Programmable Battery Management System

by

Jason Erbert

Senior Project

Electrical Engineering Department

California Polytechnic State University San Luis Obispo

December 2020

Table of Contents

List of Tables…….……..…….....iii

List of Figures...……..…….....iii

Abstract……..…….iv

Chapters

1. Introduction……...…….....1
2. Customer Needs, Requirements, and Specifications……...……...5

    2.1 Customer Needs Assessment…..……...……..……...5

    2.2 Requirements and Specifications…….……...……...6
3. Functional Decomposition……...…….....9

    3.1 Level 0 Decomposition…….……..……...9

    3.2 Level 1 Decomposition……...…….10

    3.3 System Level Diagram……...……..12
4. Project Planning……...……..13

    4.1 Timeline…….…..……13

    4.2 Cost Estimation……...……17
5. Design, Build, Evaluate…….....…….....19

    5.1 Battery Construction…….....…….....19

    5.2 Arduino LTC6804 BMS PCB…….....…….……....20

5.3 Charge/Discharge Cutoff Circuitry…….....…….....26

5.4 Cell Balancing…….....29

5.5 Current Sensor…….....…….....…….....32

5.6 Temperature Sensor…….....…….....…….....34

5.7 Graphical User Interface…….....…….....37

5.8 Arduino Code …….....…….....….....39

5.9 Enclosure ……..…….....…….....42

5.10 Design Troubleshooting and Evaluation……..…….....…….43

6. Conclusions……..…….....……...…….....48

References……..…….....……....…….....49

Appendices

A. ABET Senior Project Analysis.……..……..52

B. Arduino Code…….……..63

C. Bill of Materials…….…….……..75

List of Tables

I. Summary of Lithium-Based Batteries……...….2
II. Types of Battery Management Systems and Functions……...…..3
III. Programmable BMS Requirements and Specifications……7
IV. Level 0 Block Diagram Description…….10
V. Level 1 Block Diagram Description – User Interface……..…11
VI. Level 1 Block Diagram Description – Battery Control Unit……..…..11
VII. Level 1 Block Diagram Description – Microcontroller…….…..11
VIII. Programmable BMS Milestones and Deliverables…….….16
IX. Predicted Parts and Labor Costs……..….17
X. Actual Development Cost…….…...18
XI. Cell Balancing Design Tradeoffs…….....30
XII. ACS712 Current Sensor Characterization…….…….....34
XIII. MCP9700 Temperature Sensor Characterization…….…….....36
XIV. Component Dimensions…….....42
XV. System Evaluation…….…….…….…….45

List of Figures

1. Level 0 Block Diagram……9
2. Level 1 Block Diagram…….10
3. System Level Diagram…….…….....12
4. Gantt Charts
   a. Predicted Timeline…….……..14
   b. Mid-Project Timeline Update…….……...……..15
   c. Final Project Timeline…….…….16
5. Battery Construction...……...……19
6. Arduino LTC6804 BMS Circuit Diagram. …….……..22
7. Arduino LTC6804 BMS PCB ViewMate..……..…….…….……23
8. Arduino LTC6804 BMS Shield. …...…….…….…….…….…24
9. Wiring modification to Arduino MEGA2560…...…….…….…….…….…26
10. Cutoff Circuit Diagram…...…….…….…….…….……...27
11. Cutoff Circuit Prototype…...…….…….…….…….…….28
12. Cell Balancing from LTC6804 Datasheet…….…….…….…….29
13. LTC6804 Circuit Diagram with External Cell Balancing…….31
14. Modified Cell Balancing PCB…….....…….…….…...32
15. ACS712 Current Sensor…….....…….…….…….…….…...33
16. MCP9700 Temperature Sensor…….....…….……....35
17. MCP9700 Characterization…….....…….…….…….37
18. Graphical User Interface Page One……39
19. Graphical User Interface Page Two…….…….…….39
20. PBMS V1.0 Software Flowchart…….…….…….40
21. PBMS V1.0 Hardware Setup…….…….43

Abstract

Lithium batteries provide excellent energy storage capabilities at a relatively high density; however, precautions must be taken with these high energy devices to ensure safe operation. A battery management system (BMS) provides protection by monitoring cell and pack voltage levels and maintaining them in a specific range. They limit the output current and disable the output in extreme conditions. Most devices in the targeted power range (<1000W) do not allow the user to manipulate the values for maximum current, cut-off voltage, or other limits. This project introduces the Programmable BMS (PBMS), which instead allows the user to select these values through a physical interface. The interface displays measurements including pack voltage and output current, and it reports additional characteristics of interest such as the battery's temperature, state of charge, and cumulative number of charge cycles. This level of access and control permits users to receive the maximum performance and safety from common lithium battery packs.

Chapter 1. Introduction

This section introduces the motivation to develop a programmable battery management system and discusses both previous and current technology. As the current state of the art for consumer battery technology, lithium batteries find use every day in devices across the world due to factors including their energy density and versatility. Lithium battery applications include portable consumer electronics, electric vehicles, and off grid storage systems. In most applications, batteries require monitoring systems that ensure safe operation and prevent catastrophic failure. Lithium batteries sustain damage from improper use and may fail when they reach temperatures above 100°C resulting in an event known as thermal runaway [1][2]. Failure often results in fire and injury. Improper usage includes drawing excessive amounts of current, discharging the battery too far, and overcharging the battery; the values for these safety limits differ for each style and brand of lithium battery. Most lithium ion cells in the 18650 and 21700 form factors operate safely between 2.5V and 4.2V, but each cell design from various manufacturers and chemistries may have a different current rating [3]. Table I on the following page, from Battery University, provides a comparison of various lithium battery chemistries. With the widely varying limits for batteries, each chemistry, form factor, and brand can require unique management to remain safe in most situations. Battery management systems often passively control batteries; simple electronics monitor battery conditions and disable the battery input and output if the operating conditions exceed predetermined limits for that specific model of battery [4].

TABLE I: SUMMARY TABLE OF LITHIUM-BASED BATTERIES [3]

Chemistry	Lithium Cobalt Oxide	Lithium Manganese Oxide	Lithium Nickel Manganese Oxide	Lithium Iron Phosphate	Lithium Nickel Cobalt Aluminum Oxide	Lithium Titanate Oxide
Short form	Li-cobalt	Li-manganese	NMC	Li-phosphate	Li-aluminum	Li-titanate
Abbreviation	$LiCoO_2$ (LCO)	$LiMn_2O_4$ (LMO)	$LiNiMnCoO_2$ (NMC)	$LiFePO_4$ (LFP)	$LiNiCoAlO_2$ (NCA)	Li_2TiO_3 (common) (LTO)
Nominal voltage	3.60V	3.70V (3.80V)	3.60V (3.70V)	3.20, 3.30V	3.60V	2.40V
Full charge	4.20V	4.20V	4.20V (or higher)	3.65V	4.20V	2.85V
Full discharge	3.00V	3.00V	3.00V	2.50V	3.00V	1.80V
Minimal voltage	2.50V	2.50V	2.50V	2.00V	2.50V	1.50V (est.)
Specific Energy	150–200Wh/kg	100–150Wh/kg	150–220Wh/kg	90–120Wh/kg	200-260Wh/kg	70–80Wh/kg
Charge rate	0.7–1C (3h)	0.7–1C (3h)	0.7–1C (3h)	1C (3h)	1C	1C (5C max)
Discharge rate	1C (1h)	1C, 10C possible	1–2C	1C (25C pule)	1C	10C possible
Cycle life (ideal)	500–1000	300–700	1000–2000	1000–2000	500	3,000–7,000
Thermal runaway	150°C (higher when empty)	250°C (higher when empty)	210°C(higher when empty)	270°C (safe at full charge)	150°C (higher when empty)	One of safest Li-ion batteries
Maintenance	Keep cool; store partially charged; prevent full charge cycles, use moderate charge and discharge currents					

In general, battery management systems divide into two groups: digital and analog. Table II below distinguishes the different functions for each type of BMS. A BMS may accomplish any of the following functions: monitor the battery, protect the battery, estimate the battery's state, maximize the performance, and report to users or external systems [4][5]. For lithium batteries, BMS monitor the cell voltage, current flow, and temperature; some systems add to this by balancing the cell voltages to maximize overall pack capacity. This project aims to provide the maximum benefit to the user and, thus, falls under the digital "protector" category. Many chip manufacturers, including Texas Instruments and Analog Devices, offer ICs with an array of available functions that facilitate lithium BMS design [6][7]. While battery management systems are not a new technology, consumer BMS rarely provide direct control of the battery operating limits. A typical BMS restricts the battery to a single cutoff voltage, maximum current, and maximum temperature [3][5][8]. This provides sufficient protection for most consumers due to the added safety relative to an unprotected battery and low cost. This project improves on the basic system by adding external control over these quantities and displaying the operating conditions to the user. Commercial battery management systems that offer similar programmability typically target large scale applications including large electric vehicles [4][9]. Small scale (<1000W) applications rarely feature this technology due to the cost of production relative to the price of the battery. Current options require additional electronics to program the device, and the manufacturers may not include programming instructions or sell directly to consumers [5][9]. Smart lithium battery chargers offer similar advantages to a programmable battery management system; however, this technology has yet to reach common usage and only protects the battery during charging [10].

TABLE II: TYPES OF BATTERY MANAGEMENT SYSTEMS AND FUNCTIONS [4]

		Reports individual cell voltages	Balances the battery	Requests that battery be switched off	Includes switch to turn off battery
Digital: knows where the problem is and by how much	Protector	✓	✓	✓	✓
	Balancer	✓	✓	✓	
	Monitor	✓		✓	
	Meter	✓			
Analog: knows there's a problem, but doesn't know where and by how much	Protector		✓	✓	✓
	Balancer		✓	✓	
	Monitor			✓	
	Regulator		✓		

Battery longevity is a vital aspect of widespread electric vehicle adoption. While lithium batteries can store large amounts of energy, this capability degrades as the battery charges and discharges over time. In addition, the fast discharge rates of electric vehicles hasten the degradation process. To counteract this and extend the lifetime of the battery, it must operate within a narrower voltage range. For example, instead of charging a battery to 100% and

discharging it down to 0%, it may last up to twice as long if only charged to 80% and discharged to 20% [11]. Increased longevity helps reduce waste and environmental effects caused by battery production [12]. This requires calculation of the battery's SOC, or state of charge, which estimates the remaining battery capacity relative to its maximum. The simplest method measures the open circuit voltage of the battery and compares this to the general operating range of the battery. The equation below demonstrates the relationship between the SOC and open circuit voltage [13].

$$V_{OC}(t) = a_1 \times SOC(t) + a_0$$

In this equation, a_0 represents the battery voltage at 0% SOC, and a_1 represents a constant found by plugging in the open circuit voltage at 100% SOC. For example, a lithium battery ranging from 2.5V to 4.2V contains approximately 60% of its maximum capacity when the voltage measures 3.5V. However, the nonlinear relationship of the battery's capacity to its voltage introduces error near the bottom and top of the operating range. Additional methods, such as Coulomb counting, attempt to improve this by measuring the amount of charge entering and exiting the battery [13]. This technology provides convincing advantages to lithium battery users who value safe and controlled battery use; the following chapter considers the factors driving demand for a PBMS and the expected capabilities of the device.

Chapter 2. Customer Needs, Requirements, and Specifications

This chapter evaluates the needs and expectations of users of small scale lithium battery systems. The PBMS must meet the requirements of common devices at this scale while focusing on user safety. A brief list provides the device's specifications and justification for each. The chapter then discusses how the device meets the specifications and how later iterations may be improved to achieve additional specifications.

2.1 Customer Needs Assessment

Lithium batteries provide enormous amounts of energy and require simple and effective control for safe, efficient usage. Most lithium battery packs in the targeted power range utilize primitive battery management systems; they always charge the battery to its maximum capacity and allow it to discharge down to a static, predetermined limit [8]. Additionally, these battery management systems fail to provide the user with measurements or feedback. Many small electric vehicle operators realize the limitations of these systems and seek additional capabilities; internet forums of EV enthusiasts who build their own batteries indicate a present need for a low cost, programmable BMS. A knowledgeable user who wishes to build and maintain a long-lasting high performance battery, while protecting against the associated hazards, requires a device that adds capability to the simple passive systems by offering access to battery operating conditions and limits. The variable parameters provide adaptability that most primitive BMS lack; for example, instead of requiring individuals to choose a BMS that matches their required output current limit, any battery under the maximum current capability of the PBMS may use the same PBMS. In addition, the device enables users to actively monitor the battery conditions during operation, including cell voltage, pack voltage, output current, and temperature. This provides advantages in many situations; for example, if one cell bank shows a consistently different voltage than the others, the overall capacity and utility of the pack decreases. The live measurements allow the user to observe this directly and replace whichever cell bank causes issues without needing to replace the entire pack.

Lithium batteries pose an inherent risk to both the users and public due to their high energy density relative to other battery chemistries; however, failures rarely occur (on the order of one part per million) when following all safety standards. Ignoring safety precautions may cause batteries to hiss, bulge, or leak before catching fire and even exploding. A failure can cause third degree burns and permanent injury to anyone within a few feet of the device [3]. Thus, battery safety remains a key factor for all users, especially when considering lithium-based chemistries. The PBMS prioritizes safety by educating the operator about lithium batteries and allowing the user to determine the operating limits they deem safe. It responds to potentially catastrophic

events much faster than any person can react. The following section discusses project requirements that provide the user with the best and safest experience for devices in the targeted power range.

2.2 Requirements and Specifications

The primary objective of a BMS includes monitoring battery characteristics and triggering an immediate shutoff when reaching predefined limits that indicate a dangerous fault. BMS include protection from faults including overvoltage (OVP), undervoltage (UVP), overcurrent (OCP), and overtemperature (OTP). A programmable BMS expands on the commonplace BMS by adapting to meet the needs of each unique user. The requirements of the programmable BMS center around providing the user with the most control over the battery while maximizing safety. Table III lists and justifies each PBMS specification. It also describes the marketing requirements of the PBMS; a successful product provides tangible benefits to the consumer. Each engineering specification listed in the table targets at least one marketing requirement. In short, the marketing requirements dictate that a successful BMS covers a wide range of devices, provides a simple user experience, maintains high efficiency, improves the experience of the user, and encourages sustainable practices.

Since consumer lithium batteries range in power from milliwatts to many kilowatts, the project targets a common range for users who gain the most by the added capabilities of the PBMS: ~100W to ~1000W (peak). With pack capacity a priority, low power consumption provides the maximum benefit to the user. This project defines low power consumption as utilizing less than 1% of the maximum allowable wattage of the device; for example, the device consumes less than 5W maximum for a 500W battery. A power heavy user interface including an LCD and microcontroller draws no more than 500mA and 5V, and the electronics performing battery control and measurements draw small amounts of power relative to the size of the battery [6]. Efficiency limits the device's utility at smaller scales – interface electronics consume relatively low power and remain constant as the product scales to larger power capabilities.
Common voltage and current ratings for battery packs in this range span from 11V up to 48V and from 5A to over 30A. The initial estimate of a maximum output voltage of 27V originates from the Texas Instruments BQ76PL536, used in similar senior projects at Cal Poly, [7][10]. Further research and evaluation leads to new and improved options with a larger voltage range thus enhancing versatility. The selected range balances simple electronic design and safety versus the number of applicable devices. The well documented and common LTC6804 BMS meets the voltage requirements for the targeted range of devices [6]. In addition, the LTC6804 features efficient, passive cell balancing. Every cell bank requires voltage balancing for even discharge, maximum capacity, battery longevity, and optimal safety [11]. This chip also includes extensive documentation from the manufacturer and use by lithium BMS expert Davide Andrea, author of "Battery Management Systems for Large Lithium Ion Battery Packs" and experienced designer of lithium BMS since 2004 [4].

TABLE III: PROGRAMMABLE BATTERY MANAGEMENT SYSTEM REQUIREMENTS AND SPECIFICATIONS

Marketing Requirements	Engineering Specifications	Justification
1,5	Allows output voltage from 11 - 50V (with adjustable OVP and UVP for 3-12 cell banks in series)	This range applies to a large variety of consumer scale lithium battery powered systems. It helps prioritize safety during development, facilitates part acquisition, and reduces complexity.
1,5,6	Measures output current up to 20A and provides adjustable OCP limit	A 20A overcurrent protection (OCP) limit simplifies electronics/heat dissipation design while remaining safe for low wattage batteries. This stems from the maximum current of many common 18650 cells, like the Samsung 25R used in this project [14].
1,5,6	Balances individual cell bank voltages within 2-3% of each other using passive balancing	Cells range 2.5 – 4.2V; 3% = 75 – 126 mV, achievable with LTC6804. Passive balancing simplifies design and lowers cost of development and consumer product.
2,6,7	Actively displays pack and cell voltage (± 100mV), output current (± 1A), temperature (± 5°C), state of charge, and number of cycles on user interface	These vital characteristics inform the user about the safety of a lithium battery's operating conditions. The user must have access to the measured characteristics and limits during operation.
6	Monitors pack temperature within 5°C; adjustable maximum operating temperature	Lithium battery heat sensitivity requires active temperature monitoring. After reaching the tipping point, thermal runaway causes dangerous reactions.
6	Responds to fault event in under 1 second	Reacts faster than humans thus increasing safety. Similar systems show similar or slower response times [8].
3	Active power consumption under 5W	"Active" means user interacts with device interface by changing settings or observing measurements. The 5W estimation minimizes impact on battery life and comes from simulating components and analyzing competition [9][5].
1,4	Physical dimensions smaller than 1"x4"x4"	The BMS must not significantly impact the overall dimensions of the battery. Lithium batteries at low (<1000W) wattages significantly exceed this size.

1,2,3,4,7	First time setup under 1 hour	Simple configuration procedure allows implementation with minimal setup time for a user with moderate electronics experience (<1 year soldering); also allows user time to learn interface and educates on lithium battery technology.
1,2	Settings changes require less than 5 minutes of user interaction	Low voltage cut-off or maximum output current are accessible without significantly interrupting battery usage. Simple and intuitive interface increases ease of use.
5	Cost of prototype unit below $500	Can scale down to competitive and consumer friendly price with higher volume. Estimated using typical component prices (see Appendix C for BOM).
Marketing Requirements 1. Versatile and easy to implement 2. Simple and intuitive interface 3. Low power consumption 4. Compact and portable 5. Low cost 6. Increased utility and safety from lithium batteries 7. Encourage sustainability		

The device must feature similar size and form to current BMS technology; the shape of the BMS should not significantly increase the overall size of the battery pack. Typical BMS have a minimal effect on overall battery dimensions [5]. The small scale focus of the device's design helps minimize the risk of injury during the project, but it also restricts the overall dimensions of the product which adds complexity to the design process and difficulty during prototyping.
To appeal to a large demographic, the PBMS features a simple integration experience and user-friendly interface that requires less than 5 minutes to change device settings. The PBMS provides operating conditions and up-to-date measurements of voltage, current, temperature, and more at any point during operation to ensure constant safety. It also allows the users to change the operating limits to meet their needs. A logically organized interface with a gentle learning curve minimizes device downtime during settings changes.

The requirements and specifications table format derives from [15], Chapter 3. The specifications listed in the table provide a usable balance between device compatibility, development safety, design simplicity, and cost of production. Many specifications allow room for later improvement and refinement after proving the concept of a truly programmable BMS in this power range. The follow chapter details the system from a top down approach by providing multiple block diagrams and explanations of internal subsystems.

Chapter 3. Functional Block Diagrams

This section breaks down the programmable battery management system into functional blocks at various levels of detail. The first level looks solely at the inputs and outputs of the system. The next level separates the design into the three subsystems; user interface, central processing unit, and battery management. Tables V-VII list the inputs and outputs of each subsystem. Finally, a system level diagram including all components and subsystems demonstrates internal connections and organization.

3.1 Level 0 Decomposition

Figure 1 below offers a top-level block diagram of the PBMS including the main inputs and outputs described in Table IV. The battery pack consists of several individual cell banks in series. The BMS measures the voltage of each cell bank, current entering or exiting the battery, and pack temperature. It accepts user settings to determine maximum operating limits and outputs the protected power from the battery. The PBMS also provides real time measurements and data on a graphical interface.

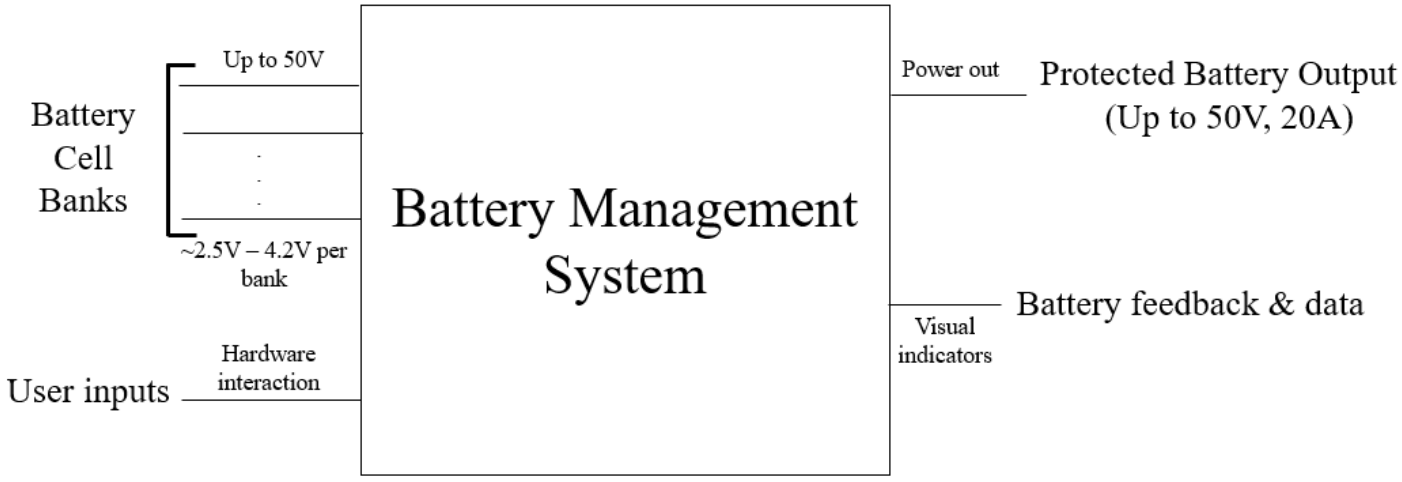

Figure 1: Level 0 block diagram for battery management system

TABLE IV
PRROGRAMMABLE BATTERY MANAGEMENT SYSTEM LEVEL 0 INPUTS, OUTPUTS, AND FUNCTIONALITY DESCRIPTION

Inputs	• Battery power from cell banks (up to 50V, up to 20A) • User selected settings including maximum output current, minimum battery cell voltage, maximum operating temperature
Outputs	• Allows up to 50V at up to 20A (user specified) • "Battery feedback & data": Battery characteristics and measurements including operating voltage, cell bank voltage, output current, battery state of charge, and present settings for OVP, UVP, OCP, and OTP.
Functionality	The battery management system balances individual cell voltages; monitors, reports in real time, and limits the battery pack's overall operating voltage, current, and temperature; and offers user control over input/output cutoff limits.

3.2 Level 1 Decomposition

Figure 2 on the following page shows the internal modules of the PBMS design. The subcomponents fall under three main categories: user interface hardware, battery characteristic monitor/control unit, and the microcontroller. The functional breakdown derives from the block diagram in Cheng [16] Figure 1. Tables V, VI, and VII describe the functionality of each category and the associated inputs/outputs.

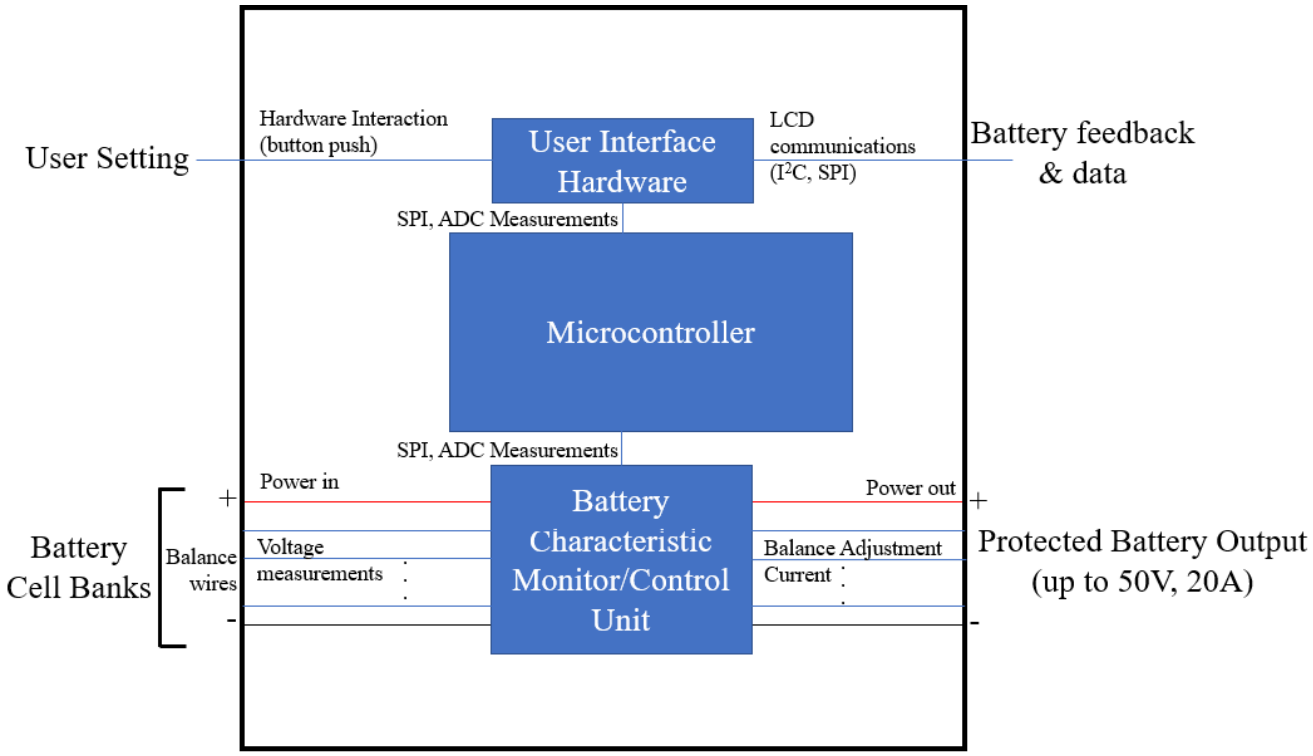

Figure 2: Level 1 Block Diagram of Programmable Battery Management System

TABLE V:
PBMS LEVEL 1 BLOCK DIAGRAM DESCRIPTION – USER INTERFACE HARDWARE

Inputs	• Hardware interface featuring button pushes on touchscreen LCD
Outputs	• User specified setting – maximum charge voltage, maximum output current, among others to microcontroller for interpretation. • LCD data to display – user settings, user interface, battery feedback & data
Functionality	The user interface offers direct control over system settings. The microcontroller interprets the input and alters system settings accordingly. Internal storage maintains user settings, which the user may view and access at any point during device operation on the included screen. The interface displays real time battery measurements including voltage and output current.

TABLE VI:
PBMS LEVEL 1 BLOCK DIAGRAM DESCRIPTION – BATTERY CHARACTERISTIC MONITOR/CONTROL UNIT

Inputs	• Battery cell bank voltages • Cell temperature • Input/output current
Outputs	• Battery measurements to microcontroller for processing
Functionality	The battery control unit consists of several subcomponents that ensure the lithium battery operates safely. Main components include cell balancing module, current monitor, and temperature monitor. Contains high power input/output cutoff circuitry. The LTC6804 provides simple cell balancing options and voltage readings [6]. It communicates directly with the microcontroller using a 4-wire SPI interface

TABLE VII:
PBMS LEVEL 1 BLOCK DIAGRAM DESCRIPTION – MICROCONTROLLER

Inputs	• User inputs from physical interface • Battery measurements from battery characteristic monitor
Outputs	• Battery measurements to display on interface • Internal settings to display • Control signals to battery monitor
Functionality	The microcontroller obtains battery measurements from the battery control unit and alters internal settings accordingly. Signals from the microcontroller may initiate a current cutoff. The microcontroller contains a simple user interface with a display. User settings adjust the battery output and operating conditions.

3.3 System Level Diagram

Figure 3 below illustrates an entire lithium battery system built around the programmable battery management system. The battery monitor unit from Figure 2 comprises the LTC6804 PCB, cell balancing PCB, cutoff circuit, current sensor, and temperature sensor all pictured in Figure 3. The user interface hardware consists exclusively of the Nextion 4.3” Touchscreen LCD. These peripherals all communicate with the Arduino MEGA2560, which serves as the central processing unit of the battery management system. The battery powers the Arduino through a DC-DC converter. The Arduino provides power to the LCD, current sensor, and temperature sensor through its 5V supply. It communicates with the LTC6804 using SPI and the LCD using UART; the current sensor and temperature sensor provide analog voltages to pins A8 and A9 on the Arduino. The Arduino also provides a digital (TTL) signal to the cutoff circuit to disable the battery’s input and output using pin D26.

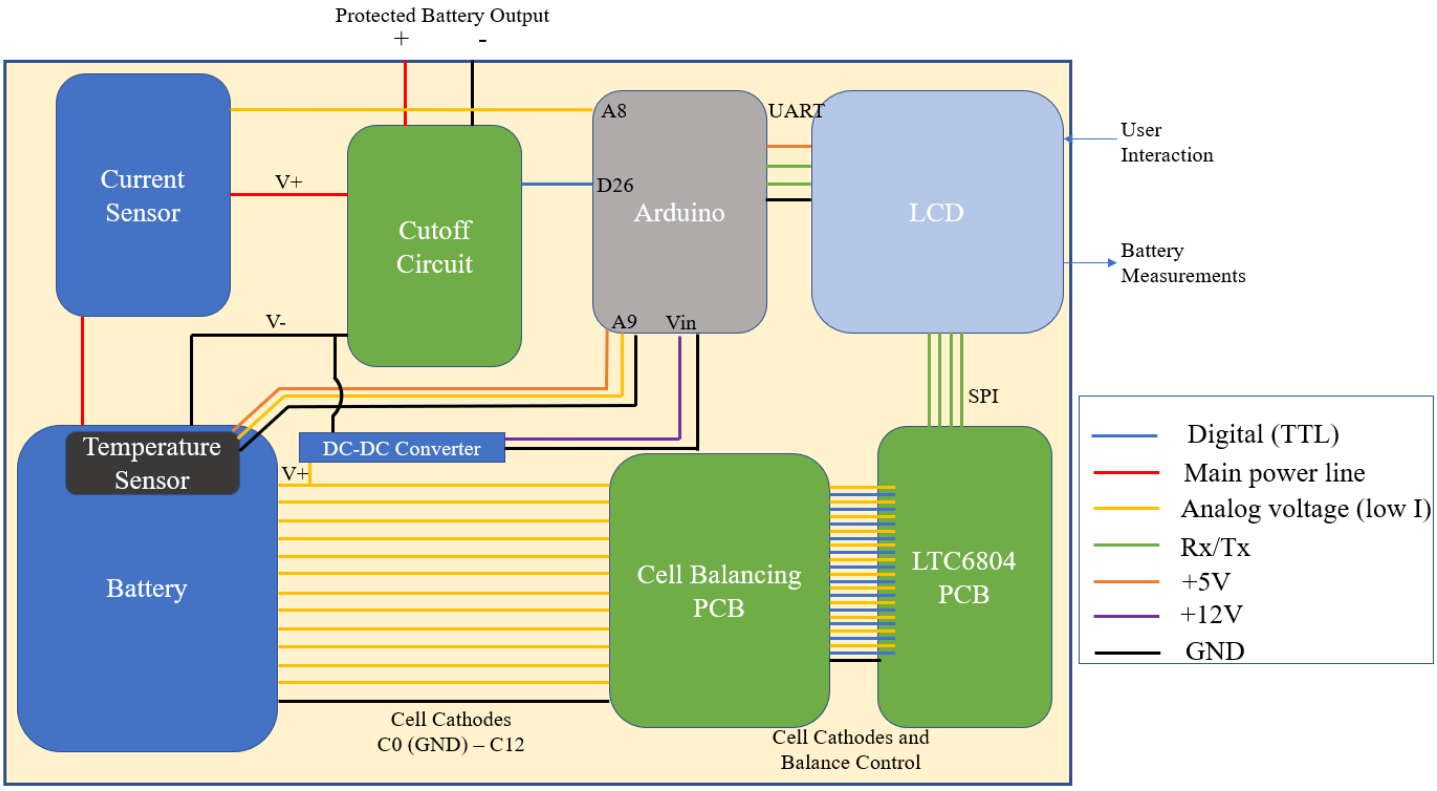

Figure 3: System diagram containing all components and subsystems after integrating with a battery

The following chapter details the project planning, milestones, and costs.

Chapter 4. Project Planning

This chapter explains the planning process for the BMS development. First, a Gantt chart showing the initial timeline predictions helps estimate how much time the project requires and when to expect deliverables. Then, an updated Gantt chart illustrates the actual timeline of the project and allows the comparison of expectations versus reality. Finally, Table VIII and IX demonstrate the difference between the projected development cost and actual cost.

4.1 Timeline

The project spans approximately 11 months as demonstrated in the Gantt chart from EE460 in Figure 4a below. The Project Plan phase consists mainly of researching a chosen topic and submitting an outline of a possible project, as part of EE460. Time allocation must initially allow at least two design, build, test iterations to optimize device performance. The first design iteration involves lower level system design; the process of building a system from scratch provides excellent learning opportunities absent from top level approaches. The second design makes use of knowledge gained during the first iteration and considers the benefits of a top-down approach. The expected time spent on a senior project varies from 180 – 230 hours per student. The weekly time requirement generally varies from 4-6 hours throughout EE460, EE461, and EE462. This approximation derives from a simple calculation: 180 hours divided into three quarters, with 10 weeks each quarter, results in 6 hours per week not including summer.

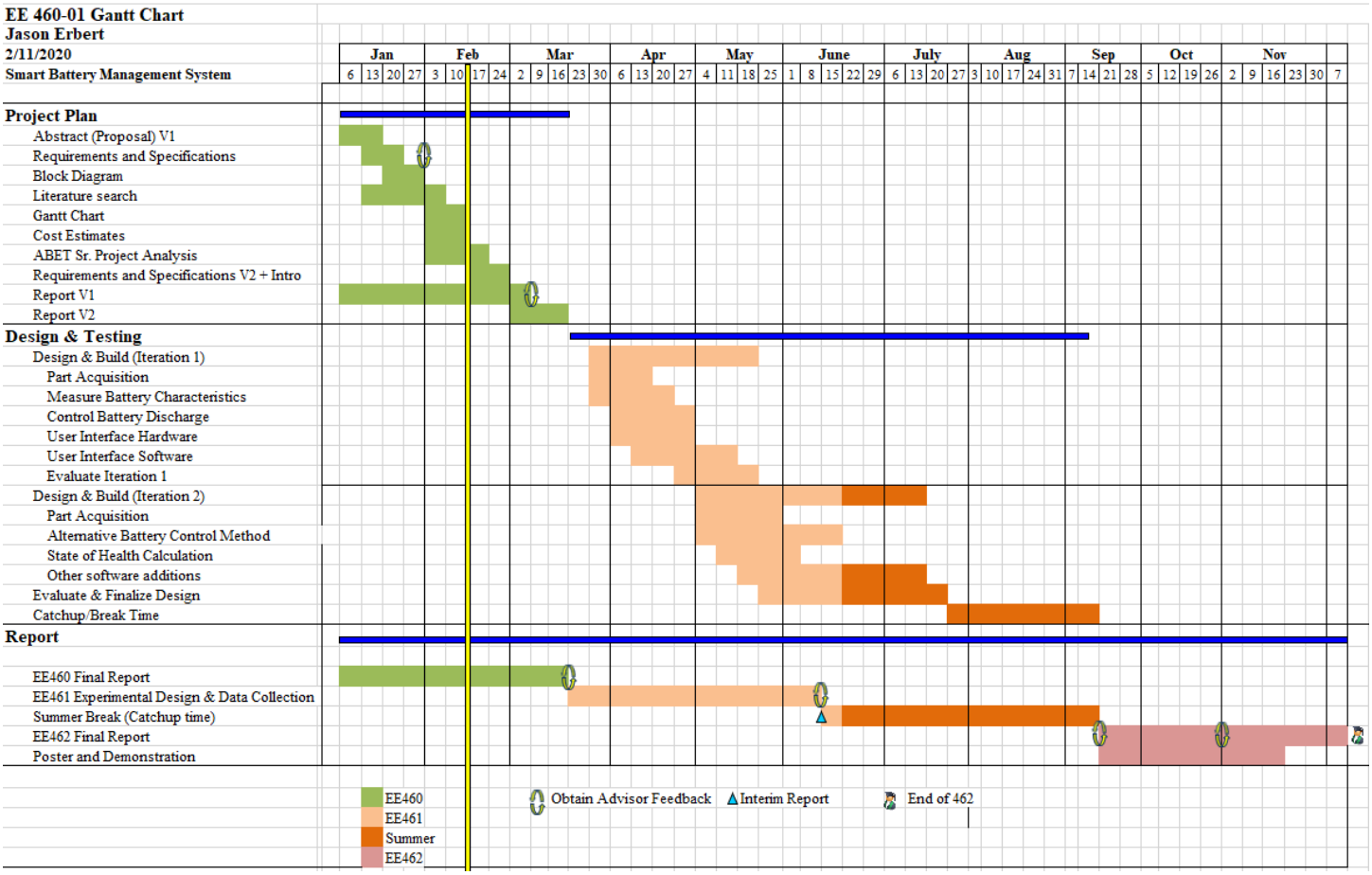

Figure 4a: Preliminary Gantt chart for programmable BMS design and documentation process.

EE461 and EE462 mainly include additional researching, designing, building, and evaluating the design. Figures 4b and 4c on the following pages demonstrate the updated project timeline at each stage of the development process. The initial predictions utilize the summer quarter gap between EE461 and EE462 to provide additional time to complete development as needed. However, as the COVID-19 global pandemic developed during the first and second phases of the project, unforeseen circumstances continue to require constant reevaluation of the available time for development. The loss of development time during summer increases the required weekly time requirement to at least 6 hours during EE462. The project's actual timeline in EE461 trails the predicted timeline by approximately four weeks; part acquisition and unforeseen additional steps, including circuit design and PCB assembly, require at least twice the initial allotted time.

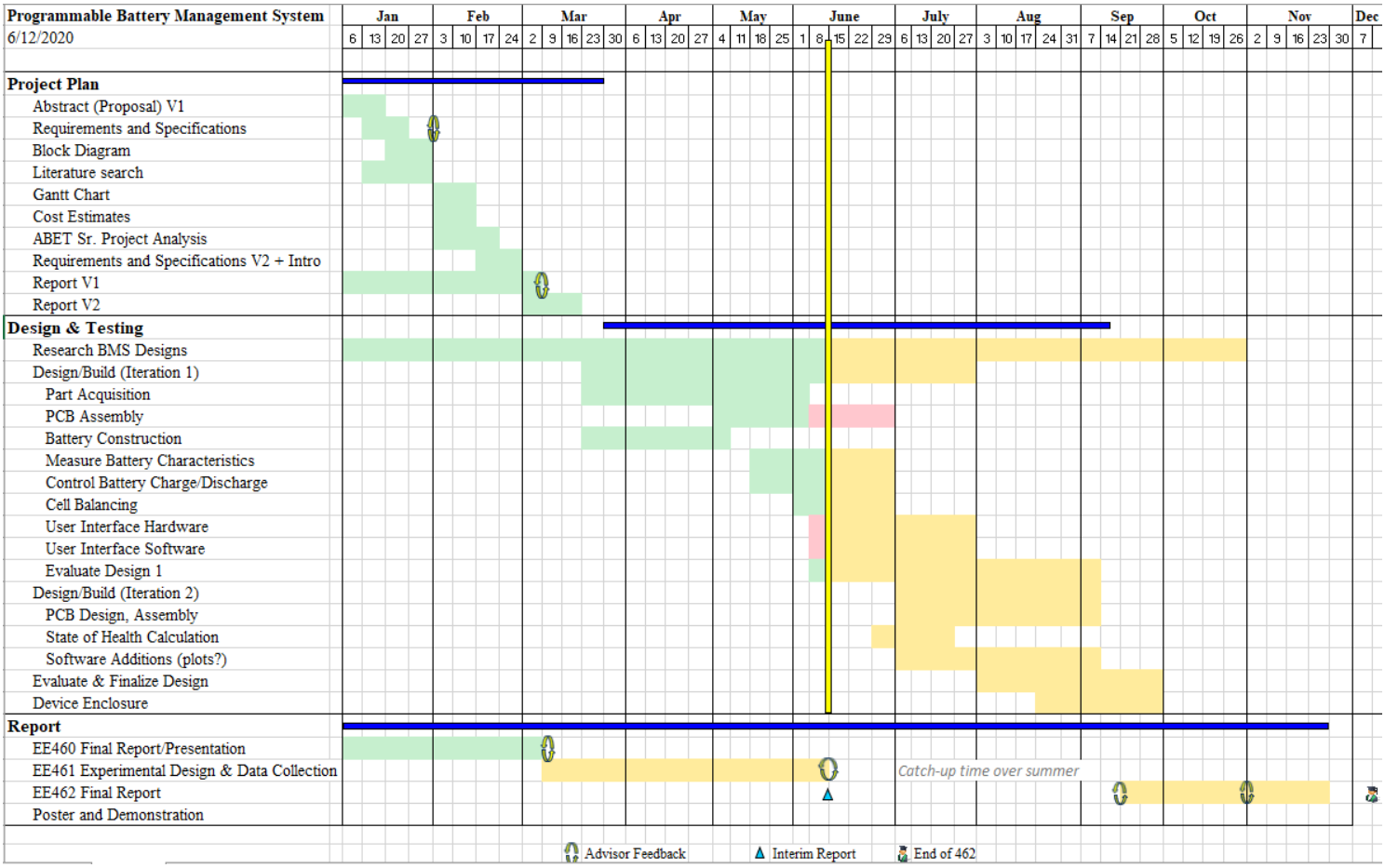

Figure 4b: Gantt chart for programmable battery management system design and documentation process from EE461. Green bars signify completion; red indicates the specified objective is off track; yellow indicates work progressing.

Additionally, the Gantt chart in Figure 4c includes only one design iteration due to time constraints. While building and evaluating two designs provides the best result, this requires two complete designs to evaluate. Near the beginning of EE462, the project concentrates on completing a single functioning system that accomplishes the requirements. The knowledge gained from working with each subsystem increases the educational value of the project versus an approach that incorporates an off the shelf BMS. Creating a complete system from scratch that meets the minimum specifications provides a better foundation for future development and improvements like those listed in the initial Gantt charts.

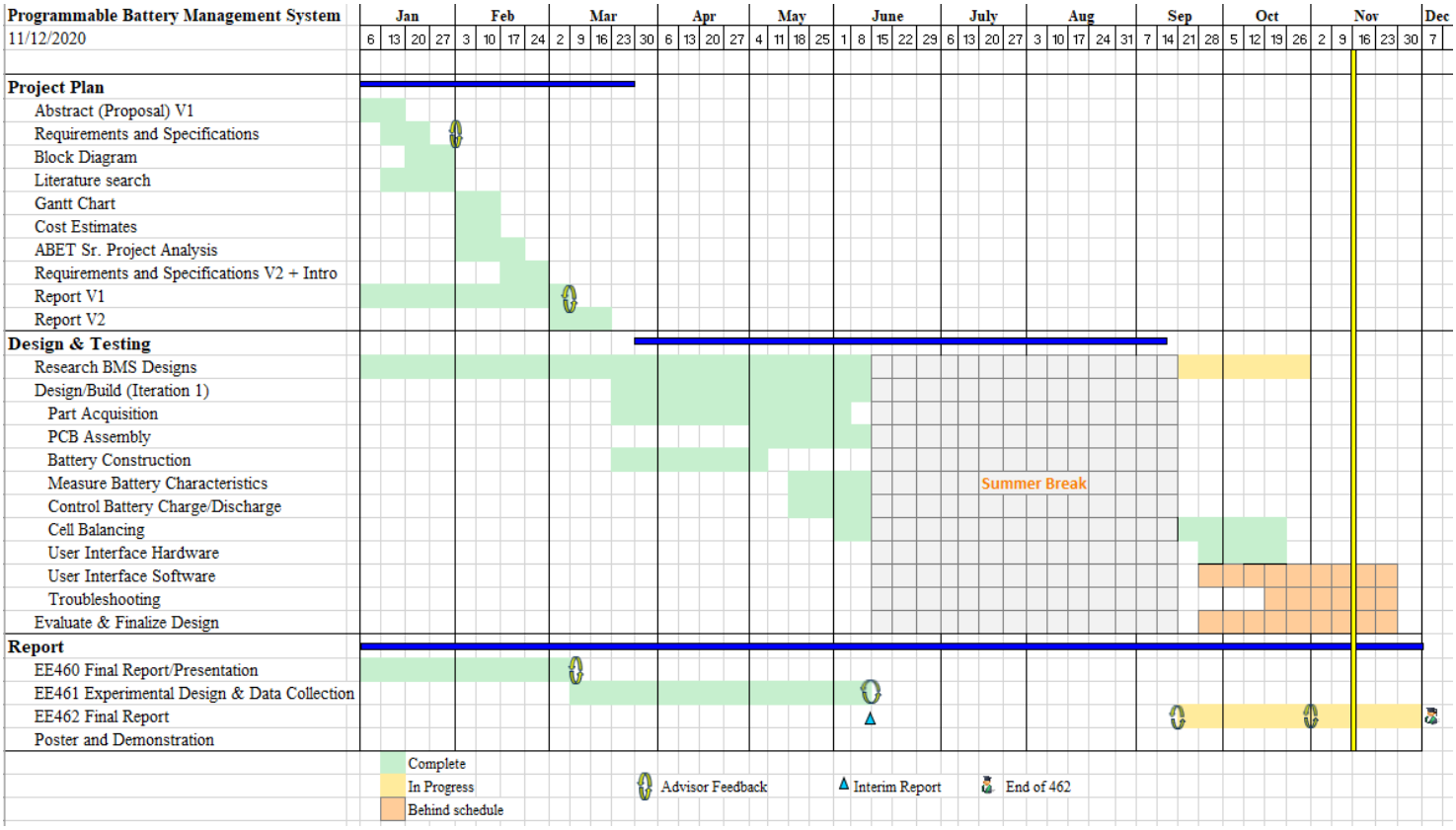

Figure 4c: Actual Gantt chart update near the end of EE462

Table VIII below lists major milestones in the project reporting process. Expected project completion occurs approximately one month before the end of EE462 according to initial estimates. Various setbacks throughout the course of the project delay project completion, resulting in a final demonstration that includes various subsystems but fails to meet some requirements.

TABLE VIII:
PROGRAMMABLE BATTERY MANAGEMENT SYSTEM DELIVERABLES

Delivery Date	Deliverable Description
04/17/2020	EE 461 Design Review
05/29/2020	EE 461 Demonstration
06/12/2020	EE 461 report
09/15/2020	Project Status Update
12/02/2020	EE 462 Demonstration
12/04/2020	ABET Sr. Project Analysis
12/04/2020	EE 462 Report

4.2 Cost Estimation

Most costs associated with BMS production include obtaining the hardware (excluding labor). High quality batteries and control circuitry demand higher prices but provide maximum safety and benefit to the user and public. Development costs occur predominately in Spring 2020 and extend to Fall 2020.

The total predicted cost for this project including labor should not exceed $4,870 as demonstrated in Table IX below. The Gantt chart in Figure 4a and the calculation described earlier provide the approximate time allocation. A fair hourly rate for a fourth-year engineering intern in California averages $30 per hour and is the figure used to estimate labor costs. Individual component costs may vary greatly from the initial prediction, but a balance of conservative and exaggerated values provide an accurate overall cost estimation. For example, the interface hardware may consist only of a $10 LCD touchscreen whereas the battery management modules may include costly systems and fabrication of printed circuit boards. Inexpensive resistors, capacitors, and most necessary passive components likely sum to less than $20. Potential interface hardware components including potentiometer knobs and LCDs average $10; a simple interface requires no more than ten components. The BMS also requires internal modules like a cell balancing circuit, which should generally cost about $10. A basic BMS like the one in Cheng [16] Figure 1 can be built using 10 of these modules or fewer. A microcontroller contains built in modules and offers the ability to develop solutions using software instead of additional hardware. Assembly of a complete system requires hardware including wire, solder, battery connectors, batteries, and an enclosure; the cost estimation for these stems from similar projects in previous courses.

TABLE IX: PREDICTED PARTS AND LABOR COSTS

	Price	**Qty**	**Total**
Labor	$30/hr	150 hrs	$4500
Parts			
Resistors, Capacitors, etc	$20	1	$20
Interface Hardware	$10	10	$100
Microcontroller	$25	2	$50
Battery Management Modules	$10	5	$50
Batteries	$10	10	$100
Wires, Connectors	$50	1	$50
Enclosure	$50	1	$50
Project Cost			$4870

The actual costs mostly line up with predictions, although some individual costs far exceed predictions due to unforeseen development challenges. Table X below features a summary of the

true cost of parts and labor. The method of exaggerating some expenses and conservatively estimating others results shows success; while the interface hardware cost approximately half the initial prediction, passive components require over five times the expected capital. Initial estimates assume a complete LTC6804 BMS costs approximately $200 to produce, and additional modifications necessary to meet the requirements increase both development time and parts cost. The cutoff circuitry explained in Chapter 5.3 allow the BMS to meet the current and voltage requirements listed in Chapter 2, but it adds at least 20 hours of labor and $50 in hardware. The Bill of Materials in Appendix C provides a complete list of the hardware prices. The BOM indicates approximately $424 for the total price of the hardware required for a complete PBMS. Development includes ordering additional parts in case prototypes break thus increasing up front cost. In addition, the BOM quotes only a single PCB for both the LTC6804 Arduino BMS and the cell balancing PCB. This results in a lower unit cost than possible for a single board; the true cost for the PCBs increases to approximately $400 total when factoring in the number of boards ordered. The total sum for all development hardware equates to $866.

TABLE X: ACTUAL DEVELOPMENT COST BREAKDOWN

	Price	**Qty**	**Total**
Labor	$30/hr	180 hrs	$5400
Parts			
Batteries	$2.75	40	$110
Res, Caps, FETs, etc	$60	2	$120
LCD Touchscreen	$60	1	$60
Microcontroller	$15	1	$15
Sensors	$10	2	$20
Battery Management Modules	$25	3	$75
LTC6804 BMS PCB	$26	5	$130
Cell Balance PCB	$64	4	$256
Wires, Connectors	$30	1	$30
Miscellaneous Supplies	$10	5	$50
Project Cost			$6266

Most of the increase in the cost of development relative to predictions stems from the labor involved. Design modifications, complications, and other unrelated issues result in at least 30 hours of additional time dedication. This 20% increase raises the total cost of labor by $900 to $5,400. The following chapter details the project design, construction, troubleshooting, and evaluation process.

Chapter 5. Design, Build, and Evaluate

This chapter covers all completed steps of the design phase including battery design and construction, BMS design, circuit assembly, and graphical interface design. Each section includes evaluation plans for the subsystem, and the final section describes the plan for, and results of, testing the BMS as a complete system.

5.1 Battery Construction

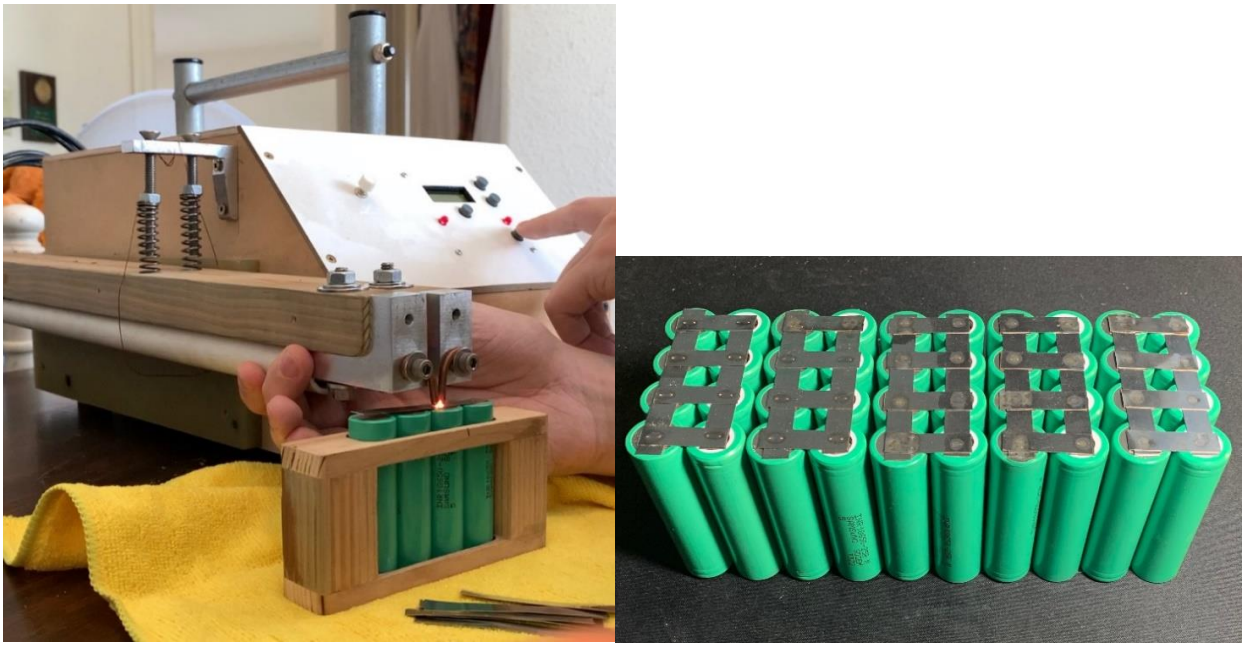

Figure 5: Spot welding one bank of 18650's together with nickel strip (left) and welded battery (right).

Construction begins with an appropriately sized battery in need of management. Batteries intended for small electric vehicle use generally contain 500Wh or lower. This project uses a 36V 10S4P battery with a maximum capacity of 10Ah resulting in a 370Wh rating, which can power a 500W electric bicycle motor. 10S4P indicates ten lithium battery cells in series and four in parallel. The Samsung 25R 18650 lithium ion cells used for construction provide a relatively low capacity, low cost option useful for the scope of this project. Each cell features a nominal voltage rating of 3.6V, with a total range from 2.5V at 0% to 4.2V at 100% [14]. The cells can each receive up to 4 amps of charging current and output up to 20 amps continuously, meaning the pack can theoretically safely receive up to 16 amps and output up to 80 amps. The 2500mAh cell capacity adds in parallel resulting in the 10Ah rating. Pure (99.5% or higher) nickel strips measuring 0.2mm x 8mm provide low contact resistance and high current carrying capability between cells. Spot welding the strips to the cells results in a reliable connection without injecting dangerous amounts of heat into the cells during construction. Each button push on the

welder sends 2 pulses, 100 ms and 50 ms each, between the copper tips; the pulses include hundreds of amps of current and quickly melt the nickel strip to the electrodes of the cells at the contact points. Welding starts by joining cells in parallel before connecting them in series. Ten sets of parallel banks, each containing four 18650s and measuring 3.50V during construction, alternate polarity to create the required voltage as in Figure 5. The white ring indicates the top of the battery cell (positive). All cells are tested for quality before connection; each cell selected for use measures 3.50V $\pm$ 5mV. "Dead" cells measure below 2.5V and reduce the reliability and capacity of a pack [17][18]. Additionally, connecting unbalanced cells in parallel forces the voltage to immediately equalize, driving large amounts of current from one cell to another. As discussed previously, high charge/discharge rates lead to decreased battery longevity and may cause fire due to excessive heat production. Figure 5 above shows the complete battery ready for a management system. After soldering cell balancing wires to the individual cell banks, heat shrink encapsulates the battery and exposed terminals thus preventing accidental shorts. All main power connections from the battery utilize XT60 connectors for their high current capability (60 amps), protection against reverse polarity, and ease of swapping.

According to the specifications list in Chapter 2, the battery falls in the desired power range for the PBMS. The design requires a battery that can supply up to 20 amps at a voltage between 11V and 50V. Measurement across the terminals with a multimeter proves the output voltage of ~36V nominally (34.8V at time of testing). A load test using the BK Precision 8540 150W DC Electronic Load in constant resistance mode shows the battery safely provides at least 3 amps at 11 ohms, which approaches the power limit of the electronic load and proves the battery serves the required purpose for this project.

5.2 Arduino LTC6804 BMS PCB

The PBMS design makes use of the common Linear Technologies LTC6804 battery management system intended for automotive applications. Many experts, including Davide Andrea, utilize the LTC6804 in lithium BMS designs for its expansive feature list, precision, compatibility in daisy chains, and C++ library support from Analog Devices [4]. The inspiration for this project builds on the "Arduino LTC6804 BMS" article written by Instructables.com user David M. Caditz, which provides many opportunities for expansion [19]. The LTC6804 allows up to twelve banks of parallel batteries to connect in series resulting in a maximum output voltage 50.4V, which enables the device to meet the 1000W power requirement assuming a maximum 20 amp current draw. The 6804 measures the voltage of each cell bank using a 16 bit ADC and sends the values to the Arduino using either 2-wire isoSPI or 4-wire SPI up to 1 MHz. The LTC6804 facilitates cell balancing by providing internal circuitry while also supporting external balancing hardware. The specific part number for this project, LTC6804HG-2, includes the "HG" to denote the higher temperature rating relative to the "IG". The "2" distinguishes between the LTC6804-1 (allows connecting multiple BMS in series for higher voltage packs) and LTC6804-2 (allows connecting multiple BMS in parallel).

The Arduino LTC6804 BMS design by Caditz utilizes a shield approach to connect the BMS with the microcontroller. It provides five general purpose input/output ports for external temperature sensors, current sensors, and other peripherals. Filter networks made of resistors and capacitors, suggested on the LTC6804 datasheet, attenuate high frequency noise during operation [6]. Based on the Arduino Uno, Caditz's design lacks the number of inputs and outputs necessary to utilize both the BMS and user interface while allowing future expansion. An Arduino MEGA2560 replaces the Uno due to the increased I/O capacity and shield compatibility. Selected circuit diagram segments shown below in Figure 6 demonstrate the LTC6804 and Arduino connections. The PBMS excludes the relay driver circuit (right) and instead uses pin D26 (not pictured) to control the cutoff circuitry. The PCB designed by Caditz provides advantages and disadvantages compared to designing a PCB from scratch. Starting with a functioning BMS significantly reduces the required development time associated with board layout, component selection, and debugging. However, the board includes connections and circuitry not needed for this project, which increases fabrication cost and troubleshooting difficulty. Modifications to the design necessary to fulfill the requirements, such as the cutoff circuit in Section 5.3, also add unexpected development time.

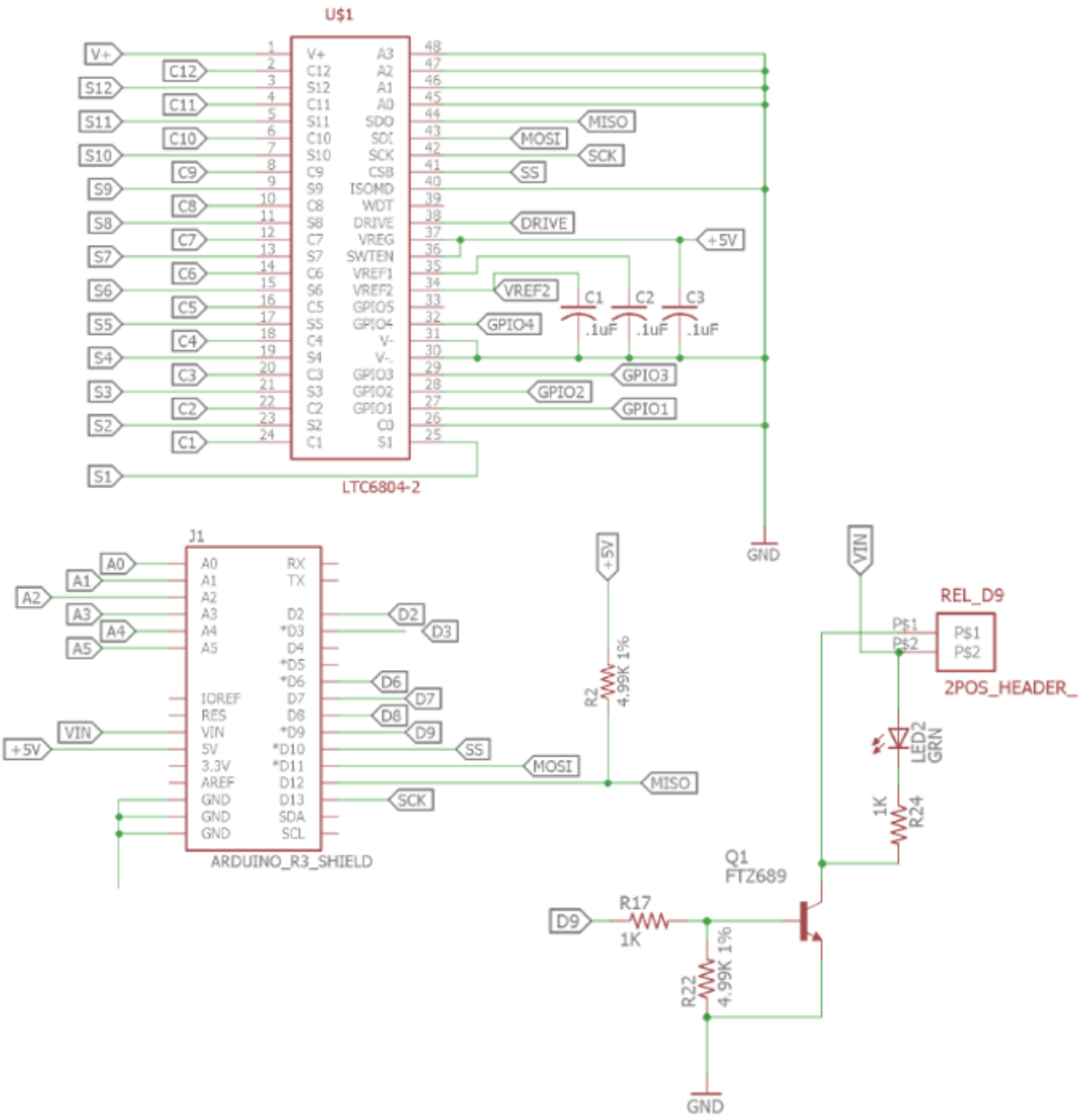

Figure 6: Selected segments of Arduino LTC6804 BMS circuit diagram by Caditz demonstrate connections between the Arduino and LTC6804 [19].

Caditz suggests Sunstone Circuits as the PCB manufacturer. Figure 7 on the following page shows a snapshot from ViewMate with all layers of the PCB. Upon obtaining quotes from competitors, Sunstone provides the best balance of quick turnaround, confidence, and cost. Other PCB manufacturers advertise lower prices but require additional time for fabrication and shipping. The proven history of producing a satisfactory BMS PCB according to the designer distinguishes Sunstone as the best option. Due to the time sensitive nature of the project, faulty PCBs due to manufacturing errors have significant potential to impact the project's success. The Gantt chart in Figure 4a lacks time allocation for delayed PCB acquisition.

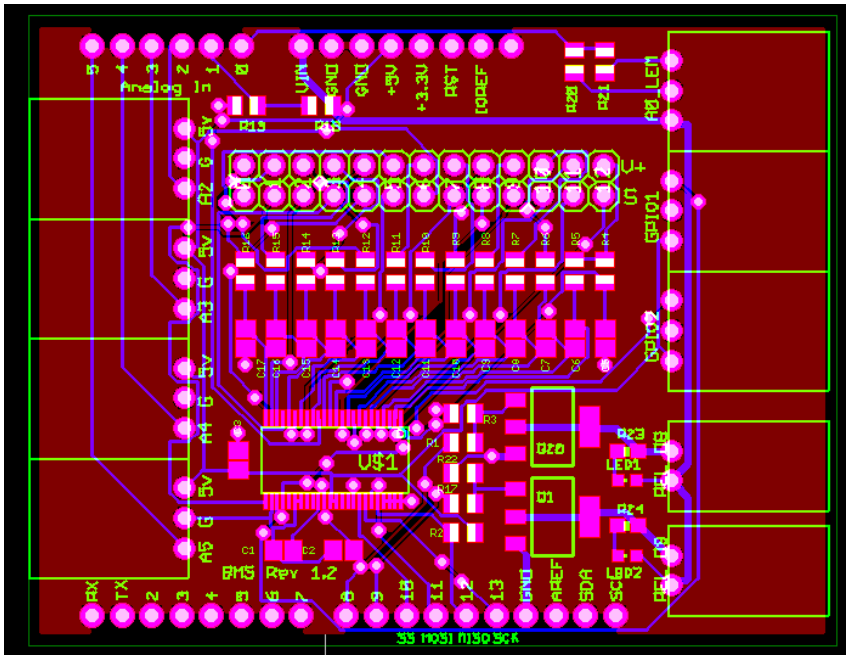

Figure 7: ViewMate snapshot of PCB files designed by Caditz [19]

Budgetary constraints result in manual soldering for all components in this project. Although paying a supplier to fabricate and populate the PCB with components reduces both turnaround time and likelihood of soldering mistakes, the prohibitive cost forces manual assembly. The PCB uses predominantly surface mount components, which complicate the assembly process but decrease overall size relative to through-hole components. The LTC6804 features a 48 pin SSOP package with leads spaced 0.5mm apart; John Gammel's surface mount soldering tutorials provide the methods necessary to solder similarly sized components [20]. Copper braid solder wick helps to remove unintended solder bridges without overheating the sensitive components. Until verifying the LTC6804 functions as intended, the PCB uses the fewest components necessary as shown in Figure 8 below. The LTC6804 requires the battery voltage, V+, to set internal reference voltages according to the block diagram on page 19 of the datasheet. For this reason, construction of the cell balancing circuitry occurs before evaluating the 6804. The LTC6804 also requires three bypass capacitors between Vref1, Vref2, Vreg, and ground. A 4.99k pullup resistor between MISO and ground permits the device to communicate using SPI. Connection of the ISOMD pin to the V- pin sets the device to use traditional 4-wire SPI instead of 2-wire isoSPI [6], which filters out communication noise.

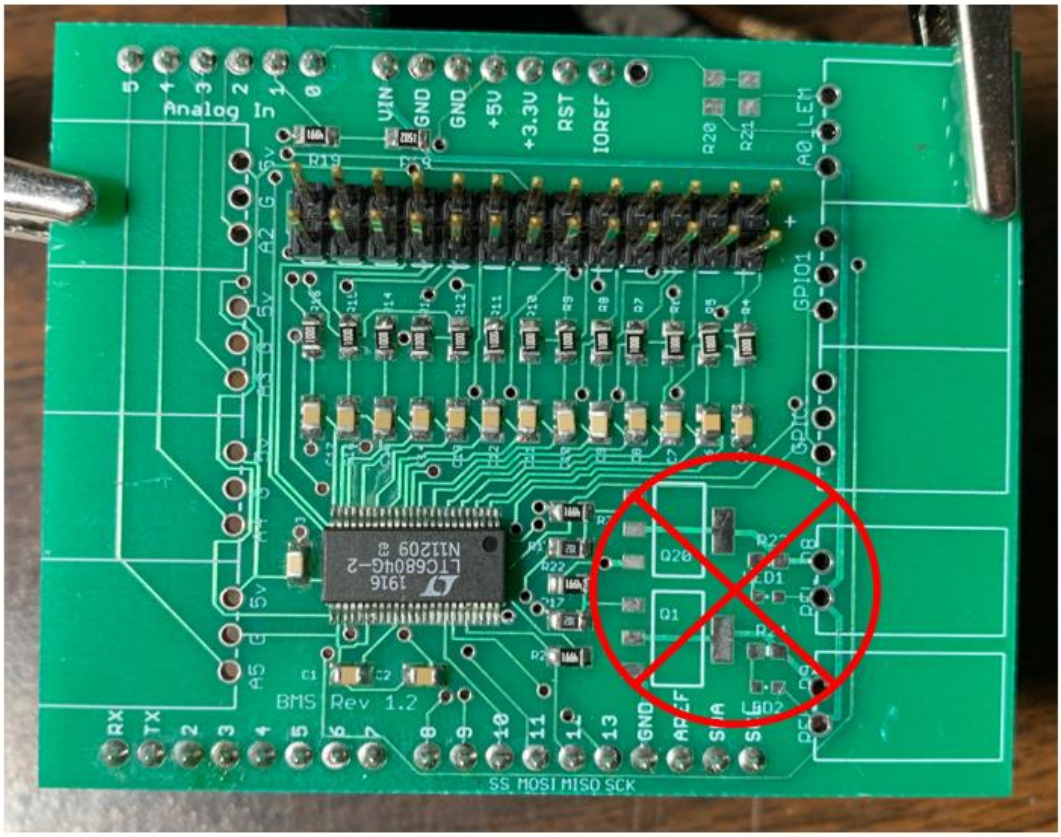

Figure 8: Partially populated Arduino BMS shield. The red circle contains an unused relay driver circuit.

To confirm the LTC6804 BMS PCB helps the project meet the specifications in Chapter 2, it must possess the ability to accurately measure and communicate the cell voltages to the Arduino. The datasheet claims a cell measurement error of less than 1 mV at specified voltages. However, according to Davide Andrea, the protection circuits may introduce a small voltage difference between C0 and V-, significantly affecting the error between the reported “SOC” voltage and actual string voltage [4]. Verification begins by measuring the voltage of the entire battery pack connected to the BMS with a known reliable voltmeter. The voltage reading from the BMS should agree within 100mV. Repeating this test for each bank of cells proves the device functions as required. Next, verification includes measuring the BMS reaction speed to overvoltage and undervoltage faults; cutoff must occur less than 1 second after meeting the condition. The device also must balance the voltage of each cell bank; see Section 5.4.

After uploading code to the LTC6804, covered in section 5.8, expected behavior includes a response from the chip indicating successful configuration of internal registers. The code then sends the cell voltages to the LCD to display. Failure to establish communication results in receiving 255 on the Arduino. Before debugging the code, a multimeter in continuity mode assists in verifying hardware connections and ruling out any mishaps like solder bridges. Connection between ISOMD and V– verifies the LTC6804’s configuration for 4-wire SPI mode.

Pins 45-48 (A0 – A3) connect to V- (ground) on the 6804 to set the device address to 0 in a daisy chain. The datasheet requires connection of pins 36 and 37 to enable the software timer. Measuring various reference voltages on the 6804 assists in ruling out other hardware issues. VREG must read between 4.5V and 5.5V. The device generates reference voltages on pins 34 and 35 measuring 3.2V and 3V; it also outputs between 5.2V and 6.0V on the DRIVE pin. Confirmation of all hardware connections and reference voltages indicates that any existing errors likely stem from programming the device.

During initial construction, an unknown issue causes unintended current to flow from C2 to C1 and from C1 to ground when the BMS and battery are connected; after measuring with the multimeter, only 104 ohms separates C1 from ground, and 66k ohms separates C2 from C1. This creates a dangerous situation as uncontrolled current flow can damage a cell. A functioning PCB has much higher isolation between cells, on the order of mega ohms. Disconnecting the cell balancing PCB from the LTC6804 PCB demonstrates that the issue lies in the LTC6804 PCB; after populating a new LTC6804 PCB with new components, the issue resolves and the device can undergo safe evaluation. Possible causes of this problem include excessive heat during soldering and failed components; further analysis is required to confirm the root cause and address the potential design flaw.

A communication issue between the Arduino and LTC6804 initially stifles further progress. An oscilloscope shows the Arduino sends some data over SPI but receives no response from the LTC6804. Careful measurement with a multimeter confirms all pin connections with the schematic, and reference voltages with the datasheet. Upon further inspection, the Arduino UNO R3 and MEGA2560 use different pins for the default SPI communication. The SPI hardware within the UNO defaults to pins 10, 11, 12, and 13, whereas the MEGA utilizes pins 50, 51, 52, and 53. Figure 9 above demonstrates the necessary modification to use code and shield hardware designed for the UNO on the MEGA. Additionally, the code sets pins 10-13 to high impedance inputs to allow the signals to pass through. After correcting this issue, the device measures the overall pack voltage (V+) approximately 0.5V below the true value, and it accurately measures the voltage of cell banks 1 through 7; an unknown issue leads to inaccurate and fluctuating values for cells 8, 9, and 10. Altering the configuration array solves related problems; the issue likely stems from improper initialization of the LTC6804 ADC or related internal hardware (see Section 5.8 for code discussion).

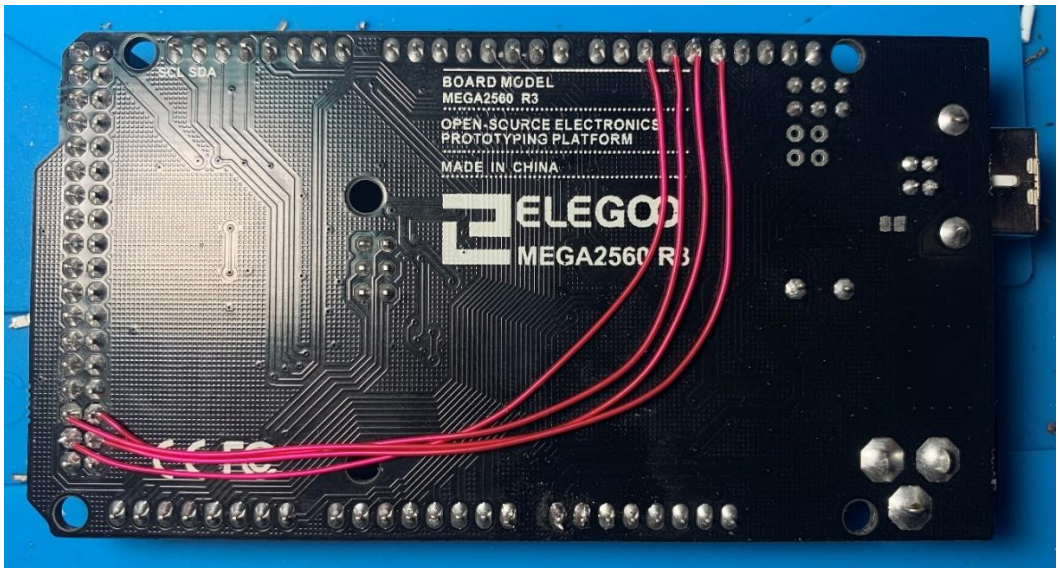

Figure 9: A wiring modification to the MEGA2560 solves the communication issue.

5.3 Charge/Discharge Cutoff Circuitry

Caditz's design uses relays for cutoff switches to disconnect the battery from the load due to a fault condition; a signal generated by the Arduino turns on a transistor that activates the relay. However, relays capable of withstanding the required 50V and 20A specifications significantly increase the cost and size of the device, requiring consideration of alternative cutoff circuits. This project modifies the design by replacing the two charge/discharge relays with versatile, high power MOSFETs to act as a high side switch. The circuit diagram in Figure 10 illustrates the suggested circuit replacement, and the red circle in Figure 8 demonstrates the unused section of Caditz's PCB design. Figure 6 contains the obsolete relay driver circuit diagram; Arduino pin D26 provides the cutoff signal to the optocoupler in the replacement circuit. Before construction, LTSpice simulation of the circuit in Figure 10 verifies correct output toggling with minimal power dissipation. The design requires two cutoff circuits; one circuit switches current that enters the battery (OVP), and the other switches current that exits the battery (UVP, OCP, overtemperature).

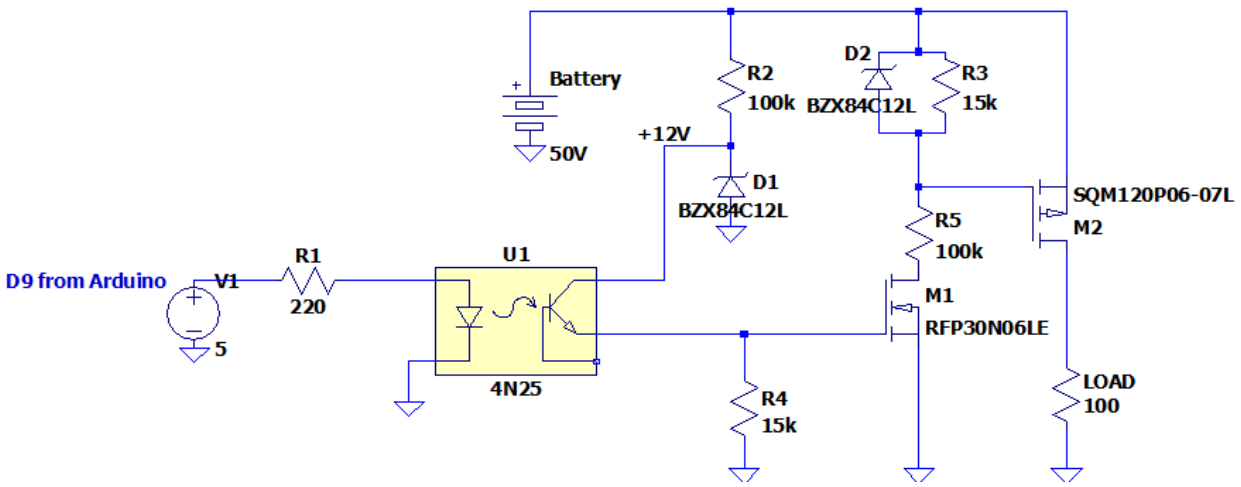

Figure 10: Proposed MOSFET-based cutoff circuit from LTSpice simulation.

The proposed replacement circuit utilizes the common 4N25 optocoupler to isolate the Arduino and power FETs. The circuit requires both an N-type and P-type power FET to act as a high side switch. The optocoupler requires at least 10mA from the Arduino to turn on the LED, which turns on the N-type FET thus turning on the P-type FET and allowing the battery to connect with the load. Zener diodes provide the 12V necessary to turn on the NMOS and limit the gate-source voltage drop across the PMOS. Vishay's SQM120P06 P-Channel MOSFET features high voltage, current, and power tolerances required for use in automotive applications without significant external heat sinks. Vishay claims the FET features an R_{DS}(on) of 6.7 milliohms; the maximum current limit (20A) and on resistance (.0067 Ω) limit the maximum power dissipation to 2.68W using Ohm's Law. The Vishay FET can handle up to 375W and features a junction-to-ambient temperature resistance of 40 °C/W [21]. Under these assumptions, temperatures of the FET remain safe under normal operating conditions with a maximum of approximately 110 °C versus the specified maximum of 175 °C. The N-type FET experiences lower currents than the P-type in this circuit; thus, the specifications of this FET carry less significance, and cost becomes the primary factor in the component selection process. The RFP30N06LE enjoys widespread usage in BMS for its cost to performance ratio. Additionally, the widely available Spice models for these FETs simplify the design and simulation process.

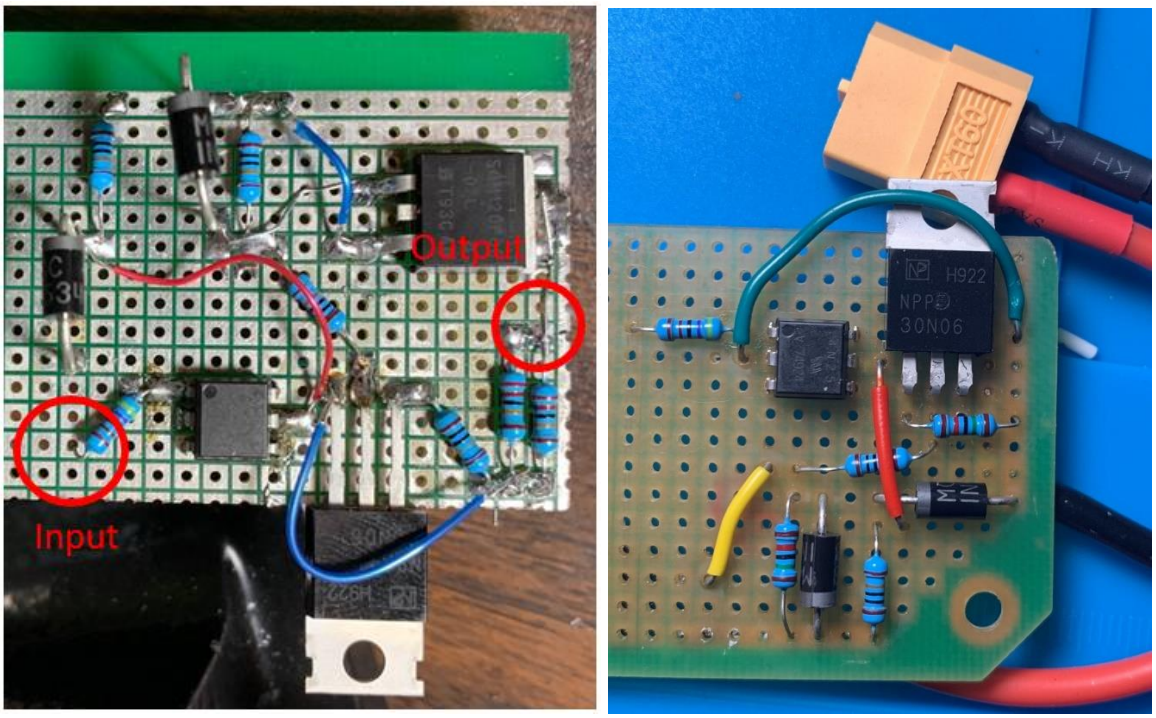

Figure 11: Prototype of MOSFET based cutoff circuit (left) and final version (right).

Figure 11 depicts the functioning prototype cutoff circuit before implementation with the LTC6804. Scrap prototyping board provides a safe and somewhat reliable backbone for testing compared to breadboards. Simple tests show the board functions as expected by simulations. Confirming functionality of the prototype permits construction of the final version of the cutoff circuit used in this project as shown in Figure 11. Evaluating the cutoff circuit includes verifying the circuit's capability to quickly disconnect the battery from the load based on an input received from the Arduino. The device outputs zero volts when the Arduino pulls the optocoupler input to ground, and it outputs the battery voltage (34.8V) in the presence of the 5V signal. Verification requires testing for the entire specified voltage range from 11V to 50V. The device cannot introduce significant voltage drop between the battery and the load; the drop across the FET measures less than 10mV at 1A. Finally, the time between receiving a fault signal and turning off the voltage must span less than 0.5 seconds while coordinating with the rest of the system to meet the 1 second timing specification in Chapter 2.

Initial attempts at evaluating the circuit fail to produce the expected results after integrating with the rest of the system. After addressing an issue with insufficient current driving the optocoupler, the circuit performs as expected. The initial design uses a 4.7k Ω resistor between the Arduino digital pin and the input to the optocoupler; this provides less than the 10mA required to turn on the LED, but the device appears to function in simulation and with an external power supply. Replacing the 4.7k Ω resistor with a 220 Ω fixes the issue and allows the circuit to perform as expected with the Arduino. The next steps include thorough evaluation of the cutoff circuit to verify it acts as a safe replacement for the relays. If the circuit fails to meet the specifications,

future design improvements can likely address any issues. For example, the SQM120P06 features poor compatibility with the chosen prototype board because it requires surface mount soldering, resulting in poor connection quality between the FET and circuit output. Future work also involves designing a dedicated PCB for this section of the BMS to safely handle the high current, increase volumetric efficiency, and improve connection quality with other components of the BMS.

5.4 Cell Balancing

As mentioned in the specifications list in Chapter 2, the BMS must feature passive cell balancing. The LTC6804 offers multiple options for balancing; the user may utilize the FETs internal to the IC or provide their own external circuitry. The internal FETs limit the balancing current to milliamps, and the external circuits allow the user to decide the balancing current as needed. Figure 12 illustrates the two options from the LTC6804 datasheet.

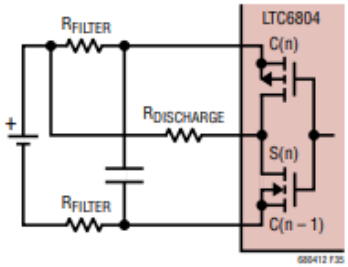

Figure 35. Internal Discharge Circuit

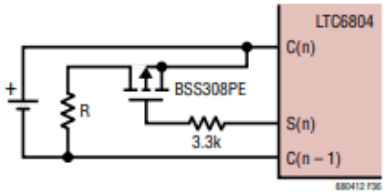

Figure 36. External Discharge Circuit

Figure 12: Cell balancing options offered by the LTC6804 [6].

While the simple method using internal discharge circuitry shows potential, the relative lack of available information highlights the attraction of the external method. Caditz provides an additional PCB that stacks vertically with the LCT6804 Arduino BMS PCB. While this facilitates connection to the Arduino, it complicates troubleshooting, because it covers the LTC6804 Arduino BMS entirely. PCB fabrication also adds significant production cost to the project. Multiple design paths emerge from this tradeoff, and they are summarized in Table XI along with their weights for each aspect of the development process and score in each section. Option 1 involves ordering and using the PCB designed by Caditz. Option 2 uses the same components as Option 1 but saves money by constructing the circuit on prototyping board

instead of a custom PCB. Option 3 includes an unknown amount of researching the internal cell balancing within the LTC6804 and constructing the required additional circuitry.

TABLE XI: CELL BALANCING DESIGN TRADEOFF MATRIX

Design Consideration (weight)	Option 1	Option 2	Option 3
Time requirement (9)	3	1	1
Cost (5)	1	2	3
Complexity (3)	2	2	1
Presentation (1)	3	1	2
Robustness (1)	3	1	2
Score:	44	27	31

Weight	Meaning
1	Not important
3	Somewhat Important
5	Important
9	Critical

Score	Meaning
1	Worst
2	Ok
3	Best

Initially, the design tradeoff matrix suggests Option 2 over the others due to misjudging the complexity and time requirement. Option 3 carries excessive uncertainty and limits the device's capabilities. After attempting Option 2, it fails to meet expectations; issues related to soldering small components and ensuring safe connections prevent the idea from succeeding. Thus, the design path changes to incorporate Option 1 in the final design. The cell balancing components electrically connect to the LTC6804 in parallel with the cell filtering networks as shown in Figure 13. Since the design uses fewer than the maximum 12 cell banks in series, the top two cell pins of the LTC6804 (C11 and C12) connect to the V+ pin for proper functionality. Both Options 1 and 2 utilize the concept shown in Figure 36 from the LTC6804 datasheet (found in Figure 12 above). When the device detects an imbalance in cell voltage, the LTC6804 signals the FETs to drain current from the overcharged cells through the resistors. The FETs in this design (BSS308PE) permit up to two amps of balancing current; however, the resistor used (16 ohms) limits the current to approximately 250mA. This exceeds the capabilities of the internal circuits by a factor of 5.

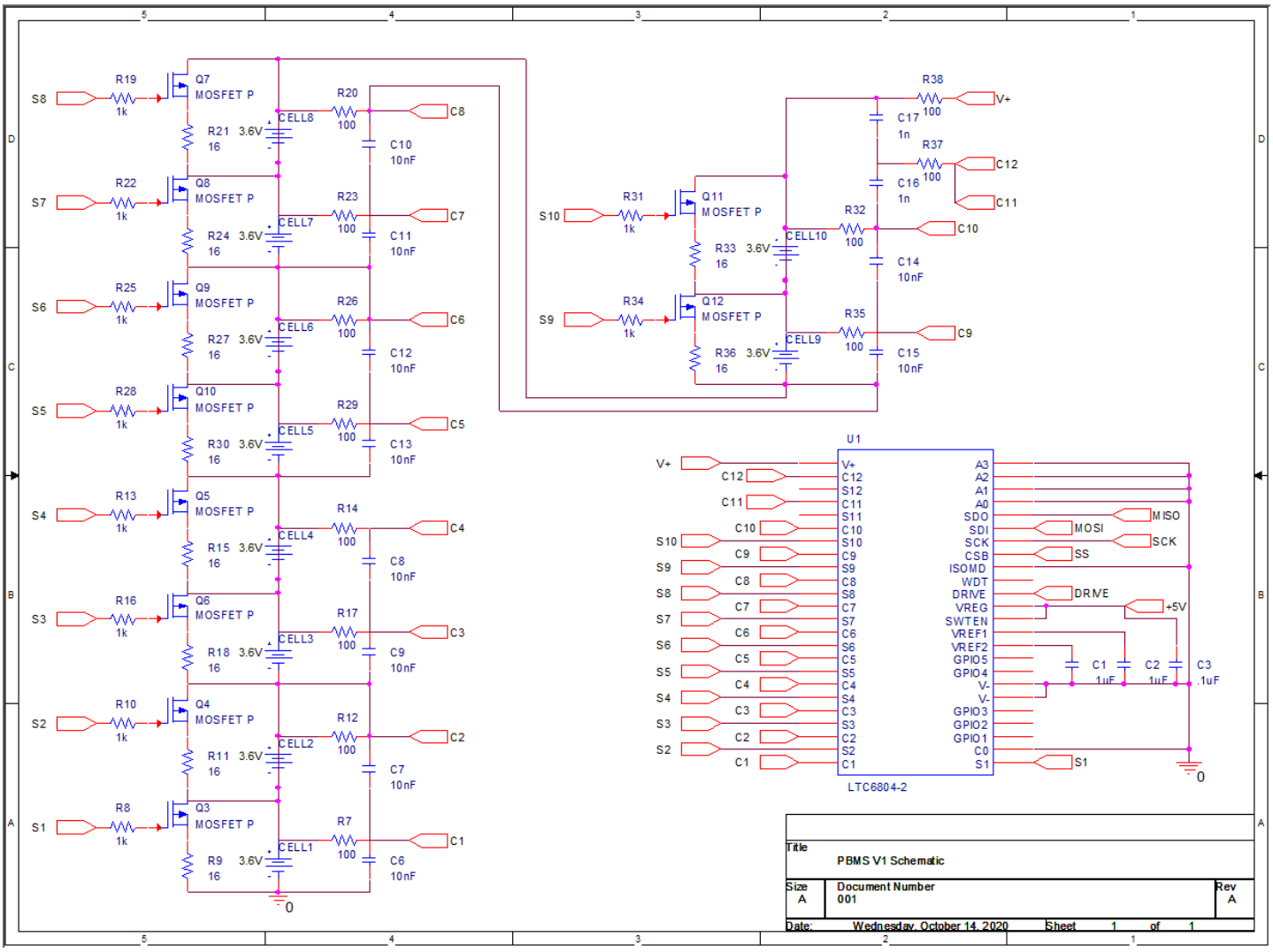

Figure 13: LTC6804 cell balancing circuit diagram. The reference designations differ from those in Figure 14.

This project makes two modifications to Caditz' balancing PCB. First, the programmable BMS excludes the LEDs that indicate when each cell's balancing MOSFET conducts. The LEDs consume power and provide an additional point of failure. Although the cell balance PCB stacks vertically on the LTC6804 BMS PCB using 2.54 mm headers, this prevents access to the pins of the LTC6804 and significantly complicates troubleshooting during the design process. Thus, the alternative connection method in Figure 14 using jumper wires provides the best method to meet present needs. The final implementation benefits from the compact design achieved using the headers. The connections between the battery voltage banks and LTC6804 BMS require extreme attention. Reversing the polarity destroys the costly LTC6804. The design requires modification to include a keyed connector that prohibits reverse polarity.

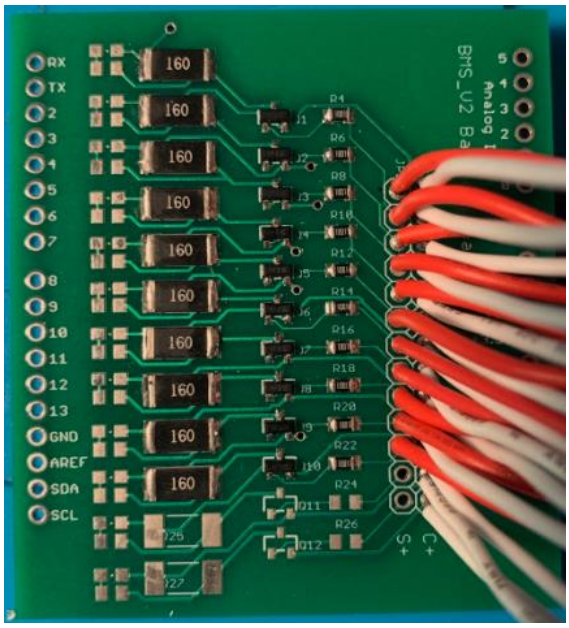

Figure 14: Cell balancing PCB designed by Caditz; this implementation eliminates the LEDs and uses minimal components.

The specification list requires the BMS to balance the cells within 2-3% of each other. With cell voltages ranging from 2.5 – 4.2V, this results in acceptable measurement error up to 75 – 126 mV. To confirm the BMS balances the cells, at least one cell bank must read a different voltage than the others. To achieve this, the battery initially reads around 20-50% of the full capacity. Then, additional current applied to only one cell bank increases the voltage of that bank relative to the others. After connecting the entire battery to a charger, the device charges the cells equally until any one bank reaches a predetermined voltage set within the LTC6804. Once detecting this voltage, the LTC6804 diverts current from the overcharged cell through the 16 ohm resistors until the cells all read equal voltage. The cells should all read the same voltage, within 126 mV, once the pack reaches full capacity to meet the specified requirements. After establishing communication between the LTC6804 and the Arduino, the display must show accurate cell voltages for all banks before evaluating the cell balancing functionality. If the device falsely detects one cell bank much lower than the others, it attempts to balance the pack according to the lowest measured value. The measurements for cells 8, 9, and 10 currently fluctuate and prevent evaluation of the cell balancing feature.

5.5 Current Sensor

The specifications in Chapter 2 demand current measurement up to at least 20 amps. The ACS712 module, based on the Hall effect, supports up to 30 amps in either direction and promises simple integration with the Arduino microcontroller [22]. The current sensor detects the current of the positive wire from the battery while in series with the load/charger. The ACS712 datasheet specifies a measurement resolution of 66 mV/A and 1.2 mΩ conductor resistance. XT60 connectors facilitate removal of the sensor during prototyping, if necessary.

The XT60 connectors use 14 AWG wire; however, the current sensor input requires a smaller diameter despite supporting up to 30 amps. For this reason, soldering the wires directly to the PCB provides the best connection. Containing only 3 pins, few hiccups arise while integrating the current sensor with the Arduino. The sensor requires a 5V and ground connection from the Arduino; it outputs an analog voltage to the Arduino on pin A8. As a widely used device, an internet search provides countless examples of implementation. The current sensing circuit's small dimensions allow versatility in its location within the BMS. Figure 15 features the current sensor with the battery connectors and wires to the Arduino.

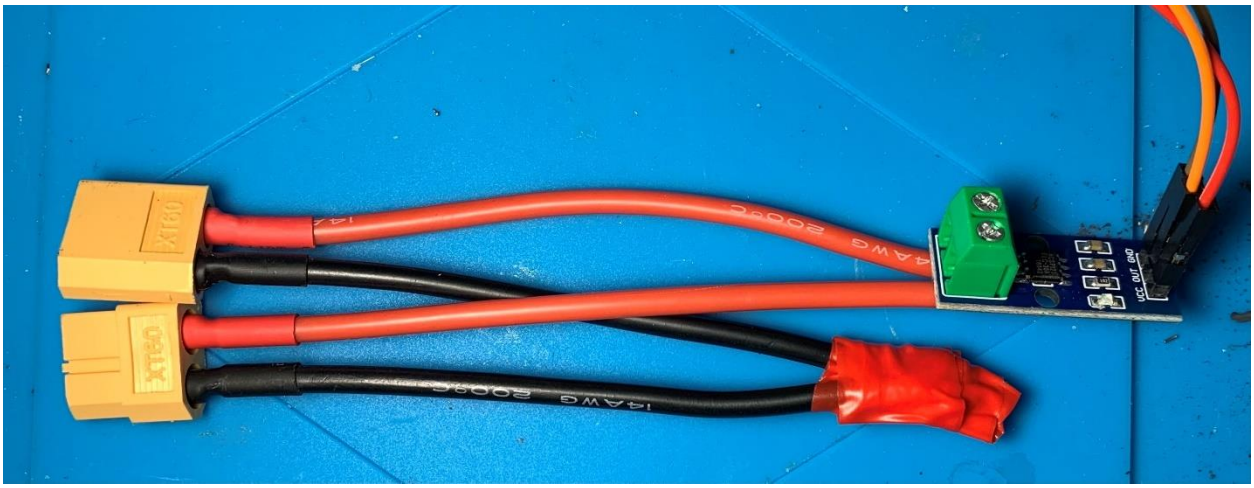

Figure 15: Current sensor based on ACS712 IC.

A library built for the ACS712 by GitHub user Ruslan Koptiev facilitates incorporation of the module with the Arduino [23]. Utilization of the library only requires the sensor model number to set the input sensitivity. The library contains functions to automatically calibrate the sensor, measure AC or DC current, and set the zero point of the sensor. The Arduino calibrates the sensor on startup and measures the current during each loop. Recalibration occurs when the output is disabled, as calibration requires zero current flow through the ACS712. After measurement, the Arduino stores the value and displays it on the Nextion 4.3" LCD.

The datasheet for the ACS712 claims 1.5% measurement error from -30A to 30A at 66mV/A sensitivity for the 30 amp model (x30A). Verification of the current sensor's capabilities requires measuring the current under different loads and comparing the sensor's measurement to the real value. The test setup includes the battery, Arduino, current sensor, and load. USB noise forces the Arduino to use two spare 18650 cells in series as a low noise, isolated power supply. While unable to test the device at the maximum 36V and 30A, the BK Precision 8540 Electronic Load features a maximum power sink of 150W, permitting verification up to 3 amps safely at the nominal voltage. The device permits the user to adjust the resistance and displays the current absorbed by the load with 1mA precision. Comparing the measured value from the sensor with the reading on the BK Precision 8540 indicates the sensor features higher error than expected. The measurement constantly fluctuates despite the code averaging many samples. The results in

Table XII summarize the device's capabilities in this implementation. The results indicate higher uncertainty at lower current values than higher values, with a minimum measurement error of 4% versus the claimed 1.5%. With this accuracy, the device succeeds in meeting the specification of 1A accuracy while including room for improvement. At full load (20A), 5% error converts to a margin of 1A, an acceptable error for a battery measurement device in this operating range [8]. The measured on resistance of ~1 milliohm indicates a maximum of 0.5W dissipation at full load, helping the device remain under 5W total consumption.

TABLE XII: MEASURED CURRENT SENSOR RESULTS VS ACTUAL CURRENT READING FOR VALUES UP TO 3 AMPS AT 35V

Actual Current Value (A)	Current Sensor Reading (A)	% Error
0.00	-0.04 $\pm$ 0.06	-
0.50	0.5 $\pm$ 0.03	$\pm$ 6%
1.00	1.01 $\pm$ 0.03	$\pm$ 4%
2.00	2.05 $\pm$ 0.04	$\pm$ 4.5 %
3.00	3.10 $\pm$ 0.03	$\pm$ 4.3%

The root cause of the discrepancy between the observed and expected accuracy lies in the combination of the low sensor sensitivity and the low resolution ADC. At 66mV/A, the maximum voltage output from the sensor relative to the baseline is 66mV/A * 30A = 1.98V. For the Arduino's 10 bit ADC, this corresponds to approximately 2mV per bit. While the system detects these small changes, noise can lead to significant error in the reported measurement. The accuracy may also stem from poor calibration or Arduino ADC instability. Arduinos using the 5V power from USB experience noise which may cause issues with the reference voltage. A test replaces the USB power with two 18650s to eliminate noise and provide a consistent, high-power supply; however, the current reading shows little improvement. To improve resolution, precision, and volumetric efficiency, future designs can incorporate a low resistance shunt resistor on the cutoff circuit PCB and measure the voltage drop using a precision ADC.

5.6 Temperature Sensor

The PBMS design allows incorporation of multiple temperature sensors. The low cost MCP9700 thermistor pictured in Figure 16 provides a simple and inexpensive method of confirming this functionality. The sensor features only 3 pins; +5V, ground, and an analog voltage for the Arduino. Implementation in the software requires simply reading the analog pin value and converting the ADC measurement to the corresponding temperature, as demonstrated in the code example below. The voltage generated by the temperature sensor increases by 10mV for each degree Celsius [24].

```
float T;
int tempPin = A9;

void loop () {
  T = (analogRead(tempPin)*5./1024. - 0.5) * 100.;
};
```

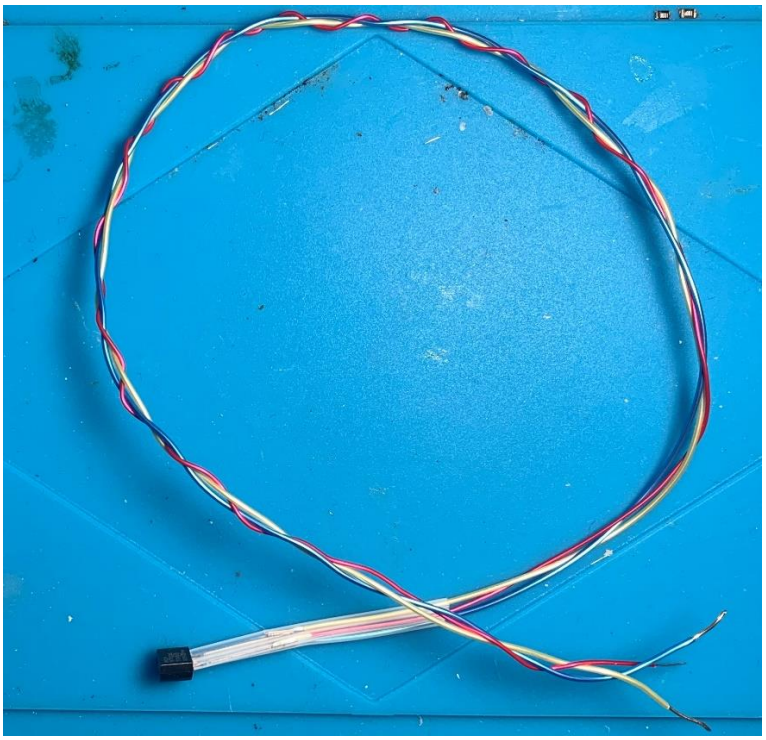

Figure 16: MCP9700-E/TO temperature sensor and cable.

The attached wires allow a maximum distance of 12 inches from the Arduino, providing sufficient length to measure any location of the battery. The sensor features a measurement accuracy of ± 4°C from 0°C to +70°C when placed directly against a battery cell [24]. Evaluation includes comparing the measured temperature of the cell by the MCP9700 to a reading by a device with verified accuracy. With a relative lack of equipment at disposal, a multimeter with thermocouple provides the best option for the reference temperature measurement. To ensure equal temperature of both sensors, they are suspended in warm water while taking care not to short the leads of the MCP9700 in the water. The test then compares the measurement from the thermocouple with the values generated by the sensor before and after ADC conversion. The measured voltage from the middle pin of the MCP9700 converts to temperature using the 10mV/°C equation from the datasheet. The code snippet above generates the temperature after ADC conversion. Table XIII below illustrates the results and relative error for each measurement.

Table XIII: MCP9700 characterization results. All measurements listed in degrees Celsius.

Thermocouple Measurement	Calculated Temperature from Measured Sensor Voltage	Absolute error	Temperature Calculated After Measurement by Arduino ADC	Absolute error
72.8	71	1.8	68	4.8
66.7	64.5	2.2	62	4.7
62.8	61.2	1.6	58	4.8
59.4	58	1.4	57	2.4
57.2	55.7	1.5	54	3.2
53.9	52.5	1.4	51	2.9
51.7	50.4	1.3	50	1.7
48.9	47.6	1.3	48	0.9
46.1	44.5	1.6	44	2.1
43.3	42	1.3	41	2.3
40.5	39.5	1	38	2.5
37.8	36.8	1	35	2.8
35	34	1	33	2
31.7	31	0.7	30	1.7
28.9	28.2	0.7	27	1.9
26.7	26	0.7	25	1.7

Actual tests show accuracy to approximately 2 degrees Celsius using the voltage from the sensor, and 2 - 5 degrees of fluctuation after ADC conversion. The data directly from the sensor shows a reasonably linear response as in Figure 17, but the ADC conversion introduces nonlinearity. Despite these issues, the temperature sensor meets the $\pm$ 5 °C specification in Chapter 2. After characterizing the sensor, additional Arduino code takes 25 temperature measurements and reports the average result. Averaging greatly improves accuracy of display value; now, the measurement from the ADC indicates agreement within 1-2 degrees. The nonlinearity introduced by the ADC likely stems from noise from the USB power supply influencing the ADC reference voltage AREF. The measured value using USB power reads 4.8V instead of the require 5.0V.

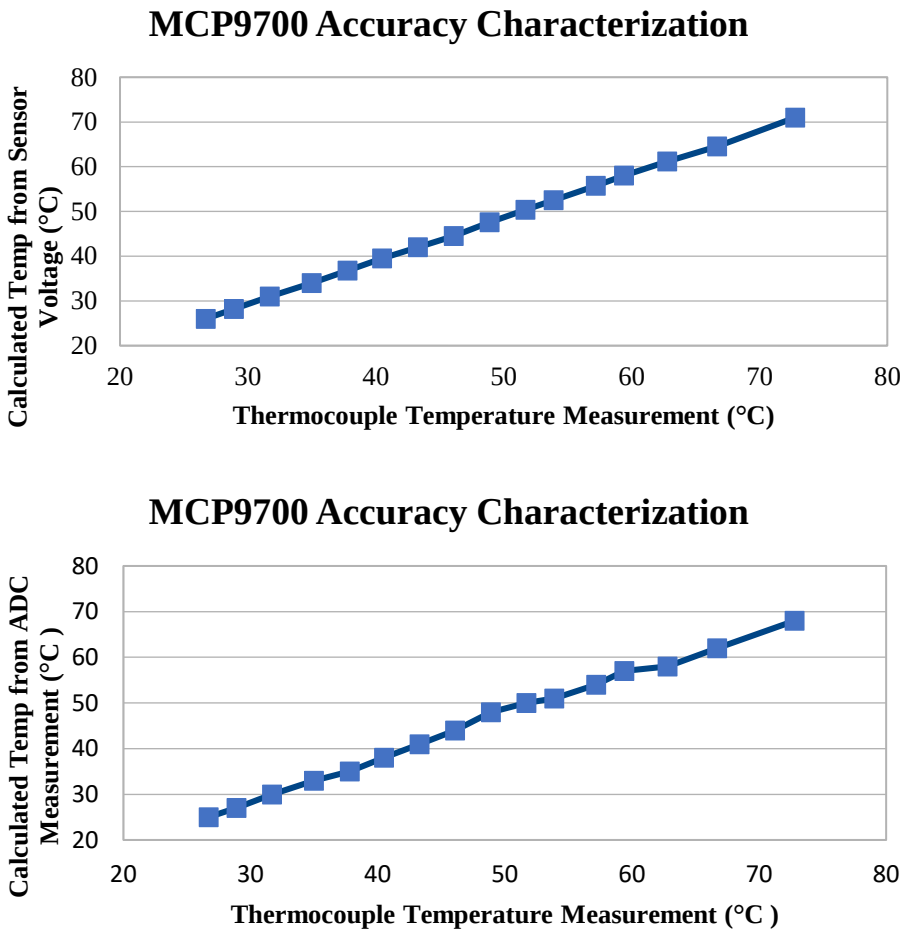

Figure 17: MCP9700 Characterization results. Linearity of sensor voltage (top) vs linearity after ADC conversion (bottom).

Although the temperature sensors meet the requirements, the design leaves room for improvement. The next step incorporates the temperature sensors with different 3-pin connectors on the BMS PCB to improve connection quality. The connectors from Caditz's design (TE/APM 5-103634-2) fit together loosely and fail to make consistent contact. The MCP9700 datasheet also includes a calibration method for $\pm$ 0.5C accuracy for further enhancements to the PBMS temperature measurement accuracy (see AN1001 [25]).

5.7 Graphical User Interface

The requirements of the project include a simple user interface that provides direct access to the BMS limits. The Nextion Touchscreen LCD allows fast and simple prototyping of a user-friendly interface; the manufacturer provides software called Nextion Editor which significantly decreases complexity. The editor includes quick methods to place pages, buttons, and values. The picture used as the background for each page must match the screen's resolution (480 x 272 pixels). Each picture starts as shapes in Microsoft PowerPoint, and Microsoft Paint permits

saving the image in the correct resolution. Placing some of the static objects in the background reduces the number of objects stored by the display and simplifies the code. The Arduino easily controls these objects through serial communications (UART) and preset commands. For example, to change the value of the "Output Voltage" monitor, the Arduino sends the command "Vout.txt=" before sending the output voltage as a string. The Nextion display features poor support for floating point numbers; the simplest method converts floats in Arduino to strings before sending to the LCD to display. Likewise, the user sends signals to the Arduino by tapping the screen. Buttons created in the Nextion editor contain a method of sending string objects over the serial port after detecting a touch. The Arduino waits for any user input and determines the next action by comparing the received string with a list of expected commands. After designing the GUI in the editor, an SD card transfers the code from the computer to the LCD. The display must connect to a 5V source that supplies at least 500mA during an SD update; the computer's USB port lacks the current necessary to power the screen and Arduino during this update.

The graphical interface features two sections; the first page shows battery measurements and characteristics, while the second allows the user to change the operating limits. Depicted in Figure 18 below, page one of the PBMS features measurements of the battery pack's output voltage, output current, pack temperature, number of cycles, and state of charge (SOC). It also indicates the status of the output with a colored indicator and provides a reset button for the user if the BMS encounters a fault. The current implementation uses the simple SOC estimation based on the open circuit voltage described in Chapter 1. The number of cycles does not currently reflect the actual number of battery cycles; this feature requires additional programming after verifying the accuracy of the voltage measurement from the LTC6804. Both page one and page two display the real time cell voltage measurements (C1 – C10). If the user presses the green "Edit Mode" button at the bottom of the GUI, the LCD changes to display page two, seen in Figure 19. Indicators for the presence of each type of fault event can provide useful information to the battery operator and warrant inclusion in future versions.

Page two provides the user access to the maximum pack voltage, minimum pack voltage, maximum output current, and maximum pack temperature. The user changes the values by tapping either the "-" or "+" buttons to decrement/increment the value. The voltage and current values change by 0.1 V or A for each tap. The maximum temperature in memory changes 1 degree Celsius for each button press, although the MCP9700 sensor used in this iteration only provides ± 4°C accuracy. The user must set the maximum temperature to 4°C below the actual maximum allowable temperature. The corresponding values stored in the Arduino change immediately as the user interacts with the device. When the user finishes changing the values, tapping "Monitor" returns the user to page one.

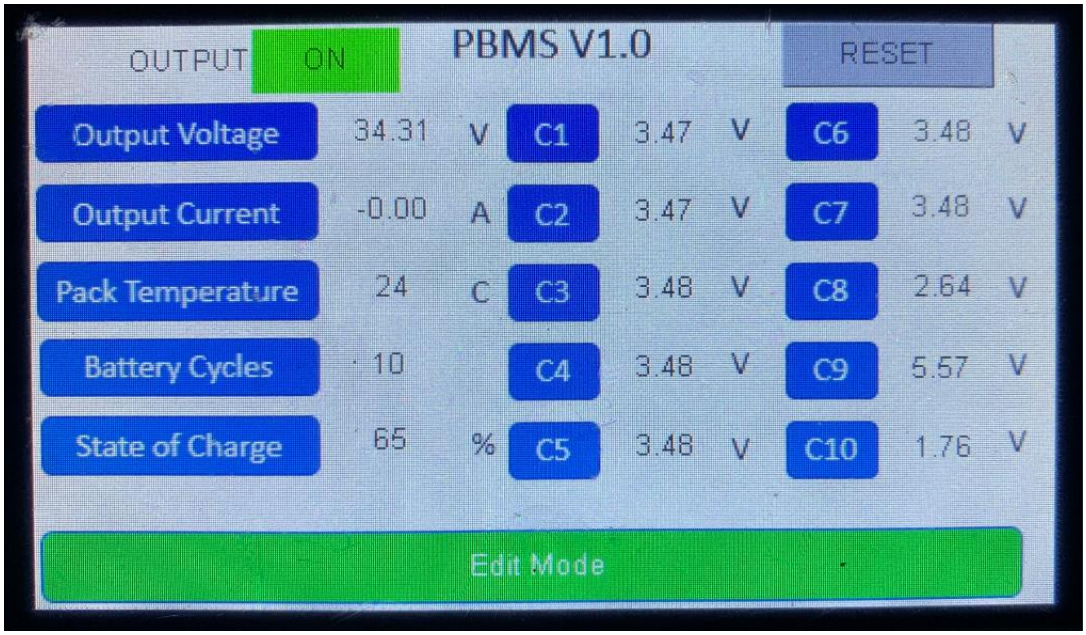

Figure 18: Page one ("Monitor") of PBMS graphical interface.

Figure 19: Page two ("Edit Mode") of PBMS graphical interface.

5.8 Arduino Code

The Elegoo MEGA2560 (Arduino) provides the central processing unit of the programmable battery management system. The peripherals, including the LTC6804, LCD, and sensors, act as minion devices communicating with the controller. While nonfunctional, Appendix B includes the complete Arduino code used in development. The program builds from the code provided by Caditz in the Instructables article to incorporate this design's modifications including the unique

cutoff method, temperature sensor, current sensor, and touchscreen based user interface. The flowchart in Figure 20 illustrates the theory behind the BMS operation. Each loop includes monitoring each subsystem and updating stored values as needed. The program compares the measured values with the stored limits and sends a signal to the cutoff circuit if necessary. The device then waits for the user's input before continuing operation.

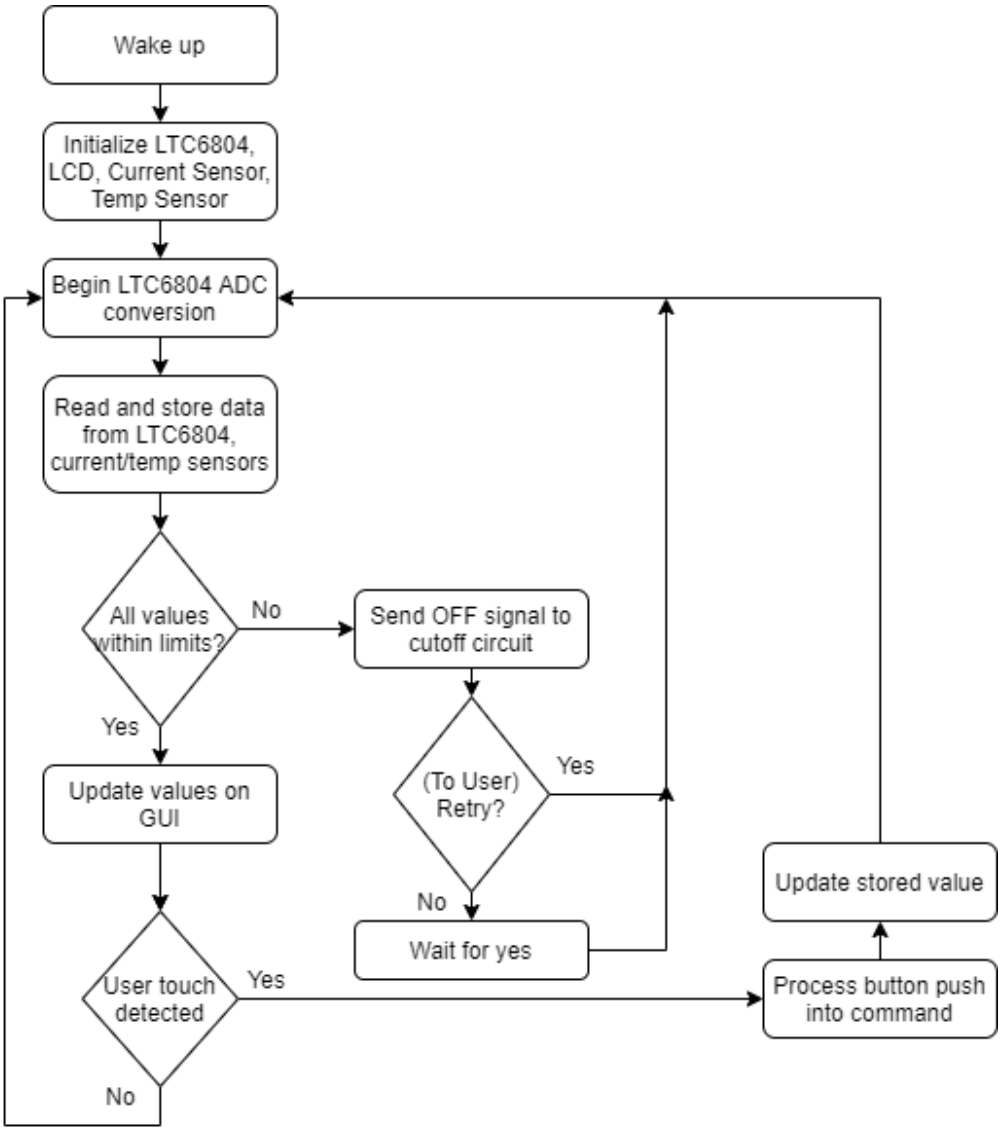

Figure 20: Top level flowchart of PBMS V1.0 software.

One of the advantages gained by the programmable battery management system includes the user's ability to adjust the operating limits during operation. If the user reaches the lower limit of the pack voltage, the software permits the user to lower the limit and continue operation. Comparable battery management systems disable the output until the user charges the battery voltage to the minimum value. However, a software bug may reset the user's selection if they press the "reset" button after changing a value. In order to properly change the value in storage,

the user must remove the condition causing the fault before pressing reset; then, the value entered by the user serves as the new limit until pushing the “reset” button again. Future refinements to software include allowing the user to set a default value at the beginning of operation and storing changes without requiring the user to re-enter the value after resetting the output. This also enables the user to continue battery operation without removing the fault condition.

The code contains functions to measure the pack temperature, measure the output current, and communicate with the LCD. The ACS712 library contains all functions necessary to use the current sensor. The temperature sensor, also based on an analog voltage measurement, utilizes similar functions. Both the current and temperature measurements include averaging of multiple readings to reduce measurement error. The code to communicate with the LCD simply checks for user inputs by waiting for an indication through the serial port, and it updates objects on the display by sending the values back though UART.

Linear Technologies (LT) provides multiple libraries to facilitate designs utilizing the LTC6804. The code uses “Linduino”, “LT_SPI”, and “LTC68042” to reduce the required software development. The first two simplify communication setup between the Arduino and LTC6804 BMS. Specifically, they allow the program to utilize SPI write and read functions with the LTC6804. The IC’s library contains functions that initialize the BMS, begin specific processes within the BMS like ADC conversion, and force the IC to “wake up” or “sleep”. These included functions save time by automatically generating the bytes that configure the chip’s internal registers. They also facilitate interpretation of the PEC (packet error code), which communicates the BMS error status. Additional functions provide access to features such as the LTC6804’s GPIO pins, but the PBMS ignores these in the current design. The LTC6804 datasheet includes extensive explanation of utilizing multiple LTC6804 chips with one another. The PBMS can expand to make use of this capability in future applications as the design scales to larger batteries.

After solving communication issues, the code fails to configure the LTC6804 to measure all 10 cell voltages accurately. At first, the device only reports the values of the first three cell banks; further research reveals errors in the configuration array sent to the device using Caditz’ code for this implementation. The initialization function configures the LTC6804 using the CFGRx[] arrays. CFGR0 bits 3-7 initialize the LTC6804 GPIO pins (unused); bit 2 indicates use of the WDT; bit 1 is a “don’t-care”; bit 0 configures the ADC operating mode. The other registers set over/undervoltage limits (CFGR1-3) and dictate which cells may discharge for balancing (CFGR4-5). A presentation by GitHub user Ayush Agrawal assists in setting these registers [26]. After altering the configuration array to exclude the GPIO and UVP/OVP features, the device reports cell voltages C1 through C7 within a few mV of the true value; C8 through C10 fluctuate and do not represent the actual voltage of that cell bank. The next steps include closer inspection of the WDT and ADC initialization (CFGR0[2] and CFGR0[0]) and consultation with the datasheet to determine the proper configuration data.

5.9 Enclosure

Although this project lacks an enclosure during the development phase, a case facilitates eventual integration with a battery for regular usage. Table XIII below lists the dimensions of each component in preparation of enclosure design for a final product. The current design fails to meet the specification of 4”x4”x1” due to the connection methods during troubleshooting. Fully integrated as a final product, the estimated dimensions measure 0.75” deep by 3” wide by 5” long using the measurements from Table XIV. A new shield containing all supporting circuitry, including the BMS, cell balancing hardware, cutoff circuitry, and peripheral interfacing allows a depth of approximately 0.7”, since the thickest component is the BMS PCB. Integrating the microcontroller onto the same PCB as the other electronics permits a total depth under the specified 1”. Additionally, the LCD used in prototyping features an excessively large size for the function it provides. A screen measuring closer to 2” by 3” accomplishes the same task while saving significant space. This product adds 11.25 cubic inches of volume to the battery’s 69.3, which equates to an increase of 16%. At smaller scales, the size of the BMS becomes a significant factor; however, the added volume is negligible in larger implementations as the product scales up. Commercial systems with similar capabilities measure between 5 and 114 cubic inches in volume [9][5]. For the battery used in this project, this correlates to a volume increase of 7.2% to 164%. The smaller BMS from Bestech features no screen nor simple method of adjusting limits. The larger system from Orion generally fits in stationary systems due to the enormous volume. Based on other offerings in the industry, the potential size of the final BMS meets the requirement of adding an insignificant volume to the battery while adding valuable features.

TABLE XIV:
SUBSYSTEM AND COMPONENT DIMENSIONS

	W	**L**	**D**
Battery Pack	3.15”	8.0”	2.75”
Microcontroller	2.11”	4.26”	0.50”
LTC6804 BMS PCB	2.11”	2.7”	0.68”
Cell Balance PCB	2.11”	2.35”	0.22”
Cutoff Circuit	1.63”	3.1”	0.5”
Nextion LCD	2.92”	4.73”	0.48”
Current Sensor	0.52”	1.25”	0.56”

5.10 Design Evaluation

After building and evaluating each subsystem independently, the next step involves incorporating all components into a complete system. Figure 21 encompasses the first prototype programmable battery management system with all subsystems integrated. The first prototype system uses the Arduino's USB port for power; final implementation requires a DC-DC converter between the battery and the Arduino's VIN pin. A converter like the 3796 from Pololu Corporation outputs a constant 12V up to 600mA with an input ranging anywhere from 12.2V to 50V, allowing the system to function using power from the battery pack itself [27]. After finding the USB supplies insufficient power for the entire system, two 18650 cells in series (not pictured) replace the USB power supply and provide a noiseless source for troubleshooting.

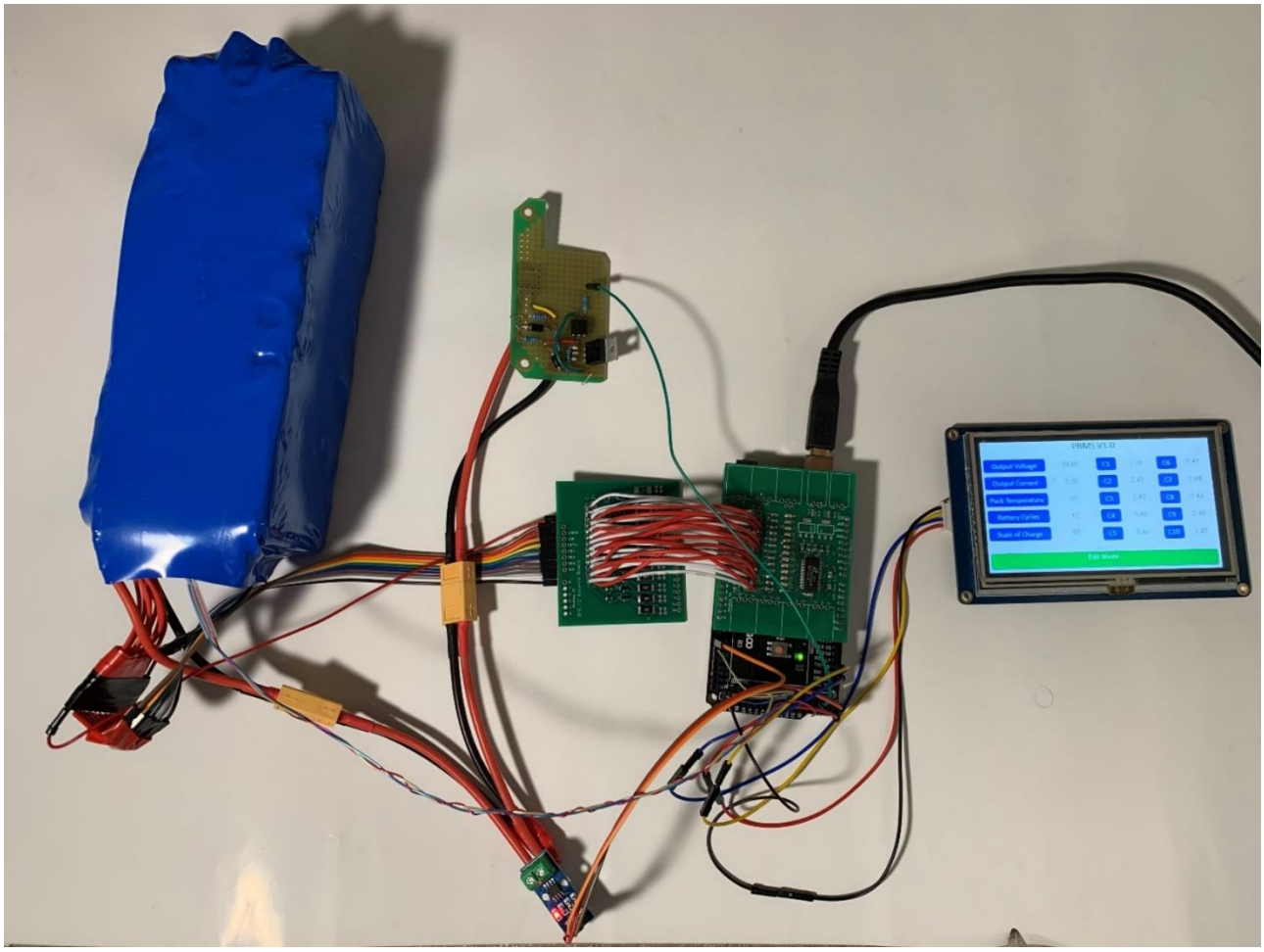

Figure 21: Programmable Battery Management System Version 1.0 with all subsystems integrated

Despite many of the subsystems functioning as intended, the final device fails to perform some of the required functions. The code provided by Caditz does not function in this implementation; the wiring modification in Section 5.2 and the configuration array change in Section 5.8 correct the issues and allow the device to communicate. After addressing these, additional hardware and software problems stagnate progress. The unintentional current flow, mentioned in Section 5.2,

poses a potential safety issue as the root cause remains unknown. Measurement errors likely stem from incorrect configuration of the LTC6804. In its current form, the LTC6804 provides a voltage measurement of the pack approximately 0.5V below its true value, and it accurately reports the voltages of cells 1 through 7. As altering the configuration array resolves similar issues, the next step includes further inspection of the values in these registers to verify proper initialization. The failure to produce accurate cell measurements inhibits further progress and evaluation and remains the top priority in development.

The cutoff circuit initially performs as predicted by simulation; it achieves the goal of quickly disconnecting the battery from the BMS power input/output with a low voltage drop and power consumption. This design adds immense capability relative to the relay based approach while reducing cost. It also features component flexibility if future design expansions require different limits. Issues with both the prototype and final version reveal the design flaw documented in Section 5.3; addressing this flaw appears to allow the circuit to function as expected with the Arduino. While the current sensor offers an inexpensive and simple means of prototyping, the design leaves room for improvement and requires additional research regarding methods of accurately measuring current. After integrating the current sensor with the other subsystems, the current measurement appears to fluctuate approximately 100mA and suffers from poor sensitivity. One method to possible address this problem inserts a current calibration function in the code with a button on the GUI. Alternative current measurement methods may provide increased reliability without the need for constant calibration by the user in future design iterations. One current measurement concept utilizes a precision ADC measurement of the voltage drop across a low resistance shunt resistor. The temperature sensor provides accurate results, but a lack of proper testing equipment prevents thorough evaluation. The sensor meets the requirements of this project and can detect when pack temperature exceeds safe values, but the measurement contains room to improve. Future opportunities for design improvement begin by taking advantage of the calibration method provided by Microchip to achieve $\pm$0.5C precision.

Preliminary tests indicate the device successfully measures and reports the output current and operating temperature within the required margins from Chapter 2; the voltage measurements fail by approximately 400mV due to a measurement error from the LTC6804. Discovering the source of issues late in the development cycle limits the extent of system integration and evaluation currently possible. Before evaluating the complete system, the BMS must report accurate voltage measurements for C8 – C10; this also enables verification of the cell balancing function. After resolving all issues and verifying the success of each subsystem, complete system evaluation indicates whether the PBMS design achieves the requirements from Table III. Table XV below lists the steps to evaluate the system after completion.

TABLE XV: EVALUATION PLAN FOR PBMS V1.0

Requirement	Test	Result
Measures pack and cell voltages to 100mV accuracy	Compare the reported values from the LCD to the multimeter values for all voltages from 11.1V to 50.4V	The system measures the pack voltage with ~0.5V error. The BMS only measures cells 1 through 7 within 100mV
Measures current to 1A accuracy	Compare reported values on LCD to multimeter values for all currents from 1A to 20A	The PBMS provides approximately 100mA of accuracy for all measured currents above 1A (up to 3A possible on test platform)
Measures temperature to 5C accuracy	Compared reported values on LCD to thermocouple values for temperatures from 0 C to 70 C	The measured values agree within 3 - 4 C
Balances cell voltages within 2-3%	Observe cell voltages throughout multiple charge/discharge cycles	Currently impossible; to be completed
Responds to OVP, UVP, OCP, OTP faults in less than 1 second	Verify each fault triggers at the specified value; measure the time between initiating a fault and disconnecting battery output	The device initiates an output cutoff within approximately 250ms for each type of fault
Reports SOC and number of cycles	Observe the values on LCD; verify accuracy (if possible)	The system provides the SOC but lacks the number of cycles.
Consumes less than 5W	Measure power consumption of all subsystems with various loads (0A up to 20A)	Some subsystems approach this limit; verification requires further evaluation
Small volume	Measure dimensions of final system	The system fails due to the experimental setup prioritizing ease of troubleshooting over compactness
Fast setup	Remove the system from the current battery and time the setup process on a new battery	The system fails due to inefficient connection methods between the battery, BMS PCB, and subsystems
Fast settings change	Measure the required time to change the operating limits using the GUI	The device allows changes within seconds. A reset bug erases user settings when recovering from a fault.
Low cost prototype	Estimate the cost of development for a single PBMS	The prototype costs approximately $420 as explained in Section 4.2

With safety a priority, steps to evaluate the PBMS include monitoring the output voltage under various loads and conditions. The system must quickly respond to events that warrant battery cutoff; full evaluation involves measuring the reaction time in each scenario. Overcurrent, overvoltage, undervoltage, and overtemperature events should all measure approximately the same time (under 1 second) from initiating a fault event to turning off the battery's input/output for all expected conditions. Additionally, all cell voltages should remain within approximately 100mV throughout full charge and discharge cycles. Further evaluation includes measuring the power consumption of the device during operation. Initial tests estimate less than 5W of power consumption by the entire system. Using a regulated 12V supply, the Arduino and LCD together draw 330mA resulting in 4W. The other peripherals, including the LTC6804, current sensor, temperature sensor, and cutoff circuit, draw relatively small amounts of current (less than 100mA total) that do not significantly impact overall power consumption. While current estimates predict the system meets specifications, the system includes room to improve efficiency in all subsystems.

The current state of the BMS serves as a proof of concept that allows many options for future development. The device allows programmability and interfacing that other systems lack, accomplishing the main objective of the design. Additional software can increase the number of features of the PBMS, like an adjustable balance threshold voltage or a "sleep" mode for the LCD to improve efficiency. The design permits the comparison of multiple SOC calculation methods to select the best option, such as those provided by Xiong et al. [28] and Wilkinson [29]. Other possible additions include graphs and plots that facilitate battery monitoring and offer useful information to the user. The Nextion display offers excellent potential for expansion during development; GUI development requires no prior experience, and the Nextion Editor facilitates troubleshooting. However, the screen contains excessive capabilities for the function it provides to the project; to meet the price needs of a consumer, the design must port the GUI from the Nextion display to a smaller, simpler, and lower cost display. An alternative method includes developing an application for a cellphone; this reduces the size of the device and cost to manufacture over time but increases the required investment up front.

Future designs may reconsider utilizing the LTC6804, as search engine results indicate many cases of communication issues [4]. While it provides useful functionality by measuring and balancing cell voltages, the unneeded features, excessive price, and communication problems encourage research into alternative methods. For example, a standalone ADC can measure all cell voltages, costs less to manufacture, and easily integrates with the Arduino, but they generally do not contain the ability to balance the cells or trigger an over/undervoltage fault. These functions require additional systems in hardware or software. Future versions of the PBMS also incorporate all electronics on a single PCB; this simplifies and strengthens the connections while increasing safety and reducing volume. This also permits higher current specifications than listed in Chapter 2 and reduces the number of possible points of failure. Many

of the included components feature higher limits than required, but other bottlenecks, like cable diameter and connection quality, limit the maximum safe current. The next and final chapter wraps up thoughts about the PBMS design and development.

Chapter 6. Conclusions

Currently in the troubleshooting phase, the Programmable Battery Management System remains incomplete. Several factors inhibit progress as expected, stemming from unanticipated design modifications to a global pandemic. Due to the number of subsystems incorporated in the PBMS and extra care required when working with lithium batteries, excessive fabrication provides a significant time sink in this project that ultimately prevents completion. For example, constructing the battery from bare cells adds unnecessary development time to the project; an off the shelf battery can provide the same utility for the purposes of this project with slight modification. Additionally, any setback in assembly delays evaluation and integration with the rest of the system. Manual soldering of all components leaves many opportunities for mistakes costing both time and money. Weeks spent on a failed approach add value to the project from the knowledge gained but hinder progress toward a fully functioning system. Failure to consider the connections between all subsystems before beginning to build the system introduces significant slow downs during development; thus, future designs of all types require closer attention to connectors early in the design phase.

In general, developing a successful programmable battery management system requires significantly more time investment than anticipated. Designing a BMS using multiple subsystems seems simple initially, but the intricacies of each subsystem hinder progress. Struggling for a long period with one system forces attention to time management such that no one part of the project gets neglected. The current design succeeds in its main objective despite failing to meet some requirements; it adds the programmable functionality to a BMS but requires additional troubleshooting and refinement. However, each success and failure provide valuable learning opportunities that ultimately benefit both the project and designer eventually. As an electric bicycle enthusiast, attempting to construct a PBMS offers invaluable insight into the field of lithium battery management systems. The field features a seemingly infinite number of rabbit-holes to explore from advanced calculation methods for evaluating the remaining charge of a cell to methods of reducing noise in ADC measurements on the Arduino. The new experiences from this project include ordering a custom PCB from the Gerber files, developing a graphical interface using a touchscreen display, and exposure to the vast field of power electronics design, which is a new interest and potential career focus. This project lays an excellent foundation for future BMS development and expansion by providing a platform open to refinement from all directions. After addressing issues with the LTC6804 cell measurements, opportunities for improvement exist from changing the method of current measurement to incorporating all components on one PCB, adding new features in software, and designing an enclosure that houses a finished system.

References

[1] Occupational Safety and Health Administration (OSHA), "Preventing Fire and/or Explosion Injury from Small and Wearable Lithium Battery Powered Devices" Safety and Health Information Bulletin 2019. Available: https://www.osha.gov/dts/shib/shib011819.html

[2] Q. Wang, P. Ping, X. Zhao, G. Chu, J. Sun, and C. Chen, "Thermal Runaway Caused Fire and Explosion of Lithium Ion Battery," *Journal of Power Sources*, vol. 208, no. 24, pp. 210–224, 2012.

[3] Buchmann, Isidor, "Battery University," Cadex Electronics Inc. 2020. Available: https://batteryuniversity.com/

[4] D. Andrea, *Battery management systems for large lithium battery packs*. Norwood: Artech House Publishers, 2010. See also: http://liionbms.com

[5] BesTech Power, Programmable Battery Management System Product Description, PN HCX-D270. [Accessed 16 Mar. 2020] Available: http://www.bestechpower.com/communicationbms/BMS-D270.html

[6] Analog Devices, "Multicell Battery Stack Monitor," LTC6804 Datasheet, Rev. B, June 2019. [Datasheet] Available: https://www.analog.com/media/en/technical-documentation/data-sheets/LTC6802-1.pdf

[7] Texas Instruments, BQ76PL536A Datasheet. June 2011. [Datasheet] Available: http://www.ti.com/lit/ds/symlink/bq76pl536a.pdf [Accessed 3 Feb. 2020].

[8] JTT Electronics, "Portable Series Lithium-ion Battery Management System Data Sheet" 2013 [Datasheet]. Available: http://www.jttelectronics.com/files/pdf/datasheets/DN000223.C-P-Series-BMS-Data-Sheet.pdf

[9] Orion BMS, Orion BMS2 Technical Specifications Sheet. [Accessed 16 Mar. 2020] Available: https://www.orionbms.com/downloads/documents/orionbms2_specifications.pdf

[10] DeSando, Michael, "Universal Programmable Battery Charger with Optional Battery Management System," June 2015. DOI: https://doi.org/10.15368/theses.2015.68 Available: https://digitalcommons.calpoly.edu/theses/1409

[11] Chad [unknown], "Charging - research and methodology" *AccuBattery*. Jan. 2020. [Online article]. Available: https://accubattery.zendesk.com/hc/en-us/articles/210224725-Charging-research-and-methodology [Accessed 3 Feb. 2020].

[12] McManus, M.C., 2011, "Environmental consequences of the use of batteries in low carbon systems: The impact of battery production," Applied Energy 2012 (93) p. 288-295.

[13] W.-Y. Chang, “The State of Charge Estimating Methods for Battery: A Review,” *ISRN Applied Mathematics*, vol. 2013, Jul. 2013.

[14] Samsung SDI, “Introduction of INR18650-25R,” Oct. 2013. [Datasheet]. Available: https://www.powerstream.com/p/INR18650-25R-datasheet.pdf**.**

[15] R. Ford and C. Coulston, *Design for Electrical and Computer Engineers*, McGraw-Hill, 2007, p. 37

[16] K. W. E. Cheng, B. P. Divakar, H. Wu, K. Ding, and H. F. Ho, “Battery-Management System (BMS) and SOC Development for Electrical Vehicles,” *IEEE Trans. on Vehicular Technology*, vol. 60, no. 1, pp. 76–88, Jan. 2011.

[17] Buchmann, Isidor, “What Causes Li-Ion to Die?” *batteryuniversity.com*, Aug. 2017. [Online]. Available: https://batteryuniversity.com/learn/article/bu_808b_what_causes_li_ion_to_die.

[18] C. Mikolajczak, M. Kahn, K. White, and R. T. Long, “Lithium-Ion Fire Hazard Assessment,” *Fire Protection Research Foundation*, Jul. 2011.

[19] D. M. Caditz, “Arduino LTC6804 BMS - Part 1: Main Board,” *Instructables*, Dec. 2019. [Online]. Available: https://www.instructables.com/id/Arduino-LTC6804-Battery-Management-System/.

[20] J. Gammell, (2010, July 25). Professional SMT Soldering: Hand Soldering Techniques - Surface Mount [Video file]. Available: https://www.youtube.com/watch?v=5uiroWBkdFY

[21] Vishay Siliconix, “Automotive P-Channel 60V (D-S) 175 °C MOSFET,” SQM120P06-07L Datasheet. Rev. B, 12 July 2012. [Datasheet] Available: https://www.vishay.com/docs/67026/sqm120p0.pdf

[22] Allegro Microsystems, “ACS712: Fully Integrated, Hall-Effect-Based Linear Current Sensor IC with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor,” ACS712 Datasheet. Rev. 19, 30 Jan. 2020. [Datasheet] Available: https://www.allegromicro.com/en/products/sense/current-sensor-ics/zero-to-fifty-amp-integrated-conductor-sensor-ics/acs712

[23] Ruslan Koptiev, “ACS712-arduino,” *GitHub*, 11 Sep. 2018. [Online]. Available: https://github.com/rkoptev/ACS712-arduino.

[24] Microchip, “Low-Power Linear Active Thermistor ICs,” MCP9700 Datasheet. Rev. G, June 2016. [Datasheet] Available: http://ww1.microchip.com/downloads/en/DeviceDoc/20001942G.pdf

[25] Microchip, “IC Temperature Sensor Accuracy Compensation with a PIC Microcontroller,” MCP9700 Application Note AN1001. 8 Sep. 2015. [Online] available: http://ww1.microchip.com/downloads/en/Appnotes/00001001C.pdf

[26] Ayush Agrawal, “BMS-Arduino,” *GitHub*¸ 26 Aug. 2016 [Online]. Available: https://github.com/ayushagrawal/BMS-Arduino

[27] Pololu Corporation, “12V, 600mA Step-Down Voltage Regulator D36V6F12,” 3796 Product Page. [Datasheet] Available: https://www.pololu.com/product-info-merged/3796

[28] Xiong, R., He, H., Sun, F., Zhao, K., “Evaluation on State of Charge Estimation of Batteries With Adaptive Extended Kalman Filter by Experiment Approach,” *IEEE Trans. on Vehicular Technology*, 62(1), pp.108-117, Jan. 2013. [Online]. Available at: https://ieeexplore.ieee.org/document/6323045 [Accessed 3 Feb. 2020].

[29] J. P. Wilkinson, “Systems and methods for estimation and prediction of battery health and performance,” U.S. Patent 10 209 314, Nov. 21, 2016.

Appendix A – ABET Senior Project Analysis

Project Title: Programmable Battery Management System

Student's Name: Jason Erbert

Advisor's Name: Prof. David Braun Initial: Date:

1. Summary of Functional Requirements
   The programmable battery management system (BMS) provides additional capabilities and functionality to any small scale lithium battery pack (<1000W, 11V to 50V, up to 20A). Maximum charge percentage, minimum voltage, and maximum current among other quantities directly influence the battery's health and longevity. The programmable BMS allows the user to set these values and change all settings without removing the device from the battery pack. The device protects the user from dangerous events like thermal runaway by detecting hazardous conditions and intervenes by turning off the battery output.
2. Primary Constraints
   The table in Chapter 2 lists the design specifications, the main source of the project constraints. The required voltage and current levels complicate component selection and force design modification (see explanation of cutoff circuitry in Chapter 5.3). The unexpected modifications early in the project delay progress and leave insufficient time for complete evaluation of all subsystems. The BMS device primarily targets small scale applications (<1000W) to minimize risk of injury during development. This restricts the overall dimensions of the product which adds complexity to the design process. Efficiency limits the device's utility at smaller scales – interface electronics consume relatively low power and remain constant as the product scales to larger power capabilities.

   Challenges to developing the programmable BMS primarily include time restraints, excessive fabrication, small part size, expense, and lack of documentation. The overall system complexity limits what is achievable by one project, with five subsystems that each require design, construction, troubleshooting, and evaluation. The amateur equipment used for this project creates difficulty in evaluation, occasionally requiring a creative approach that increases the required time allocation. An example of this includes developing a method to verify the temperature sensor accuracy using household objects. In addition, soldering mistakes and equipment failures due to small component size and inadequate tooling result in unexpected project delays. Three LTC6804 BMS chips burnt during prototyping, requiring new BMS PCBs. An attempt to save cost by using prototyping board instead of proceeding with a custom PCB significantly set back production of the cell balancing hardware and project development. In addition, the libraries supplied by LT lack sufficient documentation to quickly program the LTC6804.

While the provided functions help code development, the LTC6804 remains relatively complex and requires significant time investment to fully understand all features and intricacies. Failure to foresee the additional development associated with LTC6804 itself and supporting hardware severely constrain the success of the project.

Additionally, the COVID-19 global pandemic continues to influence the entire world in many ways throughout the course of the project. Numerous setbacks due to the pandemic prevent completion of the PBMS as initially envisioned; development interruptions and supplies-related issues slow the rate of progress. Equipment access limits the scope of evaluation to what is achievable with an amateur home setup. The timeframes estimated during the project planning phase change often due to pandemic related uncertainty.

3. Economics
    a. Human Capital
    A sufficiently useful BMS requires over 100 hours of work to design and build. A conservative estimate using Equation (6) from Coulston [1] suggests the project requires at least 150 hours due to complexity and only one engineer. This includes design, production, and evaluation of both hardware and software. Once completed, the device generates demand for human capital in manufacturing everything from the individual ICs to complete lithium battery powered systems. The device has the potential to influence the global labor market by encouraging transition of short distance delivery services to light EV as the technology advances.
    b. Financial Capital
    Since the project focuses on small scale devices, the battery and electronics used in development do not require significant financial capital. Components incorporated in the design cost less than $500 as shown in Appendix C. The self funded project requires no outside capital; the design prioritizes commonly found components, with many salvageable from previous school projects at little cost. However, the resulting product's additional features, including the screen and microcontroller, require significant financial capital relative to a simple BMS. This increases the upfront cost to the consumer and cost of manufacturing.
    c. Manufactured Capital
    Since the subsystems mainly consist of entire modules, a significant portion of the manufactured capital stems from the processes used to create the components including the LCD screen, microcontroller, and batteries. This project requires additional manufacturing capital during development as the design evolves; much of this consists of manual assembly of each subsystem. Development of the prototype device uses low cost and freely available machines such as soldering irons, multimeters, amateur oscilloscopes, and basic computers. Assembly of a

complete system requires additional hardware such as wire, solder, battery connectors, and an enclosure.

d. Natural Capital

As the current state of the art in battery storage chemistry, lithium experiences high demand. Current mining tactics show extremely detrimental effects to the environment [2]. The previously mentioned subcomponents generate pollution while shipping from overseas. Each manufactured component comes at a cost to the environment beginning with material extraction and ending with disposal after usage.

e. Cost and Benefit Accrual

Most of the costs associated with BMS production arise in obtaining the hardware (excluding labor). High quality batteries and interfaces demand higher prices but provide maximum safety and benefits. Development costs occur predominately in Spring 2020 and extend to Fall 2020. Benefits to the engineer, primarily knowledge gained, accrue throughout the project's entire lifecycle beginning with project planning and extending beyond the end of EE462. Each phase of development provides a new learning opportunity that helps to improve future designs.

TABLE VI:
PREDICTED PARTS AND LABOR COSTS

	Price	**Qty**	**Total**
Labor	$30/hr	150 hrs	$4500
Parts			
Electrical Components			
Resistors, Capacitors	$20	1	$20
Interface Hardware	$10	10	$100
Microcontroller	$25	2	$50
Battery Management Modules	$10	5	$50
Batteries	$10	10	$100
Wires, Connectors	$50	1	$50
Enclosure	$50	1	$50
Project Cost			$4870

$$Cost = \frac{Cost_a + 4Cost_M + Cost_B}{6}$$

Equation (6) from Coulston [1]

The total predicted cost for this project sums to $4,870 and splits between labor and equipment cost, as in Table VI above. Time commitment and price estimations result from equation (6) from Coulston [1]. The first term in the

numerator represents the optimistic estimation; the second term the realistic estimation; and the third term a pessimistic estimation. A fair rate for a fourth year engineering intern in California averages $30 per hour and provides the expected labor costs. According to Prof. Dolan, students should expect to spend around 100 hours on a senior project. The individual nature and ambitious goals of the project require at least an additional 50 hours. Using time instead of cost in the equation above results in 150 hours required. Thus, most of the expected project cost comes from labor at $4500. Initial hardware price estimates predict approximately $370 for all subsystems and components. They also assume free access to supporting hardware, including solder and tools. Actual development costs exceed predictions as demonstrated in Table VIII in Chapter 4.2. Chapter 4.2 also explains the sources of additional expense.

f. Timing

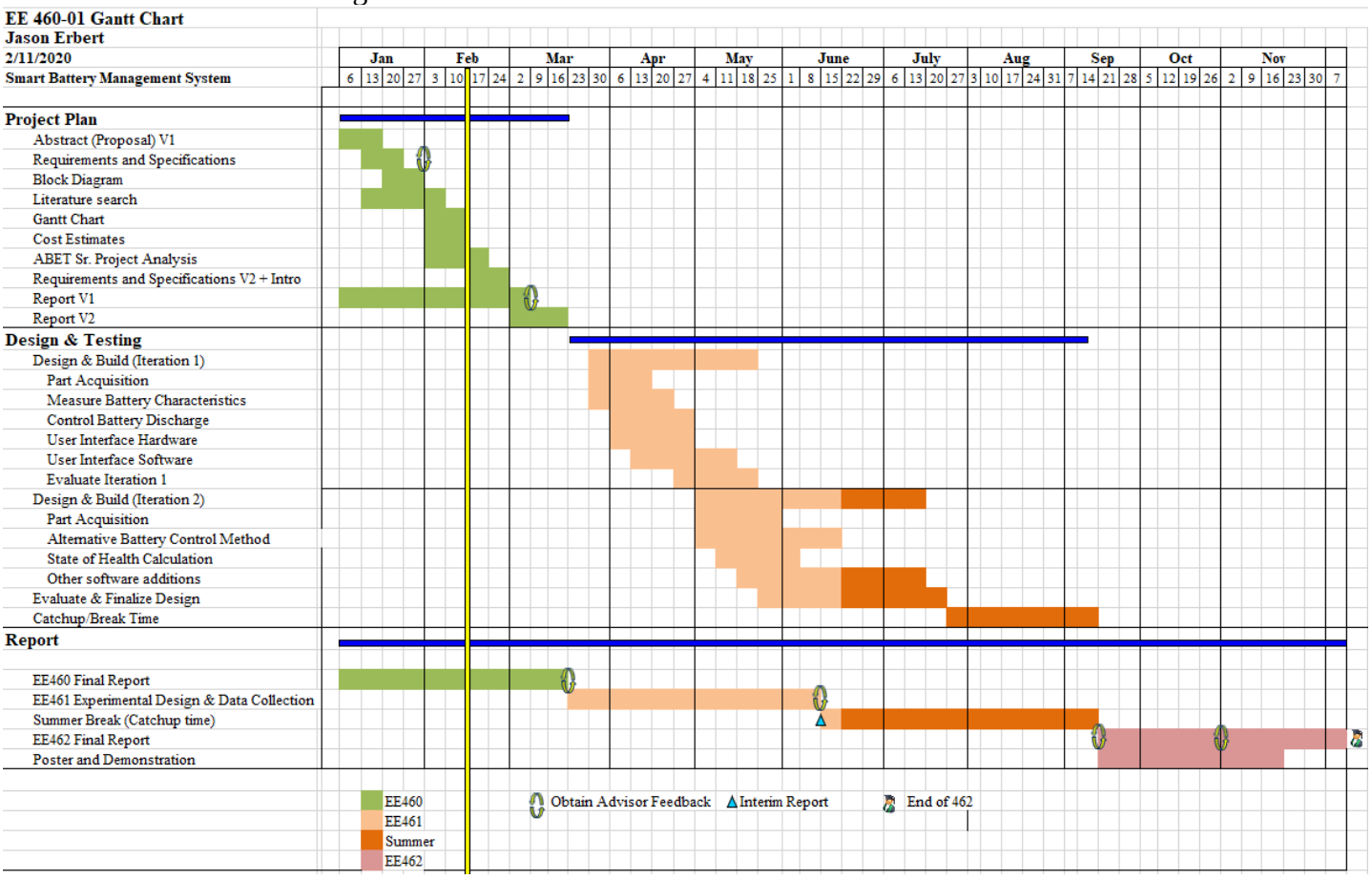

Figure 22: Preliminary Gantt chart for programmable BMS design and documentation process.

The project spans approximately 11 months as predicted in Figure 22 above; excluding summer quarter leaves 8 months. Initial estimates budget 6 hours per week of development time. A physical product emerges in Fall 2020 during EE462. Reevaluation at the end of EE461 and

numerous setbacks during EE462 result in the updated Gantt chart in Figure 23 below. Chapter 4.1 contains the Gantt chart created during EE461 as well as explanations of time-reallocation.

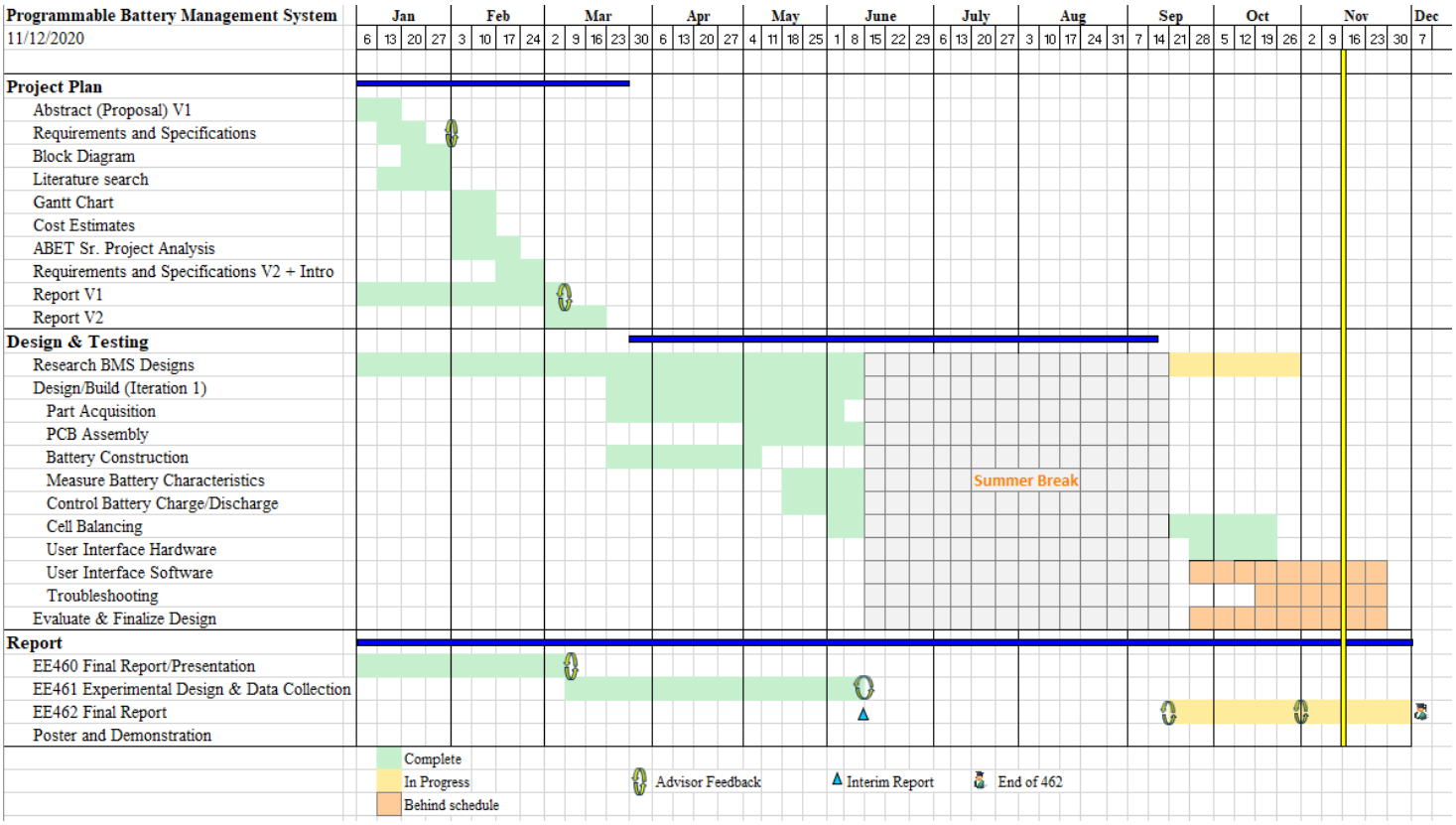

Figure 23: Actual Gantt chart of design process

4. If manufactured on a commercial basis
    a. Estimated number of devices sold per year: 100,000
    High volumes of lithium batteries are produced annually to meet the growing demand for electric devices. A small but present percentage of these users desire direct control over the batteries' characteristics.
    b. Estimated manufacturing cost for each device: $50-$100
    The first device costs the most, but after optimizing the production process, the price decreases dramatically.
    c. Estimated purchase price for each device: $100-$200
    d. Estimated profit per year: $5,000,000 - $10,000,000
    e. Estimated operating cost for user: Less than $5 per year at 50Wh daily consumption and $.20/kWh.
5. Environmental
    Lithium is a relatively scare element in the Earth's crust found in small quantities. Extraction requires large mining operations with severe impacts to the local environment [2]. In addition, lithium batteries and electronics production occurs predominately in

countries including China where manufacturing operations have significantly harmed on the environment. Maximizing battery longevity and utility helps decrease the demand for additional lithium and lessens the environmental impact from battery production. Devices utilizing lithium batteries like cars benefit from this device, which further helps the global ecosystem by facilitating the transition from fossil fuel powered transportation to electric.

6. Manufacturability

   Lithium batteries provide large amounts of power which carry danger during handling. Assembling and testing the device requires proper safety precautions to prevent risk of injury from fires or explosions. Production faces many challenges in the current economic climate due to COVID-19. Businesses across the world are shut down and supply chains are interrupted. Electric vehicle components primarily originate overseas where travel is currently limited, possibly decreasing the available supply and/or increasing prices until a vaccine permits full production. Budgetary and time constraints work together to inhibit manufacturability during project development; production of a satisfactory device requires delicate balancing of functionality/feature list, time, and cost. Small component size allows the potential for solder bridges on the LTC6804; connecting the wrong pins can lead to catastrophic failure.

7. Sustainability

   This product encourages reuse intrinsically; the versatile design allows a wide array of applications and adjustability after implementation with a battery. This cuts down on waste and extends the planet's limited resources. While this holds true for devices up to the specified voltage and current limit, future development is required to allow the BMS to have higher overall limits and decrease the need for different devices. Society's transition to battery powered transportation also influences the infrastructure; increased demand on the electrical grid arises with more battery charging from EVs. With the grid in California experiencing issues each year, this issue need a solution before widespread EV adoption approaches feasibility.

8. Ethical

   From a utilitarian point of view, the Programmable BMS attempts to achieve the greatest good for the greatest number. It provides users who value safety over performance direct control over the battery to prioritize this; users who desire increased longevity can achieve their goals with the same device. However, while many benefit from the device, some may experience negative effects. Increased demand for low cost manufacturing of batteries and electronics enables continued abuse of workers from various countries; the number of affected workers could possibly outnumber those who benefit from the device, thus failing from a utilitarian standpoint. Additionally, the safety of the public is a top priority during the design process, as described by the IEEE code of Ethics #1 guideline [3]. However, the design process must also consider the potential negatives of allowing extreme control over batteries to potentially unexperienced users. For example, a user who removes the temperature limit out of ignorance puts all those around themselves at

risk. An ethical solution could include temporary preset limits within the device; if the user wishes to change the limits, a one-time process initiates developer mode and removes the limits. This product also offers direct interaction with important technology in society's daily life; as a byproduct, users receive lithium battery education. The device offers a unique means of informing the public about the dangers, capabilities, and operation of lithium batteries (see IEEE Code of Ethics #2 [3]).

Additionally, the design requires improved connections between subsystems; prototyping allows development at low power levels but a consumer product requires extreme attention to connection quality and safety at full power. Public usage mandates a full evaluation of the device and analysis of the possible modes of failure to prevent possible injury (see IEEE Code of Ethics II).

The higher up front cost relative to a traditional BMS brings the potential to exclude users on a tight budget; devices that drastically increase safety should be available regardless of fortune, but economic feasibility poses an issue due to the difference in production cost.

9. Health and Safety

   All devices incorporating lithium battery technology carry a certain amount of risk. The devices store large amounts of energy which causes overheating, fires, and even explosions according to a recent OSHA Safety and Health Information Bulletin [4]. Precautions must be taken during product testing and manufacturing to limit risk of injury. The extreme fire danger mandates extra attention to accidental shorts across battery terminals, with a type B.C. extinguisher on standby. It also requires the assembler to avoid adding excessive heat to the battery cells during pack construction. Production of the PBMS involves soldering with lead based solder, which causes lead poisoning if ingested in high quantities. Assembly must occur in a well ventilated area where the assembler avoids inhaling any fumes. In addition, all connections must meet the current and voltage specifications listed in Chapter 2, and the design requires thorough safety evaluation before public usage. Users of the product must take similar precautions, especially if operating the device publicly. As mentioned previously, punctures in the side walls, excessive current, high temperatures, and numerous other events in lithium batteries result in fire [4].

10. Social and Political

    Lithium battery technology acts as one of the main bottlenecks in the transition from petroleum powered to electric vehicles. As battery technology improves and costs decrease, the demand for electric vehicles grows while the demand for oil decreases. The oil industry across the globe, with a history of significant political turmoil, suffers if electric vehicles adequately replace petroleum powered vehicles. The device has a large impact on the public; it facilitates the transition to electric transportation. As transportation becomes more dependent on lithium batteries, the demand for the metals required to manufacture the batteries increases. Certain geographic locations with rich

lithium, copper, and nickel deposits may see political turmoil and conflict regarding the rights to the materials.

The primary stakeholders include small scale lithium battery operators and manufacturers. Direct benefits of the PBMS to the user include reduced waste, longer battery lifetime, and improved safety. However, the larger up front cost relative to a typical BMS reduces accessibility and increases difficulty to manufacturers. Indirect impacts received by the users include awareness and education about battery safety and optimization, as having programmable control requires the user to understand the potential for harm from and to the battery.

11. Development
    a. New Tools and Techniques

       New methods of managing lithium batteries undergo research each day. As the PBMS proceeds through development, various approaches are considered including known reliable methods and relatively new, experimental methods. New technology learned for this project consists of the graphical user interface created on the Arduino using the Nextion display, the cutoff circuit that serves as a replacement for the relays, and the method of spotwelding the 18650 cells using nickel strip. The unexpected problems from each step of the development cycle documented in Chapter 5 provided excellent opportunities to learn about the field of power electronics, which now poses as a potential career focus. Research completed during EE460 and EE461 assisted me in my summer internship project between EE461 and EE462. Industry research throughout the course of the project continues to produce new commercial battery management systems and inspires ideas for alternative battery control designs.

    b. Literature Search:

[1] R. Ford and C. Coulston, *Design for Electrical and Computer Engineers*, McGraw-Hill, 2007, p. 37

[2] McManus, M.C., 2011, "Environmental consequences of the use of batteries in low carbon systems: The impact of battery production," Applied Energy 2012 (93) p. 288-295.

[3] *IEEE Std 1233, 1998 Edition*, p. 4 (10/36), DOI: 10.1109/IEEESTD.1998.88826
See also: IEEE.org. *IEEE (Institute of Electrical and Electronics Engineers) Code of Ethics*. [online] Available at: https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed 17 Feb. 2020].

[4] Occupational Safety and Health Administration (OSHA), "Preventing Fire and/or Explosion Injury from Small and Wearable Lithium Battery Powered Devices" Safety and Health Information Bulletin (2019). Available: https://www.osha.gov/dts/shib/shib011819.html

[5] Chad [unknown]. “Charging - research and methodology” *AccuBattery*. Jan. 2020. [Online article]. Available: https://accubattery.zendesk.com/hc/en-us/articles/210224725-Charging-research-and-methodology [Accessed 3 Feb. 2020].

- This article corrects common misconceptions regarding lithium battery charging and suggests methods for improving battery longevity. I reference this in the first chapter when explaining the advantages of increased charging control.
- This article provides primary sources for each claim, and analyzes the integrity of the data from its sources in-text. The unknown author appears to work for the company that produces the AccuBattery smartphone application and provides convincing arguments with evidence for his claims.

[6] K. W. E. Cheng, B. P. Divakar, H. Wu, K. Ding, and H. F. Ho, “Battery-Management System (BMS) and SOC Development for Electrical Vehicles,” *IEEE Trans. on Vehicular Technology*, vol. 60, no. 1, pp. 76–88, Jan. 2011.

- The journal article includes a functional block diagram of a BMS with explanations of each block. It also features mathematical methods for determining state of charge. This helped develop the initial vision for the BMS structure in Chapter 3.
- Features 232 paper citations and 7 patent citations according to IEEE. Carefully explains claims using diagrams and evidence.

[7] T. Dragicevic, J. M. Guerrero, J. C. Vasquez, and D. Skrlec, “Supervisory Control of an Adaptive-Droop Regulated DC Microgrid With Battery Management Capability,” *IEEE Trans. on Power Electronics*, vol. 29, no. 2, pp. 695–706, Feb. 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6497633 [Accessed 3 Feb. 2020]

- The authors provide a unique and robust method of regulating cell voltages during charging. They provide clear and thorough explanations with diagrams and simulation results.
- The authors’ credentials and support to back up claims inspire confidence. The article has 390 citations.

[8] Einhorn, M., Roessler, W. and Fleig, J., “Improved Performance of Serially Connected Li-Ion Batteries With Active Cell Balancing in Electric Vehicles,” *IEEE Trans. on Vehicular Technology*, 60(6), pp.2448-2457, July 2011. [Online]. Available: https://ieeexplore.ieee.org/document/5766056 [Accessed 3 Feb. 2020].

- Includes detailed discussion of active cell balancing methodologies. They use a similar experimental configuration to my planned prototype.
- IEEE claims 159 citations to this paper. The authors also include numerous in-text citations to other well reviewed papers.

[9] Xiong, R., He, H., Sun, F., Zhao, K., "Evaluation on State of Charge Estimation of Batteries With Adaptive Extended Kalman Filter by Experiment Approach," *IEEE Trans. on Vehicular Technology*, 62(1), pp.108-117, Jan. 2013. [Online]. Available at: https://ieeexplore.ieee.org/document/6323045 [Accessed 3 Feb. 2020].

- The authors reinforce the safety aspect of lithium battery control and provide an advanced approach to state of charge calculation.
- IEEE published the article which has 175 citations since 2013. The main author's affiliation with electric vehicle research increases credibility, and they have a number of publications with hundreds of citations each.

[10] Rahimi-Eichi, H., Ojha, U., Baronti, F., Chow, M., "Battery Management System: An Overview of Its Application in the Smart Grid and Electric Vehicles," *IEEE Industrial Electronics Magazine*, 7(2), pp.4-16, June 2013. [Online]. Available at: https://ieeexplore.ieee.org/document/6532486 [Accessed 3 Feb. 2020].

- This article offers examples and specific requirements of a BMS to maximize battery efficiency. It also provides methods, results, and suggestions for improvements.
- Over 300 citations by papers and 1 patent citation since 2013 demonstrate the information's thorough review.

[11] J. P. Wilkinson, "Systems and methods for estimation and prediction of battery health and performance," U.S. Patent 10 209 314, Nov. 21, 2016.

- The patent provides computerized implementations of battery state calculations, which may find application in the PBMS. It introduced me to the problem of calculating the SOC, which is not as simple as measuring the voltage.
- This patent references a previously cited paper and was produced by Idaho National Laboratory.

[12] BQ769x0 Multicell Battery Monitor System [Author Unknown]. Texas Instruments. Oct. 2013. [Datasheet] Available: http://www.ti.com/lit/ds/slusbk2h/slusbk2h.pdf [Accessed 3 Feb. 2020].

- The document contains useful descriptions of BMS and their subcomponents. It lists applications that overlap with the intended usage of the Programmable BMS.
- Texas Instruments has an excellent reputation as a trustworthy supplier for electronic parts and documentation. This part a long history and recently updated datasheet (last revised in 2019), indicating the reliability of the information.

[13] BQ76PL536A Datasheet [Author Unknown]. Texas Instruments. June 2011. [Datasheet] Available: http://www.ti.com/lit/ds/symlink/bq76pl536a.pdf [Accessed 3 Feb. 2020].

- The datasheet provides technical designs, specifications, and functionalities for a similar device (expandable BMS), including many concepts possible to incorporate in the Programmable BMS.
- Again, Texas Instruments is a reliable supplier of both parts and information. This datasheet has nearly a 10 year history with a recent update within the past 5 years.

[14] Jiang, J., Zhang, C. 2015. *Fundamentals and application of lithium-ion battery management in electric drive vehicles*. Wiley. [Book].

- This book contains background information necessary to understand lithium battery management. Each chapter includes thorough explanations dedicated to important topics such as battery state of charge estimation. The book also features a chapter focused on technologies used to implement BMS and applications of such devices.
- The recent publish date permits the assumption that the book contains new and relevant information. The author uses mathematical proofs to explain concepts and has over 50 publications available on IEEE, with some featuring over 100 citations.

[15] BQ78350-R1 Datasheet [Author Unknown]. Texas Instruments. Aug. 2015 (Rev Nov. 2018) [Datasheet] Available: http://www.ti.com/lit/ds/symlink/bq78350-r1.pdf [Accessed 24 Feb. 2020].

[16] DeSando, Michael. "Universal Programmable Battery Charger with Optional Battery Management System," June 2015. DOI: https://doi.org/10.15368/theses.2015.68 Available at: https://digitalcommons.calpoly.edu/theses/1409

Appendix B – Arduino Code

```
//PBMS V1.0
//1 Dec. 2020
//Jason Erbert
//This code builds on the LTC6804 Arduino BMS from David Caditz:
//https://www.instructables.com/Arduino-LTC6804-Battery-Management-System/

#include <SoftwareSerial.h>
#include <Average.h>
#include <UserInterface.h>
#include <Linduino.h>
#include <LT_SPI.h>
#include <LTC68042.h>
#include <ACS712.h>     //ACS712 Library from Ruslan Koptiev:
https://github.com/rkoptev/ACS712-arduino

#define currentPin A3
#define tempPin A4

ACS712 Isensor(ACS712_30A, currentPin);
SoftwareSerial Serial1(3, 2); // RX, TX

#define TOTAL_IC  1            // Number of ICs in the isoSPI network
LTC6804-2 ICs must be addressed in ascending order starting at 0.

//Battery measurement variables
float Vout = 34.8;
float CellV = 3.48;
float Iout = 5.2;
int T = 25;
int cycles = 10;
int charge = 65;

/***** Pack and sensor characteristics *****/
float Imax = 2.0;          // Maximum battery current(amps) before cutoff
int Tmax  = 30;                   // Maximum pack temperature (deg C) before
cutoff
float Vmin = 33.0;                  // Minimum allowable cell voltage (10S
cells). Depends on battery chemistry.
float Vmax = 42.0;                  // Maximum allowable cell voltage.
Depends on battery chemistry.
float CELL_BALANCE_THRESHOLD_V = 3.3;     // Cell balancing occurrs when
voltage is above this value

/******** Arduino digital pin definitions ********/
int dischargeCutoffPin = 4;

/********  Variables for tracking cell voltages and states **************/
int overCharge_state = LOW;           // Over charge state. HIGH = relay on,
LOW = relay off
int underCharge_state = LOW;          // Under charge state. HIGH = relay on,
LOW = relay off
int overTemp_state = LOW;             // Over temperature state. HIGH = relay
on, LOW = relay off
int overCurrent_state = LOW;          // Over current state. HIGH = relay on,
LOW = relay off
```

```
int cutoff_state;
int cellMax_i;                          // Temporary variable for holding index
of cell with max voltage
int cellMin_i;                          // Temporary variable for holding index
of cell with min voltage
float cellMin_V;                       // Temporary variable for holding  min
measured cell voltage
float cellMax_V;                       // Temporary variable for holding  max
measured cell voltage
float minV1 ;
float maxV1 ;

int error = 0;
unsigned long  tstart;

/*****************************************************
  Global Battery Variables received from 6804 commands
  These variables store the results from the LTC6804
  register reads and the array lengths must be based
  on the number of ICs on the stack
 *****************************************************/


uint16_t cell_codes[TOTAL_IC][12];
/*!<
  The cell codes will be stored in the cell_codes[][12] array in the
following format:

  |  cell_codes[0][0]| cell_codes[0][1] |  cell_codes[0][2]|    .....     |
cell_codes[0][11]|  cell_codes[1][0] | cell_codes[1][1]|  .....   |
  |------------------|------------------|------------------|--------------|--
----------------|-------------------|-----------------|----------|
  |IC1 Cell 1        |IC1 Cell 2        |IC1 Cell 3        |    .....     |
IC1 Cell 12      |IC2 Cell 1         |IC2 Cell 2       | .....    |
****/

uint16_t aux_codes[TOTAL_IC][6];
/*!<
  The GPIO codes will be stored in the aux_codes[][6] array in the following
format:

  |  aux_codes[0][0]| aux_codes[0][1] |  aux_codes[0][2]|  aux_codes[0][3]|
aux_codes[0][4]|  aux_codes[0][5]| aux_codes[1][0] |aux_codes[1][1]|  .....
|
  |-----------------|-----------------|-----------------|-----------------|--
---------------|-----------------|-----------------|---------------|---------
--|
  |IC1 GPIO1        |IC1 GPIO2        |IC1 GPIO3        |IC1 GPIO4
|IC1 GPIO5        |IC1 Vref2        |IC2 GPIO1        |IC2 GPIO2      |
.....    |
*/

uint8_t tx_cfg[TOTAL_IC][6];
/*!<
  The tx_cfg[][6] stores the LTC6804 configuration data that is going to be
written
```

```
  to the LTC6804 ICs on the daisy chain. The LTC6804 configuration data that
will be
  written should be stored in blocks of 6 bytes. The array should have the
following format:

  |  tx_cfg[0][0]| tx_cfg[0][1] |  tx_cfg[0][2]|  tx_cfg[0][3]|
tx_cfg[0][4]|  tx_cfg[0][5]| tx_cfg[1][0] |  tx_cfg[1][1]|  tx_cfg[1][2]|
.....    |
  |--------------|--------------|--------------|--------------|--------------
|--------------|--------------|--------------|--------------|-----------|
  |IC1 CFGR0     |IC1 CFGR1     |IC1 CFGR2     |IC1 CFGR3     |IC1 CFGR4
|IC1 CFGR5     |IC2 CFGR0     |IC2 CFGR1     | IC2 CFGR2    |  .....    |

*/

uint8_t rx_cfg[TOTAL_IC][8];
/*!<
  the rx_cfg[][8] array stores the data that is read back from a LTC6804-1
daisy chain.
  The configuration data for each IC  is stored in blocks of 8 bytes. Below
is an table illustrating the array organization:

  |rx_config[0][0]|rx_config[0][1]|rx_config[0][2]|rx_config[0][3]|rx_config[
0][4]|rx_config[0][5]|rx_config[0][6]  |rx_config[0][7]
|rx_config[1][0]|rx_config[1][1]|  .....    |
  |---------------|---------------|---------------|---------------|----------
-----|---------------|-----------------|----------------|---------------|----
-----------|-----------|
  |IC1 CFGR0      |IC1 CFGR1      |IC1 CFGR2      |IC1 CFGR3      |IC1 CFGR4
|IC1 CFGR5      |IC1 PEC High     |IC1 PEC Low     |IC2 CFGR0      |IC2 CFGR1
|  .....    |
*/

/*!**************************************************************************
  \brief  Inititializes hardware and variables
 ***************************************************************************/
void setup()
{
  pinMode(dischargeCutoffPin,OUTPUT);
  pinMode(tempPin, INPUT);

  Serial.begin(9600);
  while(!Serial); // wait for serial port to connect. Needed for native USB
port only
  Serial1.begin(9600);   //LCD communication

  Isensor.calibrate();

  LTC6804_initialize();  //Initialize LTC6804 hardware
  init_cfg();            //initialize the 6804 configuration array to be
written
  delay(1000);

  overCurrent_state = HIGH;
  tstart = millis();

  //turn on battery output at startup
```

```
  digitalWrite(dischargeCutoffPin, HIGH);
  Serial1.print("t1.txt=\"ON\"");
  scmd();
  Serial1.print("t1.bco=2016");   //set output indicator background to green
  scmd();
}

void loop()
{
  //measure current
  Iout = 0.05 + (Isensor.getCurrentDC()  * -1);
  if (Iout > Imax) {
      overCurrent_state = LOW;
      Serial.println("OVER CURRENT STATE DETECTED.");
    }

  // read temperature
  overTemp_state = HIGH;
  measureTemp();
  if (T > Tmax) {
      overTemp_state = LOW;
      Serial.println("OVER TEMPERATURE STATE DETECTED.");
    }

  // read cell voltages
  wakeup_idle();
  LTC6804_adcv(); // do cell AD conversion and fill cell registers
  delay(10);
  wakeup_idle();
  error = LTC6804_rdcv(0, TOTAL_IC, cell_codes); // read cell voltages from
registers
  if (error == -1)
  {
    Serial.println("PEC error detected in the received data");
  }

  // print to serial outputs:
  print_cells();
/*
  // test for over charge/undercharge states:
  minV1 = Vmin;
  maxV1 = Vmax;

  if (overCharge_state == LOW) { // add hysteresis
    maxV1 = maxV1 - .2;
  }

  if (underCharge_state == LOW) { // add hysteresis
    minV1 = minV1 + .2;
  }

  // get maximum and minimum cells:
  cellMax_i = -1;
  cellMin_i = -1;
  cellMin_V = 100.;
  cellMax_V = 0.;
  for (int i = 0; i < 10; i++)
```

```
  {
    float V = cell_codes[0][i] * 0.0001;
    if (V < cellMin_V) {
      cellMin_V = V;
      cellMin_i = i;
    }
    if (V > cellMax_V) {
      cellMax_V = V;
      cellMax_i = i;
    }
  }
*/
  underCharge_state = HIGH;
  overCharge_state = HIGH;
  overCurrent_state = HIGH;

  if(Vout > Vmax){
    overCharge_state = LOW;
    Serial.println("OVER VOLTAGE STATE DETECTED.");
  }

  if(Vout < Vmin){
    underCharge_state = LOW;
    Serial.println("UNDER VOLTAGE STATE DETECTED.");
  }

  if(Iout > Imax){
    overCurrent_state = LOW;
    Serial.println("OVER CURRENT STATE DETECTED.");
  }
/*
  if (cellMin_V <= minV1)
  {
    underCharge_state = LOW;
  }
  if (cellMax_V >= maxV1)
  {
    overCharge_state = LOW;
  }

  //  cell balancing:
  //  Turn on switch Sx for highest cell x if voltage is above threshold
  //  Note: DCP is set to 0 in initialize() This turns off discharge when
cell voltages are read.
  // set values in tx_cfg

  if (cellMax_V >= CELL_BALANCE_THRESHOLD_V)
  {
    balance_cfg(0, cellMax_i);
  } else {
    balance_cfg(0, -1);
  }

  // write tx_cfg to LTC6804. This sets the LTC6804 DCCx registers which
control the S pins for balancing:
  LTC6804_wrcfg( TOTAL_IC, tx_cfg);
  */
```

```
  // set cutoff state:
  cutoff_state = overCurrent_state &&  underCharge_state && overCharge_state
&&  overTemp_state;

  charge = (Vout - 25.0)/17.0;  //calculate state of charge using voltage
method

  //Send values to GUI and receive input
  displayVals();

  if(!cutoff_state){
    digitalWrite(dischargeCutoffPin, LOW);

    Serial1.print("t1.txt=\"OFF\"");
    scmd();
    Serial1.print("t1.bco=63488");    //set output indicator background to
red
    scmd();
  }

  //check for user input on touchscreen
  if(Serial1.available()){
    String data_from_display = "";
    data_from_display = (Serial1.readStringUntil('c'));
    storeVals(data_from_display);
  }

}

void init_cfg()
{
  for (int i = 0; i < TOTAL_IC; i++)
  {
    tx_cfg[i][0] = 0x04;
    tx_cfg[i][1] = 0x00;
    tx_cfg[i][2] = 0x00;
    tx_cfg[i][3] = 0x00;
    tx_cfg[i][4] = 0x00; // discharge switches  0->off  1-> on.  S0 = 0x01,
S1 = 0x02, S2 = 0x04, 0x08, 0x10, 0x20, 0x40, 0x80
    tx_cfg[i][5] = 0x20; // sets the software timer to 1 minute
  }
}

/*!**********************************
  \brief sets  the configuration array for cell balancing
  uses CFGR4 and lowest 4 bits of CGFR5
 ***********************************/
void balance_cfg(int ic, int cell)
{
  tx_cfg[ic][4] = 0x00; // clears S1-8
  tx_cfg[ic][5] = tx_cfg[ic][5]  & 0xF0; // clears S9-12 and sets software
timer to 1 min

  if (cell >= 0 and cell <= 7) {
    tx_cfg[ic][4] = tx_cfg[ic][4] | 1 << cell;
  }
```

```
  if ( cell > 7) {
    tx_cfg[ic][5] = tx_cfg[ic][5] | ( 1 << (cell - 8));
  }
}

/*!*********************************************************
  \brief Prints Cell Voltage Codes to the serial port
 *********************************************************/
void print_cells()
{
  unsigned long elasped = millis()  - tstart;

  serialPrint(elasped);   //ELAPSED TIME:
  Vout=0;
  //INDIVIDUAL CELL VOLTAGES:
  for (int current_ic = 0 ; current_ic < TOTAL_IC; current_ic++)
  {

    for (int i = 0; i < 10; i++)
    {
      Vout = Vout + cell_codes[current_ic][i] * 0.0001;
      serialPrint(cell_codes[current_ic][i] * 0.0001);
    }
  }

  serialPrint("\r\n");

}


/*!*****************************************************************************
**
  \brief print function overloads:
 *****************************************************************************
*/
void serialPrint(String val)
{
  Serial.print(val);
  Serial.print("\t");
}

void serialPrint(unsigned long val)
{
  Serial.print(val);
  Serial.print("\t");
}


void serialPrint(double val)
{
  Serial.print(val, 4);
  Serial.print("\t");
}

void serialPrint(int val)
{
  Serial.print(val);
```

```
  Serial.print("\t");
}

void print_config()
{
  int cfg_pec;

  Serial.println("Written Configuration: ");
  for (int current_ic = 0; current_ic < TOTAL_IC; current_ic++)
  {
    Serial.print(" IC ");
    Serial.print(current_ic + 1, DEC);
    Serial.print(": ");
    Serial.print("0x");
    serial_print_hex(tx_cfg[current_ic][0]);
    Serial.print(", 0x");
    serial_print_hex(tx_cfg[current_ic][1]);
    Serial.print(", 0x");
    serial_print_hex(tx_cfg[current_ic][2]);
    Serial.print(", 0x");
    serial_print_hex(tx_cfg[current_ic][3]);
    Serial.print(", 0x");
    serial_print_hex(tx_cfg[current_ic][4]);
    Serial.print(", 0x");
    serial_print_hex(tx_cfg[current_ic][5]);
    Serial.print(", Calculated PEC: 0x");
    cfg_pec = pec15_calc(6, &tx_cfg[current_ic][0]);
    serial_print_hex((uint8_t)(cfg_pec >> 8));
    Serial.print(", 0x");
    serial_print_hex((uint8_t)(cfg_pec));
    Serial.println();
  }
  Serial.println();
}

/*!*********************************************************************
  \brief Prints the Configuration data that was read back from the
  LTC6804 to the serial port.
 **********************************************************************/
void print_rxconfig()
{
  Serial.println("Received Configuration ");
  for (int current_ic = 0; current_ic < TOTAL_IC; current_ic++)
  {
    Serial.print(" IC ");
    Serial.print(current_ic + 1, DEC);
    Serial.print(": 0x");
    serial_print_hex(rx_cfg[current_ic][0]);
    Serial.print(", 0x");
    serial_print_hex(rx_cfg[current_ic][1]);
    Serial.print(", 0x");
    serial_print_hex(rx_cfg[current_ic][2]);
    Serial.print(", 0x");
    serial_print_hex(rx_cfg[current_ic][3]);
    Serial.print(", 0x");
    serial_print_hex(rx_cfg[current_ic][4]);
    Serial.print(", 0x");
```

```
    serial_print_hex(rx_cfg[current_ic][5]);
    Serial.print(", Received PEC: 0x");
    serial_print_hex(rx_cfg[current_ic][6]);
    Serial.print(", 0x");
    serial_print_hex(rx_cfg[current_ic][7]);
    Serial.println();
  }
  Serial.println();
}

void serial_print_hex(uint8_t data)
{
  if (data < 16)
  {
    Serial.print("0");
    Serial.print((byte)data, HEX);
  }
  else
    Serial.print((byte)data, HEX);
}

/*
   * Measure Temperature from MCP9700 with averaging for noise reduction
   * 10mV/ 1C sensitivity; 0.5V offset at 0 C
   * Vcc = 5V
   */
void measureTemp() {
  int avgs = 25; //number of temp measurements to average
  T = 0.0;

  for(int i = 0; i < avgs; i++)
  {
    T += ((analogRead(tempPin) * 5. / 1024.) - 0.5) / 0.01;
  }
  T = T / avgs;
}

void storeVals(String data_from_display){
  if(data_from_display.indexOf("VmaxIn") > -1){
    Vmax += 0.1;
  }
  if(data_from_display.indexOf("VmaxDe") > -1){
    Vmax -= 0.1;
  }
  if(data_from_display.indexOf("VminIn") > -1){
    Vmin += 0.1;
  }
  if(data_from_display.indexOf("VminDe") > -1){
    Vmin -= 0.1;
  }
  if(data_from_display.indexOf("ImaxIn") > -1){
    Imax += 0.1;
  }
  if(data_from_display.indexOf("ImaxDe") > -1){
    Imax -= 0.1;
  }
  if(data_from_display.indexOf("TmaxIn") > -1){
```

```
    Tmax += 1;
  }
  if(data_from_display.indexOf("TmaxDe") > -1){
    Tmax -= 1;
  }
  if(data_from_display.indexOf("Reset") > -1){
    softReset();
  }
}

//Send all measurement values to display
void displayVals(){
  char buff[6];//buffer for float to str)

  dtostrf(Vout, 3, 2, buff);
  Serial1.print("Vout.txt=\"");
  Serial1.print(buff);
  Serial1.print("\"");
  scmd();

  dtostrf(Iout, 3, 2, buff);
  Serial1.print("Iout.txt=\"");
  Serial1.print(buff);
  Serial1.print("\"");
  scmd();

  Serial1.print("T.val=" + String(T));
  scmd();

  Serial1.print("Cycles.val=" + String(cycles));
  scmd();

  Serial1.print("Charge.val=" + String(charge));
  scmd();

  dtostrf(Vmax, 3, 2, buff);
  Serial1.print("Vmax.txt=\"");
  Serial1.print(buff);
  Serial1.print("\"");
  scmd();

  dtostrf(Vmin, 3, 2, buff);
  Serial1.print("Vmin.txt=\"");
  Serial1.print(buff);
  Serial1.print("\"");
  scmd();

  dtostrf(Imax, 3, 2, buff);
  Serial1.print("Imax.txt=\"");
  Serial1.print(buff);
  Serial1.print("\"");
  scmd();

  Serial1.print("Tmax.val=" + String(Tmax));
  scmd();

  displayCells();
```

```
}

//Display voltage measurements for each cell
//converts float to string and changes value of text object on LCD
void displayCells(){
  char buff[6];

  dtostrf(cell_codes[0][0]/10000., 3, 2, buff);
  Serial1.print("C1.txt=\"");
  Serial1.print(buff);
  Serial1.print("\"");
  scmd();

  dtostrf(cell_codes[0][1]/10000., 3, 2, buff);
  Serial1.print("C2.txt=\"");
  Serial1.print(buff);
  Serial1.print("\"");
  scmd();

  dtostrf(cell_codes[0][2]/10000., 3, 2, buff);
  Serial1.print("C3.txt=\"");
  Serial1.print(buff);
  Serial1.print("\"");
  scmd();

  dtostrf(cell_codes[0][3]/10000., 3, 2, buff);
  Serial1.print("C4.txt=\"");
  Serial1.print(buff);
  Serial1.print("\"");
  scmd();

  dtostrf(cell_codes[0][4]/10000., 3, 2, buff);
  Serial1.print("C5.txt=\"");
  Serial1.print(buff);
  Serial1.print("\"");
  scmd();

  dtostrf(cell_codes[0][5]/10000., 3, 2, buff);
  Serial1.print("C6.txt=\"");
  Serial1.print(buff);
  Serial1.print("\"");
  scmd();

  dtostrf(cell_codes[0][6]/10000., 3, 2, buff);
  Serial1.print("C7.txt=\"");
  Serial1.print(buff);
  Serial1.print("\"");
  scmd();

  dtostrf(cell_codes[0][7]/10000., 3, 2, buff);
  Serial1.print("C8.txt=\"");
  Serial1.print(buff);
  Serial1.print("\"");
  scmd();

  dtostrf(cell_codes[0][8]/10000., 3, 2, buff);
  Serial1.print("C9.txt=\"");
```

```
  Serial1.print(buff);
  Serial1.print("\"");
  scmd();

  dtostrf(cell_codes[0][9]/10000., 3, 2, buff);
  Serial1.print("C10.txt=\"");
  Serial1.print(buff);
  Serial1.print("\"");
  scmd();
}

//send end bytes to complete command to Nextion display
void scmd(){
  Serial1.write(0xff);
  Serial1.write(0xff);
  Serial1.write(0xff);
}

void softReset(){
asm volatile ("  jmp 0");
}
```

Appendix C – Bill of Materials

Item	Name/Part Number	Description	Qty	Price/ea	Total
1	Elegoo Mega2560	Microcontroller compatible with BMS shield PCB and other peripherals	1	15.99	15.99
2	NX4827T043	Nextion 4.3" Touchscreen LCD used for graphical user interface	1	58.97	58.97
3	BMS PCB	PCB for LTC6804 and its connection to battery/peripherals	1*	25.9	25.9
3.1	LTC6804HG-2 BMS	Battery management IC; handles cell voltage measurements, balancing	1	29.7	29.7
3.2	CC0805KRX7R9BB103	10nF Capacitor for filtering circuits	13	0.077	1.001
3.3	GCM21BR72A104KA37K	0.1 uF protection capacitors for LTC6804	3	0.36	1.08
3.4	RC0805FR-07100RL	100 ohm resistor for filtering circuits	13	0.04	0.52
3.5	RC0805FR-074K99L	4.99k ohm pullup resistor	1	0.04	0.04
3.6	22284134	2.54MM Headers connect PCB to Arduino	10	1.06	10.6
4	Balance PCB	PCB for external cell balancing circuitry	1*	63.63	63.63
4.1	CRGH2512J16R	16 ohm resistor	10	0.454	4.54
4.2	ERA-6VEB1001V	1k ohm resistor	10	0.579	5.79
4.3	BSS308PEH6327XTSA1	Cell balancing MOSFET	10	0.256	2.56
5	SQM120P06	PMOS rated for 60V, 120A; used in cutoff circuit	1	2.7	2.7
5.1	FQP30N06	NMOS rated for 60V, 30A; used in cutoff circuit	1	1	1
5.2	4N25	Optocoupler used in cutoff circuit	1	0.387	0.387
5.4	BZX84C12L	12V zener diode	2	0.47	0.94
5.5	SFR2500001003JA500	100k ohm resistor	2	0.12	0.24
5.6	MBA02040C1502FRP00	15k ohm resistor	2	0.14	0.28
6	ACS712	Current sensor module	1	3.5	3.5
7	MCP9700-E/TO	Temperature sensor	2	0.25	0.5
8	Samsung 25R 18650	Lithium battery cells used in pack	40	2.75	110
8.1	Nickel Strip	0.2mm x 8mm pure nickel strip	1	20	20
8.2	XT60 Connectors	Battery/main power connectors	8	1.75	14
8.3	Heatshrink	Used for battery and cables	1	11.98	11.98
9	LTW-Pcbboard-8-12-Green-2	Perfboard for prototyping circuits	1	5.99	5.99
9.1	Cable (12 AWG)	Handles up to 20A from battery	5	1.05	5.25
9.2	Cable (22 AWG)	Handles cell balancing current	25	0.172	4.3
9.3	Solder	60/40	1	4.9	4.9
9.4	Solder wick	Braided copper wick for soldering	1	3.59	3.59
9.5	CQ8LF	Flux pen	1	7.95	7.95
9.6	4330582961	DuPont Wires	1	6.49	6.49
		Total Hardware Cost			424.318

*=must order more than 1 for price