Battery Cycling Algorithm for Laptop Power System

A Major Qualifying Project

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

By

Huy Cao

Jaeyong Oh

Liam Wolozny

Date: May 5th, 2021

Approved by:

Professor Stephen Bitar, Advisor

Abstract

Battery-powered devices that are plugged in most of the time without proper management result in shortened battery life when running on their own power, which can be costly on the customer side and potentially a hazard to the environment. This project proposes a battery cycling algorithm aiming at laptop power system application that could potentially improve battery life for such devices. A proof-of-concept prototype with discrete battery cells was designed to demonstrate the proposed algorithm. DC-DC converters and microcontroller circuits were successfully assembled and tested. Future work is needed to assemble and integrate the battery switching circuit, verify the improvement to battery life, as well as integrate this system into a laptop designed for commercial production.

Acknowledgements

We would like to thank our project advisor, Professor Stephen Bitar, for his advice, guidance on this project and his continuous motivation for us to accomplish this project. We also want to thank Mr. William Appleyard from the ECE Shop for his help with parts ordering, PCB soldering and debugging during the testing phase of our project. Without them, this project would have not been possible.

Authorship

Section	Primary Author
1. Introduction	Huy, Liam
2. Background Research	Huy, Jaeyong
2.1 Type of Rechargeable Battery	Jaeyong
2.1.1 Lithium-Ion (Li-Ion) Batteries	Jaeyong
2.1.2 Nickel-Based Batteries	Jaeyong
2.1.3 Lead-Acid Batteries	Huy
2.2 Battery Charging Methods	Huy
2.2.1 Constant Current Charging	Huy
2.2.2 Constant Voltage Charging	Huy
2.2.3 Constant Current – Constant Voltage charging	Huy
2.2.4 Multi-stage constant-current charging (MCC)	Huy
2.3 Electrical Switches	Huy, Jaeyong
2.3.1 Solid-State Relay (SSR)	Jaeyong
2.3.2 Electromechanical Relay (EMR)	Jaeyong
2.3.3 Other types of relays	Jaeyong
2.3.4 Diodes	Huy
2.3.5 Metal Oxide Silicon Field Effect Transistor (MOSFET)	Huy
2.3.6 Bipolar Junction Transistor (BJT)	Huy
2.4 Battery State of Charge Estimation	Jaeyong
2.4.1 Coulomb Counting Method	Jaeyong
2.4.2 Kalman Filter Method	Jaeyong
2.4.3 Enhanced Coulomb Counting Method	Jaeyong
2.5 Unbalanced Battery Cell	Jaeyong
2.6 Laptop Power System	Huy
2.7 Shielding for Current Sensor	Liam
3. Project Objectives and Scopes	Huy, Liam
3.1 Project Scope	Huy
3.2 Project Objectives	Huy
4. Methodology & Approach	Jaeyong, Liam
4.1 Battery Cycling Algorithm	Liam
4.2 Chosen Battery State of Charge Estimation Algorithm	Jaeyong
5. System Design and Implementation	Huy, Jaeyong, Liam
5.1 Top Level System Block Diagram	Huy
5.2 Detailed Module Description	Huy, Jaeyong, Liam
5.2.1 DC-DC converter/regulator stages	Jaeyong
5.2.2 Battery Bank	Liam
5.2.3 Battery Charging IC	Huy
5.2.4 Switches	Huy
5.2.5.1 Measuring Voltage	Huy
5.2.5.2 Measuring Current	Liam
5.2.6 Microcontroller Unit (MCU)	Huy
5.2.7 Load Stage	Liam
5.3 System Implementation	Huy, Jaeyong, Liam
5.3.1 Implementation plan	Huy
5.3.2 PCB Design of the DC-DC Converters and MCU	Jaeyong
5.3.2.1 Component Placement in PCB Design	Jaeyong

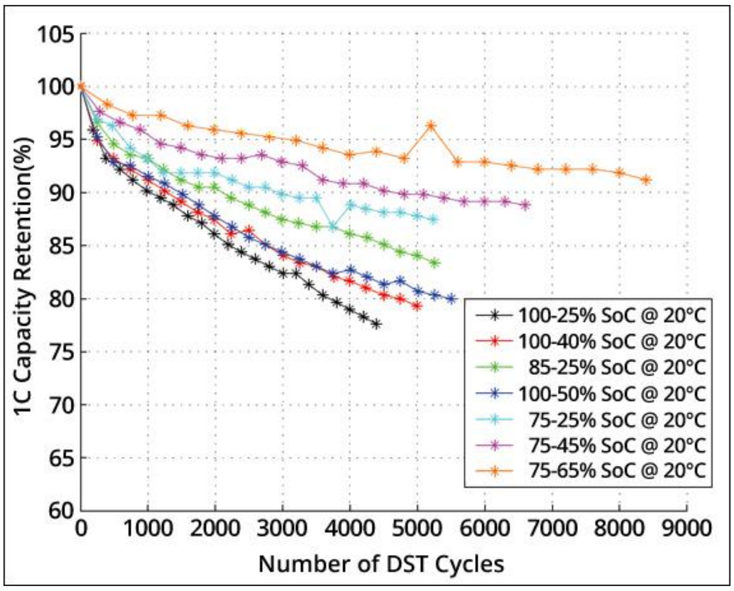
5.3.2.2 Tracing in PCB Design	Jaeyong
5.3.3 PCB Design of the Battery Switching Circuit	Jaeyong
5.3.4 PCB Design of the Battery Bank	Jaeyong
5.3.5 MCU Programming	Huy
6. System testing and result	Huy, Jaeyong
6.1 DC-DC Converters Testing	Jaeyong
6.2 Microcontroller Circuit Testing	Huy
6.3 Load Stage Testing	Liam
7. Future Recommendation	Huy, Jaeyong, Liam
7.1 Functionality testing of the Batter Switching and Battery Holder PCBs	Huy
7.2 Cell Balancing Circuit	Jaeyong
7.3 Battery Life Testing	Liam
7.4 Protective mechanism for the prototype	Huy
7.5 System integration into laptop	Huy
7.6 Other rechargeable application implementation	Liam
8. Conclusion	Huy
9. Reference	Huy, Jaeyong, Liam
Appendix	Huy

Executive Summary

As time progresses, the society becomes more and more dependent on technology, especially the battery-powered devices such as smartphones and laptops. This was proven to be true especially during the global pandemic situation when these devices appear to be important for the sake of work or entertainment. That said, prolonging the usage time of these devices by improving the lifespan of the battery has become an utmost importance matter. Though batteries always run out at the end, it is possible to achieve longer active duration of a battery before it runs out of charge through correct management of battery charging process. However, this is generally not the case for many battery-powered systems or devices that spend a significant amount of their operating time powered by an outlet. Without efficient management and battery charging algorithms, the batteries of such devices stay in a constant state of charging. When these devices are then disconnected from their outlet and are made to run on their own batteries, they diverge from their original battery life specifications, which indicates that the batteries have lost most of their initial charging capabilities. Eventually, not only they become costly for customers, who must frequently spend money on replacing and repairing these batteries, but also negatively impact the environment when they are discarded.

The goal of this project is to propose a battery management algorithm that would manipulate the charging cycles and maximize the lifespan of the battery. Given the fact that it is a common application of the previously described situation, the proposed algorithm focuses on the laptop power system applications. Additionally, this project aims to design, assemble and test a prototype served as the proof-of-concept for the proposed algorithm. This prototype consists of a battery bank with 4 Li-Ion cells in the configuration of 2 series cells connect in parallel with another series branch with 2 cells.

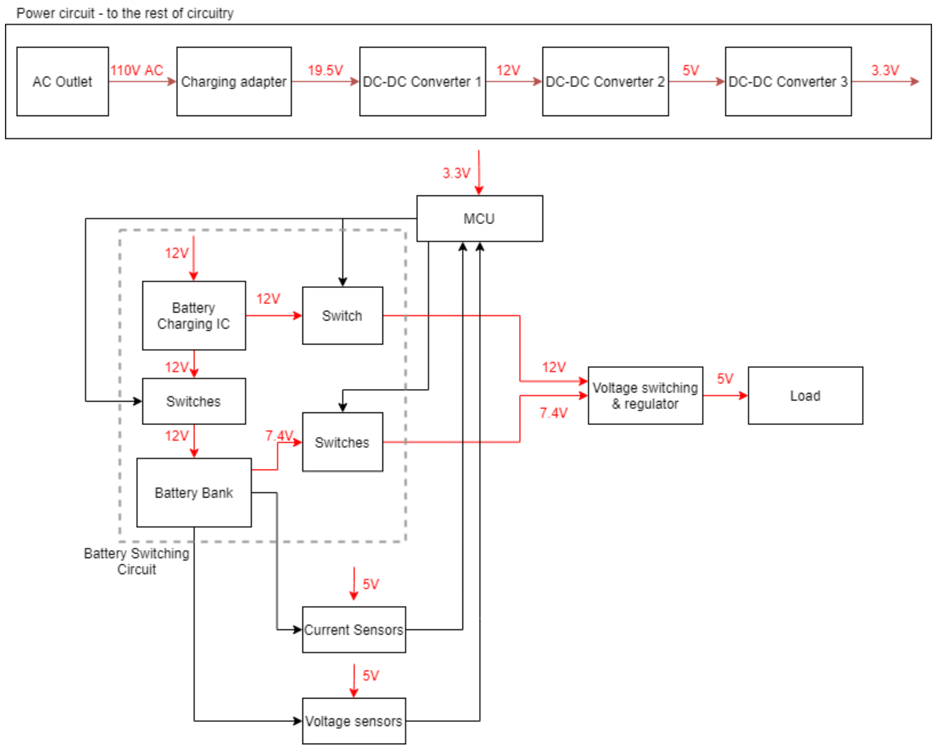
Battery life is subject to deterioration from the moment that it is first manufactured. Charging the battery from different states of discharge to different states of charge can have a marked effect on battery life. This can be seen in the figure below which shows the overall capacity decay of a Li-Ion battery with respect to the number of charge-discharge cycle for different amounts.



Dynamic Stress Test of a Lithium-Ion Battery (Xu et al., 2016)

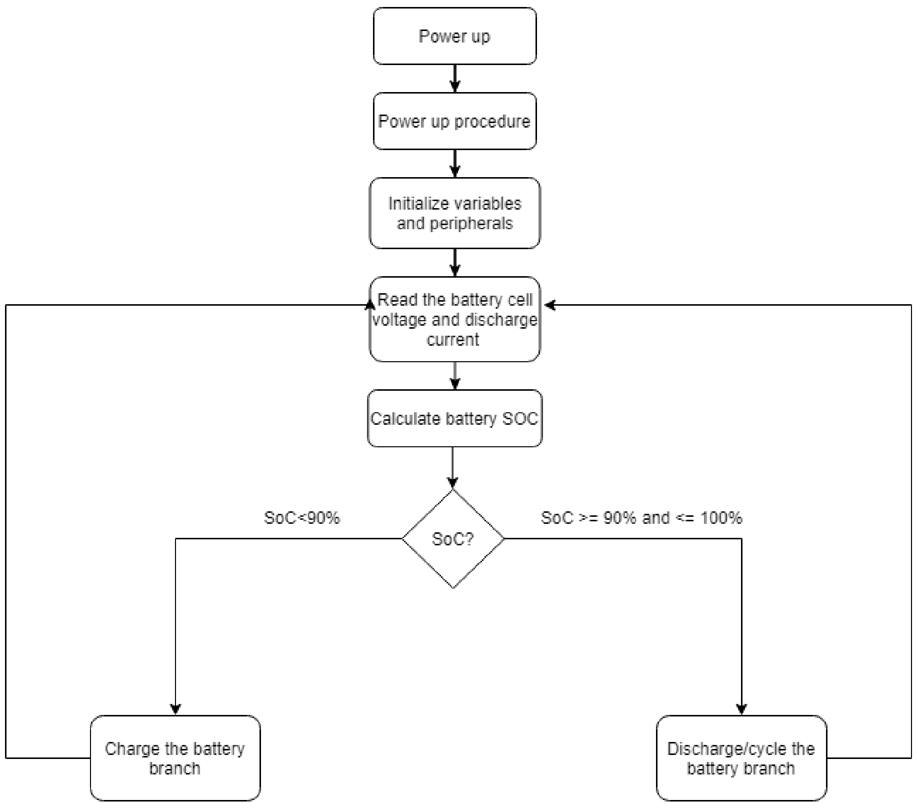
As seen in the above figure, the most effective battery cycling range at maintaining a battery at high max capacity for the longest amount of time is from 65%-75%, while the worst range is from 100%-25%. Given that our battery must be capable of prolonged use while unplugged, ¾ of max charge is not enough and the laptop user would not be satisfied. Therefore, even though fully charging the battery will be harmful in the long run, the proposed algorithm uses a battery cycle of 90-100% as we believe it is a necessary tradeoff for increased disconnected operating time. The complete algorithm operates as follow: when the battery pack is fully charged, a battery branch in the bank would be discharged while the other branches are kept charging. When this battery branch reaches a State of Charge level of 90%, it will be put back to the charging power and the other battery branch will be discharged to the same State of Charge level. The process will be repeated until the charging state is disrupted, or the device is unplugged from the charger.

Below is the top-level system block diagram for the proof-of-concept prototype. As we target laptop power system application, this system mimics the same power system inside the laptop with the addition of the battery switching circuit, microcontroller and the voltage and current sensors to perform the proposed cycling algorithm.



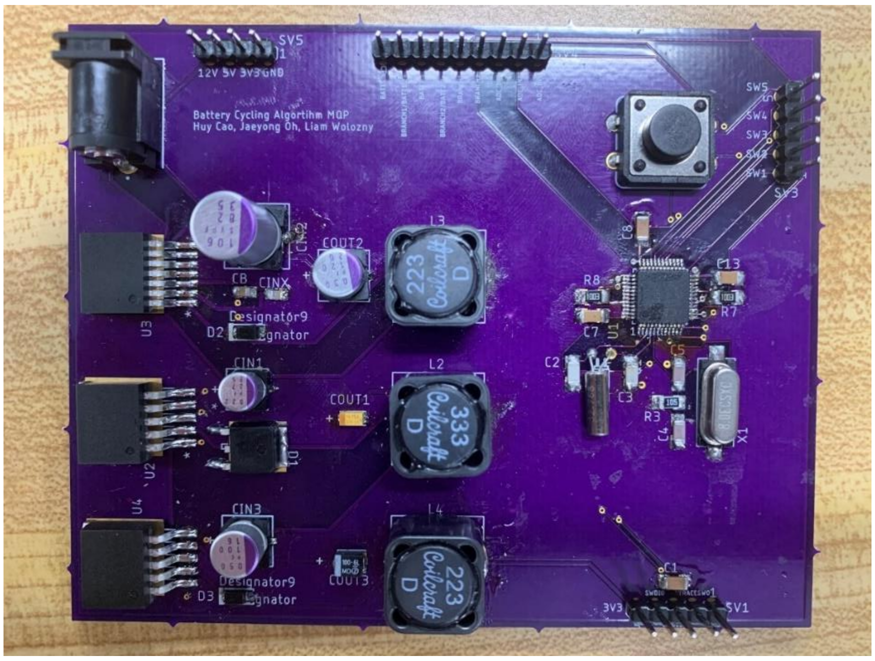
Top-level system block diagram

The microcontroller is where the functions to calculate the battery State of Charge and to control the battery switching circuit are implemented. The flowchart for the microcontroller program is shown below. The program described in the flowchart was implemented to ST Microelectronics' STM32F103C8T6 microcontroller. The code for this program was developed using the Keil µVision® 5 IDE for code compiling and uploading, and STM32CubeMX for hardware code initialization.


Power up
Power up procedure
Initialize variables and peripherals
Read the battery cell voltage and discharge current
Calculate battery SOC
SoC?
SoC<90%
SoC >= 90% and <= 100%
Charge the battery branch
Discharge/cycle the battery branch

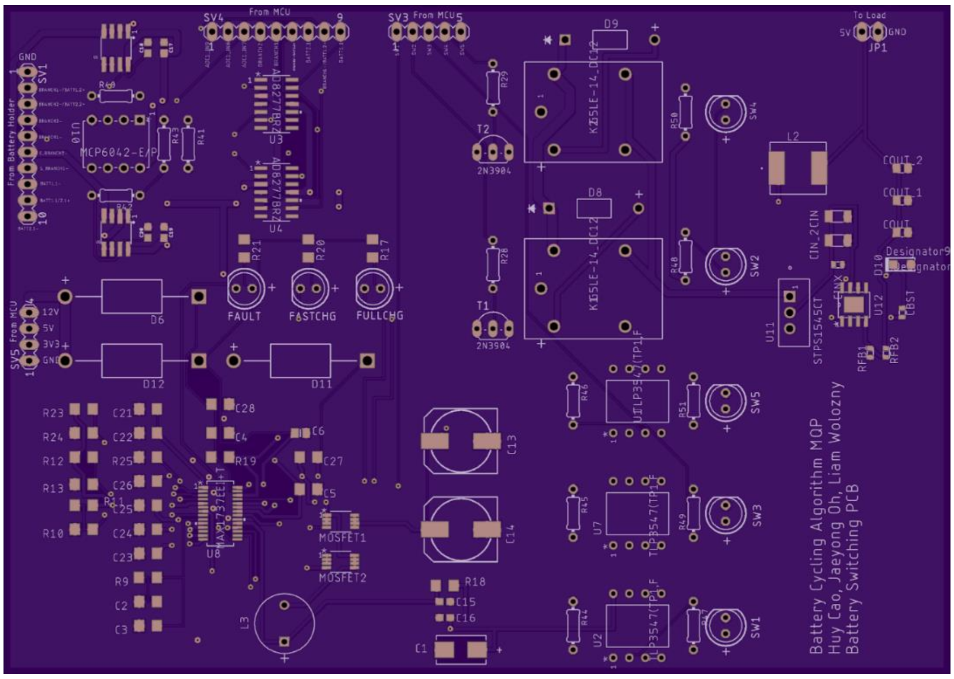
Microcontroller Program Flowchart

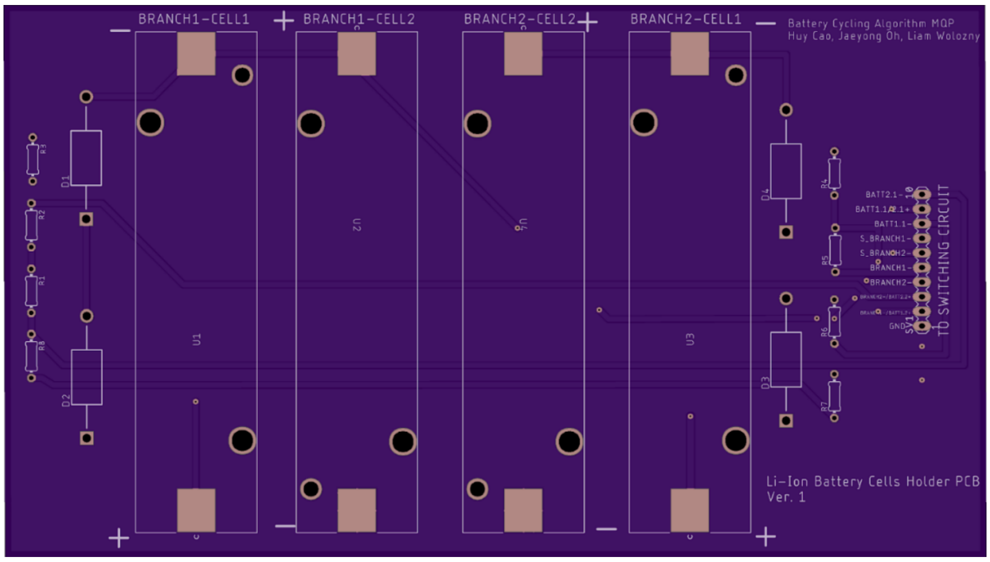
The proof-of-concept prototype was implemented using 3 different Printed Circuit Boards (PCBs). Those are the board for the DC-DC converters and the microcontroller circuit, the board for the battery switching circuit, and finally the board for the battery bank. The PCB having the DC-DC converters and the microcontroller was successfully assembled. The complete board of the DC-DC converters and the MCU is shown below.


Battery Cycling Algortihm MQP
Huy Cao, Jaeyong Oh, Liam Wolozny
12V 5V 3V3 GND

Assembled board of the DC-DC converters and microcontroller

On the other hand, although the PCB design was completed the manufacturing of the 2 other boards was significantly delayed due to the disruption caused by the global pandemic. The figures below show the preview of the 2 boards provided by the manufacturer.


Preview of the Battery Switching Circuit Board (OSH Park, 2021)


Preview of the Battery Holder Circuit Board (OSH Park, 2021)

The functionality of the DC-DC converters and the microcontroller circuit was successfully tested. For the DC-DC converters, the output voltage of the 12V, 5V, 3.3V converters was measured to be 11.95 V, 5.005V, and 3.26 V, respectively. For the microcontroller circuit, first the communication between the laptop and the microcontroller PCB through JTAG/SWD serial debugging was tested. The debug setting menu in Keil uVision 5 recognized the MCU as an ARM core along with its serial code, signaling a successful communication between the software and the MCU through the JTAG/SWD debugging pins. Next, the hardware initialization code for the MCU was run in debug mode, and the result showed no Error Handling breakpoint being run into, showing that the connection of hardware and peripherals was correct. Finally, the physical connection on the board of the selected GPIO and ADC pins was verified with a testbench code, with the result showing the correct representation of the output signal for each pin being monitored. The performance of the written microcontroller was also tested using this board, with the ADC readings in the voltage range of interests showing an acceptable deviation from the standard reference measurement of 1.5% to 3.5%.

Future work is needed to finalize the goals of this project. First, the two circuit boards for the battery switching circuit and the battery holder needs to be assembled, tested and integrated into a complete system with the existing board. Second, some features such as the cell balancing circuit and protective mechanism for the supply rails should be implemented to further perfect the prototype. Third, a battery life testing methodology needs to be developed and performed for this system. Finally, the possible implementation of this algorithm into an actual laptop as well as other rechargeable battery applications should be looked into.

Overall, the team has been able to accomplish most of the project objectives laid out in the beginning. We as a team were able to research and propose an algorithm to cycle the laptop battery branches in the State of Charge range of 100%-90%. On top of that, a battery switching circuit, along with a microcontroller-based control circuit to carry out this algorithm, was successfully designed. In addition, printed circuit boards for the battery switching circuit and the control circuit were successfully printed out and assembled to meet the goal of building a proof-of-concept prototype for our proposed algorithm and serve as the deliverables for this project. However, because of the time constraints and resource limitations due to the global pandemic situation, the team was not able to carry out the full functionality testing of the whole prototype. Further work is necessary to reach the ultimate goal of this project, as well as enable a possible integration of this algorithm into laptops designed for commercial production.

Table of Contents

1. Introduction

As time progresses, society becomes more and more closely interconnected with and dependent upon technology. Be it used to catch up with the news, to accomplish tasks across the internet, or to notify first responders in the event of an emergency, it is impossible to deny the important role that technology plays in the world. But what happens when this technology is not available to us? When the batteries run out, or become dead from poor use, there is little else that can act as a substitute for a failing modern appliance in case of an emergency.

Though batteries will eventually run out, it is possible to achieve longer active duration of a battery before it runs out of charge through correct management of battery charging processes. However, this is not generally the case for many battery-powered systems or devices. For instance, some devices spend a significant amount of their operating time powered by an outlet. Without efficient management and battery charging algorithms, the batteries of such devices stay in a constant state of charging. When these devices are then disconnected from their outlet and are made to run on their own batteries, they diverge from their original battery life specifications, which indicates that the battery has lost most of its initial charging capabilities. Eventually, not only it becomes costly for customers, who must frequently spend money on replacing and repairing these batteries, but also negatively impacts the environment when they are discarded.

This Major Qualifying Project seeks to patch this issue by developing a battery management algorithm for systems that do not use battery power often but needs battery power in situations during which energy becomes scarce. To this end, we aim to design a device capable of delivering power using multiple batteries while simultaneously providing the longest battery life possible. By optimizing battery charge cycles and operation times, it is possible to maximize battery lifespan and ensure that the battery avoids a state of deep discharge for as long as is feasibly possible. Given its simplicity, the fact that it is a common application of the previously described situation, and that it only draws a small amount of current from our device, we have chosen to mimic this system to the power system of a laptop.

2. Background Research

2.1. Types of Rechargeable Battery

2.1.1. Lithium-ion (Li-ion) Batteries

Lithium-ion batteries are widely used in consumer electronics such as mobile phones and laptop computers. They are generally characterized by a high cell voltage, good charge retention, and higher specific energy, but poorer high-rate performance and poorer battery cycle life than traditional rechargeable batteries. Because of their various potential advantages, these lithium-ion batteries are used in a wide variety of applications. It is recommended that any lithium-ion batteries avoid deep cycling of charging and discharging since the mid-state-of-charge give the best longevity. It is also considered optimal to store lithium-ion batteries partially charged at room temperature (Linden, Reddy, 2001).

A type of lithium-ion batteries, lithium-ion polymer (LiPo), is identical to a normal lithium-ion battery in its general characteristics. However, lithium-ion polymer battery is much better than normal lithium-ion batteries in that they are much lighter in weight and safer, although more expensive (Linden, Reddy, 2001).

2.1.2. Nickel Based Batteries

Nickel-based batteries are known for their ability to maintain steady voltage until they are close to completely discharging. Because nickel-based batteries have low internal resistance, their charge/discharge speed is fast while maintaining low internal temperature. Among the many types of nickel-based batteries, two of the most well-known ones are nickel-cadmium (NiCd) and nickel-metal hydride (NiMH) batteries. Nickel-cadmium battery, normally used in heavy-duty industrial applications, has long life and need little maintenance; however, nickel-cadmium batteries carry a risk of intoxication of toxic metal component, cadmium. Nickel-metal hydride batteries, which were developed after the nickel-cadmium batteries, can maintain higher energy than standard NiCd batteries, and are more environmental-friendly since they contain less toxic metal components than the cadmium in NiCd batteries (Linden, Reddy, 2001).

2.1.3. Lead-Acid Batteries

Lead-Acid batteries has been a popular battery in industrial applications for over a century. They are characterized by a low energy-to-weight and energy-to-volume ratio and a high power-to-weight ratio. With this characteristic, this battery can supply a high surge of current, which explains for its application in automotive and industrial motor starters. Lead-acid battery is also electrically efficient, with a turnaround efficiency, which compares the discharge energy out with charge energy in, of 70% (Linden, Reddy, 2001). It also has an easy state-of-charge indication by the means of measuring the electrolytes specific gravity. Additionally, this battery has a good charge retention for intermittent charge application. Lastly, this battery has a low manufacturing and capital cost compared to other types of batteries. Despite the described advantages, lead-acid batteries still have few drawbacks that make it unsuitable for consumer applications. First, the battery has a relatively low cycle life with only 50-500 charging cycles. Second, it is hard to manufacture these batteries in smaller sizes due to the chemical and construction of the battery itself. Third, long-term storage of lead-acid batteries would result in the irreversible polarization of electrodes, or known as sulfation (Linden, Reddy, 2001).

As stated above, some of the traditional applications that used lead-acid batteries are automotive and motor starters. Apart from that, some up-and-coming industries such as photovoltaic (PV) power system also used lead-acid batteries, as it is cheap, easy to transport and low maintenance requirements (Manimekalai, 2013).

2.2. Battery Charging Methods

2.2.1. Constant current charging

For this charging method, it is required a constant current is maintained throughout the whole charging process. This constant current is normally determined by a prediction of the battery's State of Charge (SOC). By having a constant charging current based on the battery's capacity and the required charging time, the problem of overcharging or overcurrent in the charging process can be prevented. However, those problems could only be prevented with an accurate, close estimation of the battery's State of Charge (SOC). A sufficient cumulative error in the battery SOC estimation algorithm could lead to either overcharging or undercharging the battery, thus greatly reducing the battery life (Lin et al., 2019).

There are two types of constant current charging, which are standard charging and trickle charging. Standard charging is characterized by a high constant current, typically 0.2C to 1C where 1C specifies the amount of current that would discharge the whole battery in 1 hour (MIT Electric Vehicle Team, 2008). This charging method is normally used at the early stage of the charging process when the battery SOC is low. Trickle charging, on the other hand, provides a constant low current, typically about 0.01C to 0.1C, to the battery. This method is used to compensate the self-discharge rate of the battery when the battery is near full capacity, thus remaining the full-charged condition (Lin et al., 2019).

2.2.2. Constant voltage charging

As the name suggest, this charging methods require the charging power source of the battery to maintain a constant voltage throughout the whole charging process. As the charging voltage stays the same, the voltage across the terminal of the battery increases gradually, resulting in a proportional decrease in the charging current. Once the charging current drops to a lower limit, the charging process is stopped, avoiding the problem of overcharging. This charging method allows the charging current to be adjusted accordingly to the actual SOC of the battery, thus eliminating potential issues caused by the cumulative error in estimating the battery SOC. However, this charging method results in a higher than desire current at the initial charging state at low battery SOC level, which would cause damages to the battery. Additionally, as the charging current reduces with the increase in battery SOC, the current would be at a low level once the battery reaches the high range of SOC, resulting in the reduction of charging speed (Lin et al., 2019)

2.2.3. Constant current – Constant voltage charging

The constant current – constant voltage (CC-CV) is currently the most common method of charging Lithium-Ion batteries in modern Electric Vehicles (EVs) application. It is designed to take the advantages of both constant current and constant voltage charging methods while overcome their drawbacks. Specifically, the charging source would provide a constant, preset current in the CC phase, and when the voltage of the battery reaches an upper threshold the charging process switches to CV charging. During the CV charging, when the charging current drops to a lower threshold, the whole charging process is complete. CC is typically the main part of the charging process using this method, as generally CC charging until the battery reaches about 85% of its capacity. Figure 1 shows the current-voltage vs time graph for the whole CC-CV charging process. One of the advantages of this charging method are that the circuit to charge the battery is easy to implement and design. Also, there is no need to know the battery circuit model when charging with this method. However, one potential drawback of this method is that since the method is independent of battery models, it cannot distinguish between different battery cells of different types in a bank of batteries. Another disadvantage of this method is that the internal resistance of the battery is neglected in the analysis of this charging method, therefore it might cause a surge in battery temperature during the process, thus affecting the charging efficiency. Additionally, the CV charging phase is time-consuming although this phase is only responsible for charging the upper 15% of the battery's capacity, therefore this charging method is unsuitable for fast charging application (Lin et al., 2019).

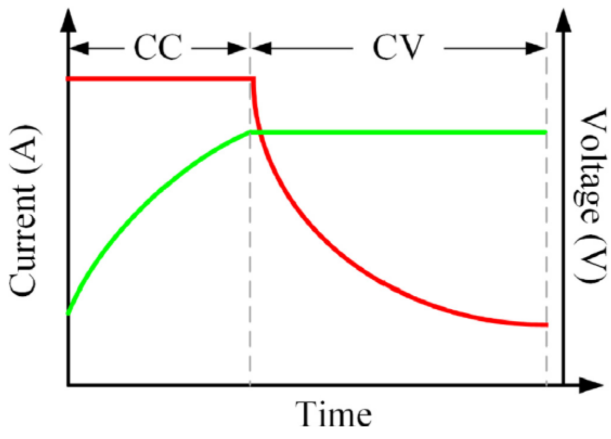


Figure 1. Current-Voltage vs time graph for the CC-CV charging method (Lin et al., 2019)

2.2.4. Multi-stage constant-current charging (MCC)

As stated above, one problem with CC-CV is the excessive amount of time spent in CV charging stage although this stage only accounts for charging the last 15% of the battery capacity, which is unsuitable for fast charging application. For that reason, multi-stage constant-current charging method was developed to address this problem. In this method, multiple preset constant current is applied in different stages. To be more specific, in the first stage, the first preset current, which is usually the highest preset current, is used to charge the battery until the battery voltage reaches an upper limit of cut-off voltage. At this point, the charging circuit would shift to the next preset current and repeat the same process again. The whole process is continually repeated until all of the preset current is used. Typically, there is a gradual decline in the preset charging current at each stage to prevent the battery voltage from hitting the upper limit of the cut-off voltage too quickly (Lin et al., 2019). The most popular form of this method is a five-stage constant current charging. It is proven that this charging method can extend battery cycle life by 25% with an attenuation ratio of 25% (Liu et al., 2005). Figure 2 shows the current-voltage vs time graph for this charging method.

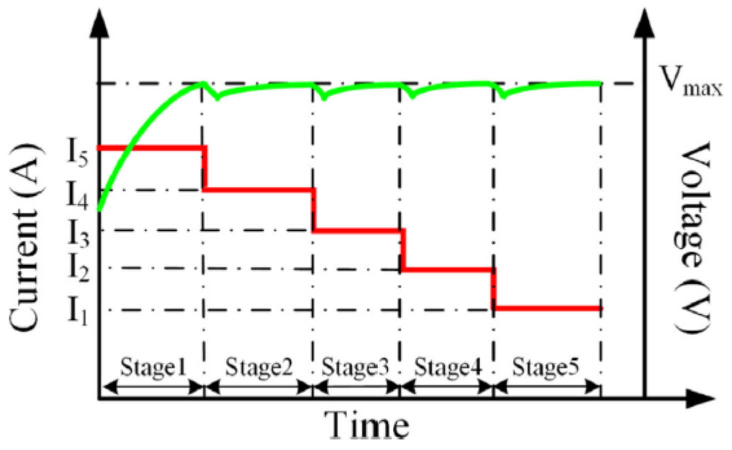


Figure 2. Current-Voltage vs time graph for the five-stage constant current charging method (Lin et al., 2019)

2.3. Current Sensors

Measuring current is integral for Battery SoC estimation as this must be effectively monitored. To this end, we have isolated four kinds of sensing devices and from this group performed a value analysis in order to determine which will better serve our purposes. These are: Shunt Resistors, Hall Effect Sensors, GMR Sensors, and Inductive Coil Current Sensors.

For our value analysis we have settled on a series of parameters that we will give special consideration to. These are, in order of highest to lowest importance: Power Draw, Measuring Range, Accuracy, Types of Measurable Currents, Heat Generation, Linearity, Form Factor, Cost, and Ease of Implementation into Our Current Designs. These will be discussed in greater detail in the sections below.

2.3.1. Shunt Resistor

Cheap and small, shunt resistor sensors shine in the simplicity with which their output can be measured. It can provide precise results through simple application of Ohm's Law. Power draw is highly dependent upon the individual need of each circuit which, should we elect to use it, should grant us some versatility. However, it should be noted that shunt resistors, due to their ohmic nature, have a comparatively higher power draw compared to other sensors. Heat generation is highly dependent upon the resistor's role in the circuit however our PCB can be designed to account for this and should not present a challenge. Shunt resistor measuring range is essentially infinite as sensors can be selected with a massive variety of ohmic values. These sensors have excellent linearity and are especially accurate at low currents as outside noise is less likely to have noticeable effects. It should be noted however, that these resistive elements must often be on the order of milliohms as anything higher would affect current flow. (Leibson, 2018)

These kinds of sensors do require either outside circuitry in the form of transducers (for conversion of output analog signal to a digital one), or inbuilt digital functionality (which is likely to raise our cost). Another weakness lies in poor performance at higher levels of current as some resistors may not be rated for these forms of operation (though it is possible to plan for such operation and choose a sensor accordingly).

2.3.2. Hall Effect

About the size of a small three-pin breadboard transistor, Hall Effect sensors leverage the voltage generated perpendicular to both a current and a magnetic field, themselves perpendicular to each other. Hall Effect sensors have far lower power draw than our previously described shunt counterparts. (Mathas, 2012) Their main advantage however is in their highly sensitive nature, lending it high accuracy for few tradeoffs. Their low primary resistances allow for low heat generation and their indirect current measuring nature means they are quite easy to implement as they can be placed anywhere along the current path while barely affecting the current itself. Their high linearity as well as inclusive measuring range and galvanically isolated outputs and inputs make these devices strong contenders. (Mathas, n.d.)

The Hall Effect sensor's highly sensitive nature makes these sorts of devices quite vulnerable to outside noise such as errant magnetic fields. Their own frequency response is also a factor when considering utilizing these devices at higher frequencies of operation. It should also be mentioned that digital Hall Effect devices capable of handling high current can cost in the tens of dollars and would not be cheap to include in mass production.

2.3.3. GMR

The Giant Magneto Resistive Sensor is a relatively new technology. With "Giant Magento Resistance" referring to the 10-20% change in resistance when exposed to a magnetic field, the GMR sensor is a contender for highly sensitive magnetic sensing devices. Selling for less than five dollars on Digikey, this device is small and compact and is comparative to a small rectangular IC chip. Its low primary resistance gives it a slightly higher power draw than its Hall Effect counterparts. They can also be orders of magnitude more sensitive than HE sensors as well. GMR sensors measure current by sensing the magnetic field generated by a current flowing through its primary resistance. (Allegro Microsystems, 2019)

GMR sensors are generally non-linear. At the middle of its admittedly very large gaussian range, there is an area that does perform resistively. However, its ohmic range is about half as much as that of the Hall Effect Sensor. Despite these drawbacks, where GMR shines is in its sensitivity. Its 25 times stronger sensitivity (compared to Hall Effect) makes it ideal for small current applications. (NVE Corp, n.d.)

2.3.4. Inductive Coil

Inductive Coil sensors have by far the most negative impact upon form factor of the options discussed here. For these types of sensors, our team focused upon specialized types of Current Transformer sensors such as the Rogowski Coil. Sitting at a low of $8.6 and a high of $136 on Digikey, the Rogowski Current Sensor is one of the more flexible sensors available, capable of operating in harsh environments, with a long life, and predictable performance. Aside from its large size, some of its drawbacks are its poor frequency response as well as its inversely related resolution and measurement range (OEM, n.d.). Despite this, when it comes to DC measuring purposes, it is a strong contender.

2.3.5. Value Analysis

As described in Section 5.2.6.2 we have chosen several parameters of importance. For convenience, they are repeated here. These are: Power Draw, Measuring Range, Accuracy, Types of Measurable Currents, Heat Generation, Linearity, Form Factor, Cost, and Ease of Implementation into Our Current Designs.

Given that our end prototype will be a device that provides power while simultaneously maintaining battery longevity, it is imperative that our chosen current sensor behaves passively upon the circuit and does not change/affect power consumption so that battery longevity is only affected by battery chemistry and the charging algorithm itself. It is for these reasons that we have given Power Draw a weight of 10, the highest in our value analysis.

The two parameters tied for secondary importance are both Measuring Range and Accuracy. Given that our device will draw power from a set of several batteries, it was important that our current sensor be capable of reading information at a relatively high level of operation. Not only that, given the sensitive nature of maintaining battery longevity, our team was highly interested in a sensitive device capable of providing error-resistant outputs. For these reasons, Measuring Range and Accuracy were both given a weight of 9.

At this point of research, it was unknown as to what kinds of current would be present within the device. As a result, it was necessary to decide upon a device that would be capable of handling either AC or DC current and be resistant to their accompanying drawbacks such as potential harmonics present during AC operation. As a result, we have weighted Types of Current with a value of 8.

Again, given the sensitive nature of this device, factors that may impact nominal battery operation must be avoided. When batteries are fully charged for example, excess charging will cause the dissipation of heat. Excess heat present in the system can deteriorate the performance of the system by introducing noise and error where there was previously none. For this application, it was valuable that our current sensor be as unobtrusive as possible, mainly meaning that it was necessary for it to not be negatively affective towards the system's generation of heat. It is for this reason that we have weighted Heat Generation with a value of 7.

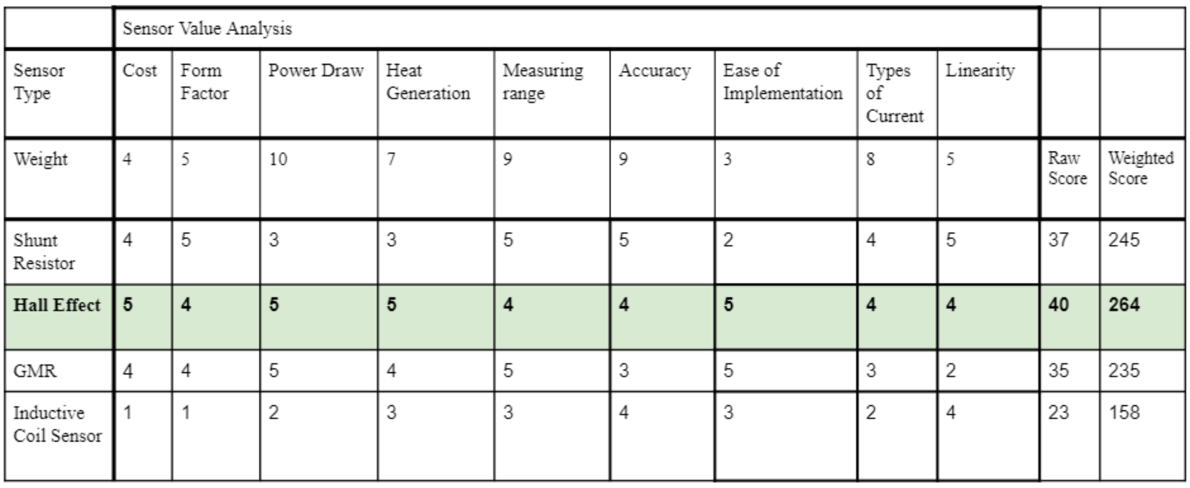
As mentioned previously, it is of great import that our device, and the sensors from whose outputs the microcontroller will orchestrate the battery management algorithm, behave predictably. As such, we have settled upon the general importance of linear device behavior so that its output might remain ohmic through the entire range of operation our device will undergo. For this reason, Linearity has been designated with a weight of 5.

Form factor is not our area of greatest concern. Given that we do not have any size considerations beyond avoiding gross over-sizing, the only real necessity for size is that our device be small without trading away functionality for size. As we aim to design our own PCB, the device should not be a hindrance to other components that might find themselves alongside it. Another advantage of a small size, though minimal, is lower susceptibility to noise as a result of greater distance between our sensor and outside factors. For these reasons Form factor has been weighted with a value of 5.

Of least importance are Cost and Ease of Implementation. In WPI every individual ECE student is given a budget of $250 for their MQP project. As we are three in our team, our budget for this project is high and as such our concern is not to save money but to create a functioning prototype within these constraints. As such, cost is given a weight value of 4. Ease of Implementation goes hand in hand with cost, as this parameter refers to outside circuitry needed and/or difficulties with installation of the current sensor into our current designs. Given that this is our Major Qualifying Project, our team has no qualms with investing extra effort and time into matching the physical work required for the completion of this project and our vision for what it could be. For these reasons, we have given Ease of Implementation a weight of 3.

This value analysis seeks to add special consideration to these previously described parameters of interest. As a result, these are weighted on a scale of 1-10, one for values of lowest importance and 10 for those of highest importance. The selected values are then multiplied to the score each sensor has received. This score is on a scale of 1-5 and follows the same 'importance' scheme as the one for parameter weights. The total summation of these products results in a value we designate as "weighted score." The sensor that receives the highest weighted score is the one that we will use on our system.

Below is our completed value analysis. Due to a combination of low cost, excellently low power draw, small size, good accuracy, high linearity, and fitting measuring range, we have settled upon a Hall Effect current sensor for our device. With a weighted score of 264, its only real contender was the Shunt Resistor current sensor, which despite its completely linear nature, generates far more heat and requires outside circuitry for implementation. Twenty points above its closest contender, the Hall Effect sensor is the clear choice moving forward.



	Sensor Value Analysis										
Sensor Type	Cost	Form Factor	Power Draw	Heat Generation	Measuring range	Accuracy	Ease of Implementation	Types of Current	Linearity		
Weight	4	5	10	7	9	9	3	8	5	Raw Score	Weighted Score
Shunt Resistor	4	5	3	3	5	5	2	4	5	37	245
Hall Effect	**5**	**4**	**5**	**5**	**4**	**4**	**5**	**4**	**4**	**40**	**264**
GMR	4	4	5	4	5	3	5	3	2	35	235
Inductive Coil Sensor	1	1	2	3	3	4	3	2	4	23	158

Table 1. Value Analysis of Four types of current sensing devices for use in Battery SoC measurements.

2.4. Electrical switches

2.4.1. Solid State Relay (SSR)

Solid-State relay uses transistor output, instead of mechanical contacts to control the switch. As shown in Figure 3 below, the output transistor is optically coupled to the LED light in the relay, and this LED light is usually powered by low DC power. Because the switching is optically coupled, the speed of switching of an SSR is much faster than any mechanical relays. However, SSR can have relatively more power dissipation to heat than mechanical relays because it has more resistance when switching current. Therefore, SSR needs some sort of heat sinks for stability. Moreover, SSR usually fails while the switch is closed and therefore carries more risk of an electric shock, whereas mechanical relay is less risky in that regard since its failure usually occurs when the switch is open (Kuphaldt, 2007)

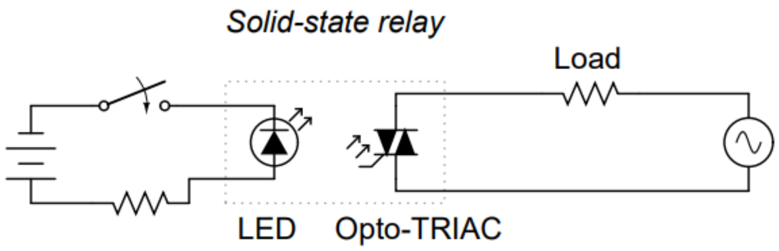


Figure 3. Solid-State Relay (Kuphaldt, 2007)

2.4.2. Electromechanical Relay (EMR)

Electromechanical relay, as shown in Figure 4 below, uses a conductor coil which is energized by AC or DC voltage source and then produces a magnetic field which will physically move the gate of the switch to open or close the switch. These mechanical relays are useful in handling large current/voltage load with small electrical signals. However, compared to modern semiconductor relays (SSR), mechanical relays suffer many limitations, such as EMR can be costly, have a limited contact cycle life, large size, and most importantly, slow in switching speed. The switching speed of the EMR is slower than that of an SSR because the gate of the switch opens and closes physically (Kuphaldt, 2007).

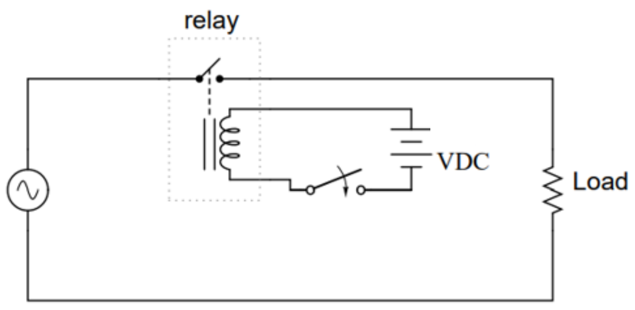


Figure 4. Electromechanical Relay (Kuphaldt, 2007)

2.4.3. Other types of relays

There are different types of relays among the solid-state or the mechanical relays such as time-delay relays, thermal relays, and reed relays. Time-delay relays, with some sort of "shock absorber" mechanism attached to the armature, make it possible to delay the time of switching by delaying armature motion on coil energization, de-energization, or both. Thermal relays, using the heat coming from the difference in the expansion coefficients of a bimetal, opens or closes the switch. Reed relays are sealed-contact type of switches that use a pair of thin, magnetic, metal strips that are in contact with each other; reed relays are known for its ability to handle wide range of supply voltages with a much more rapid switching speed than standard mechanical relays; however, the amount of current they can handle is relatively low (Kuphaldt, 2007).

2.4.4. Diode

A diode is a simple one-way electrical switch that is built based on the P-N junction. This device has two terminals: anode and cathode. Figure 5 shows the construction of the device. The on and off condition of this device are determined by the voltage and current in the circuit. Specifically, the diode is turned on, or so called forward-biased diode, when the current through the diode (i_d) is positive, and a positive voltage higher than the forward voltage drop, which is typically 0.7V, is applied across the diode. On the other hand, the circuit would be turned off, or so-called reverse-biased diode, if a negative voltage is applied across the diode (Sedra, Smith, 2015). Figure 6 shows the v-i characteristics of the diode. As mentioned, a typical diode has a forward voltage drop of 0.7V, but also there exists other types of diodes with a lower forward voltage drop that is widely used in power electronics applications, notably the Schottky diode with a forward voltage drop of 0.3V (Hart, 2011).

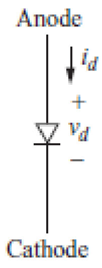


Figure 5. Construction of a diode (Hart, 2011)

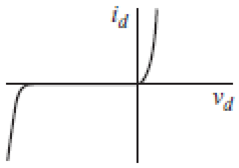


Figure 6. v-i characteristics of a diode (Hart, 2011)

2.4.5. Metal Oxide Silicon Field Effect Transistor (MOSFET)

MOSFET is a three-terminal, controllable electrical switches. Its three terminals are the Gate (G), Drain (D) and Source, and its construction produces a body diode across the Source and Drain terminal. There are two types of MOSFET, which are P-channel and N-channel MOSFET. Those two types are distinguished by the direction of current flow from Drain to Source, as well as the polarity of the voltage applied at the Gate terminal to turn on the device. Figure 7 shows the construction of a N-channel MOSFET. This device is a voltage-controlled switch, in particular the device would be activated, or turned on, if a sufficient Gate-to-Source voltage (V_{GS}) is applied, resulting in a small Drain-to-Source voltage (V_{DS}) and Drain current (i_D). Figure 8 shows the $i_D – V_{DS}$ characteristic of a MOSFET. At its on state, the MOSFET can be modeled as a resistor, and this resistance is called the turn-on resistance (r_{DS}). Power MOSFETs can handle a Drain-to-Source voltage as high as 1500V and a drain current as high as 600A (Hart, 2011).

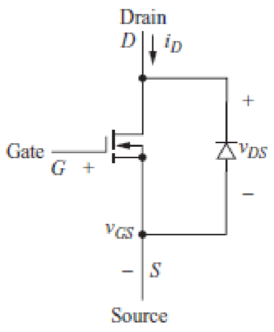


Figure 7. Construction of a N-channel MOSFET (Hart, 2011)

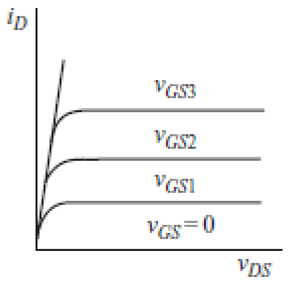


Figure 8. i_D-v_{DS} characteristics of the N-channel MOSFET (Hart, 2011).

2.4.6. Bipolar Junction Transistor (BJT)

BJT, similarly to MOSFET, is also a three-terminal, controllable electrical switches, but there are a few differences that distinguishes the BJT and MOSFET. First, the three terminals of the BJT are, namely, Base (B), Collector (C) and Emitter (E). Figure 9 shows the construction of a NPN BJT. Second, BJT is a current-controlled device, in particular the transistor is activated into on state by providing a sufficient current at the Base terminal (i_B) to drive the BJT to saturation. This would result in a collector current (i_C) flowing through the switch and a voltage drop across the Collector and the Emitter (V_{CE}), typically 1-2V. Figure 10 shows the typical i_D – V_{DS} relationship of the BJT. The relationship between the applied base current (i_B) and the collector current (i_C) is described by the current gain coefficient h_{FE}, specifically $i_C = h_{FE} * i_B$. Power BJT have a low h_{FE} value, typically 20, but it is rarely used in power applications compared to power MOSFET (Hart, 2011).

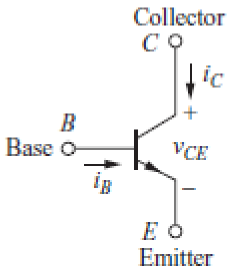


Figure 9. Construction of a NPN BJT (Hart, 2011)

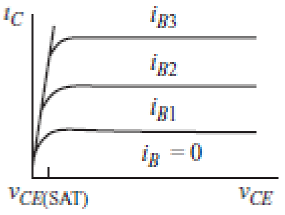


Figure 10. i_D- V_{DS} relationship of a NPN BJT (Hart, 2011)

2.5. Battery State of Charge Estimation

2.5.1. Coulomb Counting Method

Coulomb counting method is the most famous, traditional method used regularly in laptops and portable devices in their battery management systems. This method measures the discharging current of the battery and integrates the discharging current over time in order to estimate SOC. The formula for calculating the SOC using this method is shown in equation 1.

$$SOC = SOC(t_0) + \frac{1}{C_{rated}} \int_{t_0}^{t_0+\tau} (I_b - I_{loss})\, dt \qquad \text{(eq. 1)}$$

In this equation, where SOC(t_0) is the initial SOC, C_{rated} is the rated capacity, I_b is the battery current, I_{loss} is the current consumed by the loss reaction. I_{loss} subtracted by I_b returns discharge current.

Coulomb counting method does not have a high efficiency since it requires an accurate initial SOC value which is difficult to estimate, and the Coulomb efficiency; that is, the ratio of the number of charges that can be extracted from the battery during discharging and the number of charges that enter the battery during charging). Therefore, this method does not account for other losses and inefficiencies during the charging/discharging process (Murnane, 2017).

2.5.2. Kalman Filter Method

To increase the accuracy of the coulomb counting a filter called Kalman Filter can be applied to correct the initial value used in the coulomb counting method. Even though there is more computational burden, Kalman filter method is more accurate than the normal Coulomb counting method. Kalman Filtering is a commonly used mathematical approach to any linear-quadratic problem for estimating the instantaneous state of a linear dynamic system. If the system is non-linear, a linearization process can be applied to every time step to approximate the non-linear system with a linear time varying system; this is called an extended Kalman filter method. The extended Kalman filter method estimates the concentrations of the chemical species by using the terminal current and voltage measurements.

More specifically, this method uses the principle of an observable system, where there exists an observer that can determine the initial state of a system given noisy inputs and output. The filter utilizes a negative feedback to self-correct the estimation of initial states, and the input in this case can be the open-circuit voltage, terminal voltage, battery current, temperatures, and other chemistry parameter (Figure 11).

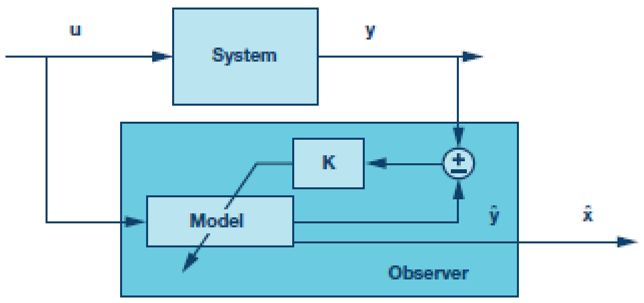


Figure 11. Block Diagram of Kalman Filter Principle (Murnane, 2017)

2.5.3. Enhanced Coulomb Counting Method

The enhanced coulomb counting method is another improved version of a traditional coulomb counting method. In this method, the charge losses during charging/discharging are compensated by considering the charging/discharging efficiencies. More specifically this charging method can be explained by the flowchart shown in figure 12 below.

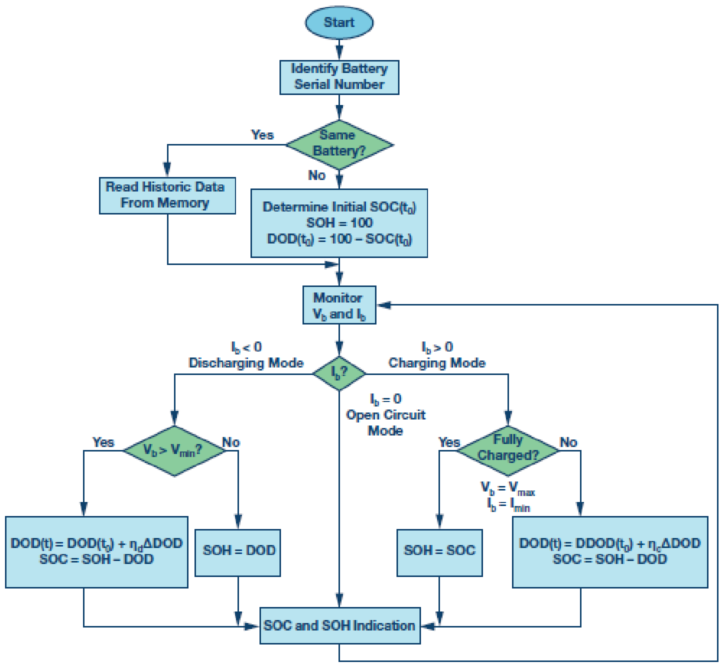

Start
Identify Battery Serial Number
Same Battery?
Yes
No
Read Historic Data From Memory
Determine Initial SOC(t_0) SOH = 100 DOD(t_0) = 100 – SOC(t_0)
Monitor V_b and I_b
$I_b < 0$ Discharging Mode
$I_b > 0$ Charging Mode
I_b?
$I_b = 0$ Open Circuit Mode
$V_b > V_{min}$?
Yes
No
Fully Charged?
Yes
No
$V_b = V_{max}$ $I_b = I_{min}$
DOD(t) = DOD(t_0) + η_dΔDOD SOC = SOH – DOD
SOH = DOD
SOH = SOC
DOD(t) = DDOD(t_0) + η_cΔDOD SOC = SOH – DOD
SOC and SOH Indication

Figure 12. The flowchart of the Enhanced Coulomb Counting Algorithm (Murnane, 2017)

Here, the state-of-health (SOH) of the battery is assumed to be at 100 percent for a new battery. SOH is a simple indicator of the battery health, which is calculated by taking the maximum capacity of the battery and dividing it by the rated capacity (initial maximum capacity), as shown in equation 2 below.

$$SOH = \frac{C_{\text{max}}}{C_{\text{rated}}} 100\% \qquad \text{(eq. 2)}$$

The current and voltage of the cell is then monitored after the SOC is estimated. Depending on the sign of the current, the algorithm can decide if the battery is in charging or discharging mode. In charging mode, SOH is the same as SOC when the cell is fully charged. If the cell is not fully charged, the SOC value can be determined by subtracting the depth of discharge (DOD). Depth of discharge is the percentage of the charge capacity that has been discharged relative to the initial maximum capacity of the battery, as shown in equation 3 below.

$$DOD = \frac{C_{\text{released}}}{C_{\text{rated}}} 100\% \qquad \text{(eq. 3)}$$

When calculating the depth of discharge, the enhanced coulomb counting method takes the discharging efficiency into account for a better accuracy, and the same concept applies when the battery is in charging mode, with the charging efficiency being taken into account.

Charging efficiency can be calculated by taking the amount of charge discharged at a constant minimal rate to the cut-off voltage and dividing it by the amount of charge charged at a constant maximal rate to the designated capacities. The equation is shown in equation 4 below.

$$\eta_C = \frac{C_{\text{discharge, rate min}}}{C_{\text{charge, rate max}}} \qquad \text{(eq. 4)}$$

Discharging efficiency can be calculated by adding up the products of discharging currents and discharging period from the first two stages (the first stage with a specified current to a designated depth of discharge, and the second stage with a minimal rate to the cut-off voltage) and dividing them up by maximum charge amount. The equation is shown in equation 5 below.

$$\eta_d = \frac{I_1T_1 + I_2T_2}{C_{\max}} \quad \text{(eq. 5)}$$

2.6. Unbalanced Battery Cells

Battery cells in a series branch are not always kept at the same charge, or in another word unbalanced cells. This phenomenon is typically caused by multiple factors. One of the most common factors is that batteries are charged with different amount. While the percentage of SOC unbalance remains constant during entire discharge, voltage differences between the cells vary with state of charge because the rate at which the voltage change varies with the change of SOC; as the charging process goes towards the end of discharge, the voltage difference increases exponentially (Figure 13).

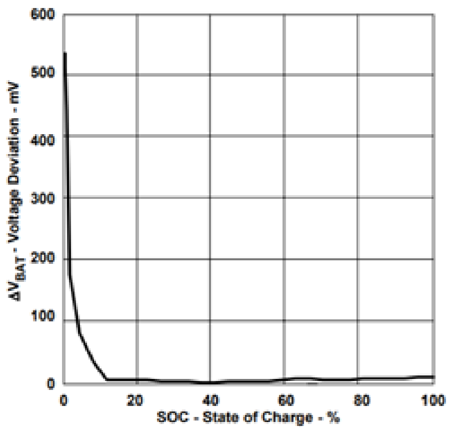


Figure 13. Open-Circuit Voltage dependence on SOC (Barsukov, n.d.)

The unbalancing of cell can also happen because of total capacity difference among the batteries. In the charging process, if the battery cell's total capacity is different to start with, the open-circuit voltage will start to differ as the process goes, which is very similar to the SOC unbalance problem. The difference between the SOC imbalance and the capacity imbalance is that capacity imbalance causes less voltage difference than the SOC imbalance because SOC correlates with voltage more than the total capacity itself.

Another factor of the batteries that can cause cell unbalancing is the difference in impedance. Even though impedance differences do not cause differences in open-circuit voltages, they do cause differences in cell voltages during the discharge process. Impedance imbalance can be a harmful factor when it comes to balancing SOC. Figure 14 indicates the distortion caused by impedance deviation is larger than that caused by SOC unbalance during the majority of the discharging stage (from 10 percent to 100 percent).

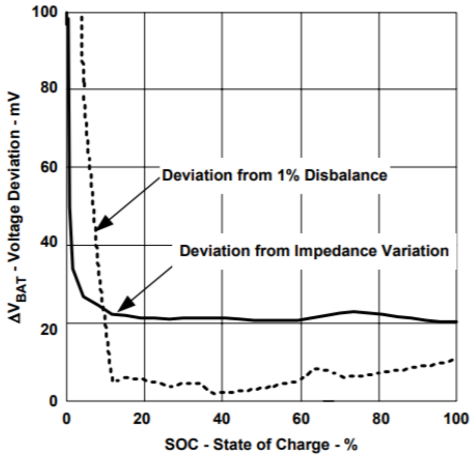


Figure 14. delta V between 2 cells with 15% impedance unbalance at C/2 discharge rates (Barsukov, n.d.)

Now that we looked at the causes of battery cell unbalance, we need to know what kind of problems can be caused by unbalanced cells. When there is a cell unbalance, the cell with higher SOC is exposed to higher voltage. The cell with lower capacity will have a much higher voltage than the remaining cells with normal capacity, which will have lower voltages than expected (Figure 15). When the lower cell has a total capacity deficiency above a certain percentage, its cell voltage begins to rise into dangerous limit. This causes in degradation of the battery cell, and moreover, a safety concern when overcharged, especially for Li-ion batteries which has high energy concentration in small volume.

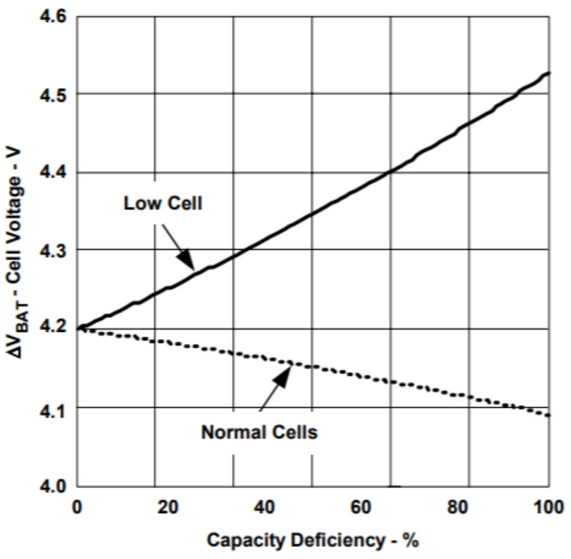


Figure 15. Individual Cell Voltage vs Capacity Deficiency from Nominal (Barsukov, n.d.)

There are two major ways of cell-balancing known as current bypassing method, and charge redistribution method. Current bypassing method can be used by implementing field-effect transistors (FETs) parallel with each cell (figure 16). These FETs are controlled by a comparator or by microcontroller for simple voltage-based algorithms that turn-on the bypass FETs during the onset of voltage differences. The downside of this method is that the by-passed charge energy goes wasted when the power is supplied only by the battery.

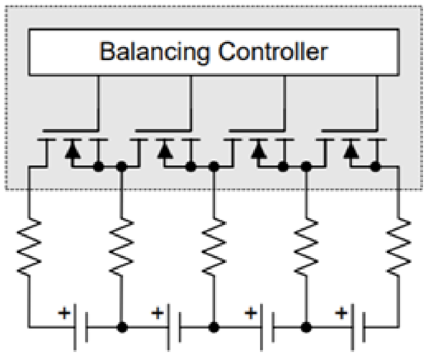


Figure 16. By-pass FETs Based Cell Balancing Circuit (Barsukov, n.d.)

Another method for cell-balancing is charge redistribution method, or in another words, charge shuttle method. Literally, this method redistributes the charges from the higher SOC cells to the lower ones by connecting a capacitor first to higher voltage cell, than to lower voltage cell (Figure 17). Charge shuttle method has an disadvantage in that there is a significant energy loss in capacitors being charged, and that it is only efficient near the end of discharge (Barsukov, n.d.).

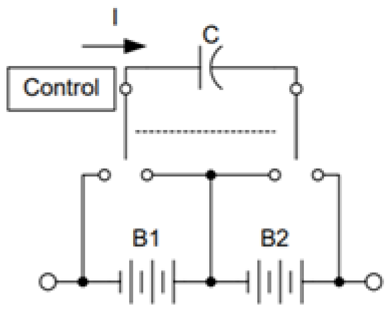


Figure 17. Simple Capacitor-Based Shuttle Cell Balancing Circuit (Barsukov, n.d.)

2.7. Laptop Power System

A typical laptop power system block diagram is shown in Figure 18. The source of power for the laptop circuitry comes from the battery pack and the laptop charging adapter. The battery pack contains different numbers of Li-Ion battery cells depending on the models with a typical output voltage of 8.7V – 16.8V, a smart battery controller chip and a thermal protection circuit (Wei, Lee, 2004). The main purposes of the smart battery controller chip are to calculate the battery charge level and report it back to the system as well as control the battery charge/discharge cycle (Cao, Emadi, 2011). For the thermal protection circuit, as the name suggests its main objective is to protect the battery and the rest of the system from the event of overheating. Particularly, if either the temperature of the battery cells or the processor rises above a certain threshold, the circuit would disconnect the supply power from the battery from the rest of the laptop circuitries to protect the system (Cao, Emadi, 2011). The AC Adapter, or the laptop charging adapter, takes on the outlet AC power of 110V-220V, 50/60 Hz depending on the region, and output a DC voltage of 19 V, which is an industry standard for laptop chargers (Wei, Lee, 2004). The power provided from the battery or the AC adapter goes through a variety of DC-DC converter stages, which convert the DC voltage from one level to another to provide the needed voltage for different circuitries inside the laptop, such as the battery charger, processing units, memory, ...When the laptop is plugged into the AC adapter, the DC output from the adapter provides power to the laptop, including the battery charging circuit to charge up the battery. In this case, the battery is charged and does not provide power to the laptop. When the AC adapter is unplugged, the laptop is powered by the battery pack.

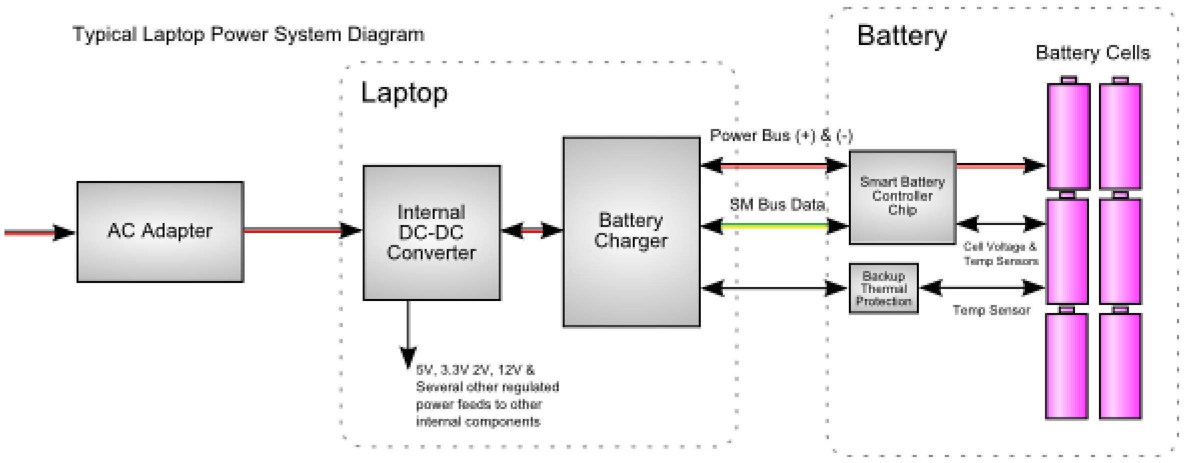

Typical Laptop Power System Diagram
Laptop
Battery
Battery Cells
AC Adapter
Internal DC-DC Converter
Battery Charger
Power Bus (+) & (-)
SM Bus Data
Smart Battery Controller Chip
Backup Thermal Protection
Cell Voltage & Temp Sensors
Temp Sensor
5V, 3.3V 2V, 12V & Several other regulated power feeds to other internal components

Figure 18. Block Diagram for a Typical Laptop Power System (4infor, n.d.)

2.8. Shielding for Current Sensor

As we have chosen a Hall Effect Sensor, which generates a voltage proportionate to a current applied at its input while producing its own magnetic field, we need to avoid any inaccuracies this field may cause. As the device requires a magnetic field to function properly, it is also susceptible to errant magnetic fields. In an application such as this, in which knowledge of the device's moment to moment SsC is important for its continued nominal functioning, we must avoid all problems that might arise with the sensors. As such, we have elected to use an RF Shield Cover (Dickinson, Bentley, 2013), purchased from Digikey. This tin-ferrous alloy shield will protect our sensor from outside noise and our entire system from the sensor itself. This will ensure that our sensor provides the MCU with accurate data.

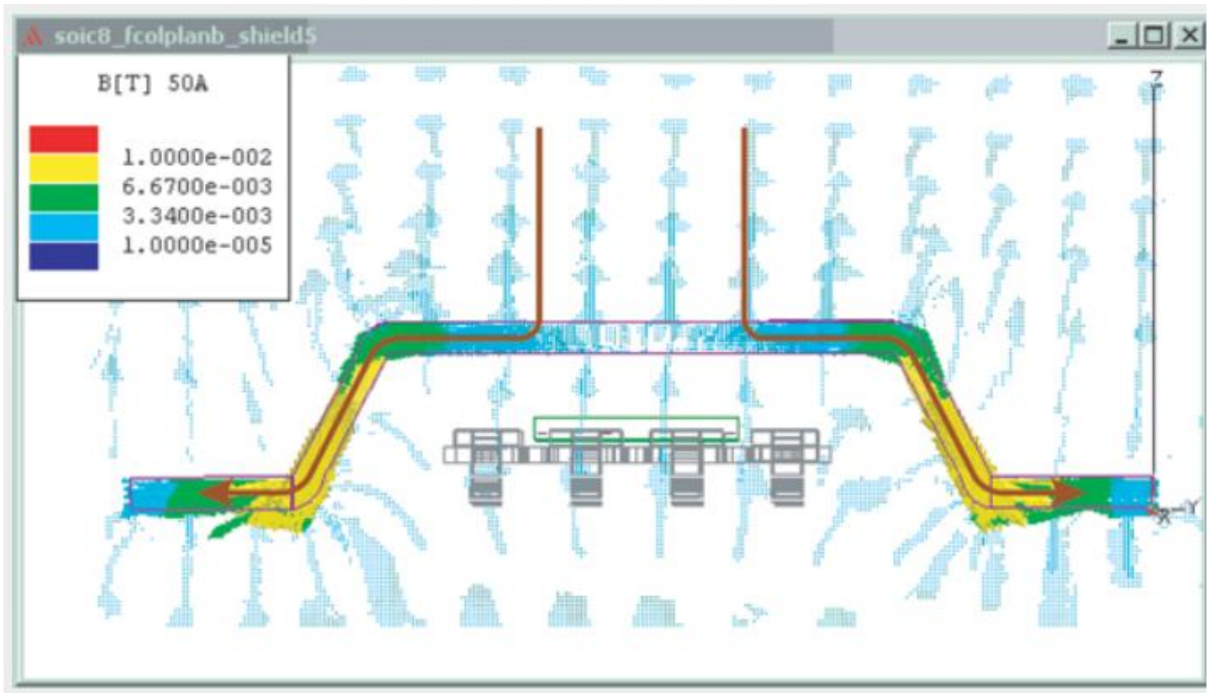


Figure 6. Magnetic Field Map of Shield. A simple ferrous alloy shield deflects virtually all perpendicular flux lines, effectively shielding the Hall element.

Figure 19. Magnetic Field Map for the current sensor with the presence of the shield (Dickinson, Bentley, 2013)

3. Project Objectives and Scopes

3.1. Project Scope

As discussed in the introduction, we focus on developing a battery cycling algorithm and a proof-of-concept prototype for any rechargeable-battery-operated devices of any battery sizes that are plugged in most of the time. However, for this project we will focus on the laptop power systems, which is a commonly used and important device that falls under the described category. For the size of battery banks in our proof-of-concept prototype, we will use a battery bank with 4 Li-Ion cells in the configuration of 2 cells in series and in parallel with the other 2 cells in series. The reason that we choose this configuration will be described in detail in the later sections. The main point is to proof that our algorithm is expandable, meaning that the algorithm can be applied to any sizes and configuration of battery banks, and we feel that 4 battery cells would be enough to carry this objective out.

3.2. Project Objectives

For this project, the team want to achieve the following objectives:

- Develop and verify an algorithm to cycle the laptop batteries while being plugged into the wall
- Design a battery switching circuit to cycle different battery branches
- Design a control circuit to monitor the batteries' conditions and control the switching circuit
- Prototype the control and battery switching circuits in Printed Circuit Boards (PCB)
- Design a test load for the system
- Test and verify the system functionality as a proof of concept for the algorithm

4. Methodology & Approach

4.1. Battery Cycling Algorithm

Battery life is subject to deterioration from the moment that it is first manufactured. Deep discharge, improper charging methods and high temperatures can all damage a battery. Charging the battery from different states of discharge to different states of charge can have a marked effect on battery life. This can be seen in Figure 19 below.

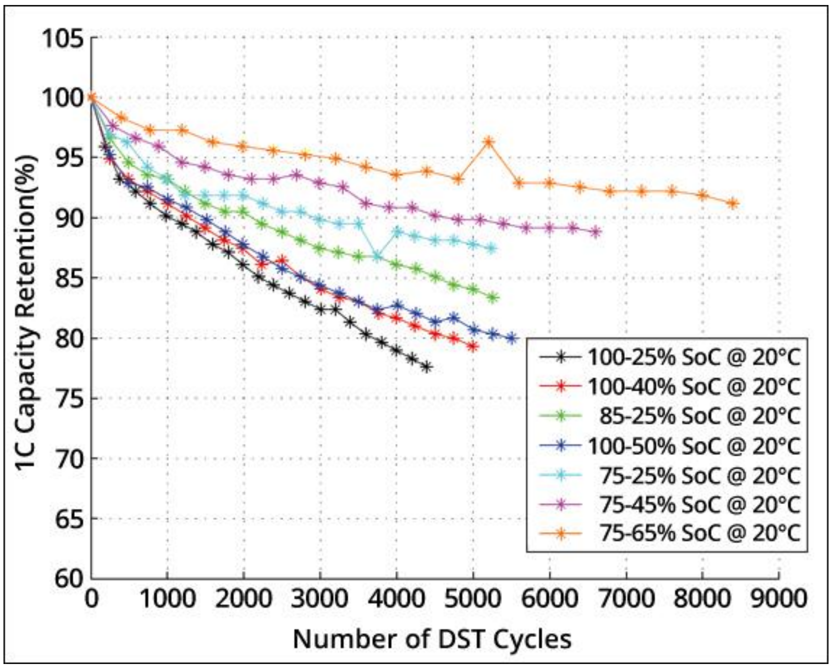


Figure 19. Degrading Battery Capacity over time using different charging cycles. (Xu et al., 2016)

Charging a battery up to and using it consistently and continuously at 100% can stress its chemistry. A battery forced to remain at a constant state of maximum capacity will quickly deteriorate and as such partial charging cycles should be used to maintain a battery in nominal operating conditions. As seen in the above image, the most effective battery cycling range at maintaining a battery at high max capacity for the longest amount of time is from 65%-75%, while the worst range is from 100%-25%. For our purposes, given that our battery must be capable of prolonged use while unplugged, ¾ of max charge is not enough and the laptop user would not be satisfied. Through careful consideration, the team has elected to use a battery cycle of 90-100% as, though fully charging the battery will be harmful in the long run, we believe it is a necessary tradeoff for increased unconnected operating time.

The complete algorithm would operate as follow. When the battery pack is fully charged, a battery branch in the bank would be discharged while the other branches are kept charging. When this battery branch reaches a State of Charge level of 90%, it will be put back to the charging power and the other battery branch will be discharged to the same State of Charge level. The process will be repeated until the charging state is disrupted, or the device is unplugged from the charger.

4.2. Chosen Battery State of Charge (SOC) Estimation Algorithm

The battery SOC estimation algorithm chosen for this project is the enhanced coulomb counting method explained in part 2.4.3. As explained in part 2.4, coulomb counting measures the discharging current of the battery and integrates it over time to estimate SOC; it is the most famous and traditional SOC estimation method used in many electronic devices, but it does not account for the errors such as losses caused by charging/discharging inefficiencies. Therefore, the enhanced coulomb counting method was used for our project to account for more specific losses. For this algorithm, as explained in the flowchart in Figure 12, factors such as state-of-health (SOH), depth-of-discharge (DOD), charging efficiency (η_C), and discharging efficiency (η_d) will be programmed into MCU to account for the losses during charging and discharging.

5. System Design and Implementation

5.1. Top Level System Block Diagram

Based on the typical laptop power system described in section 2.5, the team were able to construct the top-level system block diagram, which shows the major modules of the system as well as the interconnection between each module. The diagram is illustrated in Figure 20 below. The red lines denote the power path with the voltage level of each path, while the black lines denote the signal path of the system.

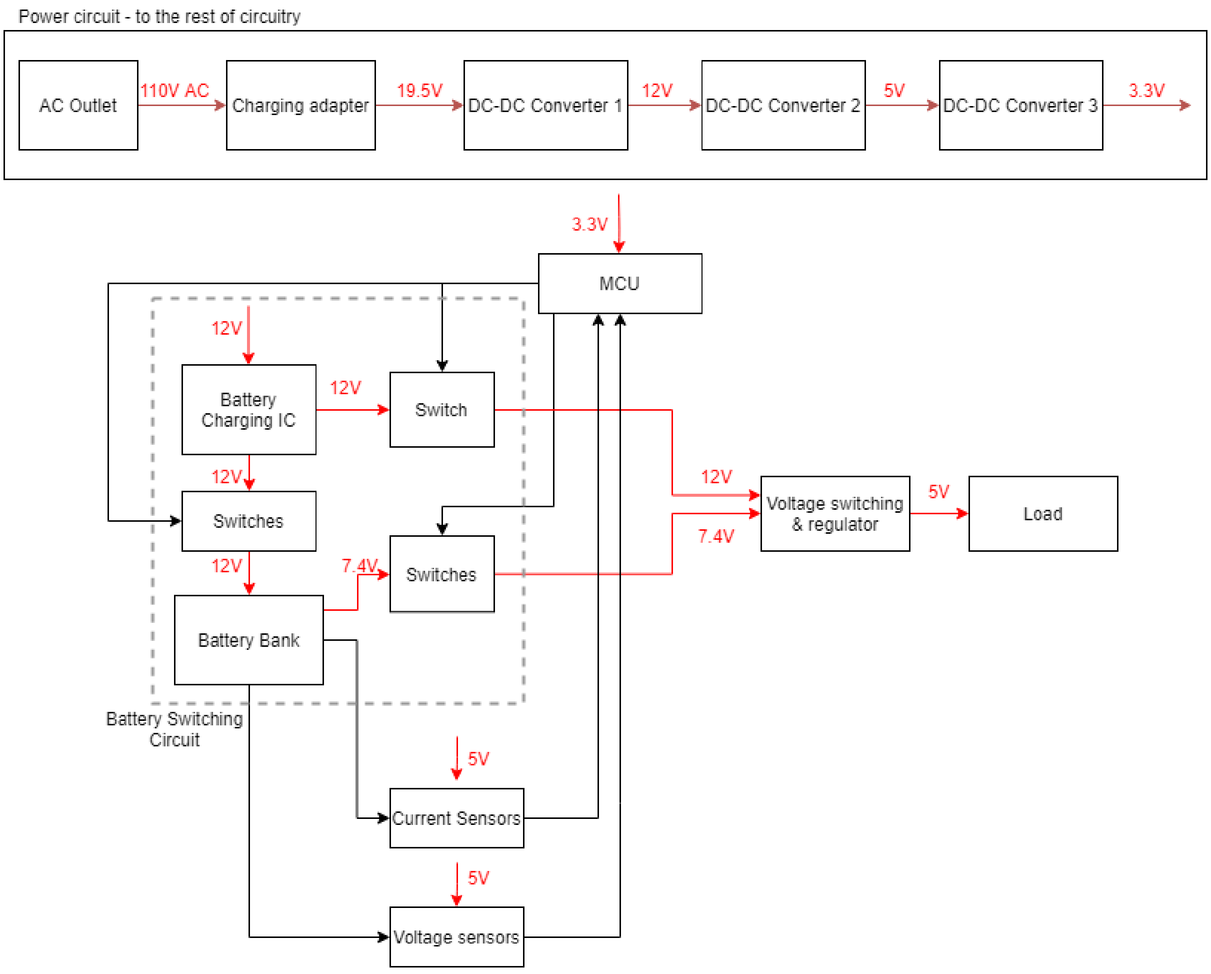


Figure 20. Top-level system block diagram

5.2. Detailed Module Description

5.2.1. DC-DC Converter/Regulator Stages

The DC-DC converter/regulator stages make it possible to bring down the voltage from the laptop adapter (19.5V) to different voltage stages, which are 12V, 5V, and 3.3V. Our main purpose for the voltage conversion is to use the 12V power line to supply power to the sub-circuitries in the main battery switching circuit, such as the instrumentation amplifier used for voltage measurement, and 3.3V to the MCU chip.

The selection of the three converters and their following passive components was done by utilizing Texas Instrument's (TI's) WEBENCH Power Designer. This power circuit generator creates customized power supply circuits based on the designer's desired requirements. When it came to our design, we focused on the minimum requirement for our current rating of 3A and fixed output voltages of 12V, 5V, and 3.3V.

The step-down converters used in our design for 12V, 5V, and 3.3V are LM2676SX-12/NOPB (Figure 21), LM2596SX-5.0/NOPB (Figure 22), and LM2596SX-3.3/NOPB (Figure 23), respectively. The 5V and 3.3V regulators use the same model, but with different fixed output voltages. Passive components such as capacitors, diodes, and inductors were selected also by TI WEBENCH Power designer considering the required equivalent series resistance values for each component.

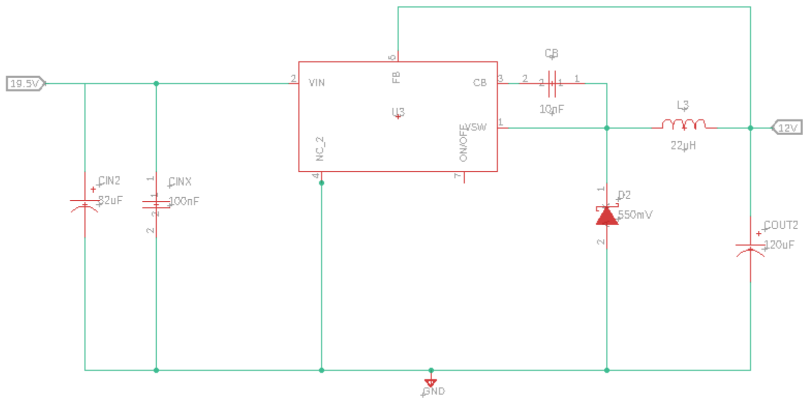


Figure 21. Step-Down Converter from 19.5V to 12V (EAGLE, 2021)

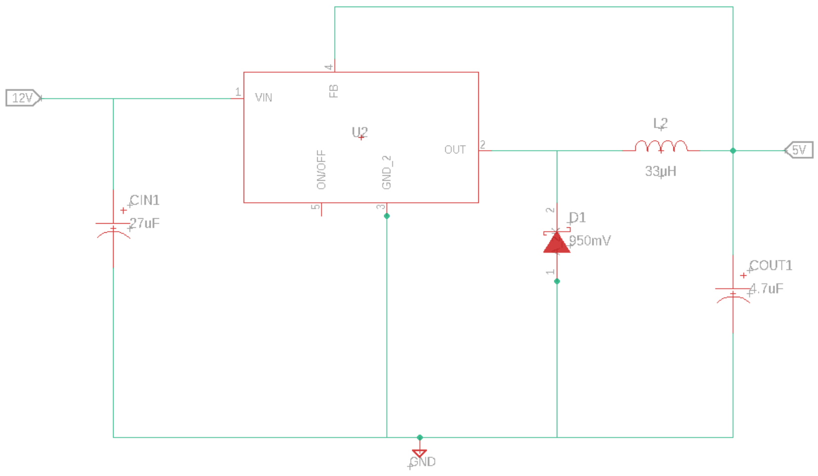


Figure 22. Step-Down Converter from 12V to 5V (EAGLE, 2021)

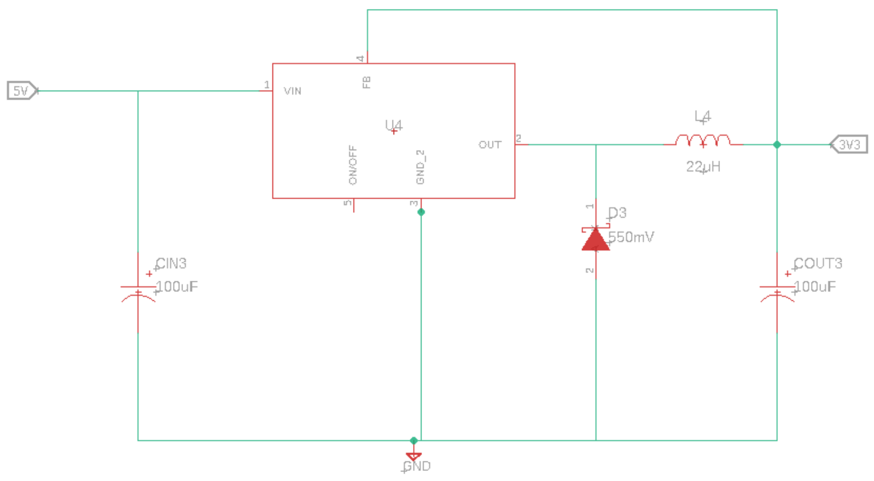


Figure 23. Step-Down Converter from 5V to 3.3V (EAGLE, 2021)

5.2.2. Battery Bank

To power our device and to engage in our main functionality, that is the switching and charge cycling of our battery packs, it was necessary to choose a battery chemistry that would fit our needs. We required a battery that could retain a large amount of charge for extended periods of time. Lithium-Ion, as discussed before, has strong rechargeability and maximum capacity and as such was a natural choice for this project. Given that our project is intended to be able to recharge larger portable devices, a single set of batteries would not suffice. The Lithium-Ion Battery Banks that we will be using will consist of two groups of two batteries placed together in series. These will be wired with switches to shift between charging and discharging between both battery groups as dictated by the MCU.

There were several choices available to us when it came to choosing our Li-ion battery cells. There was the PRT-11855 with a nominal voltage of 7.4V and capacity of 1AH would already solve our voltage requirements on its own. There was also the BL2200F6034501S2PPMK with a 2AH and nominal voltage of 3.7 V which if used as a single cell would be optimal for our purposes. Unfortunately, the first was discarded as the team elected to retain the customizability of using two individual cells rather than a prepackaged duo. The second was only available for use with Molex connectors, and in an effort to retain the design simplicity this option was also discarded.

The batteries used for this battery bank will ultimately be 2600mAh 26J 18650 Li-Ion cells manufactured by Samsung. These cylindrical batteries will serve as individual cells and will be mounted onto our PCB using Keystone's surface-mounted 1042 plastic mounts. These batteries will be placed in individual circuit branches and will be attached to their own ACS723LLCTR-10AU-T, a Hall Effect Current sensor. At a nominal voltage of 3.6 V, these batteries when wired in series will have a combined voltage of 7.2 V, which is more than capable enough of powering a laptop battery.

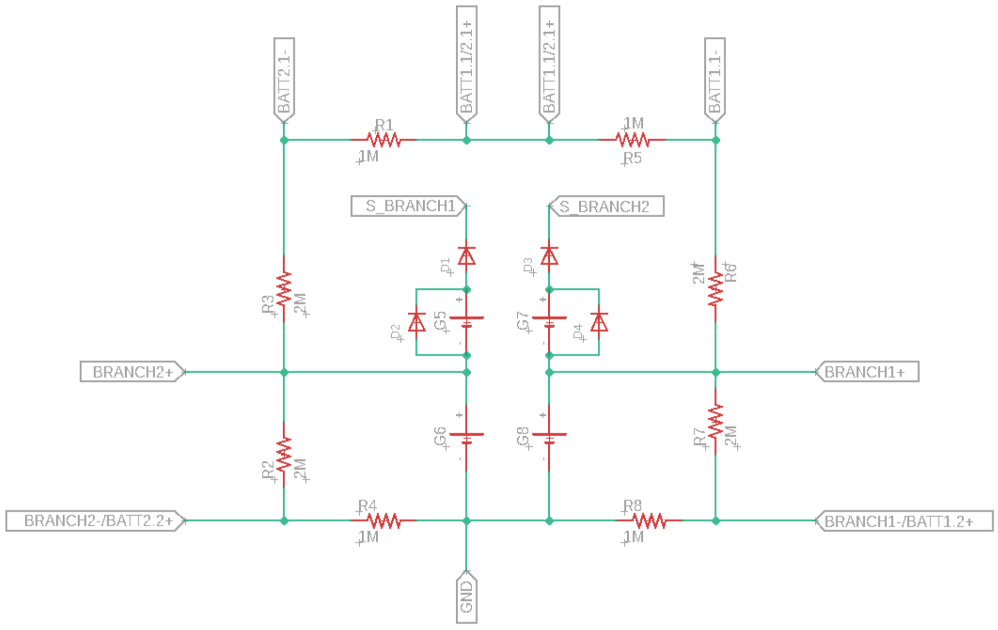


Figure 24. Circuitry of the Li-Ion battery pack (EAGLE, 2021)

5.2.3. Battery Charging Circuit

The main purposes of the battery charging circuit are to charge the Lithium-Ion battery bank with the proper voltage level and provide the protection for the battery bank throughout the charging process. Also, the circuit would need to utilize the Constant Current-Constant Voltage (CC-CV) charging method to charge up the battery bank. Designing the circuitry to achieve this job would be complex, difficult and out of the scope of this project, which focus more on cycling and switching the battery in and out during the charging process. That said, picking a pre-made IC that can do the job and is available on the market would simplify this problem and provide a reliable option to realize this module.

The general requirements for the IC are the battery chemistry supported, number of battery cells supported, charging method used and input voltage rating. Specifically, the IC must be able to support the charging of Li-Ion batteries, support up to 4 cells of battery, provide the CC-CV charging method and an input voltage rating of at least 19.5V. Additionally, the IC preferably has a stand-alone operation, meaning the chip would not require the support of any additional ICs or specific system to function properly. Such attributes would minimize the complexity of the circuit, reduce the footprint taken on the PCB and cut off on the cost associated with the bill of materials.

The next step is choosing the components that met the given requirements. After doing some research on Digikey with the applied filters for the requirements, we come up with the three main options, which are the LTC4007-1 and LT1505 from Analog Devices, and the MAX1737 from Maxim Integrated. All three chips meet the general requirements given, as they all use CC-CV as the charging method, support Li-Ion of up to 4 cells with a conversion efficiency of more than 90%. Although the MAX1737 only has an efficiency of 90% compared to 96% of the LTC4007-1 and the LT1505, it has a comparatively lower price at \$6.84 per unit in Digikey compared to \$8.83 of the LTC4007-1 and \$10.85 of the LT1505. That said, the MAX1737 is the chosen IC since it has a relatively lower price with an insignificant sacrifice in efficiency.

The circuit for this module involving the MAX1737 battery charging IC is shown in Figure 25. This circuit is designed following the guidelines and example circuits in the datasheet of the MAX1737 (Maxim Integrated, 2017). With respect to the reference voltage of 4.2V, resistor values for R10 and R11 (both1 kΩ) were decided to have per-cell voltage limit to be 2.1V. Similarly, resistor values for R12 (3 kΩ), R13 (5.6 kΩ), R23 (1 kΩ) and R24 (3 kΩ) were selected to set the input current limit to be 1.5A and the charging current to be 1.3A; during this process, value of the input and output current sensing resistors R19 (50mΩ) and R18(0.1Ω) were also decided accordingly. The inductor value of L3 (18 µH) at the output stage was also decided to have the ratio of the ripple current to DC charging current to be 40%. Moreover, the value of the output capacitor C1(at least 46 µF) was decided to absorb inductor ripple current, with enough equivalent series resistance (at most 0.1Ω) for stability. A notable thing is that we are not planning to use a thermistor to monitor the battery temperature; it is replaced with a 10 kΩ resistor, which essentially prevents the chip from shutting down the charging.

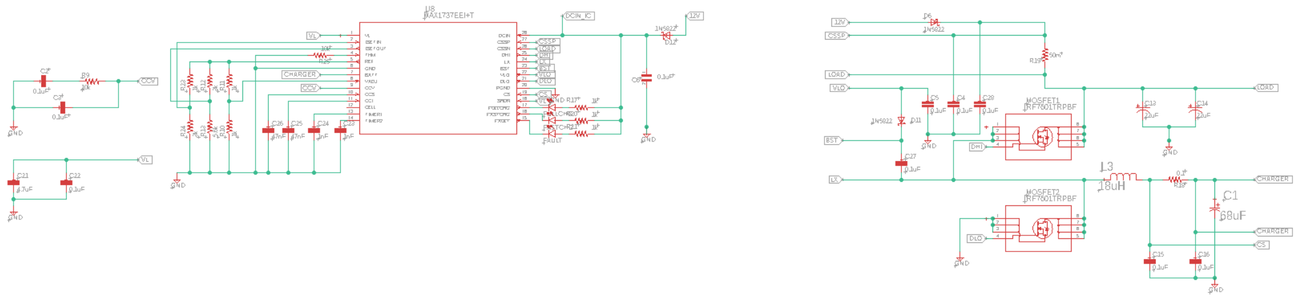


Figure 25. Schematic for the battery charging circuit using MAX1737 (EAGLE, 2021)

5.2.4. Switches

The switches are a key part in deciding which battery branch or cell would be charged or discharged based on the state of charge condition. Particularly, we need to have switches between charging source and each of the battery branches, between the charging source and the load, and finally the switches that can connect or disconnect a given branch to the load. For the switches that connect or disconnect a branch and the load, it is required that those switches are in normally closed position when there is no voltage potential or current applied to control the switch. The reason for this is that when the charger is disconnected, there would be no power provided to the control circuit and the load is supposed to be run from the battery power. In this case, there would be no control signal coming from the control circuit to the switches and using a normally open switch would prevent the load from being powered by the batteries.

Given that application, we choose the solid-state relays to be one of our continuous switches which would switch between any two parts with 2 different potential levels such as between the charging source and the battery banks and branches. Additionally, for the normally closed switches that connect the charging source and the battery branches, we decide to use the Single Pole Double Throw (SPDT) mechanical relay, since this type of switch has double throws, one of which is normally connected to the pole. Also, we opt to go with the MOSFET for any switching to ground. For both types of switches, they need to be able to handle the current and voltage ratings of the circuit. Those ratings are determined to be 5 A for the current and 12 V for the voltage. For the solid-state relay specifically, the turn-on current needs to be lower than 20 mA since most microcontroller GPIO pins cannot source more than that amount of current (ST Microelectronics, 2015). For the MOSFET specifically the turn-on resistance must be relatively small to ensure that the voltage drop across the MOSFET is minimal, and the drain-to-source current at the turn-on voltage of 3.3V, which is the logic high output from the microcontroller module, must be at least 5A, the current ratings of the circuit.

We proceed to pick the component that meets the requirements. For the solid-state relay, we chose TLP3547 from Toshiba. This part has a maximum on-state current of 5A, which meet the current ratings requirement, and the turn on current is only 5 mA, which makes it possible to drive this relay using the microcontroller GPIO pins. Another important factor that we take into consideration is that the price of this part, at $8.23, is the cheapest among the switches with the same ratings according to Digikey at the time of writing (Toshiba, 2020). For the mechanical relay, we decide to go with the G5LE-1 from OMRON. Some notable parameters of this part are the maximum switching current of 10A and a maximum switching voltage of 125 VDC or 250 VAC for the contacts, a rated current of 33.3 mA with a coil resistance of 360Ω when the coil are being driven with a 12V supply, and a maximum switching time of 10ms (OMRON Corporation, n.d.). As the GPIO pins of the microcontroller can only supply a maximum of 20 mA of current, we would need to use a current amplifier circuit using an NPN BJT (ST Microelectronics, 2015). For the MOSFET, we opt for the CSD17585F5 from Texas Instruments (TI). The key parameters are a low turn-on resistance of approximately 45 mΩ, a high current handling ability with the drain-to-source current of more than 10 A at a turn-on gate voltage of 3.3V. Also, this part is capable of high-speed switching, with a turn on delay time of only 4 ns (Texas Instruments, 2017).

We have been able to implement the solid-state relays in the battery switching circuit using EAGLE. The schematic of such implementation is shown in Figure 26 below. The MOSFETs are implemented in the battery charging circuit, and its implementation was shown in the schematic in Figure 25 in section 5.2.3 above.

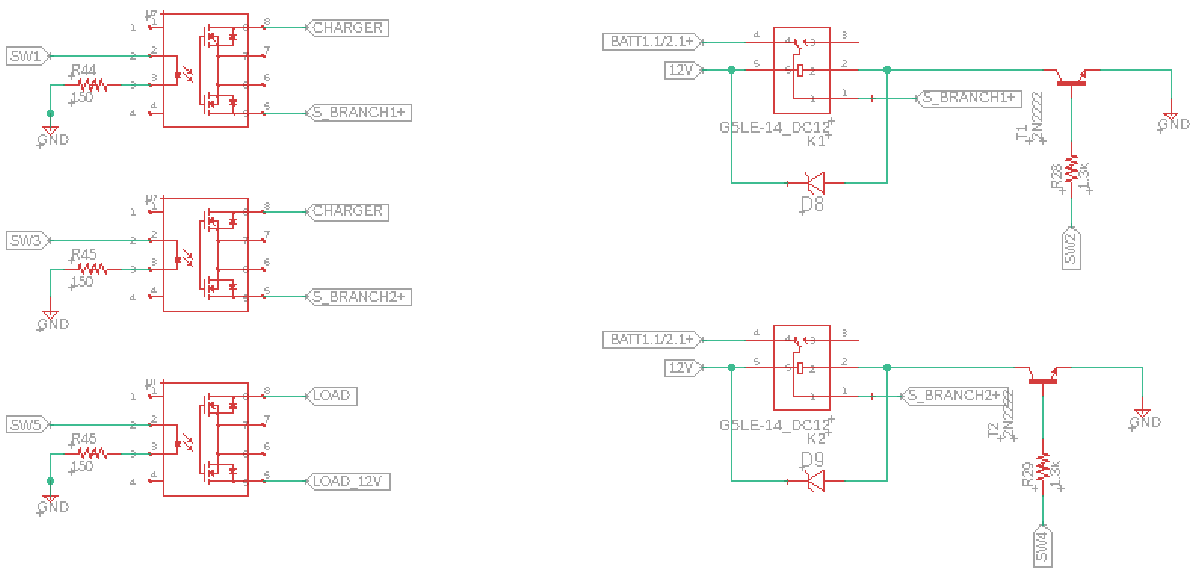


Figure 26. Implementation of solid-state relay and mechanical relay in the battery switching circuit (EAGLE, 2021)

5.2.5. Voltage and Current Sensors

5.2.5.1. Measuring Voltage

As part of the battery state of charge estimation, the voltage of each battery branch and cell needs to be monitored. Typically, these voltages can be routed directly to the microcontroller's internal Analog-to-Digital converter (ADC). However, as mentioned in the Background section, the nominal voltage of a single Li-Ion cell is 3.7 V, which can go up to 4.2 V at full charge. The equivalent nominal and maximum voltage for a series battery branch is therefore 7.4 V and 8.4 V, respectively. As we supply the microcontroller module with a 3.3 V supply, the maximum reference voltage of the MCU internal ADC is also 3.3V. Therefore, the battery cell and branch nominal and maximum voltage is well above the range of conversion of the ADC, so we need to design a circuit to interface those signals to the microcontroller.

Figure 27 below shows the circuit to measure the battery branch and cell voltage with the microcontroller's internal ADC. The idea is to step down the branch and cell voltages by a third, and inside the microcontroller software we will convert the ADC readings back to the original readings. There are 2 stages in this circuit: the voltage divider stage and the unity gain voltage buffer-differential amplifier stage. The voltage divider stage steps down the voltage of interests by a third, for instance the equivalent voltage for the nominal and maximum voltage of a battery cell would be 1.23 V and 1.4 V, respectively, and the equivalent voltage for the nominal and maximum voltage of a battery branch would be 2.46 V and 2.8 V, respectively. The resistors in this circuit are chosen to be in the order of MΩ, which result in a minimal current draw by this circuit and prevent the battery to be significantly drained by the measurements taken. The detail relationship between the original and converted cell voltage is shown in equations 6 to 8 below, with cell 1 denoting the upper battery cell in the branch and $V_{\text{blocking diode}}$ denoting the voltage drop across the blocking diode, which are D_1 and D_2 in figure 27:

$$V_{cell\ 1,converted} = \frac{1}{3} * \left(V_{cell\ 1,original} - V_{blocking\ diode}\right) \quad \text{(eq. 6)}$$

$$V_{cell\ 2,converted} = \frac{1}{3} * V_{cell\ 2,original} \quad \text{(eq. 7)}$$

$$V_{branch,converted} = \frac{1}{3} * V_{branch,original} \quad \text{(eq. 8)}$$

The second stage is the unity gain voltage buffer-differential amplifier, which consists of an op-amp circuit in a voltage buffer configuration and another op-amp circuit in the differential amplifier configuration with a gain of 1. As the converted voltages for the first cell and the battery branch shown above are the voltage drop across the resistors, we need to take the voltage potential difference between the nodes and route it to the microcontroller internal ADC. To achieve this, we use a differential amplifier configuration with a gain of 1, and to avoid the amplifier to load down the converted voltage we add a voltage buffer circuit in between of the voltage divider and the differential amplifier for each of the first cell and branch converted voltage. The whole configuration mentioned can be realized using a pre-made unity-gain instrumentation IC available on the market. The chosen IC for this application is the AD8277 from Analog Devices, which provide 2 unity-gain instrumentation amplifiers in a single package, saving up on the footprint required on our PCB. The detailed schematic of the circuit used to measure the battery cell and branch voltage is shown in figure 27 below.

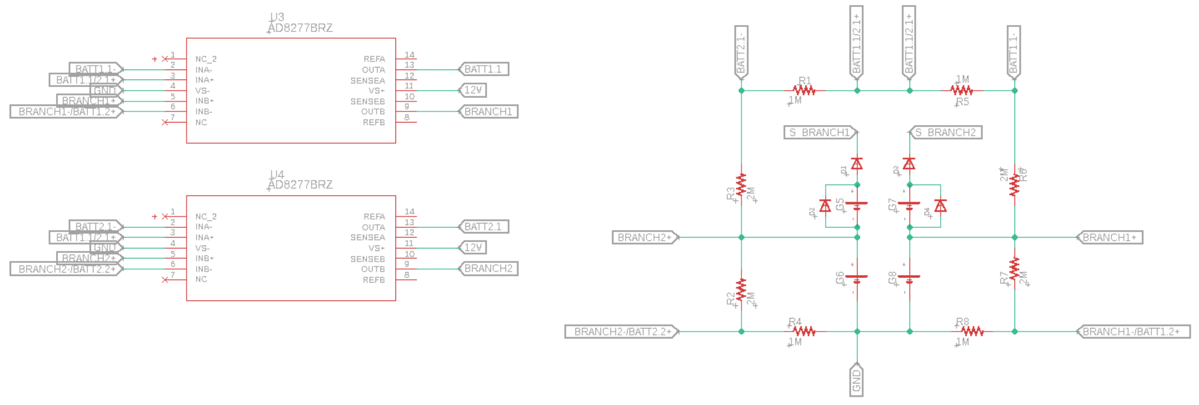


Figure 27. Voltage divider and Unity-gain Voltage Buffer-Differential Amplifier Circuit to measure the battery cell and branch voltage (EAGLE, 2021)

5.2.5.2. Measuring Current

The implementation of the current sensor circuit is shown in Figure 28 below. This design was referred to the example circuit on the ACS723 current sensor datasheet (Allegro Microsystems, 2019).

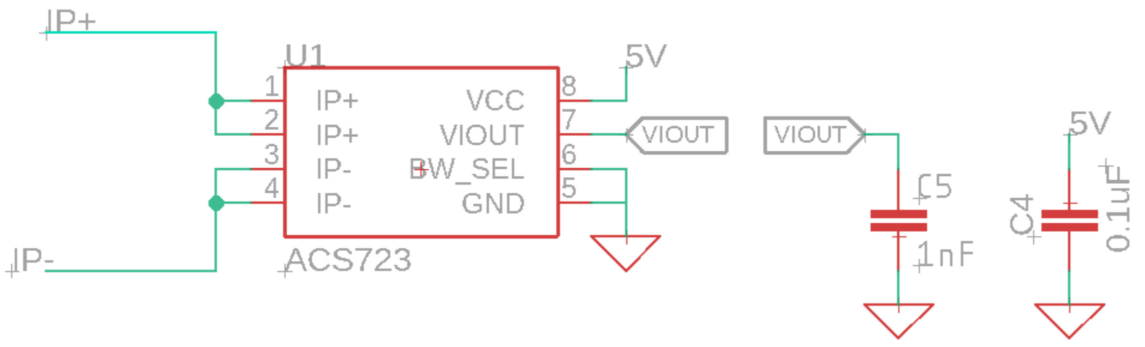


Figure 28. Circuit for the Current Sensor (EAGLE, 2021)

5.2.5.3. Interfacing the Current Sensor with the Microcontroller's ADC

According to the datasheet of the ACS723 Current Sensor, the output voltage of the current sensor ranges from 0.5V to 4.5V, with an input of 0A corresponding to an output of 2.5V and the resolution of 400 mV/A (Allegro Microsystems, 2019). However, the microcontroller operates on a supply voltage of 3.3V, which makes its built-in ADC to have a voltage conversion range of 0V to 3.3V (ST Microelectronics, 2015). Therefore, the output voltage of the current sensor needs to be scaled down to the conversion range of the ADC, specifically by 0.66 times. Figure 29 below shows the relationship between the output voltage of the current sensor, the corresponding ADC input voltage after scaling down the current sensor output and the input current to the sensor.

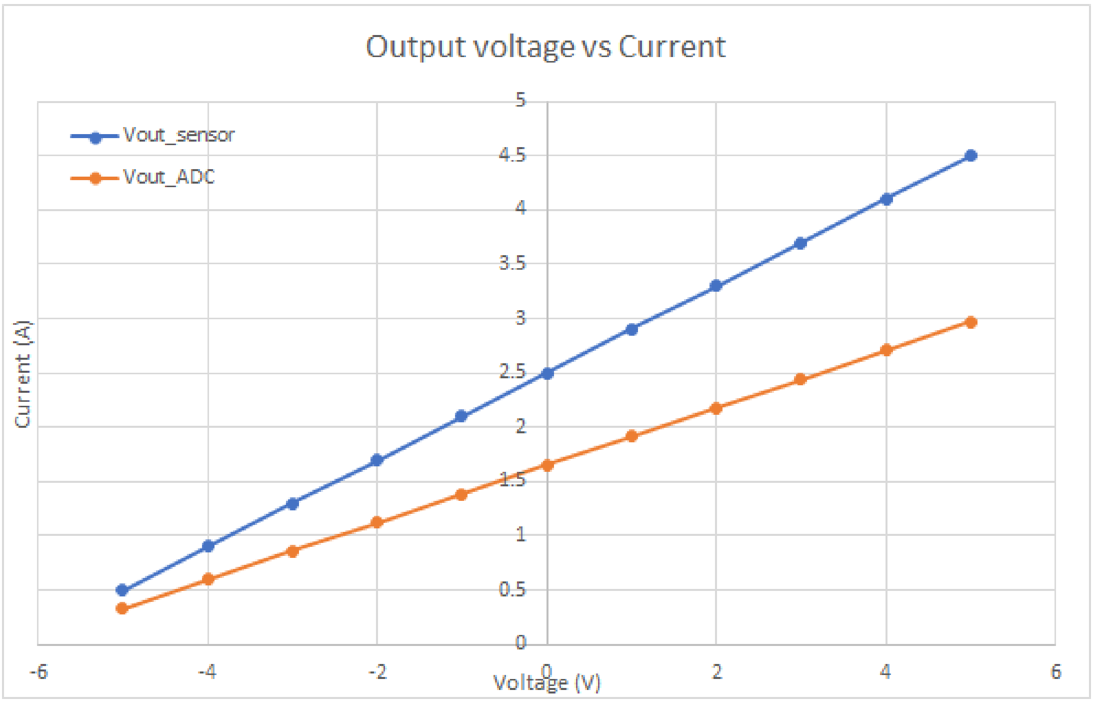


Figure 29. Relationship between current sensor output voltage and scaled-down input voltage to the ADC vs input current. (Note: the input current is on the horizontal axis while the voltage is on the vertical axis).

As the voltage needs to be scaled down, a simple voltage divider would achieve this job. To achieve the scale-down ratio of 0.66, the value of the resistors was chosen to be 1.7 kΩ and 3.3 kΩ. Those resistors will need to be 1% tolerance. According to ST Microelectronics, the input impedance of the ADC inside the microcontroller is only 50 kΩ, which might not be high enough given the value of the voltage divider and could lead to the output voltage from the voltage divider being loaded down (ST Microelectronics, 2015). That said, a voltage buffer is added to avoid the output voltage from the voltage divider being loaded down before going into the ADC input. Figure 30 below shows the implementation of this interfacing circuit.

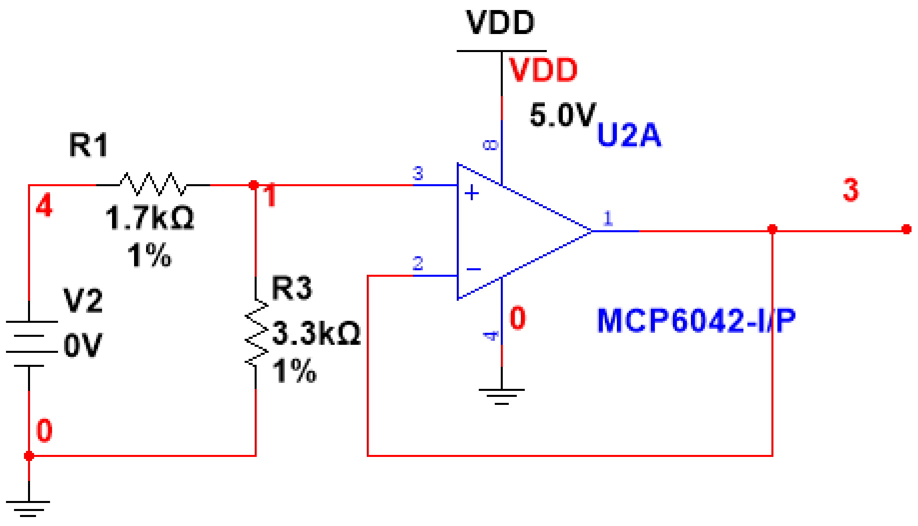


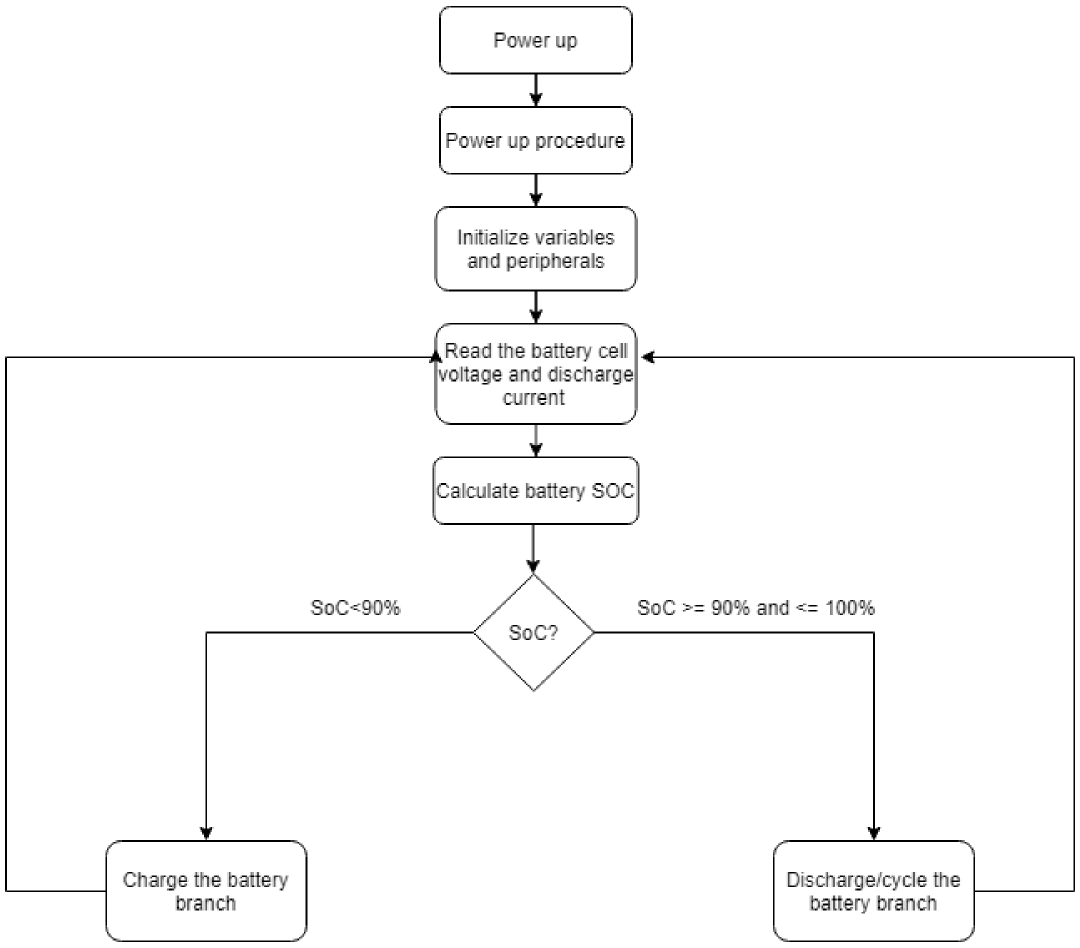
Figure 30. Implementation of the Current Sensor Interface circuit (Multisim, 2021)

5.2.6. Microcontroller Unit (MCU)

The microcontroller unit is one of the most important parts of the system. One of the purposes of this module is to take measurements from the voltage and current sensors, then using these data to carry out the Enhanced Coulomb Counting algorithm discussed in section 2.4 and 4 to estimate the battery state of charge. Based on the estimated state of charge, the microcontroller controls the charging or cycling process of the battery bank accordingly by outputting a logic high or low using its General-Purpose Input-Output (GPIO) pins accordingly. Additionally, the microcontroller unit performs the overvoltage/undervoltage and overcurrent protection of the battery pack. Specifically, based on the measured voltage and current in the ADC, the microcontroller would disconnect the appropriate switches by outputting a low logic output using its GPIO pins to stop the charging/discharging process.

This module is powered by a 3.3V supply voltage, which is achieved by stepping down the 12V voltage level in the DC-DC converter stage. The main inputs for this module come from the voltage and current sensors, which is in the form of voltage signals in the range of 0-3.3V. These signals are routed to the ADC pins of the microcontroller. The software inside the microcontroller would process these signals and convert it to a value of voltage or current accordingly. The outputs from this module are in the form of logic high or low signals, which are used to control the switches and the battery charging IC. As the supply voltage of the microcontroller is 3.3V, a logic high corresponds to a voltage of 2.4V-3.3V and a logic low corresponds to a voltage of 0V-0.5V.

The flowchart for the microcontroller software is shown in figure 31 below. As it can be seen from the chart, after the microcontroller is powered up, the software goes over the microcontroller's power up procedure to get all parts of the system running. Then, the program starts the hardware and variables initialization process, which in this case is initializing the GPIO pins, the ADC, the timer and any variables needed such as the battery cell and branch voltage, current, state of charge, etc. After finishing the initialization process, the microcontroller starts taking the measurements from the ADC pins and converts those data into battery voltage and discharge current. Based on these data, the microcontroller calculates the battery state of charge using the state of charge estimation algorithm. If the calculated State of Charge of a given battery branch is below 90%, the microcontroller would control the switches and battery charging IC accordingly to charge the said battery branch. On the other hand, if the State of Charge of a given branch is somewhere between 90% and 100%, the microcontroller would control the switches and the battery IC accordingly so that the said branch is discharged.



Power up
Power up procedure
Initialize variables and peripherals
Read the battery cell voltage and discharge current
Calculate battery SOC
SoC?
SoC<90%
SoC >= 90% and <= 100%
Charge the battery branch
Discharge/cycle the battery branch

Figure 31. The software for the microcontroller program

Based on the objectives, the input-output specification and the flowchart of the microcontroller software, we choose the STM32F103C8T6, a microcontroller from ST Microelectronics, as the microcontroller of choice. This MCU operates at a nominal voltage level of 3.3V, which fits well with the mentioned voltage level that we plan to supply the microcontroller module. With a clock speed of 72 MHz, 128 kB of flash memory and 20 kB of SRAM, this microcontroller is more than capable of running the algorithm as well as storing the necessary variables and information needed for the software. Also, this microcontroller has 2 internal 12-bit ADCs with a total of 10 input channels, which is enough to capture the voltage and current measurements from the battery bank. The maximum voltage conversion range of the internal ADC is 0V-3.6V, which fits our needs given that the voltage signal coming from the voltage and current sensor have the maximum voltage of 3.3V, and the resolution of the ADC is 0.80586 mV/LSB, which allows us to have precise readings. The GPIO pins can source up to 20 mA of current and operate at a high switching speed of 50 MHz, which allows the switches in the battery switching circuit be controlled in a short amount of time, although fast switching speed is not a requirement for the battery switching circuit. Also, this MCU has a lot of communication peripherals such as SPI, I2C, UART, CAN, which gives us the freedom to implement other types of sensors or peripherals using these communication protocol should we decide to change the sensors (ST Microelectronics, 2015). Additionally, the software for this microcontroller is fairly easy to develop with the help of the ST-developed library called Hardware Abstraction Layer (HAL), which helps us not having to deal with the low-level registers of the microcontroller when programming, as well as the ST-developed graphical-based software called STM32CubeMX, which allows the user to generate the hardware initialization code for the microcontroller by configuring the desired settings graphically. Finally, it is worth mentioning that this microcontroller and its development board are available for a very low price of $5.5 for the MCU and around $6 for the development board according to Amazon and Digikey.

As we are planning to implement the project on multiple PCBs, it is required for us to come up with a circuit schematic for the microcontroller and its required peripherals. Figure 32 below shows the schematic for the circuitry of the STM32F103C8T6 microcontroller that we want to implement in our PCB, which were built in the EAGLE software. This schematic was designed based on the STM32F103x Datasheet and Reference Manual, as well as the schematic of the STM32 Blue Pill development board, a board that used the same MCU (ST Microelectronics, 2015).

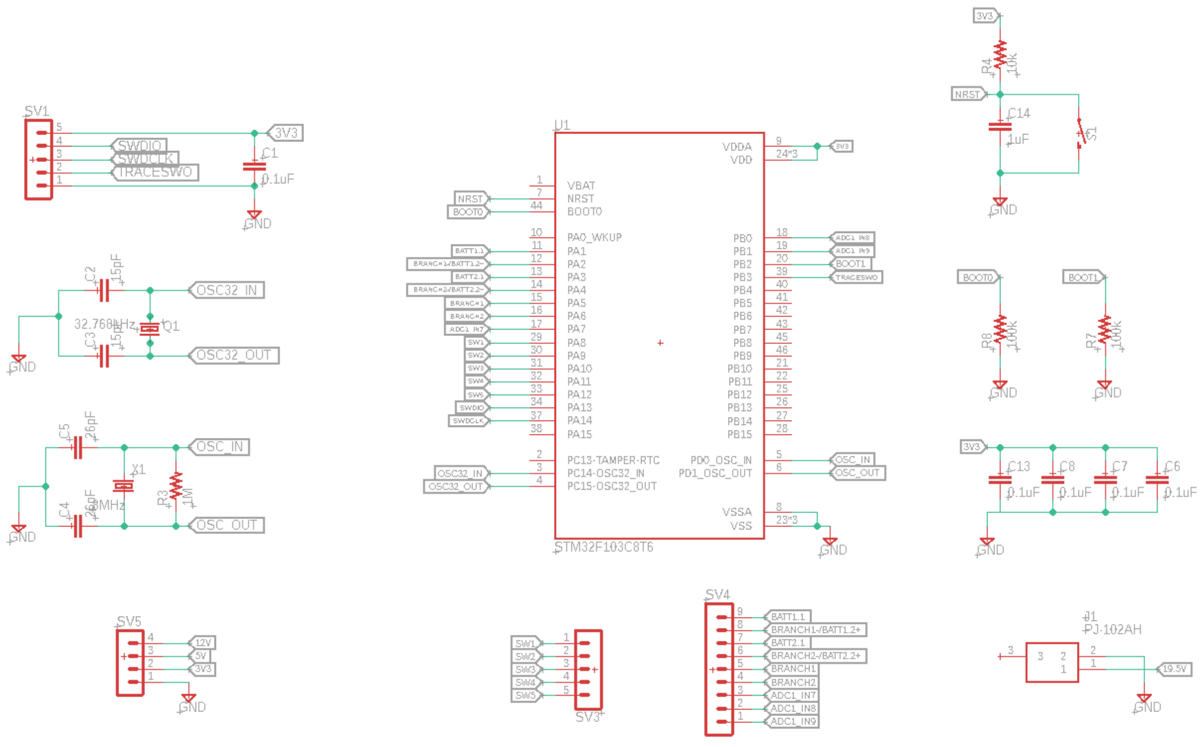


Figure 32. Schematic for the STM32F103C8T6 Microcontroller Circuit (EAGLE, 2021)

5.2.7. Load Stage

The load stage for this project could have been a less dynamic and more simplified load. However, it was decided that to be able to test the reliability of our circuit, it was necessary that the load be more dynamic than a single lightbulb. To fulfill this vision, the load stage breadboarded to test the overall system is composed of an LED matrix of 81 20mA LEDs of 1.9V forward voltage controlled by two CD4017B Decade Counters which are themselves pulsed by an LM555 Timer IC.

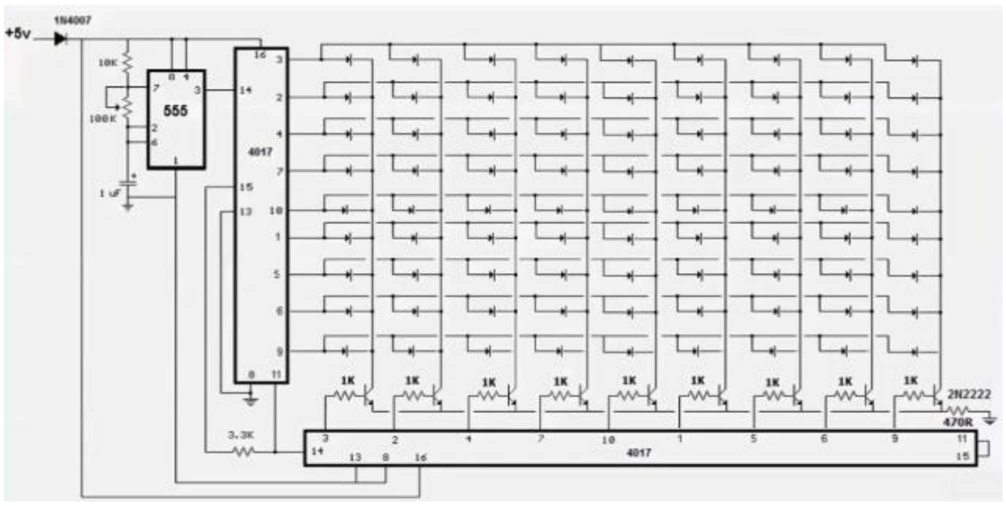


Figure 33. Schematic for the LED Matrix Load Circuit

The relationship between the LM555 Timer and its accompanying passive elements is described in the equations below:

$t_h = (R_A + R_B) * C * \ln(2)$ (eq. 9)

$t_l = R_B * C * \ln(2)$ (eq. 10)

With t_h & t_l referring to both the time it takes for the capacitor to charge to its higher limit of 2/3 Vcc and discharge to its lower limit of 1/3 Vcc, the switching of which is responsible for creating the clock output. Adding these to one another results in the period of the sawtooth waveform which can then be used to calculate the frequency, as shown in the equations below:

$T_{Period} = t_l + t_h$ (eq. 11)

$F = \frac{1}{T_{Period}}$ (eq. 12)

By using a potentiometer instead of a resistor for R_B, we have general control over the output waveform's frequency from the LM555 Timer, which will affect how fast the LEDs switch. The counters themselves are responsible for sequentially triggering each LED in every respective column.

If every LED were to be on at the same time, we should expect a current draw of 1.62 A. However, since this is a switching circuit, and the components that will generally remain on will only be the LM555 Timer and Decade Counters, which will in total have an estimated current draw of 30mA. This added to however many LEDS are on at the same time (20mA per) will result in our total load current draw, as shown in the equation below. n is the total number of LEDS on at a time.

$I_{Load\ Total} = I_{LM555} + 2I_{Decade\ Counter} + n * I_{LEDs}$ (eq. 13)

5.3. System Implementation

5.3.1. Implementation plan

As mentioned in the project objectives, the team plans to implement the circuitry in the form of Printed Circuit Boards (PCB). The PCBs and their associated schematics are implemented using the software EAGLE. Due to the board size limitation of 800 mm^2 in the free version of EAGLE that the team are currently using, we design 3 separate PCBs of the system. Specifically, those PCBs are for the Microcontroller Circuit, the Battery Switching Circuit and the Battery Bank. The detailed design procedures and the final image of the PCBs will be presented in section 5.3.2, 5.3.3 and 5.3.4.

5.3.2. PCB Design of the DC-DC Converter Stage and the Microcontroller Circuit

The PCB design of both the DC-DC converter and MCU was done on a same board using EAGLE. Both circuits were first drawn in the schematic file which is later converted to PCB layout using the board conversion feature. For proper conversion from schematic to PCB layout, it was required to make sure that the model libraries for all components used in the schematic are compatible.

5.3.2.1. Component Placement

Once the schematic was converted to PCB layout, the primary task was to place each component at the proper places. For DC-DC converters, we kept components within each stage close to each other, with output capacitors and inductors relatively far away from other components to prevent any noise. Directions of the components were adjusted to make the equivalent pins face each other for easier tracing (wiring).

Same method was applied to the placement of MCU components, but with more caution near the two oscillators. The challenging part during the MCU component placement was that the MCU pins for both oscillators were on the same side, which made it difficult to place both oscillators without interfering with other components; we were able to solve this problem by using via hole to place the overlapping components on the opposite (bottom) layer.

5.3.2.2. Tracing

The next step was tracing, which was done by using automatic tracing feature in EAGLE. Even though the traces designed automatically can seem relatively more complex than those designed manually, they are more accurate and can still achieve the same result as manual tracing as long as the placement of the components are properly done.

It is important to use proper trace thickness for certain parts of the circuit to handle proper amount of current. The DC-DC converter stages are required to handle at least 3A, including the traces connected to MCU that provide 3.3V. We were able to calculate the proper width of the wire by using the calculation tool provided by Advanced Circuits (Advanced Circuits, 2018) which uses the standard formula from IPC-2221A (IPC, 2003). The challenging part for deciding the trace thickness was that our original calculation gave the thickness of 27 mil, which was too thick for the traces to go be connected to the converter pins. Therefore, we changed the amount of temperature rise from 10 to 15 degrees Celsius to bring down the thickness to 21 mil. The width of the traces supplying 3.3V to MCU were set to be 10 mil to be safe in current handling (6 mil was default trace width with automatic tracing).

Another challenge in the tracing process was that the automatic tracing feature did not apply our desired trace thickness to the traces. Therefore, we had to change the thickness manually which introduced interference with other components/traces. We were able to solve this problem by rearranging the traces/components and using via holes. No changes were made to the traces connected to ground since ground plane were made in the last step.

Lastly, a ground plane was made for both top and bottom layers for proper ground connection to each component in the circuit. When ground planes are formed, most empty spots of the board are filled with either red (for the top layer) or blue (for the bottom layer). A completed PCB board design is shown in the figure below, with Figure 34 having no ground plane for clear visualization of each component and Figure 35 having the complete ground plane added. A manufactured board with all the components soldered onto is also shown in figure 36.

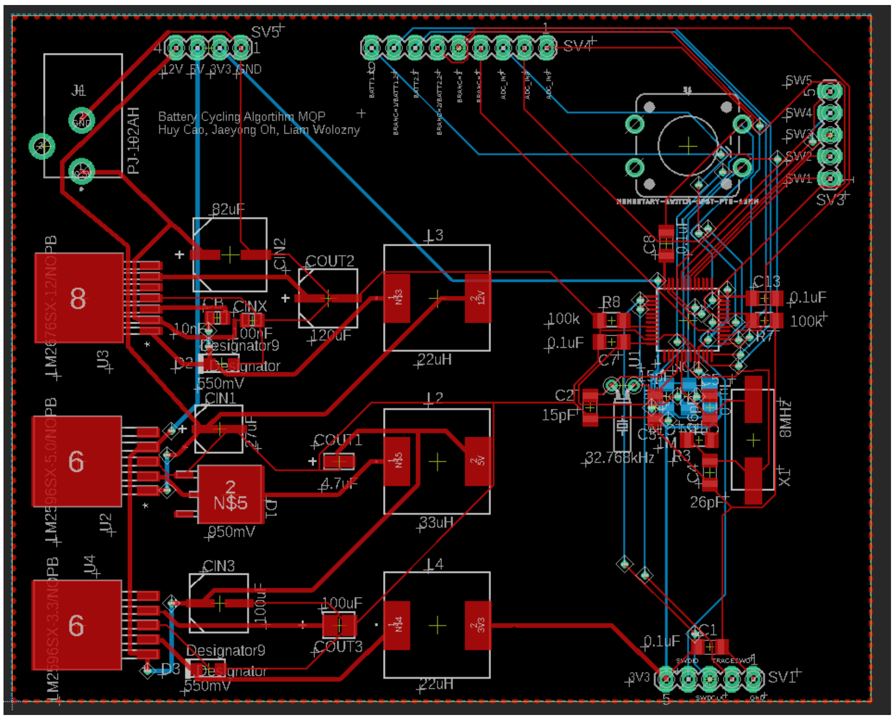


Figure 34. PCB Design of DC-DC Converter and MCU (no ground plane) (EAGLE, 2021)

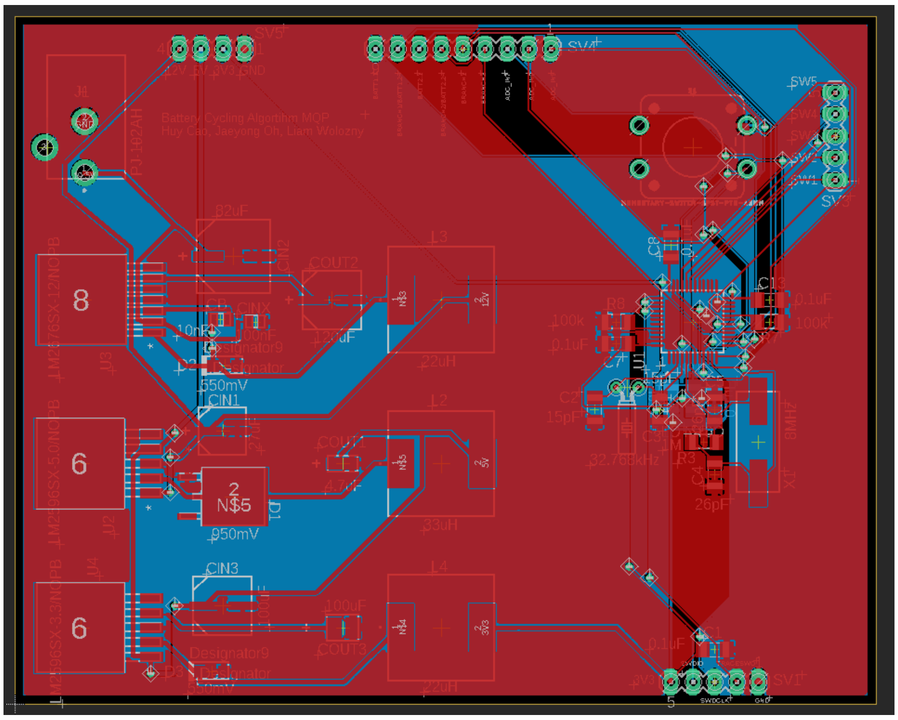


Figure 35. PCB Design of DC-DC Converter and MCU (with ground plane) (EAGLE, 2021)

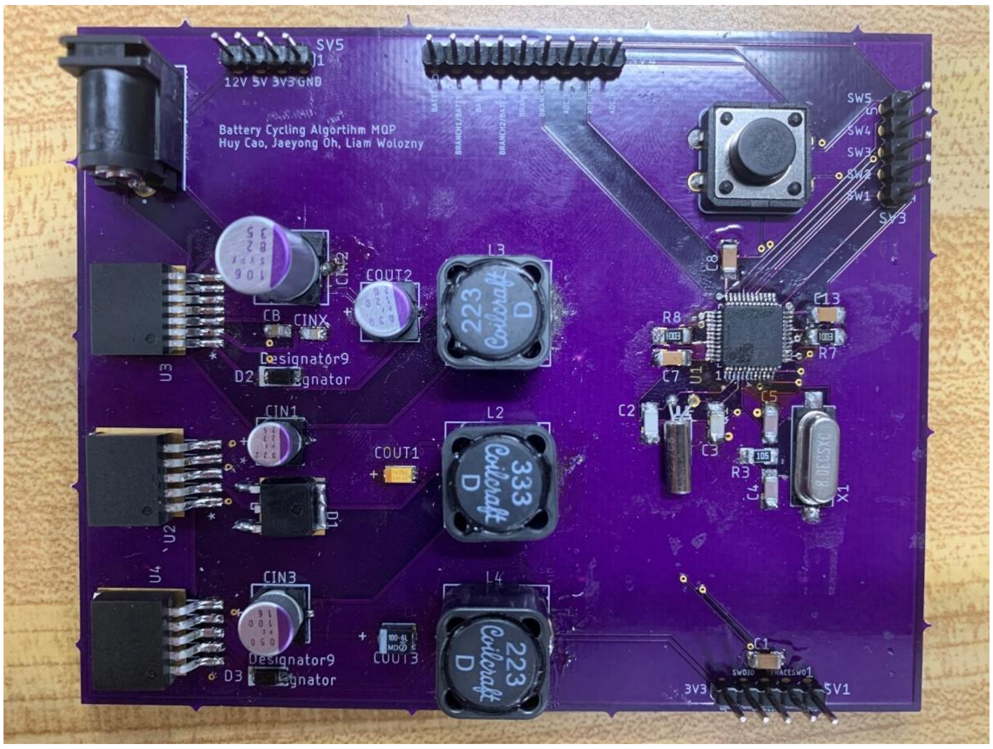


Figure 36. Complete PCB Board (components soldered)

5.3.3. PCB Design of the Main Battery Switching Circuit

The PCB design of the battery switching circuit was done on two separate boards, with one main board and one additional board for batteries and the battery holders. This section shows the design of the main board which includes the main components of the switching circuit such as switches, charging IC, instrumentation amplifier, DC-DC converter, and current sensor.

5.3.3.1. Component Placement

Component placement of the main battery switching circuit was done following the same procedure explained in section 5.3.2.1.

5.3.3.2. Tracing

Tracing of the main battery switching circuit was done following the same procedure explained in section 5.3.2.2. The only difference is that due to a more generous spacing available, the trace thickness for the main current path is set to 60 mil. This trace will be capable of carrying about 5.4A with a copper thickness of 2 oz/ft^2 (Advanced Circuits, 2018). A complete PCB design of the battery switching circuit is shown in the below figures. Figure 37 is a completed design of the main board, and Figure 38 is the same board with ground planes added on both layers. Finally, Figure 39 is the preview image of the board provided by the board manufacturer before the actual production.

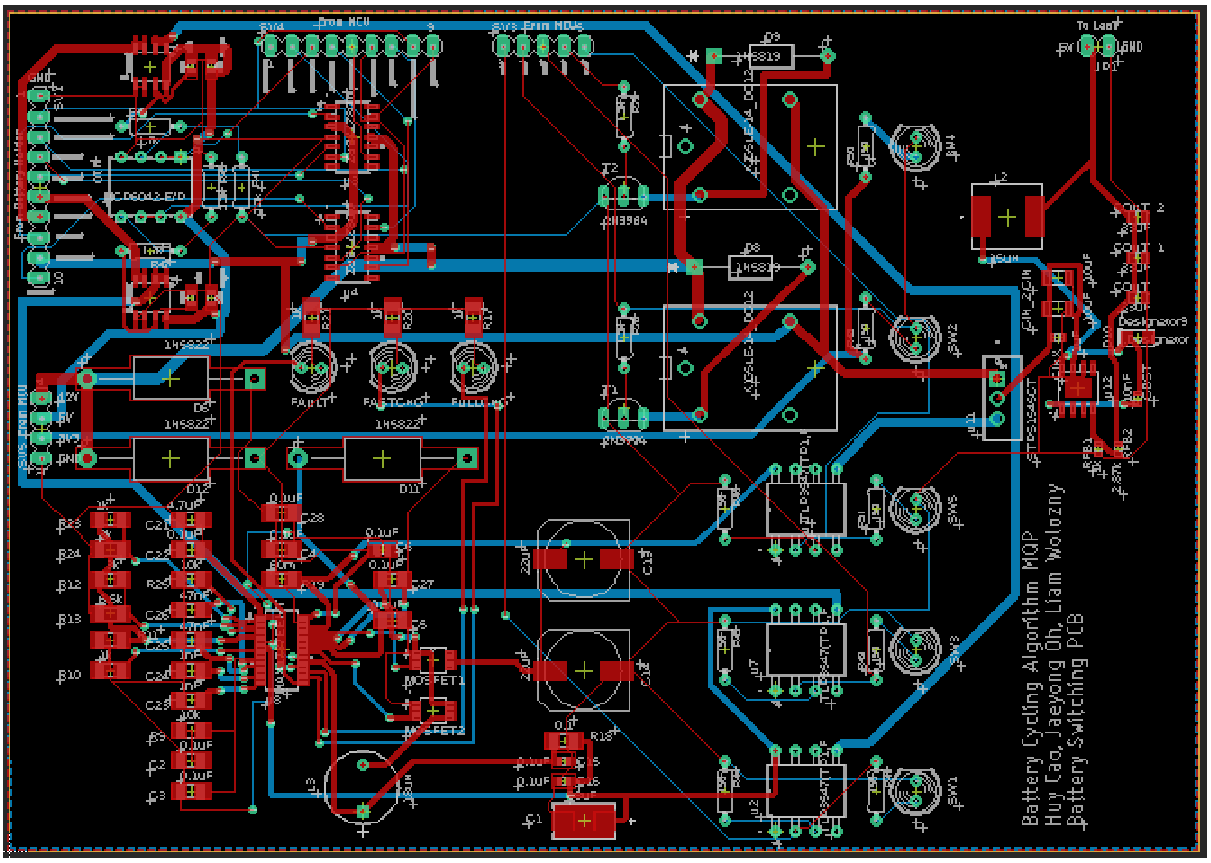


Figure 37. PCB Design of the Main Battery Switching Circuit (no ground plane) (EAGLE, 2021)

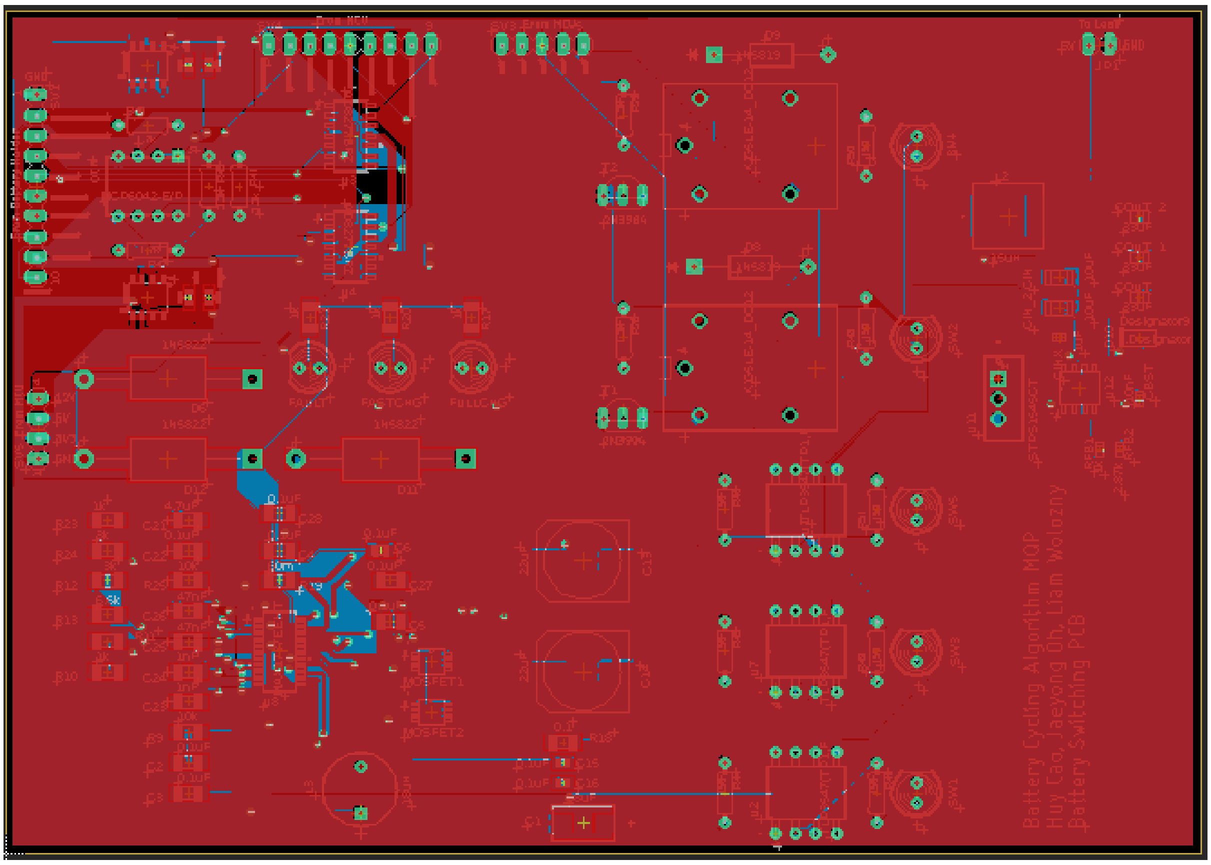


Figure 38. PCB Design of the Main Battery Switching Circuit (with ground plane) (EAGLE, 2021)

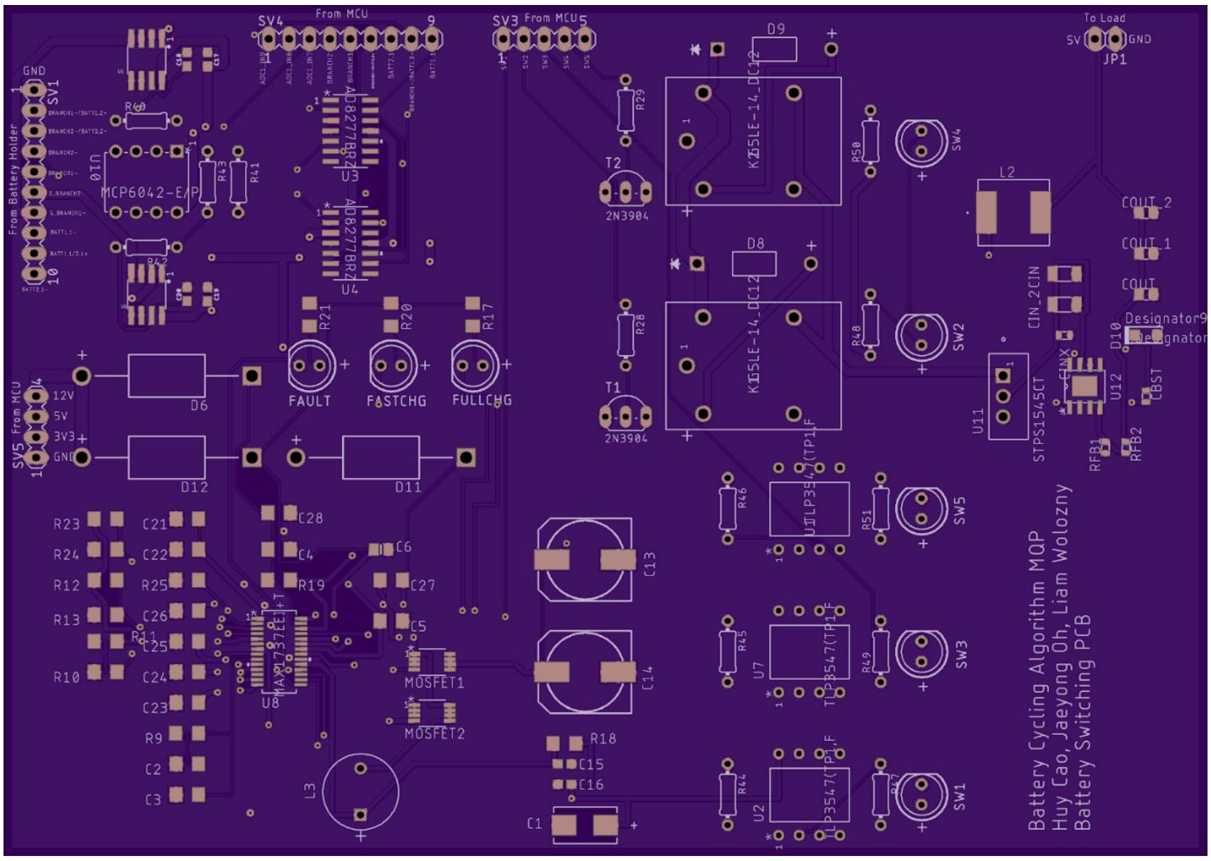


Figure 39. Preview of the Main Battery Switching PCB Board (OSH Park, 2021)

5.3.4. PCB Design of the Battery Holder Circuit

The additional battery holder circuit only has the holders as its major component, since the size of the holders were too large to fit in the main board. Later on, the main circuit and the holder circuit are connected via wires plugged into pin headers to form a complete battery switching circuit.

5.3.4.1. Component Placement

Component placement of the battery holder circuit was done following the same procedure explained in section 5.3.2.1.

5.3.4.2. Tracing

Tracing of the battery switching circuit was done following the same procedure explained in section 5.3.2.2. Similar to the battery switching circuit, the trace width for the main current carrying path in this board will have a thickness of 60 mil, which is capable of carrying 5.4 A of current at a copper thickness of 2 oz/ft^2 (Advanced Circuits, 2018). Figure 40 shows the completed design of the battery holder board, and Figure 41 shows the same board with ground planes added on both layers. Finally, Figure 42 is the preview image of the board provided by the board manufacturer before the actual production.

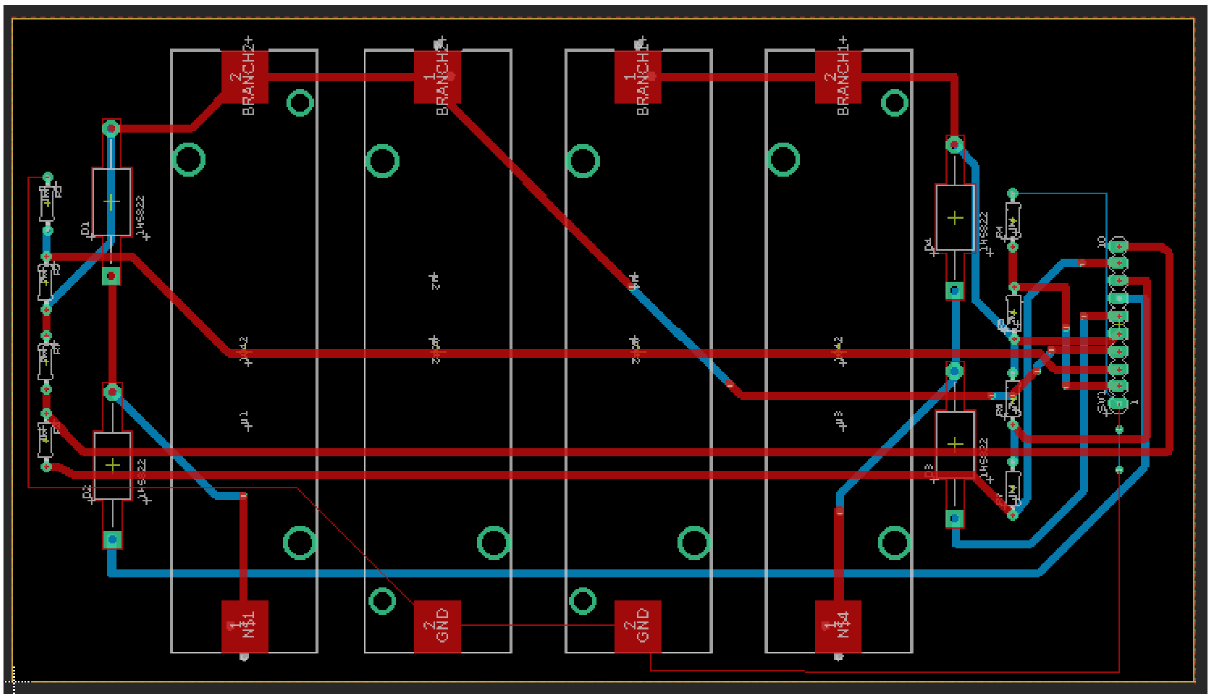


Figure 40. PCB Design of the Battery Holder Circuit (no ground plane) (EAGLE, 2021)

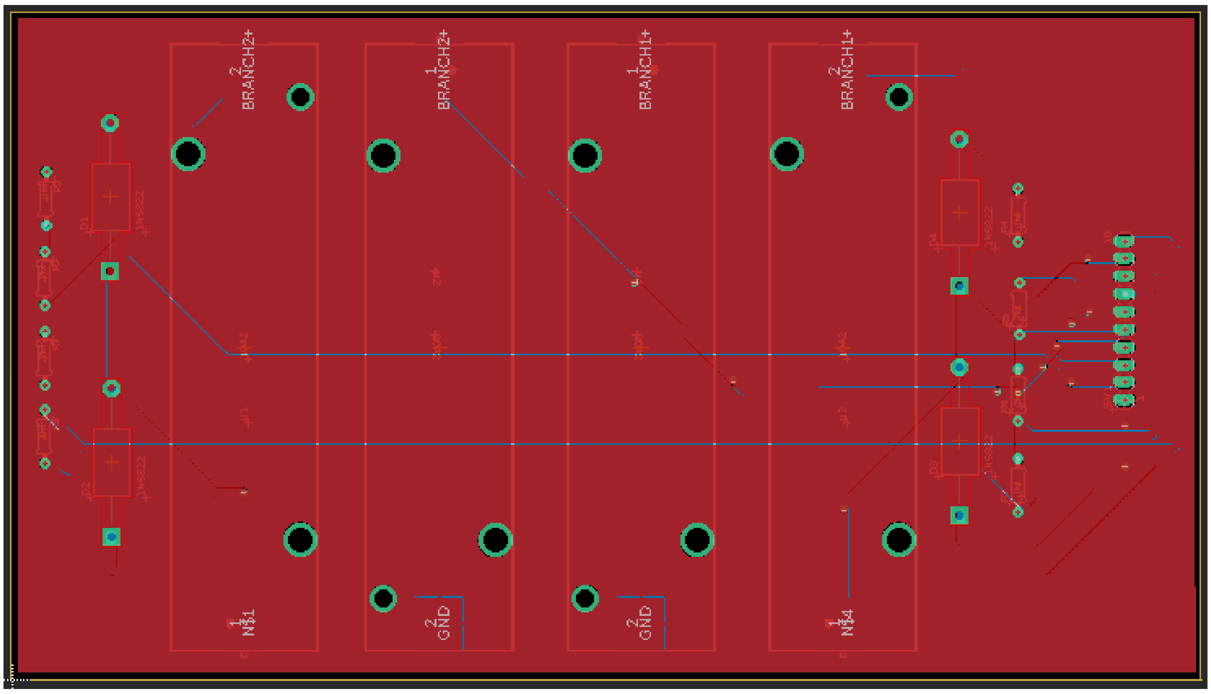


Figure 41. PCB Design of the Battery Holder Circuit (with ground plane) (EAGLE, 2021)

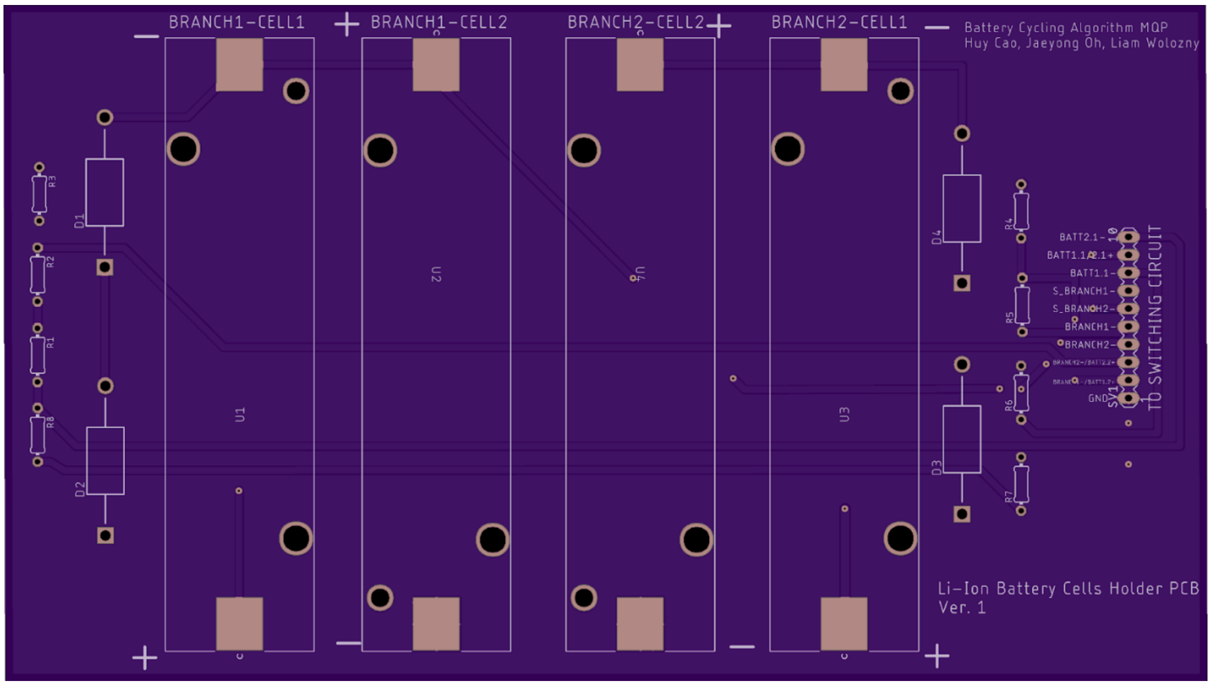


Figure 42. Preview of the Battery Holder PCB Board (OSH Park, 2021)

5.3.5. MCU Programming

The STM32F103C8T6 microcontroller that we used in this project will be programmed through the JTAG/SWD serial debugging pins of the microcontroller using the ST-Link-V2, a serial programmer for 8-bit and 32-bit ST MCU and MPU. The programmer is shown in Figure 43 below.

Figure 43. ST-Link-V2 programmer (ST Microelectronics, 2021)

For the code of this microcontroller, we use a library released by ST Microelectronics called Hardware Abstraction Layer (HAL). This library is high-level enough to shorten the code development time while keeping the low-level features to give the user control over specific registers for high-speed, low-latency operation, thus justifying our choice. Also, using this library gives us the chance to utilize an ST-developed, graphical-based software called STM32CubeMX. This software allows us to select the desired settings for the MCU graphically and generate the hardware initialization code using the HAL library. Figure 44 below shows the user interface of the software, which shows the pin assignments of the MCU that we set up for this project.

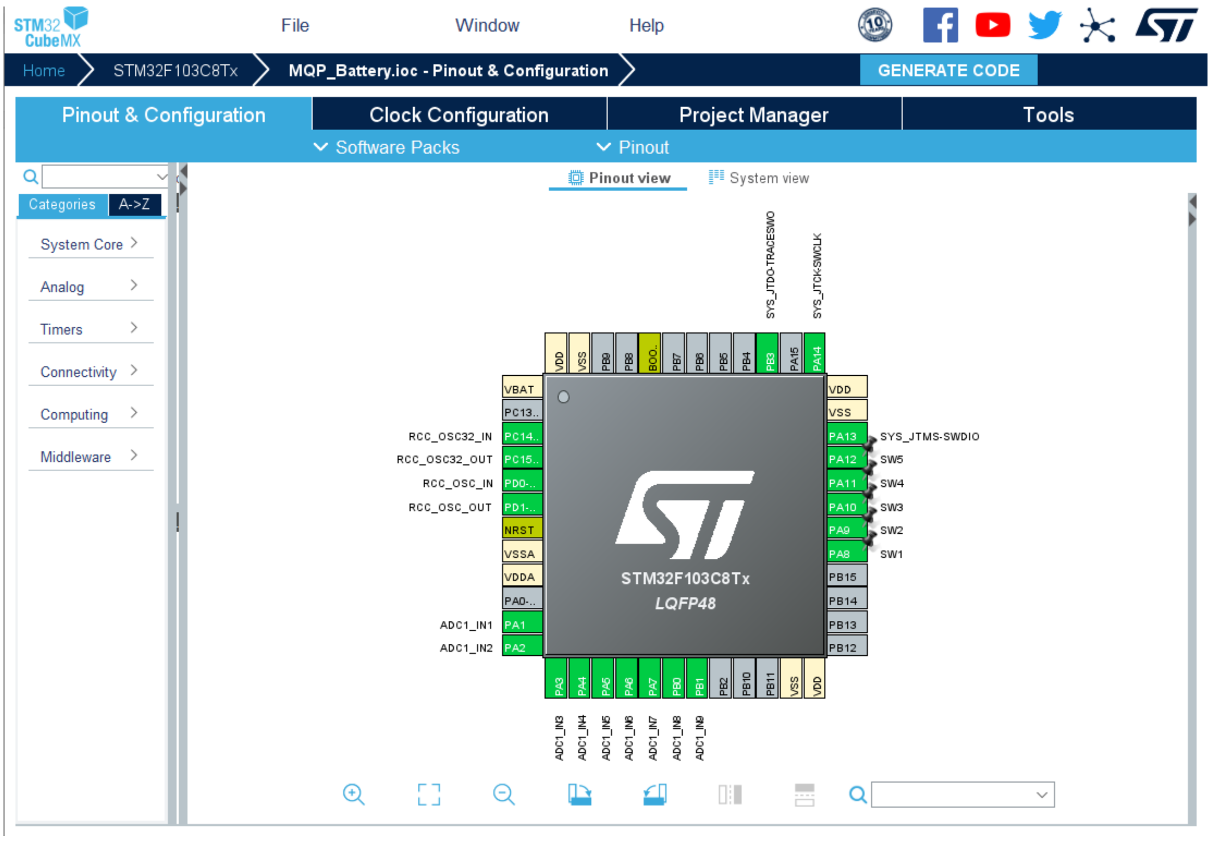
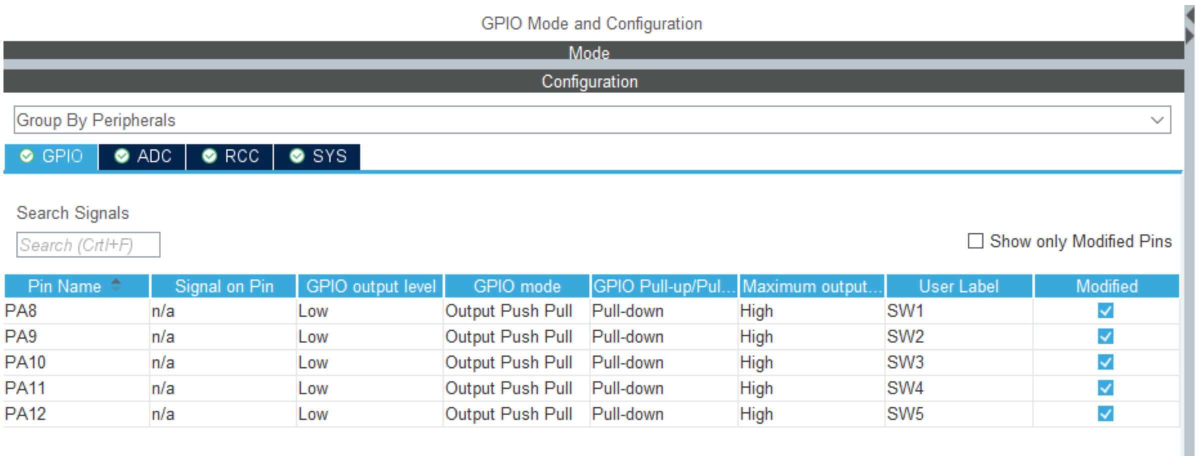


Figure 44. User interface of the STM32CubeMX and pin assignments of the MCU (STM32CubeMX, 2021)

As it can be seen from Figure 43, we set up a total of 9 ADC channels from ADC1_IN1 to ADC1_IN9 for voltage and current measurement readings, 5 GPIO pins SW1 to SW5 to control the battery switches, 4 pins RCC_OSC32_IN, RCC_OSC32_OUT, RCC_OSC_IN, RCC_OSC_OUT for our 32.768 kHz and 8 MHz crystal oscillators, and 3 serial debugging pins SYS_JTDO-TRACESWO, SYS_JTCLK-SWCLK and SYS_JTMS-SWIO. Figure 45 and 46 below shows the detailed settings of the GPIO and ADC settings in STM32CubeMX.


GPIO Mode and Configuration

Mode

Configuration

Group By Peripherals

GPIO | ADC | RCC | SYS

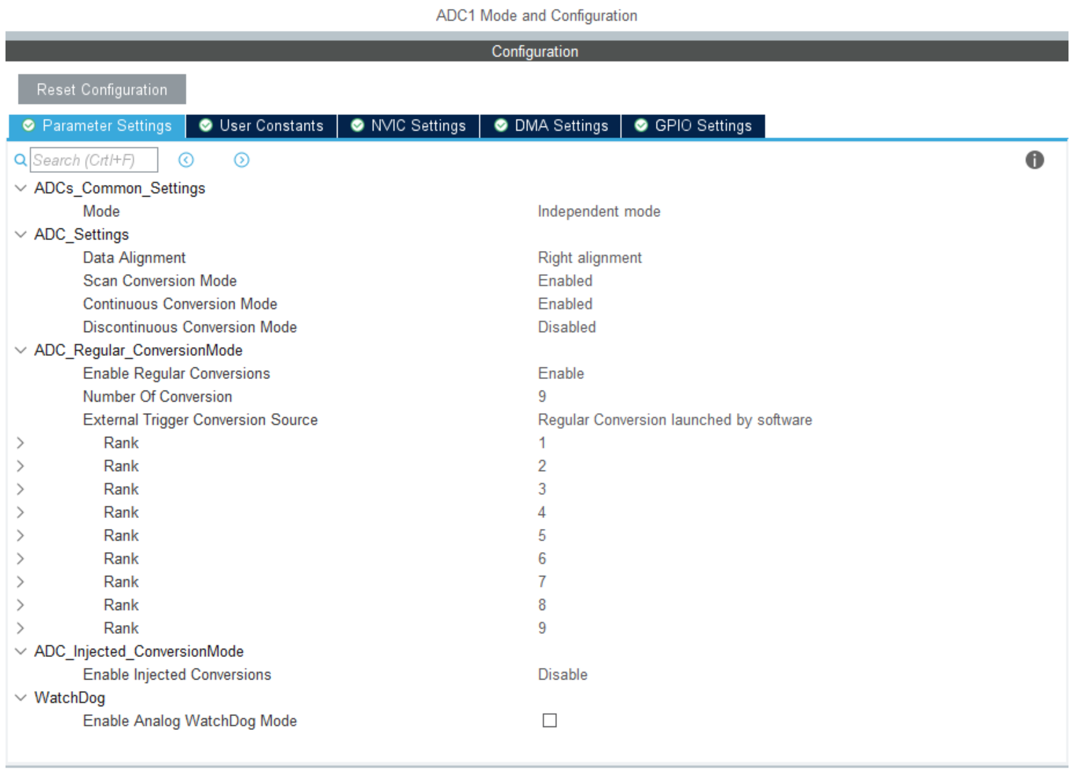
Search Signals

Search (Crtl+F)

☐ Show only Modified Pins

Pin Name	Signal on Pin	GPIO output level	GPIO mode	GPIO Pull-up/Pul...	Maximum output...	User Label	Modified
PA8	n/a	Low	Output Push Pull	Pull-down	High	SW1	✓
PA9	n/a	Low	Output Push Pull	Pull-down	High	SW2	✓
PA10	n/a	Low	Output Push Pull	Pull-down	High	SW3	✓
PA11	n/a	Low	Output Push Pull	Pull-down	High	SW4	✓
PA12	n/a	Low	Output Push Pull	Pull-down	High	SW5	✓

Figure 45. GPIO settings for the MCU in STM32CubeMX (STM32CubeMX, 2021)


ADC1 Mode and Configuration

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

Search (Crtl+F)

ADCs_Common_Settings	
Mode	Independent mode
ADC_Settings	
Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Enabled
Discontinuous Conversion Mode	Disabled
ADC_Regular_ConversionMode	
Enable Regular Conversions	Enable
Number Of Conversion	9
External Trigger Conversion Source	Regular Conversion launched by software
Rank	1
Rank	2
Rank	3
Rank	4
Rank	5
Rank	6
Rank	7
Rank	8
Rank	9
ADC_Injected_ConversionMode	
Enable Injected Conversions	Disable
WatchDog	
Enable Analog WatchDog Mode	☐

Figure 46. ADC settings for the MCU in STM32CubeMX (STM32CubeMX, 2021)

The software also allows us to set up the clock source and clock speed for the core and peripherals of the system using a clock tree setting. Figure 47 below shows the clock tree setup that we have in STM32CubeMX.

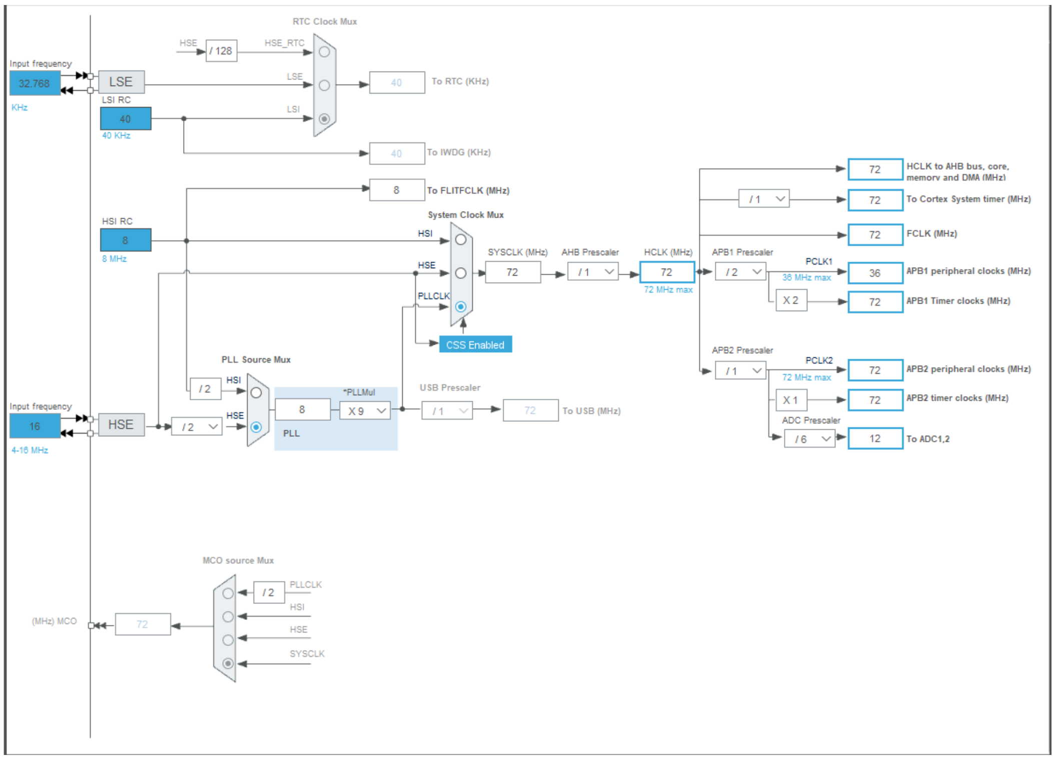


Figure 47. Clock tree settings for the MCU in STM32CubeMX

After finishing up the settings in STM32CubeMX, we generate the code and use another software called Keil µVision® 5 IDE to edit, compile and download the code to the MCU. This software is an industry-standard and popular software among the ARM microcontroller developers with a lot of support forums and resources available, thus explaining our choice of the software.

The following project structured was used in Keil µVision® 5 IDE: a main.c file containing the hardware initialization code and the main() function, a file called soc_estimation.c containing the necessary functions for the battery switching and SOC estimation, and a header file called soc_estimation.h for macro and function definition. This structure is shown in Figure 48 below. Note that the 2 files stm32f1xx_it.c and stm32f1xx_hal_msp.c are for the operation of the mentioned HAL library.

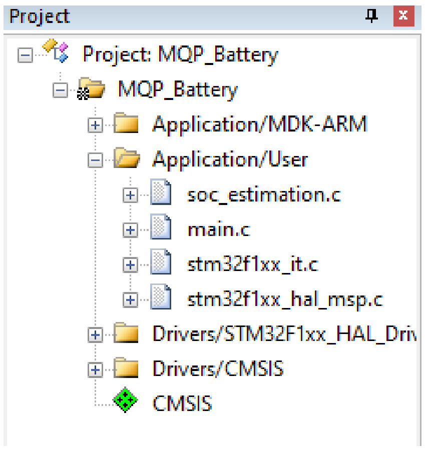

Project
Project: MQP_Battery
MQP_Battery
Application/MDK-ARM
Application/User
soc_estimation.c
main.c
stm32f1xx_it.c
stm32f1xx_hal_msp.c
Drivers/STM32F1xx_HAL_Driv
Drivers/CMSIS
CMSIS

Figure 48. Project structure in Keil uVision 5 (µVision® IDE, 2021)

The following table contains the list of functions that we implement in our MCU programming to carry out the battery cycling algorithm and battery protection functionalities. The detailed microcontroller code in C for those functions locates in the Appendix A: Microcontroller code section.

Function name	**Description and IO**
Read cell voltages	Description: Read the voltages of each battery cell (denoted cell 1, cell 2...) Input: Battery cell to be read the voltage from Output: Cell voltage
Read branch voltages	Description: Read the voltages of each battery branch (denote Branch 1, Branch 2…) Input: battery branch to be read the voltage from Output: Branch voltage
Read branch current	Description: Read the current from the branch Input: Battery branch to be read the current from Output: Branch current
Calculate initial battery SOC	Description: Determining the initial State of Charge of the battery on system initialization Input: Battery branch Output: Initial State of Charge of each branch
Estimate battery SOC	Description: Executing the battery SOC estimation algorithm (Enhanced Coulomb Counting method) based on the battery cell/branch voltage and current Input: battery branch for SOC and Depth-of-Discharge (DOD) calculation Output: State of Charge of the battery branch
Branch 1 charge	Description: Charge branch 1 by outputting logic high/low through GPIO and control the appropriate switch
Branch 1 discharge	Description: Discharge branch 1 by outputting logic high/low through GPIO and control the appropriate switch
Branch 2 charge	Description: Charge branch 2 by outputting logic high/low through GPIO and control the appropriate switch
Branch 2 discharge	Description: Discharge branch 2 by outputting logic high/low through GPIO and control the appropriate switch
Battery pack discharge	Description: Charge the whole battery pack by outputting logic high/low through GPIO and control the appropriate switch
Battery pack charge	Description: Discharge the whole battery pack by outputting logic high/low through GPIO and control the appropriate switch
Overvoltage/undervoltage protection	Description: Protect the battery pack from overvoltage/undervoltage situation by monitoring the voltage and control the appropriate switch
Overcurrent protection	Description: Protect the battery pack from overcurrent situation by monitoring the current and control the appropriate switch

Table 2. List of implement functions for the microcontroller

6. System testing and result

6.1. DC-DC Converters testing

The testing of the DC-DC converter was performed by using a simple digital multimeter (DMM) and power supply. Before supplying any power to the board, a continuity test was performed using DMM to make sure that nothing is shorted in the circuit. Then a power supply in the lab was used to supply a voltage of 19.5 V with limited current (1A) for a safer testing before using an actual adapter which draws a maximum current value of 3A. The pin headers for each converted voltage values (12V, 5V, 3.3V and GND) were on the board, which made it easier to connect the DMM nodes onto to measure the voltages.

The DMM successfully read 11.95 V, 5.005V, and 3.26 V, respectively, for the output voltage of each stage; this meant that there will not be any problem if tested with the AC-DC adapter connected. Therefore, another test was performed with the adapter connected. Once again, the DMM read the same desired voltage values, which confirmed that the DC-DC converters are working properly.

6.2. Microcontroller Circuit testing

The testing of the microcontroller circuit consists of two main parts. Those are verifying the functionality of the circuit on the PCB design and the performance of the written MCU program.

To verify the functionality of the MCU circuit, first the communication between the laptop and the microcontroller PCB through JTAG/SWD serial debugging was tested. After plugging in the ST-Link-V2 serial programmer to the JTAG/SWD debug pins on the microcontroller PCB and to the USB port of the laptop, the debug setting menu in Keil µVision® 5 IDE recognized the MCU as an ARM core along with its serial code, signaling a successful communication between the software and the MCU through the JTAG/SWD debugging pins. Next, the hardware initialization code for the MCU was run in debug mode to see if the code was run into any hardware or startup Error Handling breakpoints and signal a potential fault in hardware setups or connections. The result showed no Error Handling breakpoint being run into, showing that the connection of hardware and peripherals such as boot pins, reset pins, oscillators… was correct. Finally, to verify that the physical connection on the board of the selected GPIO and ADC pins was correct, a piece of testbench code was written so that each pin were pulled high and then low successively, with the signal at each pin being monitored using the oscilloscope. The result showed a correct representation of the output signal for each pin being monitored.

The performance of the written MCU code was also verified, specifically the readings of the microcontroller's built-in ADC and the current conversion result based on the ADC readings. First, to test the accuracy of the ADC reading, a simple voltage divider circuit consisting of a fixed 10 kΩ and a 10 kΩ potentiometer. The output of this voltage divider is then connected to each of the 9 ADC input channels. This test circuit is shown in Figure 49 below.

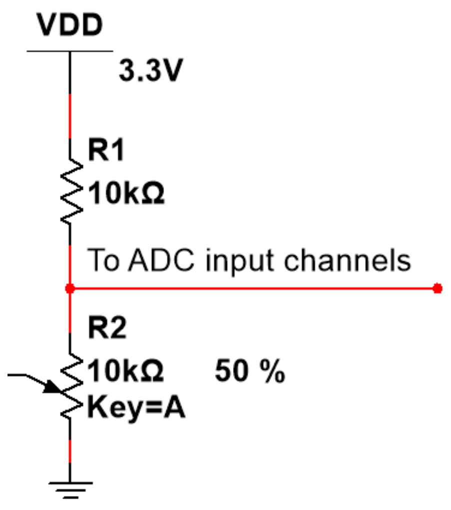


Figure 49. Test circuit for the ADC readings (Multisim, 2021)

The reading of this output voltage from the ADC is then compared to the reading using the lab bench digital multimeter (DMM), which was used as the standard measurement to compare the ADC readings to. The potentiometer knob was changed after each complete measurement so that 3 test cases with 3 different output voltages could be measured. Table 3 to 5 below shows the measurement results of the ADC readings compared to the lab bench DMM readings in each of the 3 cases.

ADC Channel	DMM Measurement	ADC Measurement	Error
CH1	3.25	3.299	1.5%
CH2	3.25	3.292	1.292%
CH3	3.25	3.295	1.385%
CH4	3.25	3.292	1.292%
CH5	3.25	3.299	1.5%
CH6	3.25	3.293	1.32%
CH7	3.25	3.299	1.5%
CH8	3.25	3.299	1.5%
CH9	3.25	3.299	1.5%

Table 3. ADC readings vs DMM readings for the first case where output voltage is about 3.3V

ADC Channel	DMM Measurement	ADC Measurement	Error
CH1	2.817	2.88	2.236%
CH2	2.817	2.889	2.55%

CH3	2.817	2.868	1.81%
CH4	2.817	2.873	1.98%
CH5	2.817	2.876	2.094%
CH6	2.817	2.916	3.514%
CH7	2.817	2.868	1.81%
CH8	2.817	2.867	1.77%
CH9	2.817	2.864	1.668%

Table 4. ADC readings vs DMM readings for the second case where output voltage is about 2.8V

ADC Channel	DMM Measurement	ADC Measurement	Error
CH1	0.423	0.450	6.38%
CH2	0.423	0.392	7.326%
CH3	0.423	0.460	8.747%
CH4	0.423	0.371	12.3%
CH5	0.423	0.506	19.6%
CH6	0.423	0.402	4.96%
CH7	0.423	0.427	0.945%
CH8	0.423	0.431	1.89%
CH9	0.423	0.431	1.89%

Table 5. ADC readings vs DMM readings for the third case where output voltage is about 0.4V

From the result, the error between the ADC readings and the actual DMM readings were derived. For the first case, the deviation ranges from 1.29% to 1.5%. For the second case, the deviation ranges from 1.66% to 3.5%. For the third case, the deviation ranges from 0.945% to 19.6%. It can be observed that at a lower voltage level, the ADC readings become more erroneous and less accurate. This is expected as the lower the voltage amplitude, the more susceptible the signal is to the noise, as the noise also has a low voltage amplitude. However, the typical voltage range going to the ADC channels would be about 1V – 2.8V based on the specification of the battery voltage and current being monitored. Therefore, we can take into consideration the error of ADC readings at this expected voltage ranges, which runs from 1.5% to 3.5%. This error percentage is lower than the acceptable reading deviation of 5%, which shows that the ADC readings of the MCU is accurate enough for the requirements.

6.3. Load Stage Testing

The load stage as first presented in Figure 33 was breadboarded rather than fabricated into a PCB due to long manufacturer delays. The LED rapid switching did not function as expected. Instead of having each LED in the circuit switch rapidly from on to off, we had a trio of LEDS that would alternate between being fully on and slightly flickering, while the rest would only flicker. This caused no impact in current draw or short-circuiting issues as we detected a solid 100mA current draw. This would fully discharge one of our battery branches if the same was disconnected from the wall outlet and running solely under its own power, in 26.5 hours given our 2650mAh total branch battery capacity. The output clock signal for the LM555 Timer is reduced by 0.7V due to the protective diode at the LM555 input. This results in a clock signal of 4.3Vpk. The counters generally function as expected, however internal noise within these caused some flickering and erratic switching. The circuit, though behaving oddly, functions as an effective and dynamic load capable of demonstrating the capabilities of our system.

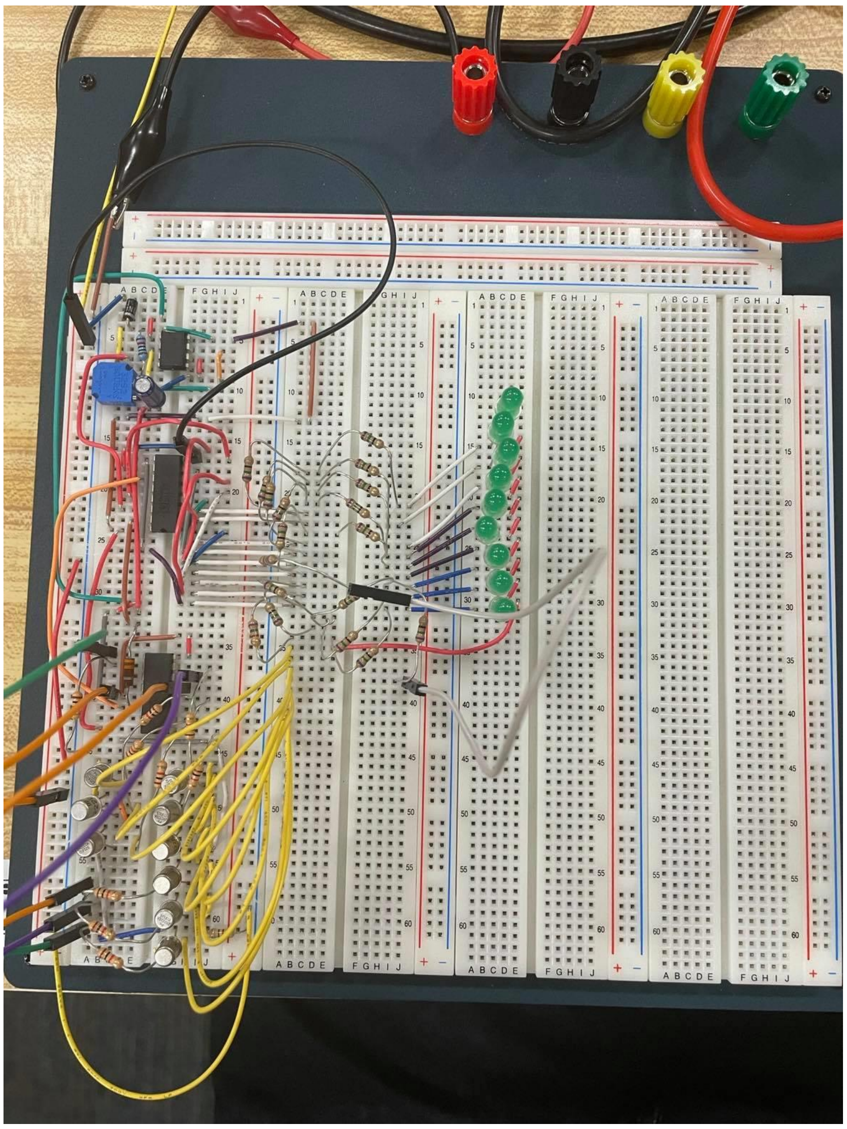


Figure 50. Breadboarded Load Stage

7. Future Recommendation

7.1. Functionality testing of the Battery Switching and Battery Holder PCBs

Due to the current pandemic situation and disruption in global supply chain, there was a significant delay in the turn out time on the manufacturer side for the Battery Switching and Battery Holder PCBs that we designed. Therefore, the team did not have enough time to assemble, test and debug these two PCBs as well as integrate them to the rest of the system.

That said, future work on this project is needed to complete the assembly of these two boards, as well as to perform functionality testing for each individual boards. Additionally, the two boards would need to be integrated with the existing board that contains the DC-DC converters and the microcontroller circuit. Completion of these tasks would mark the successful implementation of the proof-of-concept prototype for the proposed algorithm, as well as enable the performance testing of the algorithm in terms of battery life and integration into laptops.

7.2. Cell Balancing Circuit

For this project, we decided not to implement the cell-balancing circuit since most circuits using cell-balancing IC that are available on the market involve transformers which can be problematic in their sizes for proper spacing. Even some of the simplest cell-balancing designs such as charge shuttles (active balancing with flying capacitors) have their drawbacks coming from their simplicity such as the charge/discharge process being relatively slow and having energy losses at the capacitors. Other cell-balancing designs such as inductive converter circuit could have been used as well, but the complexity of the design would have added difficulty in the overall completion of the project within the given time frame.

However, it is recommended that cell-balancing feature is added in the design for future reference after all. This is because there will always be an imbalance between the two cells in the branch caused by factors such as SOC imbalance and internal resistance difference. Cell-balancing will be more important in the long run since batteries will degrade over time and the imbalance of the SOC will get worse; this imbalance of SOC is likely to cause issues such as overcharging/heating, undercharging, and degradation which will decrease the overall battery life.

7.3. Battery Life Testing

Given the arguably short scope of this project (21 weeks), we are unable to test whether battery life can be prolonged using our prototyped method. As the batteries must be continuously charged and discharged many hundreds and even thousands of times before any change can be reliably measured, this is work that will be left for future members of this MQP. Companies capable of testing this project for the full duration of time a project of this scope demands may very well find themselves with a very useful module that could be installed in larger portable electronic devices.

It is recommended that future work on this device continue in the form of devising a battery life testing methodology. Whether by testing the battery for aberrations in its float voltage, or by discharge testing, work should be done to prove or disprove our hypothesis.

7.4. Protective mechanism for the prototype

Through the testing and debugging stage of this project, it was found that our prototype is vulnerable to short-circuit between the input supply voltage traces and ground. In the case of this event the microcontroller, as well as some other ICs such as the current sensor or the instrumentation amplifier would be damaged since there is no internal short-circuit protection for those ICs. Also, it is not ideal to put the DC-DC converters, which consists of voltage traces as high as 19.5V, on the same PCB as the microcontroller circuit, as any spike or short at those traces would damage not only the MCU, but also any other devices communicating with the microcontroller through JTAG/SWD debugging.

It is recommended that future works on this prototype adds in a short-circuit protective mechanism at the supply voltage rails of the board, such as using Schottky or rectifier diodes at the input supply traces. Also, to address the problem of having high-voltage traces on the same PCB as the MCU, digital isolators could be used to isolate any problems at the DC-DC converters or the input supply without damaging the MCU or any devices communicating with the MCU.

7.5. System integration to laptop

Given the time constraints of this project, the goal of this project was set to only build a proof-of-concept prototype for our proposed battery cycling algorithm. However, as this project is aimed towards laptop power system application, it is more desirable to be able to implement this system and verify the performance on an actual laptop. Therefore, future iterations of this project should look into the possibility of performing such implementation.

A few modifications would need to be made to allow for this integration. First, a Central Processing Unit (CPU) would need to be used in place of an MCU. As the CPU is readily available inside the laptop, using the laptop's CPU to implement the cycling algorithm would provide for a robust solution that would save on the additional footprint and cost associated with an external MCU, as well as keep the software code developed by the manufacturer consistent. Second, any type of integrated switches inside the laptop battery pack would be preferred over the solid-state or mechanical relays. For a proof-of-concept prototype, solid-state and mechanical relays prove sufficient, but in an actual application those relays would take up too much space and would also wear out over extended uses. Therefore, it is suggested that a more robust, fast-switching integrated switch be implemented inside the battery pack in place of these relays. Finally, the system needs to drive actual loads inside the laptop, such as CPU, Graphics Processing Unit (GPU), Random Access Memory (RAM) or hard drives. Success in driving those loads would prove the capability of this system, which would open the possibility for laptop manufacturers to apply this algorithm into commercial products.

7.6. Other rechargeable application implementation

Once more the time constraints of this project did not allow for variations in terms of our implementation. Regardless, future work on this project should continue on a myriad of smart devices. Anything with enough processing power such as a smart phone could benefit greatly from our algorithm. One of the major problems that surround cell phone batteries nowadays is damage to their batteries caused by poor charging practices. Using our algorithm to switch between battery cells within the phones themselves could prove beneficial in greatly increasing overall battery life. This algorithm could potentially be applied to every kind of personal portable device available, helping to circumvent human error by charging malpractice, and increasing the longevity of these rechargeable devices.

8. Conclusion

Over the course of 6 months, the team was able to accomplish most of the project objectives laid out in the beginning. We as a team were able to research and propose an algorithm to cycle the laptop battery branches in the State of Charge range of 100%-90%. On top of that, the battery switching circuit, along with the microcontroller-based control circuit to carry out this algorithm was successfully designed. In addition, printed circuit boards for the battery switching circuit and the control circuit were successfully printed out and assembled to meet the goal of building a proof-of-concept prototype for our proposed algorithm and serve as the deliverables for this project. A test load stage circuit was also designed and assembled on the breadboard to test the functionality of the prototype. However, because of time constraints and resource limitations due to the global pandemic situation, the team was not able to carry out the full functionality testing of the whole prototype.

Future work on this project is necessary to complete the assembly and testing of the battery switching and battery holder boards, as well as to integrate these two boards to the rest of the current system. Additionally, future continuation of this project is welcome to make possible improvements upon the design of the current prototype, such as implementing a cell-balancing circuit and a short-circuit protective mechanism. Moreover, it is recommended that any future iterations of this project to work on possible methods to perform the battery life testing of this prototype and compare it to the current battery management algorithm in laptops. Lastly, potential implementation of this algorithm into an actual laptop needs to be looked into. If the implementation is successful, this algorithm could be applied to commercial products, which would be beneficial to both the customers and the environment.

9. Reference

4infor (n.d.). *Block Diagram for a Typical Laptop Power System* https://sites.google.com/site/4abcxyz123/laptop/battery/typical-laptop-power-battery-system-diagram

Advanced Circuits (2018). *PCB Trace Width Calculator.* https://www.4pcb.com/trace-width-calculator.html

Allegro Microsystems (2019). *ACS70331: High Sensitivity, 1 MHz, GMR-Based Current Sensor IC in Space-Saving Low Resistance QFN and SOIC packages.* https://www.allegromicro.com/en/products/sense/current-sensor-ics/zero-to-fifty-amp-integrated-conductor-sensor-ics/acs70331

ARM Keil (2021). µVision® IDE [Computer Software].

Association Connecting Electronics Industries (IPC) (2003). *Generic Standard on Printed Board Design.* https://www.ipc.org/TOC/IPC-2221A.pdf

Autodesk EAGLE [Computer Software] (2021). Retrieved from eagle.autodesk.com

Barsukov, Y. (n.d.). Battery Cell Balancing: What to Balance and How. *Texas Instruments*, https://www.ti.com/download/trng/docs/seminar/Topic%202%20-%20Battery%20Cell%20Balancing%20-%20What%20to%20Balance%20and%20How.pdf

Cao, J., & Emadi, A. (2011, March 1). Batteries Need Electronics. *IEEE Industrial Electronics Magazine, 5*(1). https://ieeexplore.ieee.org/abstract/document/5742579?casa_token=SeRU8jbYl68AAAAA:rpHs4EraYc42kL6zISoldUOqyycTYtaIwZJCG-VhLRLxOA3TTG9bfwRSwtIY386kf1Ga4q6A

Dickinson, B. & Bentley, W. (2013). Managing External Magnetic Field Interference When Using ACS71x Current Sensor ICs. *Allegro Microsystems*, https://allegromicro.com/en/insights-and-innovations/technical-documents/hall-effect-sensor-ic-publications/managing-external-magnetic-field-interference-acs71x-current-sensor-ics

Hart, D. W. (2011). Electronic switches. *Power electronics* (pp. 6-10). New York: McG raw-Hill.

Leibson, S. (2018). Fundamentals of Current Measurement: Part 1 – Current Sense Resistor. *Digikey*, https://www.digikey.com/en/articles/fundamentals-of-current-measurement-part-1-current-sense-resistors#:~:text=The%20most%20common%20sensing%20element,the%20current%20passing%20through%20it.

Lin, Q., Wang, J., Xiong, R., Shen, W., & He, H. (2019). Towards a smarter battery management system: A critical review on optimal charging methods of lithium ion batteries. *Energy, 183*, 220-234. doi:10.1016/j.energy.2019.06.128

Linden, D., & Reddy, T. B. (2001). *Handbook of batteries*. McGraw-Hill.

Manimekalai, P., Harikumar, R., & Raghavan, S. (2013). An overview of batteries for photovoltaic (PV) system. *International Journal of Computer Applications, 82*(12), 28-32. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.401.7780&rep=rep1&type=pdf#:~:text=Lead%20acid%20battery%20with%20deep,Metal%20hydride%20batteries%20are%20used.

Mathas, C. (2012). The Basics of Current Sensors. *Digikey,* https://www.digikey.com/en/articles/the-basics-of-current-sensors

Maxim Integrated (2017). MAX1737 Stand-Alone Switch-Mode Lithium-Ion Battery-Charger Controller. https://datasheets.maximintegrated.com/en/ds/MAX1737.pdf

Murnane, M., & Ghazel, A. (2017). *A Closer Look at State of Charge (SOC) and State of Health (SOH) Estimation Techniques for Batteries*. Analog Devices. https://www.analog.com/media/en/technical-documentation/technical-articles/A-Closer-Look-at-State-Of-Charge-and-State-Health-Estimation-Techniques.pdf

National Instrument (2021). Multisim [Computer Software].

OEM (n.d.). CT sensors – An Introduction. *Learn – OpenEnergyMonitor,* https://learn.openenergymonitor.org/electricity-monitoring/ct-sensors/introduction

OMRON Corporation (n.d.). G5LE PCB Power Relay – Cubic, Single-pole 10A Power Relay. https://omronfs.omron.com/en_US/ecb/products/pdf/en-g5le.pdf

OSH Park (2021). Project History. https://oshpark.com/project_history

NVE Corp (n.d.). Application Notes for GMR Sensors. https://www.nve.com/Downloads/apps.pdf

Texas Instruments. (2017, January). CSD17585F5 30-V N-Channel FemtoFET™ MOSFET.

Toshiba. (2020, March 25). Photocouplers, Photorelay TLP3547,TLP3547F.

Sedra, A. S., & Smith, K. C. (2015). Diodes. *Microelectronic circuits* (pp. 175-177). New York: Oxford University Press.

ST Microelectronics (2021). ST-LINK/V2 in-circuit debugger/programmer for STM8 & STM32. https://www.st.com/en/development-tools/st-link-v2.html

ST Microelectronics (2021). STM32CubeMX [Computer Software].

ST Microelectronics. (2015). STM32F103x8, STM32F103xB - Medium-density performance

line ARM®-based 32-bit MCU with 64 or 128 KB Flash, USB, CAN, 7 timers, 2 ADCs, 9 com. interfaces.

Wei, J., & Lee, F. C. (2004, February). Two-stage voltage regulator for laptop computer CPUs and the corresponding advanced control schemes to improve light-load performance. *Nineteenth Annual IEEE Applied Power Electronics Conference and Exposition, 2004. APEC '04.* https://doi.org/10.1109/APEC.2004.1295990

Yi-Hwa Liu, Jen-Hao Teng, & Yu-Chung Lin. (2005). Search for an optimal rapid

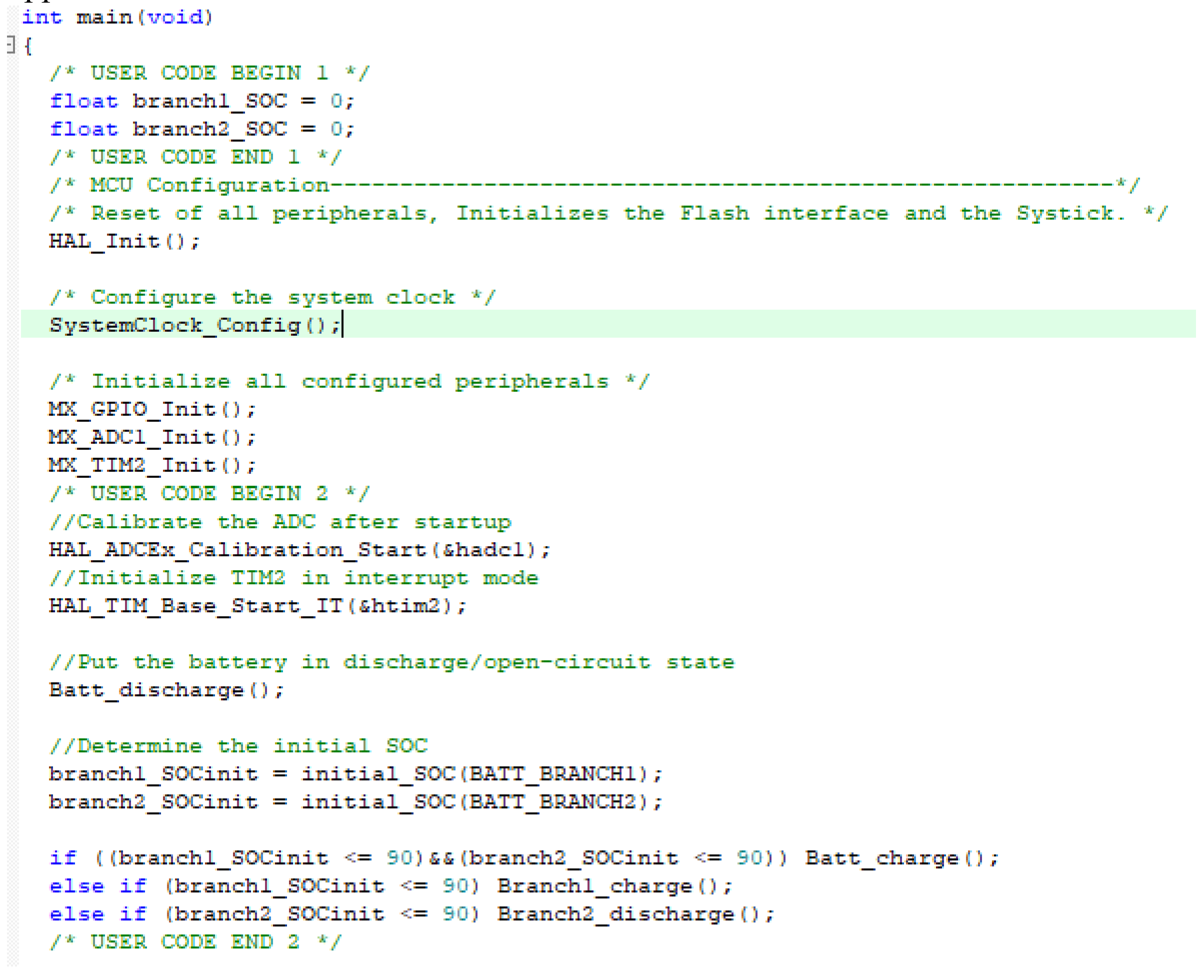
charging pattern for lithium-ion batteries using ant colony system algorithm, do i:10.1109/TIE.2005.855670

Xu, B., Oudalov, A., Ulbig, A., Andersson, G. (2016). Modeling of Lithium-Ion Battery

Degradation for Cell Life Assessment. *IEEE Transection on Smart Grid 99(2):1-1.* https://doi.org/10.1109/TSG.2016.2578950

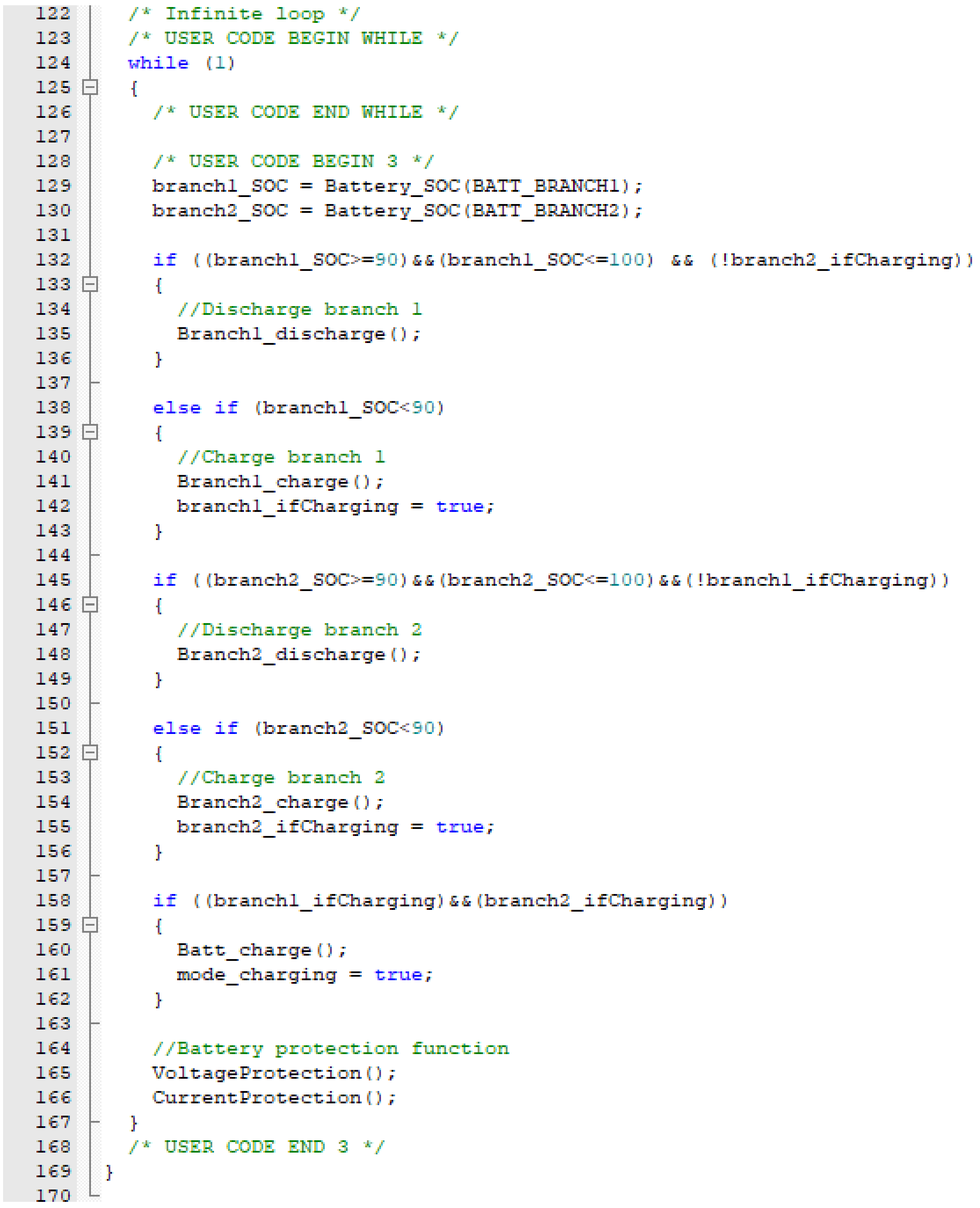
Appendix

Appendix A: Microcontroller Code



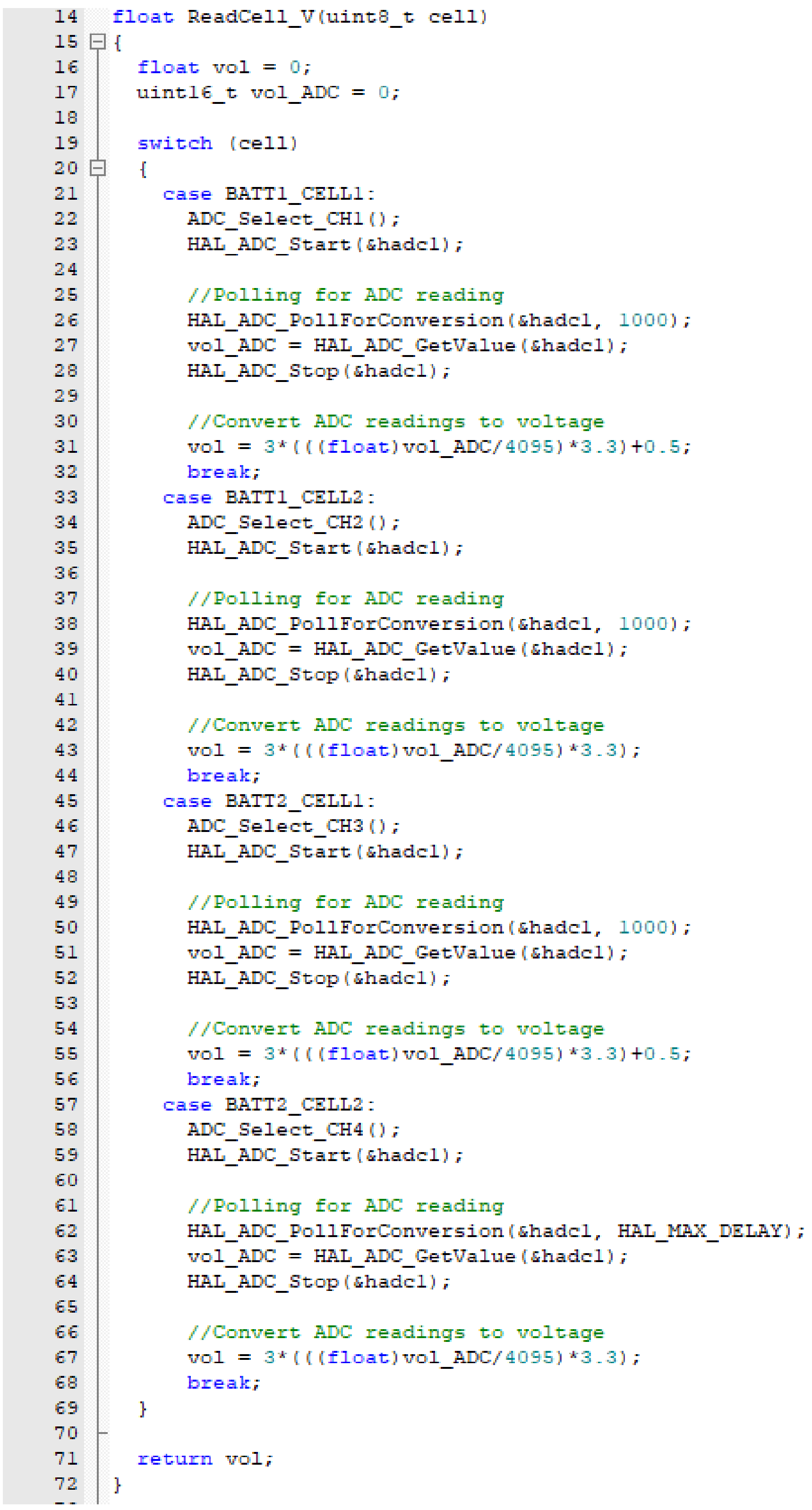
```c
int main(void)
{
  /* USER CODE BEGIN 1 */
  float branch1_SOC = 0;
  float branch2_SOC = 0;
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_ADC1_Init();
  MX_TIM2_Init();
  /* USER CODE BEGIN 2 */
  //Calibrate the ADC after startup
  HAL_ADCEx_Calibration_Start(&hadc1);
  //Initialize TIM2 in interrupt mode
  HAL_TIM_Base_Start_IT(&htim2);

  //Put the battery in discharge/open-circuit state
  Batt_discharge();

  //Determine the initial SOC
  branch1_SOCinit = initial_SOC(BATT_BRANCH1);
  branch2_SOCinit = initial_SOC(BATT_BRANCH2);

  if ((branch1_SOCinit <= 90)&&(branch2_SOCinit <= 90)) Batt_charge();
  else if (branch1_SOCinit <= 90) Branch1_charge();
  else if (branch2_SOCinit <= 90) Branch2_discharge();
  /* USER CODE END 2 */
```

Part 1 of the main() function: initialization code and initial battery State of Charge calculation (µVision® IDE, 2021)



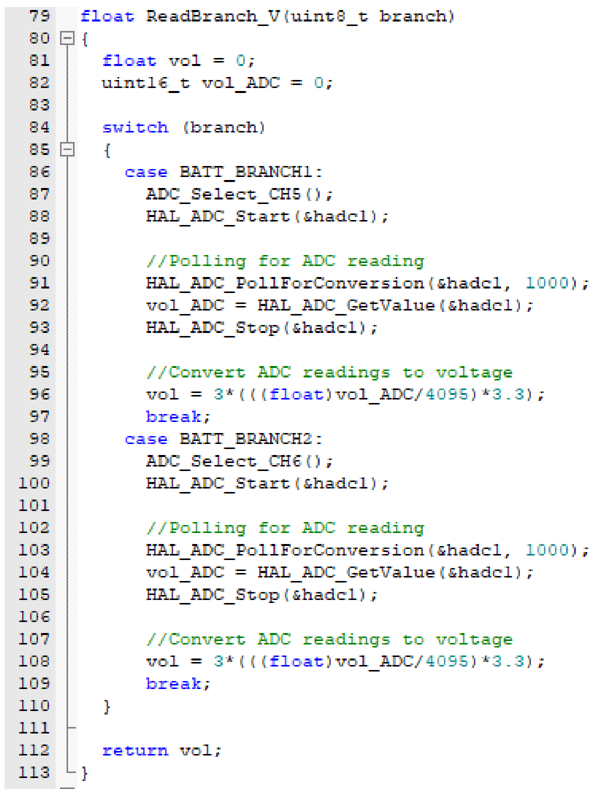
```c
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    branch1_SOC = Battery_SOC(BATT_BRANCH1);
    branch2_SOC = Battery_SOC(BATT_BRANCH2);

    if ((branch1_SOC>=90)&&(branch1_SOC<=100) && (!branch2_ifCharging))
    {
      //Discharge branch 1
      Branch1_discharge();
    }

    else if (branch1_SOC<90)
    {
      //Charge branch 1
      Branch1_charge();
      branch1_ifCharging = true;
    }

    if ((branch2_SOC>=90)&&(branch2_SOC<=100)&&(!branch1_ifCharging))
    {
      //Discharge branch 2
      Branch2_discharge();
    }

    else if (branch2_SOC<90)
    {
      //Charge branch 2
      Branch2_charge();
      branch2_ifCharging = true;
    }

    if ((branch1_ifCharging)&&(branch2_ifCharging))
    {
      Batt_charge();
      mode_charging = true;
    }

    //Battery protection function
    VoltageProtection();
    CurrentProtection();
  }
  /* USER CODE END 3 */
}
```

Part 2 of the main() function: infinite while(1) loop carrying the program flowchart (µVision® IDE, 2021)



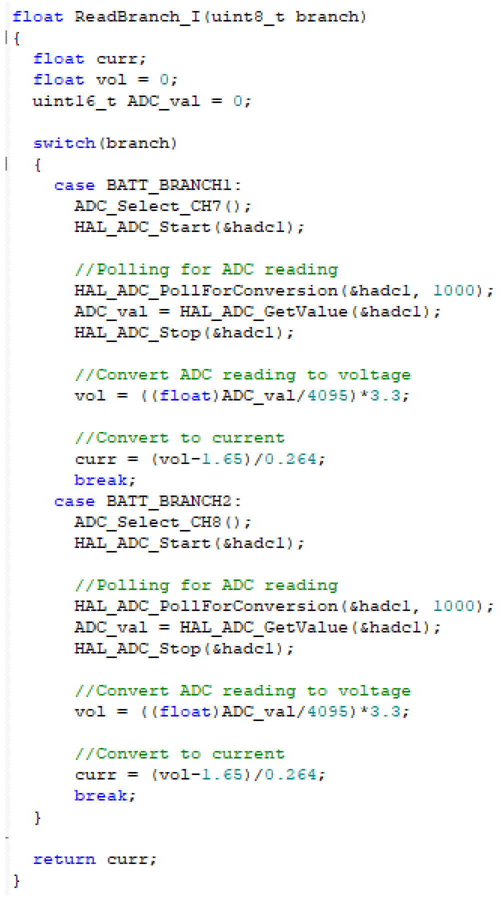
```c
float ReadCell_V(uint8_t cell)
{
  float vol = 0;
  uint16_t vol_ADC = 0;

  switch (cell)
  {
    case BATT1_CELL1:
      ADC_Select_CH1();
      HAL_ADC_Start(&hadc1);

      //Polling for ADC reading
      HAL_ADC_PollForConversion(&hadc1, 1000);
      vol_ADC = HAL_ADC_GetValue(&hadc1);
      HAL_ADC_Stop(&hadc1);

      //Convert ADC readings to voltage
      vol = 3*(((float)vol_ADC/4095)*3.3)+0.5;
      break;
    case BATT1_CELL2:
      ADC_Select_CH2();
      HAL_ADC_Start(&hadc1);

      //Polling for ADC reading
      HAL_ADC_PollForConversion(&hadc1, 1000);
      vol_ADC = HAL_ADC_GetValue(&hadc1);
      HAL_ADC_Stop(&hadc1);

      //Convert ADC readings to voltage
      vol = 3*(((float)vol_ADC/4095)*3.3);
      break;
    case BATT2_CELL1:
      ADC_Select_CH3();
      HAL_ADC_Start(&hadc1);

      //Polling for ADC reading
      HAL_ADC_PollForConversion(&hadc1, 1000);
      vol_ADC = HAL_ADC_GetValue(&hadc1);
      HAL_ADC_Stop(&hadc1);

      //Convert ADC readings to voltage
      vol = 3*(((float)vol_ADC/4095)*3.3)+0.5;
      break;
    case BATT2_CELL2:
      ADC_Select_CH4();
      HAL_ADC_Start(&hadc1);

      //Polling for ADC reading
      HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
      vol_ADC = HAL_ADC_GetValue(&hadc1);
      HAL_ADC_Stop(&hadc1);

      //Convert ADC readings to voltage
      vol = 3*(((float)vol_ADC/4095)*3.3);
      break;
  }

  return vol;
}
```

Code for the function to read cell voltages (µVision® IDE, 2021)



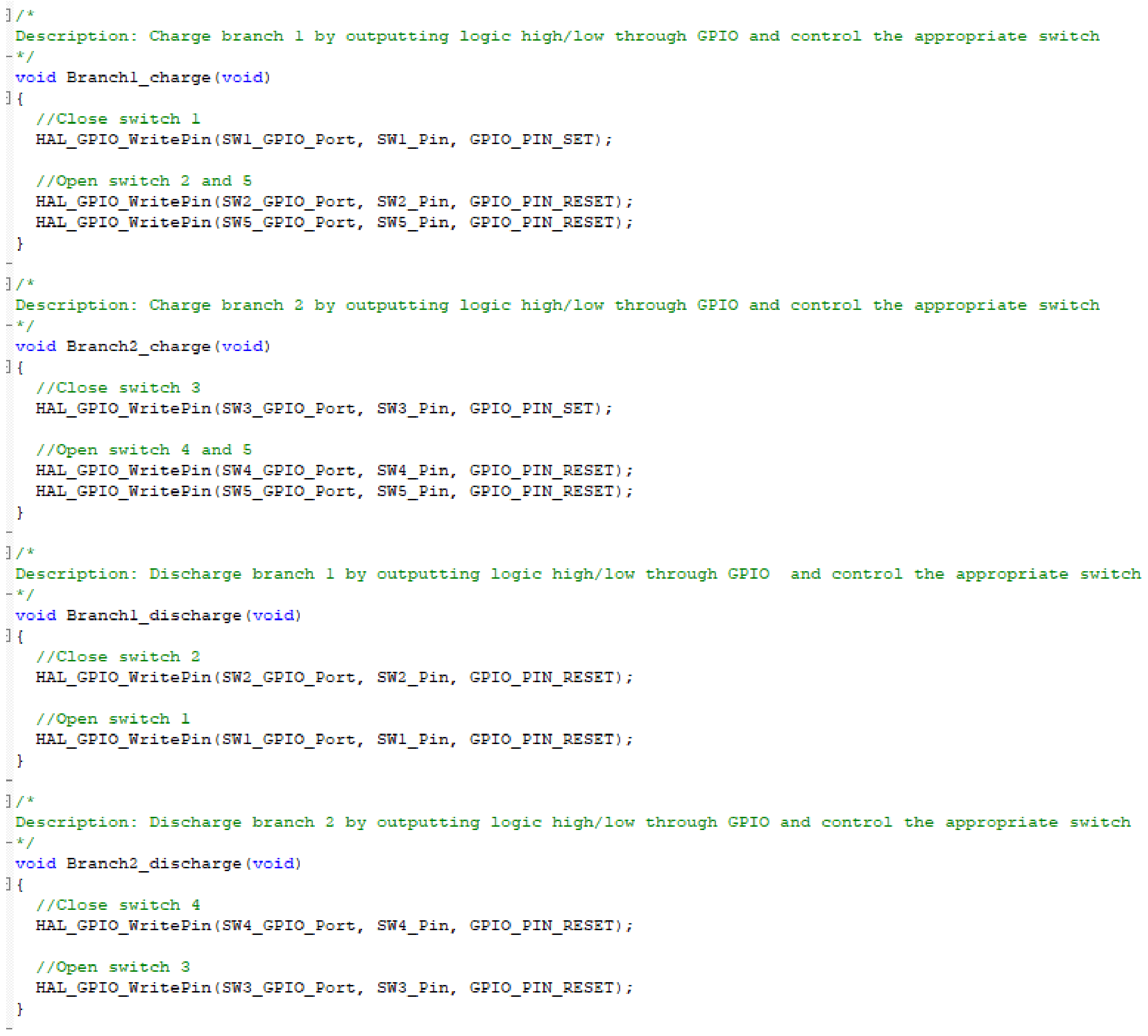
```c
float ReadBranch_V(uint8_t branch)
{
  float vol = 0;
  uint16_t vol_ADC = 0;

  switch (branch)
  {
    case BATT_BRANCH1:
      ADC_Select_CH5();
      HAL_ADC_Start(&hadc1);

      //Polling for ADC reading
      HAL_ADC_PollForConversion(&hadc1, 1000);
      vol_ADC = HAL_ADC_GetValue(&hadc1);
      HAL_ADC_Stop(&hadc1);

      //Convert ADC readings to voltage
      vol = 3*(((float)vol_ADC/4095)*3.3);
      break;
    case BATT_BRANCH2:
      ADC_Select_CH6();
      HAL_ADC_Start(&hadc1);

      //Polling for ADC reading
      HAL_ADC_PollForConversion(&hadc1, 1000);
      vol_ADC = HAL_ADC_GetValue(&hadc1);
      HAL_ADC_Stop(&hadc1);

      //Convert ADC readings to voltage
      vol = 3*(((float)vol_ADC/4095)*3.3);
      break;
  }

  return vol;
}
```

Code for the function to read the branch voltage (µVision® IDE, 2021)



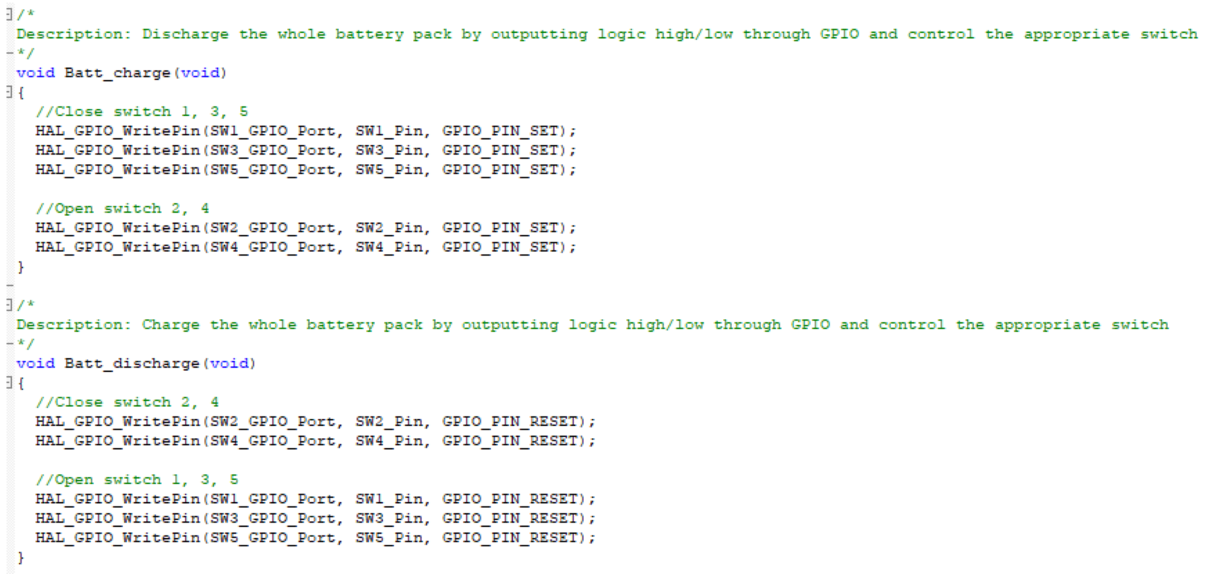
```c
float ReadBranch_I(uint8_t branch)
{
  float curr;
  float vol = 0;
  uint16_t ADC_val = 0;

  switch(branch)
  {
    case BATT_BRANCH1:
      ADC_Select_CH7();
      HAL_ADC_Start(&hadc1);

      //Polling for ADC reading
      HAL_ADC_PollForConversion(&hadc1, 1000);
      ADC_val = HAL_ADC_GetValue(&hadc1);
      HAL_ADC_Stop(&hadc1);

      //Convert ADC reading to voltage
      vol = ((float)ADC_val/4095)*3.3;

      //Convert to current
      curr = (vol-1.65)/0.264;
      break;
    case BATT_BRANCH2:
      ADC_Select_CH8();
      HAL_ADC_Start(&hadc1);

      //Polling for ADC reading
      HAL_ADC_PollForConversion(&hadc1, 1000);
      ADC_val = HAL_ADC_GetValue(&hadc1);
      HAL_ADC_Stop(&hadc1);

      //Convert ADC reading to voltage
      vol = ((float)ADC_val/4095)*3.3;

      //Convert to current
      curr = (vol-1.65)/0.264;
      break;
  }

  return curr;
}
```

Code for the function to read the branch current (µVision® IDE, 2021)



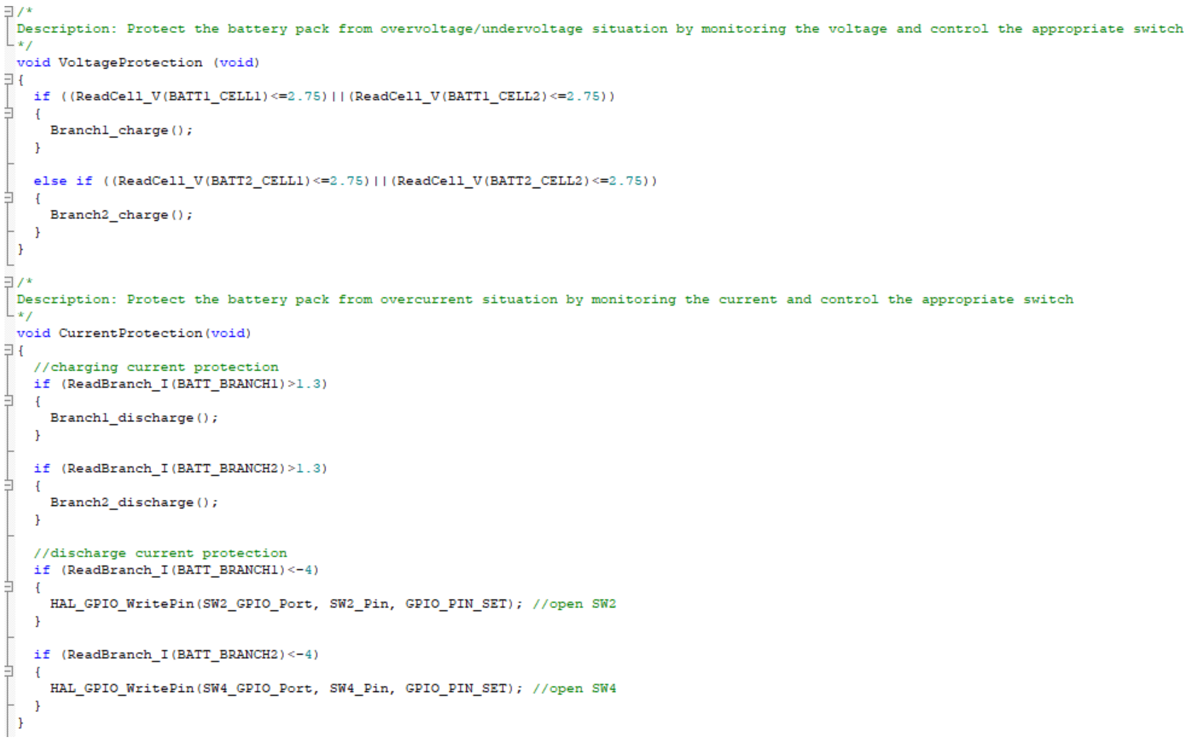
```
/*
Description: Charge branch 1 by outputting logic high/low through GPIO and control the appropriate switch
*/
void Branch1_charge(void)
{
  //Close switch 1
  HAL_GPIO_WritePin(SW1_GPIO_Port, SW1_Pin, GPIO_PIN_SET);

  //Open switch 2 and 5
  HAL_GPIO_WritePin(SW2_GPIO_Port, SW2_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(SW5_GPIO_Port, SW5_Pin, GPIO_PIN_RESET);
}

/*
Description: Charge branch 2 by outputting logic high/low through GPIO and control the appropriate switch
*/
void Branch2_charge(void)
{
  //Close switch 3
  HAL_GPIO_WritePin(SW3_GPIO_Port, SW3_Pin, GPIO_PIN_SET);

  //Open switch 4 and 5
  HAL_GPIO_WritePin(SW4_GPIO_Port, SW4_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(SW5_GPIO_Port, SW5_Pin, GPIO_PIN_RESET);
}

/*
Description: Discharge branch 1 by outputting logic high/low through GPIO  and control the appropriate switch
*/
void Branch1_discharge(void)
{
  //Close switch 2
  HAL_GPIO_WritePin(SW2_GPIO_Port, SW2_Pin, GPIO_PIN_RESET);

  //Open switch 1
  HAL_GPIO_WritePin(SW1_GPIO_Port, SW1_Pin, GPIO_PIN_RESET);
}

/*
Description: Discharge branch 2 by outputting logic high/low through GPIO and control the appropriate switch
*/
void Branch2_discharge(void)
{
  //Close switch 4
  HAL_GPIO_WritePin(SW4_GPIO_Port, SW4_Pin, GPIO_PIN_RESET);

  //Open switch 3
  HAL_GPIO_WritePin(SW3_GPIO_Port, SW3_Pin, GPIO_PIN_RESET);
}
```

Code for the functions to charge and discharge each battery branch (µVision® IDE, 2021)



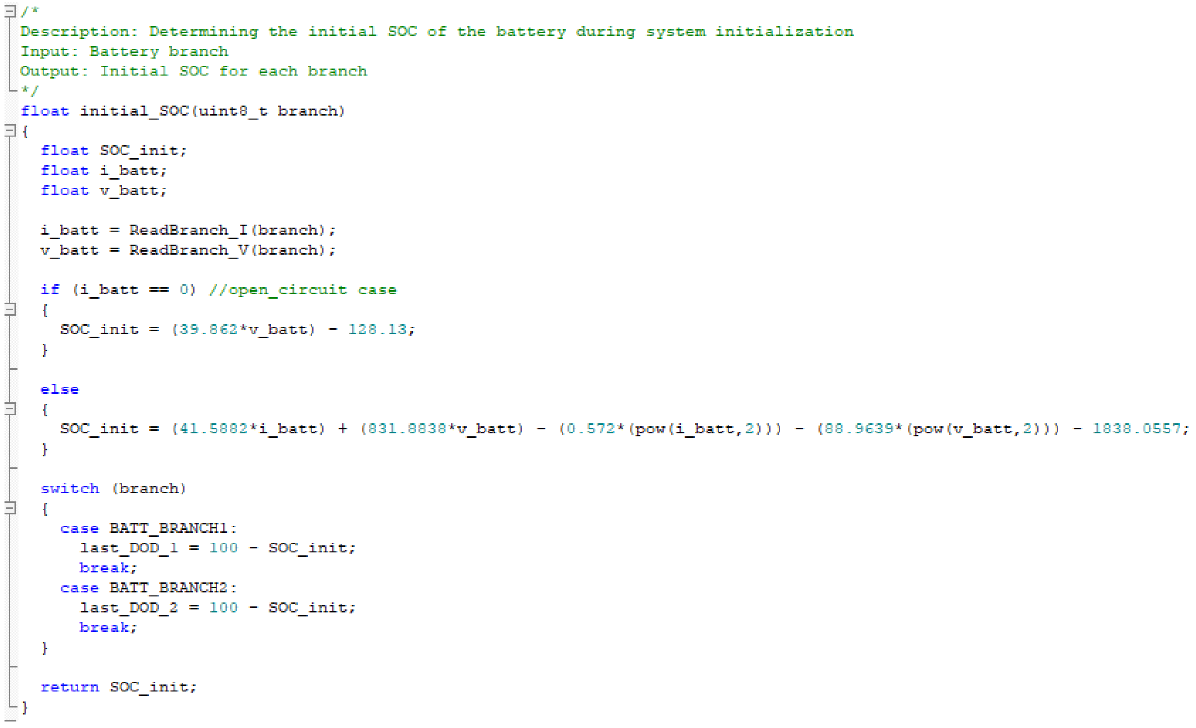
```
/*
Description: Discharge the whole battery pack by outputting logic high/low through GPIO and control the appropriate switch
*/
void Batt_charge(void)
{
  //Close switch 1, 3, 5
  HAL_GPIO_WritePin(SW1_GPIO_Port, SW1_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(SW3_GPIO_Port, SW3_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(SW5_GPIO_Port, SW5_Pin, GPIO_PIN_SET);

  //Open switch 2, 4
  HAL_GPIO_WritePin(SW2_GPIO_Port, SW2_Pin, GPIO_PIN_SET);
  HAL_GPIO_WritePin(SW4_GPIO_Port, SW4_Pin, GPIO_PIN_SET);
}

/*
Description: Charge the whole battery pack by outputting logic high/low through GPIO and control the appropriate switch
*/
void Batt_discharge(void)
{
  //Close switch 2, 4
  HAL_GPIO_WritePin(SW2_GPIO_Port, SW2_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(SW4_GPIO_Port, SW4_Pin, GPIO_PIN_RESET);

  //Open switch 1, 3, 5
  HAL_GPIO_WritePin(SW1_GPIO_Port, SW1_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(SW3_GPIO_Port, SW3_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(SW5_GPIO_Port, SW5_Pin, GPIO_PIN_RESET);
}
```

Code for the function to charge and discharge to whole battery branch (µVision® IDE, 2021)



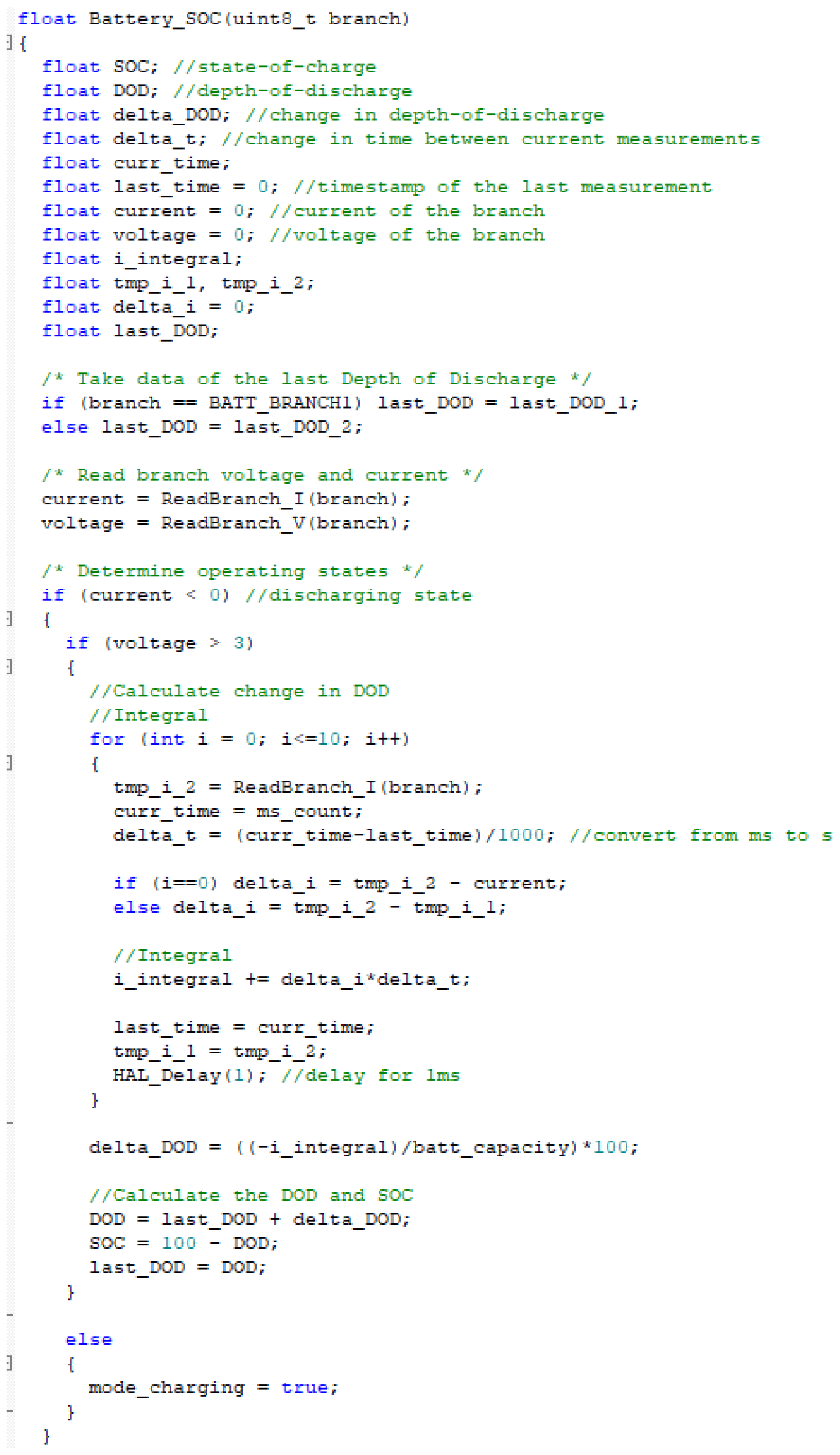
```
/*
Description: Protect the battery pack from overvoltage/undervoltage situation by monitoring the voltage and control the appropriate switch
*/
void VoltageProtection (void)
{
  if ((ReadCell_V(BATT1_CELL1)<=2.75)||(ReadCell_V(BATT1_CELL2)<=2.75))
  {
    Branch1_charge();
  }

  else if ((ReadCell_V(BATT2_CELL1)<=2.75)||(ReadCell_V(BATT2_CELL2)<=2.75))
  {
    Branch2_charge();
  }
}

/*
Description: Protect the battery pack from overcurrent situation by monitoring the current and control the appropriate switch
*/
void CurrentProtection(void)
{
  //charging current protection
  if (ReadBranch_I(BATT_BRANCH1)>1.3)
  {
    Branch1_discharge();
  }

  if (ReadBranch_I(BATT_BRANCH2)>1.3)
  {
    Branch2_discharge();
  }

  //discharge current protection
  if (ReadBranch_I(BATT_BRANCH1)<-4)
  {
    HAL_GPIO_WritePin(SW2_GPIO_Port, SW2_Pin, GPIO_PIN_SET); //open SW2
  }

  if (ReadBranch_I(BATT_BRANCH2)<-4)
  {
    HAL_GPIO_WritePin(SW4_GPIO_Port, SW4_Pin, GPIO_PIN_SET); //open SW4
  }
}
```

Code for the function to perform overvoltage and overcurrent protection of the battery branch (µVision® IDE, 2021)



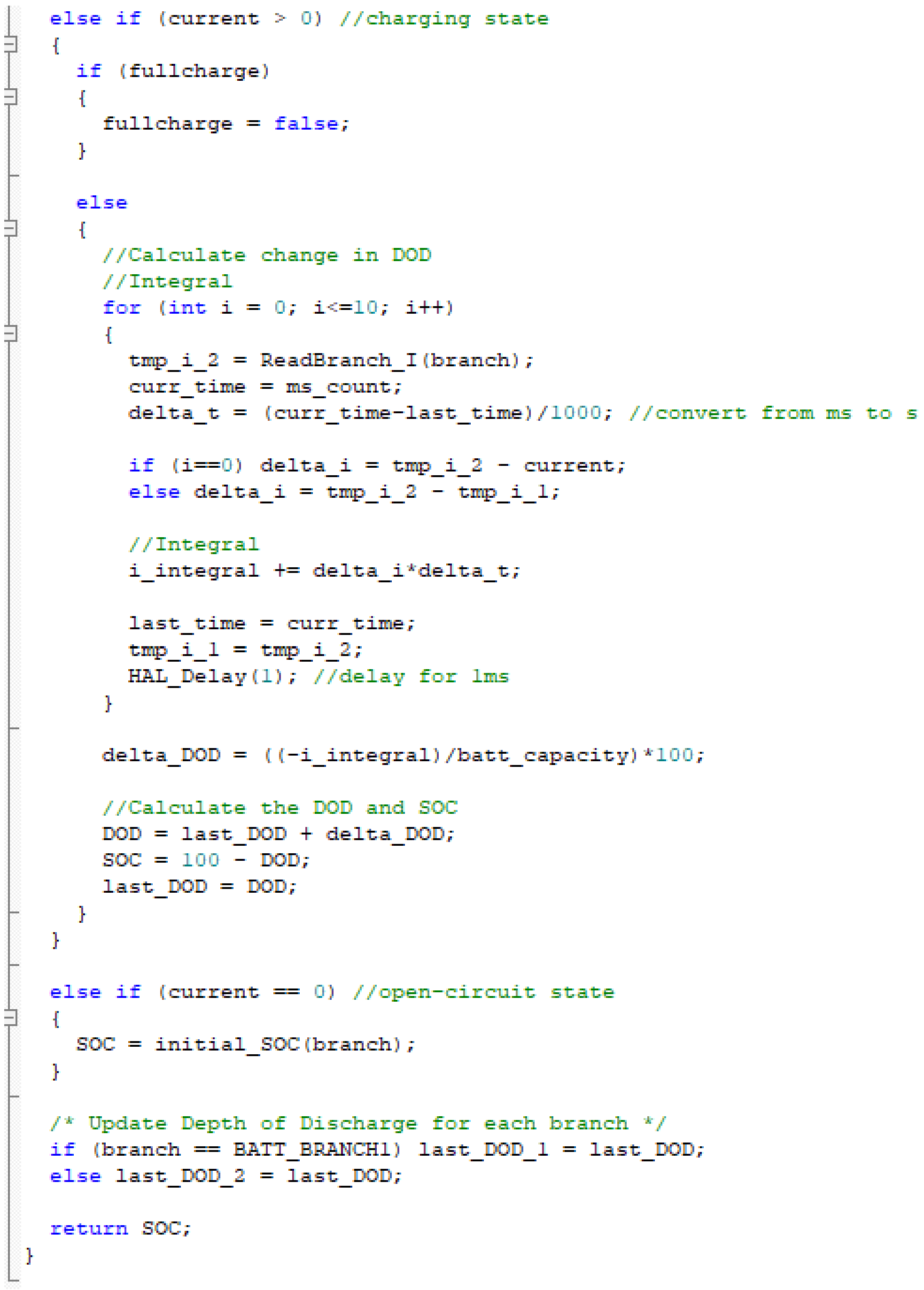
```
/*
Description: Determining the initial SOC of the battery during system initialization
Input: Battery branch
Output: Initial SOC for each branch
*/
float initial_SOC(uint8_t branch)
{
  float SOC_init;
  float i_batt;
  float v_batt;

  i_batt = ReadBranch_I(branch);
  v_batt = ReadBranch_V(branch);

  if (i_batt == 0) //open_circuit case
  {
    SOC_init = (39.862*v_batt) - 128.13;
  }

  else
  {
    SOC_init = (41.5882*i_batt) + (831.8838*v_batt) - (0.572*(pow(i_batt,2))) - (88.9639*(pow(v_batt,2))) - 1838.0557;
  }

  switch (branch)
  {
    case BATT_BRANCH1:
      last_DOD_1 = 100 - SOC_init;
      break;
    case BATT_BRANCH2:
      last_DOD_2 = 100 - SOC_init;
      break;
  }

  return SOC_init;
}
```

Code for the function to read the initial battery State of Charge (µVision® IDE, 2021)



```c
float Battery_SOC(uint8_t branch)
{
  float SOC; //state-of-charge
  float DOD; //depth-of-discharge
  float delta_DOD; //change in depth-of-discharge
  float delta_t; //change in time between current measurements
  float curr_time;
  float last_time = 0; //timestamp of the last measurement
  float current = 0; //current of the branch
  float voltage = 0; //voltage of the branch
  float i_integral;
  float tmp_i_1, tmp_i_2;
  float delta_i = 0;
  float last_DOD;

  /* Take data of the last Depth of Discharge */
  if (branch == BATT_BRANCH1) last_DOD = last_DOD_1;
  else last_DOD = last_DOD_2;

  /* Read branch voltage and current */
  current = ReadBranch_I(branch);
  voltage = ReadBranch_V(branch);

  /* Determine operating states */
  if (current < 0) //discharging state
  {
    if (voltage > 3)
    {
      //Calculate change in DOD
      //Integral
      for (int i = 0; i<=10; i++)
      {
        tmp_i_2 = ReadBranch_I(branch);
        curr_time = ms_count;
        delta_t = (curr_time-last_time)/1000; //convert from ms to s

        if (i==0) delta_i = tmp_i_2 - current;
        else delta_i = tmp_i_2 - tmp_i_1;

        //Integral
        i_integral += delta_i*delta_t;

        last_time = curr_time;
        tmp_i_1 = tmp_i_2;
        HAL_Delay(1); //delay for 1ms
      }

      delta_DOD = ((-i_integral)/batt_capacity)*100;

      //Calculate the DOD and SOC
      DOD = last_DOD + delta_DOD;
      SOC = 100 - DOD;
      last_DOD = DOD;
    }

    else
    {
      mode_charging = true;
    }
  }
```

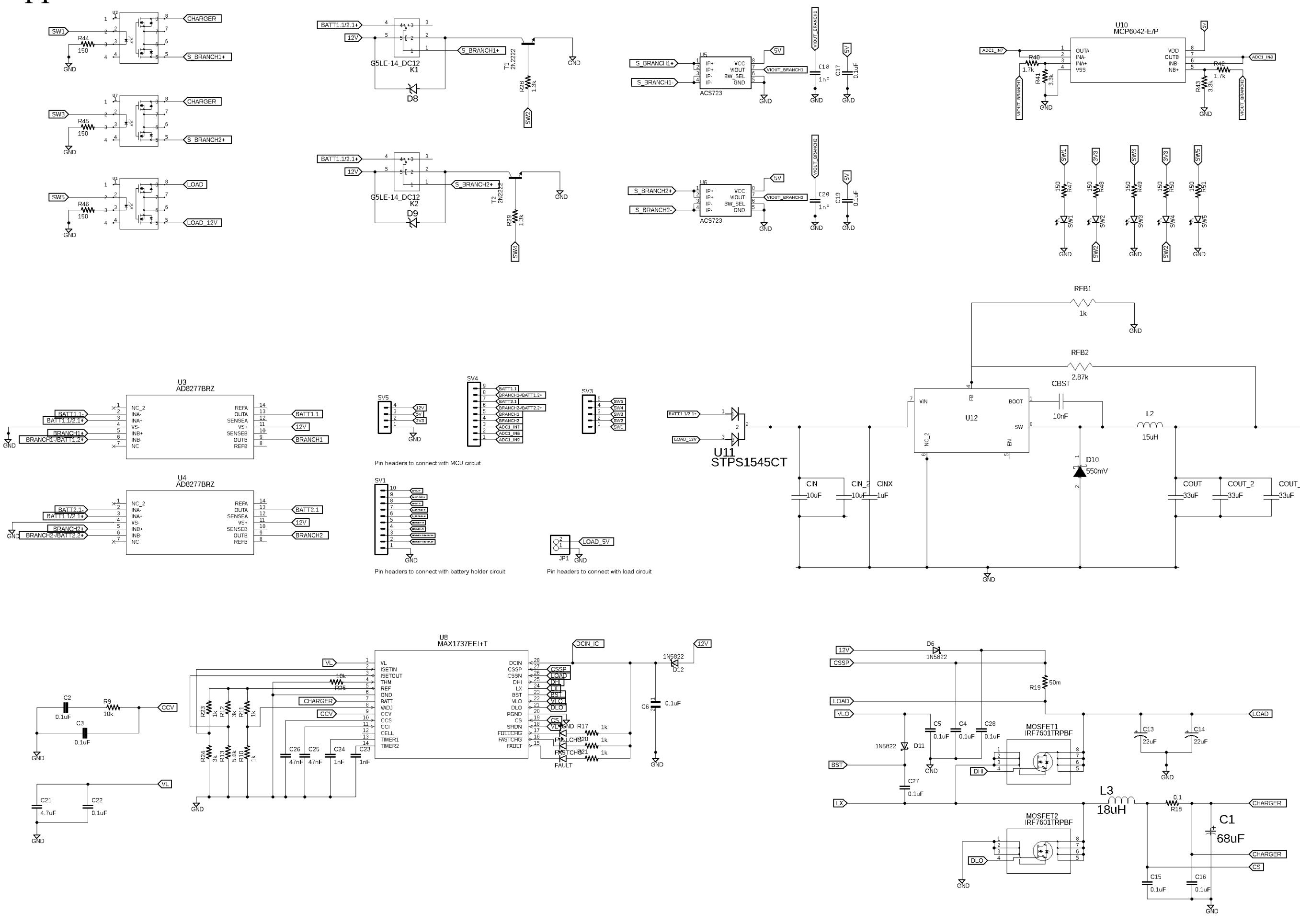
Part 1 of the code for the function to calculate the Battery State of Charge during operation (µVision® IDE, 2021)



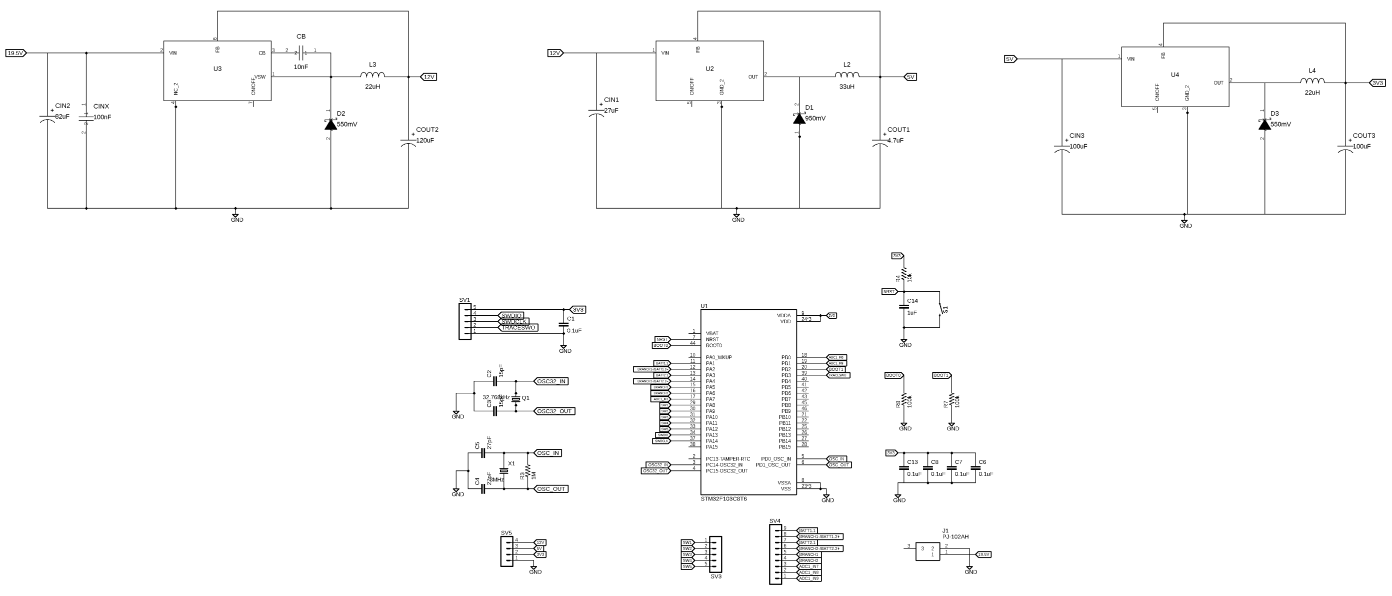
```
  else if (current > 0) //charging state
  {
    if (fullcharge)
    {
      fullcharge = false;
    }

    else
    {
      //Calculate change in DOD
      //Integral
      for (int i = 0; i<=10; i++)
      {
        tmp_i_2 = ReadBranch_I(branch);
        curr_time = ms_count;
        delta_t = (curr_time-last_time)/1000; //convert from ms to s

        if (i==0) delta_i = tmp_i_2 - current;
        else delta_i = tmp_i_2 - tmp_i_1;

        //Integral
        i_integral += delta_i*delta_t;

        last_time = curr_time;
        tmp_i_1 = tmp_i_2;
        HAL_Delay(1); //delay for 1ms
      }

      delta_DOD = ((-i_integral)/batt_capacity)*100;

      //Calculate the DOD and SOC
      DOD = last_DOD + delta_DOD;
      SOC = 100 - DOD;
      last_DOD = DOD;
    }
  }

  else if (current == 0) //open-circuit state
  {
    SOC = initial_SOC(branch);
  }

  /* Update Depth of Discharge for each branch */
  if (branch == BATT_BRANCH1) last_DOD_1 = last_DOD;
  else last_DOD_2 = last_DOD;

  return SOC;
}
```

Part 2 of the code for the function to calculate the Battery State of Charge during operation (μVision® IDE, 2021)

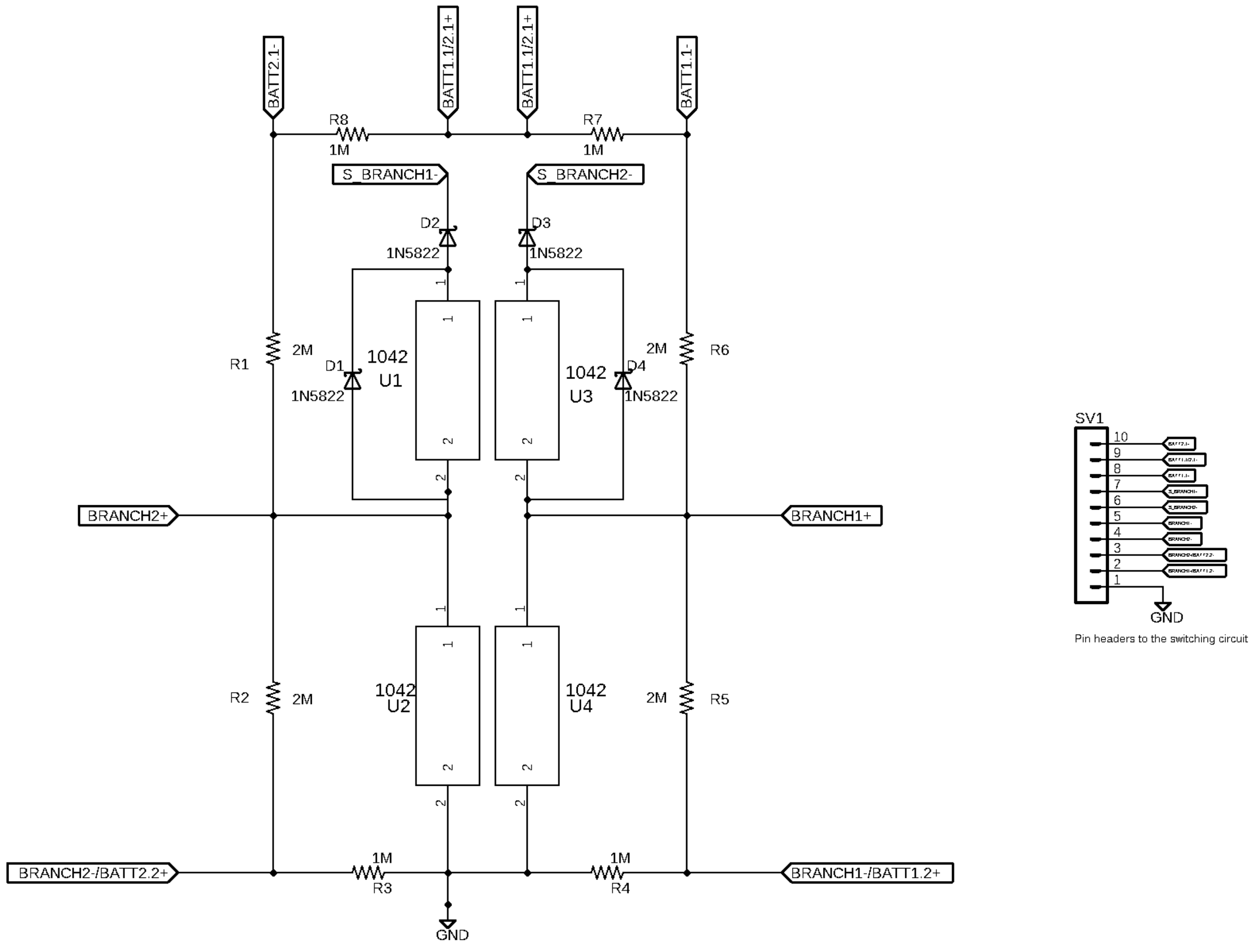
Appendix B: Detailed Electrical Schematic



Complete Schematic of the Battery Switching Circuit (EAGLE, 2021)



Complete Schematic for the DC-DC converters and Microcontroller circuit (EAGLE, 2021)



Complete Schematic for the Battery Bank (EAGLE, 2021)