The AMSAT® Journal

Volume 42, Number 3 **May/June 2019**

in this issue ...

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

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

The New AMSAT® CubeSat Simulator:

Part 3, New Activities:
Failure Simulations and Troubleshooting using Telemetry

Alan B. Johnston, PhD, KU2Y
Vice President Educational Relations, AMSAT
Assistant Professor of Electrical and Computer Engineering,
Villanova University
Villanova, 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 paper describes educational activities that can be performed with the Simulator relating to failure simulations and troubleshooting using telemetry. In addition, we demonstrate the use of an Arduino as payload.**

Keywords—cubesat, simulator, telemetry, AMSAT, ham radio, raspberry pi, arduino

I. Introduction

In 2018, we introduced the new AMSAT CubeSat Simulator as a tool for satellite technology education and demonstrations. We described the proof-of-concept prototype that we built and demonstrated at the 2018 AMSAT Annual Meeting & Space Symposium in Huntsville, Alabama.

In *The AMSAT Journal,* January/February 2019 issue, we described some educational activities that can be done with the CubeSat Simulator by looking at the activities of the original ARRL ETP CubeSat Simulator, as described by Mark Spencer, WA8SME, roughly ten years ago, as fully referenced in our earlier works.

In this article, we describe some new activities that we have developed with the new CubeSat Simulator. These include some interesting failure simulations, efficiency and maximum power point calculations. In addition, we discuss using an Arduino as a payload for the Simulator.

II. Background

The new AMSAT CubeSat Simulator, shown in Figures 1 and 2, is a Raspberry Pi Zero W-based, 3D-printed frame structure, functional model of a "1U" CubeSat that 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 cell panels. Our model currently transmits UHF telemetry on the 70 cm ham radio band using the AMSAT OSCAR 7 (AO-7) format using AFSK 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.

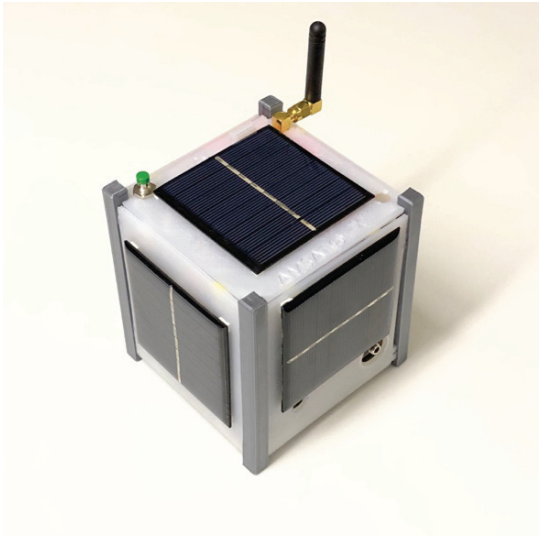

Fig. 1. The AMSAT CubeSat Simulator Proof of Concept Prototype.

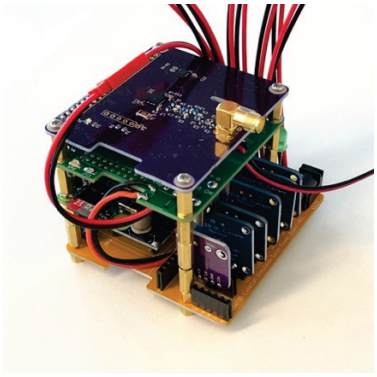

Fig. 2. The latest CubeSat Simulator board stack showing (from the top) the Digital Transceiver Board, MoPower UPS V2 Board, Raspberry Pi Zero W Board, and the custom AMSAT Solar Power Board with Current and Voltage Sensors Mounted Vertically.

The telemetry data graphs shown in this paper were generated by placing the CubeSat Simulator on a rotating turntable in front of a halogen work lamp, which simulates a spinning satellite in space, as shown in Figure 3.

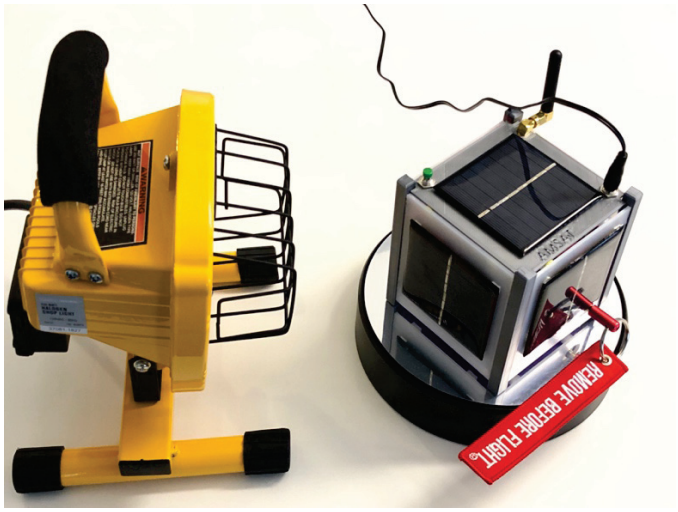

Fig. 3. The CubeSat Simulator on a Rotating Turntable under Halogen Work Lamp Illumination.

The remainder of this article describes new activities for the CubeSat Simulator. They include some real-world simulated and actual failures, with several plots to support troubleshooting, and also some efficiency calculations. Each of these simulator exercises to date provide the basis for valuable lessons in understanding satellite technology and in developing one's skills.

SATELLITE FAILURE

The literature in the small satellite community ranks the frequency of on-orbit failures by subsystem. Of the top ten major subsystems of every satellite (shown in Part 1 of this CubeSat Simulator series), it is the Electrical Power Subsystem (EPS) that seems to appear at or near the top of such lists. The EPS includes the solar cell strings and panels, the batteries, charging controller, heaters, thermostats, and the power distribution circuits, among other components as well. The literature shows a horror story involving any number of solar panel anomalies [2].

In studying the Physics of Failure, we find two major categories of root causes in an otherwise well-designed and well-built system. One is from latent (or built-in) defects of a part. The other is from overstress. Overstress is the exposure at any level of assembly to an excessive electrical, mechanical, thermal or other condition beyond its documented specifications or requirements. The scope of this paper is therefore on anomalies common in a typical CubeSat EPS.

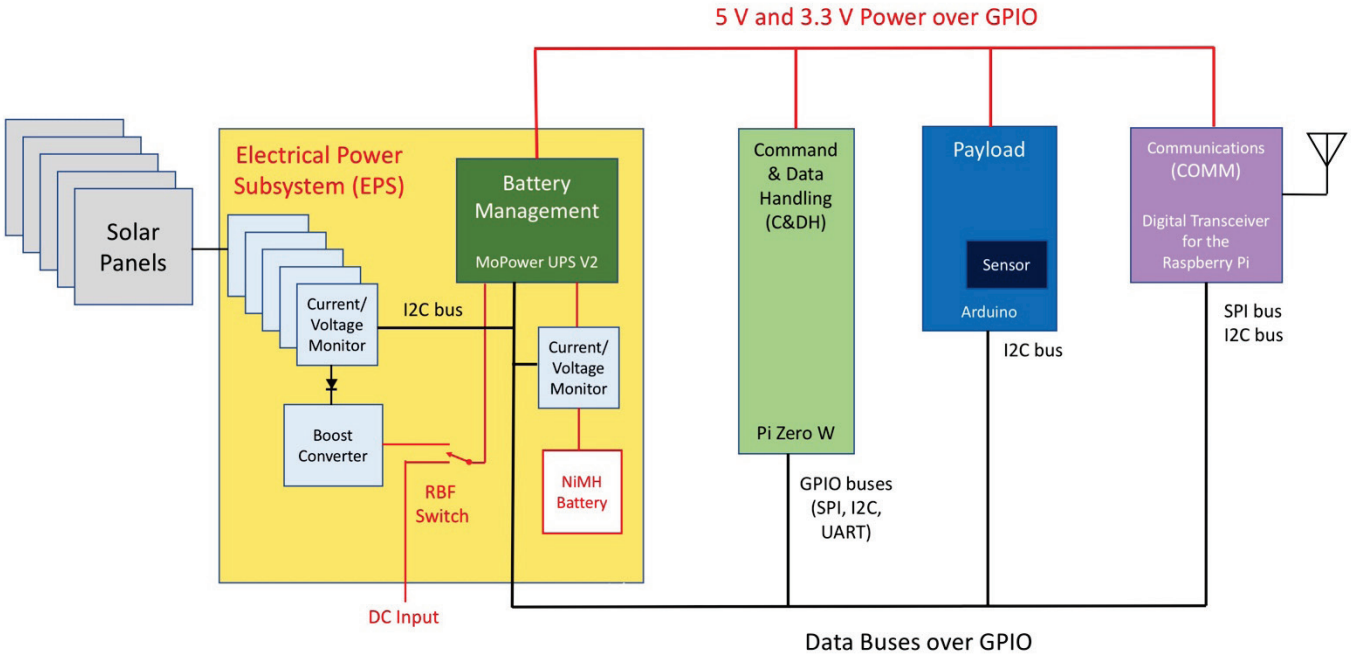

Fig. 4. Block Diagram of the CubeSat Simulator

III. Failure Simulations

CubeSat missions on orbit do not always go according to plans. Functional issues stemming from CubeSat design, electrical and mechanical parts, assembly, rework, handling, or the integration and test phase is a fact of life. The tension or strain from sources such as launch, the radiation environment, outgassing effects, loose conductive particles afloat and more can cause intermittent operations, degraded performance or even, unfortunately, a failure. Detecting and diagnosing impending failures is important, as it can help develop workarounds or solutions, or at least aid in avoiding such issues in the future. A steely-eyed missile man once confided that the worst kind of failure is having launched a "flying brick," where no telemetry is received from a satellite after being deployed, and as a result, there are no clues—and no recourse—as to what kind of failure occurred.

In this section, we will simulate a degrading performance parameter or two and a few nearly-catastrophic failures, and then walk you through some steps on how to detect and diagnose the symptoms by using the available housekeeping telemetry. Our real-life training anomalies relating to the EPS are the following:

- Solar cell short circuit
- Solar cell open circuit
- Solar cell polarity reversal
- Solar cell high impedance
- I2C sensor failure
- I2C bus failure
- Boost converter failure

These failures can be easily simulated with the AMSAT CubeSat Simulator with a few test leads. In the latest design, we use JST connectors between the solar cells and the Solar Power Board. These connectors are widely used in RC (Remote Control) vehicles and aircraft. We use a few special JST connectors as well as some mini clip test leads to simulate these failures.

The block diagram for the AMSAT CubeSat Simulator is shown in Figure 4. The large block on the left is the Electrical Power Subsystem (EPS), which is implemented as the custom AMSAT Solar Power Board in the board stack. We will reference this EPS subsystem several times in this article when discussing failure scenarios. For more information about the Solar Power Board, including a full-size schematic diagram in color, see our CubeSat Simulator Wiki [3].

To simulate a short circuit, we can simply (and safely) connect the positive side of the solar cell to ground after the current and voltage sensor module. Interestingly, this actual failure occurred in our first iteration of the concept Simulator model built using the Beta vB3 PCB. Using the telemetry, we were able to diagnose this failure and find the cause. Figure 5 shows the telemetry we observed after constructing the Simulator where one solar panel was accidentally directly tied (or shorted) to ground.

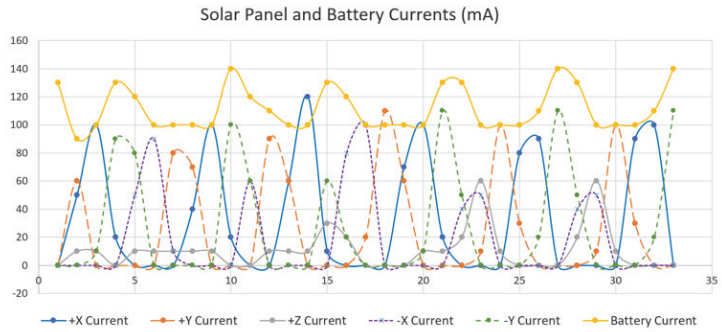

Fig. 5. Solar Panel Shorted Circuit to Ground Failure.

At first the telemetry appears to be correct, as we see four peaks of the +X, +Y, –X, and –Y panels as it rotates on the turntable in front of the halogen work lamp, as shown in Figure 3. However, upon a closer look, we see the –Y solar cell current, but there is no corresponding drop in the battery current, indicating that none of the power has been transferred to the Simulator. (Under a short circuit condition, a solar cell will produce maximum current, but since the voltage across it is zero volts, no power is produced.) This condition of maximum current but no power output indicated that the solar cell was short circuited. Using this actual telemetry, we were able to find the location of the short circuit and repair it.

An open circuit can be simulated by unplugging the JST connector to the solar cell. This results in the telemetry shown in Figure 6, where there is no current detected for the +Y panel which has been disconnected. Note that a short circuit before the current and voltage sensor would also show up in the telemetry this way.

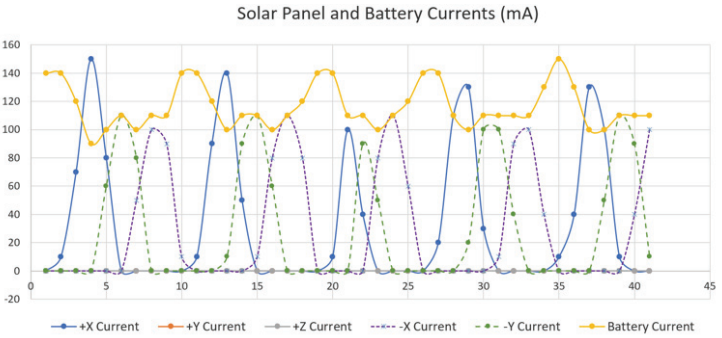

Fig. 6. Solar Panel Open Circuit Failure.

To simulate a polarity reversal, we made a back-to-back JST connector but swapping the red and black wires. This cable is shown at the bottom of Figure 7.

Note that this would be an unlikely in-flight failure but could be a construction failure that was only detected after launch. A series diode in the EPS circuit (see the diodes shown in the EPS block between the Current and Voltage Sensors and the Boost Converter module in Figure 4) in the CubeSat Simulator prevents the solar cell from drawing current from the circuit or applying a negative voltage to the output. The resulting telemetry is shown in Figure 8 where the reversal has been applied to the +X panel. The data is identical to Figure 6 where the panel was open circuited.

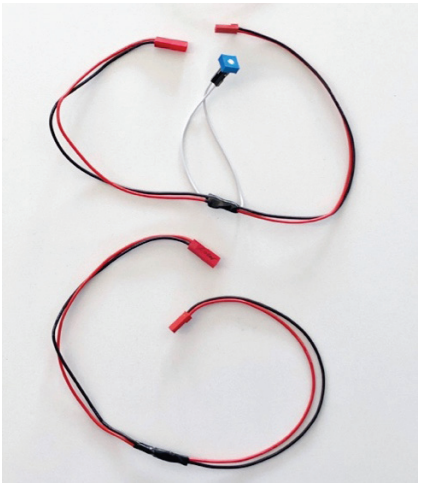

Fig. 7. Back-to-back cables used to Simulate Solar Panel Failures, top: series potentiometer cable, bottom: polarity reversal cable.

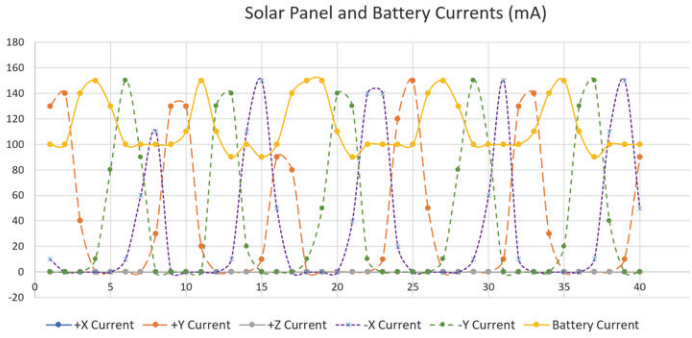

Fig. 8. Solar Panel Reverse Polarity Failure.

To simulate a failing solar cell or a high resistance contact on a cell, we made a back-to-back JST connector with a series 100 Ω potentiometer (variable resistor) which was inserted between the solar panel and the board. This cable is the top cable shown in Figure 7. The resulting telemetry is shown in Figure 9 when the resistance was connected in series with the –Y panel and set to approximately 36 Ω. The telemetry shows a current peak for the –Y panel, but it is much less than the other three panels (45 mA compared to 145 mA). If the Simulator was tilted to simulate an off-axis rotation, there would have been a corresponding change in the +Y current peak, but on this graph it is unchanged.

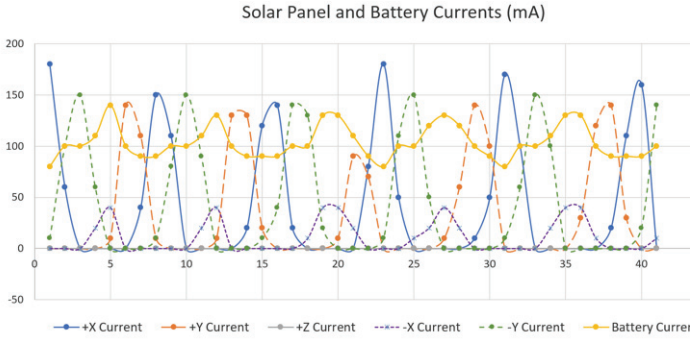

Fig. 9. Solar Panel High Resistance Failure.

The Inter-Integrated Circuit (I2C) bus is how the Raspberry Pi gathers current and voltage data for telemetry. In the CubeSat Simulator EPS, there are up to seven current and voltage sensor modules as shown in Figure 4. There is one monitor for each of the solar panels and one monitor for the battery. The CD&H accesses them via the I2C buses on the Raspberry Pi Zero W. An I2C sensor failure can be simulated by simply unplugging the current and voltage monitoring module from the Solar Power Board. This results in no current being detected, which is identical to the open circuit telemetry of Figure 6.

A failure of a complete I2C bus can be simulated in software by disabling the bus. The CubeSat Simulator CD&H uses three I2C buses on the Raspberry Pi: `/dev/i2c-0`, `/dev/ i2c-1`, and `/dev/i2c-3`. The use of these buses is shown in Table 1.

TABLE I. RASPBERRY PI I2C BUS TELEMETRY DATA

Bus	Use
`/dev/i2c-0`	–X, –Y, and –Z current and voltage sensors (addresses hex 40, 41, and 44)
`/dev/i2c-1`	+X, +Y, +Z, and battery current and voltage sensors (addresses hex 40, 41, 44, 45) and the 5V power bus current sensor (address hex 4a)
`/dev/i2c-3`	Temperature sensor on Digital Transceiver board (address hex 48)

For this simulation, we disable the i2c-0 bus on the Raspberry Pi (by commenting out the `dtparam=i2c_vc=on` setting in the `/boot/config.txt` file and then rebooting the Pi -- see [4] for details of the software configuration). Note that if the I2C bus pull-up resistors on the PCB are omitted from the board (see the Solar Power Board schematic), the I2C bus will effectively be disabled as well.

The result is a loss of current telemetry on the –X, –Y, and –Z solar cells, as shown in Figure 10. This lack of telemetry on all of the sensors on the same bus points to a bus failure rather than individual sensor failures. If all three solar cells had failed we would also have seen zeros, but we would have also seen a spike in the battery current up to 140 mA, as we saw in failures shown in Figures 5, 6, 8, and 9. Since the battery current stays under 100 mA, this confirms that it is a sensor failure, rather than a solar panel failure. If the `/dev/i2c-1` bus had failed, we would have lost the +X, +Y, +Z, and battery current telemetry information.

One curious thing about this telemetry is that the battery current dropped by approximately 40 mA from previous data. We haven't had time to investigate why this is, although it was repeatable.

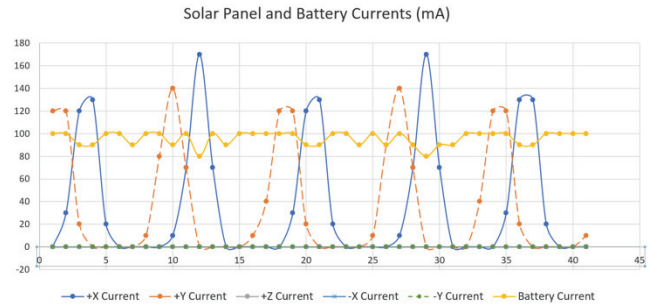

Fig. 10. I2C Bus Failure

During the building of one Solar Power Board, we incorrectly adjusted the potentiometer on the boost converter module (U1 in the schematic Figure 4). This resulted in telemetry that simulated a boost converter failure. The telemetry is shown in Figure 11. The +X, +Y, –X, and –Y current waveforms can be seen, but they are all much smaller than expected, and the resulting reduction in the battery current from 140 mA doesn't appear. Seeing this telemetry, we disconnected the output from the Solar Panel Board and measured it under illumination. Instead of the desired 15 V, it was reading 4.5 V. After re-adjusting the boost converter module so the output was 15 V under full illumination, the telemetry curves returned to normal.

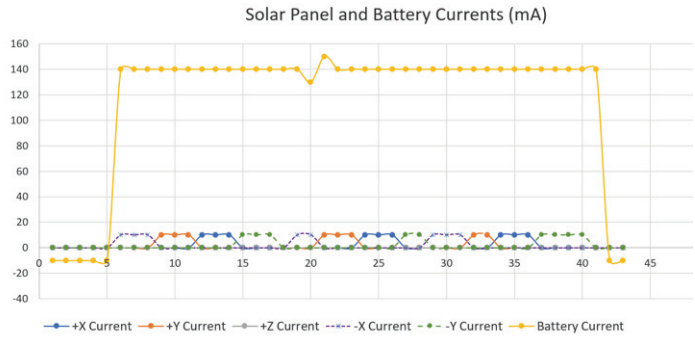

Fig. 11. Boost Converter Failure.

IV. Efficiency and Maximum Power Point

Using the voltage, current, and power sensor information available on the CubeSat Simulator, we can determine the efficiency and maximum power point (MPP).

There are two electrical efficiencies that can be determined in the CubeSat Simulator from our telemetry. The first is what we will call the battery efficiency. This is the efficiency in transferring power from the NiMH battery to the 5 V bus on the GPIO (General Purpose Input Output bus) connector. This measures the efficiency of the voltage transformation on the MoPower UPS V2 board. The product of the battery voltage and the battery current is the input power, while the 5 V bus voltage multiplied by the bus current represents the output power for this efficiency calculation.

The other efficiency is the solar power charging efficiency. The solar panels provide power to the simulator which reduces the power needed to be supplied by the battery to energize the 5 V bus. In this efficiency calculation, the solar cell power plus the battery power is the input power, while the output power is as measured on the 5 V bus.

Using the telemetry-only software on the Raspberry Pi, we measured both of these efficiencies. First, we measured the battery efficiency, running with the RBF (Remove Before Flight) pin removed but under no illumination. The data point was the battery voltage of 8.5 V, battery current of 148 mA, which is an input power of 1258 mW, and the bus voltage of 5.1 V, bus current of 209 mA, which is an output power of 1065 mW. This gave an efficiency of 85%.

Next, we used a 250 W halogen work lamp at a distance of 10 cm to illuminate one of the solar panels. We used this data to calculate the solar power charging efficiency. The data point was the solar panel voltage of 3.15 V, solar panel current of 167 mA, which is a power of 526 mW from the solar panel, a battery voltage of 8.6 V, battery current of 90 mA, which is a power of 774 mW from the battery, and the bus voltage of 5.1 V, bus current of 201 mA, which is an output power of 1025 mW. This gave an efficiency of 79 %. This is lower because the Solar Power Board includes a series diode between each solar panel and the boost converter circuit (see EPS block of Figure 4), each of which produce losses.

We also characterized the solar cell in terms of its current (I) versus voltage (V) curve and power (P) versus voltage (V). We measured this by unplugging the Vin- pin on the +X solar cell current and voltage sensor. We then connected this pin to ground through a 100 Ω potentiometer (variable resistor). This modification is shown in Figure 12. With the telemetry-only software running, we adjusted the potentiometer from 100% to 0% at 10% intervals, pausing for 1 second so that the telemetry could record the values. We then plotted this data in Excel. This graph is shown in Figure 13.

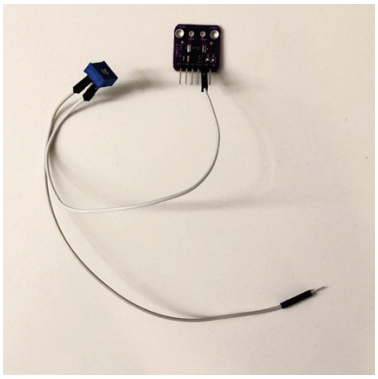

Fig. 12. Modification of Voltage and Current Monitoring Board to Characterize Solar Cell.

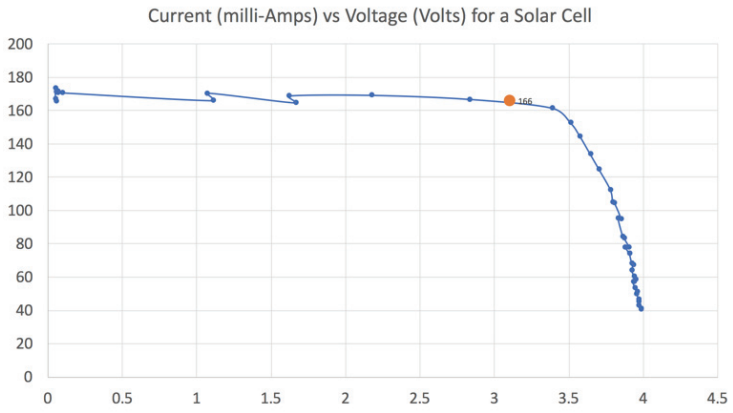

Fig. 13. Current versus Voltage for the Solar Panel.

We also graphed the Power versus Voltage characteristic for the solar cell to find the Maximum Power Point (MPP) for this illumination level. Note that a Maximum Power Point Tracker (MPPT) is an algorithm which tracks this maximum efficiency point automatically in an EPS. This curve is shown in Figure 14. The peak of this curve represents the maximum power point for this solar panel and level of illumination, which is about 3.4 V.

Dividing by the power at this point, 562 mW, gives the current of 166 mA. Note that this solar cell is rated at 4 V, 160 mA, 0.5 W, which agrees well with these results.

We then compared the actual voltage and current operating point with this maximum power point and found it to be fairly close. The operating point of 3.15 V and 167 mA is plotted on Figures 13 and 14 as a dot so you can see how close it is to the maximum power point. This indicates that the design could be improved to extract another 35 mW of power from the solar cell.

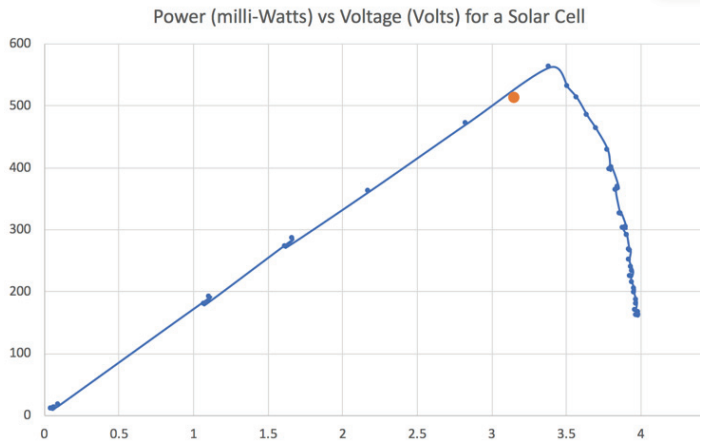

Fig. 14. Power versus Voltage for the Solar Panel Showing Maximum Power Point.

V. Arduino Payload

Just like a real CubeSat flight model, the new AMSAT CubeSat Simulator can support payloads, as shown in Figure 4. To demonstrate this, we connected an Arduino Uno [5] as a payload. An Arduino is a low-cost open source hardware and software microcontroller board which is popular with experimenters and educators. We used the Raspberry Pi `/dev/i2c-3` I2C bus to make the connection, using the expansion header on the Solar Panel Board (see Figure 4). We set the Arduino to the address of 0x4C so that it would not conflict with the other devices on the bus (the –X, –Y, and –Z current sensors). The setup is shown in Figure 15.

To demonstrate the Raspberry Pi reading data from the Arduino, we read the Arduino analog inputs A0, A1, and A2 over the I2C bus. On the Raspberry Pi, we first wrote the address number (0 to 2) then read a byte from the Arduino. The Arduino Uno sketch (C program) used the Wire library to listen on the I2C bus at address 0x4C to read the address number from the Raspberry Pi, then did an `analogRead` of the specified analog input, converted this to a single byte, then wrote it to the Raspberry Pi.

We took eight AAA NiMH battery cells and tapped each cell individually. We read the first three cell voltages using A0, A1, and A2. For example, we read the equivalent values of 230, 480, and 715 which represented the cumulative voltages of the first three cells. Converting these to voltages gave 1.12 V, 2.35 V, and 3.49 V. Taking the differences gave the first three cell voltages of 1.12 V, 1.23 V, and 1.14 V. These telemetry values could be used to detect the failure of one cell in a battery. This could be simulated by removing and discharging one cell from the battery, then replacing it back in the battery.

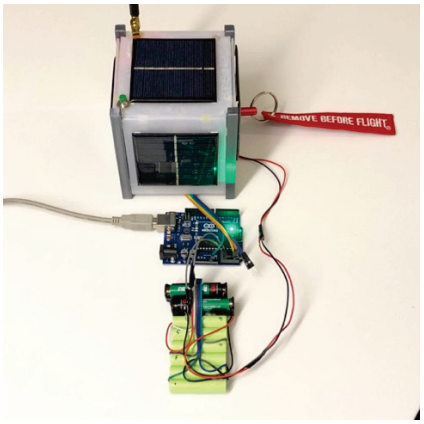

Fig. 15. Arduino Uno as a Payload on the CubeSat Simulator.

Conclusion

We have shown some new activities possible with the AMSAT CubeSat Simulator. An underlying theme of these activities is detecting and troubleshooting failures using housekeeping telemetry. This article documents and explores these activities. In the future, we will provide detailed instructions so that these activities might be more efficiently replicated in the classroom to illustrate satellite and STEM principles.

Brainstorming Future Topics

Would you like to explore the topic of adding new sensors to the AMSAT CubeSat Simulator to serve as either a payload or as a subsystem element? How about sending commands by laptop or smartphone? Or might you like to develop a satellite technology acronym "decoder" or glossary? We are listening.

If you are interested in using an AMSAT CubeSat Simulator in your classroom or public demonstration, then you have options: to build or to borrow. You are encouraged to follow our fully open source designs to build your own AMSAT CubeSat Simulator [6] or to perhaps borrow one from AMSAT. AMSAT Educational Relations has a number of simulator units available for loan. Contact us for information.

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] W. Brandhorst Jr, Henry & A. Rodiek, Julie & O'Neill, Mark. (2008). "Stretched lens array: The answer to improving solar array reliability. Conference Record of the IEEE Photovoltaic Specialists Conference.) https://www.researchgate.net/figure/Solar-array-anomalies-by-orbit_fig1_237690624

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

[4] https://github.com/alanbjohnston/CubeSatSim/wiki/Software-Install

[5] https://www.arduino.cc/

[6] https://github.com/alanbjohnston/CubeSatSim

Authors' Note: This article is Part 3 in a series. The other parts are available in the AMSAT Journal or online at http://cubsatsim.org/papers.