ECE 445

Spring 2025

Senior Design Final Report

Integrated BMS/Battery System

Team 51

Adi Nikumbh, Ritvik Kumar, Rishav Kumar

TA: Shengyan Liu

Abstract

This report describes the design and implementation of our integrated battery management system (BMS)/battery project for ECE 445. Over the course of 16 weeks, our team worked together to design a system to meet requirements set forth at the start of the semester. This report will provide a description of this project's features, design process, validation, and costs.

Contents

1. Introduction

1.1 Problem

One issue with the development of embedded systems for small companies is the issue of battery packs. Manufacturing of a battery pack can be dangerous, and can require the expensive development of a custom BMS system. There are currently few options for completely developed and integrated lightweight battery packs that also contain a high quality BMS system that has capabilities of cutting voltage off in the event of issues.

With the rapidly increasing use of drones and autonomous robots in a wide range of applications, from agriculture to logistics, the need for reliable and safe battery systems is more important than ever. Our solution will help to ensure that these systems are safe and reliable, reducing the risk of development and making embedded systems safer for everyone.

1.2 Solution

Our solution is a prototype of a battery pack integrated with a battery management system (BMS). The system monitors battery conditions using temperature and voltage sensors, which are hosted on a PCB daughterboard interfacing directly with each cell. This daughterboard connects to a mainboard that processes sensor data and initiates fault protection if abnormal conditions are detected. These fault conditions include overvoltage, undervoltage, and over-temperature and under-temperature scenarios. In response to such events, the BMS is designed to take preventive measures against thermal runaway, such as disabling the battery output via a relay. The design's battery pack is in a 12s1p configuration, delivering a maximum voltage of 49.9V and a nominal voltage of 44.4V, making it suitable for a variety of applications ranging from drones to embedded systems.

The prototype uses twelve 13Ah pouch cells arranged in series to achieve the desired voltage characteristics. For sensing, the design incorporates the ADBMS6830 on the daughterboard. Communication between the sensors and the mainboard is handled via isoSPI, using the AD LTC6820. The mainboard employs an STM32H7 microcontroller in a compact LQFP package. Figure 1 provides a visual aid illustrating the system fully assembled.

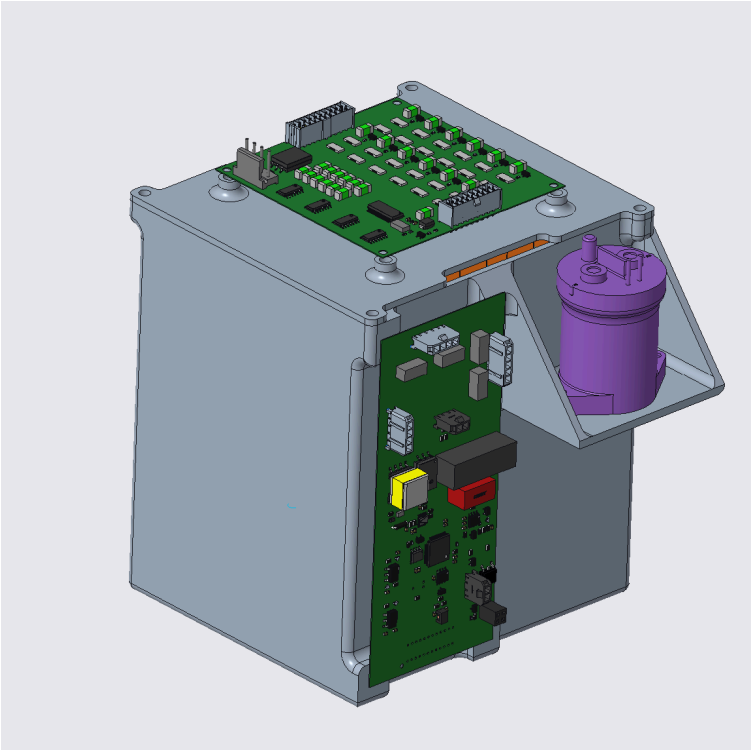

Figure 1: Visual aid depicting our prototype for BMS Mainboard (Controller) on the front, Daughterboard (Sensor) on the top, and Battery Pack encased within the enclosure.

1.3 High-Level Requirements

1. The BMS shall communicate with a “ground station” utilizing an interface to transmit updates and battery data for monitoring and SOH/SOC tracking every 100 ms.
2. The BMS shall be able to maintain cell voltages in the pack to within 100 mV via balancing.
3. The BMS shall monitor the temperature and voltage of cells in the pack and shall take immediate corrective actions when any fault conditions (overvoltage, undervoltage, under temperature, or over temperature) are detected according to the datasheet.

This report will provide details for the process and rationale behind designing such a system. Chapter 2 goes through all subsystems for this project, discussing design decisions and verification. Chapter 3 will focus on a cost analysis of this process, and Chapter 4 will detail the outcomes of the project and opportunity for future work. Appendices A, B, C, D, and E contain figures and images that provide justification for or aid in describing design decisions.

2. Design

2.1.1 Block Diagram

The block diagram is intended to provide a visual representation of the system, showing how different components interact with each other. There are four critical subsystems in our BMS, spread out across two printed circuit boards (PCBs) and a battery pack. The Battery Pack Subsystem consists of a 12s1p Li-Ion pouch cell configuration, supplying a total voltage of 44.4V. It includes temperature and voltage sensing circuits, provides a 5V reference from the first cell, and incorporates safety features such as a fuse and relays for controlled power delivery. The Daughterboard (Sensor) Subsystem features a battery stack monitor (ADBMS6830) responsible for voltage and temperature sensing, as well as balancing circuits to maintain uniform charge levels across cells. It communicates via isoSPI with the mainboard and derives its power from the battery pack. The Mainboard (Controller) Subsystem houses an STM32H7 microcontroller (MCU), which processes sensor data via SPI, controls switching mechanisms through GPIO, and manages overall system functions. It will communicate externally using CAN. Finally, the Power Subsystem utilizes a 24V external power supply to generate 5V and 3.3V outputs, ensuring stable power distribution to the MCU and other components. Figure 2 shows a block diagram of our system.

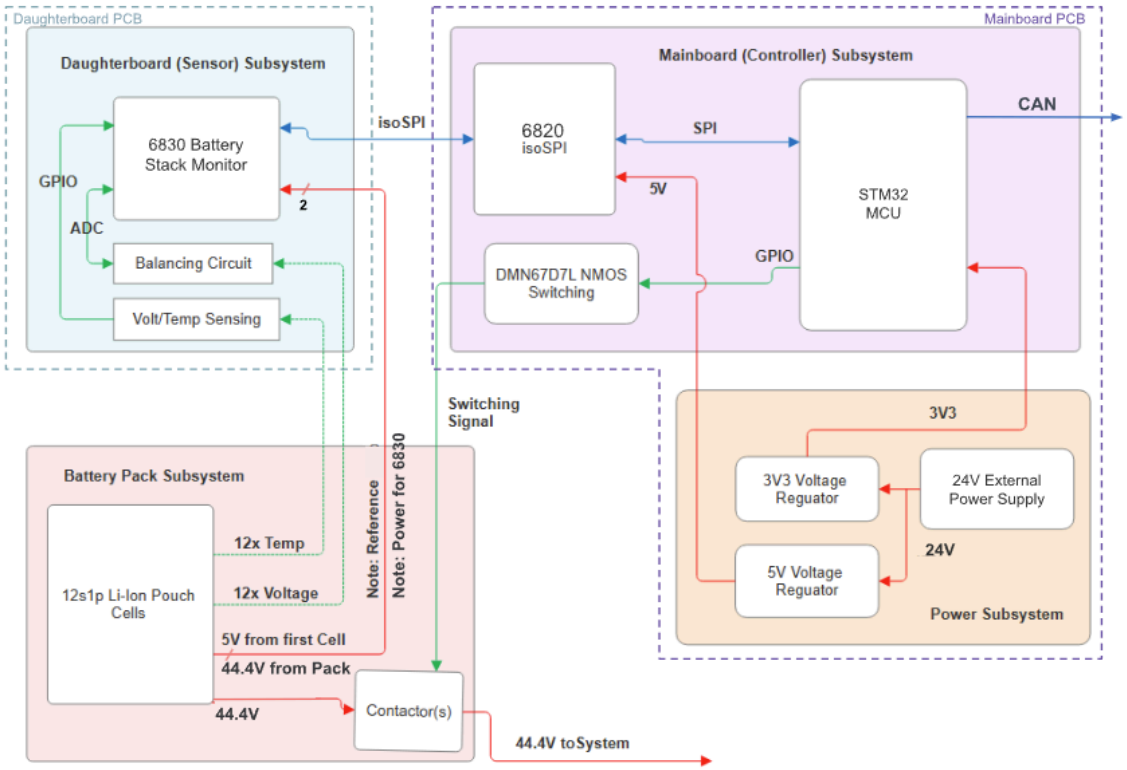

Figure 2: Block Diagram for Integrated embedded systems BMS/Battery

2.1.2 Physical Design

For this design, we use 13Ah pouch cells with dimensions of 155mm by 130mm [1]. In order to sense the temperature and voltage of the cell, we use ring terminals to affix the thermistors and the voltage sensors on the negative terminal and positive terminal respectively, effectively using them as washers in our series stackup. We 3DP a UL94 rated plastic box to house the cells, and also printed a lid to cover them. We use Molex style connectors to input our cell taps into the sensor subsystem, which are mounted to the top of the box with screws. On the side of the box, we have mounted the microcontroller subsystem. This PCB also has a Molex style connector in order to provide power and interface with the ground station.

2.2 Electronics Subsystems Overviews and Requirements

2.2.1. Battery Pack Subsystem

Description, Purpose, Justifications, and Interactions:

The battery pack subsystem is a 12s1p lithium ion battery pack. We use high capacity pouch cells with a nominal voltage of 3.7V [1]. We have chosen pouch cells due to being able to manufacture our pack without needing to spot weld. They are bolted together by the tabs. The cells are connected in series to create a 44.4V nominal battery pack, with a capacity of 13Ah.

The battery pack is housed in a lightweight and durable enclosure, with provisions for mounting the BMS and other components. The cells are bolted together with low resistance bolts, with polyimide tape insulating cell connections for electrical insulation. Furthermore, the pack is designed to be easily disassembled for maintenance and repair. The pack is a 3D printed enclosure that is UL 94 rated (industry standard flammability rating), and it is manufactured using heat set inserts for ease of assembly. These cells and cell configuration were chosen because they allow us to create a pack that is comparable to those used in robots in terms of capacity and voltage, such as Boston Dynamics' Spot [2], and enable ease of manufacturing the pack.

The battery pack only supplies voltage to the system (the relay closes) when no faults are detected. It supplies voltage and temperature sense, 5V power, and a reference to the daughterboard (sensor) subsystem. The relay in the battery pack subsystem is opened/closed by the mainboard (controller) subsystem.

Table 1: Requirements and Verification - Battery Pack Subsystem

Requirements	Verification	Equipment	Procedure	Results
The battery pack shall supply peak 44.4V ± 5% nominal voltage (max 50V) to the system when no fault conditions are present.	Measure the output voltage of the battery pack under normal operating conditions.	Multimeter or Oscilloscope	1. Connect the multimeter/oscilloscope probes to the output terminals of the battery pack. 2. Check voltage under normal operating conditions without any faults.	Voltage measured: 42.88V
The cells shall be rigidly connected to each other and mounted to the enclosure with protection against vibrations and requisite electrical isolation and safety.	Check the physical integrity of the battery pack and its mounting within the enclosure, including vibration protection and electrical isolation.	Visual Inspection	1. Inspect the battery pack to ensure it is mounted securely in the enclosure. 2. Simulate expected vibration by hand.. 3. Ensure electrical isolation is maintained between cells and the enclosure (e.g., check with an insulation resistance tester).	Image of cell assembly in Appendix A.1.

2.2.2. Mainboard (Controller) Subsystem

Description, Purpose, Justifications, and Interactions:

The mainboard is a PCB that hosts the microcontroller unit (MCU), isoSPI communication interface (Analog Devices LTC6820), the power subsystem, CAN transceivers, and various other components. It is responsible for processing data from the daughterboard and initiating fault protection in the battery management system (BMS) when improper conditions are detected. The mainboard uses a STM32H7 microcontroller in an LQFP package to process this data and uses MOSFETs to control the battery output through a relay. It also includes provisions for interfacing with the daughterboard via connectors, as well as communication links with external systems over the CAN bus.

The team selected the STMicroelectronics STM32H733VGT6 due to its high-performance ARM Cortex-M7 processor, which is capable of handling all required computations efficiently [3]. Its peripherals fulfill our requirements with its three FD-CAN interfaces used to communicate to the ground station, two 16-bit ADCs used for monitoring the power rail and detecting anomalies, and

two SPI lines used to interface with the LTC6820 and, consequently, the daughterboard. The circuit schematics for the MCU are included in Appendix B.1.

CAN was chosen as the primary communication protocol with the ground station because of its robustness, flexibility, and widespread use in industrial and automotive applications [4]. Furthermore, it sends messages every 10ms, ensuring the system operates well within the high-level requirements.

We selected the TCAN1044-Q1 as the CAN transceiver, which provides high-speed and reliable data transmission. The TCAN1044-Q1 supports speeds of up to 8 Mbps, making it well-suited for applications that require fast and efficient data exchange [5]. Additionally, it operates with a 5V power supply while supporting 3.3V logic levels, which aligns perfectly with the available power rails on our mainboard. The CAN transceiver will receive CAN signals from the MCU and convert it into the differential pair used to output the CAN signal. The circuit schematics are included in Appendix B.1.

The Analog Devices LTC6820 was selected for the isoSPI interface to enable bidirectional communication between isolated subsystems. Using isoSPI allows the high-voltage circuits on the daughterboard to remain galvanically isolated from the low-voltage circuits on the mainboard, through the use of pulse transformers at both ends to lock DC signals [6]. The LTC6820 translates isoSPI signals from the ADBMS6830 on the daughterboard into standard SPI signals for the MCU. The circuit schematics are included in Appendix B.1.

NMOS switching is used to send a fault signal to disconnect the relay on the battery pack if a critical failure occurs. The MCU will send the MOSFET the BMS fault signal, and the DMN67D7L MOSFET on the PCB will be provided with 24V for V_{DS}, which is within the 60V it was rated for [7]. However, after PCB assembly and testing, it was discovered that the DMN67D7L did not have a high enough current output to drive the relay. Therefore, we placed a FQP30N06L MOSFET at the drain of DMN67D7L, which then provided the required current for the relay [8]. The original circuitry is Appendix B.1 in and an image of the extra MOSFET is in Appendix B.2.

The MCU communicates with the daughterboard (sensor subsystem) via isoSPI and interfaces with external systems over CAN. It runs algorithms for cell balancing and fault condition monitoring and operates on a 3.3V input provided by the power subsystem.

Verification of this subsystem was performed through software testing, as described in Section 2.2.4. During these tests, the system produced the expected responses, including LEDs on the boards and signals received by the laptop acting as the ground station. Further verification was done using continuity checks, all of which passed successfully following PCB assembly.

Table 2: Requirements and Verification - Mainboard (Controller) Subsystem

Requirements	**Verification**	**Equipment**	**Procedure**	**Results**
The subsystem shall have a stable connection to the daughterboard (sensors) subsystem.	Test the continuity of the connection between the subsystem and the daughterboard.	Multimeter or Oscilloscope	1. Perform a continuity check between the subsystem and the daughterboard connections. 2. Apply a signal to the system and verify that the signal is received by the daughterboard (e.g., using an oscilloscope).	Passes continuity check and signal is received from the daughterboard (shown in Software Control).
The subsystem shall correctly determine faults in the battery pack and appropriately turn on/off the relay(s) in the battery pack.	Simulate fault conditions and check if the relays are turned on/off as expected.	Fault Simulation Equipment (e.g., power supplies/other equipment to simulate short circuit, over-voltage, under-voltage test equipment), Multimeter	1. Establish nominal operating conditions; verify circuits operate as desired. 2. Introduce fault conditions such as short circuits, over-voltage, or under-voltage. 2. Observe the subsystem's response and check if it turns on/off the relays; also monitor voltages of key signals. 3. Measure the voltage and current to ensure the fault condition is properly handled.	Test results show correct relay behavior (on/off state) and fault detection when fault conditions are induced through disconnecting isoSPI connectors.
The subsystem should send commands over SPI to the LTC chips in order to be able to discharge the	Monitor SPI communication between the subsystem and the LTC chips, and verify that the discharge	Logic Analyzer, Oscilloscope	1. Use a logic analyzer to monitor the SPI communication signals. 2. Verify that the discharge command	Correct SPI transaction for discharging cells as seen through data and LEDs.

cells.	command is correctly sent.		is being sent correctly according to the communication protocol.	
The subsystem shall send and receive messages over CAN to interface with external system(s).	Monitor CAN communication between the subsystem and computers and verify correct information is sent.	Computer, LED	1. Program MCU to blink LED when messages are sent. 2. Connect MCU to CAN bus and send expected information to the computer. 3. Verify packets from the test program are being received as expected. 4. Send packets of data from the computer to change LED blinking speed; verify changes are reflected in LED.	CAN packets include correct information; LED blinks as expected.

2.2.3. Daughterboard (Sensor) Subsystem

Description, Purpose, Justifications, and Interactions:

This subsystem is a PCB that hosts the temperature sensing circuitry, as well as the voltage taps and balancing circuits.. The daughterboard is connected to the mainboard via a two wire isoSPI interface, which will allow for easy communication between the two boards. This isoSPI interface enables galvanic isolation between the daughterboard and mainboard. The daughterboard is responsible for monitoring the temperature and voltage of each cell in the battery pack (i.e., 12 voltage and 12 temperature sense lines), and sending this data to the mainboard for processing. The daughterboard uses an Analog Devices (ADBMS6830) chip to monitor the voltage of each cell, and uses NTC thermistors to monitor the temperature of each cell.

The ADBMS6830 was chosen because it can measure up to 16 cell voltages simultaneously, provides passive balancing with modular control over every channel, and has a bidirectional isoSPI interface [9]. Our project requires 12 cell voltages to be monitored, but this chip allows for room to add cells if required. The ADBMS6830 also contains pins to have redundant cell measurement paths if desired in the future, allowing for development of a more robust system. According to the datasheet, the ADBMS6830 can read cell voltages ranging from −2.0V to

+5.5V. Our cells will operate from 3.0V to 4.28V, with a nominal voltage of 3.7V, so this chip is tolerant to our expected voltages with a safety factor [1]. The daughterboard also includes circuitry to balance the cells based on readings from the sensors. If there is an imbalance greater than the threshold detected by the voltage taps, circuitry to discharge the cells is included on the daughterboard. A blue LED is also illuminated to indicate the cells that are discharging. The circuit schematics are included in Appendix C.1.

The ADBMS6830 operates in the range of -40°C to 150°C, which is within the expected temperatures of the cell (cells are not expected to have operating temperature exceed 60°C) [9]. The ADBMS6830 also has a built-in bidirectional isoSPI interface, which will enable stable communication with the Mainboard with little additional complexity. This circuit, with associated filters, is included in Appendix C.1.

The NTC thermistors have an operating range of -50°C to +150°C, with a tolerance of ±0.5% [10]. These values are within our required specifications of ±1%. They have ring terminals and will be bolted to the cells, allowing for a stable and easy way to connect to the battery pack.

The ADBMS6830B offers 10 GPIOs [9], but temperature readings are needed from 12 cells. Therefore, the temperature sensing circuitry must include a way to reduce the number of inputs. One solution is to use an analog multiplexer to select between pairs of cell temperature readings. With this setup, one GPIO can serve as a select while six others can be used to read the outputs. However, a challenge arises in determining which of the two temperatures in each pair to use. If only the first temperature is selected and the second happens to be significantly higher, a potential fault could go undetected. To address this, we designed a sensing circuit that conservatively chooses the higher temperature from each pair. While this might occasionally overlook low-temperature faults, such conditions are unlikely during normal operation. The circuit schematic is provided in Appendix C.1. This selection circuit is replicated for each adjacent cell pair across all 12 cells.

The daughterboard is powered by the first cell in the battery pack and receives a reference voltage from there as well. The daughterboard's schematics and final layout are included in Appendix C.1.

Table 3: Requirements and Verification - Daughterboard (Sensor) Subsystem

Requirements	Verification	Equipment	Procedure	Results
Read temperatures within ±1% and read voltages within ±20mV.	Measure the temperature and voltage readings from the sensors and verify the accuracy.	Thermometer /Infrared Temperature Gun/Camera, Multimeter	1. Measure temperature using a calibrated thermometer/infrared temperature gun/camera. 2. Measure the voltage using a calibrated multimeter. 3. Compare the readings against expected values.	Graph of voltage measured through multimeter vs voltage measured through ADC is in Appendix C.2.
Communicate to the MCU over isoSPI to provide constant updates for pack monitoring.	Monitor the isoSPI communication and ensure that continuous updates are being transmitted to the MCU.	Oscilloscope, Logic Analyzer	1. Connect oscilloscope/logic analyzer to the isoSPI bus. 2. Observe the continuous data stream from the sensors to the MCU.	Correct SPI transaction for discharging cells as seen through data and LEDs.
Provide stable connection to battery pack to read voltages and temperatures.	Test the connection to ensure that stable data can be read continuously from the battery pack's sensors.	Multimeter, Oscilloscope	1. Connect measurement equipment to the battery pack's voltage and temperature sensors. 2. Verify that readings are stable over time with no significant fluctuations.	Graph showing voltage before and after balancing (~30 minutes) which indicates a stable connection is included in Appendix C.2.

2.2.4. Software Control

To accomplish this project, a significant software effort was made in order to be able to take full advantage of the hardware circuits that we designed. To do this, we made a STM32 project in order to use the STM HAL library to access the SPI, CAN, GPIO, and TIM peripherals. To allow communication with the voltage sensing chip we had on the daughterboard, we needed to

interface with it over isoSPI. Since most microcontrollers do not have an isoSPI peripheral built in, we interfaced via SPI with the LTC6820 chip, which then converted the messages we sent to it to isoSPI and talked to the 6830, essentially acting as a middleman. To do this, we used the basic process described in Figure 1 to communicate with the chip.

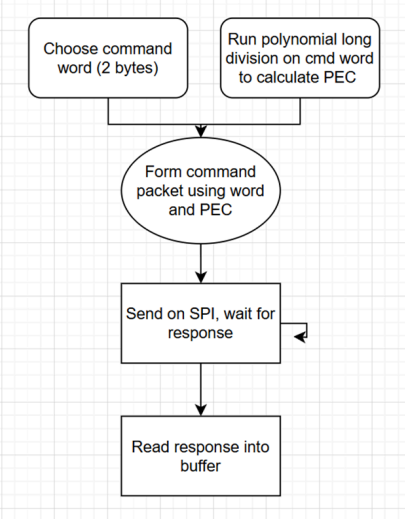

Figure 3: Flowchart for isoSPI message handling

We start by choosing the command we want to send, for example ‘read all cell voltages’. We then calculate a CRC, which is used for error checking to make sure our command is read properly. We combine the calculated CRC, or ‘PEC’ with the command word, and send this over SPI. We then wait for the chip to respond and read its response into a buffer, for example ‘voltages[12]’. This process is what we used to talk to the 6830 and read voltages and temperatures. The pseudocode we used for our breadboard demo to get a minimum viable product (MVP) of voltage sensing working is included in Appendix D.

In order to balance, we used a similar process of communication and sent the 6830 a message telling it what cell groups to discharge. This manipulated the gate of our NMOS on the daughterboard via a GPIO, and then discharged the cell for as long as the 6830 was sending the signal. We created an algorithm on top of this in order to dynamically choose which cells to balance, which is visually described in Figure 4.

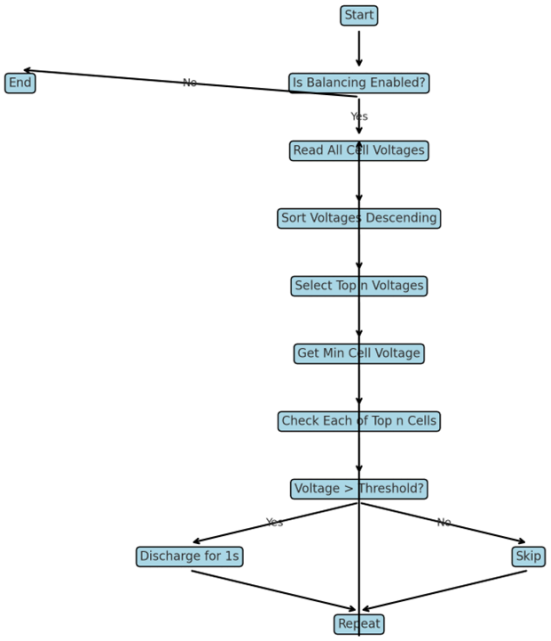

Figure 4: Flowchart for passive cell balancing algorithm

The algorithm first checks if balancing is enabled, which is controlled via the ground station. It then reads all the voltages read into an array and sorts the array in a descending order. Via the ground station, we specify the number of cells we want to balance at a time, and pick a subarray of this number of cells. We iterate through this array, and check if any of the cells have a difference from the specified voltage (15mV). If they do, we discharge them by sending the SPI command mentioned previously.

In order to check the faulting of our system, we used a very simple algorithm to determine whether to open our relay or not, as described in Figure 5.

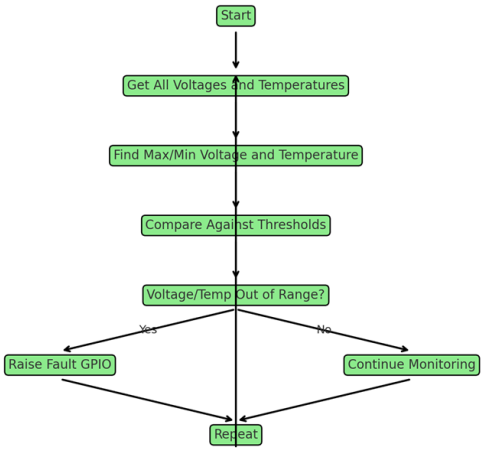

Figure 5: Flowchart for fault detection and handling

We get all the voltages and temperatures, and choose the highest and lowest of each. We then compare each of these values to our datasheet specified limits. For the threshold values, we chose 4.2V for the top end, 3.0V for the bottom, and 10℃ for the bottom end of the temperatures and 60℃ for the high end. These values are all based on the cell datasheets and the nominal operating conditions of lithium-ion cells.

In order to organize all this data, we created an array of struct called `bms`. Each struct has three members; a temperature, a voltage, and a boolean fault condition. Every loop iteration (200ms), we look through each cell and repopulate this array with new values. We have another struct of critical values that is compared every loop to see if there is a fault or not.

While in a fault condition, a GPIO pin is driven high to trigger the NMOS to open the contactors. When the system boots, we do our preliminary check of fault conditions, and then close contactors. This means that there is a few ms wait time from the low voltage of our system turning on before high voltage is enabled.

2.2.5. Power Subsystem

Description, Purpose, Justifications, and Interactions:

This subsystem receives a 24V input from an external power supply and modulates the voltage to 3.3V and 5V to power the STM32 microcontroller and isoSPI interface chips. Initially, we wished to use a 12V external power supply, however, due to the relay having a minimum input of 17V, we switched to 24V, which is a common low voltage bus in applications we are designing for.

We are using buck converters from 24V to 3.3V and 5V because they are more efficient than LDOs (dissipate less power and heat into the board), as shown below:

$$P_{LDO} = (V_{in} - V_{out}) * I = (24V - 5V) * 300mA = 5.7W$$
$$P_{Buck} = P_{out}/E - P_{out} = 5V * 300mA/0.9 - 5V * 300mA = 1.67W$$

Originally, we selected the LMR23630 regulator, however we switched to using a LM2592 regulator. Both can provide the required 1A outputs of 3.3V and 5V with a 12V input, but the LM2592 had greater efficiency, as is seen in Figure 6 [11]. This choice ensures stable power delivery while minimizing thermal losses, enhancing the overall performance and reliability of the system.

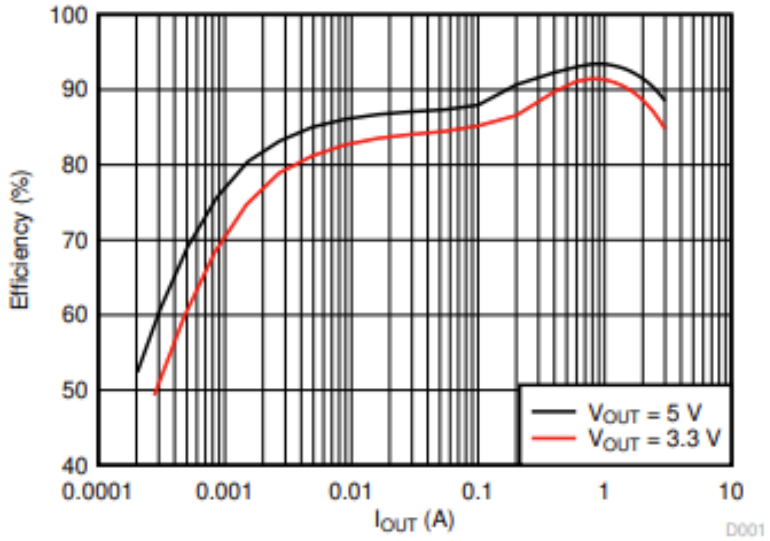

Figure 6: Efficiency vs Load with V_{in}=24V for LMR2592 [11]

After PCB assembly, the team tested the output of the buck converters and observed voltage errors of 0.2% for the 5V converter and 7.0% for the 3.3V converter, as shown in Appendix E.2. As a result, only the 5V converter remained within the tolerance range in our subsystem requirements. The discrepancy in the 3.3V converter output is attributed to the change in the external power supply from 12V to 24V. However, the measured output of 3.53V remains well within the STM32's maximum voltage rating of 4V and therefore does not affect the functionality of the mainboard [3].

Table 4: Requirements and Verification - Power Subsystem

Requirements	**Verification**	**Equipment**	**Procedure**	**Results**
Provide 3.3V ± 5% and 5V ± 5% to microcontroller and isoSPI interface chips.	Measure the supply voltages at the input pins of the microcontroller and isoSPI interface.	Multimeter or Oscilloscope	1. Measure the voltage of the microcontroller and isoSPI chips. 2. Verify that the voltage stays stable within the 3.3V and 5V tolerances as we run programs for the BMS. Validate there are no major spikes or drops in voltage when running through all operations of the BMS.	Voltages are measured at 3.53V and 5.04V with no major spikes/drops, as seen in Appendix E.2.

3. Cost and Schedule

3.1 Cost Analysis

3.1.1 Labor Costs

Below are the individual labor costs by person and the calculated the total labor cost, using the assumptions that work is equally split evenly and cost of labor is $45 per hour:

1. Adi: $45/hour * 2.5 * 100 = $11,250
2. Rishav: $45/hour * 2.5 * 100 = $11,250
3. Ritvik: $45/hour * 2.5 * 100 = $11,250
4. Total Labor Costs: $33,750

3.1.3 Parts Costs

The estimated total cost of parts is $304.30 and is broken down in Table 5. Please note that parts listed in this table are subject to change, so prices may vary.

Table 5: Cost of Required Parts

Description	**Manufacturer**	**Part #**	**Quantity**	**Price**
Battery Cells	CosMX	95B0D0HD-13 Ah	12	$120
PCB Fabrication	PCBWay	N/A	2-3	$65
Microcontroller	STMicroelectronics	STM32H733V GT6	2	$26
36-V, 3-A Synchronous Step-Down Converter	Texas Instruments	LMR2592	2	$4
isoSPI Isolated Communications Interface	Analog Devices	LTC6820	1	$3
16-Channel Multicell Battery Monitor	Analog Devices	ADBMS6830	1	$20
Thermistors	Vishay Beyschlag/Draloric/	541-10746-ND	12	$35

	BC Components			
Op-Amps	Analog Devices	OP467G	24	$0.40
MOSFET	Diodes Incorporated	DMN67D7L	2	$0.50
MOSFET	Fairchild	FQP30N06L	1	N/A
Connectors	Samtec	Misc.	6	$20
Passive Components	DigiKey	Misc.	120	$10

3.1.3 Total Cost

Using the labor and part costs derived in sections 3.1.1 and 3.1.2, we calculate the final total cost to be:

GRAND TOTAL: $33,750 + $304.30 = $34,054.30

3.2 Schedule

Here is our schedule broken down by week and person:

Table 6: Week-by-Week Schedule

Week 3/3/25	Finalize IC architecture and cells (all) Work on Breadboard Demo (Rishav, Adi) Finalize PCB schematics (all)
Week 3/10/25	Present Breadboard Demo, identify and document bugs (all) Finalize PCB layout for Revision 1 and place orders (all) Develop software (all)
Week 3/17/25	Spring break
Week 3/24/25	Receive PCB, solder, and validate (Rishav, Ritvik) Update PCB schematics and layout as necessary for Rev. 2 (all) Flash software to PCB for first time and test I/O (Adi) Test voltage sensing and temperature sensing (Rishav, Ritvik)
Week 3/31/25	Bring up communication between PCBs (all) Order Rev. 2 PCBs (all) Update PCB designs for new chips and scale - Rev. 3 (all) Start bringing up code for interaction with ground station (Adi, all)

Week 4/7/25	Receive PCB, solder, and validate Rev. 2 (Rishav, Ritvik) Keep working on software (Adi, all)
Week 4/14/25	Complete manufacturing on electronics and integration (all) 3DP all enclosures (Rishav, Ritvik)
Week 4/21/25	Finishing touches, practice demo, make look nice for Rev 2. (all) Rev. 3 received, assemble and attempt integration (all)
Week 4/28/25	Final demo and requirements (all)

4. Conclusion

The major accomplishment of the integrated BMS design is being able to meet all the high-level requirements. The BMS was able to measure the voltage and temperatures of the battery cells within tolerances, react to any faults quickly, and communicate with external devices through the CAN bus. Further success can be observed through the quality of the PCBs created, and the minimal hardware modifications required to get this project to work. Overall, we determined this project a success, as we created a single integrated package that accomplished all the goals we set out to achieve. It was an incredibly rewarding experience to pick up our integrated battery pack and have all the features onboard.

In terms of uncertainties, there are some hardware changes and analysis that we would like to run. For example, the thermal constraints on balancing that cause us to need to use the pulsed discharge method could be mitigated via active cooling, heatsinks, or running a software such as Ansys Icepak in order to see how the thermal output affects the PCB. The main constraint of the resistors is that they cannot cause the cells or the LTC chip to overheat, as this would negatively impact performance, so knowing the absolute limits in temperature we can reach with them would be ideal. Additionally, it would have been preferable to move to a different PCB mounted relay that was able to be controlled with a PCB mounted component. As noted earlier, the high current draw of the large relay we used meant that we needed to use an external NMOS chip driven off of our onboard NMOS chip in order to open the circuit. It would have been cleaner and more compact to use a PCB mounted one.

In terms of future work, we would like to be able to get the current sensing functionality working. This could be accomplished by using either a hall effect current sensor or a shunt resistor and an ADC to measure it. Doing this would allow us to output data on pack voltage, current, and power, all without having to do anything highly complex on the software side. We are able to get a current estimate with the current hardware that we are using, by taking the voltage sag of the cells over time and using their internal resistance as a base point, but this is quite inaccurate. Another thing that we would like to work on is the scalability of our design. Creating a setup so that we could add more cells to our setup would be the logical next step of our design. It would be relatively simple in order to extend our current design by daisy chaining daughterboards/sets of 12 cells with each other, using the mainboard as a central controller because of our current communication setup. We also have the ability to monitor 16 cells in one pack due to the capabilities of the battery stack monitor we chose.

4.1 Safety

The primary safety concern for this project is the use of lithium-ion batteries. As unregulated energy sources, these batteries can discharge extremely high currents if accidentally shorted across a low-resistance path. For instance, a short circuit caused by an Allen key or a piece of

wire could result in a current surge exceeding 100A, even at just a few volts across the cell terminals. To mitigate these risks, we adhered to strict safety protocols, including the use of high-dielectric-strength insulated tape and code-compliant connectors and plugs.

Additionally, we exercised extreme caution to prevent accidental contact with battery leads and avoid any potential short circuits, especially during manufacturing of the battery pack [12]. However, since our system operates at a nominal voltage below 50V, compliance with high-voltage safety standards will be less stringent.

4.2 Ethics

In terms of ethical issues, there is the issue of the usage of our project in unethical applications. Since our design is general purpose and marketed towards industry, it is possible that a use case could be found in the military complex, as there are many use cases where a high quality battery could be used; for instance, autonomous vehicles and/or drones. To follow the IEEE Code of Ethics I.3 – which states, “to hold paramount the safety, health, and welfare of the public” – we will avoid ethical breaches by making sure that our designs stay private and are only seen by people we trust, and that if safety concerns arise while working on the project we will disclose them [13].

Furthermore, in accordance with the IEEE Code of Ethics I.5 – which states “to be honest and realistic in stating claims or estimates based on available data” – we committed to making safe and realistic decisions based on available data when designing, testing, and discussing our project so as to not endanger or overestimate its capabilities.

Appendix A: Battery Pack Subsystem

A.1 Assembled Battery Pack

Figure 7: Image of battery pack during verification after manufacturing

Appendix B: Mainboard Subsystem

B.1 PCB Schematics and Design for the Mainboard Subsystem

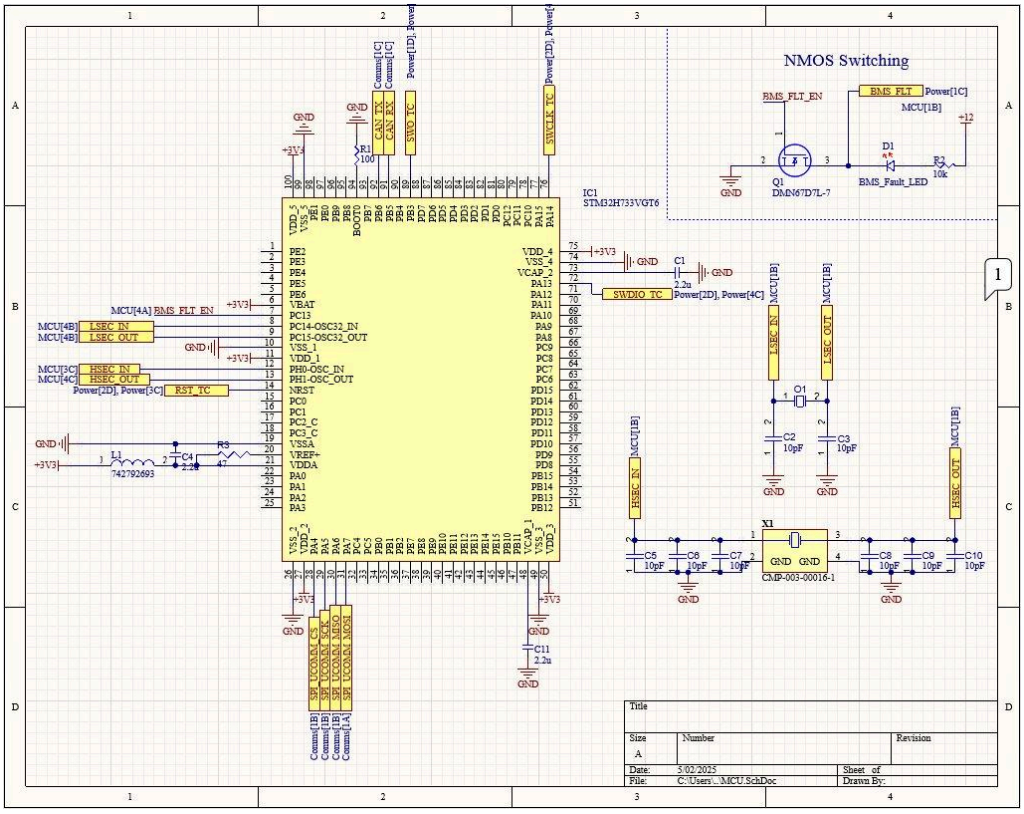

Figure 8: Schematic of MCU and NMOS switching (top right)

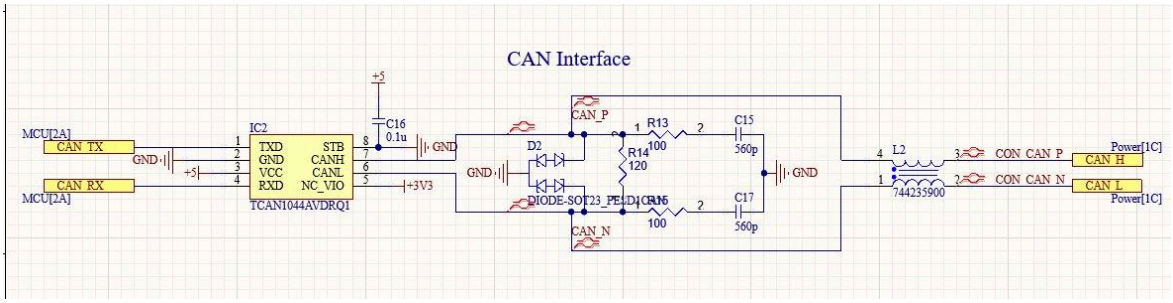

Figure 9: Schematic of CAN communication interface using TCAN1044-Q1

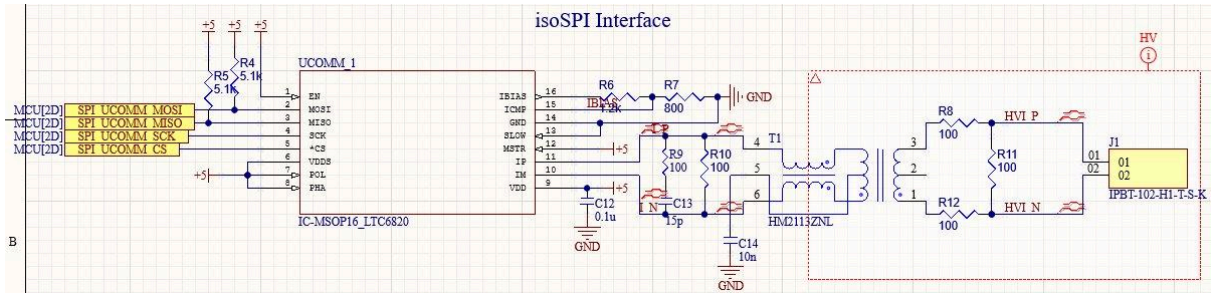

Figure 10: Schematic of isoSPI communication interface using LTC6820

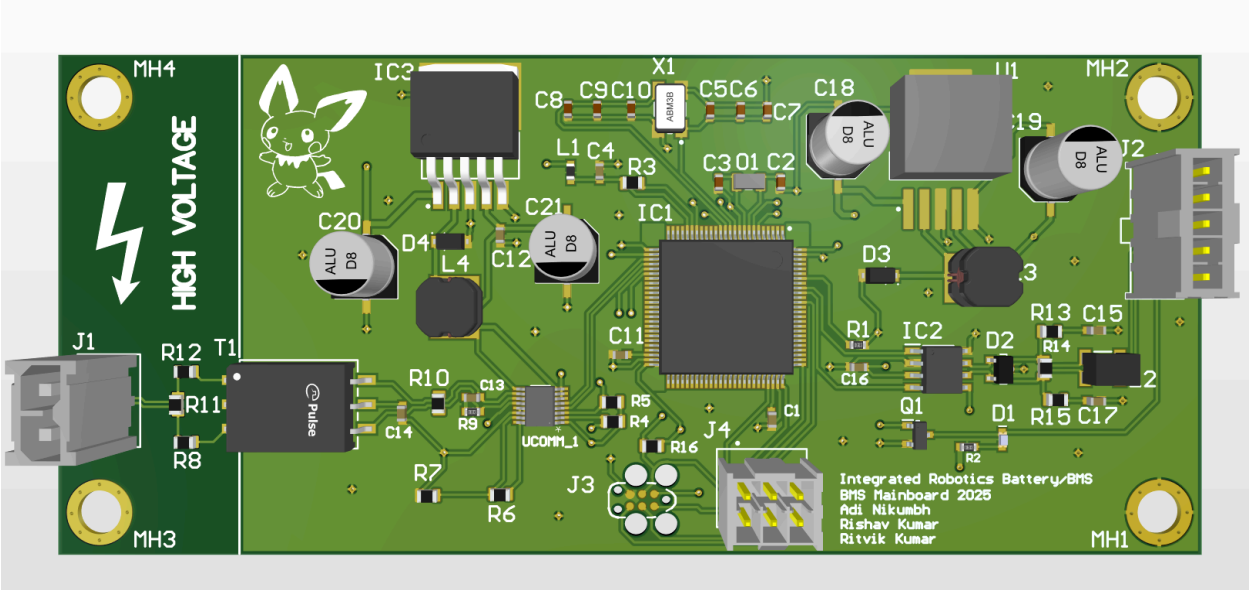

Figure 11: PCB layout of mainboard and power subsystems

B.2 Design Change after PCB Assembly to Achieve Functionality

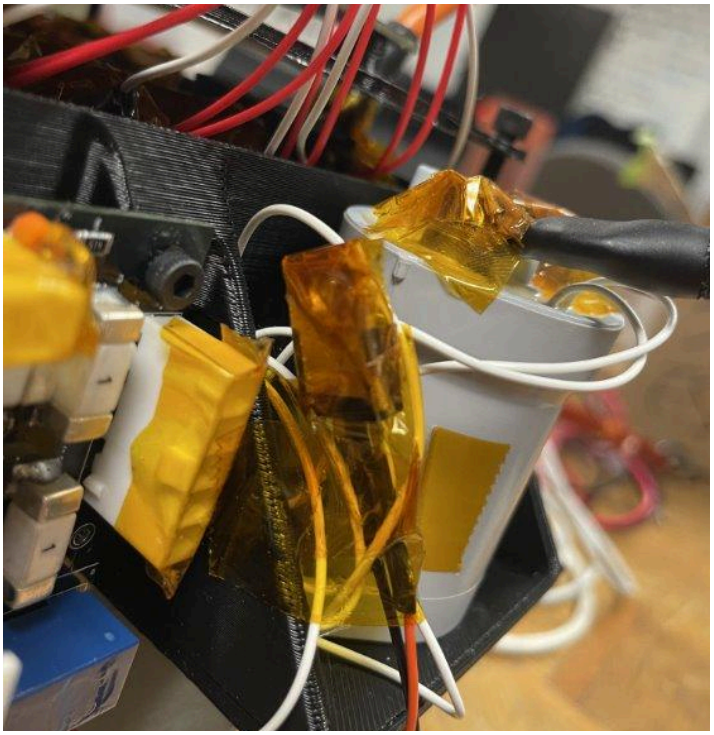

Figure 12: Image of external NMOS which ensures the relay receives required current

Appendix C: Daughterboard Subsystem

C.1 PCB Schematics and Design for the Daughterboard Subsystem

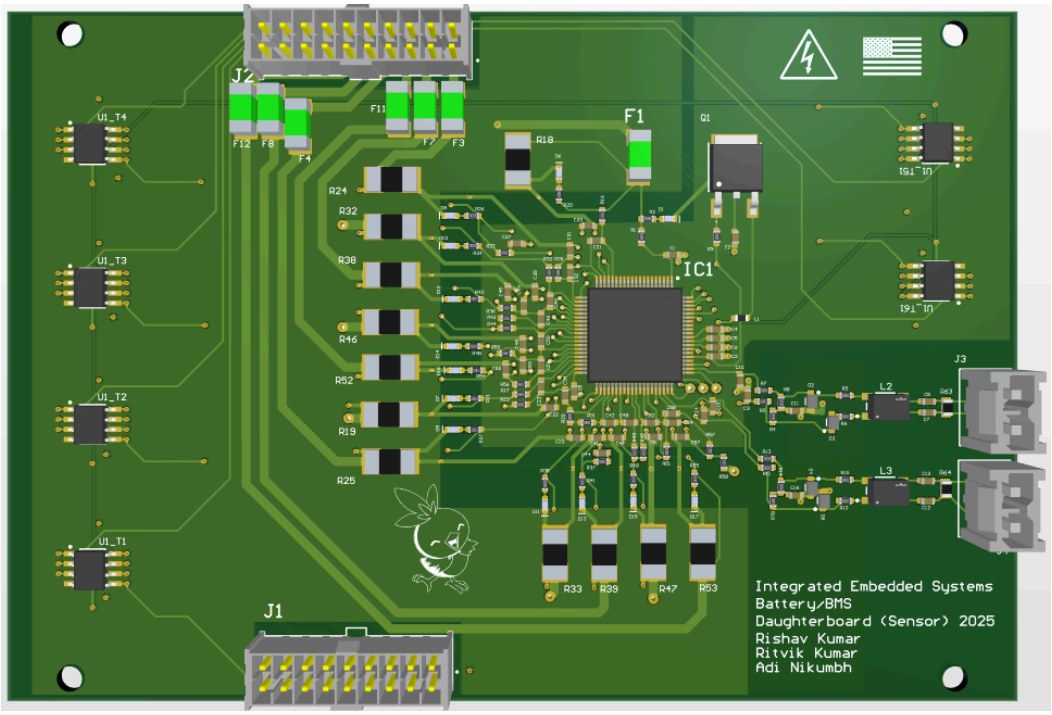

Figure 13: Front of daughterboard PCB layout with components

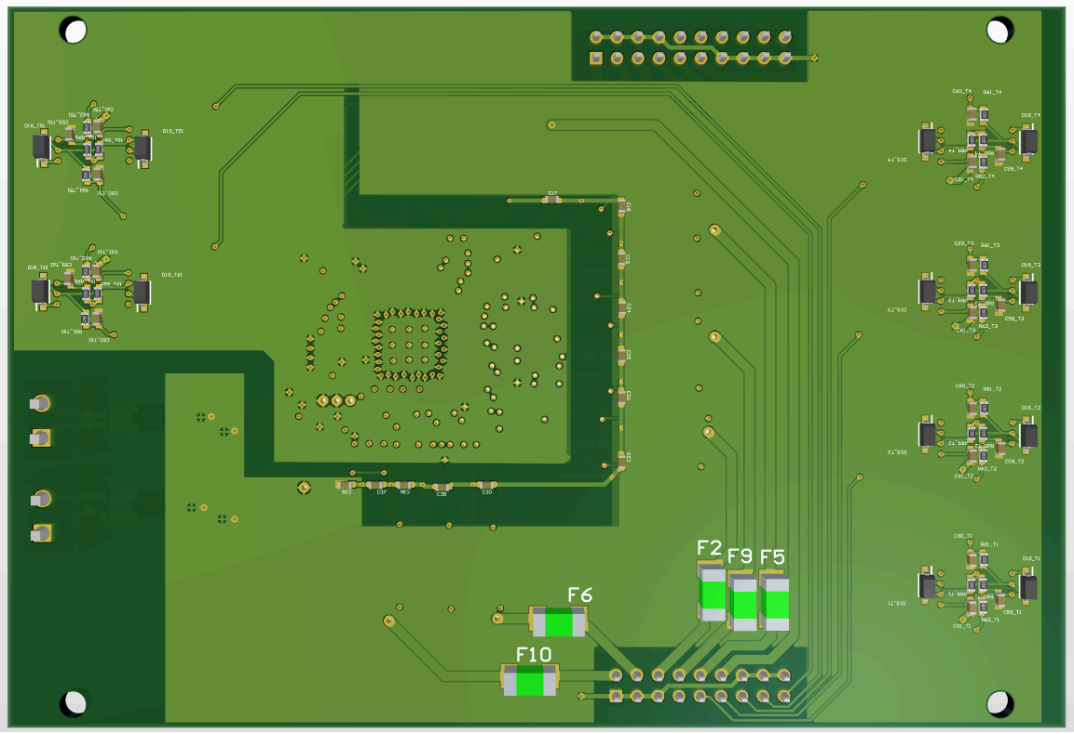

Figure 14: Back of daughterboard PCB layout with components

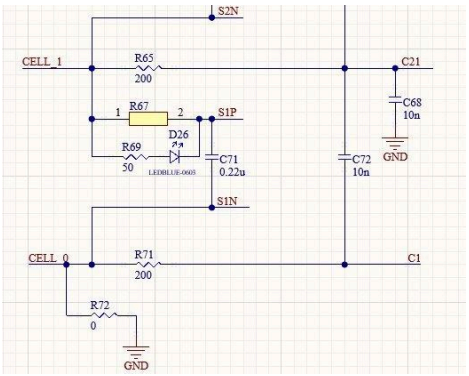

Figure 15: Schematic of cell balancing circuit

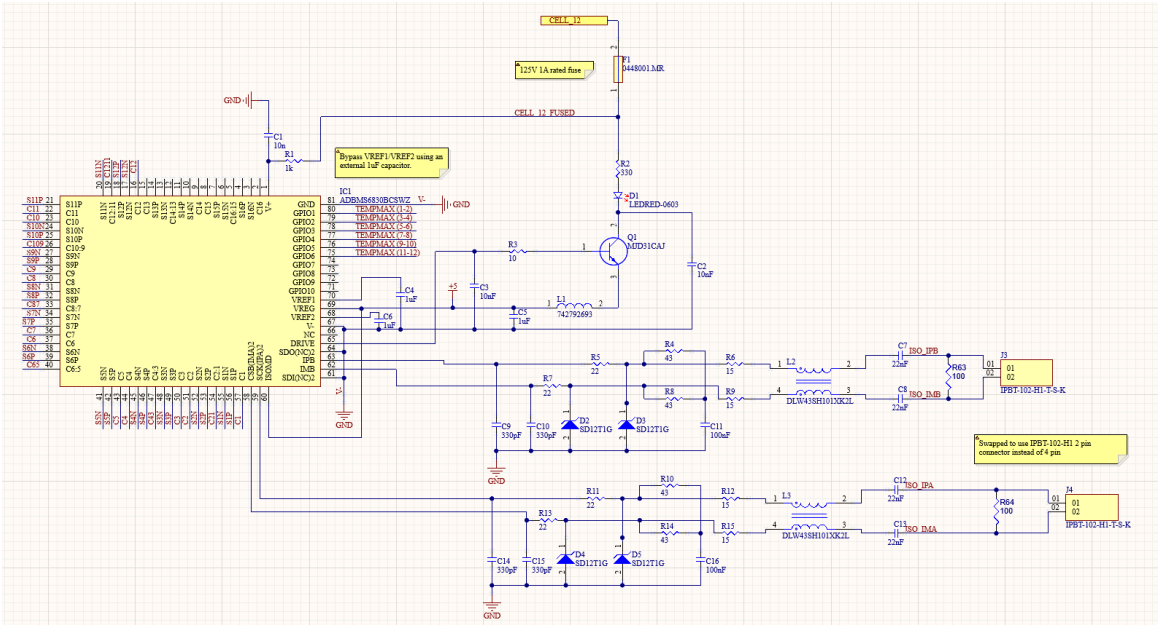

Figure 16: Schematic of ADBMS6830 circuit with isoSPI interface

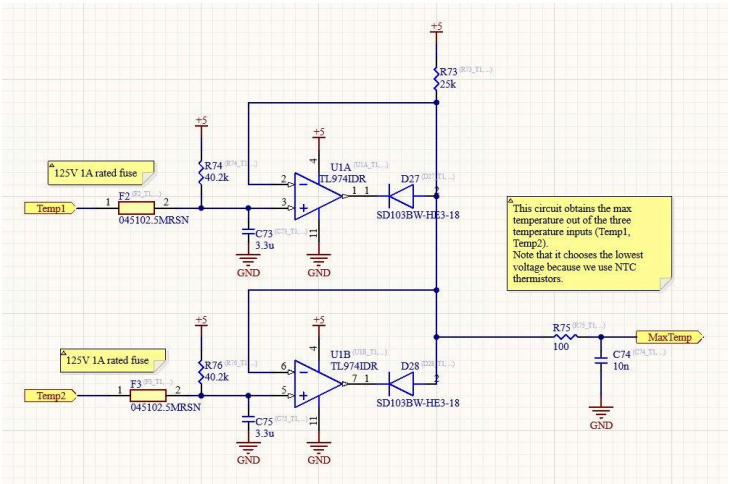

Figure 17: Schematic of analog multiplexer for temperature selection

C.2 Verification of Daughterboard Functionality

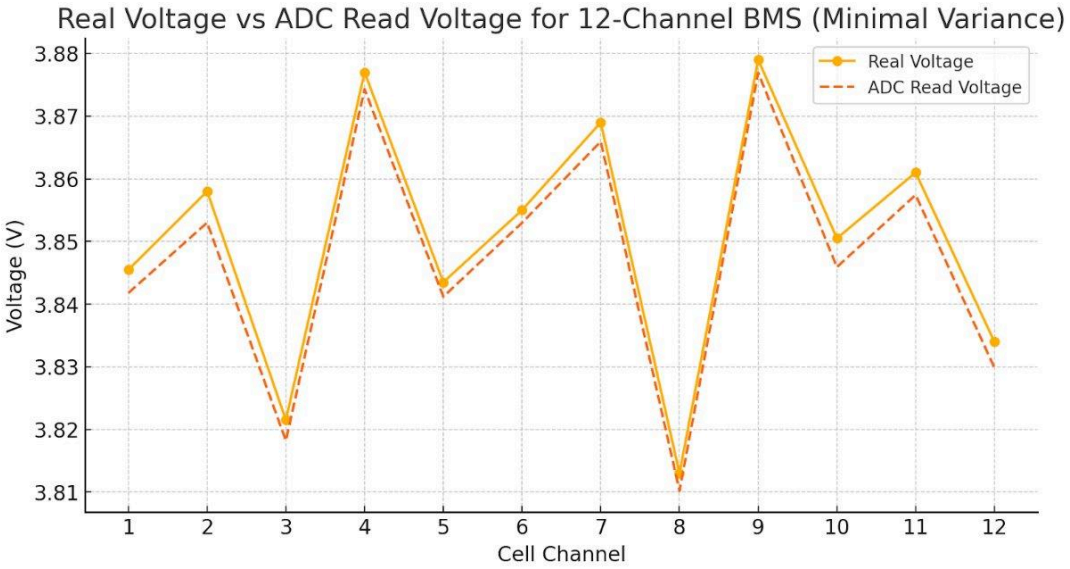

Figure 18: Graph of voltage measured using multimeter compared to ADCs

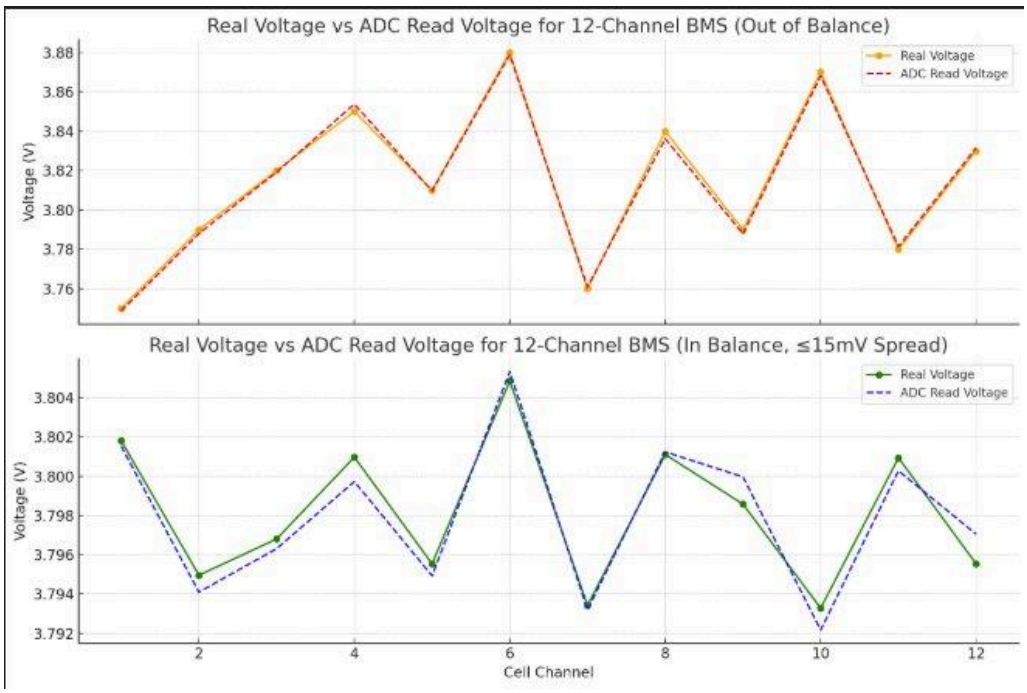

Figure 19: Graph showing cell voltages before and after balancing for 40 minutes (bottom graph's scale is 0.1x the top graph's)

Appendix D: Software Subsystem

```
loop forever:
  // Wakeup sequence
  set CS pin low
  wait 2 ms
  set CS pin high
  wait 2 ms

  // ADCV command (Start cell voltage conversion)
  create command_buffer[4]
  command_buffer[0:1] = ADCV command bytes
  calculate PEC for first 2 bytes
  append PEC bytes to command_buffer
  send command_buffer via SPI

  // SNAP command (Capture cell voltages)
  reset command_buffer
  command_buffer[0:1] = SNAP command bytes
  calculate and append PEC
  send command_buffer via SPI

  // RDCVF command (Read captured voltages)
  reset command_buffer
  command_buffer[0:1] = RDCVF command bytes
  calculate and append PEC
  send command_buffer via SPI
  wait 2 ms
  receive SPI response into read_buffer

  // UNSNAP command
  reset command_buffer
  command_buffer[0:1] = UNSNAP command bytes
  calculate and append PEC
  send command_buffer via SPI

  // Process voltage readings
  for i in 0 to 7:
      combine byte pairs from read_buffer (little-endian)
      convert to voltage: raw_value × 0.0001 volts
      store in voltages array

  // Over-voltage check
  if voltages[0] > 6.4 V:
      turn on FAULT_LED
  else:
      turn off FAULT_LED

  // Resistance calculation
  fixed_resistance = 3.25 Ω
  supply_voltage = 3.3 V
  read ADC_value from sensor
  calculate variable_resistance using ADC_value, fixed_resistance, and supply_voltage
```

Figure 20: Pseudocode for MVP for voltage sensing

Appendix E: Power Subsystem

E.1 PCB Schematics for the Power Subsystem

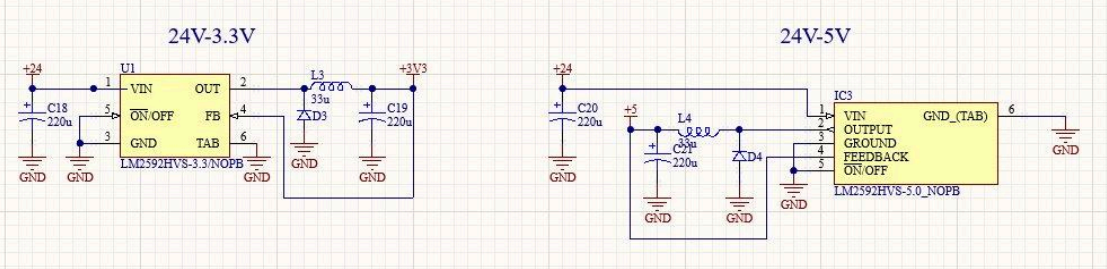

Figure 21: Schematics for the buck converters that provide 3.3V and 5V to the mainboard

E.2 Verification for the Power Subsystem

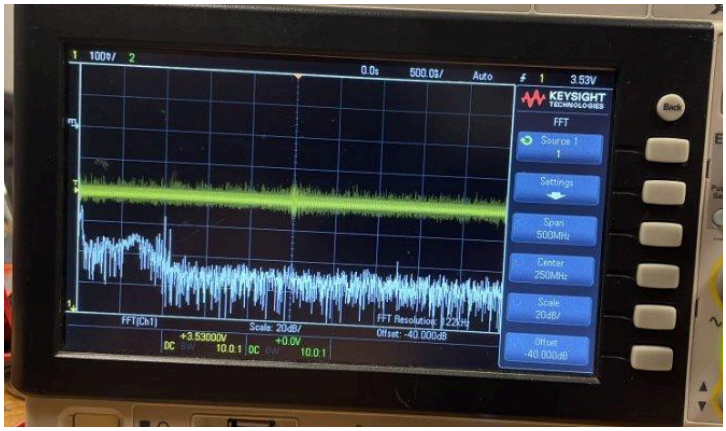

Figure 22: Oscilloscope showing the output of the 3.3V buck converter (yellow line) and the FFT to demonstrate the switching frequency (white line)

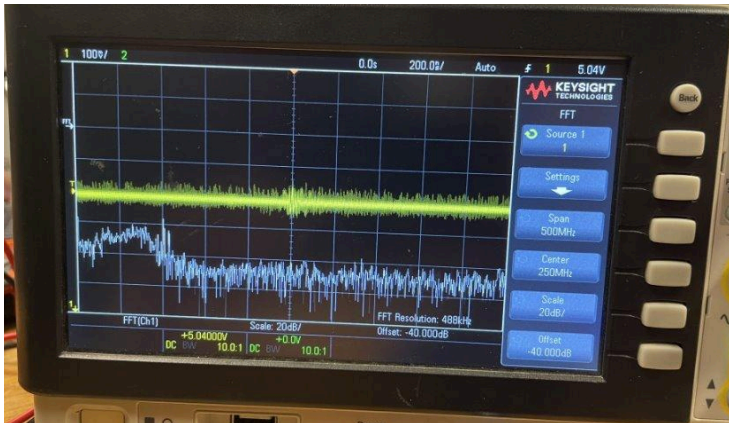

Figure 23: Oscilloscope showing the output of the 5V buck converter (yellow line) and the FFT to demonstrate the switching frequency (white line)

References

[1] CosMX, “13 Ah 95B0D0HD Pouch Cells,” 95B0D0HD datasheet. https://www.fsaeonline.com/cdsweb/gen/DownloadDocument.aspx?DocumentID=12264c1e-ce5b-46ad-b74f-b291a900ac77. [Accessed Feb. 10, 2025].

[2] Boston Dynamics, “Boston Dynamics Support Center,” Bostondynamics.com, 2024. https://support.bostondynamics.com/s/article/Spot-Specifications-49916. [Accessed Mar. 06, 2025].

[3] “STM32H7 - Arm Cortex-M7 and Cortex-M4 MCUs (480 MHz) - STMicroelectronics,” *www.st.com*. https://www.st.com/en/microcontrollers-microprocessors/stm32h7-series.html. [Accessed Feb.10, 2025].

[4] M. Falch, “CAN Bus Explained - A Simple Intro (2025),” CSS Electronics, Feb. 2025. https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial. [Accessed Mar. 06, 2025].

[5] Texas Instruments, “TCAN1044-Q1 Automotive Fault-Protected CAN FD Transceiver With 1.8V I/O Support,” Aug. 2019 [Rev. 2025]. https://www.ti.com/lit/ds/symlink/tcan1044-q1.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1743650743120&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Ftcan1044-q1.

[6] Analog Devices, “isoSPI Isolated Communications Interface,” LTC6820 datasheet, May 2019 [Rev. C]. https://www.analog.com/media/en/technical-documentation/data-sheets/LTC6820.pdf. [Accessed Feb. 11, 2025].

[7] Diodes Incorporated, “60V n-Channel Enhancement Mode MOSFET,” DMN67D7L datasheet, Oct. 2017 [Rev. 2]. https://www.diodes.com//assets/Datasheets/DMN67D7L.pdf. [Accessed Mar. 06, 2025].

[8] Fairchild, “FQP30N06L 60V LOGIC N-Channel MOSFET,” May 2002 [Rev. A1] https://cdn.sparkfun.com/datasheets/Components/General/FQP30N06L.pdf.

[9] Analog Devices, “16-Channel Multicell Battery Monitor,” ADBMS6830 datasheet, Dec.

2022 [Rev. SpD]. https://www.analog.com/en/products/adbms6830.html. [Accessed Feb. 11, 2025].

[10] "NTCALUG54A103G351A | DigiKey Electronics," *DigiKey Electronics*, 2025. https://www.digikey.com/en/products/detail/vishay-beyschlag-draloric-bc-components/NTCALUG54A103G351A/10379532?s=N4IgTCBcDaIHIBUDCBBAMgVQOIFYAsKAjAAwDMWpOhKIAugL5A. [Accessed Feb. 11, 2025].

[11] Texas Instruments, "LM2592HV SIMPLE SWITCHER® Power Converter 150-kHz 2-A Step-Down Voltage Regulator," May 2019 [Rev. 2016]. https://www.ti.com/lit/ds/symlink/lm2592hv.pdf?ts=1743644621537&ref_url=https%253A%252F%252Fwww.mouser.com%252F

[12] "Division of Research Safety | Illinois," *drs.illinois.edu*. https://drs.illinois.edu/Page/SafetyLibrary/BatterySafety. [Accessed Feb. 11, 2025].

[13] IEEE, "IEEE Code of Ethics," *ieee.org*, Jun. 2020. https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed Feb. 11, 2025].