Battery Matrix Management System

Technical Report

MCS4833 Senior Project 1

Calvin Withun

Adviser: Dr. CJ Chung

Abstract

This project is an attempt at creating both a battery management system and the battery circuit which it manages, referred to as a Battery Matrix Management System (BMMS) and Battery Matrix (BM), respectively. Such a system has the ability to optimize the lifetime of batteries, help to shield homes or particular devices against blackouts and brownouts, and potentially save on the cost of electricity if utilized intelligently. This project involves knowledge of electrical components to design the circuit diagram for the BM as well as programming knowledge to write the BMMS software. Most effort went towards developing the software since this project was developed in a software-oriented Senior Project course, but that software was created with the eventual hardware in mind. Some problems came up concerning program robustness, but every test case which was defined has been met and satisfied at this point in the project's development, which constitutes a successful project.

Introduction

In an increasingly technological world, it is critical that people have efficient means of storing and distributing power for everyday tasks. One idea gaining popularity is the battery management system, a system which facilitates the behavior of a collection of batteries. This project has the goal of designing a battery management system which will facilitate battery charge, battery discharge, and dynamically balance battery voltages. This project, in addition to designing software to accomplish this goal, will include a specific circuit which the software is designed to run. This circuit will be referred to as a Battery Matrix (BM), thus leading the software to be referred to as the Battery Matrix Management System (BMMS). The BMMS is written in Python, and is designed to be run on a Raspberry Pi.

Background – Batteries

Two batteries, connected in series, act as a single battery whose voltage is the sum of both individual batteries. For example, battery A with a voltage of 3 volts, when connected in series with battery B which also has 3 volts, creates a virtual battery C, whose voltage is 3 + 3 = 6 volts. This is a simple method for reaching high voltages, and as such the BM utilizes serial batteries to provide power. Under ideal circumstances, both of these batteries will charge and discharge at identical rates while connected in series, causing their voltages to remain identical. However, subtle differences from things such as temperature and specific chemical composition and previous battery use can cause two serial batteries to become desynchronized, allowing one to charge or discharge more rapidly than the other. Over time, this can lead to substantial differences in voltages, which will cause the batteries' lifetimes to decrease faster than normal.

This issue can be fixed if two desynchronized serial batteries are instead connected in parallel. When two batteries are connected in parallel, the higher-voltage battery will discharge into the lower-voltage battery until they reach an equilibrium and resynchronize with each other. However, the voltage of two parallel batteries is different from two serial batteries, so making this change while keeping all batteries acting as a power supply will cause the power supply voltage to fluctuate, which is undesirable. For this reason, the BM is designed so that any battery can be serially connected to its neighbors or can be connected in parallel to any other batteries such that parallel and serial batteries do not interact with each other, and only serial batteries provide power.

Background – Hardware

In order for the BM to achieve its desired functionality, it must have a series of switches which connect each battery to either the serial or parallel track, but not both. Computer-powered switches come in a variety of forms; the ones utilized for this project are MOSFETs and relays. MOSFETs come in two variety: P-type and N-type. P-type MOSFETs, without receiving a control voltage, acts as a closed switch, while N-type MOSFETs act as an open switch. Relays come in many designs, the most relevant of which to this project is the DPST (double-pole single-throw) NO (normally-open) relay. This relay facilitates two separate circuit connections, and unlike MOSFETS, can transfer charge in either direction, making it ideal for controlling each battery's parallel track connection since two parallel batteries could attempt to equalize in either direction.

The BM must also provide a means for the BMMS to measure voltages across each battery, otherwise the BMMS will be blind. One would typically use a multimeter / voltmeter when measuring voltage by hand, but a computer must use other means to perform the task. This project makes use of analog-to-digital (ADC) converters to inform the BMMS of the voltages across batteries. The particular model used allows for 8 measurements to be taken at a time, although this may not be a usable solution to the problem of multiple batteries (detailed under Problems).

Test Results

SW Feature Name	Feature Description	Expected Behavior/Output	Self-Testing Results	Notes
Battery Voltage Monitoring	A feature which measures the voltage across each individual battery	The BMMS shall detect an accurate, real-time voltage across any electric potential	Demonstrates expected behavior for 1 battery, but may not work for 2+ batteries (detailed under Problems)	
Battery Circuit Monitoring	A feature which tracks which circuit (serial, parallel, or disconnected) each battery is on	The BMMS shall be aware of which circuit each battery is connected to at all times	Demonstrates expected behavior	
Circuit Facilitation	A feature which controls what circuit a battery is connected to	The BMMS shall always move the highest- and lowest-voltage batteries onto the parallel circuit, while keeping all others on the serial circuit	Demonstrates expected behavior	
Event Logger	A feature which records all BMMS activity as well as all detected events	The BMMS logs shall contain entries for every time batteries are moved between circuits, every	Demonstrates expected behavior	

		time a blackout or brownout occurs, every time the BMMS enters or exits the desired charging window, etc.		
User Notifications	A feature which notifies the user whenever relevant events occur such as errors, blackouts, battery disconnections, etc.	The BMMS shall display notifications for its user on a Notifications tab whenever relevant events occur	Demonstrates expected behavior	
Config Manager	A feature which allows the user to modify the rules regulating the behavior of the BMMS	The BMMS shall have an interface for viewing and editing the BMMS settings	Demonstrates expected behavior, but is not robust and can be used to crash the program (detailed under Problems)	
Restrictive Charging	A feature which manages the conditions under which the BMMS may charge its Battery Matrices	The BMMS will not attempt to charge outside of its desired charging hours, unless it is set to charge outside of the desired charging hours	Demonstrates partial functionality (detailed under Problems)	

		during low charge. The BMMS shall attempt to charge during desired charging hours, but will not exceed the maximum tolerated voltage		
Time Display	A feature which displays the current time	The BMMS shall display the current time to the user	Demonstrates expected behavior	
Graphical Display	A feature which displays information to the user	The BMMS shall display information to a screen for the user to view such as time, battery voltages, projected battery life, etc.	Demonstrates functional behavior, but is not particularly aesthetically appealing due to inexperience with utilizing Python's tkinter library	
Battery Damage Detection	A feature which can detect when a battery is failing	The BMMS shall notify the user whenever a battery drops below performance voltage and recommend replacing it	Demonstrates expected behavior	

Problems

This project encountered a variety of roadblocks and problems, some of which were resolved and some of which were not. First and most foremost is the issue of the BMMS not having a viable BM to operate yet. While a conceptual diagram for the BM has been designed, there was an inadequate understanding of electrical engineering principles to construct a functional circuit from the concept. The current understanding of the issue is that the batteries being used in the test setup provided an insufficient voltage to properly power the MOSFETs being used, causing the MOSFETs to never properly open circuits. As such, two batteries could never be isolated on the parallel track, but were always connected to the serial track. It would be interesting to use more powerful batteries or less powerful MOSFETs in a future attempt and see if the circuit concept will work.

A second issue with the project is the ADC being used to measure voltages across batteries. The test setup used for development has the ADC connected to a ground and to the 5V pin of the Raspberry Pi for defining the voltage range. However, this will likely cause complications if directly translated to the BM. The ground can be connected to some common ground, but if the maximum voltage line is connected to any battery, the subsequent battery risks going beyond the voltage range provided, causing the ADC to not report accurate values. It may be necessary in the future to designate one ADC per battery to avoid this issue, or further investigate the function of the ADC in use to better understand how to connect multiple batteries to it at once in the desired way.

A third complication came from the restrictive charging feature. The feature currently suffers from a lack of robustness; it can successfully facilitate charging schedules where the charging window begins and then ends on the same calendar date, but it cannot facilitate a schedule which begins in the evening of one calendar date and ends in the morning of the next calendar date. For example, a charging window of 8:00am to 4:00pm can be handled by the BMMS, but a charging window of 8:00pm to 4:00am cannot be handled. The BMMS will never charge the system under such a charging schedule. This test case was not considered during development, and consequently was only discovered later on.

Dependencies

This project only has one external dependency, which is a library for communicating with the ADC currently being used. This library may be downloaded by running the following command in a command shell:

```
$ sudo pip3 install adafruit-circuitpython-mcp3xxx
```

Conclusion

All things considered, this project can be essentially considered a success. The BM is not available, though that issue is more focused on electrical engineering, which is not the primary focus of the course which facilitated this project. The BMMS software is functional despite not having a BM to drive, and is essentially prepared to drive one should one be made available to it. The BMMS performs all of its intended functions accurately, though there are some robustness issues which make it prone to crashes if misused. This, combined with a circuit concept diagram for the BM, makes for a promising project to continue developing in the future. Certain elements of the software, such as events for logging and notifications, are easily expandable for new types of events (such as brownouts or blackouts), meaning that the framework of the program is conducive to further development.

It is recommended that further development on the BMMS be postponed until a functioning BM has been constructed, since all progress in the BMMS is only hypothetically useful without a BM to verify its behavior. Unfortunately, since this task focuses on electrical engineering rather than software development, the author of this project is unsure whether he will continue in Senior Project 2 or begin a new project.

Images

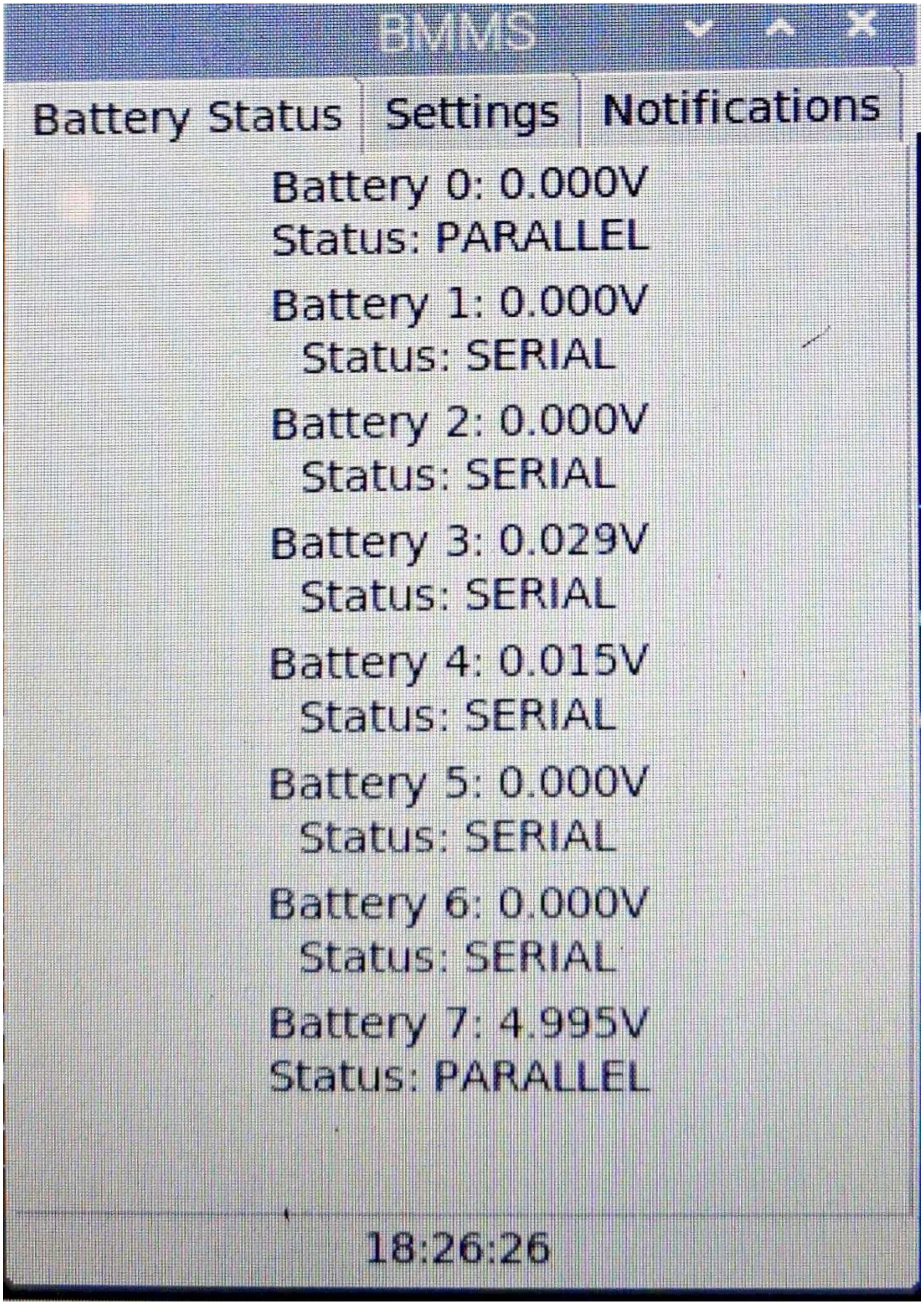

Image 1: Battery Status Tab. Each battery has a voltage display and a status display to indicate which circuit it is currently on. The end-product formatting will look nicer than this display.

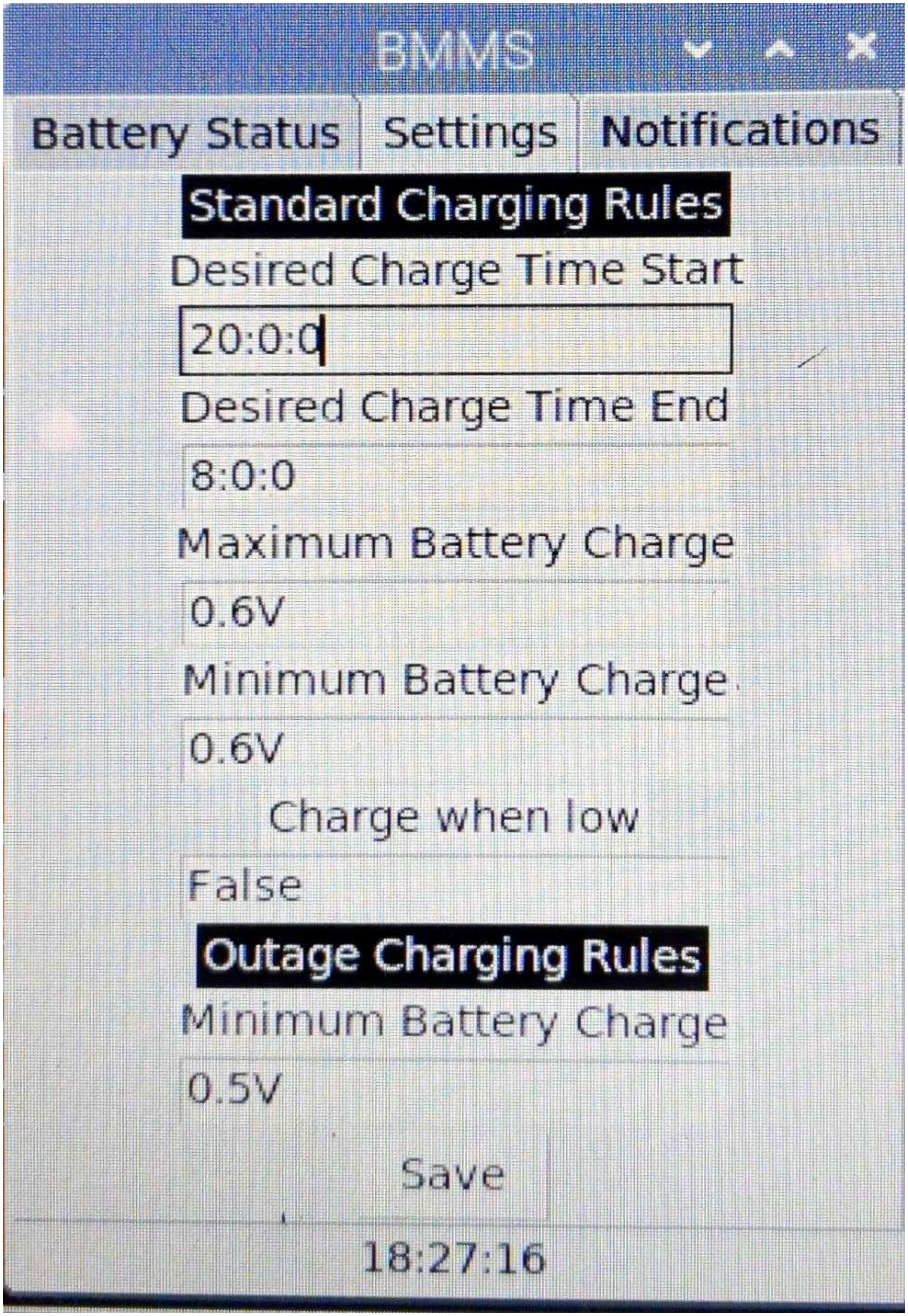

Image 2: Settings Tab. Each configurable setting appears on this tab and is editable by the user. The end-product formatting will look nicer than this display.

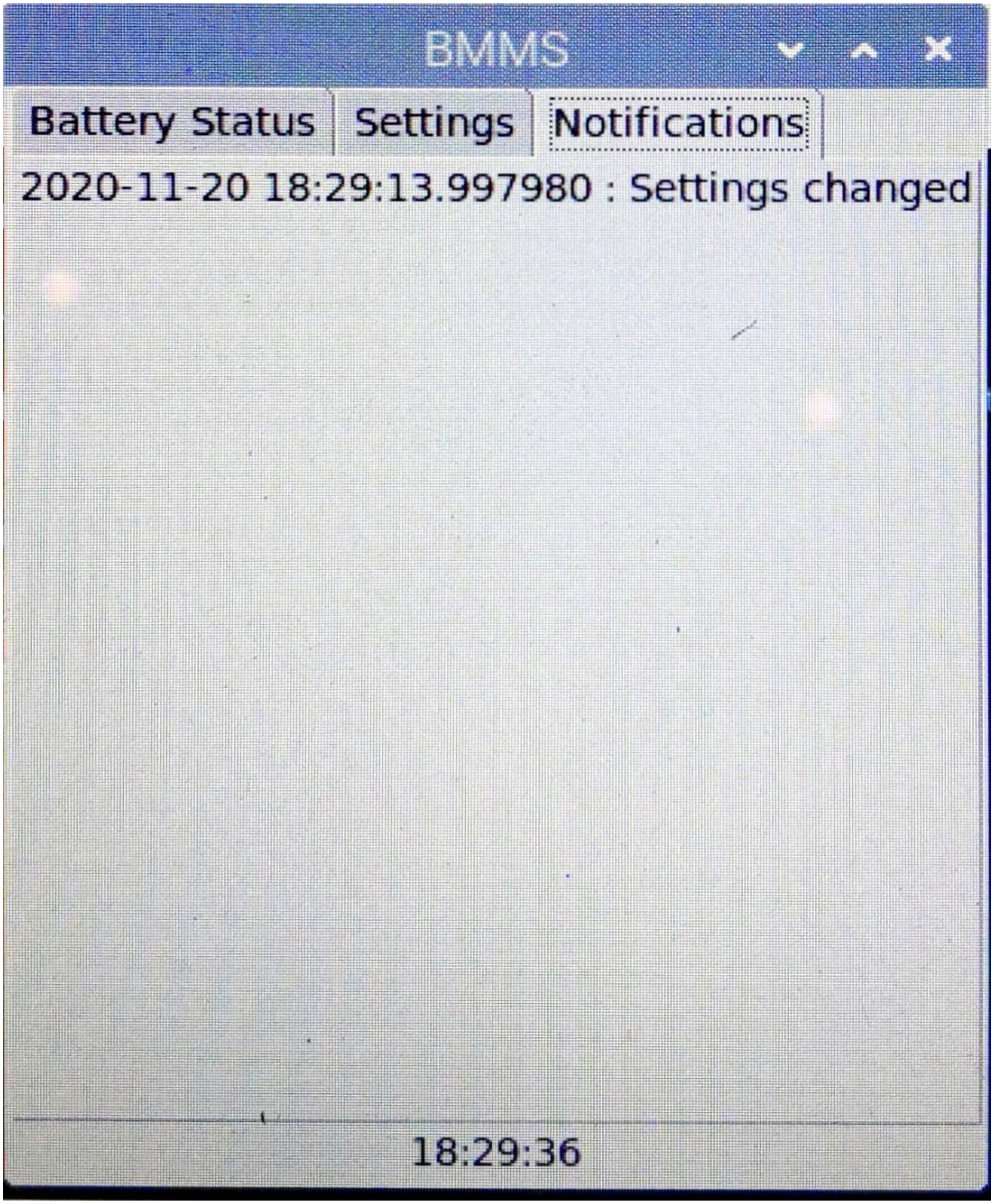

Image 3: Notifications Tab. This tab displays all notifications that are generated for the user. The end-product display will look neater, and will have buttons for dismissing each notification so that the notifications window does not fill up.

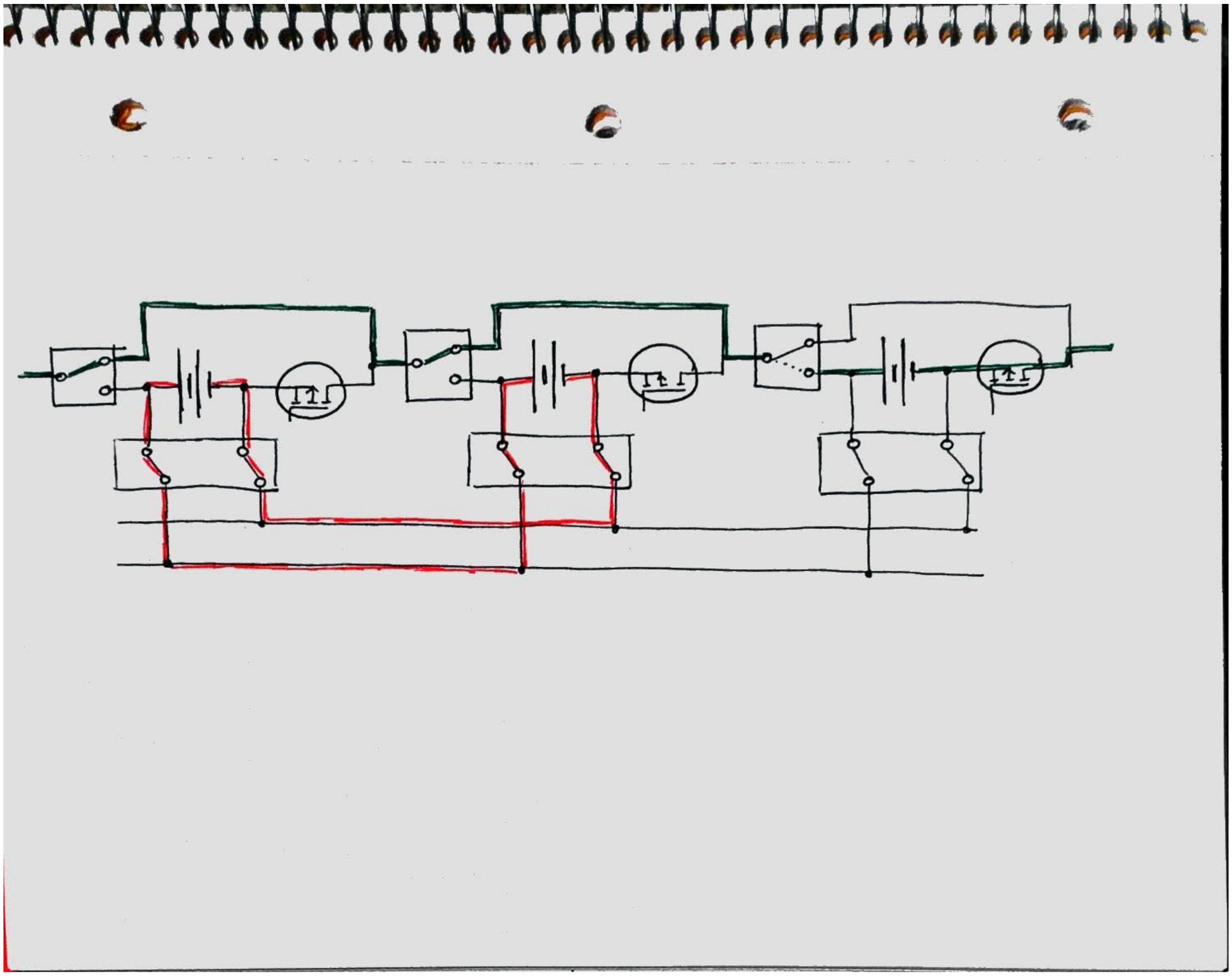

Image 4: Modular Battery Matrix Circuit Example. The green line indicates the serial circuit, and the red line indicates the parallel circuit. The prototype model for this circuit will utilize a total of eight batteries. Some circuitry not shown in this image as it is still being designed. NOTE: the SPDT relay, in the final BM design, most likely should be replaced with two opposite MOSFETs as MOSFETs are not mechanical and thus are faster to respond than relays.

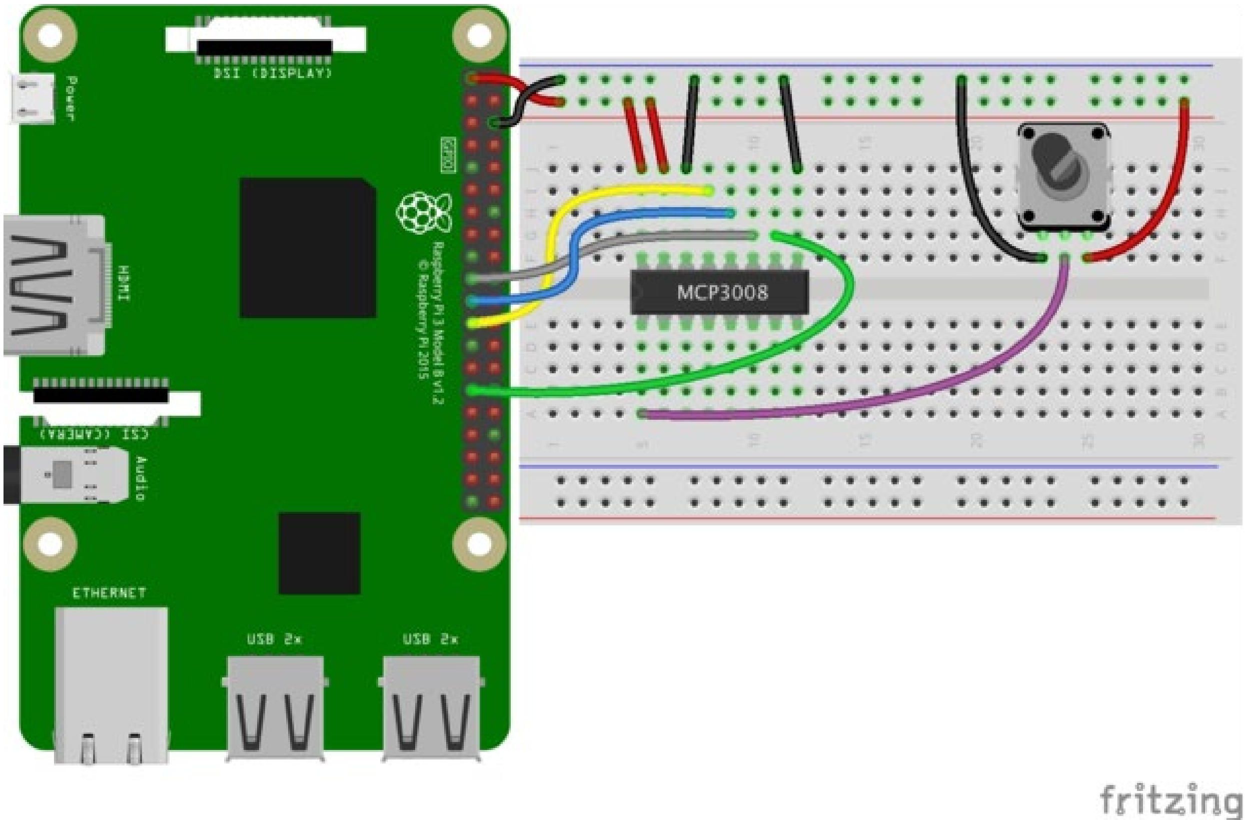

Image 5: Testing Setup for Battery Voltage Monitoring. While viable for a single battery, it may not be viable for measuring multiple batteries in serial.
Image taken from https://learn.adafruit.com/mcp3008-spi-adc/python-circuitpython

References

https://learn.adafruit.com/mcp3008-spi-adc/python-circuitpython