Modular Battery Management System

A Technical Report submitted to the Department of Electrical and Computer Engineering

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Nripesh Manandhar
Spring, 2021

Technical Project Team Members
Dipesh Manandhar
Phillip Phan
Nikilesh Subramaniam
William Zhang

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature  Date 11/1/2020
Nripesh Manandhar

Approved  Date 11/23/2020
Harry C. Powell, Department of Electrical and Computer Engineering

Modular Battery Management System (BMS)

Dipesh Manandhar, Nripesh Manandhar, Phillip Phan, Nikilesh Subramaniam and William Zhang

12/10/20

Capstone Design ECE 4440 / ECE4991

Signatures

Dipesh Manandhar

Nripesh Manandhar

Phillip Phan

Nikilesh Subramaniam

William Zhang

Statement of work:

Dipesh Manandhar

I primarily worked on the system design for both the hardware and software, as well as laying out the Cell Node board and setting up the software development environment. I designed the initial drafts of the system diagrams for both Main Node and Cell Node hardware, as well as software flow charts for both. I helped Nripesh set up some of the hierarchical sub-schematics, including the CAN transceivers and the MCUs. Major design changes during the design phase, prior to any PCB fabrication, did result in changes to the hardware system diagrams, and I was responsible for keeping track of all such changes and updating the system diagrams appropriately.

I also set up the software development environment for both Main node and Cell Node, as I had the most experience working with the tools we were using. This included setting up the PlatformIO toolchain and project files, Mbed OS custom target files, and debugging and testing environments to use for both our custom PCBs and the NUCLEO development boards. I also contributed to integrating the code Phillip and Niki wrote for the SOC and SOH calculations, fan logic, and balancing logic into the software for the Main Node, as well as writing the functions for reading the pack voltage and current. After we realized the issue with the pack voltage and current sensor accuracies, I handled the hardware and software changes necessary to reduce the range on these sensors.

Nripesh Manandhar

I primarily worked on designing the hardware schematics in KiCad for both the Cell Node and Main Node, as well as laying out the Main Node board. I set up all the KiCad component libraries for schematic symbols and footprints of real components not included in KiCad. I also designed the hierarchical structure of the schematics. I designed most of the sub-schematics for the Main Node and Cell Node, with help from Niki and Dipesh on some of the sub-schematics, such as the temperature sensor on the Cell Node and setting up the MCUs. Most major design decisions were made as a team, such as changing the power supply schematics to always use the external supplies rather than the batteries themselves. However, I made a lot of the decisions on specific components to use, such as the specific regulators in the power supply schematics.

After the first iteration of the hardware was tested, I made most of the modifications to the Cell Node's schematics and layout. I also added LEDs to the Cell Node for ease of debugging. After the hardware was finalized, I also helped write some of the software, especially for the Cell Node. I wrote some of the shared code between Cell Node and Main Node such as the CAN structs used for communication between the Cell Nodes and the Main Node, but I mostly wrote code for the Cell Node specifically. This included setting up the main loop to read

voltage and temperature inputs and setting up the background threads for CAN Rx and Tx. I also performed the final demonstration of the project, with the help of Dipesh who recorded the video, showing all the various functionality of the modular BMS.

Phillip Phan

For the modular BMS, I primarily worked on testing the boards and on designing the software for SOC and SOH estimation. I designed the testing methodology for the modular BMS so that the hardware components for the BMS system were tested first individually before doing integration testing with the combined hardware components. The software components in the modular BMS were tested afterwards. I also created diagrams for the hardware and software test plans, which present the test plan in a form of a decision tree to enable the person testing the modular BMS to know what steps to take.

In testing the main and cell boards, I first verified that the hardware components of the modular BMS functioned correctly through voltage and resistance measurements with Niki. Afterwards, Niki and I wrote test programs to verify that the readings from sensors were correct, and verify that components like the relays were controllable through software. When I was designing the software for the modular BMS, I had to conduct extensive research to weigh the pros and cons of various SOC estimation methods before deciding on our chosen SOC estimation method. I also researched various methods to calculate the error from using our chosen SOC estimation method. Additionally, I helped assemble and solder the through hole components onto the main and cell PCBs with William.

Nikilesh Subramaniam

For this project, I worked on researching cell balancing methods and pack health algorithms. Initially, I looked for active cell balancing schematics that could transfer charge between battery cells to equalize their charge. After some more research and simulating an active cell balancing, I opted for the simpler passive cell balancing circuit because it was simpler. I also researched algorithms for pack health estimation. Initially, I researched a method to use the Kalman filter to estimate state of charge (SOC) and state of health (SOH), but we ended up going with the Coulomb counting method that Phillip researched. I implemented our chosen estimation algorithm in software.

Additionally, I wrote some software for testing and logic. For testing, I wrote small test programs to instantiate GPIO pins and read or write from then. I worked with Phillip to test these GPIO pins such as the analog input to read pack voltage and pack current and the digital output to control cell balancing. Additionally, along with SOC estimation, I helped with main board logic such as fan control and cell balancing.

William Zhang

I worked on multiple different aspects of the project, helping out with different areas when needed. In the initial stages of the modular BMS project, I worked on research for the component selection, such as the microcontroller. Initially, I did extensive research on the available hardware options for the microcontrollers for both the main node and cell node, based on the requirements of each. I evaluated the pricing, package, onboard peripherals, power consumption, and software stack for each of the options and helped the team come to a decision on final component selection.

For each round of PCB send outs, once the team received the manufactured PCBs, I organized, prepared, and took the components to be populated by WWW Electronics. Once WWW finished populating the SMD components and I picked up the boards, I worked with Phillip to solder the remaining components and assemble them.

I additionally worked with Phillip and Niki on debugging the boards. With the main node board, I worked with Phillip and Niki to debug the issues we were having with the microcontroller being constantly held in reset, preventing code from being uploaded to the board. I researched different hardware and software solutions, which we tried one by one. When running the software tests on the cell node, I worked with Phillip and Niki to resolve issues with the code failing to print to the serial console properly and the ADC failing to initialize properly. I researched different Mbed target configuration options and other toolchains and tested them to try to resolve the issue.

In addition, I worked with Dipesh on setting up, testing, and debugging the embedded software toolchain and build process, particularly for the main node. I generated and collated the necessary target header and source files from the HAL libraries provided by ST Microelectronics for the toolchain to build for the STM32G473 microcontroller on the main node.

Finally, I also worked on the PC program as the external interface for the modular BMS. I wrote the CAN bridge program in Rust to allow CAN bus frames received on a CAN interface to be relayed over a WebSocket, to a GUI program, which I partially finished.

Table of Contents

Table of Figures

(This should list the page of each figure used in your document, including the full caption.) Word has tools to help you do this very easily)

Abstract

This project is a battery management system (BMS) designed to monitor, protect, and efficiently use battery packs for electric vehicles. This modular system is a star network of BMS boards that is usable for battery packs of many sizes. The star network consists of a cell board for each battery cell and a main board that interacts with the cell board. The BMS has charge and discharge protection, estimates the state of charge of the battery pack, uses passive cell balancing, and has a user interface to view battery pack data. Each BMS module is small enough to clip onto 18650 cells, rechargeable lithium-ion cells that are commonly used in electric vehicles [1]. This battery management system reduces the battery space needed in electric vehicles and is easily able to adapt to different pack sizes.

Background

Electric vehicles (EVs) have recently gained popularity as an environmentally friendly alternative to vehicles that need gas [2]. The battery is an essential component of the electric vehicle and its performance determines the driving range of EVs. A battery management system (BMS) is needed in order to prevent the battery from overcharging or supplying high currents which can deteriorate the lifetime of the battery. In addition, a BMS can report important battery information such as state of charge and extend the battery lifetime via cell balancing [3]. Many BMSs can only be used for a certain battery pack with a maximum number of cells [4]. This project proposes a modular BMS design that can handle many different pack types and sizes while still providing essential BMS services.

There have been BMS research in both academia and industry. One paper from a joint European effort detailed different battery modeling methods and cell balancing methods such as passive heat dissipation and active distributed balancing. The paper also sets standards for battery management such as system inputs, responsibilities, and possible sources of error [3]. Shandong University designed and tested a Li-ion BMS using a microcontroller that had charge and discharge protection, single cell voltage and temperature monitoring, and cell balancing. This BMS monitored 16 cells, with each group of 8 cells being monitored by a chip. This chip communicated with the microcontroller via an I^2C bus [5].

A popular BMS from industry is the Orion BMS [6]. Along with BMSs, Orion offers a user interface that lets users tweak all the parameters of the BMS. Users can monitor temperature, set current limits and device parameters, see live data being gathered, and configure CAN communications. One downfall of the Orion BMS is that it is not modular. Orion offers different BMS sizes of up to 168 cells. But if a user wants to resize their battery pack, they have to buy a new BMS instead of buying an addon module to their existing BMS.

A novel aspect of our project is the module size. The BMS cell modules are small enough to clip on to the side of an 18650 cell. Smaller modules will lead to less space needed for battery storage in EVs.

To complete this project, we have drawn from our engineering curriculum. This project incorporates embedded systems, hardware design, user interface design, and communication. The PCB design and work with power supplies uses information learned in ECE 2630, ECE 2660, and ECE 3750. ECE 3430 provided techniques to write the software of the BMS boards. The user interface that interfaces with the BMS draws from CS 3240. Finally, some of our team members have experience working on the Solar Car team, which gives them knowledge about batteries and embedded systems for electric vehicles.

Constraints

Design Constraints

This project was a Capstone project for Computer Engineering, so there were some design constraints placed by the course. First, the project had to incorporate an embedded CPU capable of real-time response. Second, the project had to interact with a device that has not been seen in earlier classwork. Finally, the project had to have professional quality mounting, a PCB.

CPU Limitations

The STM32G473CET6 microcontroller [7] was chosen for the BMS main board and the STM32F042F6P6 microcontroller [8] was chosen for the BMS cell boards. These microcontrollers were chosen because of their CAN capabilities, low power consumption, and price. The cell board microcontroller has 32 Kbytes of Flash memory, which severely limited the amount of software that was on the device.

Manufacturing Limitations

When designing the PCB layout, there were constraints placed by the PCB manufacturer, Advanced Circuits [9], and the company who soldered components onto the board, WWW Electronics Inc [10]. The PCB limitations included a maximum size of 30 square inches, minimum 5 mil line/space and minimum 10 mil hole size. WWW Electronics required legible silkscreen with a minimum height of 1.5mm and oriented left-to-right or bottom-to-top. Additionally, components such as polarized capacitors and ICs needed a marking to indicate its orientation.

Economic and Cost Constraints

This project was given a budget of $500. Since this project has many components (1 main board, at least 2 cell boards, lithium-ion batteries), few backup boards were purchased. Only 1 main board and 3 cell boards were manufactured. To save additional costs, the main board was designed with multiple through hole components which can be soldered by a team member instead of paying WWW Electronics to solder them. Further cost savings were achieved by having the UVA Solar Car Team purchasing the 18650 batteries and the Vruzend Battery Kits.

External Standards

CAN (Controller Area Network) 2.0B

In order to provide a standard interface for use in a vehicle system with potentially multiple control units, the CAN (Controller Area Network) protocol was used. In particular, CAN 2.0B was used with 11-bit message IDs. This protocol has a message payload of 8 bytes and a data rate of 1 Mb/s [11].

IPC 2221

The minimum electrical spacing for all traces on the PCBs were determined by the IPC 2221 standard [12]. This is especially important for traces that were isolated from each other, as the spacing was the biggest factor in determining the effective isolation range. This is also very important for high-voltage traces, namely the traces stepping down the pack voltage. If these minimum spacings are not followed, there may be risk of arcing caused by high voltage differences between traces.

SMD Component Packages

Surface Mount Device (SMD) packages have standardized sizes, allowing for ease of component manufacture and replacement. JEDEC is a leading standardization body [13]. The PCBs use many sizes of SMD components, including 0603, 0805, and 1210 passive components, Size A (EIA 3216-18) Tantalum capacitors, SOD-323 diodes, 8-TSSOP op-amps, SOT-23 regulators, 16-SOIC CAN transceivers, a 20-TSSOP microcontroller and a 48-LQFP microcontroller. The Main Node also uses many through-hole packages for most of its passive components, utilizing both radial and axial components.

USB (Universal Serial Bus)

USB standards define how a computer can communicate with peripheral devices. The standards are developed and maintained by the USB Implementers Forum (USB-IF) [14]. A USB connection is used by the STLINK-MINI to allow for USB debugging and to allow the microcontroller to print to a serial monitor.

SWD (Serial Wire Debugging)

The SWD specification, an ARM modification to the JTAG specification, is used for flashing and debugging embedded code [15].

UART (Universal Asynchronous Receiver-Transmitter)

A UART, or a block of circuitry used for asynchronous communication, was used for communication from the Main Node to a computer for printing pack and cell data. The standard baud rate of 9600 bits/second was used [16].

RoHS (Restriction of Hazardous Substances) Compliance

RoHS components do not use specific hazardous substances, as specified in Directive 2002/95/EC which originated from the European Union [17]. Most components used were RoHS compliant, minimizing the environmental impact of our design.

WebSocket

In order to design a BMS configuration/data visualization program (PC Program) with the ability for remote connection, the WebSocket protocol was used to communicate between the two components. The WebSocket Protocol enables two-way communication between a client running untrusted code (in our case, the web interface) in a controlled environment to a remote host that has opted-in to communications from that code, such as our bridge program relaying messages between the CAN bus and the web interface [52].

Tools Employed

KiCad [18]:

KiCad is a software suite for electronic design automation. KiCad was used to help us create the circuit schematics and design the layouts for the PCBs for the Modular BMS. KiCad was also used to generate the Gerber files that were used to eventually print out the PCBs. William, Nripesh, and Dipesh had prior experience with using KiCad as a part of the Solar Car Team. Nikilesh and Phillip did not have prior experience with KiCad, and had to learn how to use KiCad while working on the Modular BMS.

GitHub [19]:

GitHub is a code hosting platform for git version control. GitHub enabled our team to share software code for the Modular BMS across our team members. Every team member was already proficient with using GitHub.

CADLAB [20]:

CADLAB is a visual version control platform for PCB design projects. Our team used CADLAB to collaborate and share PCB designs and circuit schematics for the Modular BMS. William, Nripesh, and Dipesh had prior experience with using CADLAB as a part of the Solar Car Team. Nikilesh and Phillip did not have prior experience with CADLAB, and had to learn how to use CADLAB while working on the Modular BMS.

Visual Studio Code Integrated Development Environment (VS Code IDE) [21]:

The VS Code IDE is a source code editor. Our team used the VS Code IDE to help write the code for our project. The VS Code IDE was also used as a wrapper to use PlatformIO features. Every team member was already proficient with using the VS Code IDE.

PlatformIO [22]:

PlatformIO is an open source code editor that is an extension of VS Code. PlatformIO was used as the IDE to help develop software for the Modular BMS. PlatformIO was also used to help compile our software and the PlatformIO debugger was also used to debug problems with our software and with our PCBs. Our team did not have prior experience with using PlatformIO and had to familiarize ourselves with PlatformIO while completing the Modular BMS.

Mbed OS [23]:

Mbed OS is an open-source embedded operating system. Mbed OS was used as the software framework to write code for the Modular BMS. Our team did not have prior experience with using Mbed OS and had to familiarize ourselves with Mbed OS while completing the Modular BMS.

Ethical, Social, and Economic Concerns

Environmental Impact

The environmental impact of the project will mainly concern the 18650 batteries used for testing the project. The lithium-ion batteries have a limited lifespan and consist of materials that require a large environmental impact to extract. Lithium itself has a high environmental cost, since the mining of lithium uses huge amounts of water and toxic chemicals to extract the material [24]. At the end of the battery lifespan, the batteries will be sent to a recycling facility to reduce the environmental impact [25].

Sustainability

The project aims to improve the wider adoption of electric vehicles as a mode of transportation by creating a battery system that is easier to work with and develop for. This is enabled through modular system design, allowing for different battery pack configurations, as well as wider reusability for different electric vehicle applications. As a result, this will help reduce the amount of waste generated from the design of multiple unique battery systems for different applications, as the parts of the system can be reused. In addition, helping to drive adoption of electric vehicles will help reduce the overall carbon footprint from fossil fuels used in transportation.

Health and Safety

The voltage produced by each individual 18650 cell will be approximately 0 - 5V. When put in series, the total voltage can be over 100V. As such, when working with high voltages, caution must be taken to avoid injury. Since the design is modular, the majority of testing will be

able to be performed with relatively low voltages in small configurations. However, when testing with larger configurations, precautions such as rubber gloves and other protective equipment can be used [26].

As the Modular BMS will be used to reduce costs and popularize electric vehicles, there are additional concerns that one must be aware of. After a car accident with an EV, for example, the lithium-ion battery that powers the EV is prone to exploding or catching on fire. The same safety precautions that are used in the lab such as insulated tools, are needed for first responders to handle EV accidents and prevent exposure to high voltages or currents. EV manufacturers need to design safety mechanisms such as additional shielding of the lithium-ion battery through an external aluminum plate and firewalls between battery modules [27]. These changes are needed to minimize the risks from using li-ion batteries and ensure the safety of their passengers when using the Modular BMS in an EV.

Manufacturability

The system will be straightforward to manufacture, especially as pre-made weld-free battery holders will be used. The primary components to be manufactured are the PCBs and the housing for the main node. The housings can be 3D-printed, while the batteries to be purchased are standard 18650 cells and are easily purchasable.

Ethical Issues

An ethical risk with the Modular BMS is that the rare earth metals in the electrical components that are used in assembling the Modular BMS such as the 18650 batteries are sourced from countries that lack strong labor laws and use child labor [28]. When sourcing the components for the Modular BMS, our team should aim to try to source parts from countries that uphold worker safety and do not use child labor if possible.

Intellectual Property Issues

Previous BMS designs have been patented before such as a general BMS design that was patented by Ivan Loncarevic [29]. This BMS patent has the following as an independent claim: a central controlling microcontroller that is connected to a plurality of control circuits where each control circuit monitors and controls the charging of each battery cell. This BMS patent also has a dependent claim of having the BMS monitor the battery temperature and reduce the charging current if the temperature is above a specific threshold. In a patent filed by Hak Hon Chau for a fault tolerant modular battery management system, an independent claim for general BMS functionalities and a modular BMS design was filed [30]. In a similar patent filed by Joseph Mario Ambrosio and Konstantinos Sfakianos, an independent claim for a modular BMS design for use in EVs was filed [31]. Based on the claims made by these patents, the modular BMS that our team designed and built is not patentable due to the overlap with existing patents.

Detailed Technical Description of Project

The goal of the project was to create a modular battery management system (BMS) for lithium-ion battery packs for electric vehicles. Our modular BMS would have a cell board for each battery cell that would monitor voltage and temperature. The cell boards would connect to a main BMS board which measures pack voltage and current, estimates pack health metrics and controls a fan to regulate temperature. Additionally, the main board would control relays to prevent the battery pack from overcharging and over discharging and send commands to the cell boards to perform passive cell balancing. Regulating temperature, current through the battery pack, and overall pack health ensures that the battery pack is used with maximum efficiency. The system is broken down into the following sections:

1. Hardware
    a. Main Board
        i. Schematics
        ii. PCB
    b. Cell Board
        i. Schematics
        ii. PCB
    c. Connected System
2. Software
    a. Pack Health Algorithm
    b. CAN specification
    c. PC Program

A block diagram of the system is shown in Figure 1. This shows that each module node (or cell node) is connected to a battery cell and communicates with the main node via a CAN bus. Additionally, the main node is connected to the pack current sensor, charging and discharging contactors, and a fan. All boards are powered using an external power supply.

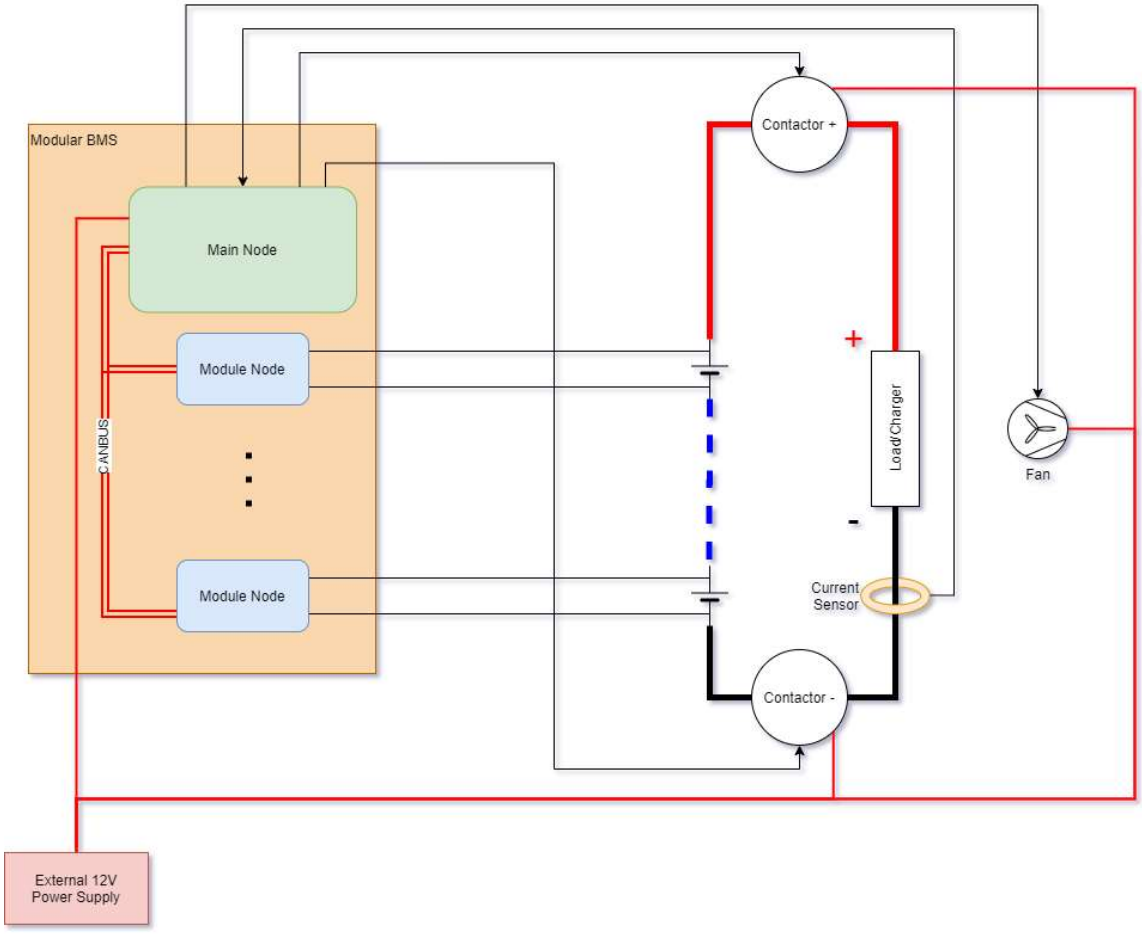


Figure 1. Block Diagram of the Modular BMS

Hardware

The hardware of this project consists of one main board and multiple cell boards. For our minimum viable product, we had two cell boards to show that the main board could control and communicate with multiple cell boards.

Main Node

The main node is the main controlling unit of this system. The main node is in charge of powering the fan and triggering relays along with gathering battery pack data to report to a PC interface. The full scope of the main board is listed in the goals below:

1. Get cell temperatures and voltages from cell boards
2. Measure total pack voltage and pack current
3. Estimate pack health parameters: State of Charge (SOC) and State of Health(SOH)
4. Trigger relays to prevent battery from supplying high currents, overcharging, or over discharging

5. Power fan and set fan speed to regulate temperature
6. Send cell balancing commands to cell boards to equalize cell voltages via CAN
7. Send battery pack data to a PC program via CAN

The pack health parameters that the main board estimates are the State of Charge (SOC) and the State of Health (SOH) of each battery cell. The SOC of a battery cell is the ratio between the current releasable capacity and the rated capacity of the cell. The SOC is the percent charge you see for phone batteries. The SOH of the cell is the ratio between the maximum releasable capacity and the rated capacity of the cell [32].

In order to charge and discharge with maximum efficiency, cell balancing is needed. Cell balancing ensures that all cells are at the same state of charge, which lets all the cells get fully charged and fully discharged. If the cells are imbalanced, one cell will overcharge while another cell is still charging which hurts the battery cells [33].

Schematic

The schematic for the main node was designed in KiCAD shown in Figure 2. The schematic was broken down into many hierarchical blocks for ease of understanding.

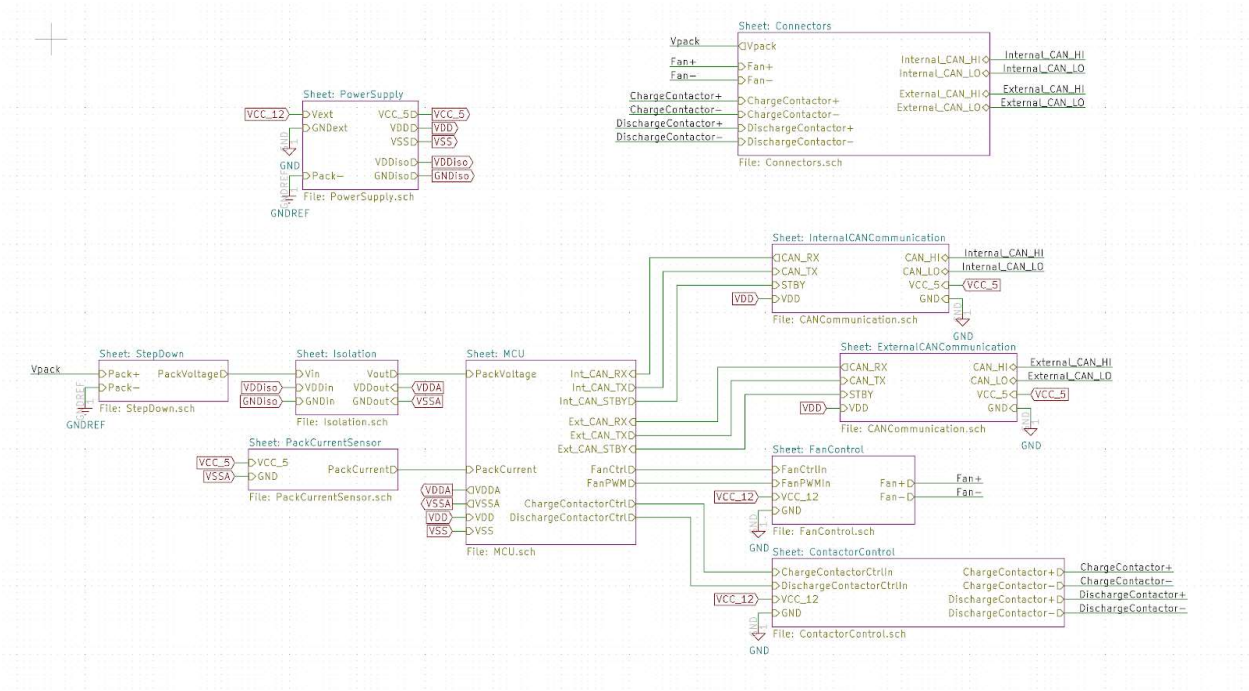


Figure 2. Main Node Schematic

The main board is powered using a 12-volt supply which is then converted to a 5V supply and 3.3 V supply. The 5V and 3.3V supplies are used for the CAN transceiver and microcontroller respectively. The main node microcontroller interfaces with a pack current

sensor, a fan, and two contactors. One contactor is to control the charging of the pack and the other is to control the discharging of the pack. The microcontroller is also connected to two separate CAN buses. The internal CAN bus is for the main board to communicate with the cell boards of the BMS. The external CAN bus is for the main board to report pack data to a PC program. Finally, the microcontroller measures the pack voltage, but is isolated from the pack ground, as it is powered by the external supply and not the pack itself. Thus, an isolator is used between the microcontroller and the pack voltage input.

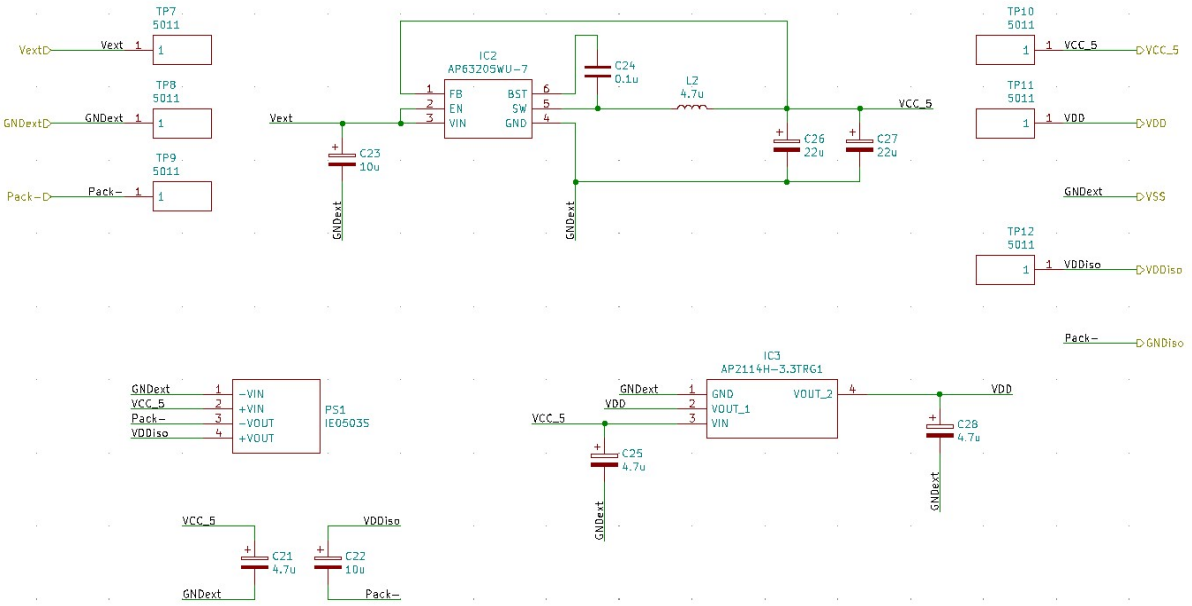


Figure 3. Main Node Power Supply Schematic

The power supply schematics of the main node is shown in Figure 3. Given a 12V of external voltage over Vext and GNDext, IC2 converts the external 12V to the internal 5V line, VCC_5. The IC is set up with four capacitors and an inductor as shown in the datasheet [34]. IC3, a low-dropout regulator, regulates the 5V to the 3.3V line, VDD. The IC is set up with two bypass capacitors as shown in the datasheet [35]. Finally, an isolated 3.3V relative to the battery pack's ground is needed to measure the battery pack's voltage. An isolated 5V to 3.3V DC-DC converter, PS1, is used [36]. The DC-DC converter is set up with two bypass capacitors as specified in datasheets of similar components, as this datasheet did not specify bypass values. Test points (TP) are connected to the power supplies for debugging.

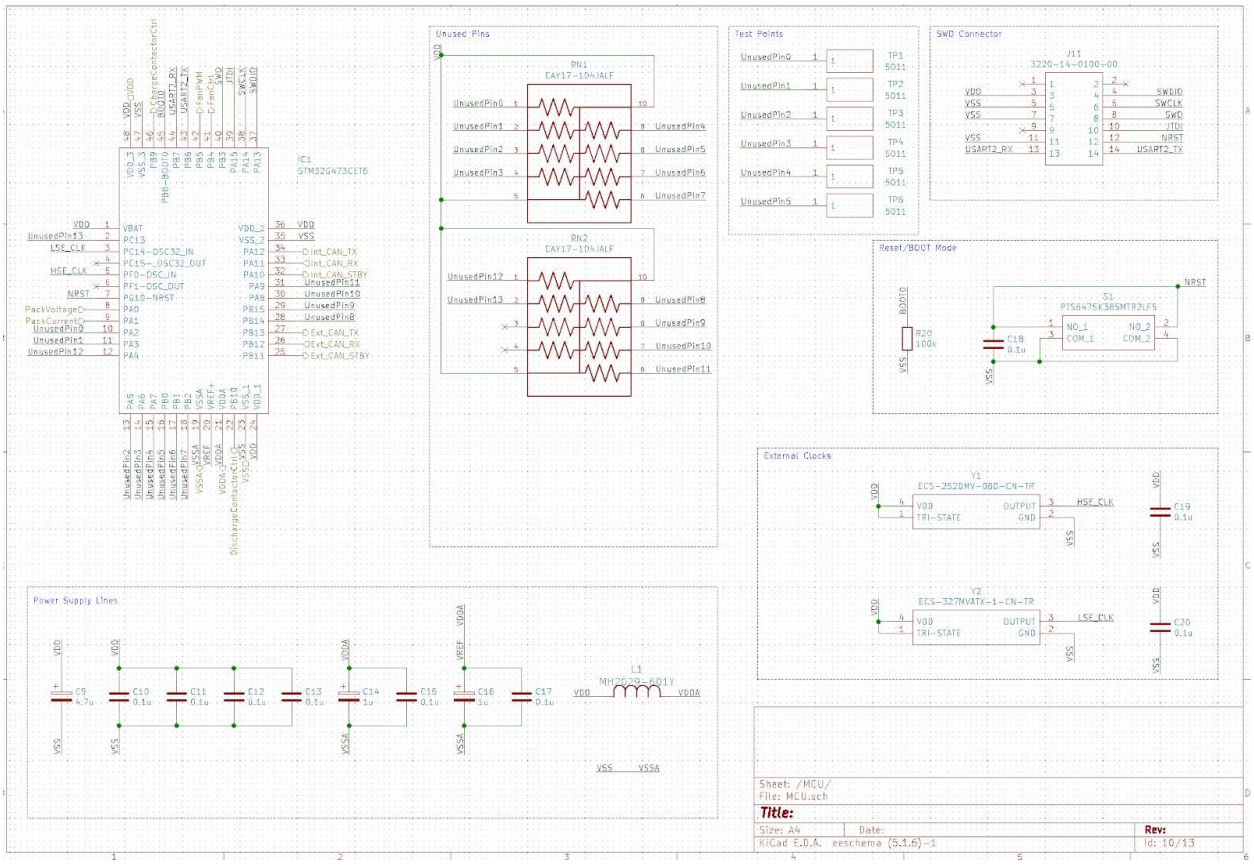


Figure 4. Main Node MCU Schematic

The MCU schematics of the main node are shown in Figure 4. The MCU was set up with bypass capacitors according to the datasheet [37]. A ferrite bead was used to connect the VDD (digital) and VDDA (analog) lines, as shown in the NUCLEO-G474RE development board schematics [38]. This effectively acts as a low-pass filter, so that high frequency noise is filtered out and thus any noise on the VDD line does not interfere with the VDDA line.

All unused pins were pulled up to VDD through resistor networks. Some unused pins were connected to test points in case they were needed for debugging.

The SWD connector was set up to connect to the STLINK-V3MINI, according to its user manual [39].

The reset button was set up similarly to the NUCLEO-G474RE development board, providing a connection to GND (VSS) on the NRST pin when pressed. Similarly, BOOT0 was pulled to GND.

External clocks (crystal oscillator chips) were added according to the respective datasheets [40], [41]. This was preferred over using crystals directly as shown in the NUCLEO-

G474RE development boards since there is less design work to be done this way (each needed a single bypass capacitor, according to the datasheets).

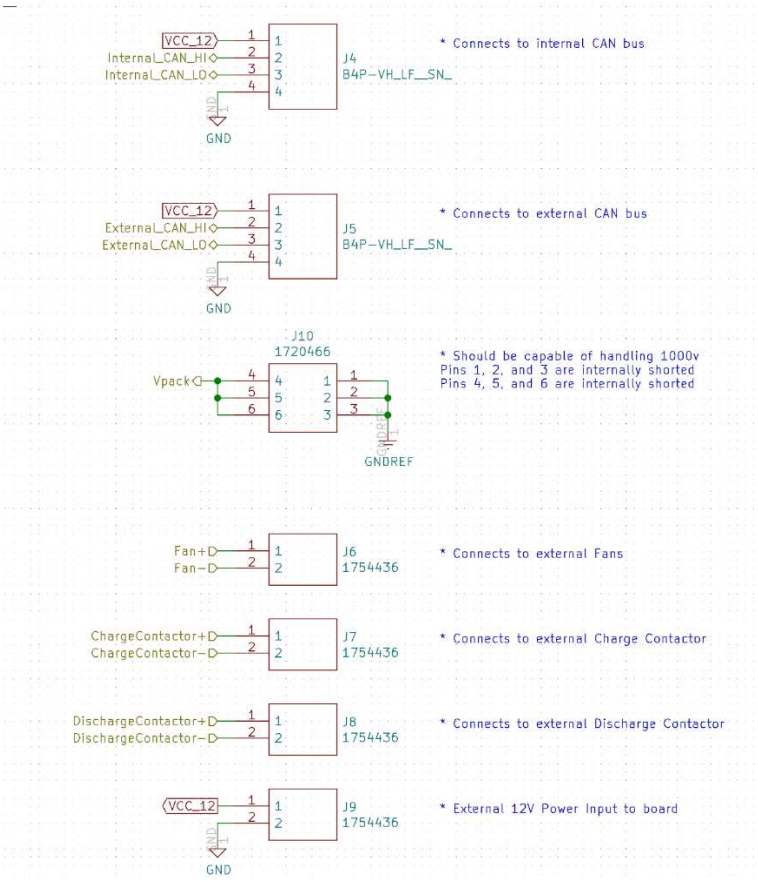


Figure 5. Connectors Schematic

As shown in Figure 1, the main node connects to two contactors and a fan. Figure 5 shows three connectors (J6, J7, J8) added onto the main board to connect the board to these peripherals. There are also two connectors (J4, J5) to connect to the CAN buses. Additionally, there are two connectors to connect to the external 12V supply (J9) and the battery pack (J10). The battery pack connector is much larger than the other connectors since it can accept up to 1000 V difference between the two input pins.

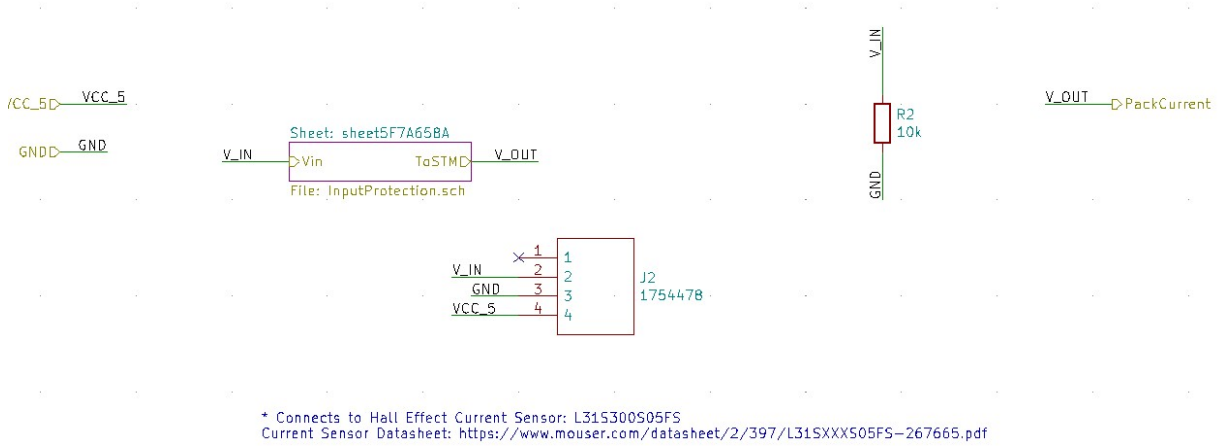


Figure 6. Current Sensor Schematic

The main board also has a connector to the pack current sensor shown in Figure 6. The output of the current sensor has a pulldown resistor (R2) as specified by the datasheet [42] and goes through an input protection circuit shown in Figure 7.

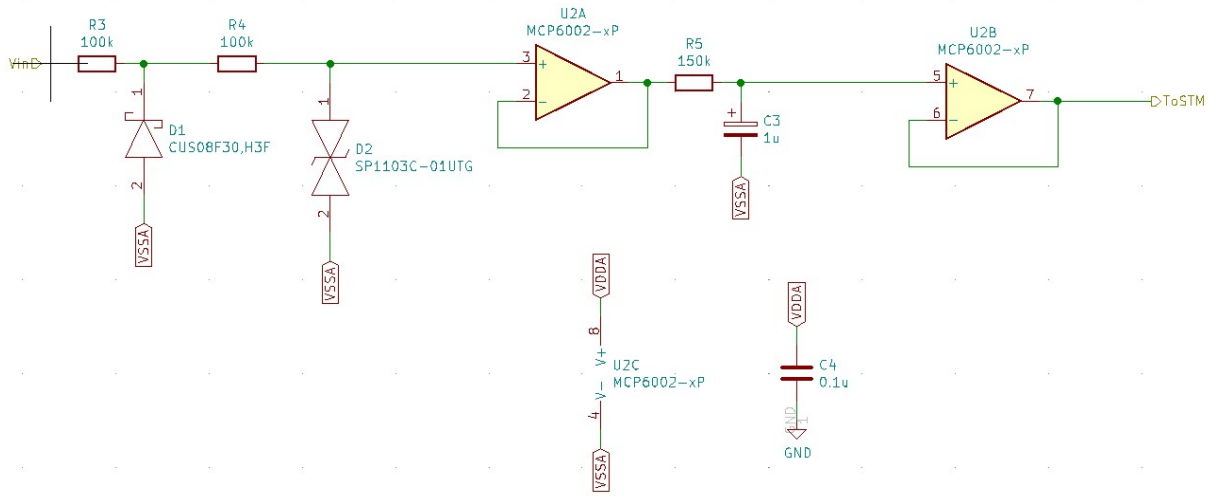


Figure 7. Input Protection Schematic

The input protection schematic shown in Figure 7 is meant to protect the inputs to the MCU from large voltage spikes that could arise due to signals traveling across long wires, and thus have a lot of inductance or capacitance associated with them. The design for this was based off of Digikey's Input Protection article [43] and modified slightly. Specifically, two unity gain buffers were added to ensure the signal going to the microcontroller's ADC inputs had no impedance associated with it. A single dual-package op amp was used for this, and a bypass capacitor was added according to the datasheet [44].

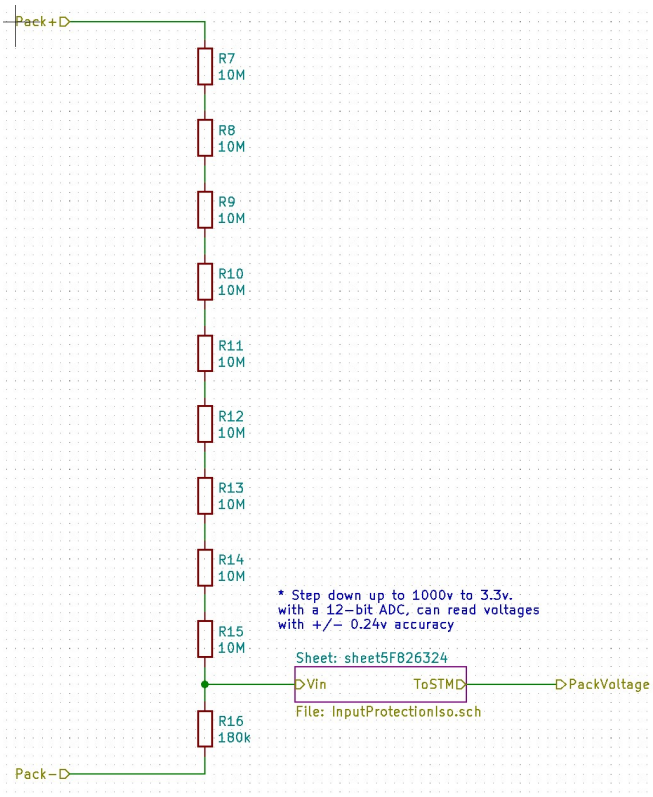


Figure 8. Main Board Step Down Voltage

To measure the voltage of the battery pack, the resistor voltage divider shown in Figure 8 was used. The voltage divider steps down the battery pack, with a maximum voltage of 1000V, to be within 3.3V which is the microcontroller's supply voltage. The output of the voltage divider is passed through the input protection circuit shown in Figure 8. The signal is then passed to the isolation circuit shown in Figure 9 below.

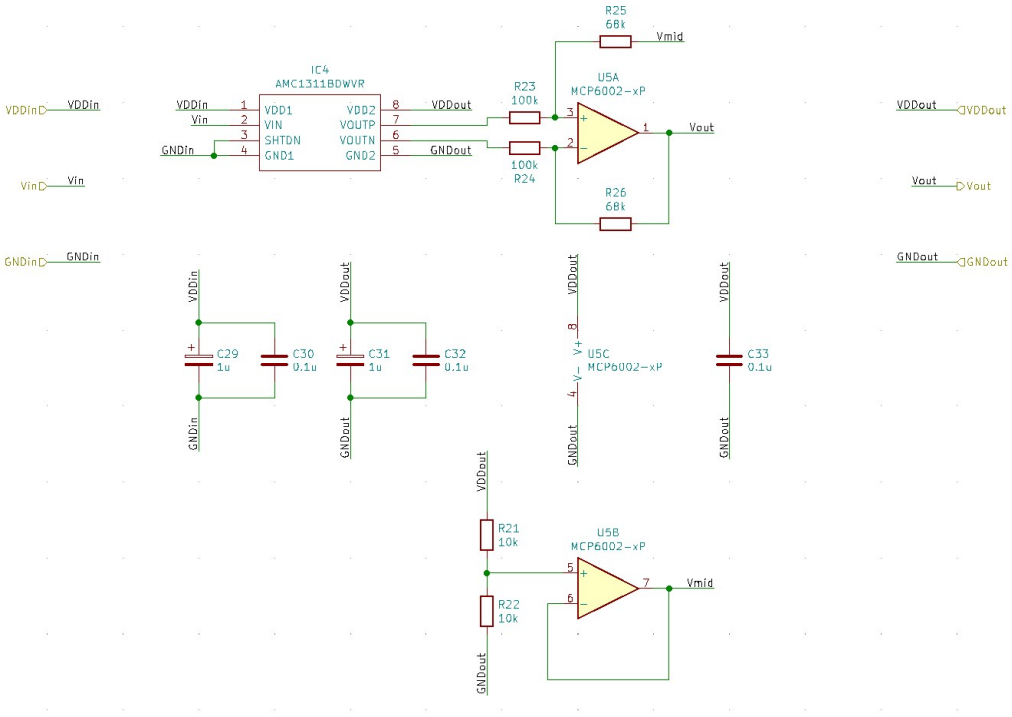


Figure 9. Isolation Schematic

The isolation circuitry shown in Figure 9 converts the stepped down voltage from being relative to the pack ground to the MCU's ground (the external supply ground). The isolator outputs a differential output, so a differential amplifier was used to convert that output to a single-ended output for the ADC to read. Because the pack voltage is expected to always remain positive, however, the Vmid voltage was reduced to 0V (GNDout) by removing R21 and shorting R22, and the differential amplifier gain was increased to increase the resolution of the pack voltage read by the ADC. Bypass capacitors for the op-amp and isolator were added according to their datasheets [45], [46].

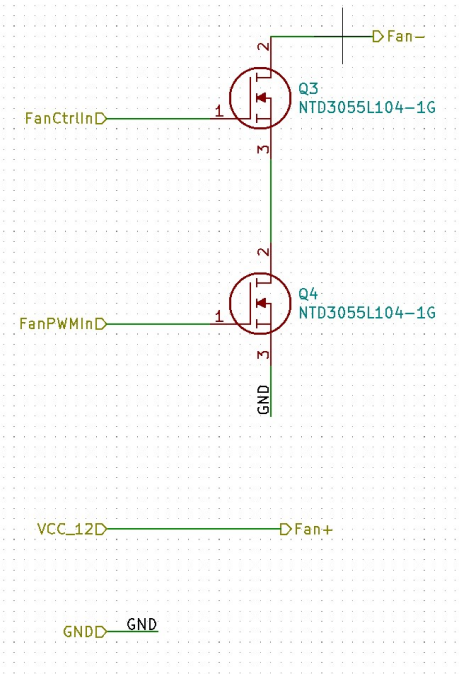


Figure 10. Fan Control Schematic

To control a 12V fan, the 12V external supply is connected to the positive end of the fan. The negative end of the fan is connected to the circuitry shown in Figure 10. The microcontroller has a control output that turns the fan on and off as well as a PWM output that controls the speed of the fan. These outputs are connected to MOSFETs in series that enable the main board to control the fan.

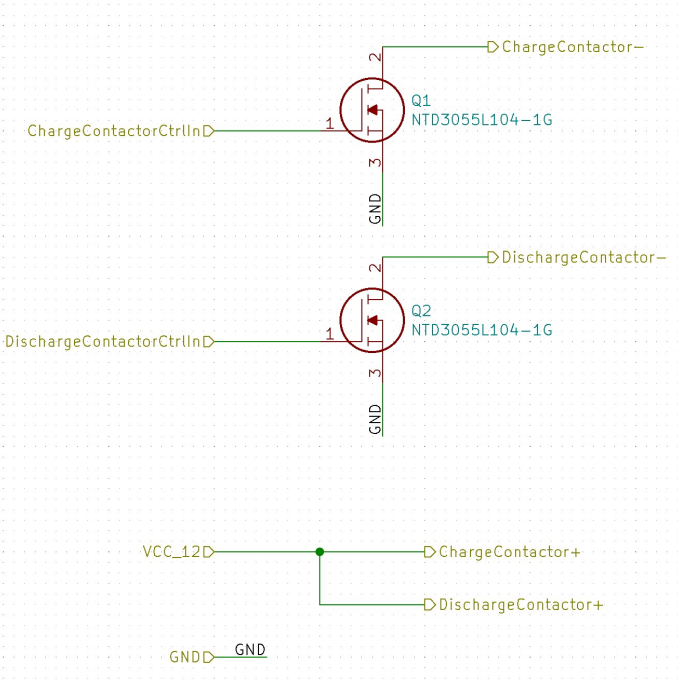


Figure 11. Contactor Control Schematic

To control the contactors, the positive ends of the contactors are connected to 12V and the negative ends are connected to the circuitry shown in Figure 11. The microcontroller's control outputs are connected to the gate of the MOSFETs which allow the microcontroller to trigger the contactors.

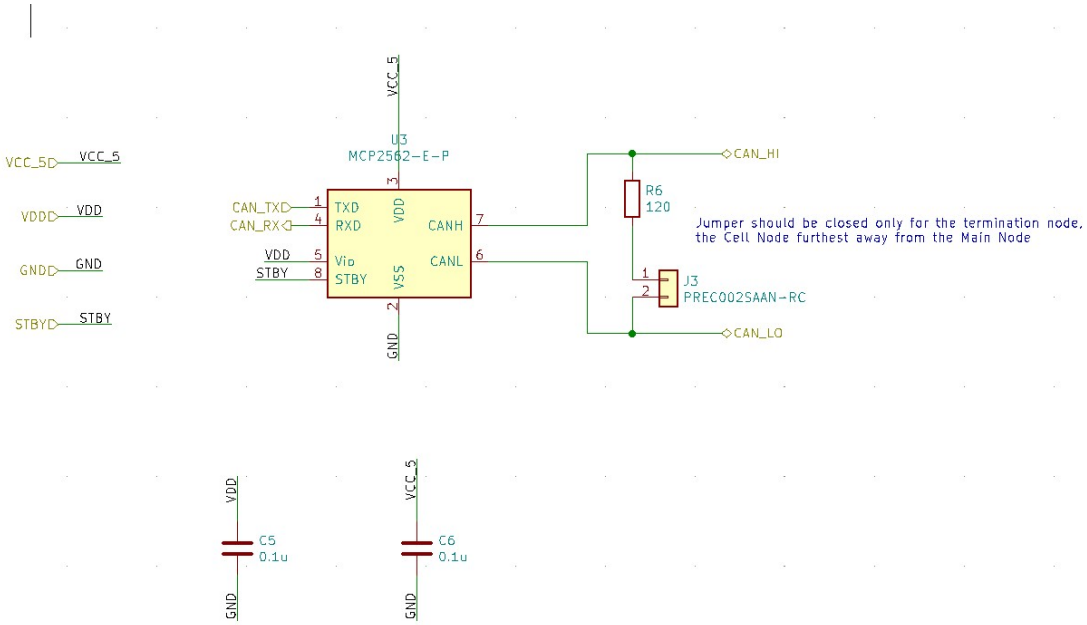


Figure 12. CAN Communication Schematic

The External and the Internal CAN communication blocks used the same sub-schematic for the CAN transceiver. The microcontroller connects to the transceivers via the TX, RX, and STBY pins, allowing the microcontroller to send and receive data as well as control the sleep mode of the transceivers via software. The 120 Ω termination resistor is connected to the CAN HI and CAN LO pins through a jumper. This jumper would only be shorted at the 2 endpoints of the CAN line. Thus, both the External and Internal CAN lines would have this jumper shorted for the main node, with a corresponding 120 Ω termination resistor at the other end of the External CAN line and only one cell node with a 120 Ω termination resistor shorted on the Internal CAN line. Bypass capacitors were added according to the datasheet [45].

PCB

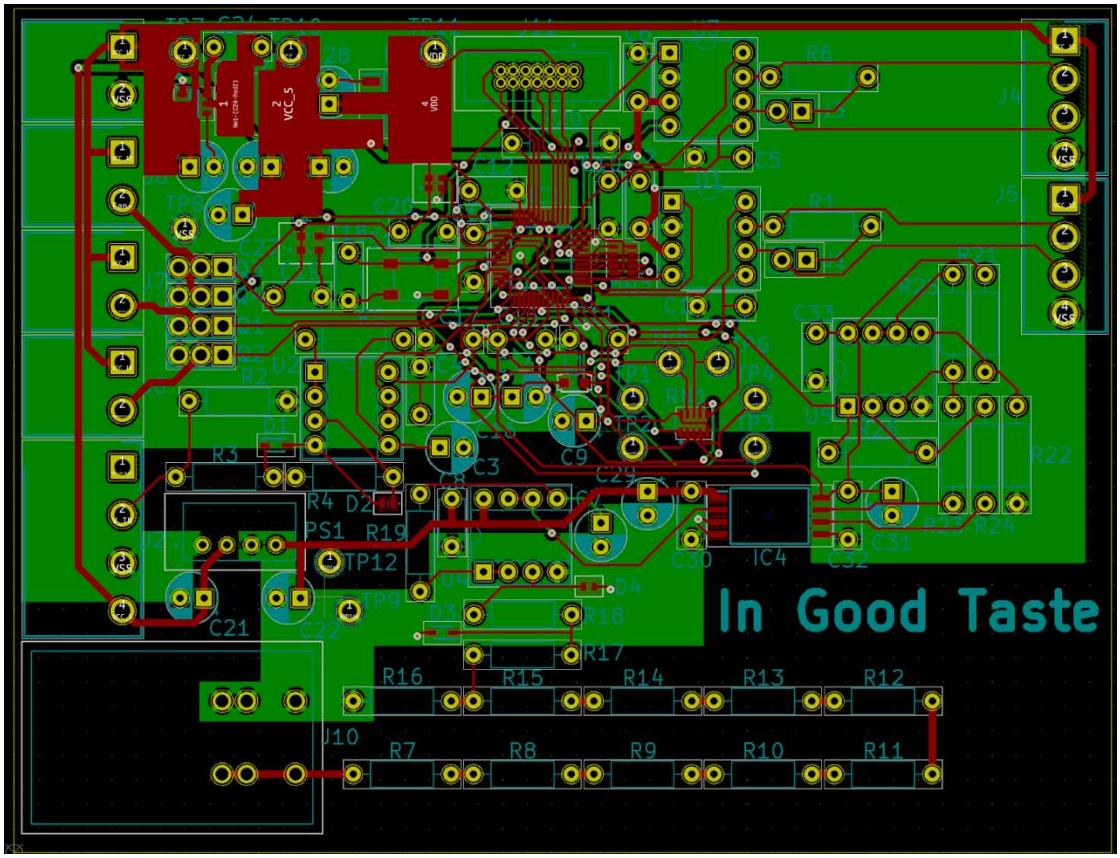


Figure 13. Main Node PCB

The PCB layout for the main node is shown in Figure 13, with the red traces showing copper top and the green traces showing copper bottom. Trace widths of 10 mils were used for signal wires and 30 mils for power lines and most 12 V lines. 30 mils was also used for the voltage divider stepping down the pack voltage.

The main node mostly used through hole components, as space was not an issue. This allowed us to reduce costs by allowing us to populate most of the components ourselves instead of 3W populating everything for us. This also allows for better power dissipation in the step-down circuitry for the pack voltage.

Two separate ground planes were used for the two isolated grounds, and a minimum distance of 60 mils were kept between them at all times to ensure 1000 V of isolation (according to the table provided in KiCad, based on IPC 2221). There was also no ground plane placed below the isolator chip, following the recommendations from the datasheet. The same minimum distance of 60 mils was kept between the pack's ground plane (the small one near the bottom of the board) and the pack's voltage input, as well as between each stage of the associated step-down voltage divider. For this reason, the pack's ground plane was not extended down to the bottom of the board, as that would violate the minimum distance requirement.

The power supply circuitry was placed at the top-left, and as close as possible to the input 12 V from the external supply. Three separate VCC planes were used for the power supplies: one for 12 V, one for 5 V, and one for 3.3 V. All of them were made as large as possible without interfering too much with other traces. This was to be able to dissipate more heat when more power is used on those lines, a good recommendation from the datasheets of the regulators.

When placing components for the layout, care was taken to place bypass capacitors as close as possible to the associated components. If layout recommendations were given in the component datasheets, those were followed as closely as was possible. Component groups were placed as close as possible to minimize the distances signal wires for active components had to travel. Low priority was given to components that were not actively used in the design, like test points and resistor networks for unused pins.

Cell Node

For every battery cell in the battery pack, a cell node is needed. The cell node reports cell voltage and temperature to the main node. The node also accepts cell balancing commands from the main node via CAN and performs passive cell balancing on the battery cell.

Schematic

The schematic for the cell node was also done in KiCad. The schematic is shown in Figure 14.

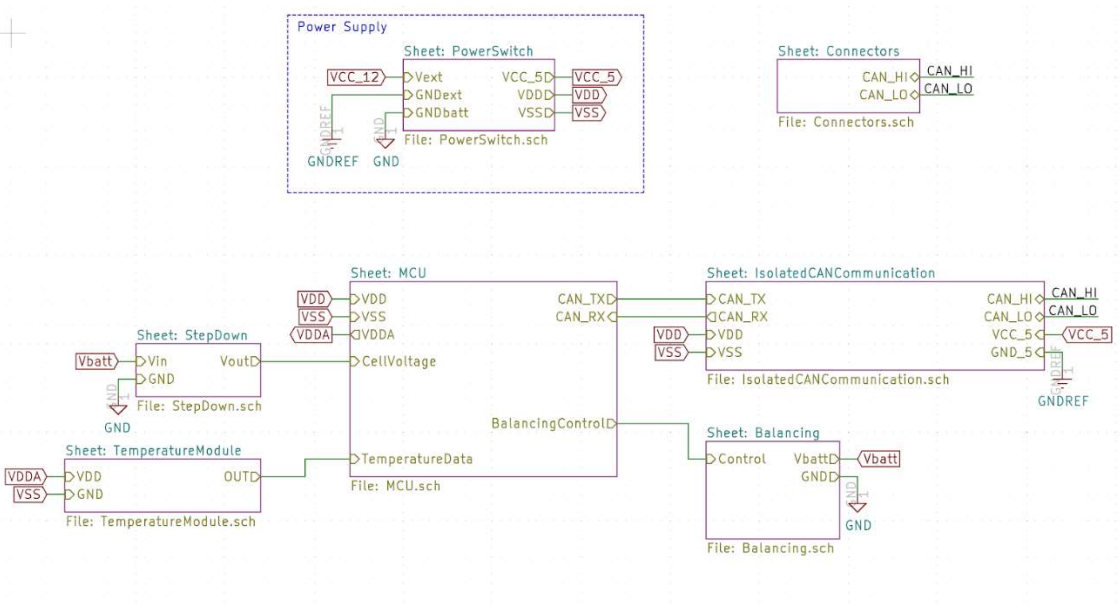


Figure 14. Cell Node Schematic

The cell node has temperature measurement circuitry, cell balancing circuitry, and similar step down, power, and CAN circuitry as the main node. The cell node is only connected to one CAN bus which is connected to the main node and all other cell nodes. The cell board is powered using a 12V external power supply which is converted to 5 volts for the CAN transceiver and 3.3 volts for the microcontroller. The passive cell balancing circuit is used to decrease the state of charge of the corresponding battery cell to equalize SOCs.

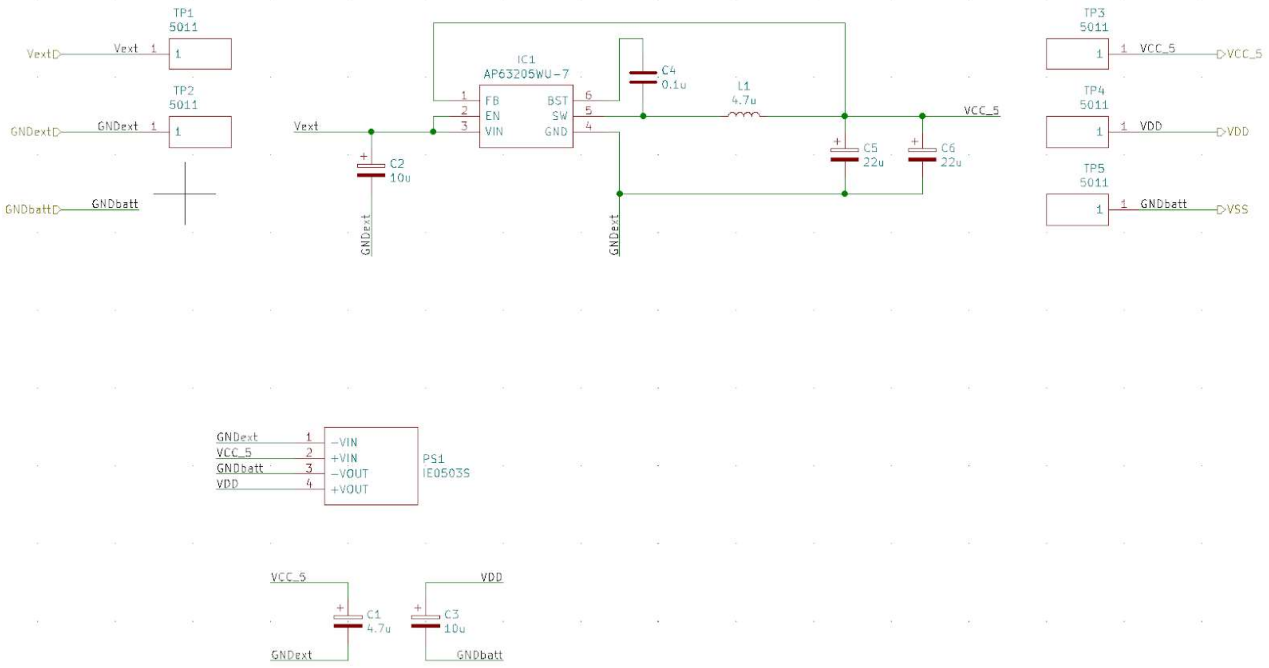


Figure 15. Cell Node Power Supply Schematic

The power supply schematics of the cell node are shown in Figure 15. Given 12V of external voltage over Vext and GNDext, IC1 converts the external 12V to the internal 5V line, VCC_5. The IC is set up with four capacitors and an inductor as shown in the datasheet [34]. Finally, an isolated 3.3V relative to the battery cell's ground is needed to measure the battery cell's voltage. An isolated 5V to 3.3V DC converter, PS1, is used [36]. Two bypass capacitors are added according to the datasheet of other similar components, since this datasheet did not mention bypass capacitors. Test points (TP) are connected to the power supplies for debugging.

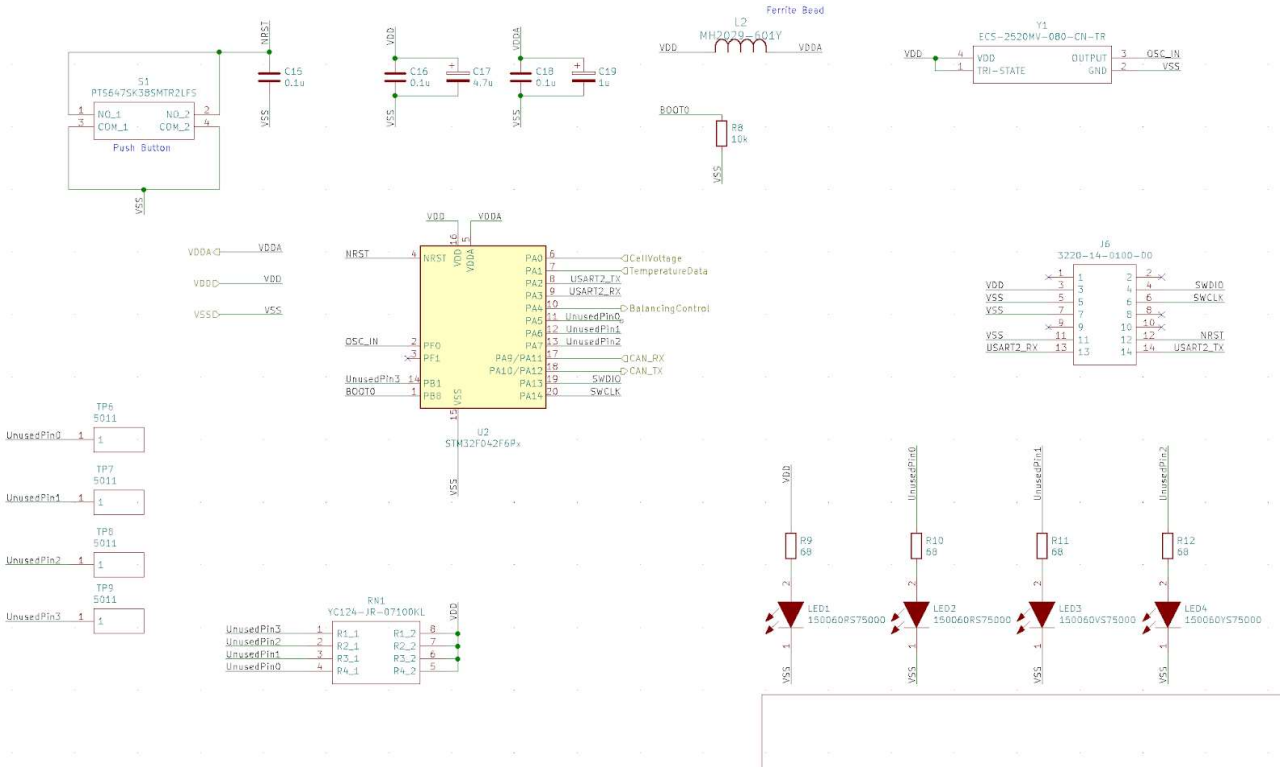


Figure 16. Cell Node MCU Schematic

The STM32F042F6P6 microcontroller was used for the cell board. Similar to the main node, the cell node's microcontroller is connected to a reset button (S1), an external clock, and a SWD connector to connect to our debugger. Additionally, the unused pins were connected to test points, pullup resistors, and LEDs for debugging. The microcontroller was set up with bypass capacitors according to the datasheet [47]. A ferrite bead was used to connect the VDD (digital) and VDDA (analog) lines, similar to the main node. This effectively acts as a low-pass filter, so that high frequency noise is filtered out and thus any noise on the VDD line does not interfere with the VDDA line.

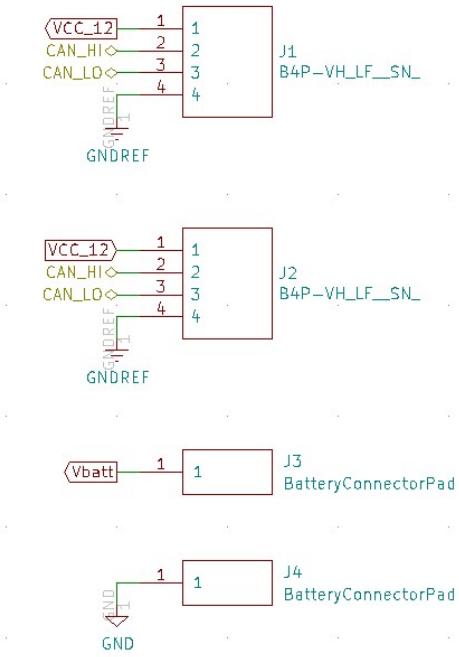


Figure 17. Cell Node Connectors Schematics

Each cell node is mounted onto its respective battery cell. Figure XX shows the connectors (J1, J2, J3, J4) that the cell board needs. J3 and J4 are to connect to the two ends of the battery (they are simply pads to solder nickel strips to), and J1 and J2 connect to the internal CAN bus, the same way the main node connects to the CAN bus.

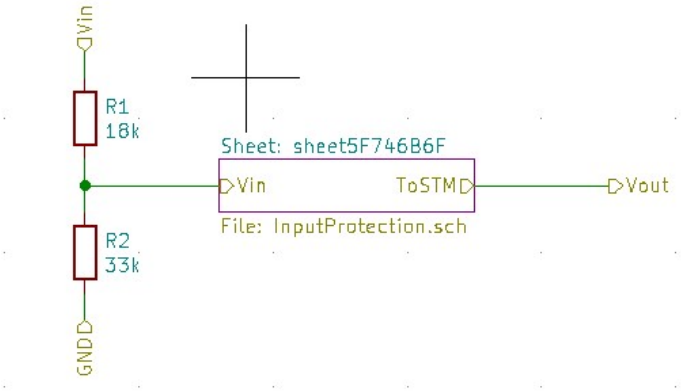


Figure 18. Cell Node Voltage Divider

Similar to the main node, the cell node has a voltage divider to measure the voltage of its battery cell. The voltage divider shown in Figure XX is designed to step down a maximum voltage of 5V to be within 3.3V. The output of the divider is passed through the input protection circuit shown in Figure 7.

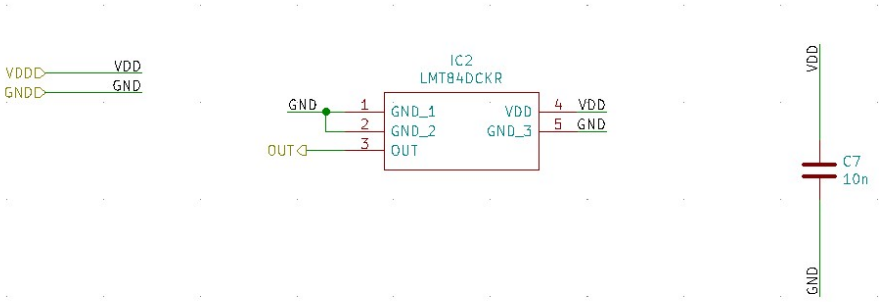


Figure 19. Temperature Module Schematic

The temperature module in Figure 19 measures the ambient temperature and scales its output voltage accordingly. The sensor can measure temperatures from -50 degrees Celsius to 150 degrees Celsius. A bypass capacitor is added to the circuit as recommended in the datasheet [48].

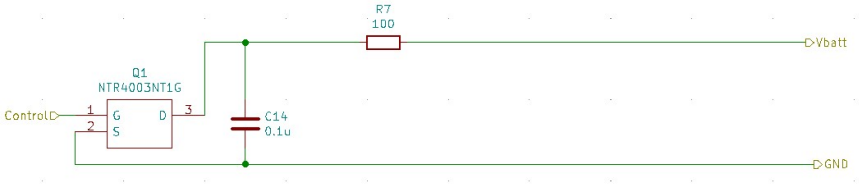


Figure 20. Cell Balancing Schematic

A passive cell balancing circuit, shown in Figure 20, is used to equalize different battery cells. When one battery cell has a higher voltage than the other batter cells, the cell board can send a control input to the MOSFET (Q1). This will cause the battery to supply more charge for current to run through the resistor (R7) which will decrease its voltage. Once the higher charged battery's voltage is decreased to be in line with the other cells, the control input can be set to low. Passive cell balancing is easy to implement, as the circuit in Figure 20 is simple. However, it is less efficient than other cell balancing methods [33].

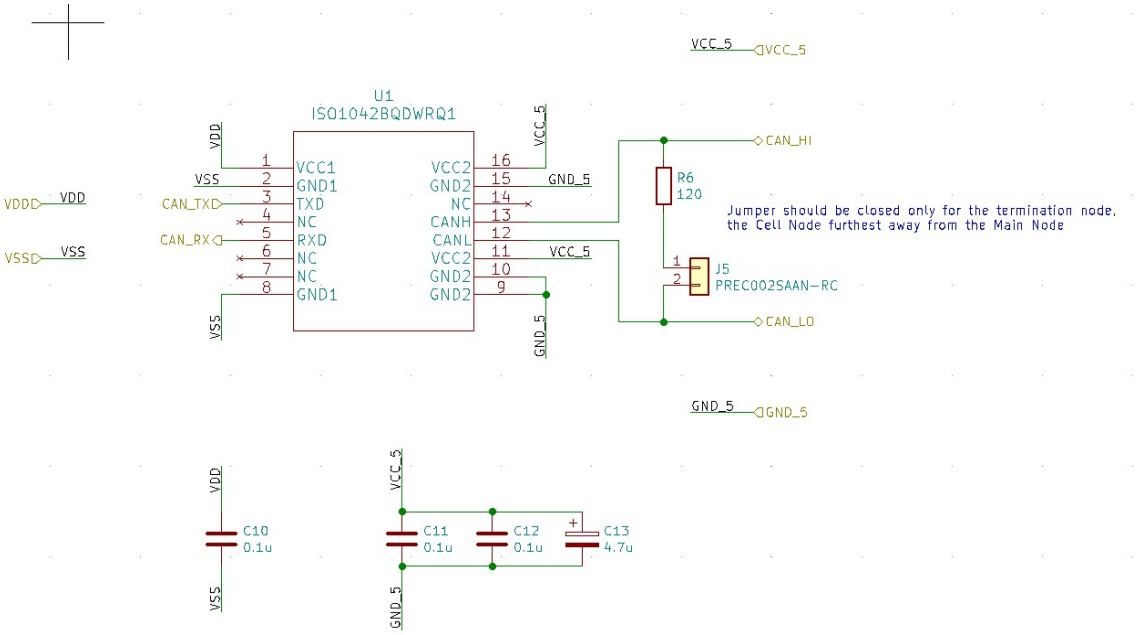


Figure 21. Isolated CAN Schematic

The CAN transceiver used in the cell node is different from the one used in the main node. This is because the cell node must use an isolated CAN transceiver, as the CAN HI and CAN LO lines are relative to the external supply's ground while the microcontroller is powered relative to the battery cell's ground. Thus, the isolated CAN transceiver is powered differently on two sides: on the microcontroller side, it is powered from the VDD line, the same 3.3 V line powering the MCU (relative to the battery cell's ground). On the other side, the transceiver is powered from VCC_5, the 5 V line relative to the external supply's ground. This was stepped down from the 12 V line passed along to this cell node with the CAN HI and CAN LO lines (also relative to external supply's ground).

The isolated CAN transceiver did not have a STBY pin, unlike the main node's CAN transceivers, so the microcontroller cannot control the sleep mode of the transceiver. Bypass capacitors were added according to the datasheet [49].

PCB

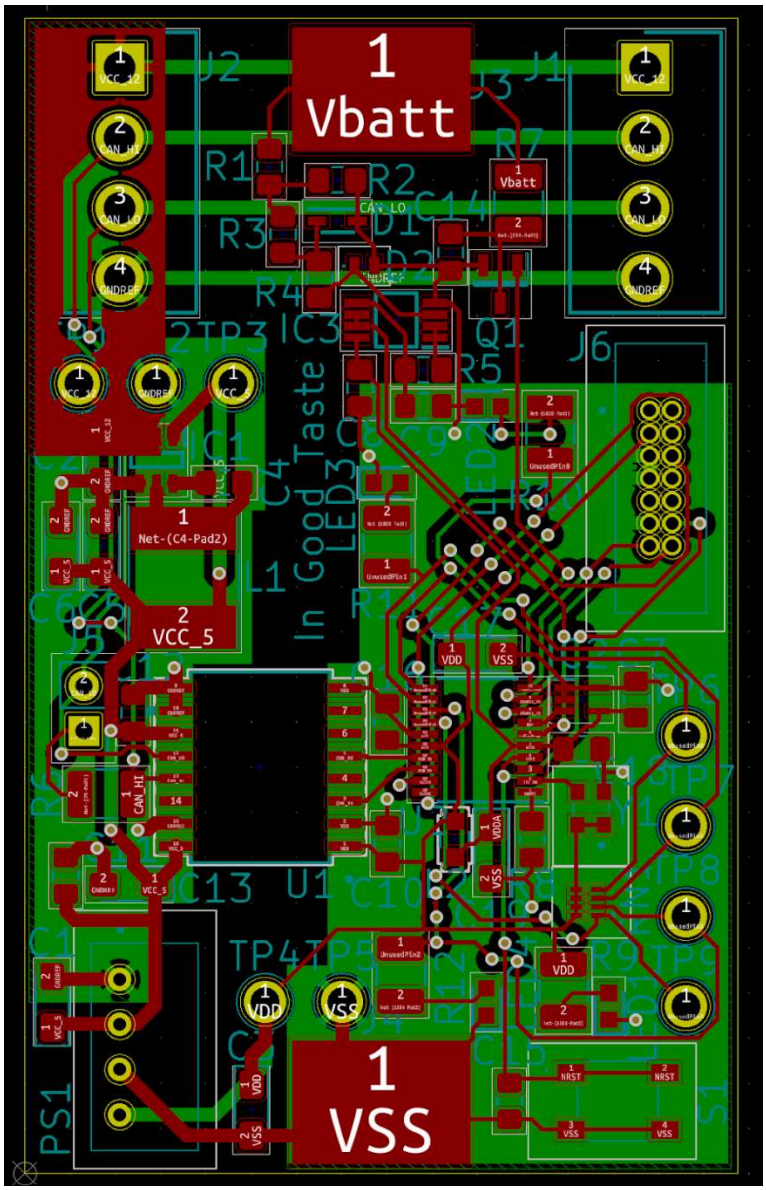


Figure 22. Cell Board PCB

Figure 22 shows the PCB layout for the cell board. This board mostly contained SMD components to keep the board size small. One goal of this project was to have the cell board be small enough to clip onto the side of the battery cell. The dimensions of the cell board were 65mm by 40mm so that the cell board is the same height and width as two batteries. [50]. This allows us to clip the cell nodes onto the sides of the battery pack by alternating the side each adjacent cell's board is attached to. Cells are attached to the battery pack via the Vbatt and VSS pads that read the cell voltage. Nickel strips were soldered to the Vbatt and VSS pads and securely screwed down under the copper bus bars used in the Vruzed battery caps.

Similar to the main node, trace widths of 10 mils were used for signal wires and 30 mils for power lines and most 12 V lines.

Two separate ground planes were used for the two isolated grounds, and a minimum distance of 60 mils were kept between them at all times to ensure 1000 V of isolation (according to the table provided in KiCad, based on IPC 2221). There was also no ground plane placed below the isolator chip, following the recommendations from the datasheet.

The power supply circuitry was placed at the top-left, and as close as possible to the input 12 V from the external supply. Due to the limited space, only one VCC plane was used for the 12 V line; the 5 V and 3.3 V lines simply used 30 mil traces.

When placing components for the layout, care was taken to place bypass capacitors as close as possible to the associated components. If layout recommendations were given in the component datasheets, those were followed as closely as was possible. Component groups were placed as close as possible to minimize the distances signal wires for active components had to travel. Low priority was given to components that were not actively used in the design, like test points and resistor networks for unused pins.

Software

For this project, software was written using the Mbed OS framework and uploaded to the boards using the PlatformIO IDE. Figures 23 and 24 below show the high-level software flow charts for the software on both the cell node and main node.

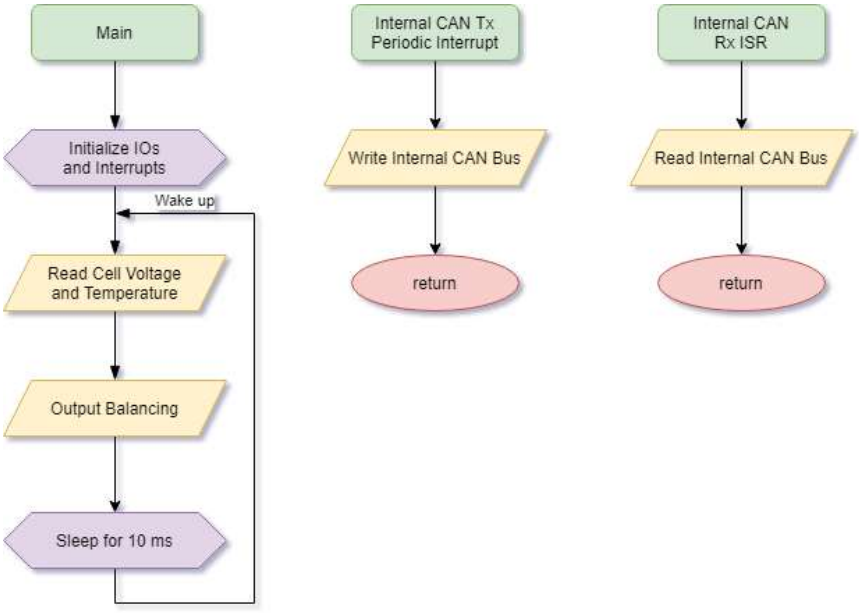


Figure 23. Cell Node Software Flow Chart

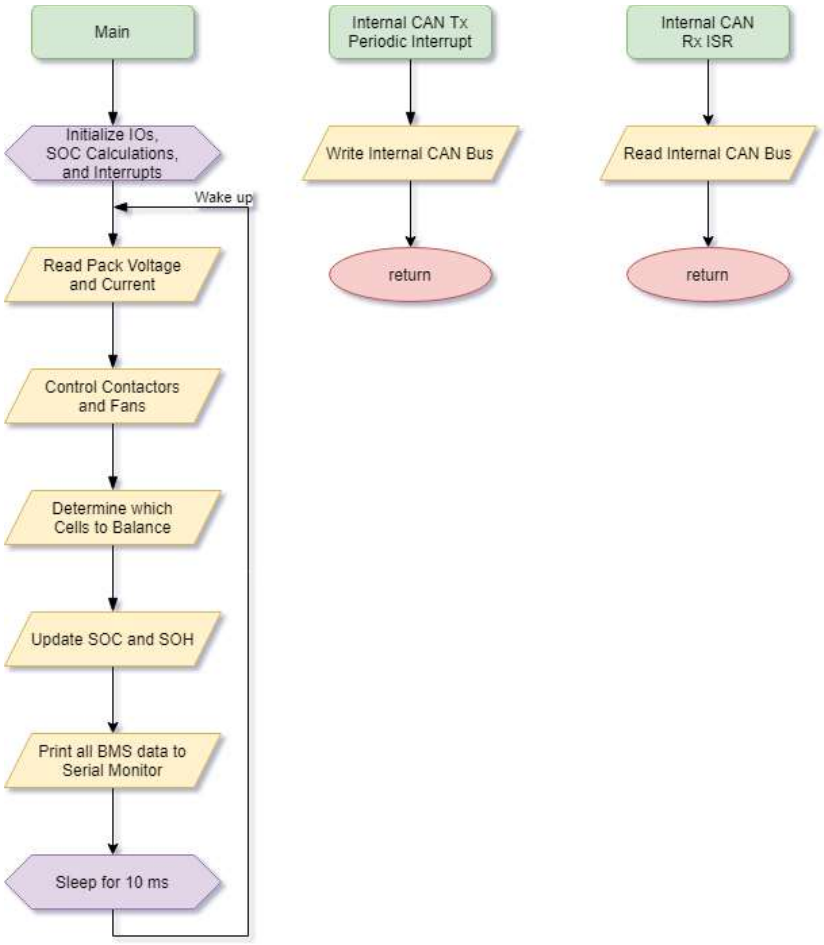


Figure 24. Main Node Software Flow Chart

The cell node software was originally planned to also perform cell health parameters, but as the microcontroller was much more severely constrained in memory usage than originally anticipated, that was moved to the main node, which has a much larger memory capacity. Thus, the cell node software only needs to measure the cell voltage and temperature and control cell balancing based on instructions received from the main node over CAN. The cell node also relays the measured cell voltage and temperature to the main node over CAN.

The main node will read pack voltage and pack current. When a CAN message comes from a cell board, the main node will read the cell node's temperature and cell voltage. Using this data, the main node will estimate pack health, control the contactors, fan, and send cell balancing commands. If the pack current is above a threshold current or the pack health algorithm (detailed in the next section) shows that a battery is fully charged or fully discharged, the main board will trigger a contactor. If the current is positive, it will trigger the discharging contactor and if the current is negative, it will trigger the charging contactor.

The fan logic is shown in Figure 25. If any cell temperature is above the temperature threshold (25 °C), the main board will turn on the fan. The speed of the fan is controlled by a PWM signal from the main board. The higher the temperature, the faster the fan will run.

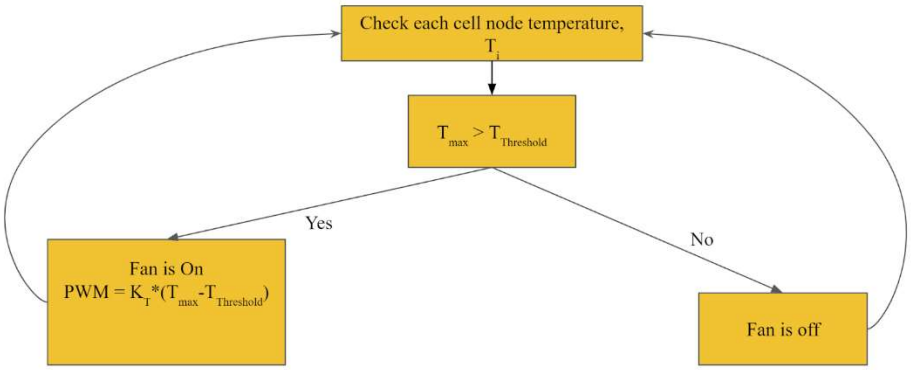


Figure 25. Fan Software Logic

The cell balancing logic is shown in Figure 26. The main node collects voltage information from each cell node. Given the minimum cell node voltage, the main node sends a cell balancing command to cell nodes who have a voltage that is greater than the minimum voltage by a threshold (.01V). Once the voltages are equal, the cell balancing command will be deactivated.

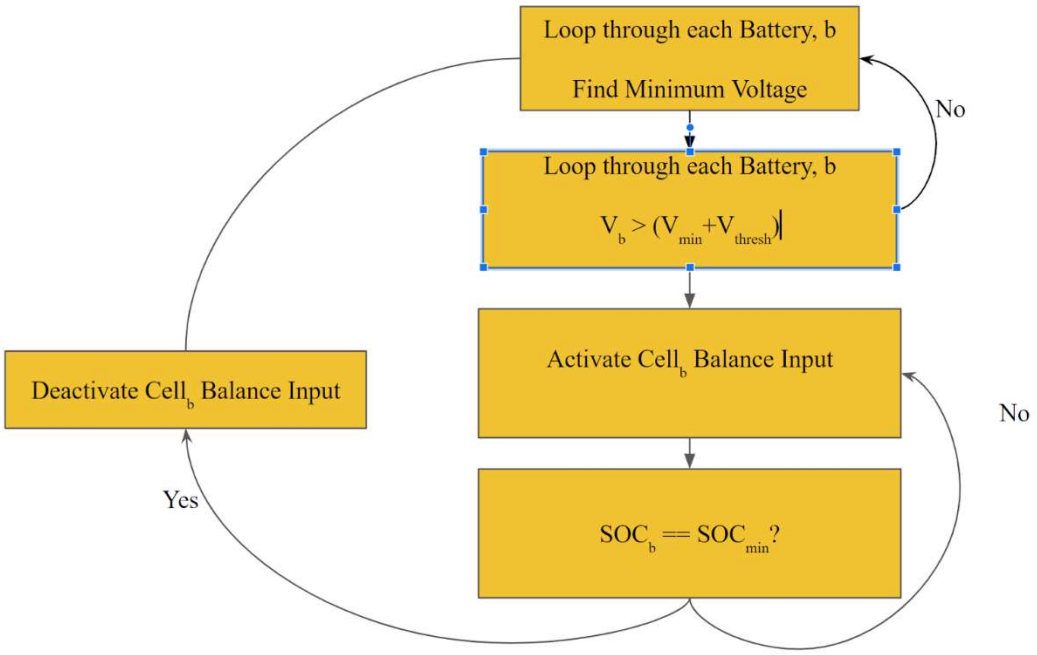


Figure 26. Cell Balancing Logic

Pack Health Algorithm

The battery pack health metrics that we wanted the BMS to estimate was state of charge (SOC) and state of health (SOH) of the battery pack. To do this, the BMS finds the SOC and SOH of each battery cell and reports the average metrics. The state of charge is the ratio of a

battery cell's releasable capacity to its maximum capacity expressed as a percentage using the following equation:

$$SOC = \frac{C_{releasable}}{C_{rated}} * 100\%$$

The SOC increases as a battery is charged and decreases as a battery is discharged. The state of health of a battery shows how new a battery is. The SOH is the ratio of the battery's maximum releasable capacity to the maximum capacity of the battery prior to any degradation, which is expressed as a percentage using the following equation:

$$SOH = \frac{C_{max}}{C_{rated}} * 100\%$$

For a new battery, the SOH starts at 100%. As the battery goes through charging and discharging, the battery degrades and the amount of charge that the battery can hold decreases. As the battery is used, SOH decreases.

Voltage based SOC estimation

The Voltage method to determine the SOC of a battery involves using the reading of the battery voltage to obtain the equivalent SOC value using the known discharge curve (voltage vs. SOC) of the battery. The voltage vs SOC graph is shown in Figure XX. Our average discharging current was 3.5A, so we used the 4A curve on the graph. This method has some inaccuracies due to how the battery voltage is affected by temperature and discharge rate.

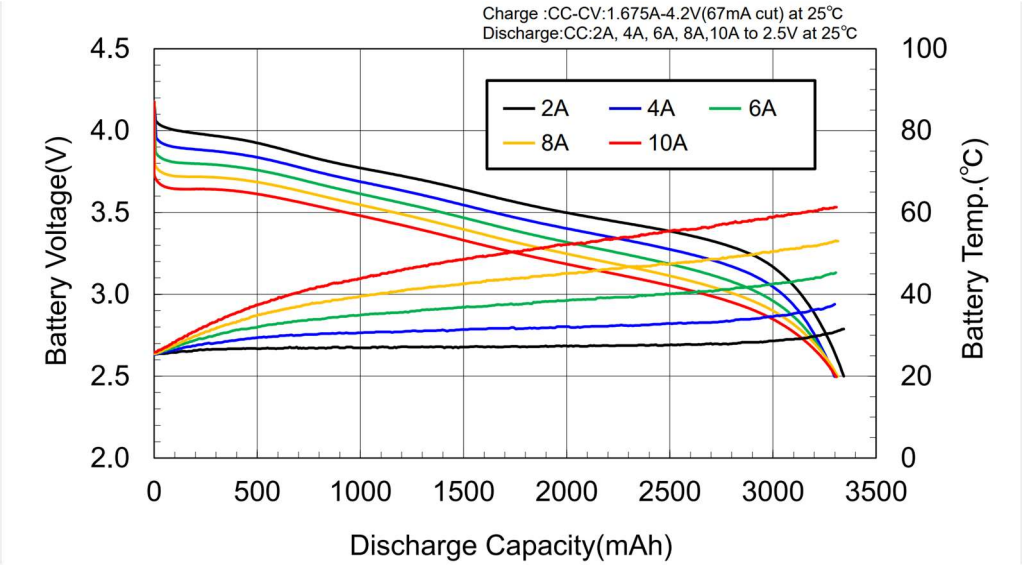


Figure 27. Voltage vs. SOC Graph for 18650 Batteries [51]

Coulomb Counting SOC estimation

The Coulomb Counting Method uses the current sensor to determine the remaining capacity. Assuming a battery starts with a SOC of 100%, we can integrate current over time to see how many Coulombs have entered and exited the battery, which results in the following equation:

$$SOC(t) = SOC(t_0) + \frac{1}{C_{rated}} \int_{t_0}^{t} I \, dt$$

When the battery is charging, the current will be positive and when the battery is giving power to a load, the current will be negative. There are downsides to this approach, however, as the accuracy of this method is dependent on the accuracy of the initial SOC estimation. This initial SOC estimation must also be re-calibrated regularly as the battery capacity decreases. Furthermore, the battery will always deliver less charge during discharge than was put into it during charging, making it difficult to get an accurate SOC estimate. This means that there would be accumulated errors in the SOC measurement. Measurement errors from the current sensor will also contribute to the accumulated errors in the SOC estimate.

Enhanced Coulomb Counting

To address the drawbacks of the Coulomb Counting Method for SOC Estimation, several additional factors are introduced to obtain a more accurate estimate of the SOC for the battery.

When the battery is discharging, the depth of discharge (DOD) can be expressed as the percentage of the capacity that has been discharged relative to C_{rated}, where $C_{released}$ is the amount discharged by the battery.

$$DOD = \frac{C_{released}}{C_{rated}} * 100\%$$

The difference in DOD over a period τ can be found using the following equation while measuring the charging and discharging current.

$$\Delta DOD = -\frac{1}{C_{rated}} \int_{t_0}^{t_0+\tau} I(t) \, dt * 100\%$$

This results in the following equation for DOD(t):

$$DOD(t) = DOD(t_0) + \Delta DOD$$

Using these values, the SOC for a battery can be estimated using the following equation:

$$SOC(t) = SOH(t) - DOD(t)$$

Initially, for a new unused battery, the SOH is assumed to be healthy and is equal to 100%. The SOH can then be recalibrated by accumulating the sum of the total charge put into the

battery after it is fully charged or by using the accumulated DOD value when the battery is exhausted. The battery is fully charged when the battery voltage reaches the upper limit voltage (V_{max}) and the current declines to the lower limit (I_{min}). The battery is fully discharged when the battery voltage is less than the lower limit (V_{min}).

The Enhanced Coulomb Counting Method for SOC Estimation can be summarized in the following flow diagram:

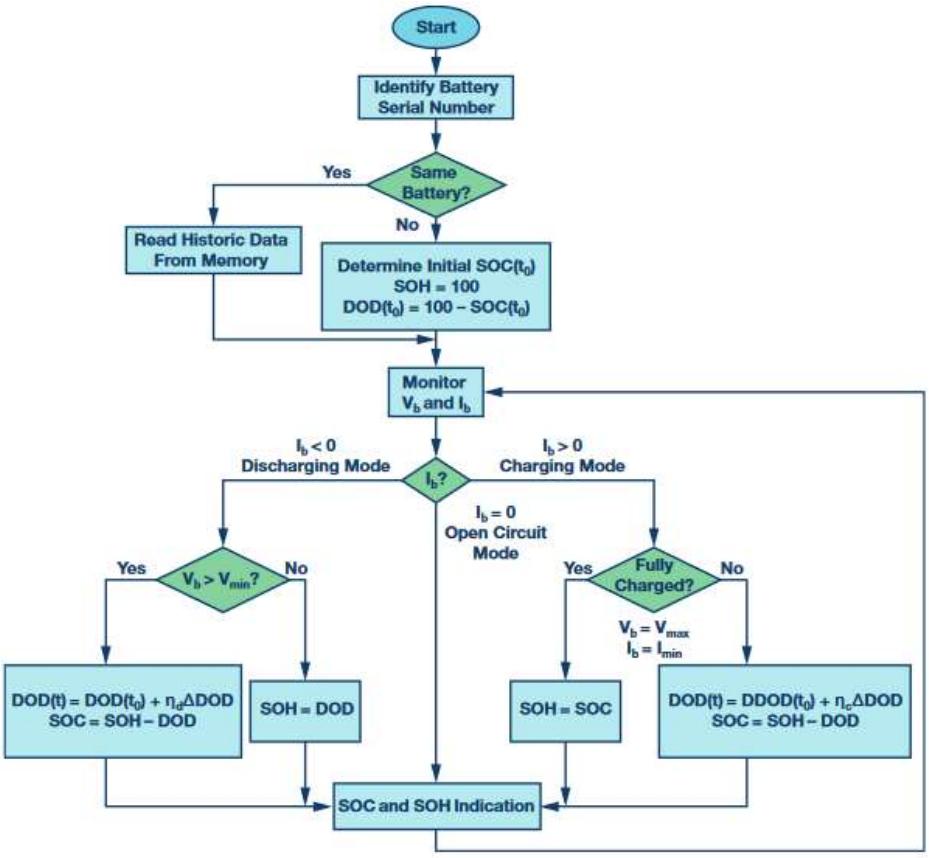


Figure 28. Enhanced Coulomb Counting Flow Diagram

Combined Coulomb Counting and Voltage-based for SOC Estimation

The cumulative error of the counting integrator from the Coulomb Counting method can be removed using a counter calibration based on the OCV characteristics while the battery is not in use. The current is otherwise integrated to get the relative charge in and out of the battery. A state diagram illustrating this approach is shown below:

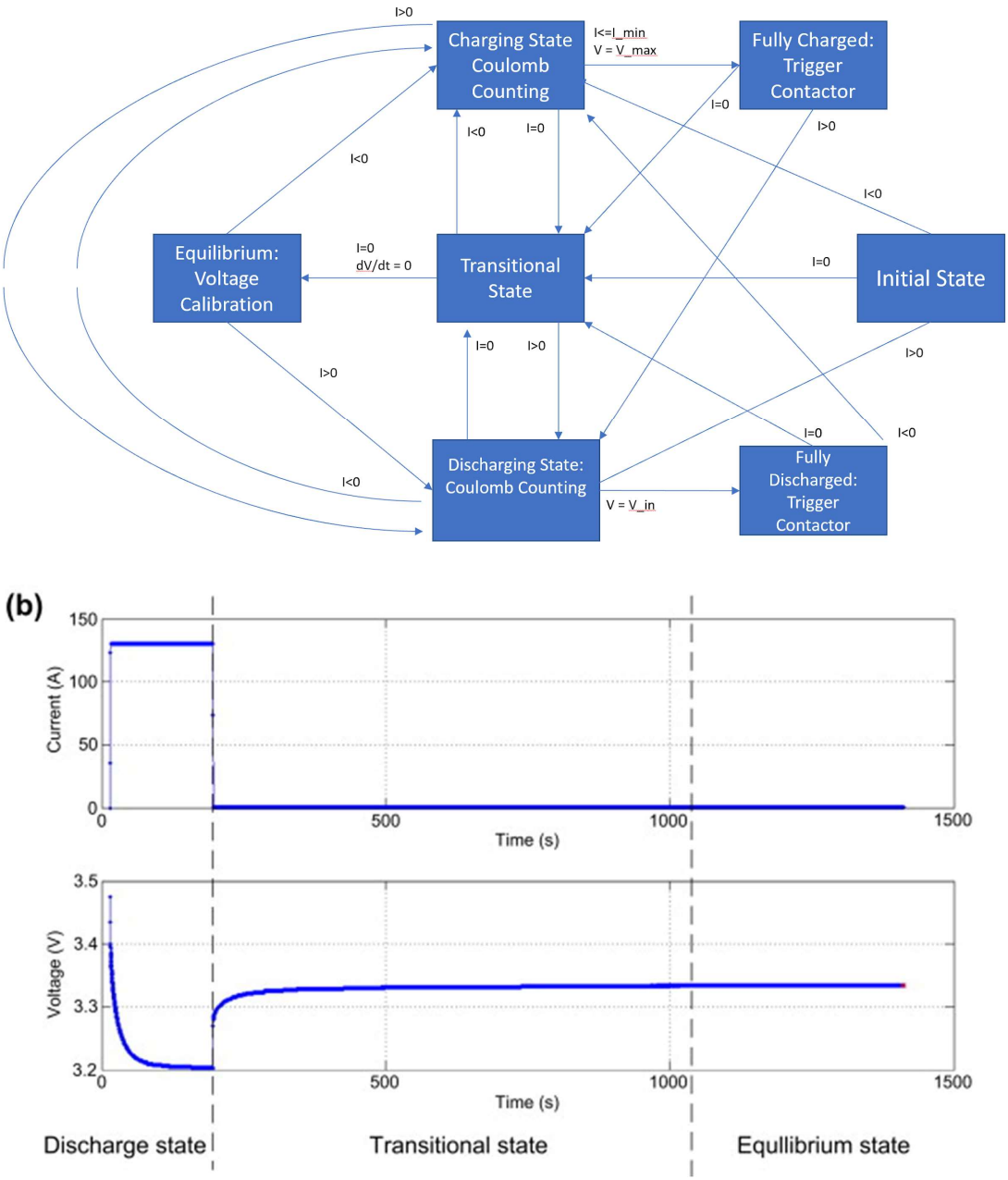


Figure 29. State Diagram of Combined Coulomb Counting and Voltage-based for SOC Estimation

A change in the state diagram is detected by monitoring the voltage and current with the coulomb counting being used in almost all states besides "Equilibrium state" and "Fully Charged". In equilibrium state, the battery voltage is stable and the OCV voltage characteristic is used to calibrate the SOC. Another calibration is done when the cell is fully charged because this state is easily detectable during the constant-voltage charging mode. When the current reaches a minimal value known as end of charging current (EoCC), the charging process is complete and the battery is fully charged.

This method has a reported estimation error of less than 3% during full cell cycling. This SOC estimation error can, however, increase to more significant levels if a battery is used continuously and resting times between charge and discharge cycles are not long enough to reach voltage equilibrium and perform calibration.

SOC Algorithm Decision

Enhanced Coulomb Counting seems to be the best option for SOC estimation because it does not need advanced battery information like the Kalman filter approach, and it is usually more accurate than voltage based SOC estimation. However, Coulomb Counting assumes that all batteries in series have the same SOC because they have the same current. This is false because each battery will be slightly different, and a BMS needs to understand these differences in SOC in order to perform cell balancing. Therefore, we decided that the SOC algorithm to use is the Combined Coulomb Counting and Voltage-based method for SOC estimation to get an estimate that uses current and voltage. This method was chosen because it would reduce the accumulated errors from using the Coulomb counting method as this method is relatively accurate in the short term, and the periodic recalibrations using the OCV would ensure that the error from the Coulomb counting method would not get too significant. The Combined Coulomb Counting and Voltage-based for SOC Estimation method was implemented in software. This algorithm uses lots of readings from the current sensor, whose uncertainty is discussed in Appendix A.

CAN Specification

CAN communication was used for the boards to communicate between each other (internal CAN bus) and for the main board to communicate with an external PC program (external CAN bus). Each CAN message had an 11-bit identifier and 8 bytes of data. The 11-bit identifier also represented the priority of the CAN message. In the internal CAN bus the 11-bit identifier is split into the upper 4 bits to show the message type and the lower 7 bits to show the board identifier. The board identifier is used to distinguish between cell boards sending the same type of message. The 7-bit board identifier allows for 128 nodes on the internal CAN bus (1 main node and up to 127 cell nodes). Floating point numbers are sent in the CAN bus by multiplying them by a constant and casting them as integers. The internal CAN specification is shown below:

- Internal Message 0, Main Board to Cell Board
    - Bits 0 to 63: Balancing command for cell boards 0 - 63 (ordered sequentially)
- Internal Message 1, Main Board to Cell Board
    - Bits 0 to 63: Balancing command for cell boards 64 - 127 (ordered sequentially)
- Internal Message 2, Cell Board to Main Board
    - Bits 63 to 48: Cell Voltage (V, Divide integer by 10000 to get cell voltage float)
    - Bit 47: Cell Temperature Sign Bit
    - Bits 46 to 40: Cell Temperature (°C)

In the external CAN bus the 11-bit identifier is split into the upper 10 bits to show message type and the lowest 1 bit for the board identifier (main board or PC program). The external CAN specification is shown below:

- External Message 0, Main Board to PC Program
    - Bits 63 to 56: Pack Average State of Charge (%, Divide integer by 2 to get float)
    - Bits 55 to 48: Pack Average State of Health (%, Divide integer by 2 to get float)
    - Bits 47 to 32: Pack Voltage (V, Divide integer by 100 to get float)
    - Bit 31: Pack Current Sign Bit
    - Bits 30 to 16: Pack Current (A, Divide integer by 100 to get float)
    - Bit 15: Pack Highest Temperature Sign Bit
    - Bits 14 to 8: Pack Highest Temperature (°C)
    - Bit 7: Pack Average Temperature Sign Bit
    - Bits 6 to 0: Pack Average Temperature (°C)
- External Message 1, Main Board to PC Program
    - Bits 63 to 0: Cell Voltage for Device 0, 1, 2, and 3 (V, Divide integer by 10000 to get cell voltage float)
- External Message 2, Main Board to PC Program
    - Bits 63 to 0: Cell Voltage for Device 4, 5, 6, and 7 (V, Divide integer by 10000 to get cell voltage float)
- …
- External Message 32, Main Board to PC Program
    - Bits 63 to 0: Cell Voltage for Device 124, 125, 126, and 127 (V, Divide integer by 10000 to get cell voltage float)

PC Program

In order to view the status of the BMS, an external interface was needed. The design of the main node accommodated a second CAN bus interface to provide a way for the BMS to communicate with either a computer connected to the same CAN bus, or with another vehicle module when the BMS is deployed in a vehicle. For demonstration purposes and simplicity, the CAN specification was designed for the former case. To enable the status of the BMS to be viewed as easily as possible, there needed to be two main components, one that allowed for a GUI for receiving BMS data as well as sending commands to the BMS, and another that relayed messages to and from the GUI as CAN frames on a CAN bus interface, acting as a bridge. The interconnection between the two was decided to be WebSockets, as the protocol is programming-language agnostic and is supported by all major web browsers. WebSocket clients can be in any language, and the bridge acting as a WebSocket server also opens up the ability for remote connection to the bridge program.

CAN Bridge Program

The bridge program was chosen to be written in the Rust programming language, as it is a type safe and memory-safe language that has performance on par with, and in some cases, faster than C. Along with the tokio library, Rust additionally provides powerful asynchronous and multithreaded programming constructs that allow for more performant programs and more efficient use of hardware resources.

The main structure of the program consists of two asynchronous background tasks and a single foreground task. The primary foreground task opens a socket to the CAN Bus interface and listens for CAN Frames. The first background task receives messages from this main foreground task using an inter-thread communication channel, specifically a multi-producer, single-consumer channel (MPSC). When the main foreground task receives a CAN Frame, it partially decodes a frame, encodes it into a JSON string, and then transmits it over the MPSC channel. When the first background task receives the encoded message, it relays it out over the open WebSocket. The second background task listens for incoming WebSocket messages and parses the message data as a command. While the original intention was for bi-directional bridge support, there was not enough time to implement both, so the CAN-to-WebSocket functionality was prioritized and finished.

GUI Program

The GUI program was chosen to be a simple front-end web application, utilizing an HTML page with a client-side JavaScript program run on page load. The original plan for the program was to both display the BMS data in tabular and graphical form. The web page would include a table showing a labeled layout of the battery packs and their associated voltages, updating on every new WebSocket message received, while graphs could show the battery pack status information over time. Additionally, the web page would have different configuration options available to change the behavior of the BMS. Unfortunately, due to time constraints, there was not enough time to implement the full functionality. The basic functionality of receiving the WebSocket message and message deserialization and some parsing logic was implemented, with text results updated live on the page on every new WebSocket message received.

Problems and Design Modifications

The team faced several challenges that led to major design modifications. Two of these design changes occurred during the design phase, prior to any PCB fabrication, while three of these were only discovered while testing the fabricated boards. The following are listed in chronological order of their occurrence.

Initially, the team wanted to use a wireless communication network such as Bluetooth Low Energy (BLE) between the nodes to reduce wiring for the various nodes. However, BLE would have severely limited the maximum number of nodes possible in the star topology the team was planning to use. On top of that, no members of the team had any experience working

with Bluetooth or BLE, so it would have been much more difficult for the team to diagnose and solve any issues encountered with using BLE. However, as some members of the team had prior experience with the wired CAN communication, it was chosen over BLE.

To complement the originally intended wireless communication, it was initially planned that the cell nodes would be able to be powered from the cells they monitored. However, with the switch to a wired communication protocol, this would provide little benefit, as each cell would be connected by a wire to the main node anyways. On top of that, it was difficult to find components for the power supply schematics for the cell node, as a non-standard voltage level would have to be used to power the microcontroller and ensure the cells' lowest voltage level would be able to power the microcontroller. Thus, the team decided to change the design requirements to only allow the cell nodes to be powered off of the external power supply. This would also ensure more stability for the BMS in the case of undervoltage of the cells

After the PCBs were fabricated, it was found that the main node could not be flashed for programming or debugging, as it was constantly held in reset. The team spent a lot more time than anticipated in diagnosing this issue, by eliminating various possible reasons the microcontroller could be held in reset one by one. In the end, cutting the hardware reset trace (NRST pin on the microcontroller) solved the issue, indicating the problem was a hardware fault with the PCB and not a defect internal to the microcontroller chip (it was later discovered that the NRST pin of the MCU was shorted to a UART pin on the STLink connector). This resulted in the team having to use software reset instead of hardware resets for the firmware to upload code to the microcontroller.

Another hardware issue was discovered on the main node while testing in which the LSE_CLK signal on the board from the low-speed external oscillator (32.768kHz) did not appear to be present and only exhibited noise, while the HSE_CLK signal was present and showing the correct signal. Upon further investigation with a microscope and reading the datasheet for the oscillator chip, it was discovered that the chip had been soldered by 3W Electronics in the wrong orientation, rotated 90 degrees from the correct placement. Unfortunately, this issue was discovered too late for the issue to be resolved. As a workaround, the software target was configured to instead use the internal LSI (low-speed internal oscillator) as a clock source by setting a flag "lse_available" to 0.

Software support for both the STM32G473 microcontroller on the main node and the STM32F042 microcontroller on the cell node was also an issue. It was discovered that in the latest version of Mbed OS v6, the support for the STM32F042 platform had been dropped. This led to issues with building the code for the cell node; however, this issue was resolved through porting over old source files from an earlier version of Mbed OS with support for the platform. The opposite problem was also encountered with the STM32G473 platform; as the platform was new enough that there did not exist fully complete support for it. This was resolved with some research into manually generating the necessary code for the platform. It was found that

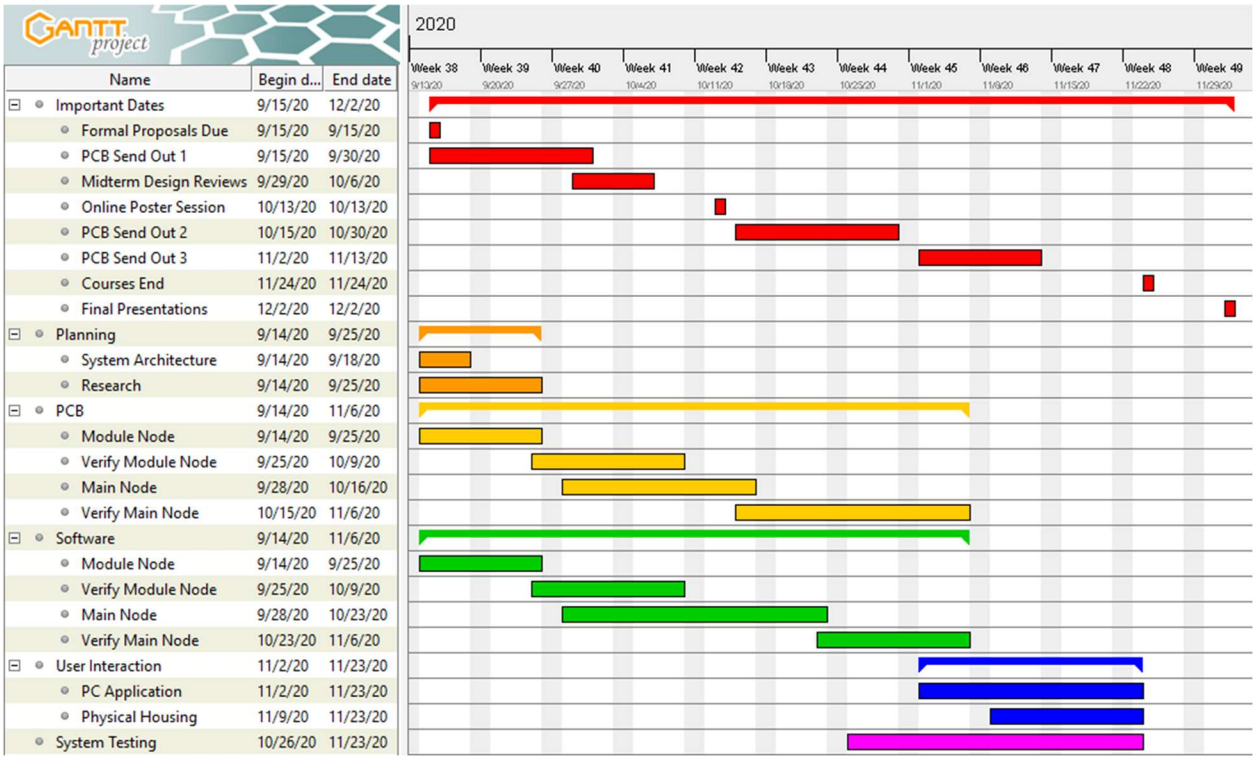
installing the STM32CubeMX program and using an included Python script "STM32_gen_PeripheralPins.py" in the Mbed OS source allowed for manual generation of the necessary microcontroller initialization code and drivers.

It was also discovered that the code size was reaching the limits of the STM32F042 microcontroller selected for the cell nodes. This was not anticipated, as the code for the cell node is much simpler and shorter compared to the main node, but the team failed to account for the size of the Mbed OS framework used. Thus, the team decided to move the cell health calculations from the cell nodes to the main node. This would reduce the memory utilization of the cell nodes enough to ensure proper functionality.

The final challenge that resulted in major hardware design changes was the accuracy of the pack voltage and current sensors on the main node. Because the team was expecting the target application could vary across a wide range of battery pack configurations, the maximum pack voltage was designed to be 1000 V and the maximum pack current was designed to be 300 A. However, the team failed to anticipate that sensors designed for these wide ranges of input voltages and currents would be vulnerable to high levels of noise for small voltage and current levels. This made the sensors useless for measuring the 5 to 8.4 V and 1 to 5 A levels used for testing. Thus, major hardware changes were required to lower the design input voltage and current ranges. The voltage range was reduced to 200 V by lowering the voltage divider ratio used to step down the pack voltage. The current range was reduced to 100 A by amplifying the signal from the hall effect signal using an instrumentation amplifier. Noise was also reduced in the current sensor output signal by bypassing the signal with two parallel capacitors.

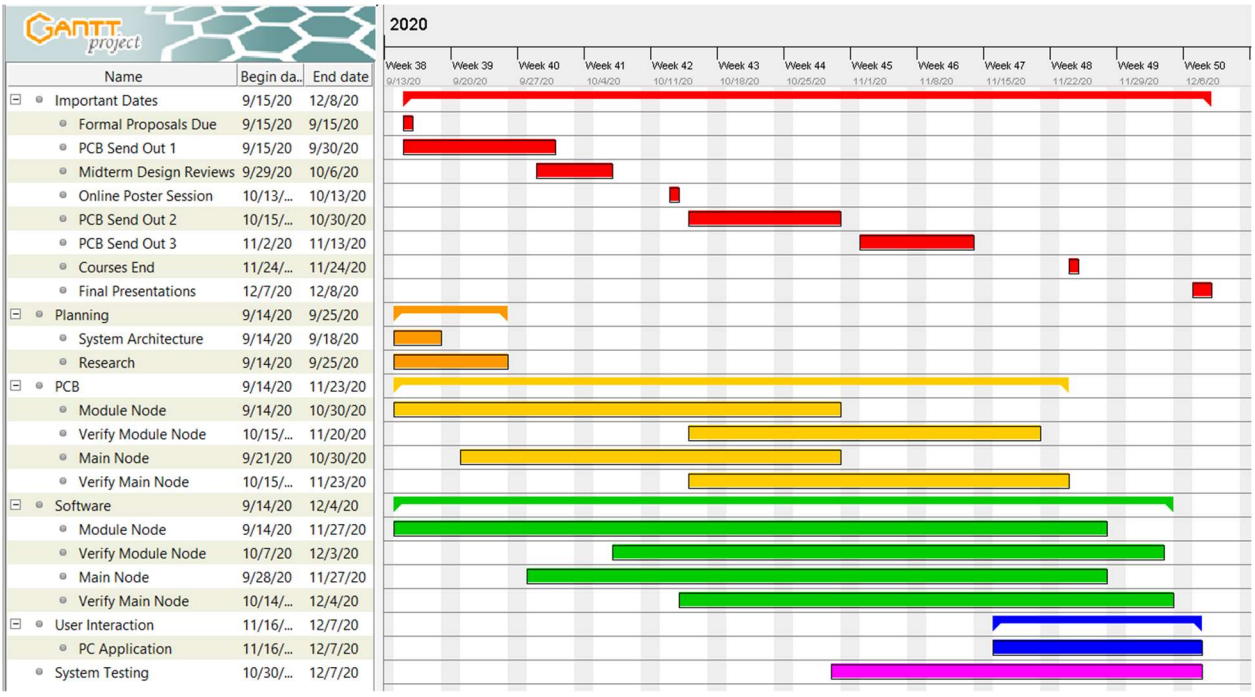
The team also ran out of time and was not able to implement the external CAN bus, so that was left as a future improvement. However, the specifications for the message types on the external CAN bus were designed, as explained in the CAN specification section above.

Project Time Line



Name	Begin d...	End date
Important Dates	9/15/20	12/2/20
Formal Proposals Due	9/15/20	9/15/20
PCB Send Out 1	9/15/20	9/30/20
Midterm Design Reviews	9/29/20	10/6/20
Online Poster Session	10/13/20	10/13/20
PCB Send Out 2	10/15/20	10/30/20
PCB Send Out 3	11/2/20	11/13/20
Courses End	11/24/20	11/24/20
Final Presentations	12/2/20	12/2/20
Planning	9/14/20	9/25/20
System Architecture	9/14/20	9/18/20
Research	9/14/20	9/25/20
PCB	9/14/20	11/6/20
Module Node	9/14/20	9/25/20
Verify Module Node	9/25/20	10/9/20
Main Node	9/28/20	10/16/20
Verify Main Node	10/15/20	11/6/20
Software	9/14/20	11/6/20
Module Node	9/14/20	9/25/20
Verify Module Node	9/25/20	10/9/20
Main Node	9/28/20	10/23/20
Verify Main Node	10/23/20	11/6/20
User Interaction	11/2/20	11/23/20
PC Application	11/2/20	11/23/20
Physical Housing	11/9/20	11/23/20
System Testing	10/26/20	11/23/20

Figure 30. Proposal Gantt Chart



Name	Begin da..	End date
Important Dates	9/15/20	12/8/20
Formal Proposals Due	9/15/20	9/15/20
PCB Send Out 1	9/15/20	9/30/20
Midterm Design Reviews	9/29/20	10/6/20
Online Poster Session	10/13/...	10/13/20
PCB Send Out 2	10/15/...	10/30/20
PCB Send Out 3	11/2/20	11/13/20
Courses End	11/24/...	11/24/20
Final Presentations	12/7/20	12/8/20
Planning	9/14/20	9/25/20
System Architecture	9/14/20	9/18/20
Research	9/14/20	9/25/20
PCB	9/14/20	11/23/20
Module Node	9/14/20	10/30/20
Verify Module Node	10/15/...	11/20/20
Main Node	9/21/20	10/30/20
Verify Main Node	10/15/...	11/23/20
Software	9/14/20	12/4/20
Module Node	9/14/20	11/27/20
Verify Module Node	10/7/20	12/3/20
Main Node	9/28/20	11/27/20
Verify Main Node	10/14/...	12/4/20
User Interaction	11/16/...	12/7/20
PC Application	11/16/...	12/7/20
System Testing	10/30/...	12/7/20

Figure 31. Final Gantt Chart

The Gantt chart in Figure 30 above shows the original project timeline, including deadlines (in red). The two main aspects of the project are the PCB and embedded software for

both the module (cell) nodes and the main node. The PCB and software can be implemented in parallel while the module nodes will be implemented before the main node, since the main node will also be acting as a module node, just with extra features (pack current sensing, data analysis, etc.). Verification of each node's PCB and software can be done individually and then testing of the whole system can be done afterwards. User interaction is also an important feature, but the PC application can be very simple (doesn't need a GUI). This task can be implemented in parallel as well, but will be implemented after the nodes are done and if there is time.

Figure 31 shows what our team's final Gantt chart looks like after completing the Modular BMS. There were significant differences between our proposed and final Gantt chart due to unexpected delays and hurdles that our team faced. We were not able to finalize the design of the cell board before the first PCB sendout as the board layouts for the cell were not completed before that time. The cell boards were therefore sent out in the second and third PCB sendouts. Our team encountered problems with flashing to the Main Node board, which required extensive troubleshooting before we were able to find a workaround to it. This significantly increased the time it took to verify that the Main Node was usable for the Modular BMS. Additionally, this setback combined with sending cell board layouts as part of the third PCB sendouts delayed the completion of the tasks that relied on these PCBs such as the software for the Main and Cell node. As our team had a significantly shorter amount of time to complete the rest of the modular BMS due to these changes in our Gantt chart, we decided to remove the task of creating the physical enclosure for the modular BMS as it was not essential to the modular BMS' core functionality.

The responsibilities for this project were divided in the following manner: Data Acquisition of individual module data was primarily handled by Nripesh. Communications, including CAN, was primarily handled by William and secondarily handled by Dipesh. Control, including relay and fan outputs as well as cell balancing, was primarily handled by Dipesh and secondarily handled by Niki. Data Analysis of the pack health was primarily handled by Niki and Phillip and secondarily handled by William. User Interaction, which involves building the PC Application, was primarily handled by William and secondarily handled by Phillip. Powering the system was primarily handled by Dipesh and secondarily handled by Nripesh. Testing was primarily handled by Phillip, and secondarily handled by Dipesh, Nripesh, Nikilesh, and William.

Test Plan

The test plan for the modular BMS was broken down into two major components: hardware and software components. The hardware component of the PCBs were tested first, and the software components were tested afterwards. To test the hardware for our boards, we would measure the voltage at various test points and pins to ensure that the PCB was designed and assembled correctly. This methodology was applied to verify the following components: Isolation, Cell and Main Node Power supply. Additionally, the voltage for the 18650 batteries

and Power Convertor, and the resistance of the reset button and resistor was verified. The test plans for these components are shown in the following figures.

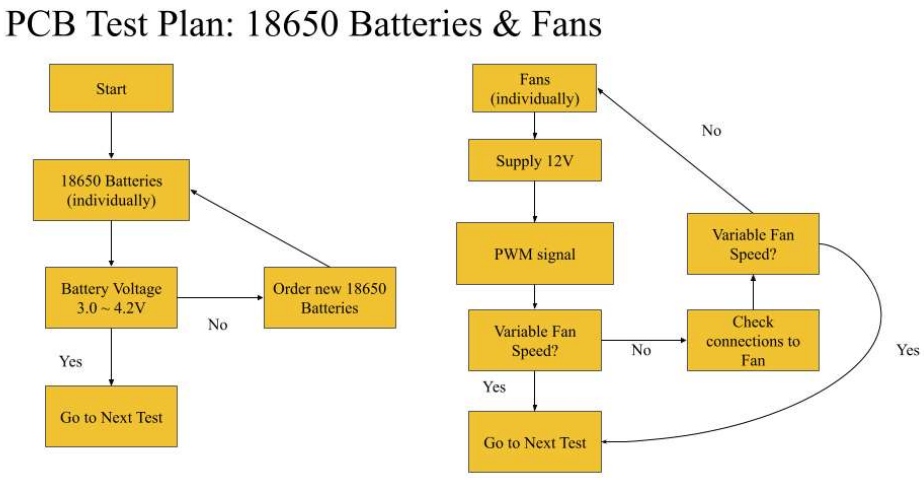


Figure 32. Battery and Fan Test Plan

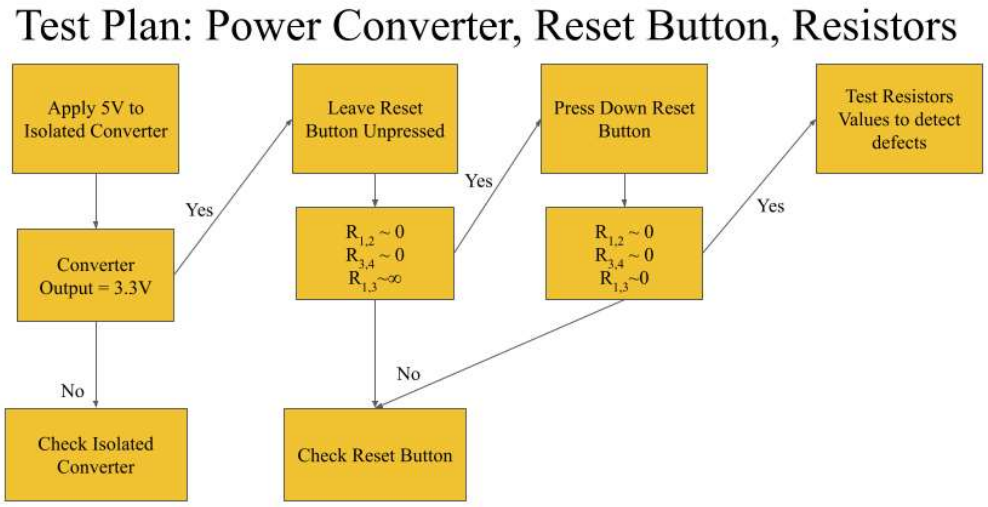


Figure 33. Power Convertor, Reset Button and Resistor Test Plan

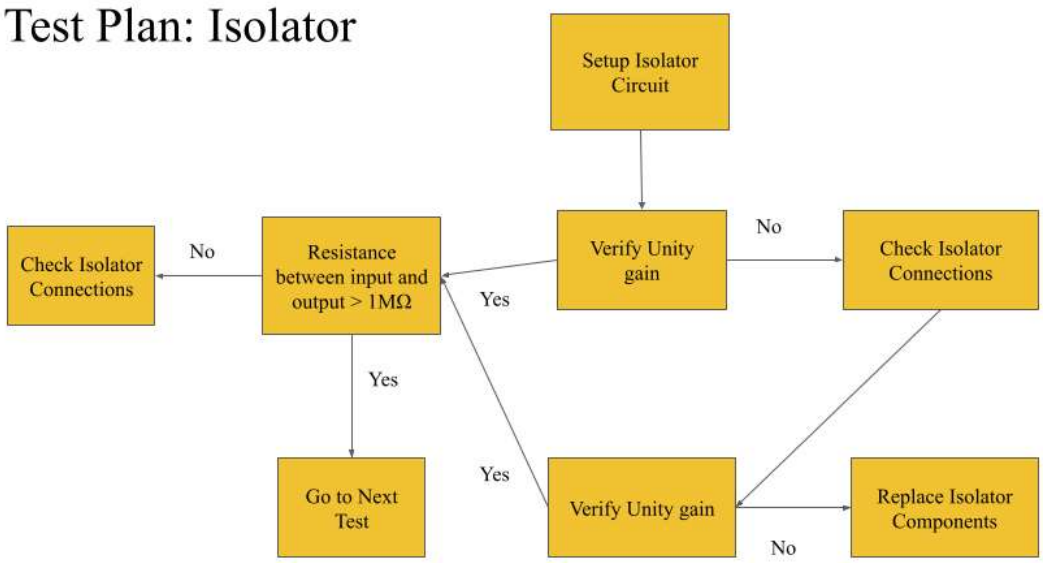


Figure 34. Isolator Test Plan

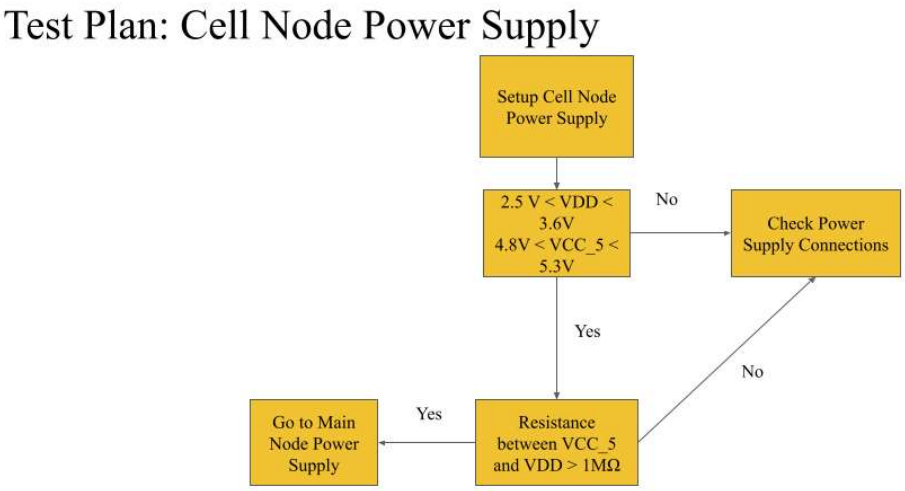


Figure 35. Cell Node Power Supply Test Plan

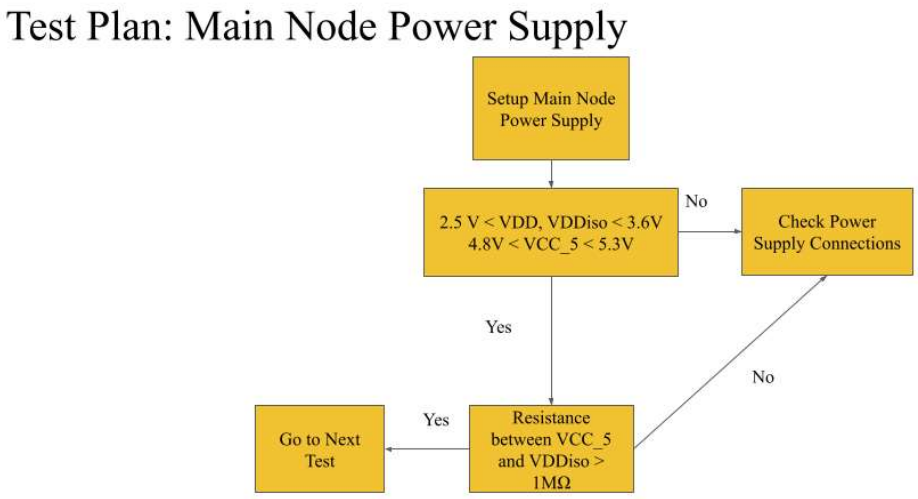


Figure 36. Main Node Power Supply Test Plan

Afterwards, we wrote test programs to test the CAN transceivers and the fans with the test plan for the fans shown in Figure 32. Additionally, test programs for the ADC, and current and temperature sensors were created. These programs were uploaded using PlatformIO to the main and cell boards. The test plans to evaluate that these components functioned correctly are shown in the following figures.

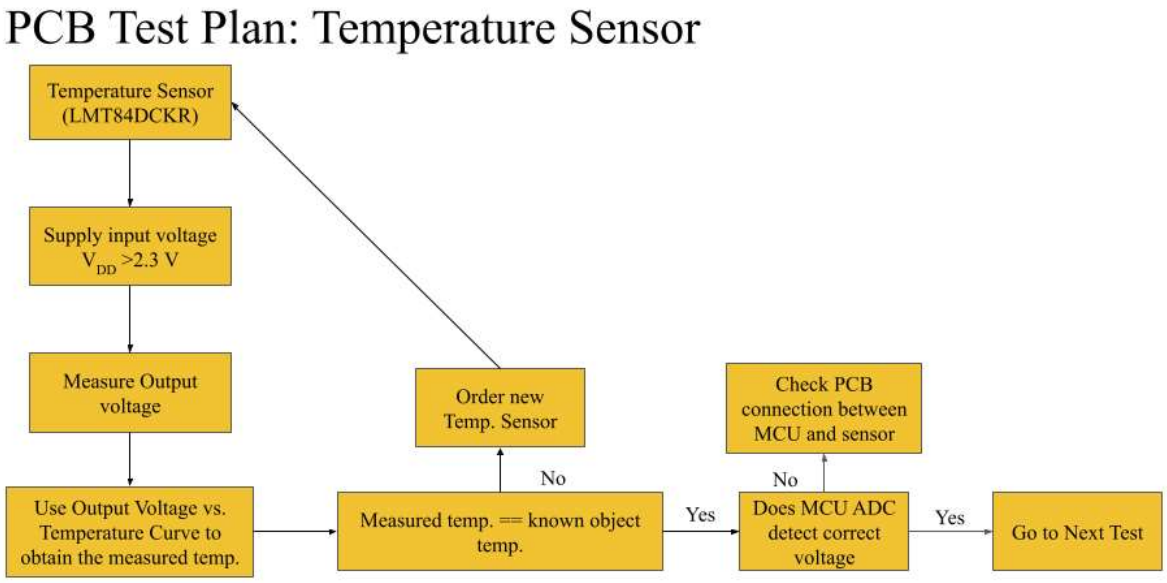


Figure 37. Temperature Sensor Test Plan

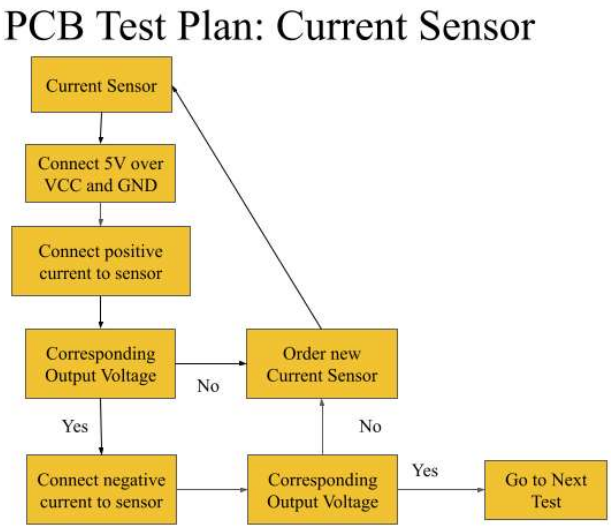


Figure 38. Current Sensor Test Plan

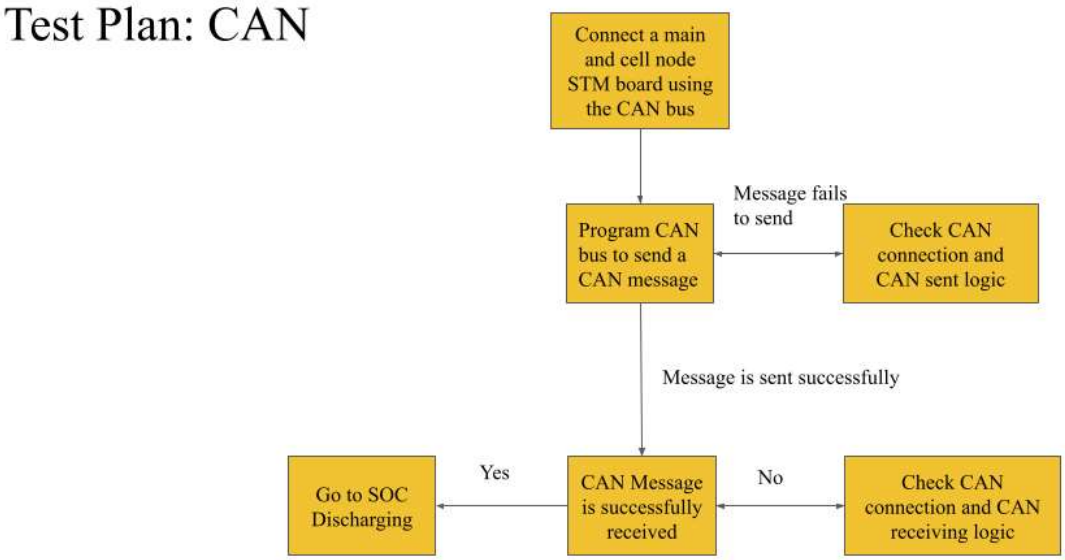


Figure 39. CAN Test Plan

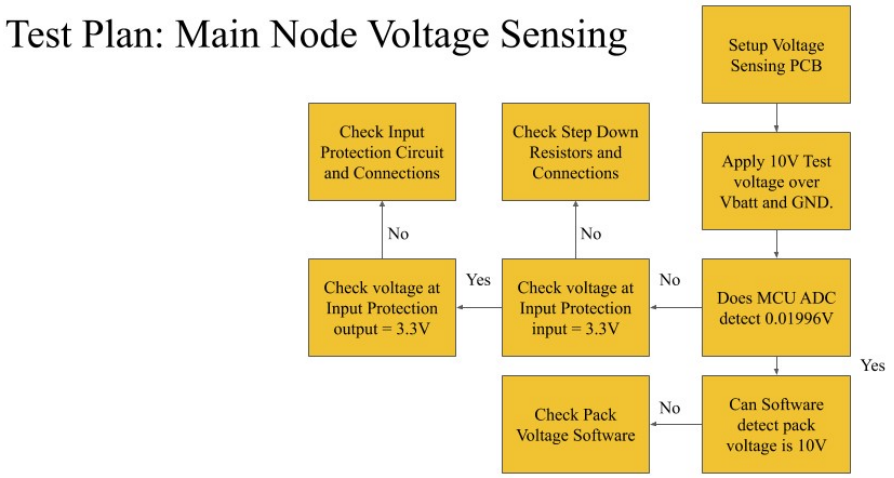


Figure 40. Main Node Voltage Sensing Test Plan

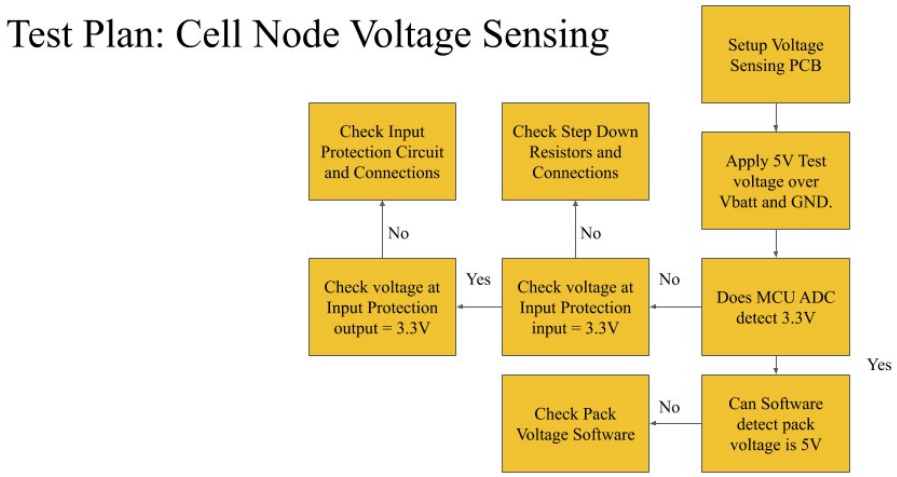


Figure 41. Cell Node Voltage Sensing Test Plan

At this stage of the test plan, there were some unexpected hurdles as we could not upload code to the main board. On the software side, it took a while to set up the correct target files to build and upload files into the main board. On the hardware side, extensive troubleshooting was required to figure out that the problem was the NRST pin of the MCU was shorted to a UART pin on the STLink connector. To resolve this hardware problem, our team had to cut the trace with a knife. After making these changes, our team was able to successfully upload files to the main board. However, cutting the trace also resulted in our team having to use software resets instead of hardware resets.

Once we verified that the hardware for the modular BMS worked correctly, our test plan moved onto testing the software logic that our team wrote. The software that our team wrote involved the following components: calculating SOC and SOH for the batteries, controlling the fans, contactors, and relays. The test plan diagrams for this software are shown in the following figures.

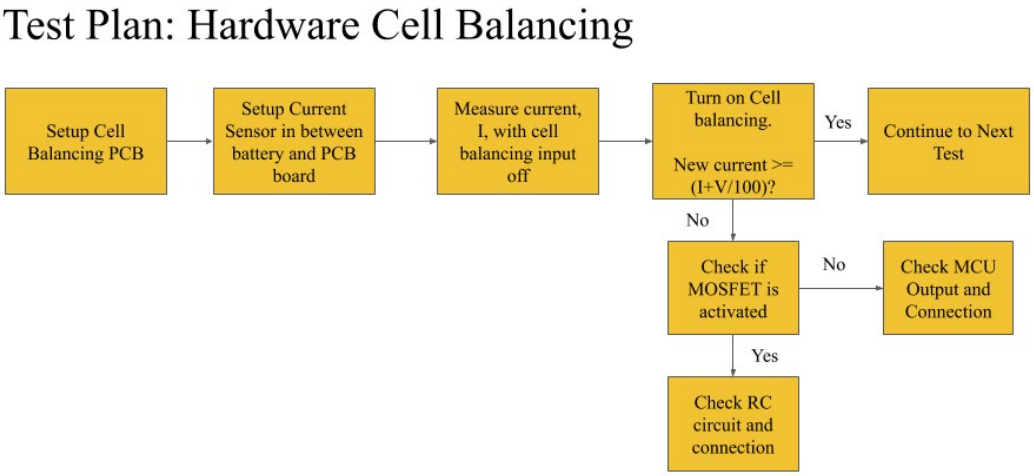


Figure 42. Hardware Cell Balancing Test Plan

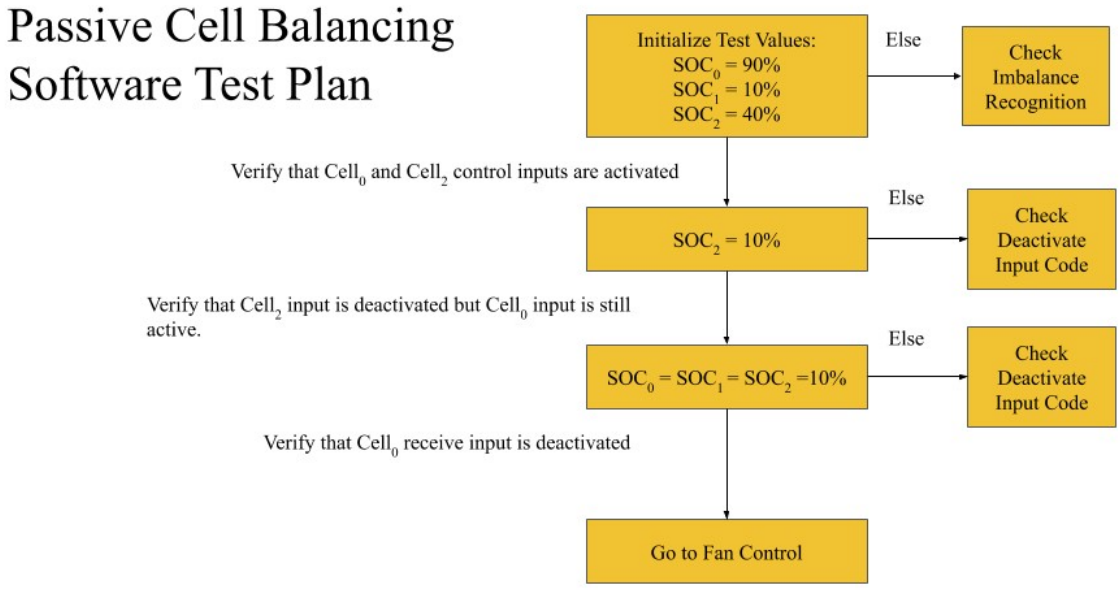


Figure 43. Passive Cell Balancing Software Test Plan

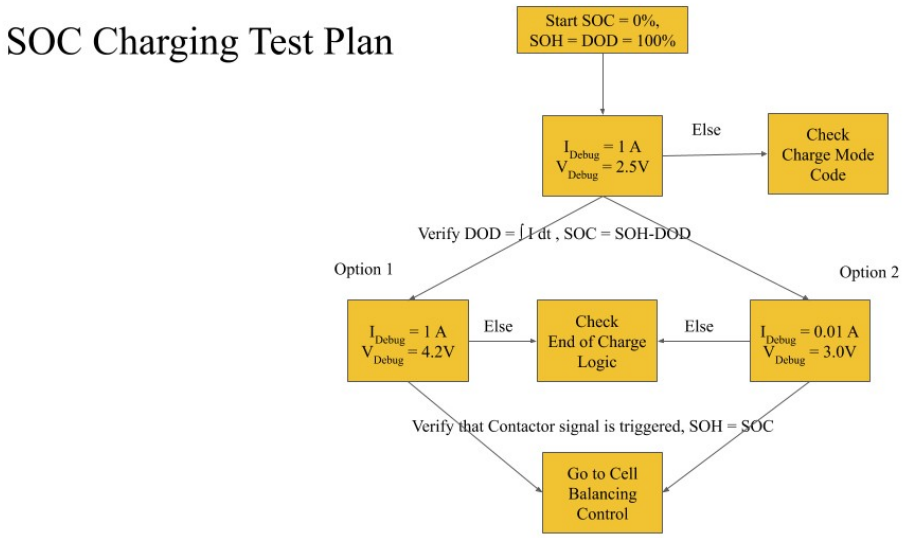


Figure 44. SOC Discharging Test Plan

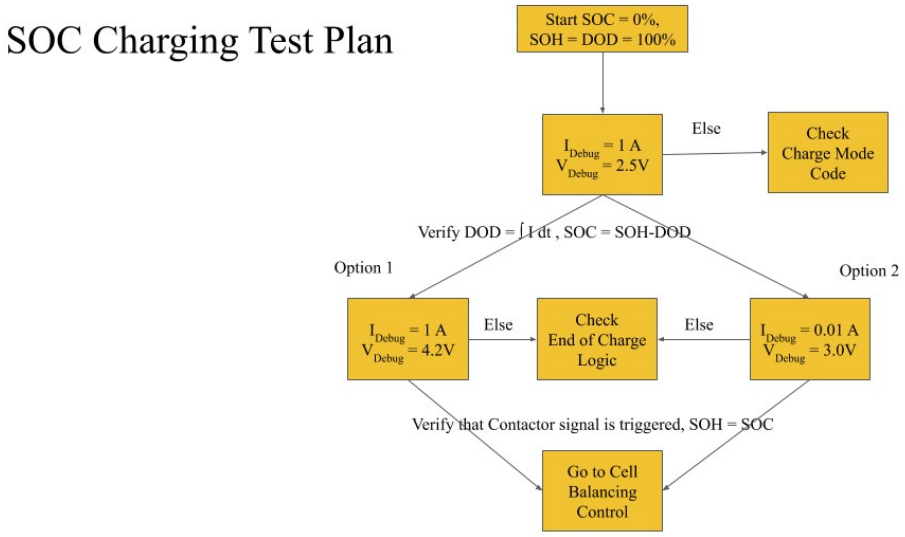


Figure 45. SOC Charging Test Plan

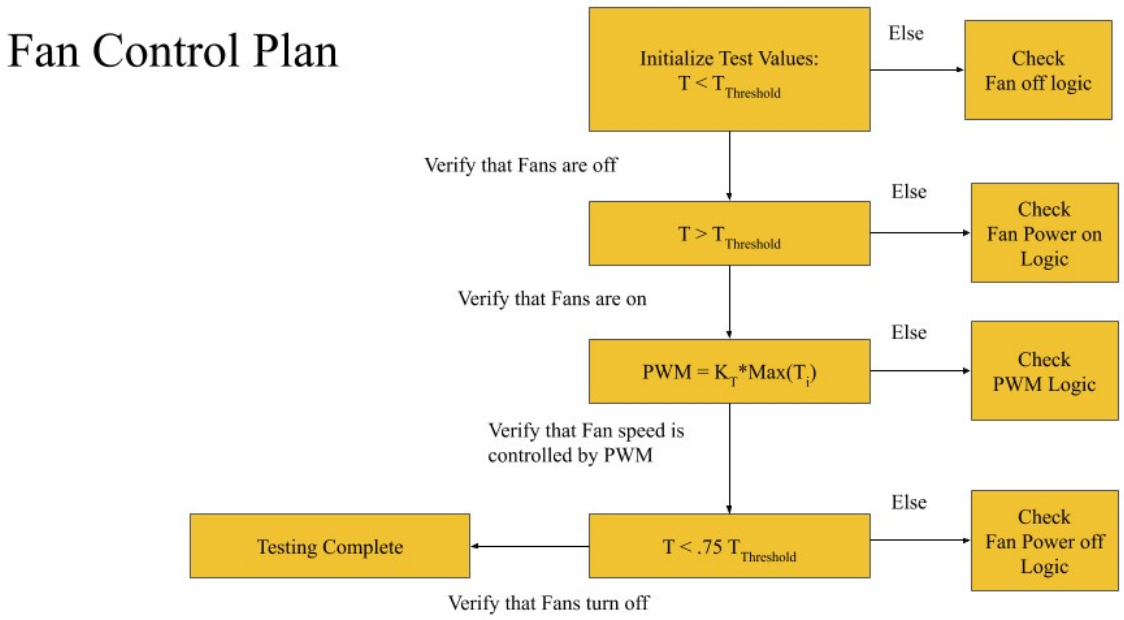


Figure 46. Fan Control Test Plan

After completing the outlined test plan, we knew that both the software and hardware components of the modular BMS function as intended.

Final Results

The team was able to create a fully functioning modular battery management system. The system contains all of the components: a main node, multiple cell nodes, battery cells, a CAN bus, and an external power supply. Our project rubric is shown in Table 1 (it was modified from the original rubric in our Project Proposal to reflect major design changes). First, the external power supply successfully powered the main and cell nodes and proper isolation was achieved. The main node was able to measure pack voltage and current and the cell nodes successfully measured cell voltage and temperature. This data was effectively transmitted via the CAN protocol with no apparent errors. Finally, this data was used to estimate state of charge and state of health as well as control contactors and a fan. Using the rubric in Table 1, we received 3 points for Data Acquisition, 3 points for Data Transmission, 3 points for Data Analysis, and 3 points for Power. The 12 points we received earns us an A for this project as shown in Table 2.

Points	Data Acquisition	Data Transmission	Data Analysis and Control	Power	User Interaction (bonus)
3	Both module voltage and temperature are acquired from multiple module nodes as well as total current and pack voltage are acquired from main node	Module data is transmitted to main node via the CAN protocol with no apparent errors	Current limits and fan output based on temperature, and cell balancing using pack health data are implemented	All nodes able to run from an external power supply. The main node is isolated from the pack ground and the module nodes are isolated from the external supply.	BMS can be programmed by a PC Application, sends data to external listeners, and is enclosed by a physical structure
2	Both module voltage and temperature are acquired from multiple module nodes	Module data is transmitted to main node via the CAN protocol with some errors	Current limits and fan output based on temperature are implemented	All nodes run on the same external power supply. The external supply is not isolated from the pack.	BMS can be programmed by a PC Application and sends data to external listeners
1	Both module voltage and temperature are acquired from a single module node	Module data is transmitted to main node over a wire with many errors	Current limits are implemented	Main node and cell nodes require separate power supplies	BMS sends data to external listeners
0	Module data is not acquired	Module data is not transmitted to main node	Module data is not analyzed	No boards can be powered	System is not usable at all

Table 1. Project Rubric

Points	Grade
13-15	A+
10-12	A
7-9	B
4-6	C
0-3	D

Table 2. Grading Rubric

Costs

Our project's total cost was $445.33, which includes manufacturing the PCBs and all the parts that were needed to populate the Main and Cell Nodes. This cost also included populating the boards by WWW Electronics and purchasing miscellaneous tools such as a crimper and debugging probes. Our cost per board would decrease significantly if our project was manufactured in 10000 unit quantities. The major costs associated with each board were the MCU, the CAN transceivers, and Isolation IC. When purchasing at a bulk quantity such as 10000 units, the costs for these components would decrease by more than 50% on average. Automated equipment can also be used to populate each individual board instead of having WWW Electronics populate the boards, which would lower the costs per a board at the expense of a higher upfront cost to obtain machinery that can populate the boards. Additionally, the cost for the 18650 batteries that our team used would also decrease significantly in bulk quantities with a net savings of ~20% for the batteries. The full list of costs for this project can be found in Appendix B.

Future Work

Future work for this project can involve creating the physical enclosure for the modular BMS as our team did not have enough time to design and build the enclosure ourselves. As the modular BMS our team built was more a proof of concept, a physical enclosure was not critical for the modular BMS to function in a controlled setting. To use this modular BMS in an applied setting such as with an EV, a physical enclosure is critical to ensure that the PCBs and the 18650 batteries are protected from any outside elements. Additionally, the Modular BMS can be further expanded upon by creating mounts for other hardware components such as the fans, which would be important to have before this BMS is able to function in an EV. The modular BMS also

runs off an external power supply, so future work might involve having the BMS run off the 18650 batteries that the BMS is monitoring.

Another task that the team was not able to finish is the PC program to be able to visualize and configure various parameters of the BMS. Currently, all BMS settings, such as current limits, pack capacities, and the number of cells in series and in parallel are all hard-coded constants in the code. However, these parameters often vary greatly between different battery pack configurations, so to ensure proper usability of the BMS, these attributes should be programmable by end users via a PC program. Also, currently, we are only displaying all BMS data on the PC via a serial monitor in the terminal, but this should also be changed to a GUI on the PC program to better visualize the BMS data.

Additional future work for this project can involve using more sophisticated SOC and SOH algorithms to better optimize the charging and discharging of the 18650 batteries. Since our team did not have advanced battery information such as their internal resistance and capacitances, regarding the 18650 batteries that we used, our team could not use the Kalman filter method to calculate SOC. If the internal resistance and capacitances of the batteries are known, the Kalman filter approach is a viable option for a future project to explore. Additionally, our team was not able to calculate and display the SOC error from our chosen SOC estimation method due to not having enough time. The SOC error calculations are shown in Appendix A, and future work could build on these calculations to display the SOC error to allow the user of the modular BMS to be confident on the accuracy of the SOC reading.

Methods to increase the accuracy of the pack voltage and current sensors over a wider range of input voltages and currents can also be researched. One proposed method to do so is to include multiple sensors configured for different ranges and use data from all such sensors to determine an accurate measurement over any of those selected input ranges. Other solutions may also exist, such as measuring the time a capacitor takes to charge and discharge the pack voltage instead of directly reading the analog voltage on the stepped-down pack voltage.

Another area of future work is reducing the power consumption of the modular BMS. Currently, only default sleep modes are utilized to save power during periods of inactivity, but more power savings can be achieved at the cost of response time if deep sleep modes on the microcontroller were utilized. Additionally, certain peripherals, such as the CAN transceivers, also support standby modes that could be used to further increase power savings. Other methods of minimizing power consumption should also be explored, such as by reducing the microcontroller system clock frequency.

References

[1] D. Power, “Cylindrical Batteries More Suitable For Electric Vehicles,” *Lithium ion Battery Manufacturer and Supplier in China-DNK Power*, Jun. 09, 2017. https://www.dnkpower.com/cylindrical-batteries-suitable-electric-vehicles/ (accessed Sep. 13, 2020).

[2] J. Xu and B. Cao, “Battery Management System for Electric Drive Vehicles – Modeling, State Estimation and Balancing,” *New Applications of Electric Drives*, Dec. 2015, doi: 10.5772/61609.

[3] “(PDF) Batteries and battery management systems for electric vehicles,” *ResearchGate*. https://www.researchgate.net/publication/229533419_Batteries_and_battery_management_systems_for_electric_vehicles (accessed Sep. 08, 2020).

[4] “Introduction to battery-management systems,” *Coursera*. https://www.coursera.org/learn/battery-management-systems (accessed Sep. 08, 2020).

[5] “A Li-ion Battery Management System Based on MCU and OZ8920,” *Procedia Engineering*, vol. 29, pp. 738–743, Jan. 2012, doi: 10.1016/j.proeng.2012.01.033.

[6] “Orion Li-Ion Battery Management System | Affordable & Reliable EV Li-Ion BMS.” http://www.orionbms.com/ (accessed Sep. 13, 2020).

[7] “STM32G473xB STM32G473xCS STM32G473xE.” STMicroelectronics, Oct. 2020, [Online]. Available: https://www.st.com/resource/en/datasheet/stm32g473ce.pdf.

[8] “STM32F042x4 STM32F042x6.” STMicroelectronics, Jan. 2017, [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f042k6.pdf.

[9] “2 Layer & 4 Layer PCB Prototyping | Advanced Circuits.” https://www.4pcb.com/pcb-prototype-2-4-layer-boards-specials.html (accessed Dec. 10, 2020).

[10] “Minimum Requirements for PCB Assembly Documentation.” WWW Electronics, Inc.

[11] S. Corrigan, “Introduction to the Controller Area Network (CAN),” Texas Instruments, SLOA101B, May 2016. [Online]. Available: https://www.ti.com/lit/an/sloa101b/sloa101b.pdf?ts=1607186769642.

[12] “IPC-2221A: Generic Standard on Printed Board Design.” IPC: Association Connecting Electronics Industries, May 2003, Accessed: Dec. 09, 2020. [Online]. Available: https://www.ipc.org/TOC/IPC-2221A.pdf.

[13] “SMD Packages: Sizes Dimensions Details » Electronics Notes.” https://www.electronics-notes.com/articles/electronic_components/surface-mount-technology-smd-smt/packages.php (accessed Dec. 10, 2020).

[14] “USB Standards: USB 1, USB 2, USB 3, USB 4 » Electronics Notes.” https://www.electronics-notes.com/articles/connectivity/usb-universal-serial-bus/standards.php (accessed Dec. 10, 2020).

[15] A. Ltd, “CoreSight Architecture | Serial Wire Debug,” Arm Developer. https://developer.arm.com/architectures/cpu-architecture/debug-visibility-and-trace/coresight-architecture/serial-wire-debug (accessed Dec. 10, 2020).

[16] S. C. | D. Electronics | 74, “Basics of UART Communication,” Circuit Basics, Feb. 13, 2016. https://www.circuitbasics.com/basics-uart-communication/ (accessed Dec. 10, 2020).

[17] “✔ RoHS Compliance FAQ.” https://www.rohsguide.com/rohs-faq.htm#:~:text=RoHS%20stands%20for%20Restriction%20of,products%20(known%20as%20EEE). (accessed Dec. 10, 2020).

[18] “KiCad EDA.” https://kicad.org/ (accessed Dec. 10, 2020).

[19] “Build software better, together,” *GitHub*. https://github.com (accessed Sep. 13, 2020).

[20] “CADLAB.io | Visual collaboration and version control platform for your PCB.” https://cadlab.io/ (accessed Sep. 13, 2020).

[21] “Visual Studio Code - Code Editing. Redefined.” https://code.visualstudio.com/ (accessed Sep. 13, 2020).

[22] PlatformIO, “PlatformIO is a professional collaborative platform for embedded development,” *PlatformIO*. https://platformio.org (accessed Sep. 13, 2020).

[23] “Mbed OS | Mbed.” https://os.mbed.com/mbed-os/ (accessed Sep. 13, 2020).

[24] A. Katwala, “The spiralling environmental cost of our lithium battery addiction,” *Wired UK*, Aug. 05, 2018.

[25] “What Do Batteries Do to the Environment If Not Properly Recycled?,” *Sciencing*. https://sciencing.com/what-do-batteries-do-to-the-environment-if-not-properly-recycled-12730824.html (accessed Sep. 15, 2020).

[26] “Plant-Wide Hazards | Other OSHA Requirements and Programs - Electrical Hazards | Occupational Safety and Health Administration.” https://www.osha.gov/SLTC/etools/poultry/general_hazards/elec_hazards.html (accessed Sep. 15, 2020).

[27] A. Chen, “Electric vehicles mean first responders have to deal with battery fires,” The Verge, Jul. 03, 2018. https://www.theverge.com/2018/7/3/17530646/tesla-battery-fire-electric-vehicles-transportation-science (accessed Dec. 10, 2020).

[28] A. Katwala, “The spiralling environmental cost of our lithium battery addiction,” Wired UK, Aug. 05, 2018.

[29] I. Loncarevic, “Battery management system,” US20190379214A1.

[30] Hak Hon Chau, “Fault tolerant modular battery management system,” US8410755B2.

[31] Joseph Mario Ambrosio and K. Sfakianos, “Vehicle charging, monitoring and control systems for electric and hybrid electric vehicles,” US7830117B2.

[32] M. Murnane and A. Ghazel , “A Closer Look at State of Charge (SOC) and State of Health (SOH) Estimation Techniques for Batteries,” Analog Devices, 2017.

[33] A. Raj , “Cell Balancing Techniques and How to Use Them,” Circuit Digest, Feb. 23, 2019. https://circuitdigest.com/article/cell-balancing-techniques-and-how-to-use-them (accessed Dec. 10, 2020).

[34] “3.8V TO 32V INPUT, 2A LOW IQ SYNCHRONOUS BUCK WITH ENHANCED EMI REDUCTION.” Diodes, Inc., Jan. 2019, [Online]. Available: https://www.diodes.com/assets/Datasheets/AP63200-AP63201-AP63203-AP63205.pdf.

[35] “1A LOW NOISE CMOS LDO REGULATOR WITH ENABLE AP2114 .” BCD Semiconductor Manufacturing Limited, Jan. 2013, [Online]. Available: https://www.diodes.com/assets/Datasheets/AP2114.pdf.

[36] “SF IE Series.” XP Power, Feb. 04, 2013, [Online]. Available: https://www.xppower.com/portals/0/pdfs/SF_IE.pdf.

[37] “STM32G473xB STM32G473xC STM32G473xE.” ST, Accessed: Dec. 10, 2020. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32g473ce.pdf.

[38] “NUCLEO64 STM32G4.” ST, Apr. 18, 2019, Accessed: Dec. 10, 2020. [Online]. Available: https://www.st.com/content/ccc/resource/technical/layouts_and_diagrams/schematic_pack/group1/69/2a/24/ce/d1/1b/45/21/mb1367-g474re-c04_schematic/files/mb1367-g474re-c04_schematic.pdf/jcr:content/translations/en.mb1367-g474re-c04_schematic.pdf.

[39] “UM2502 User manual.” Apr. 12, 2019, Accessed: Dec. 10, 2020. [Online]. Available: https://www.st.com/resource/en/user_manual/dm00555046-stlinkv3mods-and-stlinkv3mini-mini-debuggersprogrammers-for-stm32-stmicroelectronics.pdf.

[40] “ECS-2520MV.” ECS Inc., 2020, Accessed: Dec. 10, 2020. [Online]. Available: https://ecsxtal.com/store/pdf/ECS-2520MV.pdf.

[41] “ECS-327MVATX.” ECS Inc., 2019, Accessed: Dec. 10, 2020. [Online]. Available: https://componentsearchengine.com/Datasheets/1/ECS-327MVATX-1-CN-TR.pdf.

[42] “Hall Effect Current Sensors L31S***S05FS Series .” Tamura Corporation, Feb. 2014, Accessed: Dec. 10, 2020. [Online]. Available: https://www.mouser.com/datasheet/2/397/L31SXXXS05FS-267665.pdf.

[43] Solutions Cubed, “Protecting Inputs in Digital Electronics,” DigiKey Electronics, Apr. 11, 2012. https://www.digikey.com/en/articles/protecting-inputs-in-digital-electronics (accessed Dec. 10, 2020).

[44] “MCP6001/1R/1U/2/4.” Microchip, 2020, Accessed: Dec. 10, 2020. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/MCP6001-1R-1U-2-4-1-MHz-Low-Power-Op-Amp-DS20001733L.pdf.

[45] “MCP2561/2.” Microchip Technology Inc., 2013, Accessed: Dec. 10, 2020. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/25167A.pdf.

[46] “AMC1311x High-Impedance, 2-V Input, Reinforced Isolated Amplifiers.” Texas Instruments, May 2020, Accessed: Dec. 10, 2020. [Online]. Available: https://www.ti.com/lit/ds/symlink/amc1311.pdf?ts=1607629963621&ref_url=https%253A%252F%252Fwww.google.com%252F.

[47] “STM32F042x4 STM32F042x6.” ST, Jan. 2017, Accessed: Dec. 10, 2020. [Online]. Available: https://www.st.com/content/ccc/resource/technical/document/datasheet/52/ad/d0/80/e6/be/40/ad/DM00105814.pdf/files/DM00105814.pdf/jcr:content/translations/en.DM00105814.pdf.

[48] “LMT84 1.5-V, SC70/TO-92/TO-92S, Analog Temperature Sensors.” Texas Instruments, Oct. 2017, [Online]. Available: https://www.ti.com/lit/ds/symlink/lmt84.pdf?HQS=TI-null-null-digikeymode-df-pf-null-wwe&ts=1602963019481.

[49] “ISO1042-Q1 Automotive Isolated CAN Transceiver With 70-V Bus Fault Protection and Flexible Data Rate.” Texas Instruments, Oct. 2020, Accessed: Dec. 10, 2020. [Online]. Available: https://www.ti.com/lit/ds/symlink/iso1042-q1.pdf.

[50] “Sanyo NCR18650GA 3450mAh 10A Battery,” *IMR Batteries*. https://www.imrbatteries.com/sanyo-ncr18650ga-3450mah-10a-battery/ (accessed Dec. 10, 2020).

[51] “Specifications for NCR18650GA.” Panasonic, [Online]. Available: https://www.imrbatteries.com/content/sanyo_ncr18650ga.pdf.

[52] “The WebSocket Protocol.” IETF, Dec. 2011, Accessed: Dec. 10, 2020. [Online]. Available: https://tools.ietf.org/html/rfc6455.

Appendix

Appendix A: SOC Calculations Error

Using the Combined Coulomb Counting and Voltage-based method for SOC Estimation, there are some errors that can impact the SOC estimate from using this method. First, the battery temperature does not remain constant during normal operating conditions. This means that there would be a marginal error resulting from using the battery voltage vs. discharge capacity characteristic in the datasheet as these characteristics were based on a constant operating temperature. Additionally, the accuracy of the output voltage from the Hall Effect current sensor will also play a large role in the error of the SOC that is calculated. The accuracy of the Hall Effect current sensor is also affected by the temperature that the current sensor operates in. The following analysis shows the expected error in using our chosen Hall Effect current sensor and in using the chosen algorithm for SOC estimation.

A Hall Effect current sensor takes an input current, and this results in an output voltage that is proportional to the input current. The relationship between the input current and the output voltage is shown in the below Saturation Characteristic that the Hall Effect current sensor manufacturer provided.

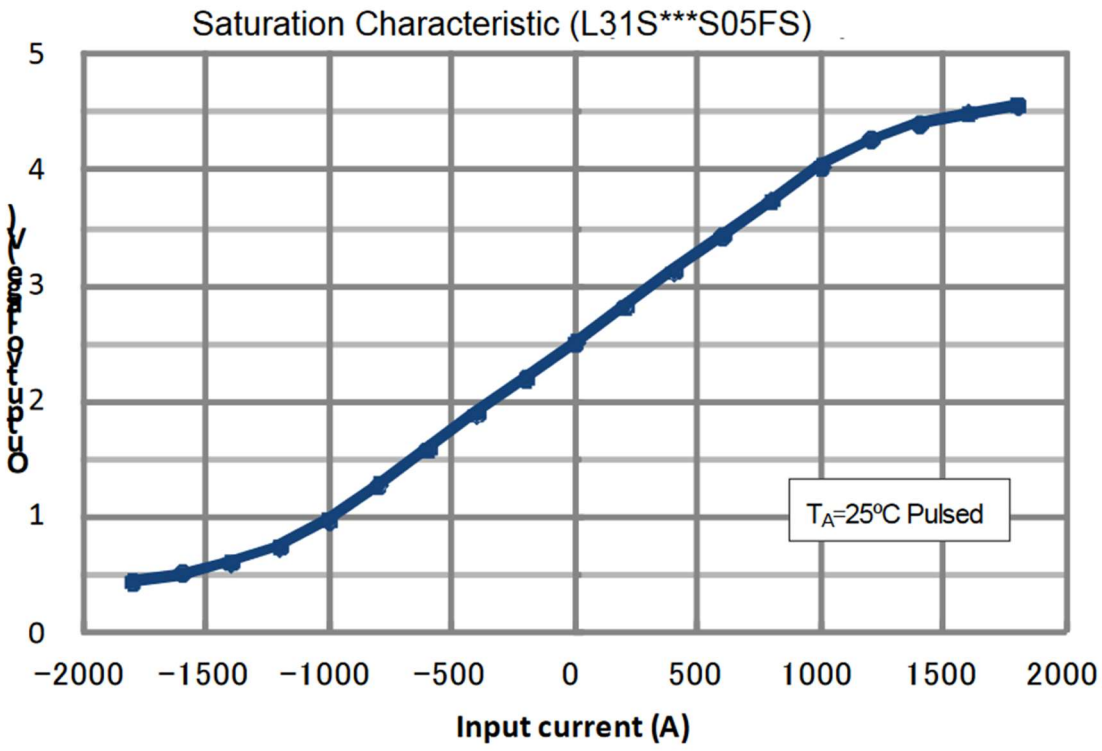


Figure 47. Saturation Characteristic of the Hall Effect Sensor [42]

From the datasheet, the reference voltage, V_{REF} is given as the following:

$$V_{REF} = 2.5V \pm 0.020V$$

The offset voltage is given as the following equation from the data sheet, which results in the following error measurements:

$$V_{of} = V_{REF} \pm .025V \text{ (at } I_f = 0A)$$

$$V_{of} = 2.5V \pm \text{sqrt}(.025^2 + .02^2)\ V \text{ (at } I_f = 0A)$$

$$V_{of} = 2.5V \pm \text{sqrt}(.025^2 + .02^2)\ V \text{ (at } I_f = 0A)$$

$$V_{of} = 2.5V \pm .032V \text{ (at } I_f = 0A)$$

Since the Hall Effect current sensor is rated for a primary nominal current (I_f) of 300A, the following is the rated output voltage for the sensor at $I_f = 300A$:

$$V_o = V_{of} + .625V \pm .015V \text{ (at } I_f = 300A)$$

$$V_o = 2.5V + .625V \pm \text{sqrt}(.025^2 + .02^2 + .015^2)\ V \text{ (at } I_f = 300A)$$

$$V_o = 3.125\ V \pm .035V \text{ (at } I_f = 300A)$$

Since there is a linear relationship from $I_f = 0$ A to $I_f = 300A$, the measured voltage from the Hall Effect current sensor can be used to determine the input current value using this linear relationship with an error of $\pm$.035V. This is an error of ~1.12% and this method will also apply for $I_f = -300$ A to $I_f = 0$ A.

We will now analyze the effect of temperature on the voltage reading from the Hall Effect current sensor. From the datasheet, the thermal drift of offset (TcVof) is shown below:

$$\text{TcVof} \leq \pm .3\ \text{mV/}^o\text{C} \text{ (at } I_f = 0A)$$

Since the BMS will be operating in a temperature range of 40 ^{o}C (25^{o}C - 65 ^{o}C), the maximum thermal drift of offset is given as the following:

$$\text{TcVof} \leq \pm .3\ \text{mV/}^o\text{C} * 40^o\text{C} \text{ (at } I_f = 0A)$$

$$\text{TcVof} \leq \pm 12\ \text{mV} = .012V \text{ (at } I_f = 0A)$$

The total error from the current sensor will then be the following:

$$V_o = 3.125\ V \pm \text{sqrt}(.035^2 + .012^2)V \text{ (at } I_f = 300A)$$

$$V_o = 3.125\ V \pm .037V \text{ (at } I_f = 300A)$$

This results in a total error of .037V or ~1.18% from using the voltage reading of the Hall Effect current sensor. Using the error that was found from the Hall Effect current sensor, the error from using the Coulomb Counting method can be determined.

The method that was chosen to determine the SOC of the battery would reduce the accumulated errors from using the Coulomb counting method as the Coulomb counting method is relatively accurate in the short term. The periodic recalibrations using the OCV would ensure that the error from the Coulomb counting method would not get too significant.

Appendix B: Full List of Costs for the Modular BMS

Item Name	Vendor	Vendor Part Number	Manufacturer Part #	Quanti ty	Unit Cost	Total Cost	Remaining Budget
Original Budget							**$500.00**
PCB Prints	**3W**			**2**	**$33.00**	**$66.00**	**$434.00**
CAP CER 0.1UF 50V X7R RADIAL	**Digikey**	**478-7335-1-ND**	**SR215C104KAATR2**	**19**	**$0.14**	**$2.57**	**$431.43**
CAP ALUM 1UF 20% 100V RADIAL	**Digikey**	**493-12811-1-ND**	**UVR2A010MDD1TD**	**6**	**$0.22**	**$1.32**	**$430.11**
CAP ALUM 4.7UF 20% 50V RADIAL	**Digikey**	**P19528CT-ND**	**ECA-1HHG4R7I**	**4**	**$0.22**	**$0.88**	**$429.23**
CAP ALUM 10UF 20% 63V THRUHOLE	**Digikey**	**399-18269-1-ND**	**ESK106M063AC3FA**	**2**	**$0.18**	**$0.36**	**$428.87**
CAP ALUM 22UF 20% 25V RADIAL	**Digikey**	**P19523CT-ND**	**ECA-1EM220I**	**2**	**$0.19**	**$0.38**	**$428.49**

DIODE SCHOTTKY 30V 800MA USC	**Digikey**	**CUS08F30H3FCT-ND**	**CUS08F30,H3F**	**3**	**$0.31**	**$0.93**	**$427.56**
TVS DIODE 3.3V 9V 2DFN	**Digikey**	**F10154CT-ND**	**SP1103C-01UTG**	**3**	**$0.35**	**$1.05**	**$426.51**
MAINSTREAM ARM CORTEX-M4 CORE WI	**Digikey**	**497-STM32G473CET6-ND**	**STM32G473CET6**	**1**	**$8.43**	**$8.43**	**$418.08**
IC REG BUCK 5V 2A TSOT26	**Digikey**	**AP63205WU-7DICT-ND**	**AP63205WU-7**	**2**	**$0.83**	**$1.66**	**$416.42**
IC REG LINEAR 3.3V 1A SOT223	**Digikey**	**AP2114H-3.3TRG1DICT-ND**	**AP2114H-3.3TRG1**	**1**	**$0.35**	**$0.35**	**$416.07**
IC ISOLATION 8SOIC	**Digikey**	**296-49549-1-ND**	**AMC1311BDWVR**	**1**	**$6.68**	**$6.68**	**$409.39**
CONN HEADER VERT 2POS 2.54MM	**Digikey**	**S1012EC-02-ND**	**PREC002SAAN-RC**	**3**	**$0.06**	**$0.18**	**$409.21**
TERM BLOCK HDR 4POS 90DEG 5MM	**Digikey**	**277-1086-ND**	**1754478**	**1**	**$1.45**	**$1.45**	**$407.76**
CONN HEADER VERT 4POS 3.96MM	**Digikey**	**455-1641-ND**	**B4P-VH(LF)(SN)**	**4**	**$0.29**	**$1.16**	**$406.60**
TERM BLOCK HDR 2POS 90DEG 5MM	**Digikey**	**277-1084-ND**	**1754436**	**4**	**$0.72**	**$2.88**	**$403.72**
TERM BLOCK HDR 2POS 90DEG 7.62MM	**Digikey**	**277-6070-ND**	**1720466**	**1**	**$3.07**	**$3.07**	**$400.65**

CONN HEADER VERT 14POS 1.27MM	**Digikey**	**1175-1630-ND**	**3220-14-0100-00**	**2**	**$0.70**	**$1.40**	**$399.25**
FERRITE BEAD 600 OHM 0805 1LN	**Digikey**	**MH2029-601YCT-ND**	**MH2029-601Y**	**2**	**$0.10**	**$0.20**	**$399.05**
FIXED IND 4.7UH 3A 40.3 MOHM SMD	**Digikey**	**587-2100-1-ND**	**NR6028T4R7M**	**2**	**$0.36**	**$0.72**	**$398.33**
DC DC CONVERTER 3.3V 1W	**Digikey**	**1470-2396-5-ND**	**IE0503S**	**2**	**$4.15**	**$8.30**	**$390.03**
MOSFET N-CH 60V 12A IPAK	**Digikey**	**NTD3055L104-1GOS-ND**	**NTD3055L104-1G**	**4**	**$0.62**	**$2.48**	**$387.55**
RES 120 OHM 1/2W 5% CF MINI	**Digikey**	**S120HCT-ND**	**CFM12JT120R**	**2**	**$0.10**	**$0.20**	**$387.35**
RES 10K OHM 1/4W 1% AXIAL	**Digikey**	**S10KCACT-ND**	**RNMF14FTC10K0**	**3**	**$0.10**	**$0.30**	**$387.05**
RES 150K OHM 1/4W 1% AXIAL	**Digikey**	**RNF14FTD150KCT-ND**	**RNF14FTD150K**	**2**	**$0.10**	**$0.20**	**$386.85**
RES 180K OHM 1/4W 1% AXIAL	**Digikey**	**S180KCACT-ND**	**RNMF14FTC180K**	**1**	**$0.10**	**$0.10**	**$386.75**
RES 68K OHM 1/4W 1% AXIAL	**Digikey**	**S68KCACT-ND**	**RNMF14FTC68K0**	**2**	**$0.10**	**$0.20**	**$386.55**
RES ARRAY 8 RES 100K OHM 1206	**Digikey**	**CAY17-104JALFCT-ND**	**CAY17-104JALF**	**2**	**$0.13**	**$0.26**	**$386.29**
TACT 4.5 X 4.5, 3.8 MM H, 2.5N,	**Digikey**	**PTS647SK38SMTR2LFSCT-ND**	**PTS 647 SK38 SMTR2 LFS**	**2**	**$0.13**	**$0.26**	**$386.03**

IC TRANSCEIVER HALF 1/1 8DIP	**Digikey**	**MCP2562-E/P-ND**	**MCP2562-E/P**	**2**	**$0.93**	**$1.86**	**$384.17**
IC OPAMP GP 2 CIRCUIT 8DIP	**Digikey**	**MCP6002-I/P-ND**	**MCP6002-I/P**	**3**	**$0.33**	**$0.99**	**$383.18**
OSC XO 8.000MHZ CMOS SMD	**Digikey**	**XC3029CT-ND**	**ECS-2520MV-080-CN-TR**	**2**	**$0.96**	**$1.92**	**$381.26**
XTAL OSC XO 32.768KHZ SMD	**Digikey**	**XC3153CT-ND**	**ECS-327MVATX-1-CN-TR**	**1**	**$1.12**	**$1.12**	**$380.14**
CAP TANT 4.7UF 10% 16V 1206	**Digikey**	**478-11461-1-ND**	**TAJA475K016UNJ**	**3**	**$0.29**	**$0.87**	**$379.27**
CAP TANT 10UF 10% 16V 1206	**Digikey**	**399-8269-1-ND**	**T491A106K016AT**	**2**	**$0.37**	**$0.74**	**$378.53**
CAP TANT 22UF 20% 16V 1206	**Digikey**	**718-2393-1-ND**	**TMCMA1C226MTRF**	**2**	**$0.49**	**$0.98**	**$377.55**
CAP CER 10000PF 50V X7R 0805	**Digikey**	**311-1136-1-ND**	**CC0805KRX7R9BB103**	**1**	**$0.10**	**$0.10**	**$377.45**
CAP TANT 1UF 20% 20V 1206	**Digikey**	**478-5748-1-ND**	**TAJA105M020RNJ**	**1**	**$0.29**	**$0.29**	**$377.16**
CAP CER 0.47UF 50V X7R 0805	**Digikey**	**1276-6482-1-ND**	**CL21B474KBFNNNG**	**1**	**$0.10**	**$0.10**	**$377.06**
SENSOR ANALOG -50C-150C SC70-5	**Digikey**	**296-38163-1-ND**	**LMT84DCKR**	**1**	**$0.67**	**$0.67**	**$376.39**
IC OPAMP GP 2 CIRCUIT 8MSOP	**Digikey**	**MCP6L02T-E/MSCT-ND**	**MCP6L02T-E/MS**	**1**	**$0.31**	**$0.31**	**$376.08**

MOSFET N-CH 30V 500MA SOT-23	**Digikey**	**NTR4003NT1GOSCT-ND**	**NTR4003NT1G**	**1**	**$0.25**	**$0.25**	**$375.83**
RES 10K OHM 1% 1/8W 0805	**Digikey**	**RMCF0805FT10K0CT-ND**	**RMCF0805FT10K0**	**1**	**$0.10**	**$0.10**	**$375.73**
RES 180K OHM 1% 1/8W 0805	**Digikey**	**738-RMCF0805FT180KCT-ND**	**RMCF0805FT180K**	**1**	**$0.10**	**$0.10**	**$375.63**
RES 330K OHM 1% 1/8W 0805	**Digikey**	**RMCF0805FT330KCT-ND**	**RMCF0805FT330K**	**2**	**$0.10**	**$0.20**	**$375.43**
RES 100K OHM 1% 1/8W 0805	**Digikey**	**RMCF0805FT100KCT-ND**	**RMCF0805FT100K**	**2**	**$0.10**	**$0.20**	**$375.23**
RES SMD 120 OHM 5% 1/2W 0805	**Digikey**	**P120ADCT-ND**	**ERJ-P06J121V**	**1**	**$0.10**	**$0.10**	**$375.13**
RES SMD 100 OHM 5% 1/2W 0805	**Digikey**	**P100ADCT-ND**	**ERJ-P06J101V**	**1**	**$0.10**	**$0.10**	**$375.03**
RES ARRAY 4 RES 100K OHM 0804	**Digikey**	**YC124J-100KCT-ND**	**YC124-JR-07100KL**	**1**	**$0.10**	**$0.10**	**$374.93**
IC MCU 32BIT 32KB FLASH 20TSSOP	**Digikey**	**497-17344-ND**	**STM32F042F6P6**	**1**	**$2.79**	**$2.79**	**$372.14**
AUTO ISO CAN TRANSCEIVER 5KVRMS	**Digikey**	**296-52277-1-ND**	**ISO1042BQDWRQ1**	**1**	**$4.85**	**$4.85**	**$367.29**
CONN HOUSING VH 4POS 3.96MM WHT	**Digikey**	**455-1185-ND**	**VHR-4N**	**6**	**$0.18**	**$1.08**	**$366.21**

TERM BLOCK PLUG 2POS STR 7.62MM	**Digikey**	**277-2434-ND**	**1777723**	**1**	**$4.03**	**$4.03**	**$362.18**
TERM BLOCK PLUG 2POS STR 5MM	**Digikey**	**277-1000-ND**	**1754449**	**4**	**$1.99**	**$7.96**	**$354.22**
TERM BLOCK PLUG 4POS STR 5MM	**Digikey**	**277-1002-ND**	**1754481**	**1**	**$3.96**	**$3.96**	**$350.26**
RES 100K OHM 1/4W 1% AXIAL	**Digikey**	**S100KCACT-ND**	**RNMF14FTC100K**	**10**	**$0.07**	**$0.65**	**$349.61**
RES 10M OHM 1/8W 5% CF AXIAL	**Digikey**	**CF18JT10M0CT-ND**	**CF18JT10M0**	**10**	**$0.05**	**$0.45**	**$349.16**
PC TEST POINT MULTIPURPOSE BLACK	**Digikey**	**36-5011-ND**	**5011**	**25**	**$0.32**	**$8.03**	**$341.13**
CAP CER 0.1UF 50V X7R 0805	**Digikey**	**478-1395-1-ND**	**08055C104KAT2A**	**10**	**$0.07**	**$0.73**	**$340.40**
CONN SOCKET 16-20AWG CRIMP TIN	**Digikey**	**455-1319-1-ND**	**SVH-41T-P1.1**	**25**	**$0.08**	**$1.97**	**$338.43**
Industrial Current Sensors CURRENT SENSOR (300A;+5V;VREFOUT)	**Mouser**	**838-L31S300S05FS**	**L31S300S05FS**	**1**	**$17.52**	**$17.52**	**$320.91**
Hardware Debuggers 16/32-BITS MICROS	**Mouser**	**511-STLINK-V3MINI**	**STLINK-V3MINI**	**1**	**$10.16**	**$10.16**	**$310.75**
Cell Board Population	**3W**			**1**	**$21.00**	**$21.00**	**$289.75**

Main Board Population	**3W**			**1**	**$11.00**	**$11.00**	**$278.75**
CAP TANT 4.7UF 10% 16V 1206	**Digikey**	**478-11461-1-ND**	**TAJA475K016UNJ**	**6**	**$0.29**	**$1.74**	**$277.01**
CAP TANT 10UF 20% 16V 1206	**Digikey**	**399-3687-1-ND**	**T491A106M016AT**	**4**	**$0.39**	**$1.56**	**$275.45**
CAP CER 0.1UF 50V X7R 0805	**Digikey**	**478-1395-1-ND**	**08055C104KAT2A**	**18**	**$0.07**	**$1.31**	**$274.14**
CAP TANT 22UF 10% 10V 1206	**Digikey**	**478-11223-1-ND**	**TAJA226K010TNJ**	**4**	**$0.31**	**$1.24**	**$272.90**
CAP CER 10000PF 50V X7R 0805	**Digikey**	**311-1136-1-ND**	**CC0805KRX7R9BB103**	**2**	**$0.10**	**$0.20**	**$272.70**
CAP CER 0.47UF 50V X7R 0805	**Digikey**	**1276-6482-1-ND**	**CL21B474KBFNNNG**	**2**	**$0.10**	**$0.20**	**$272.50**
CAP TANT 1UF 20% 20V 1206	**Digikey**	**478-5748-1-ND**	**TAJA105M020RNJ**	**2**	**$0.29**	**$0.58**	**$271.92**
DIODE SCHOTTKY 30V 800MA USC	**Digikey**	**CUS08F30H3FCT-ND**	**CUS08F30,H3F**	**2**	**$0.32**	**$0.64**	**$271.28**
TVS DIODE 3.3V 9V 2DFN	**Digikey**	**F10154CT-ND**	**SP1103C-01UTG**	**2**	**$0.35**	**$0.70**	**$270.58**
IC REG BUCK 5V 2A TSOT26	**Digikey**	**AP63205WU-7DICT-ND**	**AP63205WU-7**	**2**	**$0.83**	**$1.66**	**$268.92**
SENSOR ANALOG - 50C-150C SC70-5	**Digikey**	**296-38163-1-ND**	**LMT84DCKR**	**2**	**$0.67**	**$1.34**	**$267.58**

IC OPAMP GP 2 CIRCUIT 8MSOP	**Digikey**	**MCP6L02T-E/MSCT-ND**	**MCP6L02T-E/MS**	**2**	**$0.31**	**$0.62**	**$266.96**
CONN HEADER VERT 4POS 3.96MM	**Digikey**	**455-1641-ND**	**B4P-VH(LF)(SN)**	**4**	**$0.29**	**$1.16**	**$265.80**
CONN HEADER VERT 2POS 2.54MM	**Digikey**	**S1012EC-02-ND**	**PREC002SAAN-RC**	**2**	**$0.06**	**$0.12**	**$265.68**
CONN HEADER VERT 14POS 1.27MM	**Digikey**	**1175-1630-ND**	**3220-14-0100-00**	**2**	**$0.70**	**$1.40**	**$264.28**
FIXED IND 4.7UH 3A 40.3 MOHM SMD	**Digikey**	**587-2100-1-ND**	**NR6028T4R7M**	**2**	**$0.36**	**$0.72**	**$263.56**
FERRITE BEAD 600 OHM 0805 1LN	**Digikey**	**MH2029-601YCT-ND**	**MH2029-601Y**	**2**	**$0.10**	**$0.20**	**$263.36**
LED RED CLEAR 0603 SMD	**Digikey**	**732-4978-1-ND**	**150060RS75000**	**4**	**$0.14**	**$0.56**	**$262.80**
LED GREEN CLEAR 0603 SMD	**Digikey**	**732-4980-1-ND**	**150060VS75000**	**2**	**$0.14**	**$0.28**	**$262.52**
LED YELLOW CLEAR 0603 SMD	**Digikey**	**732-4981-1-ND**	**150060YS75000**	**2**	**$0.14**	**$0.28**	**$262.24**
DC DC CONVERTER 3.3V 1W	**Digikey**	**1470-2396-5-ND**	**IE0503S**	**2**	**$4.15**	**$8.30**	**$253.94**
MOSFET N-CH 30V 500MA SOT-23	**Digikey**	**NTR4003NT1GOSCT-ND**	**NTR4003NT1G**	**2**	**$0.25**	**$0.50**	**$253.44**
RES 180K OHM 1% 1/8W 0805	**Digikey**	**738-RMCF0805FT180KCT-ND**	**RMCF0805FT180K**	**2**	**$0.10**	**$0.20**	**$253.24**

RES 120 OHM 5% 1/2W 1210	**Digikey**	**RMCF1210JT120RCT-ND**	**RMCF1210JT120R**	**2**	**$0.10**	**$0.20**	**$253.04**
RES 100 OHM 5% 1/2W 1210	**Digikey**	**RMCF1210JT100RCT-ND**	**RMCF1210JT100R**	**2**	**$0.10**	**$0.20**	**$252.84**
RES 10K OHM 1% 1/8W 0805	**Digikey**	**RMCF0805FT10K0CT-ND**	**RMCF0805FT10K0**	**2**	**$0.10**	**$0.20**	**$252.64**
RES 68 OHM 5% 1/2W 1210	**Digikey**	**RMCF1210JT68R0CT-ND**	**RMCF1210JT68R0**	**8**	**$0.10**	**$0.80**	**$251.84**
RES ARRAY 4 RES 100K OHM 0804	**Digikey**	**YC124J-100KCT-ND**	**YC124-JR-07100KL**	**2**	**$0.10**	**$0.20**	**$251.64**
TACT 4.5 X 4.5, 3.8 MM H, 2.5N,	**Digikey**	**PTS647SK38SMTR2LFSCT-ND**	**PTS 647 SK38 SMTR2 LFS**	**2**	**$0.13**	**$0.26**	**$251.38**
PC TEST POINT MULTIPURPOSE BLACK	**Digikey**	**36-5011-ND**	**5011**	**18**	**$0.40**	**$7.20**	**$244.18**
AUTO ISO CAN TRANSCEIVER 5KVRMS	**Digikey**	**296-52277-1-ND**	**ISO1042BQDWRQ1**	**2**	**$4.85**	**$9.70**	**$234.48**
IC MCU 32BIT 32KB FLASH 20TSSOP	**Digikey**	**497-17344-ND**	**STM32F042F6P6**	**2**	**$2.79**	**$5.58**	**$228.90**
OSC XO 8.000MHZ CMOS SMD	**Digikey**	**XC3029CT-ND**	**ECS-2520MV-080-CN-TR**	**2**	**$0.96**	**$1.92**	**$226.98**
CONN HOUSING VH 4POS 3.96MM WHT	**Digikey**	**455-1185-ND**	**VHR-4N**	**4**	**$0.18**	**$0.72**	**$226.26**

CONN SOCKET 16-20AWG CRIMP TIN	**Digikey**	**455-1319-1-ND**	**SVH-41T-P1.1**	**16**	**$0.11**	**$1.70**	**$224.56**
RES 330K OHM 1% 1/8W 0805	**Digikey**	**RMCF0805FT330KCT-ND**	**RMCF0805FT330K**	**10**	**$0.02**	**$0.23**	**$224.33**
RES 100K OHM 1% 1/8W 0805	**Digikey**	**RMCF0805FT100KCT-ND**	**RMCF0805FT100K**	**10**	**$0.02**	**$0.23**	**$224.10**
IWISS SN-28B Dupont Terminal Ratchet Crimper	**Amazon**	**SN-28B**	**SN-28B**	**1**	**$29.99**	**$29.99**	**$194.11**
Duratool HT-230C/HT-236C Hand Crimp Tool	**Newark**	**69C1031**	**HT-230C/HT-236C**	**1**	**23.84**	**$23.84**	**$170.27**
Thick Film Resistors 0805 18Kohms 1% AEC-Q200	**Mouser**	**667-ERJ-6ENF1802V**	**ERJ-6ENF1802V**	**2**	**$0.10**	**$0.20**	**$170.07**
Thick Film Resistors 0805 33Kohms 1% AEC-Q200	**Mouser**	**667-ERJ-6ENF3302V**	**ERJ-6ENF3302V**	**2**	**$0.10**	**$0.20**	**$169.87**
PCB Prints	**3W**			**2**	**$33.00**	**$66.00**	**$103.87**
Cell Board Population	**3W**			**2**	**$24.60**	**$49.20**	**$54.67**
				Current Remaining Budget			**$54.67**

Appendix C: Complete Code Listing

Embedded Code Github Link: https://github.com/willzhang05/modular-ev-bms

PC Program Github Link: https://github.com/willzhang05/can-websocket-bridge