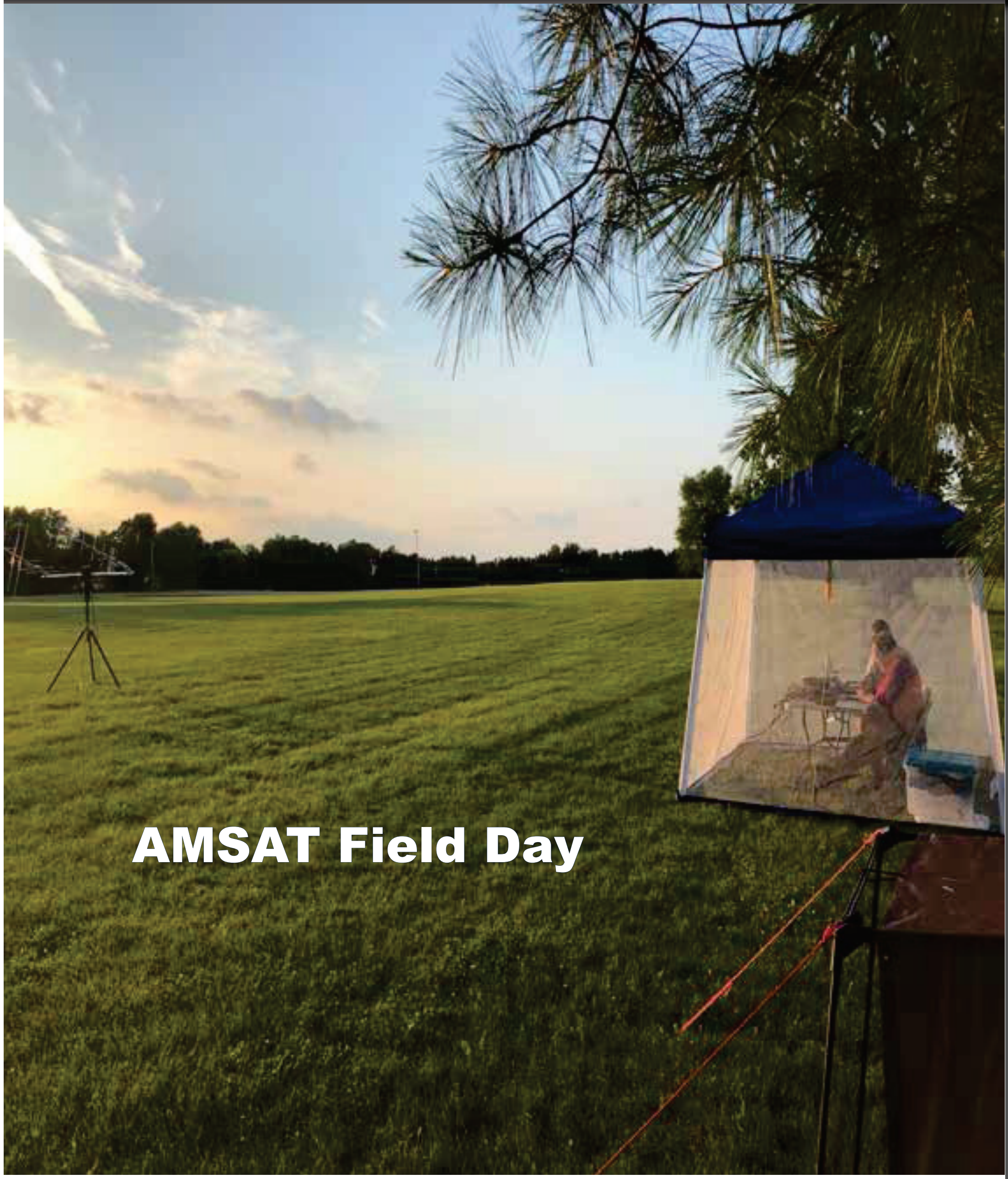
The AMSAT® Journal

Editor-in-Chief
Joe Kornowski, KB6IGK

Assistant Editors
Bernhard Jatzeck, VA6BMJ
Douglas Quagliana, KA2UPW/5
Paul Graveline, K1YUB

Volume 42, Number 4 July/August 2019



AMSAT Field Day

in this issue ...

Periodicals
POSTAGE PAID
At Kensington, MD
and at additional
mailing offices

AMSAT
10605 Concord St., Suite 304
Kensington, MD 20895-2526

The New AMSAT CubeSat Simulator:

Part 4, The Ground Station

Alan B. Johnston, KU2Y
Vice President Educational Relations, AMSAT
Associate Teaching Professor of Electrical and Computer Engineering,
Villanova University
Philadelphia, PA, USA
ku2y@amsat.org

Pat Kilroy, N8PK
Flight Systems Integration & Test (I&T) Engineer,
NASA Goddard Space Flight Center
Greenbelt, MD, USA
n8pk@amsat.org

***Abstract*—The AMSAT CubeSat Simulator is a Raspberry Pi-based, 3D printed functional model of a CubeSat satellite that transmits current, voltage, and temperature telemetry on the UHF ham radio band. This article describes options for a ground station to receive and decode housekeeping telemetry from the CubeSat Simulator.**

***Keywords*—cubesat, simulator, telemetry, AMSAT, ham radio, raspberry pi, sdr, rtl-sdr**

I. Introduction

We feel that, to the general public, the key concepts of a satellite ground station are akin to the line in an old movie, "Pay no attention to that man behind the curtain!" Not so for CubeSat developers! The ground station is essential in any space mission, as it is the only way to communicate with and command one's satellite after launch. However, we are surprised by the number of CubeSat teams who do not have a properly functioning ground station by the time their satellite is deployed on orbit. Just as we believe building a CubeSat Simulator is an important step towards developing a successful CubeSat flight model for a launch, we believe that setting up and using a CubeSat Simulator Ground Station is a vital step-wise progression towards being able to communicate with one's CubeSat after launch.

In Huntsville, Alabama, in November 2018, we introduced a proof-of-concept prototype model of the new AMSAT® CubeSat Simulator as a tool for satellite technology education and demonstrations. We initiated a Beta Builder & Beta Tester invitation to AMSAT members. Shortly after, we officially "launched" the core of an encompassing STEM program at Hamvention 2019 in Dayton, Ohio during the AMSAT Update forum. Of our four complete, working CubeSat Simulator models that debuted at the Hamvention, we loaned each of them to AMSAT members to take home to demonstrate at a high school, club meeting or public function. The results so far have been highly successful and quite encouraging.

This article is the fourth in a series introducing the Simulator. The first article explained the philosophy and design of the Simulator (AMSAT Journal, Part 1, Nov/Dec 2018). The second showed some educational activities that can be performed with the Simulator, starting with activities designed for the original ARRL ETP CubeSat Simulator, as described by Mark Spencer, WA8SME, (AMSAT Journal, Part 2, Jan/Feb 2019). The third article described some interesting failure simulations, efficiency and maximum power point calculations, and using an Arduino platform as a payload interface for the Simulator (AMSAT Journal, Part 3, May/June 2019).

This article introduces the CubeSat Simulator Ground Station and discusses the various options and tradeoffs in station design. It includes a choice between a Windows PC Ground Station and a Raspberry Pi Ground Station.

II. Background

The new AMSAT CubeSat Simulator, shown in Figure 1, features a Raspberry Pi Zero W-based multi-board stack and a 3D-printed frame structure.

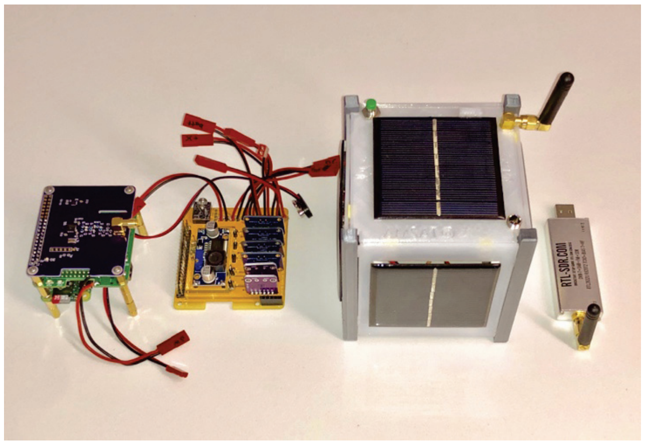


Fig. 1. An AMSAT CubeSat Simulator showing the components. From left to right: three board stack including Brandenburg Digital Transceiver Board, MoPower UPS V2; Raspberry Pi Zero; Custom AMSAT Solar Power Board with Current and Voltage Sensor daughter boards mounted vertically; 3D Printed CubeSat frame with solar panels; and RTL-SDR USB dongle with antenna for the Ground Station.

These serve as a functional CubeSat model in a 1U form factor, less the ground station component. It is designed as closely as possible to act as a standard 10 cm (4 inch) cube-sized satellite flying in Low Earth Orbit (LEO). The purpose of this system is to demystify to all how satellites work. Like typical LEO satellites, this simulator runs on rechargeable battery power and solar cell panels. Our model currently transmits telemetry on the UHF band using the AMSAT OSCAR 7 (AO-7) format using AFSK AX.25 1200 bps modulation. For details on the design and construction of the simulator, see our paper in the 2018 AMSAT Annual Meeting & Space Symposium proceedings [1] or as updated and edited for the Nov/Dec 2018 issue of the AMSAT Journal.

The remainder of this article describes a Ground Station for the CubeSat Simulator.

III. Ground Station Requirements

The main requirement of the CubeSat Simulator Ground Station is to receive telemetry transmitted by the CubeSat Simulator, and enable the decoding and display of the information contained in the telemetry. Telemetry, of course, includes the satellite's health and welfare or "housekeeping" data downlinked, as well as the science data from each payload or "instrument" on the spacecraft bus. For simplicity, most of our telemetry discussions tend to be of the common housekeeping variety because each instrument is so unique. Other requirements include that the Ground Station be reasonably low in cost, easy to setup and configure, and be similar to a real satellite ground station.

A student recently asked innocently enough, and quite insightfully, "Why not just have the satellite send down its data directly without all the complexity of modulating, demodulating, encoding and decoding"? Well, we wish the current state of the art in technology would permit that to happen. It is not, and for a number of reasons. So, such an investigation is left for the student to explore, and perhaps one day to develop a simpler solution for humankind!

The functional blocks of a Ground Station are shown in Figure 2. They are: Antenna electrical and mechanical subsystem, Feed Lines, Low Noise Amplifier, Receiver, and Decoder. We will discuss each in terms of the requirements for the CubeSat Simulator Ground Station.

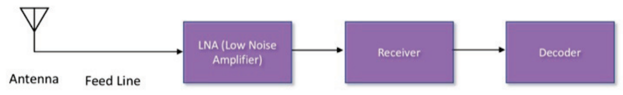


Fig. 2. Functional Blocks in a Satellite Ground Station.

In an orbital satellite ground station, the antenna is a critical and complex part of the station. However, since the CubeSat Simulator is usually in the same room as its Ground Station, the antenna can be very simple, even a short piece of wire. And since the antenna can be co-located with the Ground Station, the feedline for the Simulator Ground Station can just be a connector. Again, due to the simplicity, a Low Noise Amplifier is not needed for the Simulator.

The receiver for the Ground Station also has simple requirements. Any UHF FM receiver or scanner would work, as would a Software Defined Radio (SDR) receiver USB dongle, plugged into a laptop or PC or Raspberry Pi. The following sections will discuss the details of this.

The Decoder takes the output of the receiver and translates it to display and analyze the satellite telemetry. In addition, the output from the Receiver might also be speakers or headphones, or an audio recording device.

NASA Advice Given to CubeSat Developers

Planning to build a CubeSat flight model? Here are a precious few professional ("Been there, done that, failed a few times ourselves!") advice snippets to consider before holding your Preliminary Design Review (PDR). Can you incorporate any of these tips into the AMSAT CubeSat Simulator project that you are about to get underway?

1. "Test early and test often." Very important! Cannot be over emphasized. Fight diligently for the required schedule or budget. Start from your component and subsystem level units at your incoming inspection upon procurement. Verify the operation of each part or unit, comparing to the related spec sheet. Contact the vendor about any discrepancy.

2. "Test as you fly." That is, activate your ground station and converse with your satellite while it sits on a lab bench only a meter or two away. Celebrate its responses. This is called a Compatibility Test. Perform DITL ("Day In The Life") tests and deployment tests in the configuration as close as possible that you expect the satellite to experience on orbit, all on an open lab bench. Accumulate hours of operation and log them. Good job! Then do the same under stresses, such as thermal, mechanical and RF emission environments. Have any CubeSat developers cut way back on proper testing for any number of reasons? Unfortunately, those are the ones who also have issues and challenges once on orbit and wonder why. Hard facts. Tough love?

3. "Analyze closely your housekeeping telemetry and look for trends in your data." Learn all the characteristics of your satellite almost as though it were a new born baby growing to a toddler through your telemetry data. Watch for good trends. Watch for not-so-good trends and learn to perform troubleshooting on the ground, as opposed to in low earth orbit.

4. "Develop a properly functioning ground station." Don't rely solely on a distant, outside source or group. Otherwise, hold a sufficient number of comprehensive Compatibility Tests with your remote ground station or stations, or network. One's ground station or network seems to be too often a late oversight. Don't let it happen to you. Plan early.

IV. Radio Receiver Options

To receive the AFSK 1200 bps telemetry, almost any UHF FM transceiver or Handie Talkie (HT) will work, including the ultra low cost Baofeng HTs which sell for as little as $20. An audio cable is needed to input the signal to a computer for telemetry analysis. This cable transfers the line-level headphone audio output from the radio to an audio input on the computer. If the computer does not have a line-level audio input (but only a low-level microphone input) then adjustments with the HT volume knob or PC mixer controls might be required to avoid a distorted signal. Otherwise a small series resistor might serve well as an attenuator if needed.

V. SDR Receiver Options

Since the intent of the ground station is to analyze the telemetry, using an SDR receiver dongle is an obvious choice. There are a wide range of SDR USB dongles available today. We use the cheapest RTL-SDR dongle which sells for about $20. The "RTL" in RTL-SDR refers to the RTL2832U chip which functions as a wideband radio tuner. It can be plugged into a Windows computer or Linux computer, such as a Raspberry Pi. We will discuss the pros and cons of each computer configuration in the following sections. The next section discusses the steps involved in demodulating and decoding of the signal.

VI. Demodulation and Decoding

The current version of the AMSAT CubeSat Simulator uses AFSK AX.25 1200 bps encoding of AO-7 formatted telemetry. This was chosen as it is simple and common and easy to decode. It is compatible with APRS (Automatic Packet Reporting System) decoders. Future software updates may also support other modulation schemes, such as the DUV FSK (Data Under Voice Frequency Shift Keying) of the AMSAT Fox-1A, 1B, 1C, and 1D satellites or the BPSK (Binary Phase Shift Keying) of the Fox-1E satellite.

To understand the requirements of the Decoder and Demodulator, let's break down what "AFSK AX.25 1200 bps" means. Note this is also sometimes written as AFSK AX.25 1k2. We will start with the radio signals received from the CubeSat Simulator on the UHF band. The block diagram of this process is shown in Figure 3.

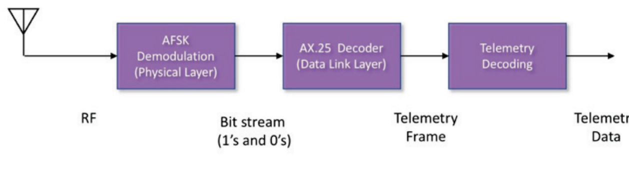


Fig. 3. Block Diagram of Demodulator and Decoder

The radio signal sent by the CubeSat Simulator is an FM modulated digital signal. A normal narrow-band FM demodulator is first used, at the carrier frequency of the RF signal. The resulting audio after demodulation is a series of tones which represent the stream of 1's and 0's -- the binary data. A 1200 Hz tone represents a 1 (a "mark") and a 2200 Hz tone represents a 0 (a "space"). The data rate is 1200 bits per second, so the tone can change up to 1200 times each second. For each period of 0.833 ms, the tone is detected in order to determine if a 1 or a 0 is being sent. This is how demodulation of an AFSK (Audio Frequency Shift Keying) signal is performed. The output from an AFSK demodulator is a stream of 1's and 0's.

Next, the AX.25 Decoder will take this stream of 1's and 0's. The AX.25 decoder removes the AX.25 header, which is at the start of the frame. This contains fields such as the Source and Destination call signs, etc. The rest of the frame is the string of telemetry data. For more information on the complete satellite telemetry decoding protocol stack, see this excellent summary of these and other protocols such as HDLC and KISS that are used in satellite telemetry decoding by Daniel Estevez, EA4GPZ / M0HXM [2].

This telemetry data represents encoded values of measured voltages, currents, and temperatures on the CubeSat Simulator. Each of these values is a coded number. We chose to encode them in a format in which AO-7 and later AMSAT satellites -- including AO-8 which Alan KU2Y worked back in the 1980s -- encoded this information suitable for transmission over CW. This is the memorable "hi hi 156 199 …" format where "hi hi" indicates the start of a packet frame of information, which is then followed by sets of three-digit numbers separated by a space. The first digit represents the channel number, and the next two digits are the encoded telemetry value. For example, the solar panel electrical current channels are encoded by this formula:

$$Current = 10 * (Value - 99)$$

where *Value* is a two-digit number, and *Current* is the current in mA. Previous articles discuss interpreting these data.

VII. Ground Station Manual Implementation

The steps in manually demodulating and decoding telemetry from the CubeSat Simulator are shown in Figure 4. It begins with an RF (Radio Frequency) signal from an antenna on the left and ends with graphs displayed of telemetry data on the right. A radio dongle produces a stream of IQ (In-phase and Quadrature) data centered about the carrier frequency of the CubeSat Simulator telemetry, around 440 MHz. An SDR receiver application is then used to perform the FM Demodulation. A Virtual Audio Cable (VAC) loops back the playback audio to the recording microphone input for the AX.25 Decoder which outputs the telemetry data. The Display function processes the mathematical formulas that convert the coded data into the originally measured voltages, currents and temperatures, which are then graphed.

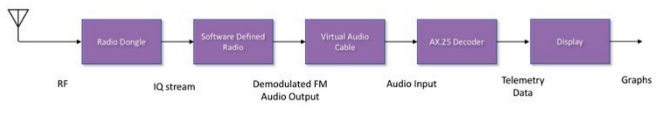


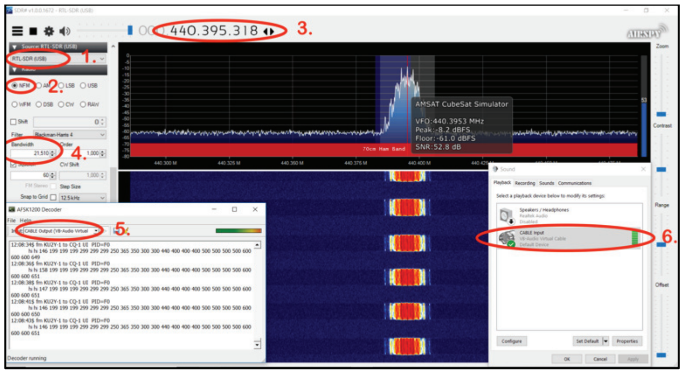
Fig. 4. Ground Station Implementation of Demodulation and Decoding of Telemetry.

For Windows, the applications corresponding to this are as follows: an RTL-SDR receiver dongle, the SDR# Software Defined Radio package, the VBCable (which provides a Virtual Audio Cable from the Sound Playback in Windows to the Sound Recording) and the Qtmm AFSK 1200 Decoder. Then the user performs a manual cut-and-paste of the telemetry data into a spreadsheet for calculation and graphing. The detailed process of how to do this in Windows is described in our project Wiki [3].

A screenshot showing all these applications working is in Figure 5. You can see the key settings circled in red and labeled with a number:

1. SDR# Source set to RTL-SDR (USB)
2. SDR# Radio set to Narrow Band FM (NFM)
3. SDR# Frequency set to 440 MHz, centered about the telemetry signal
4. SDR# Bandwidth set to cover entire signal (about 20 kHz)
5. Windows Sound Playback set to CABLE Input VB-Audio Virtual Cable
6. AFSK1200 Decoder Input set to CABLE Output (VB-Audio Virtual Cable)

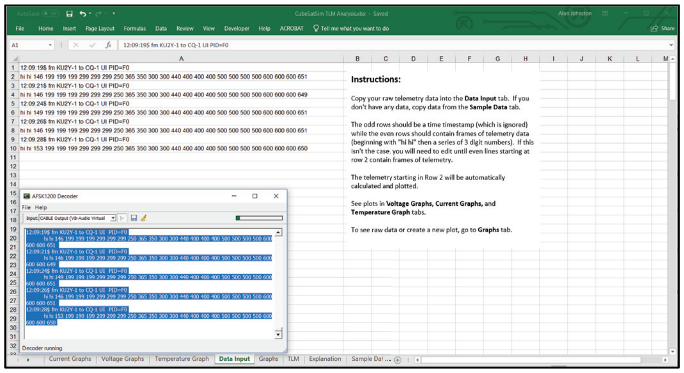
And note that when the signal is received, the Sound Playback CABLE Input, Sound Recording CABLE Output, and AFSK1200 Decoder all show full scale (green bars).



```
12:08:34$ fm KU2Y-1 to CQ-1 UI  PID=F0
        hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600
600 600 649
12:08:36$ fm KU2Y-1 to CQ-1 UI  PID=F0
        hi hi 158 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600
600 600 651
12:08:38$ fm KU2Y-1 to CQ-1 UI  PID=F0
        hi hi 147 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600
600 600 651
12:08:41$ fm KU2Y-1 to CQ-1 UI  PID=F0
        hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600
600 600 650
12:08:43$ fm KU2Y-1 to CQ-1 UI  PID=F0
        hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600
600 600 651
```

Fig. 5. Windows Screenshot Showing Key Settings in Applications.

The cut-and-paste between the Qtmm AFSK 1200 Decoder application and the spreadsheet is shown in Figure 6, where the telemetry is pasted into the Data Input tab of the spreadsheet. The telemetry graphs are then displayed in the other tabs of the spreadsheet.


CubeSatSim TLM Analysis.xlsx - Saved
File Home Insert Page Layout Formulas Data Review View Developer Help ACROBAT Tell me what you want to do
A1
12:09:19$ fm KU2Y-1 to CQ-1 UI PID=F0
AFSK1200 Decoder
File Help
Input: CABLE Output (VB-Audio Virtual
Decoder running
Current Graphs
Voltage Graphs
Temperature Graph
Data Input
Graphs
TLM
Explanation
Sample Dat ...

```
1  12:09:19$ fm KU2Y-1 to CQ-1 UI PID=F0
2  hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600 600 600 651
3  12:09:21$ fm KU2Y-1 to CQ-1 UI PID=F0
4  hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600 600 600 649
5  12:09:24$ fm KU2Y-1 to CQ-1 UI PID=F0
6  hi hi 149 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600 600 600 651
7  12:09:26$ fm KU2Y-1 to CQ-1 UI PID=F0
8  hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600 600 600 651
9  12:09:28$ fm KU2Y-1 to CQ-1 UI PID=F0
10 hi hi 153 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600 600 600 650
```

Instructions:

Copy your raw telemetry data into the **Data Input** tab. If you don't have any data, copy data from the **Sample Data** tab.

The odd rows should be a time timestamp (which is ignored) while the even rows should contain frames of telemetry data (beginning with "hi hi" then a series of 3 digit numbers). If this isn't the case, you will need to edit until even lines starting at row 2 contain frames of telemetry.

The telemetry starting in Row 2 will be automatically calculated and plotted.

See plots in **Voltage Graphs, Current Graphs,** and **Temperature Graph** tabs.

To see raw data or create a new plot, go to **Graphs** tab.

```
12:09:19$ fm KU2Y-1 to CQ-1 UI  PID=F0
        hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600
600 600 651
12:09:21$ fm KU2Y-1 to CQ-1 UI  PID=F0
        hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600
600 600 649
12:09:24$ fm KU2Y-1 to CQ-1 UI  PID=F0
        hi hi 149 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600
600 600 651
12:09:26$ fm KU2Y-1 to CQ-1 UI  PID=F0
        hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600
600 600 651
12:09:28$ fm KU2Y-1 to CQ-1 UI  PID=F0
        hi hi 153 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600
600 600 650
```

Fig. 6. Windows Screenshot Showing Cut-and-Paste from Decoder to Spreadsheet.

We realize some steps in our set up procedures are more challenging than others, but please bear with us. We will provide support and encouragement along the way!

Our experience with this Ground Station setup with Windows has shown the main benefit to be low cost, with only an RTL-SDR dongle and a simple antenna needed, as most people have easy access to a Windows computer. However, it has not proven to be easy to set up. Here are the main issues we have encountered:

1. Difficulties installing the RTL-SDR drivers. The RTL832U chip was designed for Digital TV using the DVB-T standard used in Europe. The use of this chip as an SDR is essentially a hack, and to make it work, the official drivers that ship with Windows and Linux must be replaced. This is a simple command line in Linux, but on Windows it is a procedure involving running a piece of software known as Zadig. You can see the number of steps involved here [4]. Also, this installation requires administrator privileges. Some teachers do not have these rights on school-issued laptops and computers, so this can be annoying. Also, occasionally, the driver installation simply fails for unknown reasons, which is a frustrating experience.
2. Configuring VBCable is a bit complex. Differences between Windows computers means that a set of instructions can't always be followed exactly. Different names and devices also complicates the configuration. In the Wiki instructions [3] we show eight different screen shots of the Windows Sound control panel to try to explain the steps, showing how complex it is.
3. Tuning SDR# can also be a source of problems. If the tuning isn't centered about the telemetry signal, or the bandwidth not set wide enough, decoding may fail even though there is a signal. However, this issue at least has educational value, in that it teaches the importance of correct receiver tuning, a useful skill.

Issue #1 can be eliminated by using an SDR dongle which does not require driver installation, such as the FUNcube USB dongle. However, the FUNcube dongle is much more expensive and does not have the wide software support that the RTL-SDR has. There is a no-install RTL-SDR approach that works for Windows, Linux, and even Android and iOS. It is described in the Appendix to this article.

We have not yet found a simple solution to Issue #2 in this configuration. One option might be making a physical loopback plug that connects the speaker to the microphone through the headset jack on a PC. The best solution would seem to be to avoid generating audio that must be ported internally.

Issue #3 can be solved by providing an XML file with the CubeSat Simulator frequency, bandwidth, and modulation pre-configured as a frequency pre-set. Just clicking on the name in the Frequency Manager in SDR# will correctly tune the signal.

These, and other Windows alternatives for the various applications are listed in Table 1.

TABLE I. WINDOWS APPLICATION OPTIONS

Function	Option	Pros	Cons
Radio	RTL-SDR	Cheapest SDR	Installation of Windows drivers is tricky
Radio	FUNcube Dongle	No driver install, high quality SDR	More expensive
SDR	SDR#	Free, easy to use, many plugins	Windows only
SDR	HDSDR	Free, used by many satellite operators	Windows only, complex user interface
VAC	VBCable	Works	Windows Sound configuration not easy
VAC	Virtual Audio Cable	Works	Not free, Sound configuration not easy
Decoder	Qtmm AFSK 1200 Decoder	Simple User Interface	
Decoder	UZ7HO SoundModem	Used by many satellite operators	Complex User Interface
Display	MS Excel Spreadsheet	Commonly used	Not free
Display	LibreOffice Calc Spreadsheet	All platforms, Free	Functions slightly different from MS Excel
Display	Google Docs Spreadsheet	Easy to share, Free	Web based

Note that if running on Linux, the same flow can be used, although different applications will be used. For example, instead of SDR#, Gqrx could be used, and instead of VBCable, a Virtual_Sink can be configured with pulseaudio. However, we suspect few teachers have access to a Linux desktop. A bootable USB with Ubuntu could be used to allow any laptop to run Linux, but this is a bit advanced. Soundflower provides Virtual Audio Cable functionality for Mac OS.

VIII. GROUND STATION AUTOMATIC IMPLEMENTATION

A recent CubeSat Simulator Ground Station approach that we have come up with is an automatic approach. It simplifies the process greatly. It uses two command line applications: rtl_fm and multimon-ng. The first, rtl_fm is a simple demodulator application that comes with the RTL-SDR drivers, and runs on Windows or Linux. Settings such as center frequency, bandwidth, and demodulation are set using command line parameters and options, and the output can be sent to an audio driver, a file, or 'piped' to another command line application. The other, multimon-ng is a command line demodulator for a variety of digital encodings, including AFSK AX.25 1200. The input signal can come from another program, and the output is the decoded text.

This single command both demodulates and decodes the CubeSat Simulator telemetry:

```
rtl_fm -f 440.386M -s 22050 -g 48 - | multimon-ng -a AFSK1200 -A -t raw -
```

The -f option sets the frequency, while the -s option sets the bandwidth. The | is the 'pipe' command which connects the output from the first program rtl_fm to the input to the next program multimon-ng. This completely avoids the audio porting issue of the previous approach. The -a option sets AFSK AX.25 1200 for decoding.

This one command then replaces the SDR, Virtual Audio Cable, and AFSK 1200 Decoder blocks in Figure 4, resulting in the blocks shown in Figure 7.

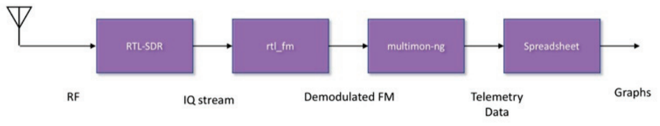
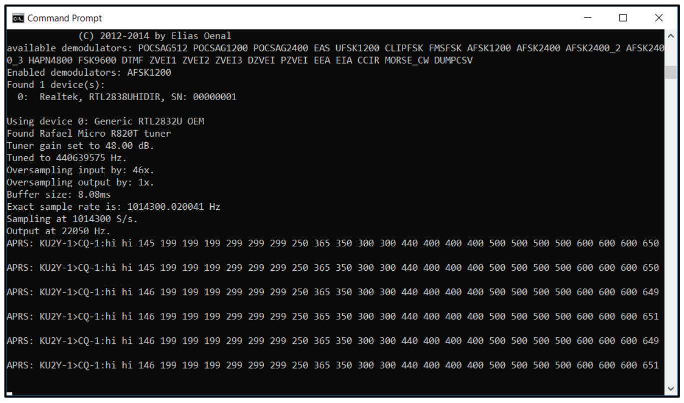


Fig. 7. Ground Station Implementation of Automatic Decoding of Telemetry.

An example output is shown in Figure 8.



```
           (C) 2012-2014 by Elias Oenal
available demodulators: POCSAG512 POCSAG1200 POCSAG2400 EAS UFSK1200 CLIPFSK FMSFSK AFSK1200 AFSK2400 AFSK2400_2 AFSK240
0_3 HAPN4800 FSK9600 DTMF ZVEI1 ZVEI2 ZVEI3 DZVEI PZVEI EEA EIA CCIR MORSE_CW DUMPCSV
Enabled demodulators: AFSK1200
Found 1 device(s):
  0:  Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Tuner gain set to 48.00 dB.
Tuned to 440639575 Hz.
Oversampling input by: 46x.
Oversampling output by: 1x.
Buffer size: 8.08ms
Exact sample rate is: 1014300.020041 Hz
Sampling at 1014300 S/s.
Output at 22050 Hz.
APRS: KU2Y-1>CQ-1:hi hi 145 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600 600 600 650

APRS: KU2Y-1>CQ-1:hi hi 145 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600 600 600 650

APRS: KU2Y-1>CQ-1:hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600 600 600 649

APRS: KU2Y-1>CQ-1:hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600 600 600 651

APRS: KU2Y-1>CQ-1:hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600 600 600 649

APRS: KU2Y-1>CQ-1:hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 500 500 600 600 600 651
```

Fig. 8. Automatic Telemetry Decoding in Windows with multimon-ng.

The Windows configuration for this Automatic Decoding is in the Wiki page [5].

This command could also be used on a Raspberry Pi. However, there are other significant advantages to a Raspberry Pi Ground Station that are discussed in the next section.

IX. RASPBERRY PI GROUND STATION

It is possible to configure a Raspberry Pi as a Manual or Automatic Ground Station, as described in the previous section. However, there are additional other benefits to using a Raspberry Pi.

1. All the software can be pre-installed on a micro SD card, which can be shipped or downloaded to turn any Pi instantly into a CubeSat Simulator Ground Station. This saves about an hour of software download and installation and configuration, and is guaranteed to work.

2. The OpenWebRX application [6] can be used to share the RTL-SDR with other computers running just a web browser. Up to twenty users can then independently tune and decode the telemetry -- very useful for a classroom exercise. See Figures 9 and 10.

3. Even the spreadsheet telemetry analysis can be done on the RPi using LibreOffice Calc. The LibreOffice suite is pre-installed in the standard "Raspbian OS with desktop and recommended software" as shown in Figure 11.

4. The Pi can be configured as a Wi-Fi Access Point (AP) using RaspAP [7] so that a dedicated Wi-Fi router is not needed.

5. A teacher without a computer or laptop can use the Pi if they have an HDMI monitor (display), USB mouse and USB keyboard.

6. An inexpensive touch LCD display and a USB power pack turns the CubeSat Simulator Ground Station into a cool handheld device, although the text size on the screen is quite small.

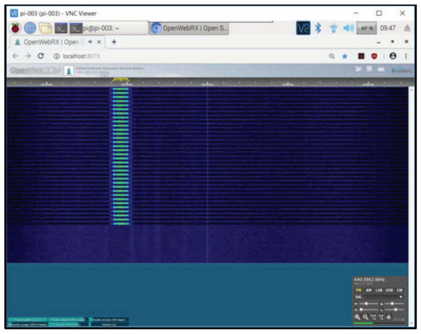


Fig. 9. OpenWebRX on a Raspberry Pi

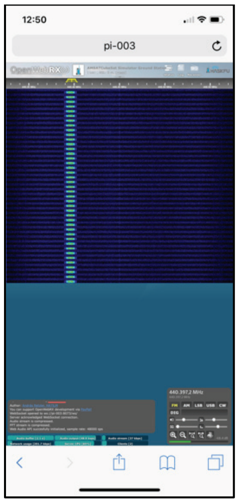
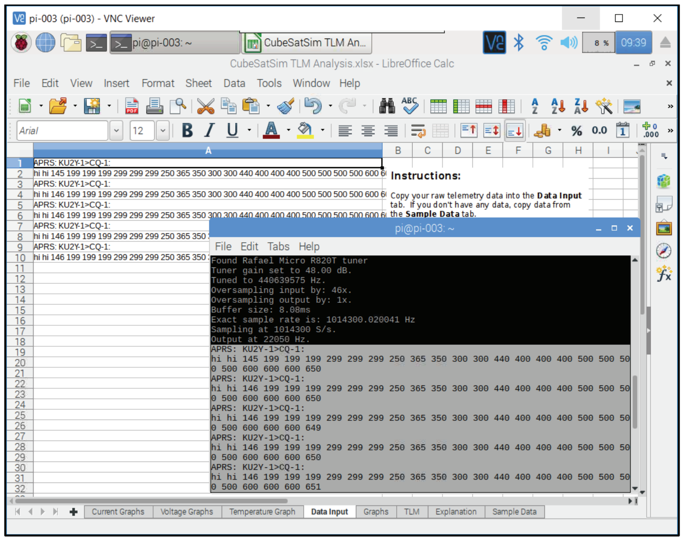


Fig. 10. OpenwebRX on iOS


pi-003 (pi-003) - VNC Viewer
CubeSatSim TLM Analysis.xlsx - LibreOffice Calc
File Edit View Insert Format Sheet Data Tools Window Help
pi@pi-003: ~
File Edit Tabs Help
Current Graphs
Voltage Graphs
Temperature Graph
Data Input
Graphs
TLM
Explanation
Sample Data

Instructions:

Copy your raw telemetry data into the **Data Input** tab. If you don't have any data, copy data from the **Sample Data** tab.

```
Found Rafael Micro R820T tuner
Tuner gain set to 48.00 dB.
Tuned to 440639575 Hz.
Oversampling input by: 46x.
Oversampling output by: 1x.
Buffer size: 8.08ms
Exact sample rate is: 1014300.020041 Hz
Sampling at 1014300 S/s.
Output at 22050 Hz.
APRS: KU2Y-1>CQ-1:
hi hi 145 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 50
0 500 600 600 600 650
APRS: KU2Y-1>CQ-1:
hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 50
0 500 600 600 600 650
APRS: KU2Y-1>CQ-1:
hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 50
0 500 600 600 600 649
APRS: KU2Y-1>CQ-1:
hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 50
0 500 600 600 600 650
APRS: KU2Y-1>CQ-1:
hi hi 146 199 199 199 299 299 299 250 365 350 300 300 440 400 400 400 500 500 50
0 500 600 600 600 651
```

Fig. 11. Cut and Paste on Raspberry Pi Between Decoder and Spreadsheet.

The full instructions on how to configure a Raspberry Pi as a ground station are on our Wiki [8] and has been tested with a Raspberry Pi 3B, 3B+ and 4B, as shown in Figure 12.

We plan to include a Raspberry Pi Ground Station with the AMSAT CubeSat Simulator units available to loan for your classroom or event.

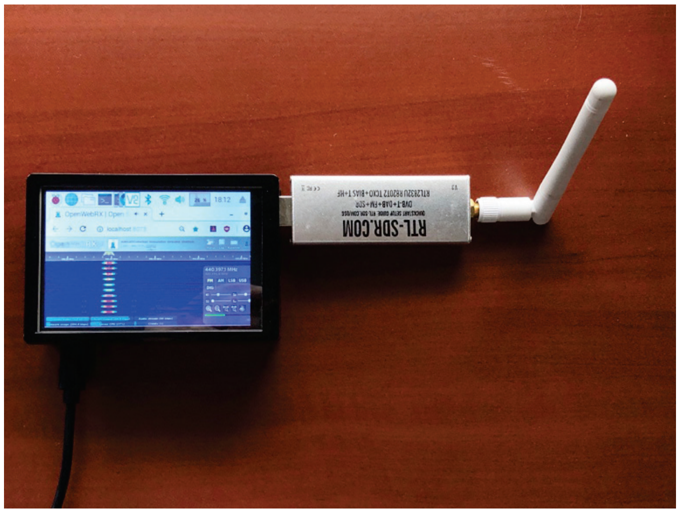


Fig. 12. Raspberry Pi Ground Station running OpenWebRX with Touch LCD Display

X. Future

In addition to running their own ground station, many CubeSat operators can benefit from SatNOGS, the open source global network of receive-only satellite ground stations [9]. To show how this works, we have been collaborating with SatNOGS to enable CubeSat Simulator data to be processed and stored in their databases. For example, you can see an experimental Dashboard for the CubeSat Simulator at this link [10].

You can also see all the other satellites telemetry dashboards displayed here [11].

A future article will describe how to use the CubeSat Simulator to explore SatNOGS.

The AMSAT CubeSat Simulator project is still a work in progress. We are continuing to improve and develop materials. We also wish to hear from you, with questions, comments and ideas which may be reflected in a future article!

Conclusion

In this paper, we have explored the options for a CubeSat Simulator Ground Station. Some of the options are very low cost, such as the Windows RTL-SDR approach. Others, have great flexibility and possibility in the classroom, such as the Raspberry Pi Ground Station. We strongly believe that CubeSat developers can benefit from setting up and using a CubeSat Simulator Ground Station. One's skills and techniques developed here will transfer directly to those required for the real flight model.

The authors and our beta testers are always looking for feedback on these activities or new activities. Please share any feedback with us at ku2y@amsat.org and n8pk@amsat.org.

References

[1] https://countingfromzero.net/amsat/CubeSatSimPaper.pdf

[2] https://destevez.net/2016/06/kiss-hdlc-ax-25-and-friends/

[3] https://github.com/alanbjohnston/CubeSatSim/wiki/Decoding-Telemetry

[4] https://www.rtl-sdr.com/rtl-sdr-quick-start-guide/

[5] https://github.com/alanbjohnston/CubeSatSim/wiki/Windows-Automatic-Telemetry-Decoding

[6] https://sdr.hu/openwebrx

[7] https://github.com/billz/raspap-webgui

[8] https://github.com/alanbjohnston/CubeSatSim/wiki/Raspberry-Pi-Ground-Station-Setup

[9] https://satnogs.org/

[10] https://dashboard.satnogs.org/d/VesVjq6mk/amsat-cubesat-simulator

[11] https://dashboard.satnogs.org

[12] https://github.com/alanbjohnston/CubeSatSim/wiki/No-Install-RTL-SDR-Adapter-for-Windows,-Mac,-iOS,-and-Android

Appendix - No-Install RTL-SDR

This appendix describes a no-install RTL-SDR approach that does not require driver installation, administrator rights on the computer, and works with Windows, Mac, Linux, Android, and iOS.

The approach is very simple and requires a $20 OpenWrt Wi-Fi hub, an Ethernet cable, and the use of an SSH client such as PuTTY or ssh. The steps to configure and install the software on the hub are described in the Wiki here [12] The RTL-SDR is plugged into the USB port of the hub, and the computer accesses it over Ethernet or Wi-Fi using the `rtl_tcp` application running on the hub.

On Windows, plug the USB power cable of the hub into a USB port on the laptop. Plug the Ethernet cable into the laptop and into the LAN port on the hub. Run SDR# software as usual, but for the input, select RTL-SDR(TCP) instead of RTL-SDR(UDP) option. You will need to enter the IP address of the hub, which for the model I used was 192.168.8.1. Click the Play triangle, and you will have access to the RTL-SDR over the Ethernet cable, as shown in Figure 13.

This can also be used without the Ethernet cable by connecting to the Wi-Fi of the hub. The downside to this is that you will lose internet connectivity since the hub does not have a WAN connection to it.

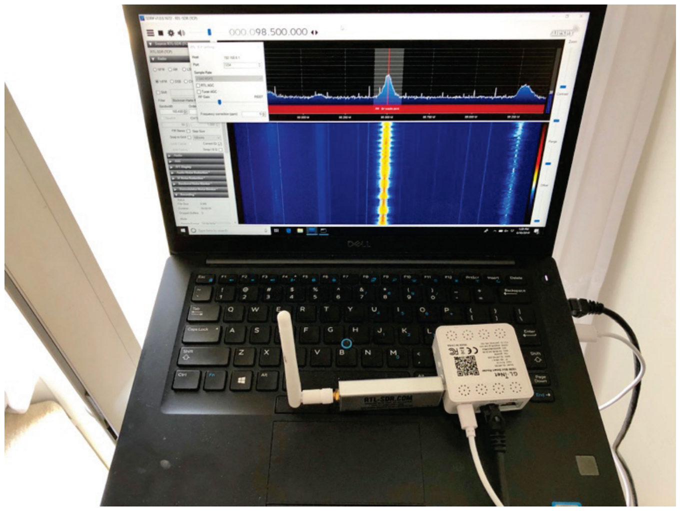


Fig. 13. No Install RTL-SDR with Windows Laptop.

This Wi-Fi approach also works for Android and iOS. If a powerpack is used to power the hub, then you have a wireless solution, as shown in Figure 14.

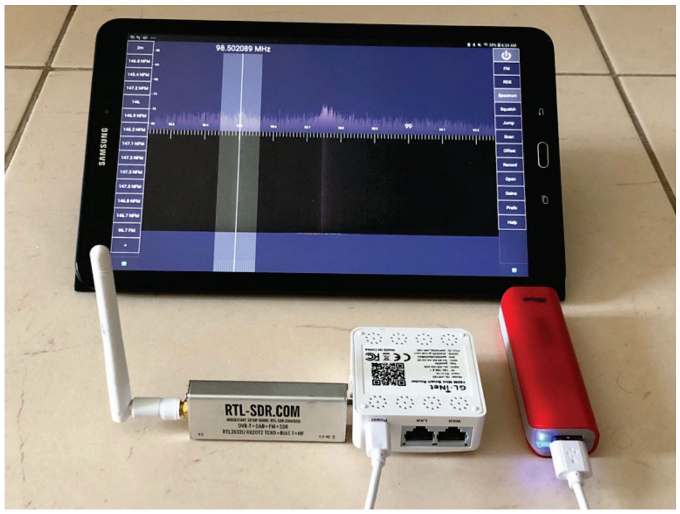


Fig. 14. No-Install RTL-SDR with Android, running SDR Touch Application.

Sometimes the RTL-TCP connection will stop working. In order to re-establish it, you will need to reset the hub by turning it off and on by unplugging it from the USB power source.

Authors' Note: This article is Part 4 in a series. The other parts are available in the AMSAT Journal or online at cubsatsim.org/papers