The AMSAT® Journal

Volume 43, Number 2 March/April 2020

A New Member Portal Launches

in this issue ...

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

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

A New Design for the AMSAT CubeSat Simulator

Alan Johnston, Ph.D, KU2Y
Vice President, Educational Relations, AMSAT
Associate Teaching Professor, Villanova University
ku2y@amsat.org

Jim McLaughlin, KI6ZUM
ki6zum@gmail.com
Zum Radio, STEM Advocate

David White, WD6DRI
Professional Geologist
Lifetime Amateur Radio Enthusiast
STEM Advocate
wd6dri@gmail.com

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

Introduction

The AMSAT CubeSat Simulator, also known as the CubeSatSim, has been under development for over 18 months. Initially shown at the 2018 AMSAT Space Symposium, it was officially launched at the 2019 Hamvention, and since then, loaner units have been shipped to events and classrooms around the country. At the 2019 AMSAT Space Symposium, some design updates were shared including a new Fox emulation mode. In this article, we describe a new hardware design for the AMSAT CubeSatSim. While backwards compatible with the old design, the new design has many new features and has a lower cost to build.

Background

The AMSAT CubeSatSim, shown in Figure 1, is a Raspberry Pi Zero W-based functional model of a "1U" CubeSat nanosatellite. It has a 3D-printed frame structure and is designed to act, as reasonably as possible, as one flying in Low Earth Orbit (LEO). Its purpose is to demystify to all how satellites work. Like typical LEO satellites, this simulator runs on rechargeable battery power and solar panels. The CubeSatSim transmits its authentic voltage, current, and temperature telemetry on the UHF Amateur Radio band. For details on the design and construction, see our series of articles in the *AMSAT journal* and in the 2018 and *2019 AMSAT Space Symposium Proceedings*, or the set of resources at https://cubesatsim.org.

Figure 1 - The AMSAT CubeSatSim Satellite Simulator.

Alan, KU2Y, and Pat, N8PK, have been collaborating on the CubeSatSim since 2018. In 2019, Jim, KI6ZUM, and David, WD6DRI, became involved in the project. David was a very early Beta Builder and provided useful feedback on his build. He and Jim got together and offered their help in redesigning the board stack to reduce costs, using Jim's experience in manufacturing other amateur radio project boards. The latest collaboration started in the summer of 2019, resumed in the fall, and a number of versions and designs were proposed in late 2019 and early 2020 that resulted in the current new design.

Jim and David have been collaborating on STEM related projects since 2010 when they became mentors to the Mt. Carmel High School Amateur Radio Club, W6SUN, in San Diego, California. (Interesting side note: One graduate of that high school program is currently a college freshman engineering student of Alan's!) They, along with several other hams, continue to mentor at the high school where the students have launched several high-altitude balloon (HAB) flights with a variety of Amateur Radio equipped payloads. In addition to HAB payloads, students are currently working on an oceanographic buoy they plan to release in the Pacific Ocean and track via an on-board ham radio equipped tracking system. Jim and David have conducted a number of ARISS (Amateur Radio on the International Space Station) events from local elementary schools and from the Ruben

H. Fleet Science Center in San Diego's Balboa Park that have allowed students to talk directly with astronauts onboard the ISS via ham radio.

New Design Goals

At the 2019 Space Symposium, we described some design updates which replaced the Digital Transceiver for the Raspberry Pi board used in the original design. This reduced the cost of the CubeSatSim and also provided additional modulation schemes such as the FSK DUV (Frequency Shift Keying Data Under Voice) and BPSK (Binary Phase Shift Keying) Fox satellite emulation modes. This was accomplished by a new Transmitter/Filter Board **(TFB)** which replaced the old transceiver board. The redesign described in this article continues along this line, replacing the MoPower V2 UPS (Uninterruptible Power Supply) battery charging board, which is a Raspberry Pi HAT (Hardware Attached on Top). This new design also adds a new STEM Payload board for one's own experiments.

The MoPower V2 UPS battery charging Pi HAT **(www.allspectrum.com/mopower/)** was chosen for the original CubeSat Simulator design because it provided a number of needed functions, including:

- Charging of the Nickel Metal Hydride (NiMH) battery
- Regulation of the 5 V power supply to the Raspberry Pi
- Current and voltage monitoring of the 5 V power supply
- An on/off and reboot pushbutton
- Automatic shutdown when the battery voltage goes below a threshold level

While the MoPower board provided these features, after the replacement of the Digital Transceiver Board, it became the single largest cost item, at nearly *$75*. A goal of the redesign was to replace this board with components that cost substantially less.

At the same time, a goal was also to improve some of the specifications, including to decrease the "flight" battery charging time. Because of its design as a UPS backup, the MoPower V2 UPS board supported only trickle charging. This meant that a fully discharged battery would require up to six hours to be fully recharged. NiMH batteries can support faster charging modes.

Battery Charging

Jim and David suggested to replace the MoPower V2 UPS board with an integrated circuit NiMH charging module, and to switch from the 9 V **NiMH** battery to a set of individual cells. At the same time, Alan investigated a higher voltage and current power solar cell, one that has a specified Voc (open circuit voltage) of 5.5 V and an lsc (short circuit current) rating of 160 mA. The new method utilizes the lower battery voltage and higher solar panel output voltage resulting in a design that is both inexpensive and has good performance. The MoPower UPS board required an input voltage of at least 12 V to charge the 9 V NiMH battery. With the original 4 V solar panels, this required a boost regulator to increase the 4 V to 15 V to charge the battery. The new design uses three cells in series for a 4.5 V battery pack, which can be either AA or AAA size. With the new 5.5 V solar panels, this eliminated the need and cost for a battery-charging boost circuit.

After testing a number of different NiMH charger boards, we standardized on the CN3058 NiMH charging board. We chose the "3S"version which uses three cells. This charger can provide up to 1 A in constant current mode, and automatically shuts off when the charging is complete.

The MoPower v2 UPS board used a 12 V DC powerpack with a barrel connector for external charging. This was an expensive powerpack, and the barrel connector is a poorly standardized connector, with diameter differences that can be seen by the eye, causing incompatibility. We took the opportunity of the redesign to switch to a micro-USE connector for a 5 V charging input. This allows for a Raspberry Pi power supply or any mobile phone micro-USB charging cable to be used. We supply the 5 V from the micro-USE through a diode for polarity protection and a 5.2 V Zener diode for overvoltage protection. In the future, we may switch to a USB-C charging input after these cables become more common and cheaper.

5 V Supply for Raspberry Pi

The MoPower V2 UPS board supplied power from the 9 V NiMH battery using a buck converter to reduce it to 5 V to power the Raspberry Pi. In our new design, we power the Pi using a boost converter regulator module to generate the 5 V for the Pi. For the boost regulator, we chose a board with 1 V - 5 V input voltage and 5 V output voltage capable of supplying up to *500* mA. We tested a number of other boost regulators which claimed similar specifications, but they did not perform well at input voltages below 4 V. Since we wanted to operate our battery in the range 4.5 V to 3.0 V to allow direct charging from the solar panels,

NiMH vs. Li-Ion Batteries

The most common question by far that we get asked about the design of the CubeSatSim is about our choice of NiMH batteries over Li-ion batteries. Li-ion batteries have much more capacity and are lighter than NiMH batteries and are much more common. We made this design decision very early on in the project for two main reasons:

1. Ease in shipping. Every commercial shipper and airline asks questions about Li-ion batteries due to the fire risk associated with them. By using NiMH batteries we can always answer no to their questions and ship them without any restrictions.

2. An on-board battery of a lower capacity is actually better for demonstrations in classrooms and in live settings for educational purposes. With a low capacity battery, you can more readily observe the effects of charging and discharging in raising or lowering of voltages in a few minutes. Compare this to a higher capacity Li-Ion battery which requires hours of charging or discharging to see changes happen.

Since the initial design, we have also realized another advantage of NiMH over Li- Ion -- safety during construction. Alan actually experienced this lesson recently while building a new prototype. During **an** initial test of charging the battery, he knew something was clearly wrong in that the charging current was too high and the battery voltage dropped as soon as it was connected to the PCB. After a few components burned out because of this, he tracked down the problem. On a two-wire JST 2.0 connector wire pigtail used to connect the battery holder to the PCB, he discovered that the red and black leads on the connector wire were reversed! (A closer reading of the Amazon reviews for the product showed that more than many reviewers pointed out this polarity reversal, such that positive was the black wire and negative the red wire -- a good reason to read reviews carefully!) Reversing the polarity of a Li-Ion battery might have caused a fire!

this ruled them out. The input to the 5 V regulator from the battery and the battery charger goes through a polyfuse resettable fuse (also known as a PTC or polymeric Positive Temperature Coefficient device) rated at 1.5 A for overcurrent protection.

To replicate the 5 V bus current and voltage monitoring, we just added additional INA219 current and voltage sense boards. With one for the battery, one for the 5 V bus, and one for each of the six solar panels, this means that there are eight INA219s in use. We exclusively use the purple board type because it has a smaller footprint than the original blue boards. Four INA219s are on the Raspberry Pi i2c-1 I2C bus, and the other four are on the i2c-3 I2C bus. The i2c-0 bus is no longer used in this design, as it should only be used for Pi HAT sensing, something we may implement in a future design consideration.

Pushbutton

Once we had the basic battery charging and 5 V regulation functionality working, we then needed to replicate the other functionality that the MoPower board was providing. For the on/off reboot pushbutton, we used a DPDT (Double Pole, Double Throw) momentary pushbutton switch. To start up the Pi, we use one of the poles of the switch to briefly short GPIO Pin 5 and Pin 6 (GPIO3 and GND, respectively) which automatically wakes the Pi from a halt state and boots it up. To shutdown the Pi, we use a small Python program, the Howchoo pi-power-button (**http://howchoo.com/g/mwnlytk3zmm/how-to-add-a- power-button -to-your-raspberry-pi**), which runs on boot and monitors GPIO Pin 27 which is wired to the other pole of the switch. When the Pi is running, if this GPIO pin is shorted to ground for less than 1 second, the Python code performs a reboot (executing the command sudo reboot
-h now). If the GPIO pin is shorted for over 3 seconds, the Python code performs a shutdown to the halt state (executing the command "sudo shutdown -h now"). To simplify construction, we chose a right angle switch that mounts directly on the PCB so that no wiring is needed and no frame mounting is required for the switch.

Remove Before Flight Switch

During the redesign, we also changed the Remove Before Flight (RBF) switch to be more like a typical flight CubeSat RBF switch. In the old design, the RBF switched between DC power input and solar power input. This meant that the Simulator could still be run with the RBF plug inserted . It also meant that the only way to fully disconnect power to the Simulator was to physically unplug the battery. In this new CubeSatSim design, the RBF switch disconnects all power when the plug is inserted. However, the battery can still be charged via DC input or the solar panels. The RBF switch in the old and new design utilizes two micro switches built into a TRS 3.5mm audio jack. The old design also used a jumper in the RBF 3.5 mm plug to route power. In the new design, there is no jumper, so it is possible to use any 3.5 mm plug or even a 3D printed plastic plug in the shape of a 3.5 mm plug as the switch.

Figure 2- The AMSAT CubeSatSim board stack: from the bottom, Raspberry Pi Zero WH, Main PCB, Battery Board, STEM Payload Board.

Automatic Shutdown

The automatic shutdown that is based on battery voltage level on the MoPower v2 UPS board is done using a microcontroller. We considered implementing some kind of hardware based detection and shutdown. Ultimately, we decided to just implement the function in software. Since we are already monitoring the battery voltage using an INA219 in the CubeSatSim software, it is easy to periodically check the voltage and perform shutdown if it drops below a threshold, such as 3.0 V.

Band Pass Filter

We also chose to integrate the 433 MHz Band Pass Filter (BPF) into the PCB, eliminating the need to buy a discrete BPF. We tried to design the BPF so that we didn't use surface mount device (SMD) components, but we were not able to easily. Instead, we implemented a select few surface mount components for this filter. If a builder is unable to obtain access to a reflow oven to solder these surface mount components, a discrete external BPF could still be used, such as the one used with the retired TFB. By shifting the antenna mounting position to the corner of the board, we are also able to have the option to incorporate a tape measure monopole antenna, as shown in Figure 1. A small screw through the center hole of the SMA connector holds the ¼" tape measure length in place. A small "rubber duck" external antenna, as was featured in the original design, can still be used by installing the SMA connector and running a short length of coax.

Stacking

In the original design, three boards were stacked to provide the transmitter, battery charging, and solar power functionality bundle. We required a stacking header on the Raspberry Pi Zero W so that we could plug in the solar power board below it, and the charging board and transmitter board above it. In the new design, we integrated all these functions into a single board, and then had the board plug on top of the Pi, as shown in Figure 2.

We continued to use stacking GPIO headers on the board so that we could stack a battery board and the new STEM Payload board

Figure 3 - The Main PCB Showing components mounted on top and bottom of PCB.

above it. To give space between the boards, we double up on the stacking GPIO headers. In this design, we no longer require a stacking header on the Pi, so a standard Pi Zero WH can be used, or a Pi Zero W with a male header added. This means our new board is a Pi HAT, unlike the previous design where boards plugged on top and under the Pi. To fit everything on the one board, we mount components on both the top and bottom of the Main PCB, and mount all the switches and connectors as shown in Figure 3. This shows the RBF, pushbutton, and charging connectors on the left side, and the antenna connector and BPF on the right side.

Frame

We had not planned to change the 3D printed frame for the CubeSatSim, but when we learned about a new 1U CubeSat frame by Juliano85 posted in Jan 2020 **(https://www.thingiverse.com/thing:4333605)** we were eager to print one and give it a try. It turns out to be a more sturdy, easy-to-print and assemble frame,

Figure 4- New 3D Printed CubeSatSim Frame by Juliano85 on Thingiverse.com.

as shown in Figure 4. Also, with the larger 5.5 V 90 mm x 80 mm solar panels we incorporated, the panels cover almost the entire side, as seen in Figure 1, so we need only a minimal frame exo-structure such as this one. By locating all the switches, connectors, and LEDs on the +X side of the board, we have the luxury of a full-sized solar panel on that side as well.

STEM Payload Board

The original CubeSat Simulator design imagined a payload subsystem, where additional sensors would be read by an Arduino which would then communicate with the Raspberry Pi over the I2C bus. While some students did implement this, not having a physically mounted board was a real problem.

Jim and David originally developed the STEM Payload Board as a STEM board in partnership with Bob Twiggs, KE6QMD, the co-inventor of the flight CubeSat standard. Bob wanted a way to draw K-12 education back into the CubeSat world and to have a model that was inexpensive enough that a whole classroom of students could afford a kit to explore and in which to be engaged.

The STEM Payload Board is built around an inexpensive STM32F103C8T6 board, sometimes called a "blue pill" board due to its shape and color. This STMicroelectronics microcontroller incorporates a 32-bit ARM processor which allows for software to be easily written using the common Arduino development environment. The board also has a BME280 temperature/pressure/humidity sensor, as well as a GY-521 module IMU (Inertial Measurement Unit) made up of a magnetometer, an accelerometer and a gyroscope. A big bang for the buck indeed!

In the new design, the STM32 communicates with the Raspberry Pi over a serial connection on the Pi Zero GPIO bus. This makes it easy to develop and debug software. We hope to share the experiments using the new STEM Payload Board in a future article.

Testing the New Design

We built a number of prototypes of the new design CubeSatSim and have run tests, and so far the results look very promising. The charging curve shown in Figure 5 shows the charging current over about Time t = 140 minutes (X-axis) for an initially nearly-discharged AAA battery. Note that the charging current is shown as negative (Y-axis) since the current is entering the battery. The charging current begins at over 300 mA. As the battery reaches capacity, the charge current drops back to about 90 mA. However, this current is too large for a trickle charge, which should be more like 15 mA for a AAA NiMH battery and 30 mA for a AA battery. Also, with the RBF switch removed, the load is such that the charger circuit never turns off. However, if the Pi is off and the

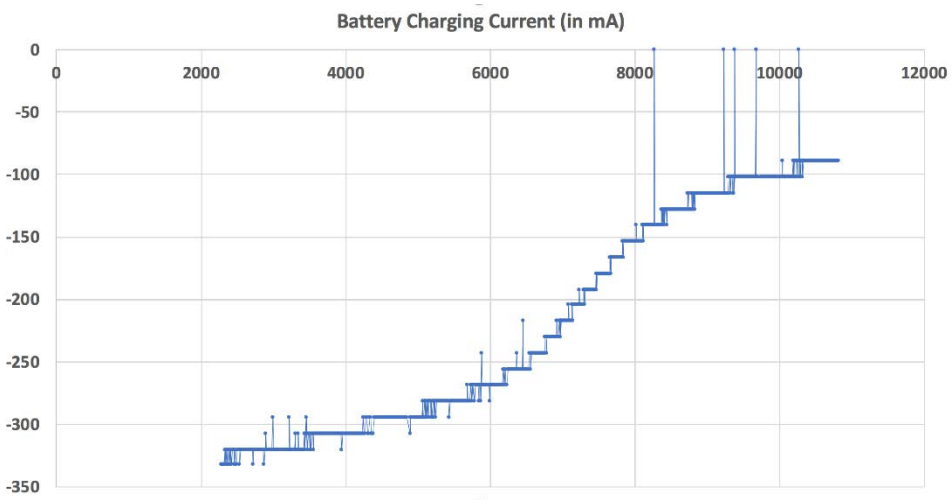

Figure 5 - Battery Charging Curve Test.

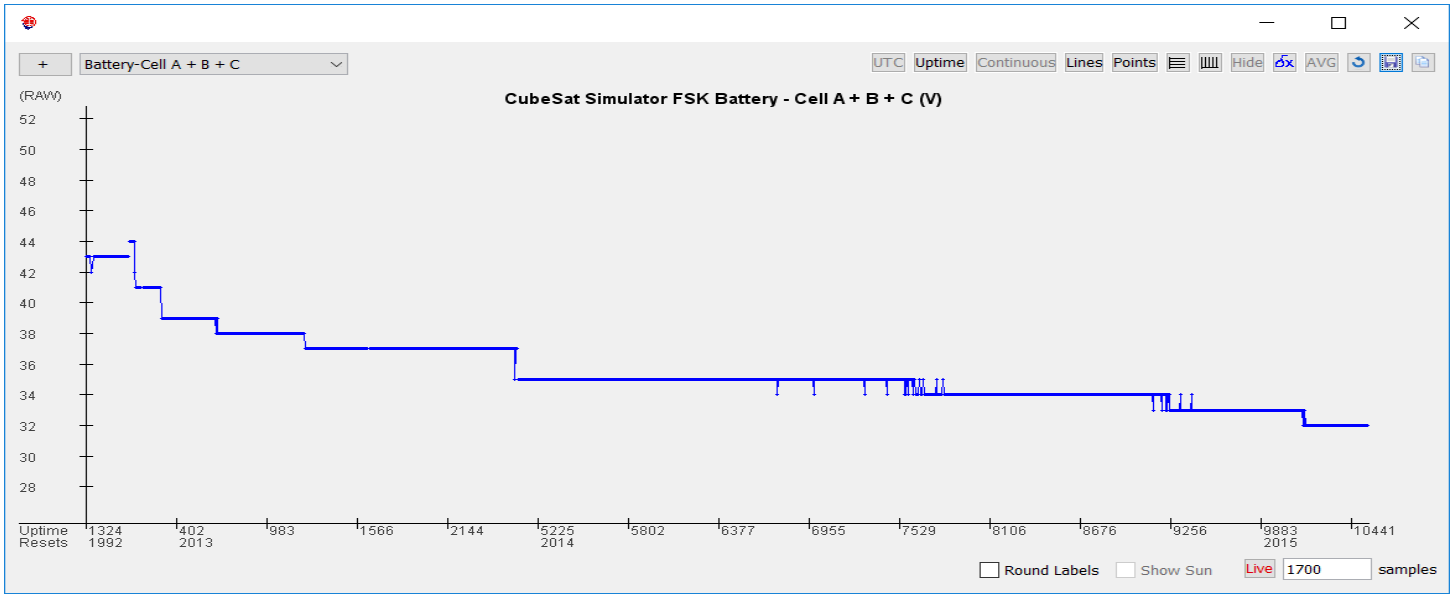

Figure 6 - Battery Discharging Voltage Curve showing Volts x 10 (i.e., 40 is 4.0 V).

Figure 7-CubeSatSim Telemetry Decoding in FoxTelem with Real-Time Graphing.

RBF switch plugged in, the charger does shut off. This means that the charger should not be left plugged in for long periods of time unless the RBF switch is in.

The discharging curve for AA batteries in Figure 6 shows that the CubeSatSim can easily run for three hours. Using AAA batteries it can run for about an hour. When the CubeSatSim is displayed at events that last for multiple hours, the AA batteries provide good longevity.

The new Electrical Power Subsystem (EPS) design is not necessarily considered "power positive." That is, the CubeSatSim will not run without batteries. This capability was a nice featur e of the old design in that it could run without a battery, but with the board mounted cells in the new design, we don't think this will be a problem.

One quirk we discovered is that if the CubeSatSim runs low on batteries and shuts down, plugging in the micro-USE charging cable does not make the Pi startup. Pressing the pushbutton also does not seem to start it up. Instead, after about 30 seconds of charging, inserting and removing the RBF switch seems to wake up the Pi and turn on the system.

The new RBF switch works as expected. One benefit is that the Pi starts up automatically when the RBF switch is removed. With a suitable delay in the demo script, this makes for a good launching demo sequence, where the CubeSatSim wakes up and starts transmitting after "launch."

Chris Thompson, G0KLA/AC2CZ, the author of AMSAT's excellent FoxTelem satellite decoding software.

(http://www.amsat.org/foxtelem-software-for-windows-mac-linux/), continued to assist us. Chris released the new test version of FoxTelem Version 1.09 with support for the FSK and BPSK modes of the CubeSatSim. See Figure 7 for a graph showing solar panel voltages, currents, and temperature under illumination and eclipse (darkness) . We are greatly enjoying the real-time telemetry graphs generated by FoxTelem! We will discuss experiments and tests which can be performed with the CubeSatSim and FoxTelem in a future article.

We have also been experimenting with the AMSAT CubeSat Simulator on a turntable in front of an LED lamp as a substitute for the fuller radiant spectrum delivered by the halogen lamp. While the LED lamp does not activate the solar panels to produce sufficient current, you can see the periodic peaks on the voltage plots. This provides an alternate demo that does not involve the extremely hot *250* W halogen lamp. For many demos, we believe this will be good enough. For others, placing the CubeSatSim model in the natural sunlight does indeed generate currents.

In particular, at high noon at our latitude, even in early spring, it is possible to generate a small charging current to the battery, something we were never able to do with the old design. In mid-summer or at lower latitudes, it should be possible to generate a decent charging current for the battery to run the Simulator in the sunlight for very long periods!

In a future article, we will show how to use FoxTelem to analyze the Simulator telemetry and to calculate the efficiency of the electrical power subsystem. We have some initial results that suggest that our efficiencies are as good if not perhaps a little better than the old design.

Conclusion

This article has described our redesign and upgrade of the AMSAT CubeSat Simulator. We are very happy with the results. The new CubeSatSim design is available on a GitHub link from https://CubeSatSim.org including full schematics, board CAD files and Gerber files, and open source software. You can explore the site at **CubeSatSim.org.**

Have you built a CubeSatSim using this new design? If so, we'd love to hear from you. Feel free to share on Twitter with the **#CubeSatSim** hashtag or on the AMSAT Facebook page. Also, we'd love to hear from you.