Solar Panel System V2

Date: Tuesday august 8th

Documented by: Caleb Hille

Outline:

1) Scope and Purpose

2) Technical overview
    a. Hardware overview
    b. Software overview
    c. Testing and Validation Results

3) Conclusions and Future Initiatives

Introduction:

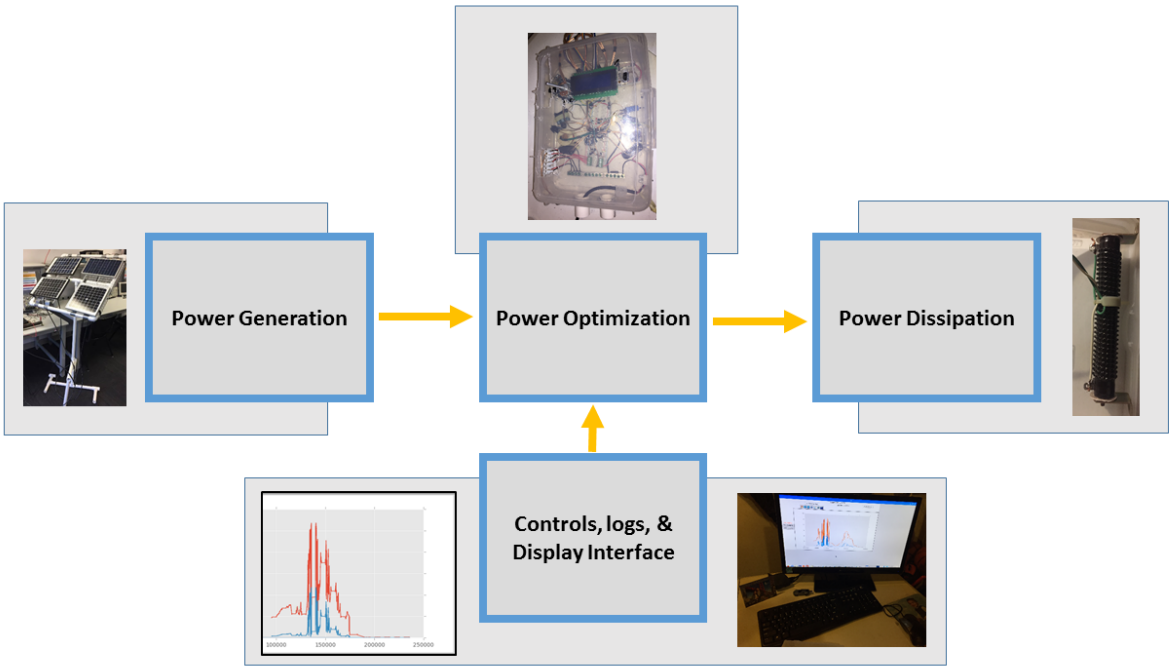
Abstract: This document outlines the work completed by Caleb Hille during the summer of 2017. The project focuses on the monitoring and management of solar power production.

Purpose: The purpose of this project was to further my knowledge in the design and implementation of solar energy systems. I also hope to be able to transition the work competed during this project to Rose-Hulman's "Engineers for a Sustainable World" (ESW) club. To expand the scope of the student designed solar technology.

Goals/Initiatives: The goals of this project were to create a DC-DC converter to optimize the power delivered by four 10 watt solar panels for a variable range of resistive loads. This initiative also included the creation of a python based control system, to allow for more complicated control algorithms, data-logging, and a graphical user-display.

Technical Overview

System Level Overview: The photo below outlines the sub-systems of this solar panel project. The system begins with solar panels as the system's source of power, which flows to a "power optimization" circuit" to be optimized for a variety of different resistive loads. The system also contains a control center consisting of an Raspberry Pi which is responsible for the controlling the power optimizer, providing a graphical user interface to the system, and logging system performance



Hardware Overview

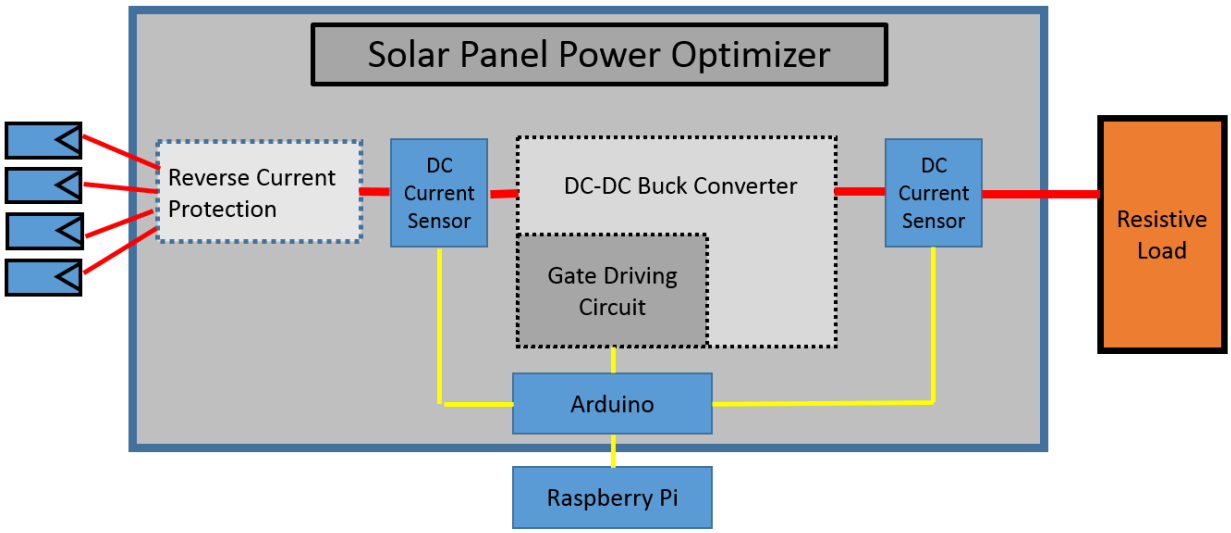
Power Generation:

1) The primary power supply was four 12V 10W solar panels, connected in a parallel configuration to deliver an output in the range of 0-22 volts, 0-3 Amps. All four panels are either from different manufacturers or are manufactured from different silicon types (mono vs polycrystalline)

2) A set of 12V wall socket plugs were also used during the testing and validation phases as simple photo-voltaic emulators. The wall plugs have similar I-V characteristics to a solar panel, and operate independently upon the sun which made them great tools to test and validate the system with.

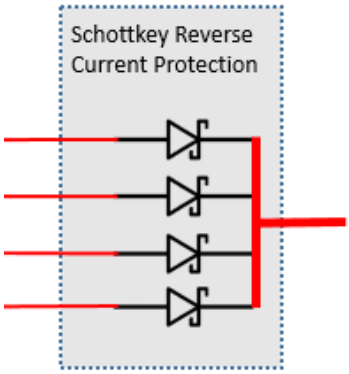
Power Optimization:

In order to be able to obtain a useable amount of power from the solar panels, the inclusion of power optimization circuitry between the panels and a resistive load was necessary to manage the voltage, current, and power flow. The power optimization circuit I designed relies on the Arduino platform, and a buck converter. An overview of the insides of the system can be viewed in the diagram below.

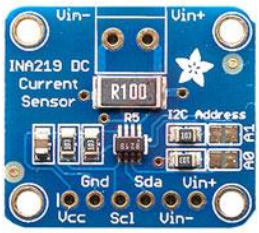


I will outline the sub components of the power optimizer below:

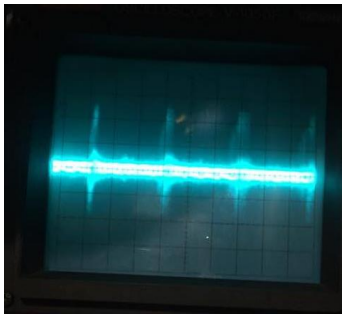
<u>Reverse current protection:</u> due to variations in panel size, manufacturer, and silicon type, and frequent shading due to cloud movement in the area, I decided to add reverse current protection to prevent one solar panel from trying to back feed another panel due to shading, ect. This is also intend to ensure that all the power flows from the panel, toward the load. The current protection circuit consists of four Schottkey barrier diodes, joining to a single HV output node. I chose these diodes for sake of simplicity and ease of implementation, for a more complicated design build I tested and contemplated the use of mosfet driving to implement reverse current protection, with the added benefit of a low-on resistance, resulting in less system power losses.



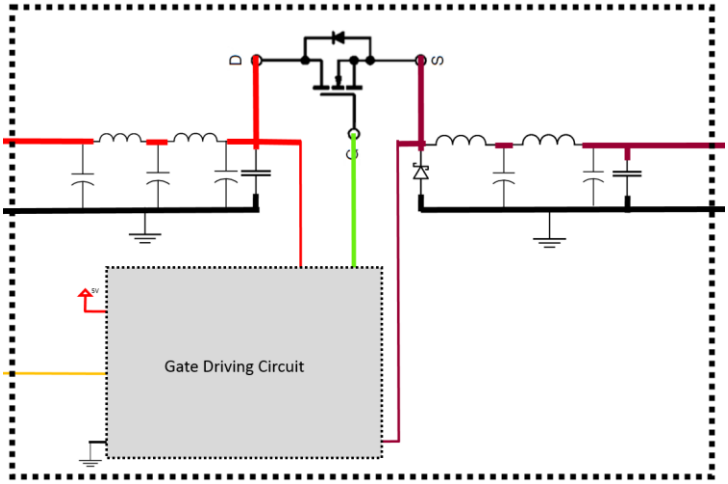
DC current sensor: Adafruit industries produces a “high-side DC current sensor” capable of measuring up to 26V DC, at 3A. I decided to choose this sensor because it was easy to implement, and its design perfectly measures signals in my region of operation.

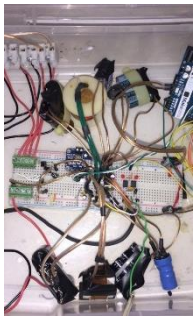


- Pros:
  - These sensors come out of the box ready to use after soldering some header pins to them.
  - They come with their own library with easy to use methods to obtain DC current and voltage measurements over and I2C bus.
  - They come with customizable I2C addresses, and therefore can have up to 4 boards per Arduino
- Cons:
  - These devices are sensitive to inductive spikes and can be fried if adequate DC filtering is not provided. I blew two of these sensors at once when I first introduced a buck converter into the circuit, without adequate DC filtering to prevent inductive spiking.
  - They are $10 a piece, so blowing a few of them can be expensive if your power optimization techniques rely on their accurate measurements.

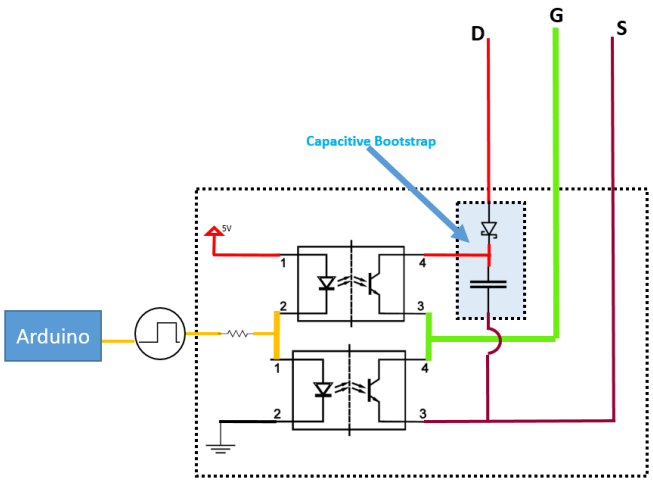


DC Buck Converter: This is my first DCDC switched mode power electronic design/implementation, so I decided to keep the power optimization circuit simple with a DC-DC step-down buck converter. My design includes some low-pass DC filtering on the input and output power lines, a single N-channel Mosfet to handle the switching, a Schottkey Barrier Diode, and some gate driving circuitry to convert a 0-5V PWM signal from an Arduino to a 0-12V PWM signal strong enough to drive the gate of the mosfet.





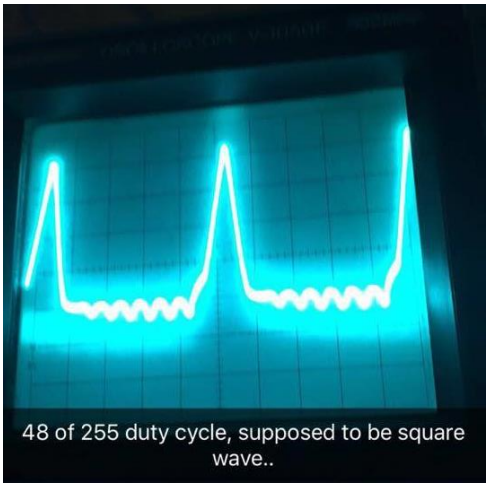
Mosfet Gate Driving Circuit: I wanted to control the buck converter using an Arduino. However, in-order to turn on, the mosfet gate requires a minimum voltage of ~9 volts. Because the Arduino operates on a 5V system, I needed separate, gate driving circuitry to boost the PWM signal from the Arduino at 5V to a 12V PWM signal.



I discovered a solution from a YouTube channel of a man named Julian Ilett. I used his circuit design of an “opto-Isolator driver” to boost the voltage of the signal sent by the Arduino, to drive the DCDC converter. The link to his video is here: https://www.youtube.com/watch?v=iNYeww1Sjzk&t=346s .

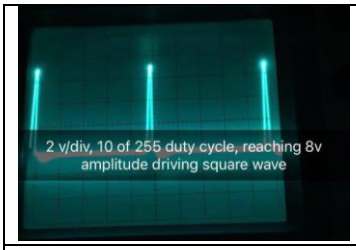
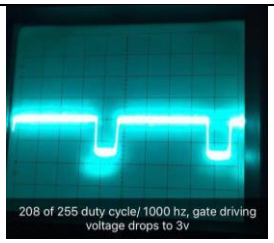
The DCDC converter had a major fault, and produced a very unstable output voltage with wide variations in power transfer over time. I believe the cause of the fault was due the gate driving circuitry. The circuitry relied on a “capacitive bootstrap” to generate the boosted voltage signal.

I noticed wild fluctuations on the output power readings, so I probed the circuit with an oscilloscope. The photo below shows the gate driving signal. The signal should be a low ~20% duty cycle PWM wave however I found this trace when probing the gate at a 30 kHz switching frequency.


48 of 255 duty cycle, supposed to be square wave..

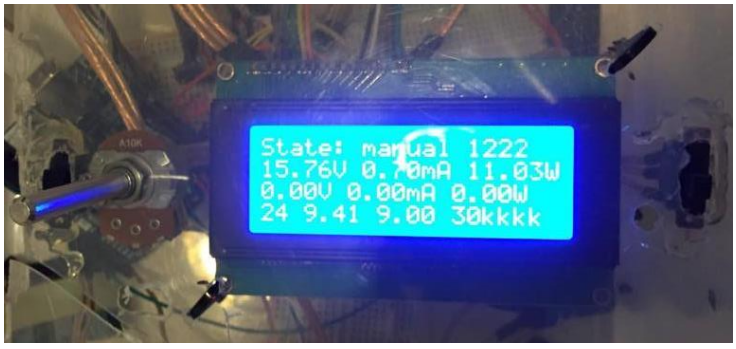
I suspected that maybe the switching frequency was too high for the optocouplers to respond to, so I lowered the frequency down to 1 kHz to inspect further. At this lower frequency, the gate driving signal did return to its intended square wave output, however I noticed that as I increased the duty cycle, the amplitude of the PWM output wave would decrease down to nearly 2V at 100% duty-cycle. This inverse relationship between PWM amplitude and duty-cycle is what caused me to suspect the capacitive boot-strap as source of the problem. The photos of the mentioned affect are shown below.

At this lower frequency, the gate was not PWM signal with a high enough voltage to conduct, and therefore at the 1000 Hz switching frequency full duty-cycle, the DCDC converter was producing ~7 mA as measured by the DC Current sensors.

		
PWM amplitude ~10V Duty-cycle = 3%	PWM amplitude=4V Duty-cycle = 17%	PWM amplitude = 3V Duty-Cycle = 80%

I abandoned the project due to the closeness to the end of the summer at this point, but the next generation power electronics of this solar system will be designed with a more “active” gate driving system to avoid the problem observed in the traces above.

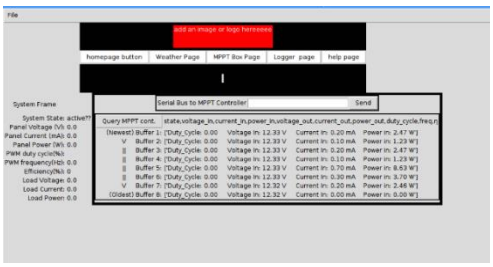
Arduino: An Arduino Uno acted as the brains of the DC-DC power optimizer, and controlled the DC current sensors, the DCDC power conversion, reported its status serially to a monitoring Raspberry Pi, and updated its instantaneous status on a 20x4 LCD display on the front of the enclosure for the entire system.

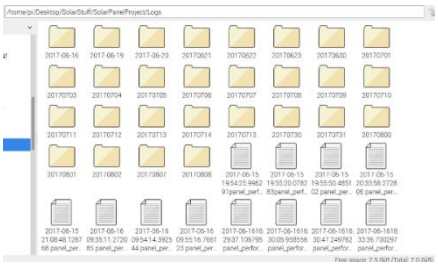


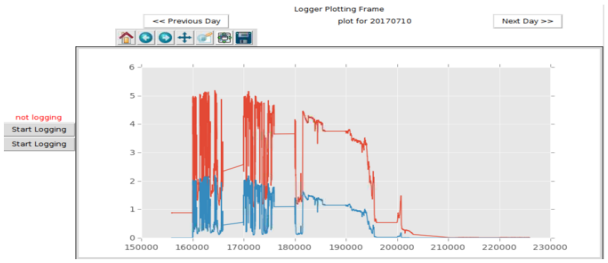
Raspberry Pi: A Raspberry Pi 2B acted as the brains of the whole operation. It served to log all the systems performance over time, provide a GUI interface, communicate with/control the DCDC optimizer circuit, and much more.

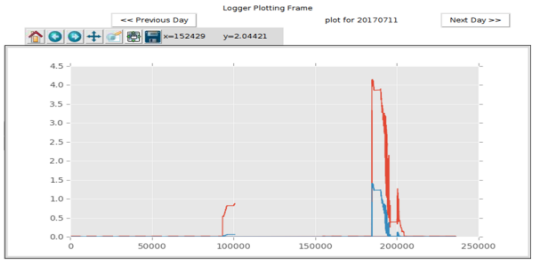
Software Overview

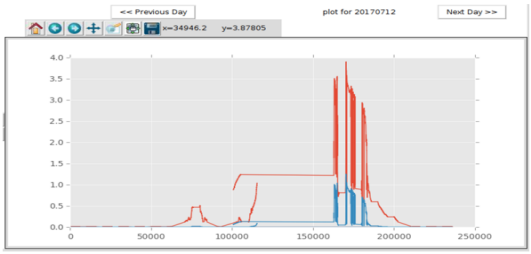
Python: The majority of the project was monitored and controlled through the use of python scripts. A series of solar_panel_system_....py scripts worked together to create a graphical user interface, that handled the control of all the subsystems such as the power-optimizer, as well as created a text-file based logging system by which the systems performance could be recorded and displayed over time. .

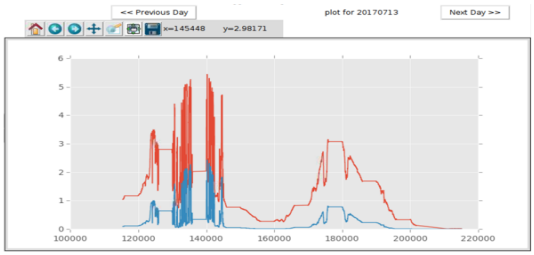




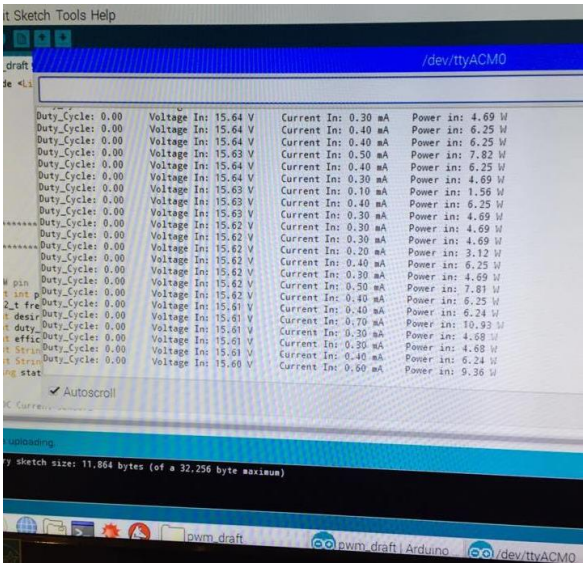




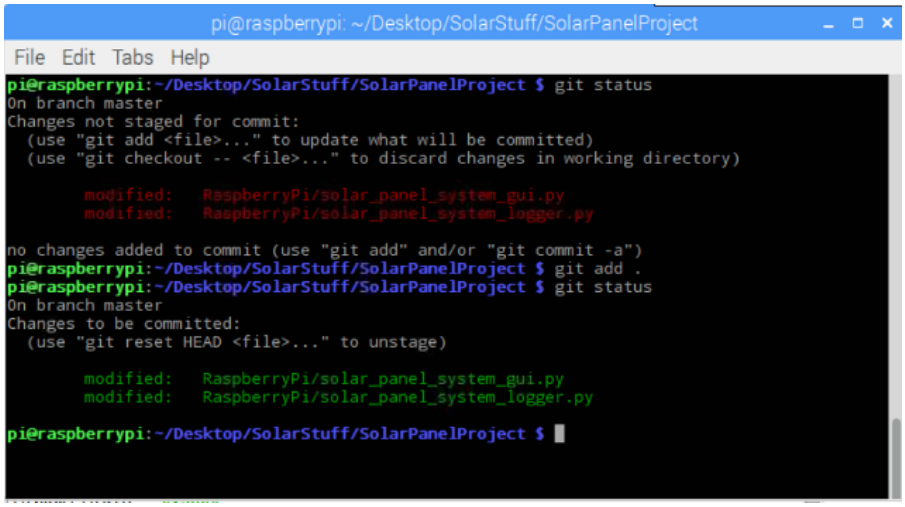




Arduino C: The DCDC converter was managed by an Arduino Uno. The code to control the Arduino was written in Arduino's variant of C. Special libraries were imported from Adafruit industries to use the Adafruit DCDC current sensors used in the DCDC converter which can be found on Adafruit's website under the Learning tab for the INA219 High Side DC current sensor section. An Arduino PWM frequency library was found on the Arduino forums page called PWM.h. This library allowed me to setup the Arduino's timers to obtain whatever desired PWM frequency I desired under ~50 kHz. This library made observing the effect that changing the switching frequency without changing the energy storage elements can have on the stability of the DCDC converters output. Here is a link to the PWM frequency library https://forum.arduino.cc/index.php?topic=117425.0. The Arduino also has nice standard serial communication functions, which made communication between the Arduino and Raspberry pi for things such as commands, telemetries, as simple as a print line statement. Below shows a photo of the Arduino printing some information it has read from the Adafruit DC current sensors on the serial bus, which could be read by the Raspberry Pi.



GIT Version control: All the software developed for this system was tracked using Git as a software version control client. The purpose of this version control was first just to practice healthy programming practices when working on an extended/extensive project and secondly to provide a solid foundation of logging on work/decisions made so that future contributors to the development of this system will be able to observe and revert to previous decisions and code states.



Results, Conclusions, and future initiatives

Hardware conclusions:

- **Solar Panels:** Small 10W 12V panels turned out to be the perfect size panels to experiment and learn about solar panels with. Their low output voltage makes them safe to work with bare handed, yet still forced me to feel the danger of high voltage by frying a few logic devices learn how to develop proper isolation in hardware between the power control and lower voltage (5V or 3.3V) logic devices.

- **Power Optimizer:** This project taught me a lot about the basics of relationships between switching frequency, HV isolation, energy storage, filtering, frequency response of components and their effects on the system, and logic control in power electronic systems.

- **Controls, logs, and User Interface:** The Arduino and Raspberry Pi were great devices with a lot of supporting documentation and hardware and code resources to make the implementation of this project very easy. It was a great "platform" to stand on that made focusing on the solar aspect of things, rather than complicated ways to implement the things needed.

- **Power Dissipation:** Experimenting with different Resistive loads gave me a good intuitive feeling for the act of power-transfer works, and the relationship between voltage, current, and power in DCDC conversion, as well as the types of filtering and energy-storage in the DCDC converter itself. I only worked with resistive loads during this project, but it was enough to inspire me to be ready to incorporate a battery management system into the next stage of the project, to begin trying to incorporate energy storage into the system for 24/7 operation capabilities.

Software conclusions

- **Python:** I was able to develop intermediate python skills during this project and become good at incorporating other modules into projects that I use. This included the creation of my own module that worked together as a system to produce a desired functionality, managing many sub-systems at once.

  I also learned to create and work with file systems and really primitive data management and storage.

  I also learned how to work with a system with a primitive communication protocol (serial bus) to communicate telemetries and system status to and from a master computer.

- **Version Control:** For the first time I really implemented version control on my own, tracking major code changes. An area to improve on would be committing more often, so that my commits only track small/individual changes rather than massively new updates to every file.

Future initiatives

- The next stage of the project I would like to pursue producing a true maximum power point tracking device to optimize the power delivered form the solar panels, as well as a battery management device to incorporate energy storage into the system.

Completed on: August 10, 2017

Completed by: Caleb Hille
