

# Automated Lithium Ion Battery Characterizer

DESIGN DOCUMENT

sddec20-25

PrISUm

Dr. Nathan Neihart

Joseph DeFrancisco - Team Leader

Bryan Kalkhoff - Hardware Design

Ryan Willman - Hardware Design

Connor Luedtke - Hardware Design

Kyle Czubak - Software Development

Ben Kenkel - Software Development

Team Email and Website:

[sddec20-25@iastate.edu](mailto:sddec20-25@iastate.edu)

[sddec20-25.sd.ece.iastate.edu](http://sddec20-25.sd.ece.iastate.edu)

November 18, 2020

# Executive Summary

## Development Standards & Practices Used

### Development Standards & Practices Used

- Agile Development
- IEEE 1679.1-2017: IEEE Guide for the Characterization and Evaluation of Lithium-Based Batteries in Stationary Applications

## Summary of Requirements

- Characterize 8 batteries at once
- Continuous monitoring for safe operation
- Perform full-cycle characterization of Lithium-Ion Batteries
  - Measure current in and out of individual batteries
  - Voltage monitoring for each battery
  - Calculate battery capacity and internal resistance from above measurements
  - Temperature measurement
- Storage of data for future analysis
- Serialize batteries and the associated data

## Applicable Courses from Iowa State University Curriculum

- EE 230: Electronic Circuits and Systems
- EE 333: Electronic Systems Design
- CPR E 288: Embedded Systems I: Introduction
- CPR E 488: Embedded Systems Designs
- COMS 363: Introduction to Database Management Systems

## New Skills/Knowledge acquired that was not taught in courses

- PCB Design
- Lithium Battery Characteristics
- Git & GitLab
  - Intense use for SE and CPR E
  - Basic Git operations for EE
- Advanced Embedded System Development

## Table of Contents

1 Introduction	29
1.1 Acknowledgment	29
1.2 Problem and Project Statement	29
1.3 Operational Environment	29
1.4 Requirements	29
1.4.1 Engineering Constraints and Non-Functional Requirements	30
1.5 Intended Users and Uses	30
1.6 Assumptions and Limitations	30
1.7 Expected End Product and Deliverables	31
2. Specifications and Analysis	32
2.2 Proposed Approach	32
2.3 Design Analysis	32
2.4 Development Process	32
2.5 Conceptual Sketch	32
2.5.1 Enclosure	34
3. Statement of Work	36
3.1 Previous Work And Literature	36
3.2 Technology Considerations	36
3.3 Task Decomposition	36
3.4 Possible Risks And Risk Management	37
3.5 Project Proposed Milestones and Evaluation Criteria	37
3.6 Project Tracking Procedures	38
3.7 Expected Results and Validation	38
4. Project Timeline, Estimated Resources, and Challenges	39
4.1 Project Timeline	39
4.2 Feasibility Assessment	40
4.3 Personnel Effort Requirements	41
4.4 Other Resource Requirements	41
4.5 Financial Requirements	42
5. Testing and Implementation	43
5.1 Interface Specifications	43
5.2 Hardware and software	43

5.3	Functional Testing	43
5.3.1	Hardware System Overview	43
5.3.2	Hardware: Board Power and PMIC Implementation	22
5.3.3	Hardware: Battery Charging Implementation	23
5.3.4	Hardware: Constant Current Electronic Load Implementation	24
5.3.5	Hardware: Voltage and Current Sensing Implementation	27
5.3.6	Hardware: Microcontroller Implementation	29
5.3.7	Software: Raspberry Pi Implementation	29
5.3.8	Software: Embedded Software Implementation	31
5.4	Non-Functional Testing	33
5.5	Process	34
5.5.1	Evaluation Board Assembly (Load and micro)	34
5.5.2	Load Prototype Testing Process	34
5.5.3	Voltage and Current Measurement Testing	34
5.5.4	Main Board Power Rail Testing	35
5.5.5	Main Board Load Testing	35
5.5.6	Main board Battery Charger Testing	35
5.5.7	Microcontroller Testing	35
5.5.8	Raspberry Pi Testing	36
5.5.9	Embedded Software Testing	36
5.6	Results	36
5.6.1	Microcontroller Test Results	36
5.6.2	Load Prototype Test Results	39
5.6.3	Main board Power Rail Testing Results	41
5.6.4	Main Board Battery Charger Testing Result	41
5.6.5	Main Board Load Testing Results	41
5.6.6	Embedded Software Testing Results	42
5.6.7	Web App Testing Results	42
6.	Closing Material	44
6.1	Conclusion	44
6.2	References	44
6.3	ACRONYMS	44
	Appendix A: Main Project Schematic and PCB	45

Appendix B: Constant Load Circuit Eval Schematic and PCB	54
Appendix C: Microcontroller Eval Schematic and PCB	57
Appendix D: Bill of Materials for Main Board	60
Appendix E: Bill of Materials for Constant Load Circuit Eval Board	63
Appendix F: Bill of Materials for Microcontroller Test Eval Board	64
Appendix G: User Manual	65
Appendix G.A: Connecting to the website	68
Appendix G.B: View Test Data	68
Appendix G.C: Create New Battery Pack	71

## List of figures

Figure 1: High-Level Block Diagram .....	33
Figure 2: Battery Test Diagram .....	34
Figure 3: Model of Enclosure .....	35
Figure 4: Flowchart of the project .....	36
Figure 5: First Semester Timeline.....	39
Figure 6: Estimated Second Semester Timeline .....	40
Figure 7: Power Supply Input, Filtering, and Protection .....	22
Figure 8: 5V Switching Regulator Module.....	23
Figure 9: Programmable Constant Current Load.....	24
Figure 10: Transient Response of Load Circuit.....	25
Figure 11: Phase Margin Test Bench.....	26
Figure 12: Programmable Constant Current Load Bode Plot.....	26
Figure 13: 8-Channel Analog Switch for Reference Voltage.....	27
Figure 14: Power Supply and Interfacing for INA3221 .....	28
Figure 15: Sense Resistor and ADC Connections for INA3221.....	28
Figure 16: Filtering Circuit for the SAMC21J.....	29
Figure 17: User Interface page the raspberry Pi boots to.....	30
Figure 18: User Interface Defining What to Label Each Cell .....	30
Figure 19: User Interface for seeing all packs previously created.....	31
Figure 20: User interface for creating a new battery pack.....	31
Figure 21: Embedded Software Testing Procedure .....	33
Figure 22: Picture of the I2C Bus .....	37
Figure 23: SDA and SCL Mirroring Each other.....	38
Figure 24: Working CAN Bus .....	39
Figure 25: Measured Load Current against ideal with 2.8V source .....	40
Figure 26: Mod Wire to Reset Pin .....	42
Figure 27: Schematic of Main Page.....	45
Figure 28: Load Circuit Schematic .....	46
Figure 29: Charge Circuit Schematic.....	47
Figure 30: Microcontroller Schematic .....	48
Figure 31: VI Sense Measurement IC Schematic .....	49
Figure 32: IC VI Measurement Schematic .....	50
Figure 33: Power Circuit Schematic .....	51
Figure 34: Relay Circuit Schematic .....	52
Figure 35: Main Board PCB Layout.....	53
Figure 36: Constant Load Circuit Eval Board Schematic.....	54
Figure 37: Constant Current Load Eval PCB layout .....	55
Figure 38: Constant Current Load Eval Board Physical PCB .....	56
Figure 39: Microcontroller Eval Board Schematic .....	57
Figure 40: Microcontroller Eval Board PCB Layout.....	58
Figure 41: Microcontroller Physical PCB.....	59
Figure 42: User Interface the Pi Boot onto .....	65
Figure 43: User Interface for selecting the active battery pack .....	65
Figure 44: User Interface defining what to label each cell .....	66

Figure 45: User interface for test in progress page .....	67
Figure 46: User Interface for failed test .....	67
Figure 47:User interface for successful test.....	68
Figure 48:User interface for desktop home screen .....	68
Figure 49:User interface for seeing all battery packs that have been created.....	69
Figure 50:User interface for seeing information about a battery pack .....	70
Figure 51:User interface for seeing battery test data (Using fake data) .....	71
Figure 52:User interface for creating a new battery pack.....	72

# 1 Introduction

## 1.1 ACKNOWLEDGMENT

Dr. Nathan Neihart, an Associate Professor in Electrical Engineering at Iowa State, will be the faculty advisor for this project and will be providing the technical support needed for the project. Additionally, PrISUM Solar Car will be providing lithium-ion batteries for us to use while testing.

## 1.2 PROBLEM AND PROJECT STATEMENT

An important step in lithium-ion battery pack manufacturing is to have an accurate characterization of the charging and discharging curves for every battery cell before it goes into a parallel module. This allows for the grouping of similar performing batteries into each module, which ensures that the batteries in a module will charge and discharge at similar rates, improving the efficiency of the pack. PrISUM Solar Car currently does not have a reliable method for performing this characterization and is proposing that we develop a system that can address this need.

Our proposed solution to this problem would be making a device that would charge and discharge the batteries and graph the current and voltage characteristics. To help group battery cells, the device would also be able to remember which cells were characterized and store that information somewhere accessible.

## 1.3 OPERATIONAL ENVIRONMENT

The operational environment for the final product will be used in a typical indoor environment of ambient temperatures between 10 and 30 °C. Due to the mechanical enclosure, and high-power dissipation at times during the characterization process the ambient temperature may rise significantly. The temperature will always be monitored and efforts to provide adequate cooling will be investigated.

This system will need two wall plug openings, one for the raspberry pi and another for the characterizer. The raspberry pi and the characterizer module need to be in close distance in order to plug them into each other.

## 1.4 REQUIREMENTS

### **Main Controller:**

The main controller will be responsible for controlling the testing process. This includes starting and stopping tests, assigning battery numbers, and collecting and storing the test data. We are using a Raspberry PI as our main controller.

### **Module Unit:**

This is where the testing will take place. We will be capable of characterizing 8 batteries at the same time on each module. Each battery will have temperature, voltage, and current monitored and reported back to the main controller throughout the test. Each battery will have a programmable current load circuit and charging circuit directed by the module's microcontroller.

### **Minimum Viable Requirements:**

- Perform full-cycle characterization
  - Measure current in and out
  - Voltage measurement
  - Temperature measurement
- Serial number tracking every battery with associated data
- Storage of data for future analysis
- Continuous monitoring for safe operation
- Characterize 8 batteries at once

### **Non-functional Requirements**

- User web interface must have minimal downtime
- Characterization results must be immediately uploaded to the web without delay



### **Economic Requirements**

- Prototype must cost no more than \$500, where final units must cost no more than \$250
- Able to be reproduced by others at a reasonable cost
- Parts must be widely available on the open market

### **Environmental Requirements**

- Room temperatures between 10 C and 30 C
- Humidity between 20% and 80%
- Lithium fire suppression
  - Class ABC Fire Extinguisher
  - Sandbags

### **Stretch Goals:**

- Battery Module Optimization Software that automatically groups batteries into modules based on gathered data
- Build a full-scale characterizer capable of ~40 batteries at once.
  - Using multiple modules.
- Web-based interface for viewing battery characteristic data.

## **1.4.1 Engineering Constraints and Non-Functional Requirements**

### **Non-functional Requirements**

- User web interface must have minimal downtime
- All characterization results must be uploaded before finishing
- Store little amount of data as possible
- Operate unsupervised

### **Constraints**

- Controlling the temperature of a battery
- Following proper charger profiles for lithium ion batteries
- Total board cost must not exceed \$500 (see section 1.4 Requirements)
- The design must be modular, the pi should be able to operate multiple test modules via CAN communication
- Each module must be able to characterize up to 8 batteries at once

## **1.5 INTENDED USERS AND USES**

The project is intended to provide the ability to characterize batteries to optimize Lithium-Ion battery packs' efficiency and longevity.

There are two intended users' groups for this project:

1. PrISUM: This project was proposed by PrISUM to fulfill a need in their solar car battery pack design.
2. Non-Professionals: Most people that are not in industry skip the characterization part of designing a battery pack due to no viable market solution. As larger-scale battery pack design is becoming more feasible due to declining lithium battery costs, it is becoming increasingly feasible to make large battery packs, which results in an increasing need for proper battery characterizing methods.

## **1.6 ASSUMPTIONS AND LIMITATIONS**

### **Assumptions**

- End users will understand basic lithium battery safety standards.
- End users will have access to computers and basic computer skills.

- Our proposed solution will be used in a climate-controlled room.

#### Our Limitations

- The cost of the final product will not exceed \$500
- The system will require an AC power source.
- The end product cannot be used if the ambient temperature is outside the battery's usable temperature range as specified by the lithium-ion battery datasheet.

### 1.7 EXPECTED END PRODUCT AND DELIVERABLES

The end deliverable will be a complete battery characterization system capable of running current, voltage, and temperature tests on up to 8 batteries at a time. The finished system will process data and upload it to a database. Once the lithium cells are inserted into the end product and the device is started, no more user input will be needed until the end of the testing cycle. To achieve this goal, multiple subsystems must be delivered, including the main node, support for multiple testing nodes, and a database for storing the test data. To meet the required deliverables by our project deadline of November 18th it is crucial to hit the second semester milestones listed in section 4.1 including finishing the PCB design, embedded code, and assembly. This all must be completed by our deadline of November 1st, to leave ample time for test and debugging.

## 2. Specifications and Analysis

### 2.2 PROPOSED APPROACH

The first semester we completed the battery test program and a significant portion of the schematic design work. The test program outlines the process for completing the capacity and internal resistance measurements and sets requirements for the hardware so that these calculations can be made accurately. For hardware design work we have completed the voltage and current measurement, charging, constant current load, temperature measurement and the microcontroller hardware. It will be challenging to verify the functionality of most of these circuits without hardware in hand. The main exception is the load circuit which has been verified in simulation.

Moving forward it would be nice to prototype pieces of the design on perfboard or breadboard. This is dependent on having lab access over the summer or next semester. This would allow us to verify design work before moving to the layout stage.

For the software app we have completed the app prototype and database schema. One issue we faced was communicating with the characterizers through CAN with a web app. This will be challenging to do with having 2 systems working together. All URL paths are created and moving forward the HTML and CSS will be created so testing can start.

### 2.3 DESIGN ANALYSIS

This semester we completed the battery test program and a significant portion of the schematic design work. The test program outlines the process for completing the capacity and internal resistance measurements and sets requirements for the hardware so that these calculations can be made accurately. For hardware design work we have completed the voltage and current measurement, charging, constant current load, temperature measurement, and the microcontroller hardware. It will be challenging to verify the functionality of most of these circuits without hardware in hand. The main exception is the load circuit which has been verified in simulation.

Moving forward it would be nice to prototype pieces of the design on perf board or breadboard. This is dependent on having lab access over the summer or next semester. This would allow us to verify design work before moving to the layout stage.

For the software app, we have completed the app prototype and database schema. One issue we faced was communicating with the characterizers through CAN with a web app. This will be challenging to do with having 2 systems working together. All URL paths are created and moving forward the HTML and CSS will be created so testing can start.

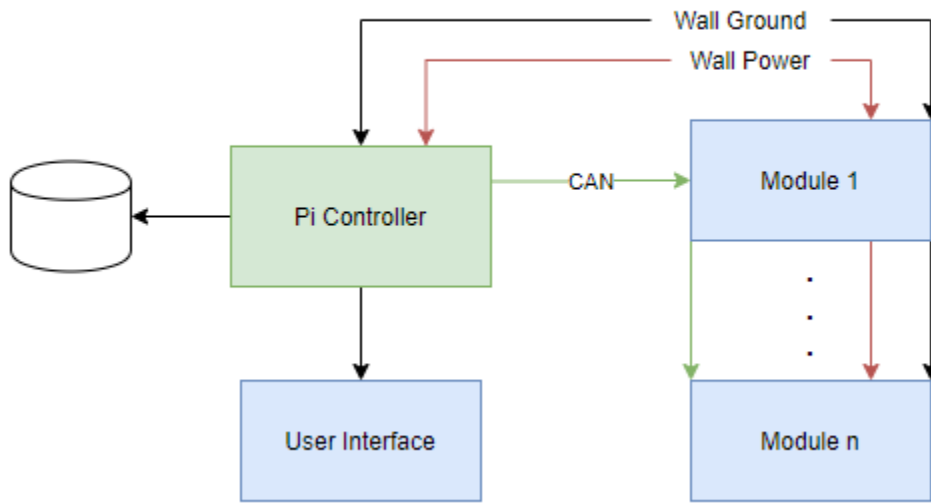
### 2.4 DEVELOPMENT PROCESS

With this project having so many different parts - hardware design, embedded system design, backend development, etc. - choosing a development process will be difficult. While agile will work well creating a website and creating a database, the waterfall process will work better with hardware design and embedded systems software. Since the hardware design and embedded systems software is the more important part of this project, we will be using the waterfall process for the vast majority of this project.

### 2.5 CONCEPTUAL SKETCH

The overall design of the project is shown in Figure 1. Figure 2 shows the subsystem view of the main controller. Figure 2 illustrates the layout of the battery measurements and communication. The team will follow these block diagrams when designing the product.

### Overall Block Diagram



### Overall Block Diagram

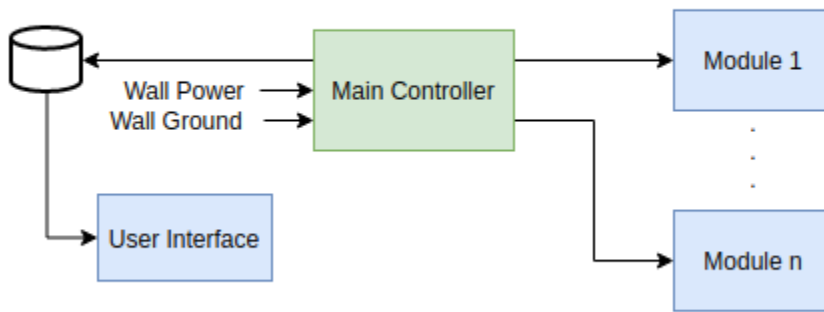


Figure 1: High-Level Block Diagram.

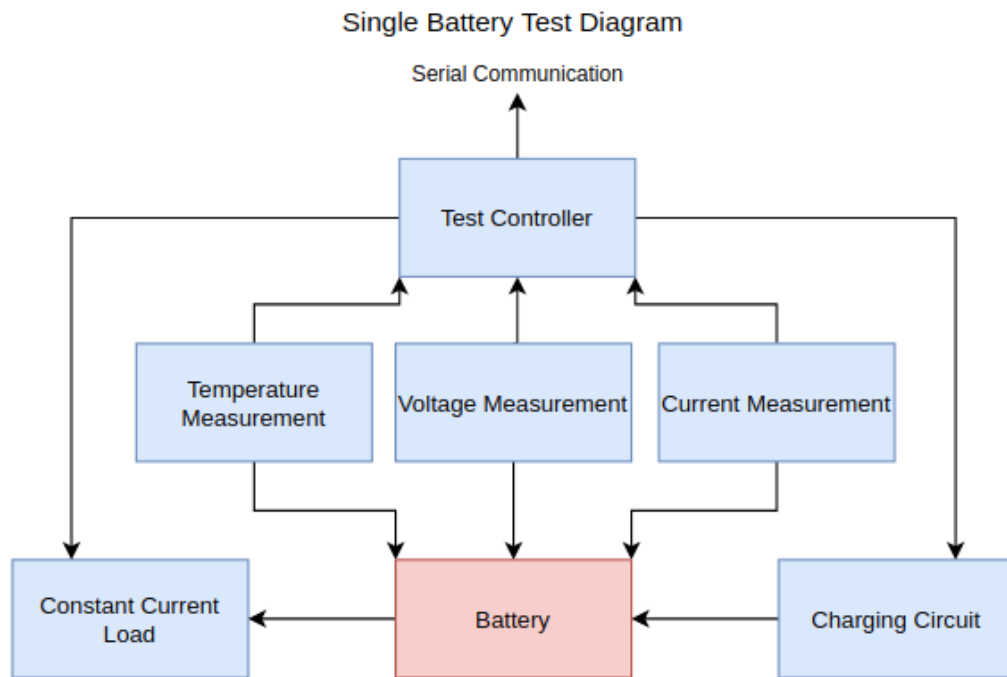
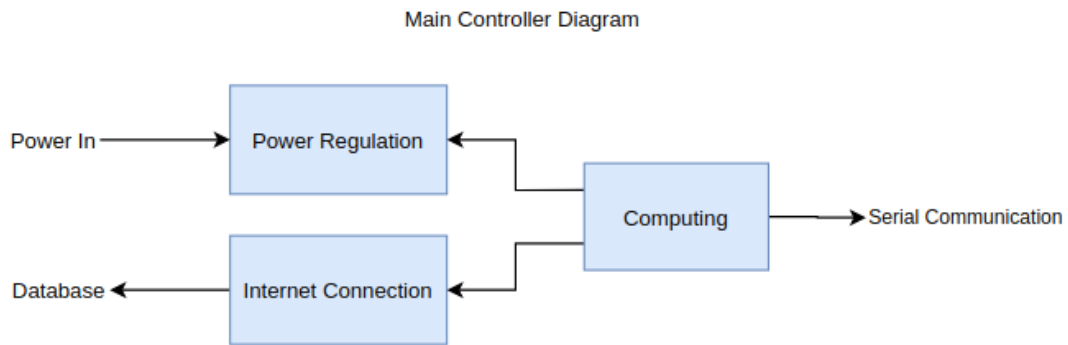


Figure 2: Battery Test Diagram

### 2.5.1 Enclosure

The 3D printed enclosure in Figure 3 was designed to incorporate a PCB and cooling fan. The fan is positioned to pull air up into the case to cool the load components. This is where the PCB rests on the built-in standoffs giving easy access to the 18650s. This design was used to create a compact design to fit all required project components. It also allows for easy serviceability for the PCB.

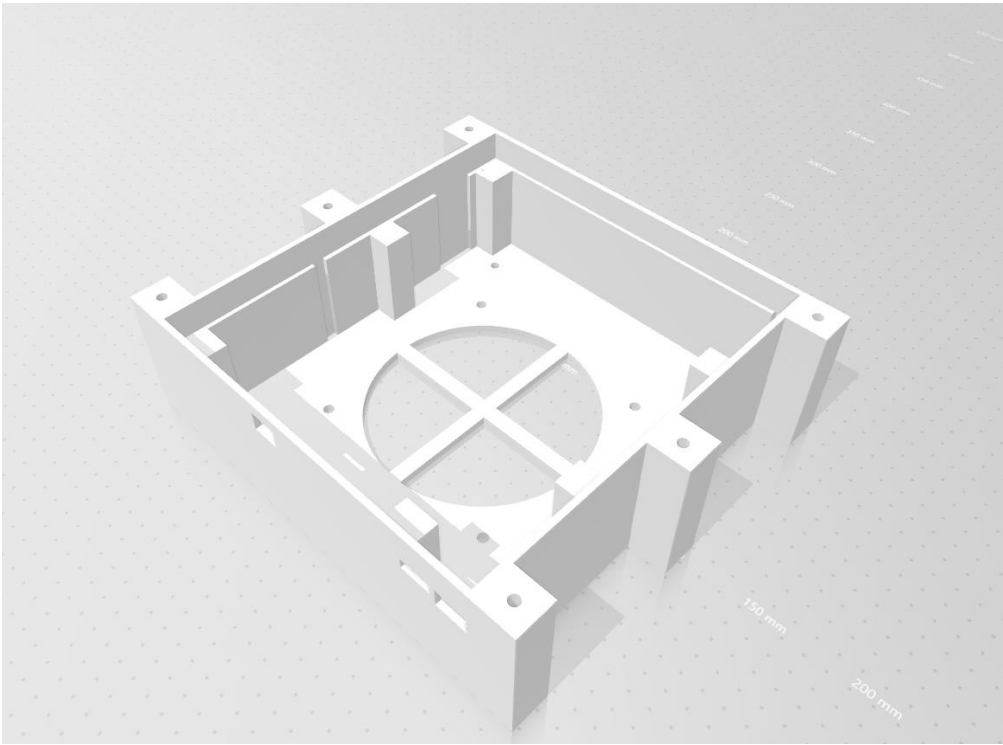


Figure 3: Model of Enclosure

## 3. Statement of Work

### 3.1 PREVIOUS WORK AND LITERATURE

There are already existing small scale battery testing equipment that can be purchased commercially, such as the Opus BT-C3100, but they are normally very limited in functionality [1]. Our team is attempting to design and implement a system that can test a larger quantity of batteries at one time.

Many members of our team have extensive experience designing circuits and creating PCBs. Also, our software developers have experience creating embedded software systems from scratch. Regarding Lithium-Ion batteries and their safety, we have a team member who is familiar with the proper handling and storage of Lithium-Ion batteries.

### 3.2 TECHNOLOGY CONSIDERATIONS

#### Hardware

- The hardware must interface with the digital communication protocols we are using
- The hardware must respond to unsafe operating conditions for the batteries
- Analog measurements reported over digital bus protocol decrease hardware complexity but increase software complexity.
- PCB will be designed using a 4-layer stack up to take advantage of large thermal mass to dissipate large amounts of heat.

#### Software Section

- The software must be deployable onto the chosen microcontroller.
- The software must be able to run continuously for the entire duration of the test.

### 3.3 TASK DECOMPOSITION

The project is broken into several subsystems. On the hardware side, there will be the power circuit, charging system, the discharge circuit, the microcontroller, and measurement circuitry. For software, there is a Flask app on the raspberry pi for interfacing with the user, a database for storing information, and the testing management software. The power circuit will provide a 5-volt rail for the digital circuitry and a 12-volt rail for all other systems. The charge/discharge systems will provide constant current charge and discharge of each cell. Measurement circuitry will relay voltage, current, and temperature data back to the main controller for each cell. Table 1 shows the in-detail breakdown of Figure 4.

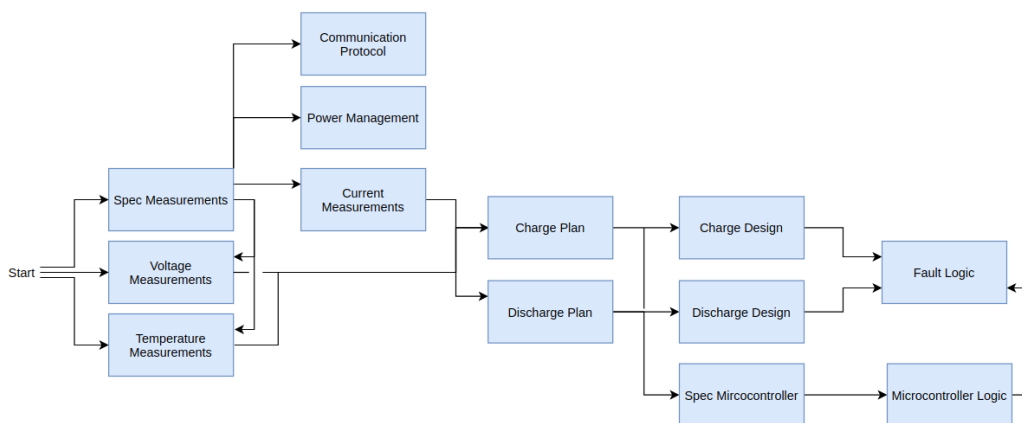


Figure 4: Flowchart of the project

Table 1: Task breakdown

Task	Description
Spec Measurements	Based on industry papers [1], determine what accuracy and how often do we need to measure the batteries.
Voltage Measurements	Based on industry papers [1], determine what accuracy and how often do we need to measure the batteries' voltages.
Temperature Measurements	Based on industry papers [1], determine what accuracy and how often do we need to measure the batteries' temperature.
Communication Protocol	Determine which communication protocol should be used to communicate between the IC and the Pi. This will include determining if the protocol can transmit at an acceptable rate.
Power Management	Determine what would be the best way to power the system. This includes battery measurements and Pi.
Current Management	Based on industry papers [1][2], determine the best way to manage the ~3A that will be required to discharge and charge the batteries.
Charge Plan	Based on industry papers [1][2], determine a plan for safely charging each of the batteries.
Discharge Plan	Based on industry papers [1][2], determine a plan for safely discharging each of the batteries.
Charge Design	Based on the finding of the "charge plan" design a circuit that will meet the requirements.
Discharge Design	Based on the finding of the "discharge plan" design a circuit that will meet the requirements.
Spec microcontroller	Determine which microcontroller will work for the system. The microcontroller shall be able to communicate with all IC and shall be able to communicate with the pi.
Microcontroller logic	Based on the findings from the "spec microcontroller" design the circuitry necessary for the microcontroller to work properly and meet all other requirements.
Fault logic	Design the logic to determine if the batteries are outside safe points.

### 3.4 POSSIBLE RISKS AND RISK MANAGEMENT

Lithium-Ion batteries are volatile and require continuous monitoring for safe operating conditions. If Lithium-Ion batteries are not handled within the specifications, batteries can react with unplanned thermal events.

To manage risk, both hardware and software solutions will be implemented. The chosen charging circuit follows JEITA guidelines [2] where thermal management components do not let the lithium cell exceed 60 Celsius. The circuit also has short circuit protection if the batteries are improperly inserted and contain fuses if too much current is drawn. In addition to our hardware protections, we will have our software monitoring all variables as well so that the testing can be halted if parameters such as temperature or current go outside of expected ranges. However, if an issue does arise, we will be following Iowa State University's Environmental Health & Safety's guidelines for handling problematic Lithium-Ion batteries, which includes isolating cells in the sand, having a fire extinguisher nearby, and we are aware of how to properly dispose of damaged batteries.

### 3.5 PROJECT PROPOSED MILESTONES AND EVALUATION CRITERIA

Overall, key milestones would be the completion of the different subsystem designs, communication, web app, database, charging, load circuit, sensing, and fault logic.

For the first semester, we have proposed the completion of all necessary schematics with simulations. This will be composed of finding components, designing the circuits, and testing using PSpice. Following this timeline is ideal to provide adequate time in the second semester to design the PCB and troubleshoot if needed. The backend and database for the web app are complete. Next semester will be working on the HTML and CSS for the web app and then do testing for it.

For the second semester we will be focused on completing the PCB schematic design alongside the embedded system programming. These two tasks must be completed in a timely manner so that we can



build and move forward with testing. Throughout the semester we will continue progressing on the web application. This semester is condensed so sticking to our timeline will be essential.

### 3.6 PROJECT TRACKING PROCEDURES

This project will be tracked mostly through the provided Git repository. All project files will be maintained through git, with the master branch being reserved for functional prototypes and the completed project. The team will be using GitLab Issues for tracking tasks that need completed, as well as documenting their progress.

In addition to git, the team has weekly meetings with Dr. Neihart to discuss the team's progress and to highlight what tasks should be worked on over the next week. Additional meetings will be held within the team for working on different tasks, planning future work, and for working on system integration and reporting.

### 3.7 EXPECTED RESULTS AND VALIDATION

The high-level end product of this project will be a battery characterizing system that is capable of automatically characterizing at least 8 battery cells at a time. The system will output data to a database that can be used to create graphs showing the charging and discharging rates of each tested battery. Specifically, we will be recording temperature, voltage, and current for each battery over time during the testing.

A breakdown of expected results is described later in section 5.3.

## 4. Project Timeline, Estimated Resources, and Challenges

### 4.1 PROJECT TIMELINE

The project timeline for the first semester is implemented by using a Gantt chart in Figure 5. Day 0 starts on 1/26/2020 while day 77 is on 4/26/2020. We will start by setting and understanding our system requirements for roughly 2 weeks. Then transition into researching for needed materials in understanding how to design our project. Once the research is complete, components will be chosen based on the parameters found in our research. Finally, the circuit will be implemented through Altium and then simulated through Spice for validation.

### Project Timeline - First Semester

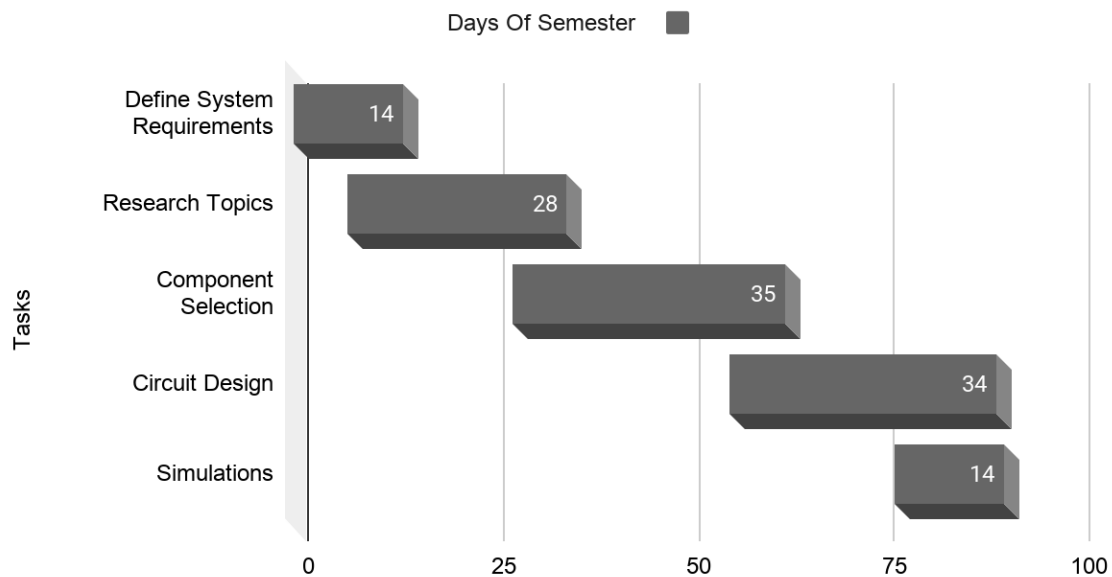


Figure 5: First Semester Timeline

Below are specific deliverables that should be completed by specific dates.

- Hardware Team
  - 2/9 - Define System Requirements
  - 3/1 - Research
    - Spec current, voltage, and temperature measurements
  - 3/29 - Component Selection
    - 3/14 - Main Components
    - 3/29 - Sub-Components
  - 4/25 - Circuit Design
    - 4/10 - Main Schematic
    - 4/12 - Micro Circuit
    - 4/15 - Load, Charge Circuit
    - 4/25 - Other Circuits
  - 4/26 - Simulations
    - 4/25 - Load Simulation
- Software Team
  - 4/26 - Web App and Database

## Current Project Timeline - Second Semester

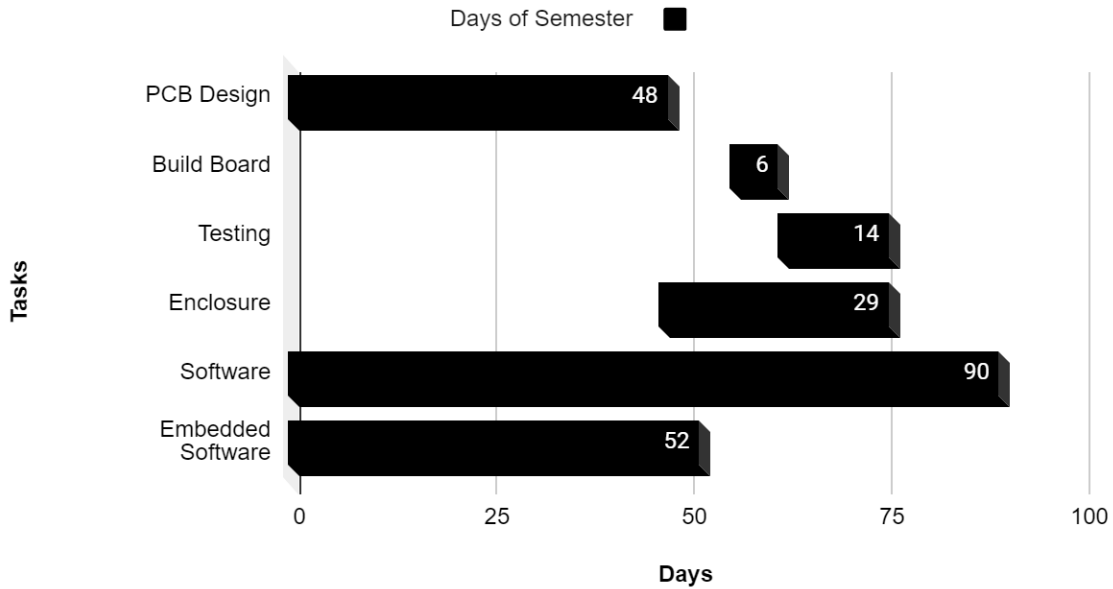


Figure 6: Estimated Second Semester Timeline

Figure 6 shows the timeline for semester 2. We wanted to have the PCB design finished by the start of October and have a week of review to find any small mistakes. Then 12 days later we would order the board. We ended up getting the board ordered on October 8th and the board arrived and was soldered by the 19th. This was on schedule like we imagined, and we started testing on October 20th. The enclosure did get a final design by the end of the semester but could not order one. The embedded software was written enough to test when the board arrived but was not fully finished before then. The embedded code did finish after testing. The web app was being worked on all semester and was finished to a point where it can work by itself by the end of the semester

Below are specific deliverables for the second semester that should be completed by specific dates.

- Hardware Team
  - 8/31 - Update schematics that are missing components
  - 10/1 - Rev 1 PCB
  - 10/12 - Order PCB
    - 10/1 - Design Review for Rev 1
    - 10/11 - Finalize changes
  - 10/21 - Solder PCB and Begin testing
  - 10/28 - Finish Enclosure design
  - 11/12 - Get Board to work on 1 module
- Software Team
  - 4/26 - Embedded Software
  - 11/15 - All software required for a successful test

### 4.2 FEASIBILITY ASSESSMENT

Despite the added challenge of having to work from home, we have concluded that the project won't need to change. We came to this conclusion because:

- We were beginning the schematic layout which can be done remotely
- Code can be done remotely, and we have access to hardware to run it on
- Conferences/meetings can be done via Hangout

During the second semester several group members were able to work on campus and access to testing equipment. In addition, we were able to stay very close to our planned schedule for the first semester.

### 4.3 PERSONNEL EFFORT REQUIREMENTS

For personal effort requirements, the overall task is listed in Table 2

“Research Tops” shall be split into subgroup:

- Microcontroller
- Charge/discharge
- Power system

This task delegation will be used for “Battery Component Selection”, “System Integration”, and “System Testing”.

*Table 2: Personal Effort Requirements*

Task	Est. Time	Description
Define System Requirements	2 Weeks	Functional requirements needed to begin research.
Research Topics	4 Weeks	Individuals will be assigned topics to explore. Such topics are Microcontrollers, charge/discharge circuits, and power systems.
Battery Component Selection	4 Weeks	Find components on DigiKey that meet project needs.
Schematic Design	5 Weeks	Use Altium to create the desired circuit designs for board functionality.
Simulations	2 Weeks	Use PSPICE to simulate the load circuit.
PCB Design	7 Weeks	Fix all Altium schematic errors before designing the PCB. Host design review to fix issues before ordering PCB.
Soldering PCB	1 Week	Solder all necessary components for the board to function so software can begin preliminary tests.
Testing	3 Weeks	Test PCB for desired waveforms and voltages.
Enclosure	4 Weeks	Design a 3D printed case to house the PCB and other required components. Ensure that it is compact and easily printable.
System Integration	2 Weeks	Integrate software and hardware for final testing.
System Testing	2 Weeks	Collect data and run tests to see if the device is functioning properly.

### 4.4 OTHER RESOURCE REQUIREMENTS

Basic lab equipment found in Coover. This will include:

- A power supply capable of supplying 12V, 8A
  - This will be needed to power the system
  - An alternate would require a change in the testing procedure to allow for lower amperage
- Multimeter capable of measuring with an accuracy of  $\pm.001$ 
  - This will be needed to confirm that the system is measuring the voltage of the batteries correctly
  - This is a requirement and therefore is no alternate
- Oscilloscope capable of measuring with an accuracy of  $\pm.001$ 
  - This will be needed to confirm the I2C line is working as expected
  - As I2C is an important component of the system, there is no alternative if the oscilloscope can't be found

#### 4.5 FINANCIAL REQUIREMENTS

The project has a budget of \$500 with the possibility of PrISUm being able to obtain items if needed. The primary goal is to stay within the budget to create the final project. Table shows the total project cost. We have successfully stayed within budget.

*Table 3: Total Project Cost*

<b>Description</b>	<b>Cost</b>
PCB	\$50
PCB Components	\$273
Other	\$50

## 5. Testing and Implementation

### 5.1 INTERFACE SPECIFICATIONS

For testing the hardware, the team has implemented several test points on the circuit board. Some notable test points are on:

- SCL and SDA of the I2C line
  - This will enable the ability to be able to confirm that the I2C bus is operating as expected
- Power in
  - This will enable the ability to be able to confirm that the system is powered properly
- 3.5V
  - This will enable the ability to be able to confirm that the 3.5V power rail is working properly
- 5V
  - This will enable the ability to be able to confirm that the 5V power rail is working properly
- Ground
  - This will aid in all test points for our reference
- Enable lines for the battery charger IC
  - This will enable the ability to be able to confirm that these IC are being enabled properly

The software will be using Atmel Studio to program the microcontroller. Using the editor allows the team to be able to test be:

- Using the debug tool in Atmel Studio
  - This will enable the ability to follow the code and step through what it should be doing at any given time.

### 5.2 HARDWARE AND SOFTWARE

Testing each sub-system and the final product will require the use of a few hardware and software tools.

**Multimeter:** The multimeter will allow us to measure both the voltage and current of the batteries to confirm the functionality of our measurement systems. Additionally, the multimeter will be greatly beneficial for debugging issues in the circuit designs.

**DC Power Supply:** Will be required to provide known and stable voltages to various parts of the circuit.

**Atmel Studios:** The built-in debugger for Atmel Studios will be beneficial for debugging any software related issues.

**Thermal Imaging Camera:** A thermal imaging camera will be used to monitor the temperature of the batteries throughout the testing process while we verify our temperature monitoring circuit.

**Chrome:** A web browser to do testing with our web app.

**Postman:** A program to test POST and GET messages to our website backend.

**Sand:** The team will have buckets of sand available to isolate batteries that are beginning to show symptoms of improper treatment to minimize the likelihood of a large Lithium-Ion fire.

### 5.3 FUNCTIONAL TESTING

At the end of 491, we did not have any physical hardware yet, so we have only completed simulations of our designs where possible. Once we get hardware in and access to lab equipment again we will be purchasing small subsets of the project components to prototype some designs.

#### 5.3.1 Hardware System Overview

Test flow from a hardware perspective is best seen on the first page of the schematic, which can be found in

Appendix A: Main Project Schematic and PCB Starting on the left hand side of the page, the battery connectors and fuses connect to the first hierarchical sheet, VI\_SENSE\_MSR. Located on this page is the precision and voltage current measurement. All of the current into or out of the batteries will pass through this page. Next, the relay page. The relays in the page are controlled by the microcontroller and choose between charger or load connections. In the default state, when the relay coil is de-energized, the charger is connected to the battery and in a high impedance state. When the test starts, each battery voltage will be noted and each battery will be charged to 4.2V through the relay. During this period, the voltage and current measurements are only being monitored for fault conditions, the data is not being logged yet. Once the batteries reach 4.2V, the microcontroller will switch the relay positions to the load circuit and provide a reference voltage to set the discharge current. During this part of the characterization the voltage and current data is being logged back to the database. As each battery reaches its end of discharge voltage, the reference voltage for that battery will be pulled low stopping the discharge and the relay will connect it back to the charger. The battery will be charged again, with voltage and current being recorded, to 4.2V. At this point, the internal resistance measurement will be made. This is done by pulling one very large current and one much smaller current from the battery in quick succession. The voltage of the battery is measured before and after each load pulse, and using these points the internal resistance of the battery is calculated in software. If the battery voltage has dropped from 4.2V, it will be recharged. And then the second capacity measurement is made following the same steps and previously mentioned. Once all of the measurements are complete, the batteries will be recharged back to roughly 80% capacity and the testing is completed.

### 5.3.2 Hardware: Board Power and PMIC Implementation

We will be using multiple power rails for this design. The primary 12V rail will be provided by a large laptop style brick charger, the specific part we have specced is the VES150. It is an isolated switching module that plugs into a 120VAC wall outlet and provides a 12VDC rail isolated from earth ground. This rail will provide the charge current for the batteries and the supply voltage for the op-amps in the load circuit.

Given that the brick is an isolated switching converter, it is not the cleanest DC voltage source. To clean the supply lines the input power is run through a ferrite bead and several shunt capacitors and this is seen in Figure 7. The ferrite bead is rated at 500nH and capable of conducting 12A. Using this inductor and several shunt capacitors, we can create a low pass LC filter. Figure 8 shows only 10uF of capacitance, this is not likely enough and in the final design we will replace it with 100uF. Increasing the capacitance on the line will reduce the cutoff frequency to about 20kHz, improving the rejection of the switching harmonics of the isolated converter which has a fundamental switching frequency of 75kHz.

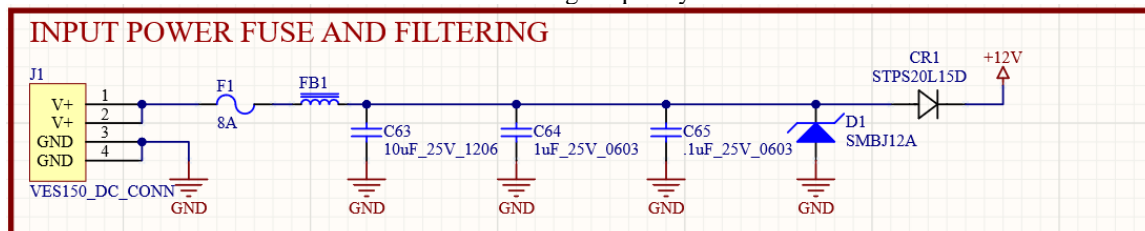


Figure 7: Power Supply Input, Filtering, and Protection

Logic level power will also need to be provided to the system to power the INA3221, ATSAMC21J microcontroller, ADG715 analog switch, and pulling up the I2C bus. All of these devices have 2.7V-5.5V operating ranges, so a 5V supply seems like a natural choice. To provide this rail we are using a Recom R-78AA5.0-1.0, it is a 12V input 5V output switching regulator module. While it is rather expensive it is a highly integrated solution that requires very few external components and no layout effort. The datasheet recommends a series diode on the output to protect the switcher from any external surges and a trimmer resistor to set the output voltage after the diode to 5V. The schematic for the switcher can be found in the following image.

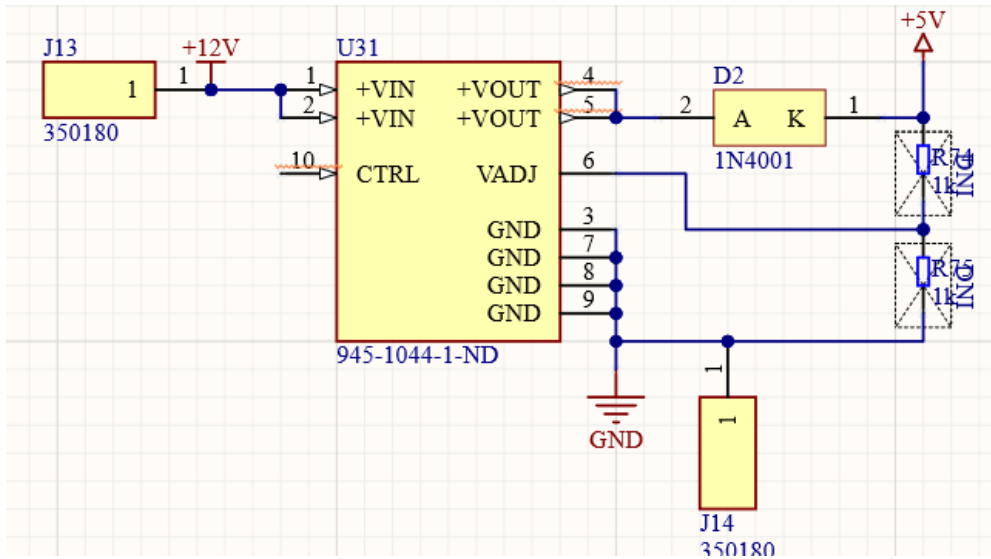


Figure 8: 5V Switching Regulator Module

### 5.3.3 Hardware: Battery Charging Implementation

For the charging circuit, we are using the TI BQ2425. It is a switched mode voltage regulator, designed specifically to charge single cell lithium-ion batteries. The charger will take a 12V input from our primary supply rail, and we have set the input current limit to 1.5A. Our ability to communicate with the device is provided by I2C, a few logic level pins, and resistor choices placed on ILIM pins.

The BQ2425 is a highly configurable device when using I2C to communicate with it. Charge currents, output voltages, charge times and fault states are all managed by the I2C communications. Although, this part did throw a wrench into some of our plans, and we did not have time to properly spec another device. There is no way of changing the address of the device, and since we need eight of them this provided another design challenge. To get around this issue, we are using an I2C multiplexer. Thankfully, Texas Instruments makes exactly the part that we need to solve this problem. This part acts more like a demultiplexer, the main I2C bus goes to its input, and then using an I2C write to the mux, we are able to open one path at a time at the output. In a future revision of this project, this would likely be the first thing we would change. It added significant complexity to our two-wire bus.

One of our design goals is to keep heat in the enclosure at a minimum to relax the stress on the various ICs and the batteries during a characterization cycle. Therefore, we are going to keep the charge current kept fairly low. This will not push the charger very hard due to its switched mode topology. According to Figure 5 in the device datasheet, Efficiency vs Output Current, with a 12V input we can expect the charger to be about 87% efficient. With an 825mA charge current and 5V output, there should only be 0.62W dissipated by the device.

$$P_{Dissipated} = P_{out} * \left(\frac{1}{\eta}\right) = (.825mA * 5V) * \left(\frac{1}{(.87-1)}\right) = .62W \quad (1)$$

$$T_{junction} = R_{\theta JA} * P_{Dissipated} = 32.9 \frac{C}{W} * .62W = 20.4C \quad (2)$$

Using equation 1 and 2, we can estimate that the junction temperature of the charger will be about 20.4 C above ambient temperature. The system is designed to be operated at room temperature, so the charger will stay well within its maximum rated junction temperature of 125C during a charge cycle.

Monitoring the temperature of a lithium-ion battery during charge and discharging cycles is critical to safe operation of the battery. The charger also has interface capability with an NTC (negative thermal coefficient) thermistor. As the temperature of the thermistor increases, the resistance decreases. Using a thermistor as part of a voltage divider with known resistances and supply voltages allows us to measure the temperature of the battery. The BQ2425 has the ability to interface with one of these voltage dividers. We spec'd a 4kΩ thermistor along with a 1kΩ resistor to provide the temperature reads to the device. We are



able to get the battery temperature from an I2C communication with the charger. We will be using this to monitor the temperature during charge and discharge cycles.

### 5.3.4 Hardware: Constant Current Electronic Load Implementation

A programmable constant current load is required for battery characterization. The system will need to be stable under fast transient response conditions for the internal resistance measurement. It will also need to be accurately programmable from a microcontroller and temperature stable during operation. Design work was primarily completed in TINA-TI. The circuit schematic can be seen in Figure 9. It consists of two NMOS transistors in a parallel configuration, with their gates being modulated by an error amplifier. The load current is pulled through a parallel combination of  $10\Omega$  resistors for an equivalent resistance of  $1\Omega$ . The load current is set by a reference voltage generated externally to the circuit. The operational amplifier modulates the transistors gates so that the voltage at its inverting input is the same as the reference voltage. This voltage is applied across a  $1\Omega$  effective resistance to pull the programmed load current from the battery.

This circuit needs a few hundred kilohertz of bandwidth for the internal resistance measurement. This bandwidth is a function of the output compensation capacitor, and the gain bandwidth product of the error amplifier. Most modern general purpose op-amps have sufficient bandwidth for this application. Our primary concern with this circuit was input common mode range and output swing. The TLV9302 fit the requirements well. It has 1MHz of bandwidth, a max supply voltage of 40V, and the input common mode can approach 2V of the supply. While this isn't the best, there are certainly op-amps that do much better, we are able to design around that problem by using 12V and ground to supply the op-amp.

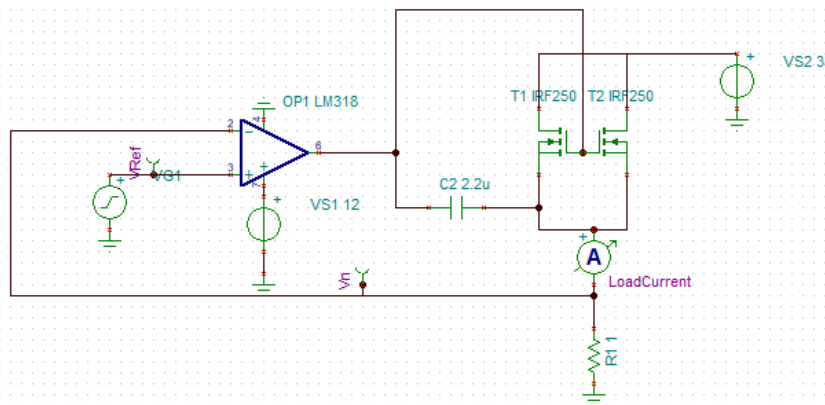
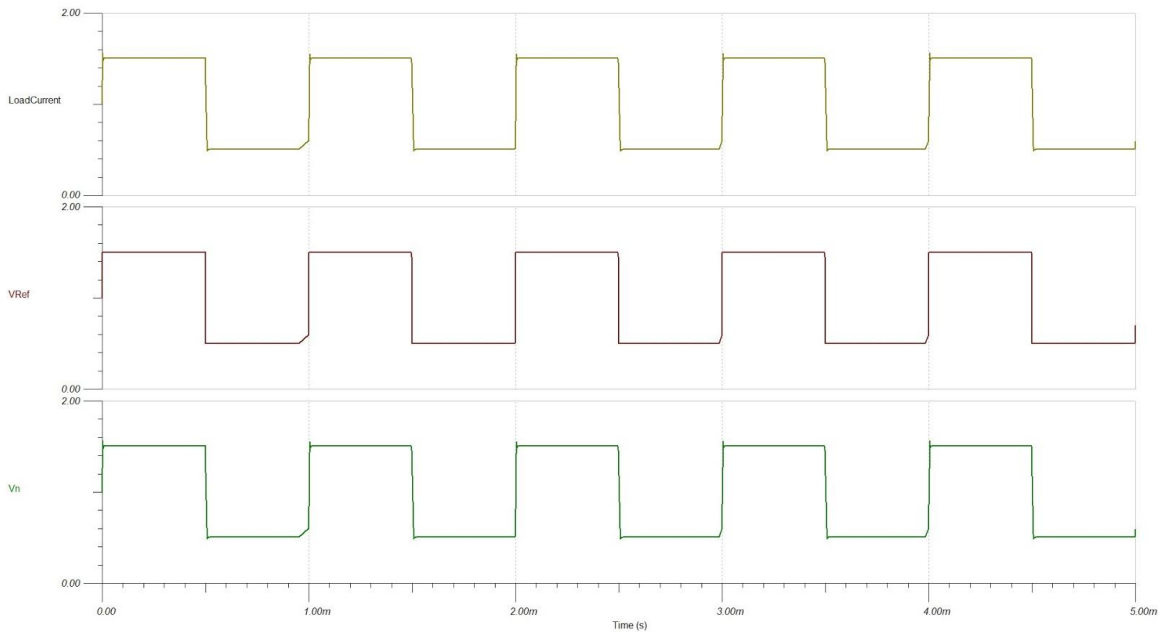


Figure 9: Programmable Constant Current Load

A transient simulation was also performed to get a better sense of the overshoot and response time of the circuit. The test bench to perform this simulation is shown in Figure 9. The voltage reference is supplied by a 1kHz square wave with a 500mV amplitude and a 1V DC offset. The load current is measured by the ammeter in series with the resistor R1. The battery voltage is supplied by VS2 and is set to 3V, although due to the design of the circuit this value is not very significant. The compensation capacitor is set to 2.2uF. The results of the simulation are shown in Figure 10. As expected, the input pins of the op-amp pull to be the same value, this has the effect of setting the load current through the resistor R1. The overshoot of this system for a 1.5A load is approximately 50mA. The settling time is less than 100 $\mu$ s.



*Figure 10: Transient Response of Load Circuit*

To test the phase margin, a test bench to measure the open loop gain of the system needs to be created. This test bench is shown in Figure 11. The magnitude and phase of the top op-amp output is plotted in Figure 12. This is done by breaking the feedback loop and injecting an AC signal of increasing frequency. The direct output of the system is the open loop gain. To prevent the system from performing differently, the broken feedback loop needs to be terminated with the same impedance that it saw before. This is done by attaching a copy of the circuit coupled through large capacitors and inductors. This allows for AC interaction between the circuits but also maintains the correct DC bias conditions. 30 degrees of phase margin is measured at the point of 0dB gain and occurs at 380kHz with a 2.2 $\mu$ F compensation capacitor. This gives a fair amount of stability headroom while also allowing the system to respond quickly.

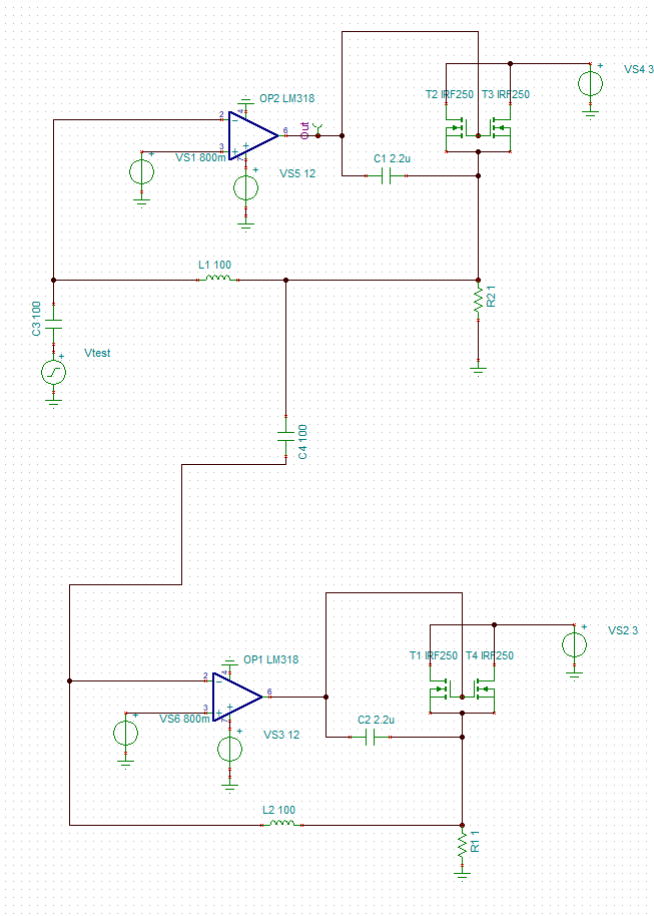


Figure 11: Phase Margin Test Bench

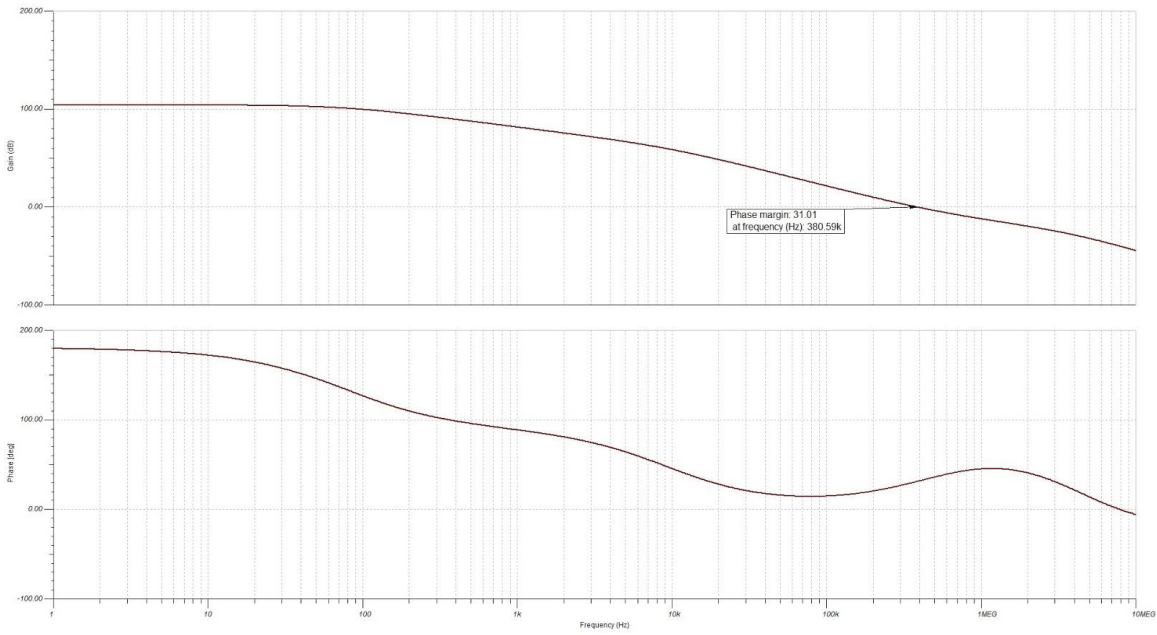


Figure 12: Programmable Constant Current Load Bode Plot

The simulation results are fairly accurate, but manufacturing variance could throw a wrench into the circuit performance. The most likely source will be resistor tolerance. Also, testing at higher ambient temperatures will need to be completed to get a more accurate picture of circuit performance. Testing with physical hardware will also occur once we have access to lab equipment. We will be able to check steady state performance as well as switching performance. Testbenches similar to the simulations will be created to verify the validity of the simulations.

Each of the 8 batteries has its own load circuit and the voltage reference will be provided by a DAC on the microcontroller. Since there is only one DAC on the SAMC21J, we would not have any way of starting or stopping the current draw from an individual battery. Either all of the batteries would be discharging or all would be static. Because the batteries are not guaranteed to discharge at exactly the same rate it would be possible to discharge a single cell too much. The solution to this problem is the ADG715, it is an 8-Channel analog switch. There are 8 series MOSFETs that are controlled using an I2C message. With this device, we are able to open and close paths for the voltage reference to travel to the load circuit op-amps. Each op-amp reference pin is pulled to ground through a 10kΩ resistor to ensure that the node is 0V when the switch is off.

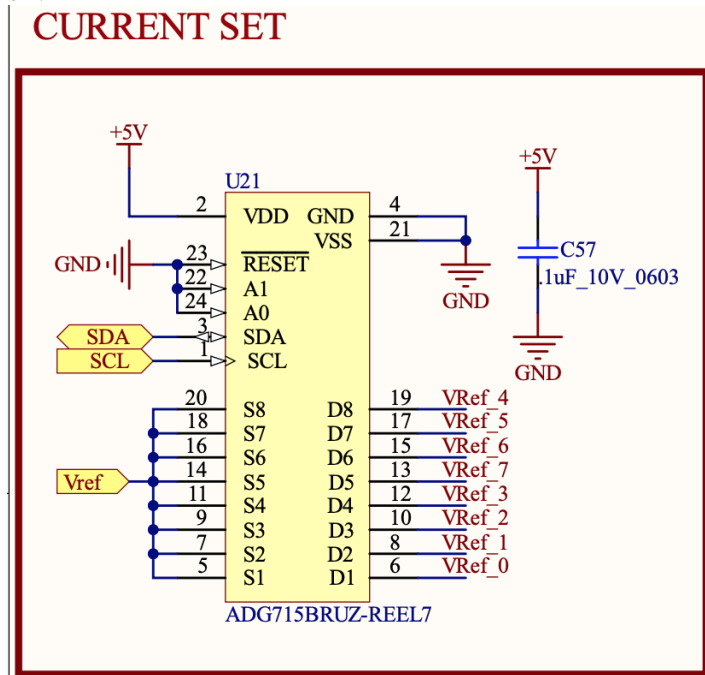


Figure 13: 8-Channel Analog Switch for Reference Voltage

### 5.3.5 Hardware: Voltage and Current Sensing Implementation

Precise measurement of voltage and current is essential to the proper characterization of Li-Ion battery capacity and resistance. Our initial plan was to design our own current sense circuit and ADC buffer in order to use the built in ADCs on the microcontroller. After doing some preliminary research, we chose not to go with this approach. Compensating for temperature effects, manufacturing tolerances, and lack of experience in precision design were all reasons to find another path. The solution we came up with is to use the Texas Instruments INA3221, it is a 3-channel current and bus voltage monitor with I2C compatible communication protocols.

The INA3221 has three internal 13-bit ADC with an ideal 1-LSB step size of 40uV on the shunt voltage measurement. The resolution of the ADC is most important for the internal resistance calculations, our goal is to have about 0.1Ohm accuracy. With the 40uV step, this should be plenty of headroom on the measurements and we should be able to trust our measurements.

There are several open-drain output pins on the INA3221 they are Critical, Warning, Power Valid and Timing Control. When the schematic was drawn we misunderstand the internal logic of the open drain connections, so on the first revision they operate backwards then what we expected. This is a minor fix, and we don't necessarily lose functionality. The lights just turn on opposite of when you would expect them to.

Since all of the results are sent back to our microcontroller using I2C we did not route the open drain pins to the microcontroller, so from a software standpoint nothing changes.

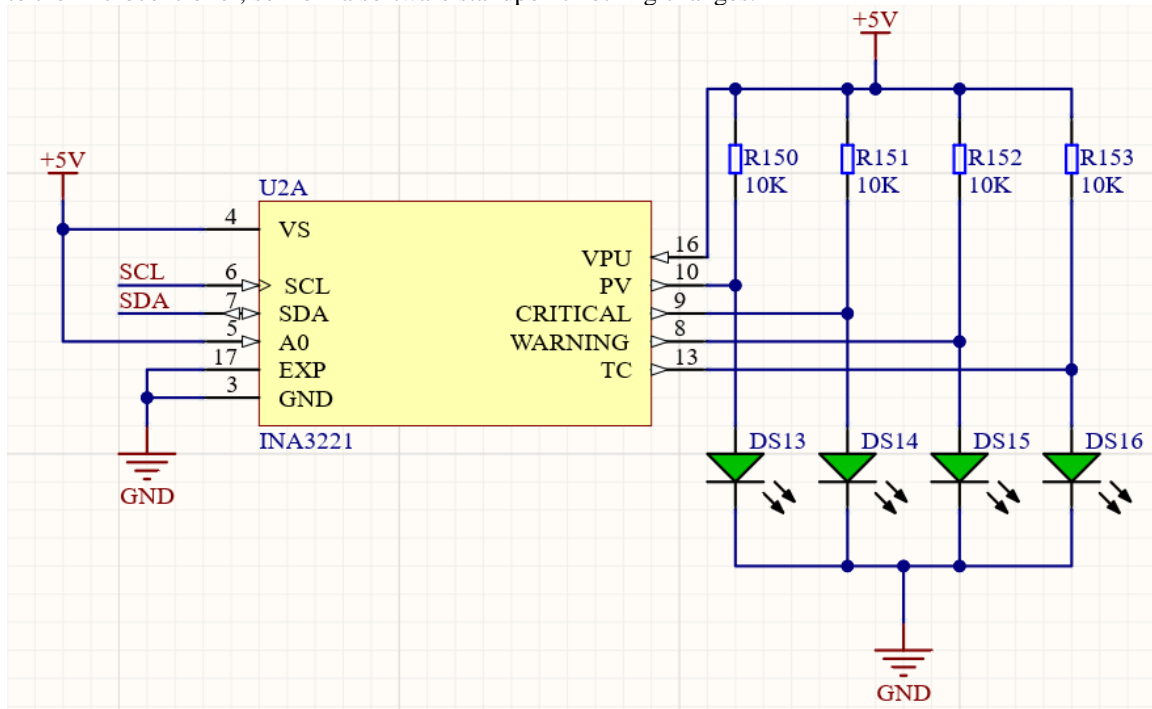


Figure 14: Power Supply and Interfacing for INA3221

Another benefit of this component is the I2C compatibility. Addressing the device is done by connecting the A0 pin to another exposed pin of the device. The possible addresses are described in Table 4.

Table 4: Addressing Option for INA3221

A0	Address
GND	1000000
VCC	1000001
SDA	1000010
SCL	1000011

Current measurements are accomplished by measuring the voltage drop across the 100mΩ sense resistor. To get the current flow values, we will read the averaged voltage dropped across the resistor from the INA3221 and divide by 100mΩ. The tolerance of the resistor is 1%, and the ADC is μV level accurate, so our measurements should be very precise.

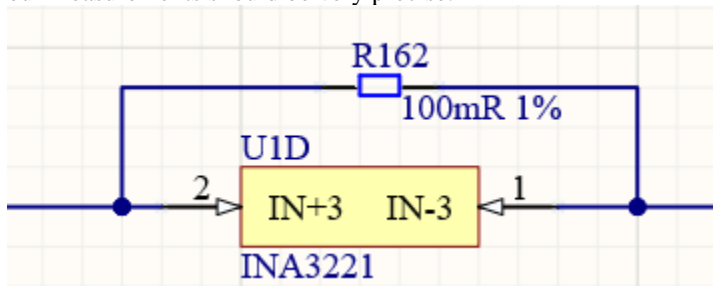


Figure 15: Sense Resistor and ADC Connections for INA3221

Battery voltages are constrained to 2.8V to 4.2V, so it is unlikely that we will need input protection between the sense resistor and the ADC inputs. They are rated up to 30V relative to device ground. Filtering on the other hand may be necessary, but hopefully software averaging is good enough to handle the noise present in the system.

### 5.3.6 Hardware: Microcontroller Implementation

All of the information gathered by the various chips must be collected and sent to the PI for interpretation. As well as collecting the data and sending it over CAN, each chip must be told what to set initial values and when they should perform their respective operations. To accomplish this an Atmel ATSAMC21J16A-AUT was utilized (AKA SAMC21J). The SAMC21J has the ability to communicate via CAN and I2C which was 2 requirements that the board had to meet. Along with being able to communicate using both I2C and CAN at the same time, the team has experience designing and programming this chip. The familiarity with the chip made it desirable with the team. The SAMC21J has very few requirements to operate it. The datasheet only states that we supply between 3.3-5V, and that we apply filtering to the power input. We accomplished the 5V power supply by using an inset part number of power supply switching regulator. We then added the filtering circuit. See Figure 16.

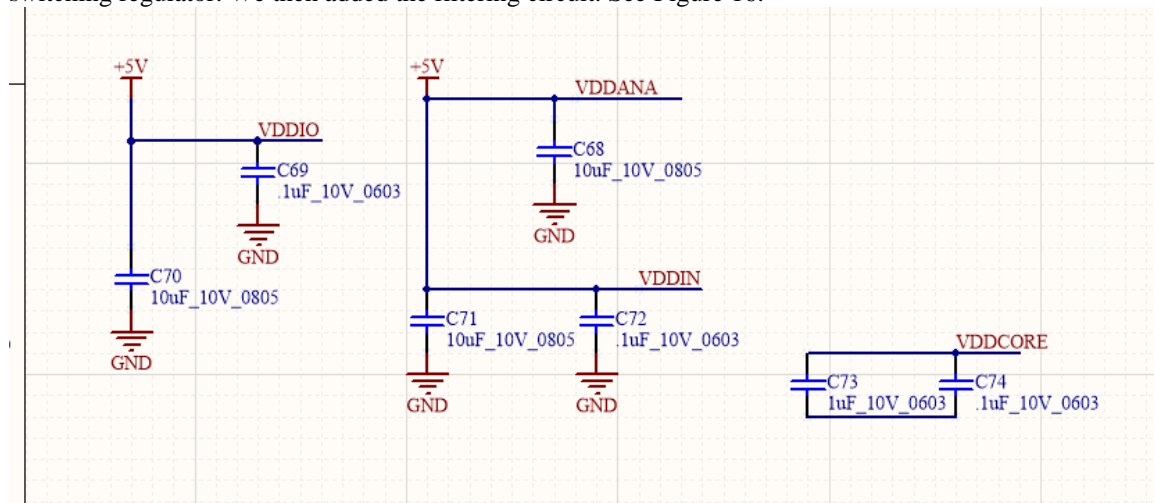


Figure 16: Filtering Circuit for the SAMC21J

While the SAMC21J can communicate via CAN, it needs another IC chip to help with the operation. The team chose to use SN65HVD232DR. The transceiver can operate between voltages of 3.3V and 5V which is desirable since there is already a planned 5V power rail.

### 5.3.7 Software: Raspberry Pi Implementation









When the CAN messages are sent to the Pi, we use python with socket can to parse the messages and save the data accordingly. Two major examples of parsing the data includes saving which slots the batteries are connected to so that we can assign battery labels and recording test data to the database. We are using a Flask web application to create a user interface and handle the processing of data. This allows us to create the user interface using HTML and Bootstrap, which provides flexibility for if the device connecting to the site is the small raspberry pi touchscreen or a full sized monitor. The user interface has a set of pages that are intended to be used on a computer, which have more advanced options and would allow the user to view test data. The user interface also has a set of pages that are meant to be used on the raspberry pi, which has a small touchscreen. These pages are meant to be simple and are only intended for actually running the test. We use sockets to be able to automatically update the screen based on the most recent data. For example, before a test begins, the page displaying what to label the cells updates as new batteries are connected, or while the test is running information is displayed about the batteries.

# Automated Li-Ion Battery Characterizer

Set Up Test

Figure 17: User Interface page the raspberry Pi boots to

## Connected Battery Labels

Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot 8
 27							

Begin Test

Figure 18: User Interface Defining What to Label Each Cell

## Available Packs

New Pack

Name	Created
Test	2020-11-10
Eli	2020-11-10

Figure 19: User Interface for seeing all packs previously created

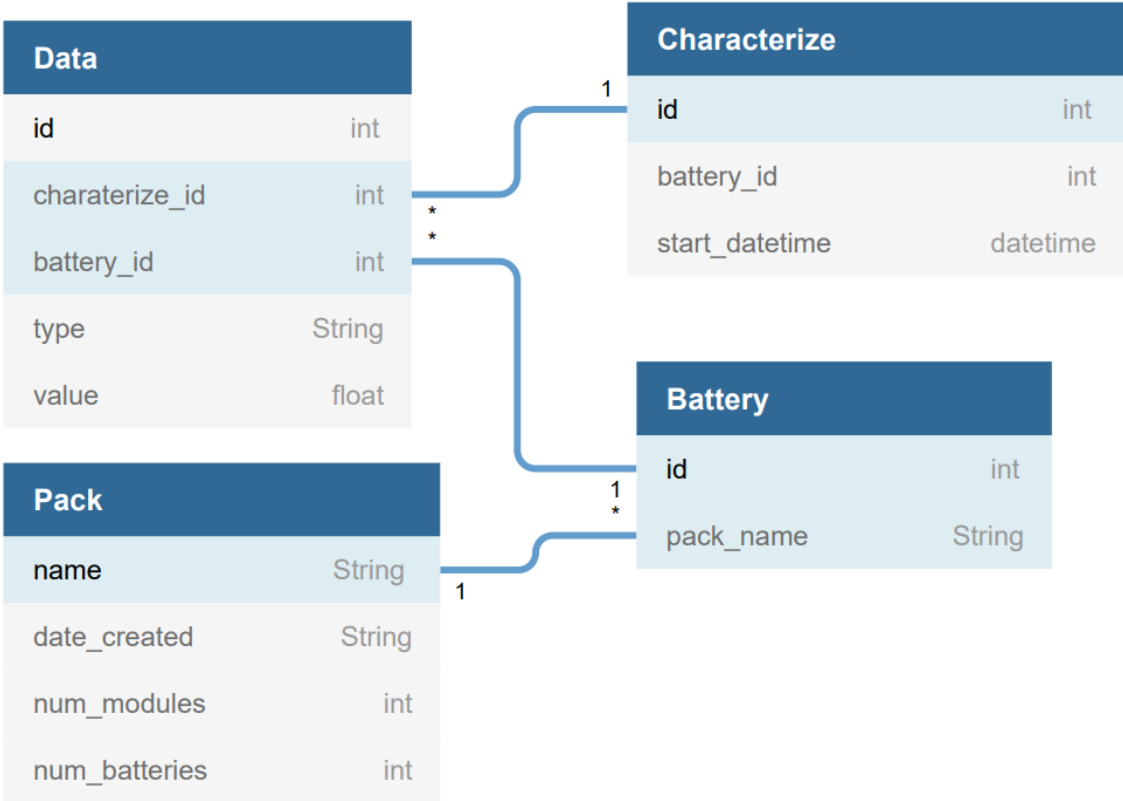


Figure 20: User interface for creating a new battery pack

### 5.3.8 Software: Embedded Software Implementation

When the device is turned on, everything is initialized to only listen to CAN messages from the pi. These include asking for the number of batteries connected to which slot or starting the characterization. Sending out the number of batteries connected allows the pi to do initialization on its side before the characterization starts. When the start message is received, the application will start the characterizing



which is shown in Figure 21. To characterize a battery, the battery must be at a full charge which is 4.2V. To charge the batteries we are using an IC which communicates through I2C. Whenever the batteries are being charged or discharged the application must be monitoring the current and temperature of the batteries. If the temperature drops below 10C or above 60C or if the current is more than 3A then we must stop the application before the battery catches fire. The IC is doing all the work charging the batteries, the application needs to set registers over I2C to start and at what rate. When the batteries are fully charged, the application will discharge the batteries to 2.8 at the same rate when charging. To discharge, the application needs to set an DAC output to the rate which we are discharging. Setting the DAC output to 0.825V will discharge the battery at 0.825A from the hardware stated before. During discharging, the application needs to log the voltage and current to find out how long it takes to discharge. The pi is logging all the data into a database so the application will send the data over CAN at an interval. The interval is unknown since full system tests couldn't happen, but it would be around every 5 seconds up to every 30 seconds. After the batteries get to 2.8V, the batteries need to be charged back to full. The next test needs the batteries to be at a full charge in order to get correct output. The reason that the application fully discharges the batteries before doing this test is because it is optimal to store batteries at more than 80% charge. So, no matter where this test is done the batteries are going to need to be charged. After waiting until the batteries are charged again, the application will set the DAC out to 2.5V for one second and then log the voltage by sending another CAN message. Then this will happen again but setting the DAC to a 1.0V output for one second and then logging that data. Finally, the characterization is done and sends the finished CAN message and then starts checking for batteries again.

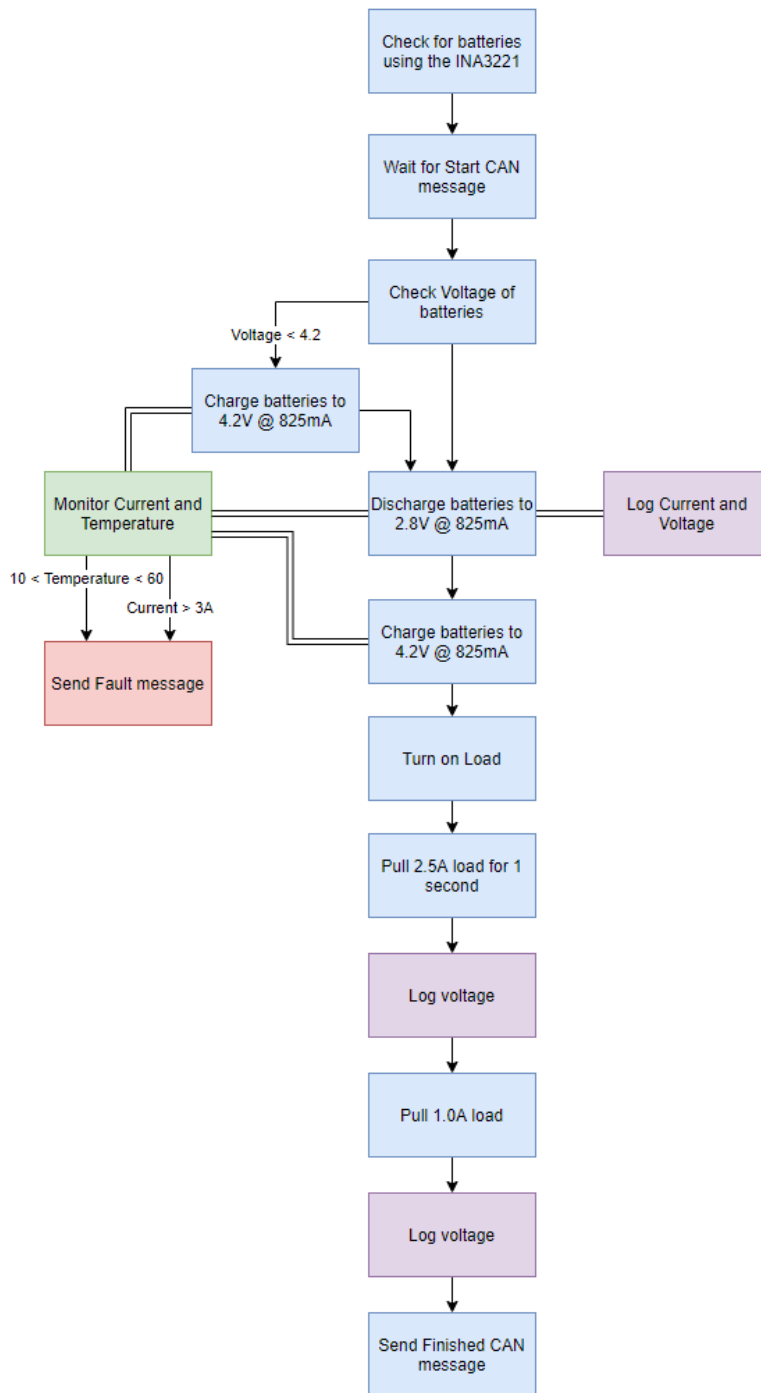


Figure 21: Embedded Software Testing Procedure

#### 5.4 NON-FUNCTIONAL TESTING

Testing our non-functional requirements was difficult because the board was not finished, and we could not run a system test. One test was to characterize one battery and log its data every second. After getting the output from this then we would take data out and then get the output without the missing data. This will allow us to figure out the interval of when to log data to allow us to log the least amount of data without

losing any precision. Since this will run unsupervised, the application must turn everything off during a fault and message the user that a fault has occurred. So, either by setting the current too high or making it too hot (without a battery connected) and make sure the message is received and nothing is happening on the board.

## 5.5 PROCESS

The testing shall be broken down by sections. The top-level testing will be:

- User interface testing
  - Confirming that the user will be able to start, stop, and review the test
- Charge and discharging of the batteries
  - Confirming that the batteries are being charged and discharged in a safe manner
- Data reporting
  - Confirming that the microcontroller is recording and reporting the data as required

A detailed breakdown of how each of these top-level tests shall be performed is detailed in section 5.3.

### 5.5.1 Evaluation Board Assembly (Load and micro)

Materials required for the assembly process are a soldering iron, solder, and some flux. With all of the parts, an hour or so is enough to assemble either of these boards.

### 5.5.2 Load Prototype Testing Process

The purpose of this prototype is to verify simulation data with real world performance and get a handle on the thermal performance of the MOSFETs. The following equipment is required to perform the tests:

- Triple Output DC Power Supply
- Multimeter in ammeter mode
- Function Generator
- Oscilloscope

To check current set accuracy, we will set the current source to 2.8V, 3.5V, and 4.2V. For each of these source values we will run the reference voltage from 100mV to 3V in 100mV increments and measure the current at each step. The measured current should track the reference voltage 1:1. So a 100mV reference voltage ideally equals 100mA and so on. Each value will be recorded and plotted against the ideal value. Verification of the device stability is done by pulsing the reference voltage with a function generator. The DC power supply providing the reference voltage will be replaced with a function generator. The function generator will be set to square wave mode at a variety of frequencies and rise/fall times. According to simulation the circuit has about 380kHz of bandwidth before the unity gain crossover frequency, so we should be able to get a sub 100kHz square wave comfortably through the circuit.

The initial design calls for two transistors to conduct the load current. We made this decision in an effort to reduce the thermal load on each transistor and be able to use smaller package sizes. To check if this is necessary we will measure the temperature of the transistor using a thermocouple. Taping the thermocouple to the top of the transistor package, and monitoring the value on a compatible multimeter should provide a fairly accurate view of the junction temperature of the transistor. At each of the battery voltages used in the accuracy test, a 500mA load, 1A load, and max load will be pulled from the source. Temperature will be recorded at 5 minutes of run time and 10 minutes of run time. Once the device cools back to ambient the next set will be started.

### 5.5.3 Voltage and Current Measurement Testing

For the voltage and current sensing, we are using the Texas Instruments INA3221 along with a 100mΩ sense resistor. The output is communicated through I2C. To eliminate initial variables we will provide device power and test voltage with a bench DC supply. The INA3221 has a power good pin with an LED connected to it. When we provide device power, the power good LED should turn on. To test the accuracy of the measurements, we will need to provide a dummy load to the circuit in the form of a potentiometer and apply a voltage with the bench supply. Using a multimeter, we will get baseline results for current that

we can compare to the INA3221 measurements. To read the output of the device, we will use the serial decoder on a lab oscilloscope to read the I2C messages.

The INA3221 will be continually monitoring the current into and out of the battery. We will be using the digital output to communicate if there are any fault conditions. To test this, we will set up the device with a low current limit to make the test safer to conduct. Then we will force a current below that limit and steadily increase it. Once we cross the threshold, we should receive a message on the I2C bus that the current limit has been exceeded.

#### 5.5.4 Main Board Power Rail Testing

Using a triple output DC power supply, 12V will be provided to the board. Using a multimeter, measure across the input capacitors at each of the chargers to ensure that 12V is making it to each one. Then measure at the output of the 5V switching supply and it should be around 5V.

#### 5.5.5 Main Board Load Testing

Testing the load circuit will require the microcontroller, analog switch, and at least one of the load circuits populated. Program the microcontroller to enable the battery 0 load circuit by enabling the DAC and setting a value and sending an I2C message to the analog switch to connect the DAC output to the op-amp reference pin. Using a multimeter, the DAC output voltage should be present at the non-inverting terminal of the battery 0 op-amp. Once this measurement has been made, break the connection between the DAC and op-amp by turning off the switch with an I2C communication. Now the voltage at the non-inverting terminal should be 0V.

The design was validated by the prototype we made early in the semester, so we just need to validate that it was assembled correctly. Using a lab power supply, connect the leads to the battery connection terminals, and program the circuit to draw 100mA. Verify on the lab supply current read out that 100mA is being drawn.

#### 5.5.6 Main board Battery Charger Testing

Testing this circuit will require careful attention as a battery will be involved. We will create a test bench that has just the battery charger, its required components and a single lithium ion cell. The cell will be monitored for temperature to ensure that it does not exceed the temperatures listed by the JEITA standards. It will also be fused to a fairly low current level. These precautions should prevent any major disasters. Of course, a fire extinguisher rated for lithium fires will be close at hand. The microcontroller will provide the charge enable signals over I2C, these messages will be monitored over the oscilloscope. Current into the battery will be measured by a multimeter and the voltage profile will be monitored on the oscilloscope. We will expect a constant voltage charge at the start to precharge the battery, once this initial stage has completed, the charger will move into a constant current charge. This will take place until the battery has reached approximately 90% of its full charge. Once that threshold has been crossed the charger will again move into a constant voltage charge to finish the charging cycle.

#### 5.5.7 Microcontroller Testing

The microcontroller we will use is an Atmel ATSAMC21J16A-AUT. The main features that will need to be verified is:

1. The microcontroller can be programmed
2. The microcontroller can communicate to the pi via CAN
3. The microcontroller can communicate to the other circuits via I2C

To verify that the microcontroller can be programmed, we will write a simple program that will simply turn on an LED. This will provide a visual indicator that the microcontroller was programmed.

To verify that the microcontroller can communicate via CAN properly, we will connect the system to a known working CAN device (provided by the client), that will send and receive messages over the CAN network. When the CAN network on the microcontroller is confirmed working, the next step would be to verify CAN working on the pi. Similarly, to verify the microcontroller we will write a simple program to make the pi send and receive CAN messages.

Finally, to verify that the I2C communication bus is working, we will have the microcontroller send messages to each of the IC connected. Probing the bus line with an oscilloscope to determine if we see the proper messages being sent over the bus.

### 5.5.8 Raspberry Pi Testing

We had originally planned to add unit testing to internal functions. Due to the lack of full testing data, we did not create the functions that would analyze the batteries and attempt to group them into modules. Without those functions, there are not really any internal functions that are unit testable. Almost all functions are either tied directly to page rendering, database management, and CAN communication. This means that most of the testing has been with manual verification. This has worked well to show where problems are, as Flask does a great job of showing debug information when something goes wrong. Flask provides a full stack trace that I can be used to find exactly what couple of lines are causing the issues, and what code can be improved to prevent it from happening. For user interface pages with input forms such as selecting or creating a pack, we have implemented client-side verification that prevents the data from going to the server without being filled out, which could cause issues on the server with the database.

### 5.5.9 Embedded Software Testing

The testing for the embedded code was difficult since there wasn't any hardware to test with until the end of the semester. We did make a micro test board so we could test the I2C output from the microcontroller, the CAN output and the DAC output. Being able to see the output of the I2C was helpful, but since it was not connected to anything else, we could not test the acknowledgements. After the full board is put together, we can again test the DAC output, the CAN output and then test talking with every IC that uses I2C. Once all is working then we can go into a full characterization test.

## 5.6 RESULTS

### 5.6.1 Microcontroller Test Results

The micro was mostly expected to do the following, control whether the circuit is charging or discharging the battery, communicate to the charge/discharge IC via I2C, and communicate with the PI via CAN. To begin, we confirmed that the micro was getting 5V. When measuring the power input, we found that the micro was getting 4.3V. The reason we were getting .7V less than what we expected was because the switching power supply wasn't outputting 5V, as well the voltage was dropped further because of the fly back diode. To fix this, we followed the equation found on the power supply's datasheet to calculate the resistor value to force the power supply to output 5V. See Equation 3.

$$R_{up} = \frac{100k\Omega(V_{ref}-5V)+V_{ref}(30.9k\Omega)(10k\Omega+10k\Omega)}{10k\Omega(5V-V_{ref})-V_{ref}(30.9k\Omega)} \quad (3)$$

To make sure the micro was being programmed and outputting the correct voltage, we wrote a simple program that would flip the relays. We expect that the voltage going to the relay drivers would be 5V. When measuring the pin, we found that the voltage was 5V. We initially planned on adding LEDs to some of the I/O pins to help confirm the SAMC21J was working, but the LEDs were never added to design due to time constraints. To confirmed that the micro was being programmed and could set a pin high and low, we had to modify the code to flip the relay. By using the relays, we were given an audio confirmation that the SAMC21J was being programmed correctly.

The I2C bus was where we ran into the most problems. We found if we only had the analog switch on the I2C bus, we were able to read and write to the part (see Figure 22)

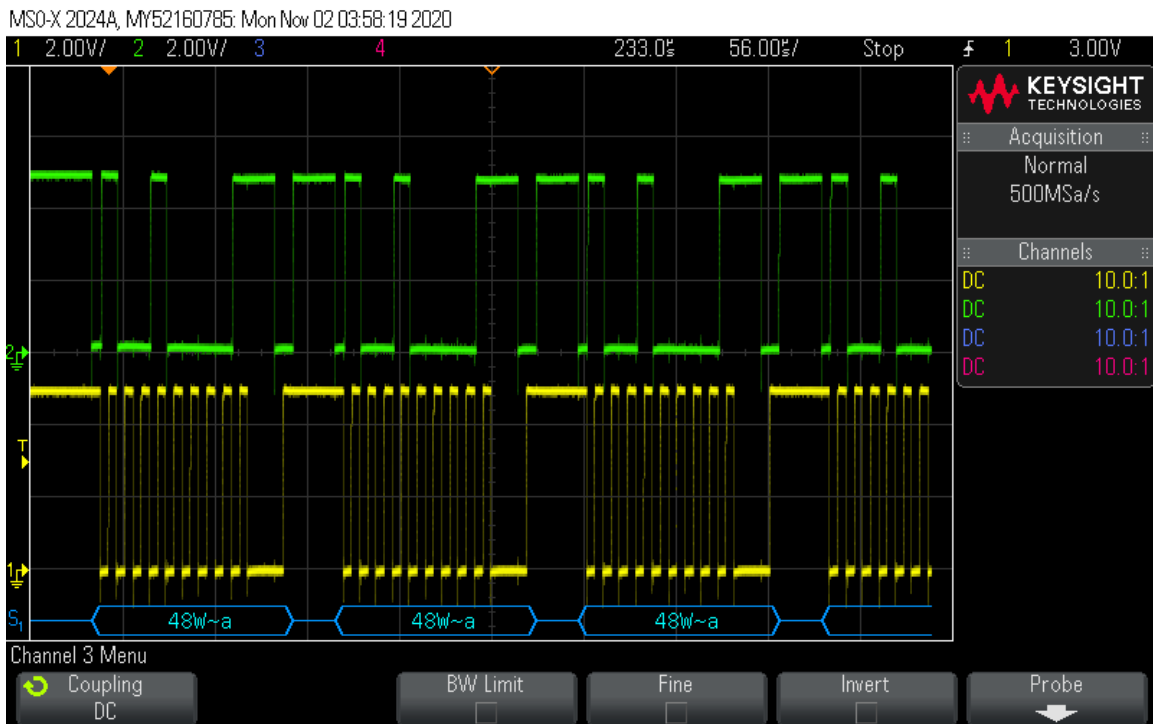


Figure 22: Picture of the I2C Bus

When adding another part, the I2C bus would break. Either SDA and SCL mirrored each other (see Figure 23), SCL wouldn't be pulled low enough to be registered as logic low, or SDA and SCL would be pulled low at all times. We couldn't determine the cause of this bug. We measured the pull-up resistor on SDA and SCL to be  $500\Omega$  and  $50\Omega$  respectively. This is even though we put a  $10k\Omega$  on each of the SDA and SCL lines. We confirmed that both the SDA and SCL lines were connected correctly to each of the IC parts and that the ICs were installed correctly and were being powered. As well as checking the hardware, we confirmed that the software was correct. Despite all these steps, we could not find out why the I2C bus only worked when the analog switch (ADG715BRUZ-REEL7) was installed. At one point we were able to communicate with the I2C multiplexer (TCA9548ARGERQ1), but in later tests we couldn't replicate the tests seen. When both the ADG715BRUZ-REEL7 and the TCA9548ARGERQ1 were install simultaneously.

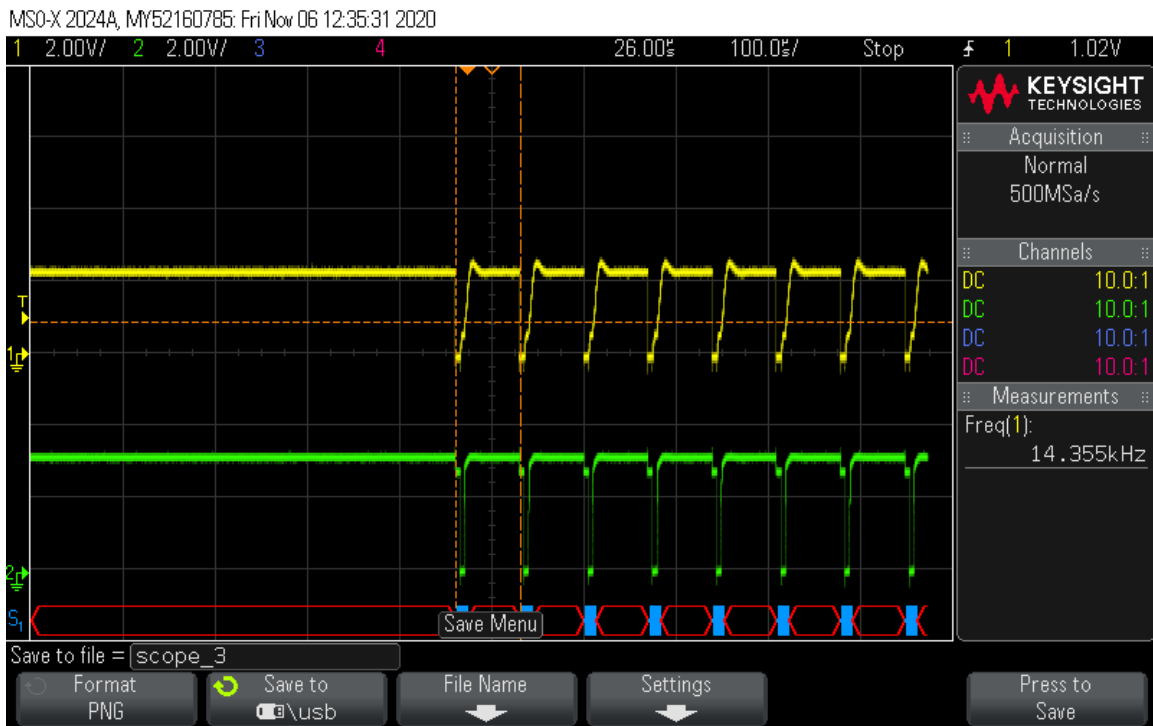


Figure 23: SDA and SCL Mirroring Each other

For CAN we had a device that could read CAN messages and print out what was read on the bus. Using this device and an oscilloscope, we were able to see messages that we sent the board and the board sent to the device. We were confirmed that the CAN bus was working via an oscilloscope (see Figure 24).

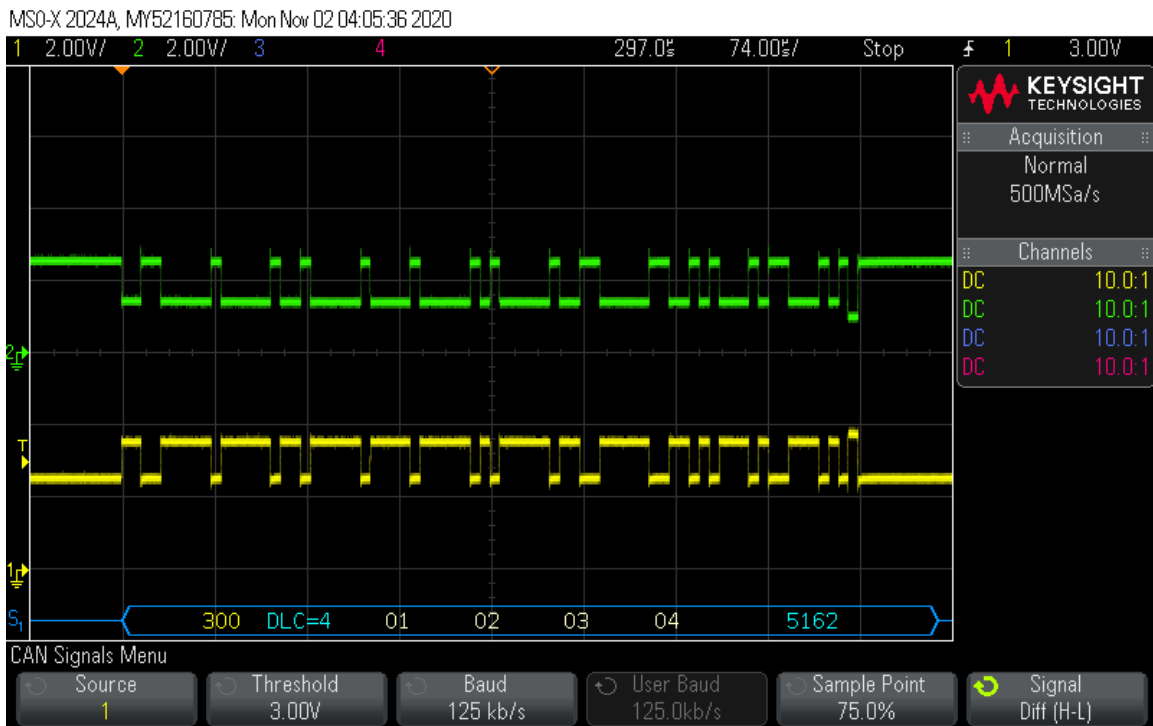


Figure 24: Working CAN Bus

### 5.6.2 Load Prototype Test Results

Overall, the electronic load circuit worked well. Figure 25 is the measured data from the prototype we designed early in the semester. The red curve is the ideal line, if the circuit was perfectly linear. The battery voltage was provided with a DC power supply, and the current was measured by one of the bench multimeters. The blue curve was generated with the power supply set to 2.8V. The yellow curve is for the supply set to 3.5V, and green is 4.2V. These voltages were selected as a low battery voltage, mid and high voltage.



TLV930 Measured Current(mA) vs. Vref(V) Battery = 2.8V, 3.5V, 4.2V

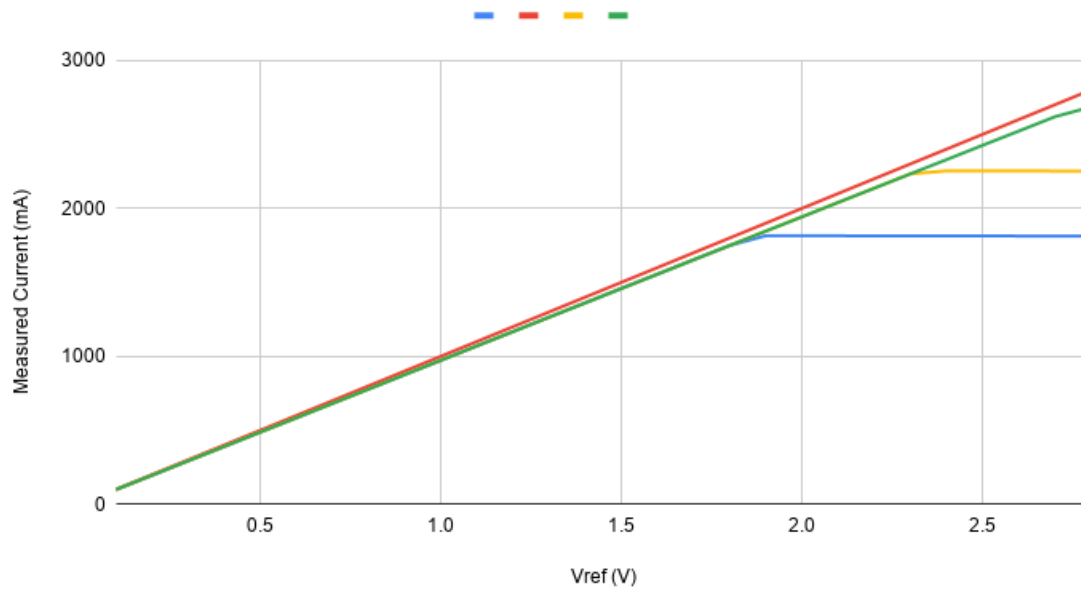


Figure 25: Measured Load Current against ideal with 2.8V source

An aspect of the performance we were not expecting, but should have, was the saturation points of the MOSFETs. Due to this limitation, we will not be able to discharge the batteries as quickly as we were expecting to. But, given our desire to keep heat down this should be fine. No matter the battery voltage, we will be able to pull up to 1.8A through the entire battery charge.

The measured load current divergence from the expected value is due to variance in the load resistor. This design is dependent on the accuracy of these resistors. One possible solution to this problem is to replace one of the 10Ω resistors with a 10Ω trimmer potentiometer. This would allow us to more precisely set the effective resistance of the circuit. We could also measure and sort the resistors to offset any imbalances in their values. Either of these would get the lines to match the ideal more closely.

Power dissipation in the transistors was a concern with the design. So another important aspect of the prototype was to get real world data on the temperature of the transistors. Starting with only a single transistor the data in the following table was generated. Between each test, the transistors were allowed to return to ambient temperature.

Table 5: Temperature measurements for load prototype.

Source Voltage (V)	Current (A)	5 Minute Temperature °C	10 Minute Temperature °C
2.8V	.5	51.7	51.7
	1	68.5	68.2
	1.8	79.4	82.9
3.5V	.5	59.9	60
	1	84.9	84.6
	2.2	104.8	106.6
4.2V	0.5	68.5	68.5
	1	98.2	96.4
	2.6	123.5	136

As expected, the power dissipation was the largest for the 4.2V source voltage, because the voltage drop across the transistor is the largest at that point. The combination of the large voltage drop and high current results in a very toasty junction. Fortunately, we shouldn't need to run the system that hard for very long. The internal resistance measurement will only last a minute or two before it is completed. And the capacity measurement will be a much lower current draw over a long period of time.

The main result we can draw from this experiment is that we do not need to use two transistors on the load circuit. A single transistor handled the load just fine, only when we got into some serious load conditions the single transistor started to struggle.

### 5.6.3 Main board Power Rail Testing Results

Unfortunately, we were unable to get one of the power bricks that we specced to power the main board. That being said, we verified the connector and its power output so it is certainly going to work just fine. None of the components it powers are particularly noise sensitive.

The 5V switching module that we specced works as expected. A scope trace of the voltage output can be seen in the following figure. As expected the DC value is sitting at 5V, and there is a small amount of ripple. Given that this is a switching supply, this is expected.

### 5.6.4 Main Board Battery Charger Testing Result

Testing the charger proved to be quite the challenge during the last few weeks. Given the significant trouble that we had with I2C we were never able to properly test the chargers. We were able to successfully communicate with the I2C mux and get a master write request to a charger but it never acknowledged the message. We are not entirely sure why this happened at this point, see section 5.6.1 Microcontroller Test Results for more details on the I2C debugging process.

### 5.6.5 Main Board Load Testing Results

The stability and thermal performance of the electronic load circuit was verified on the prototype. The only difference between the prototype and the main board is the source of the voltage reference to set the current. While testing the prototype, all of the relevant voltages were generated using a lab power supply. But on the actual device, the reference voltage for the electronic load comes from the microcontroller integrated DAC.

During testing we were able to verify that the switch for the reference voltage control responded to I2C messages and could open and close paths as we directed it to. The mistake we made with this part was the reset line, it is an active low device and we tied the pin to ground. By grounding the pin, we put the switch into a continual reset state and initially it was not able to acknowledge the I2C request. Once we cut the trace connecting the pin to ground, and mod wired the pin to VDD, the device began functioning properly (see Figure 26). At this point we set the DAC to generate a 2V output and we were able to measure it at the expected locations on the board. The below image is the mod wire job to the reset pin of the switch. The tape on the board is to prevent the mod wire shorting to ground.

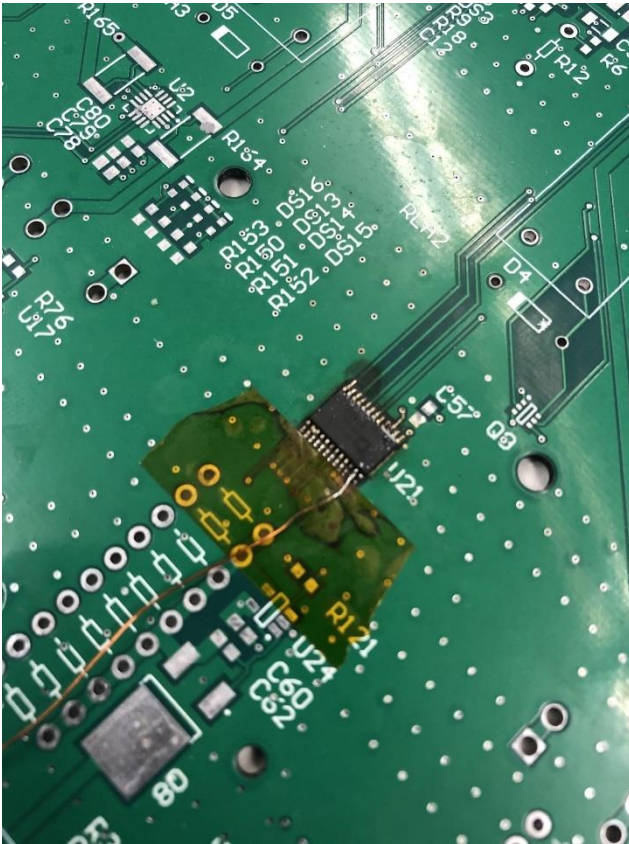


Figure 26: Mod Wire to Reset Pin

### 5.6.6 Embedded Software Testing Results

The first test that I ran was making sure the CAN was working on the board. Since I already had working code for this before, I connected another CAN node, and everything worked as expected. Both nodes could send and receive messages. Next was testing DAC output and we were able to get the output to be 0V all the way up to the max of 2.5V. Lastly before doing a full test was communicating with all the ICs through I2C. The first IC we tried communicating with did not work because its reset switch was pulled the wrong way. We were able to use an oscilloscope to see if the I2C output was correct and it was. So, communicating through I2C was working but we were only able to communicate to half of the ICs. Since the 2 ICs we could not communicate with (see Section 5.6.1 Microcontroller Test Results) were the charger IC and the measurement IC, we could not test charging or discharging because of safety reasons.

### 5.6.7 Web App Testing Results

The testing of the web app started with verifying that the flask application is able to start, which validates that all of the libraries are accessible. From there, we can validate that we can access the website by typing the pi's ip address and port 5000 into our browser. Once we connect to the main homepage, we can then navigate around to the several pages and verify that all pages are appearing as intended and have proper functionality. As pages got added, we needed to verify each page's behavior. Specifically, for pages that take input from the user, we validate that the information is validated on the client side and is received by the server correctly. Once each page has been tested, we test the flow between pages. Specifically, we checked that from the limited interface of the raspberry pi display, that you can move through the steps of setting up and starting the test and can abort it if needed. Once the flow is done being tested, we were able to validate that the server back end is

completing its required tests such as sending and receiving CAN messages. An example of this was to check that when the user started setting up a test, the server creates a new thread that will periodically send a request to the test module to get the positions of the batteries. It was important to verify that the thread will run and send a request once per second, without impeding the functionality of the rest of the server to handle requests. That specific test allows us to see that the pi can communicate back and forth with the testing module. These tests allowed us to verify that the web application behaves as expected for all implemented features. Of course, full functionality was limited since we were not able to run full tests with the testing module, but simulated data allows us to check that everything behaves as expected for the web application.

## 6. Closing Material

### 6.1 CONCLUSION

Coming to the end of the project there is still much work to be done. We think that we made a solid start on this project, and that it has a good foundation to build on if PrISUM is going to pick it up and continue on. The primary need is some more time, we do not think that the hardware problems that we encountered in the last few weeks are unsolvable and that with time to get some fresh parts and maybe a small board revision we could get the hardware into a working state.

Hypothetically if we were working on this next semester our first step would be to order some fresh parts. The ones currently on the board have been through a reflow oven several times, and a fresh version of the board might be a helpful mental reset. We also might make a quick hardware revision to improve testability. Primarily, we would be adding some more test points and spacing out some of the routing. The biggest issue we had while testing was getting to the important signals on the board, and having better test points would have made our lives much easier. After the new board is put together, we would get communication with every IC working and then start charging and discharging batteries to get accurate data for the pi to do its calculations.

### 6.2 REFERENCES

- [1] Opus BT-C3100 Digital Battery Charger. “Opus BT-C3100 V2.2 4 Bay Digital Battery Charger.” 18650 Battery Store. <https://www.18650batterystore.com/Opus-p/opus-btc3100-v2.2.htm> (retrieved April 25, 2020).
- [2] Qian, J. (n.d.). Li-ion battery-charger solutions for JEITA compliance. Retrieved April 25, 2020, from <http://www.ti.com/lit/an/slyt365/slyt365.pdf?ts=1587833638791>

### 6.3 ACRONYMS

**AC:** Alternating Current

**CAN:** Controller Area Network

**DAC:** Digital to Analog Converter

**DC:** Direct Current

**IC:** Integrated Circuit

**I2C:** Inter Integration Circuit, communication protocol

**IEEE:** Institute of Electrical and Electronics Engineers

**JEITA:** Japan Electronics and Information Technology Industries Association

**LED:** Light Emitting Diode

**PCB:** Printed Circuit Board

**SCL:** System Clock

**SDA:** System Data

# Appendix A: Main Project Schematic and PCB

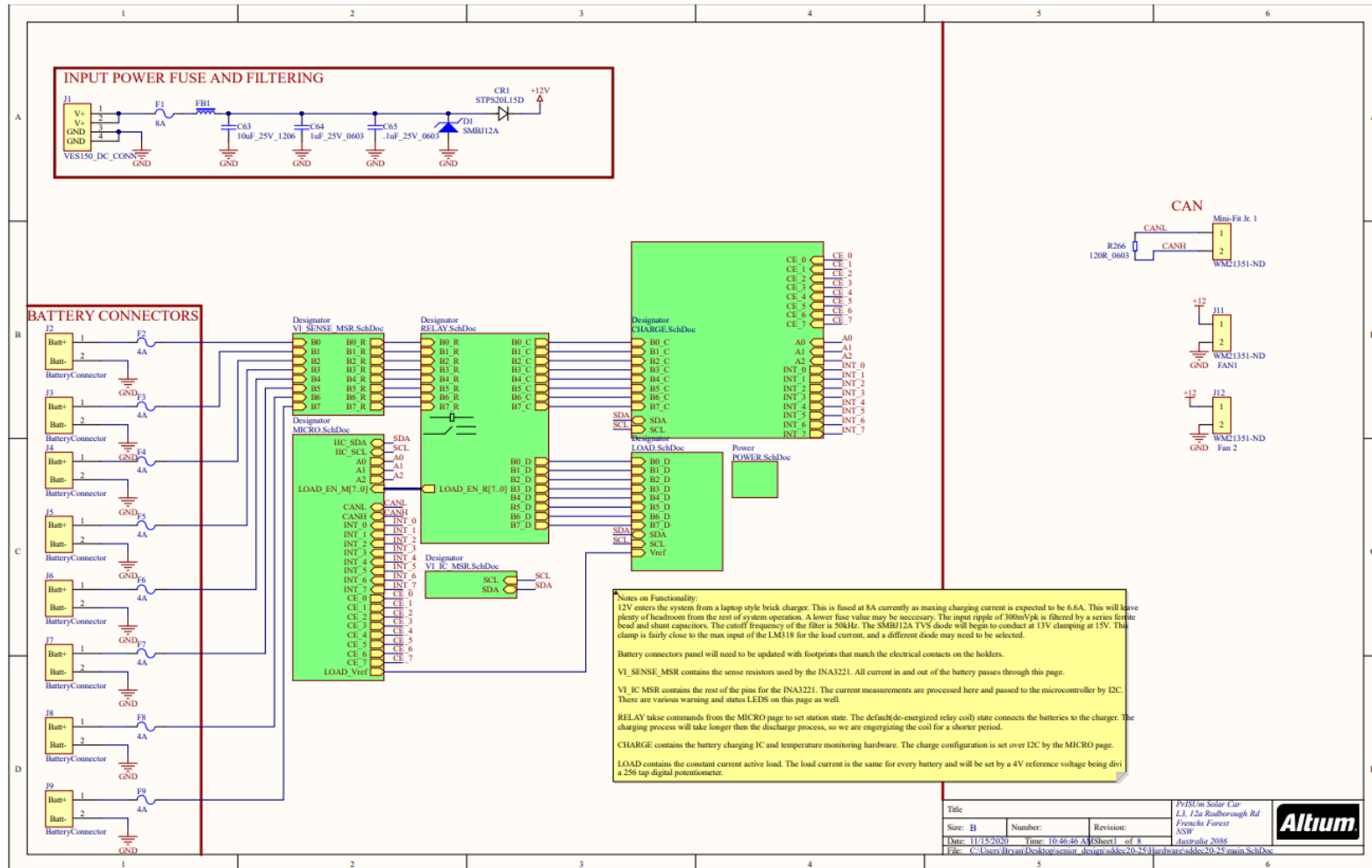


Figure 27: Schematic of Main Page

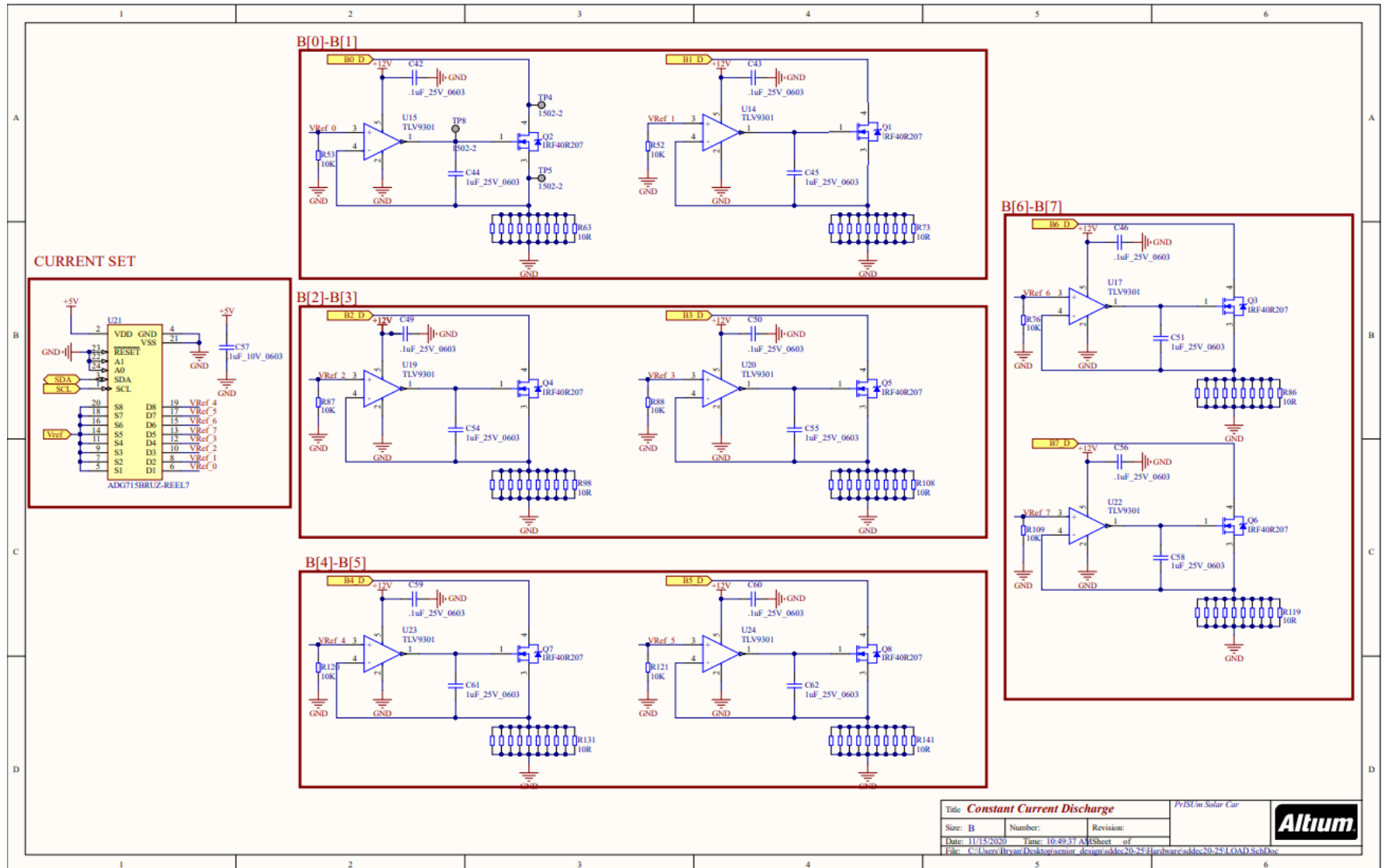


Figure 28: Load Circuit Schematic

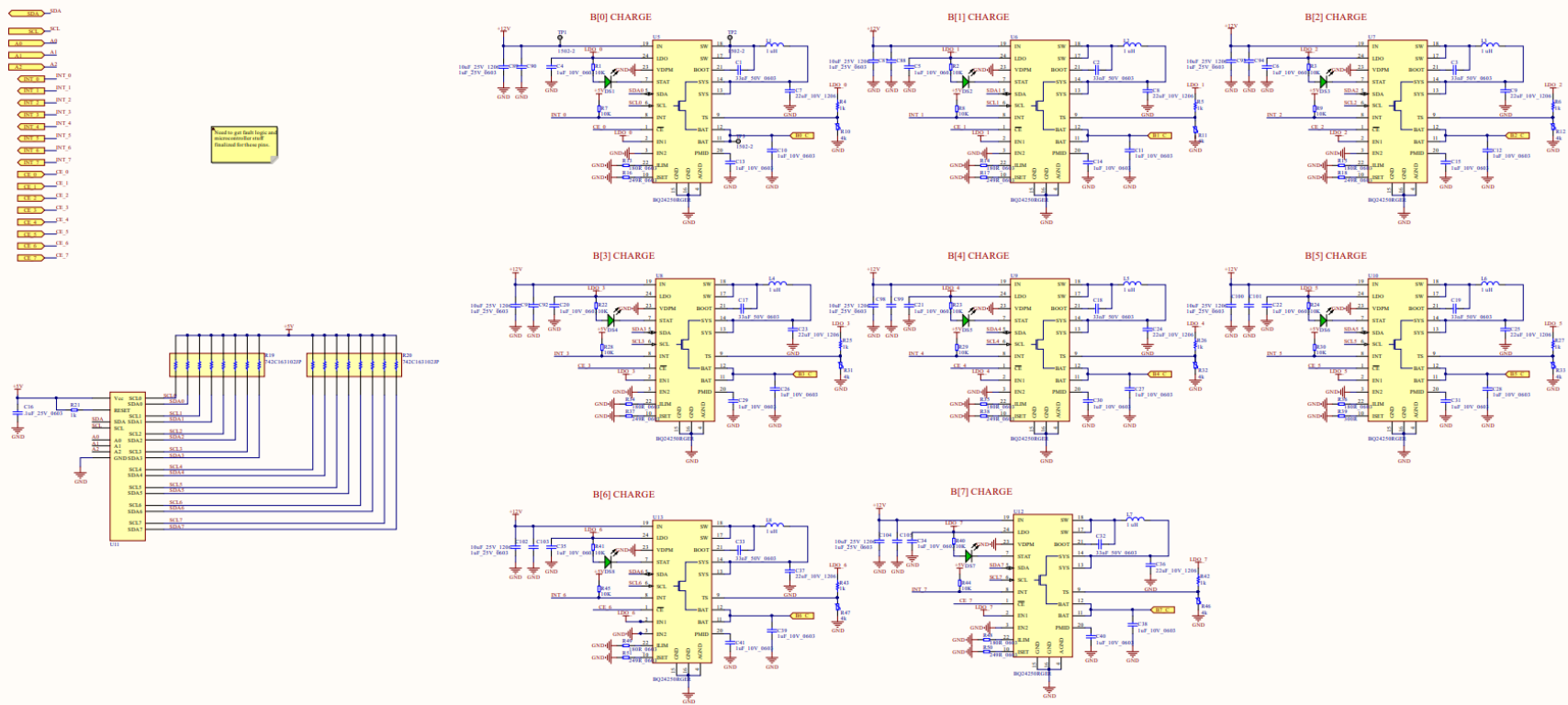


Figure 29: Charge Circuit Schematic



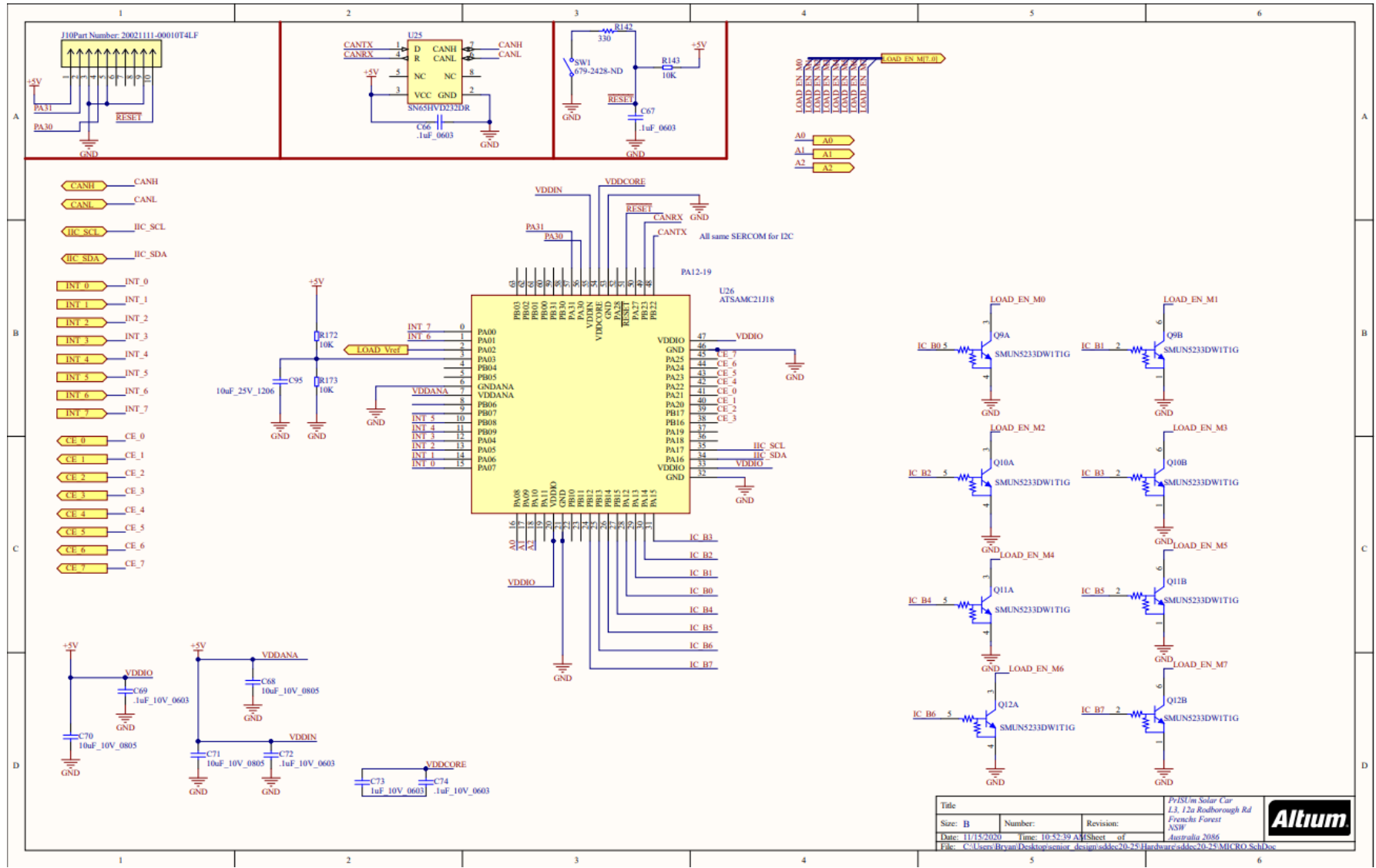


Figure 30: Microcontroller Schematic

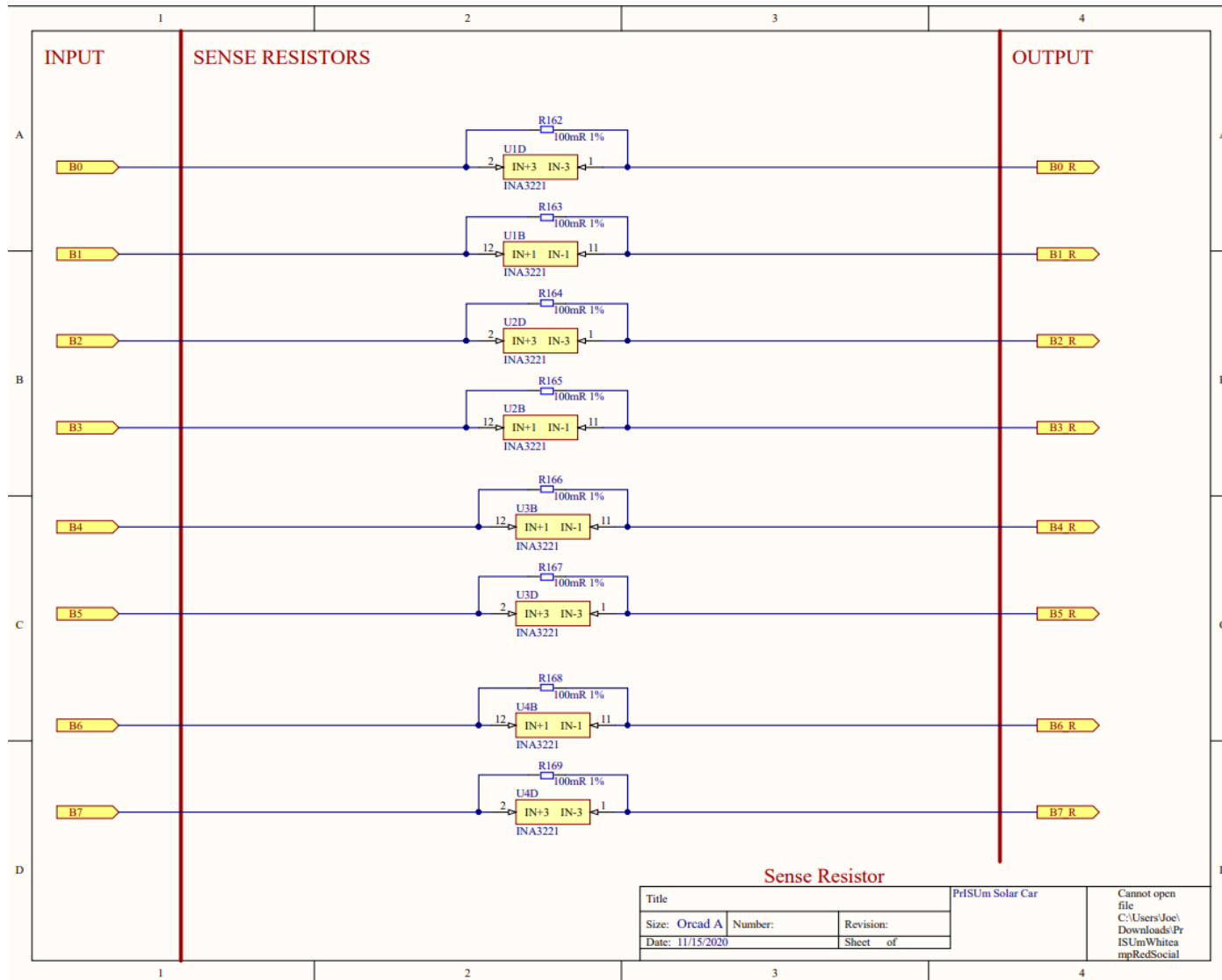


Figure 31: VI Sense Measurement IC Schematic

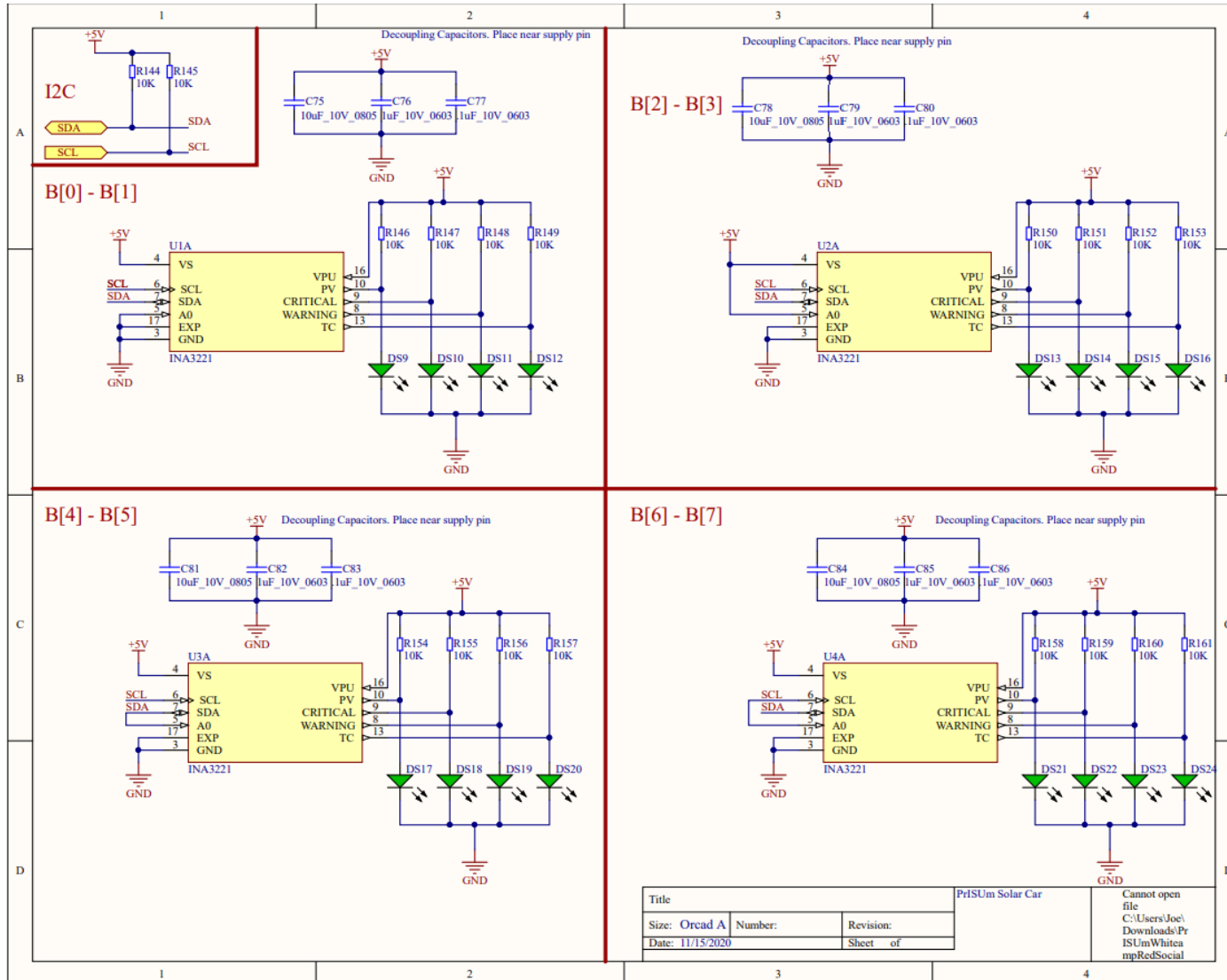


Figure 32: IC VI Measurement Schematic

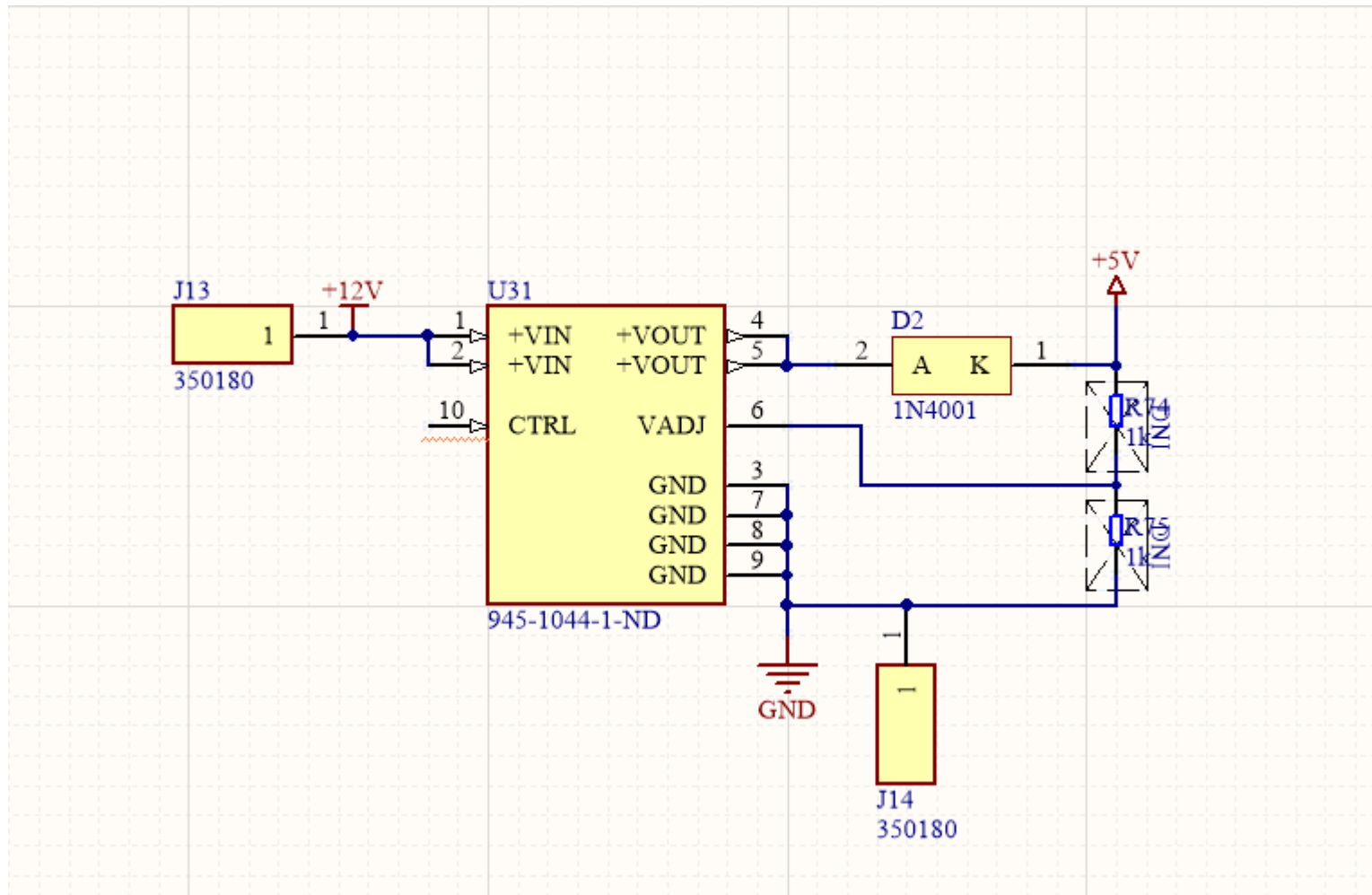


Figure 33: Power Circuit Schematic

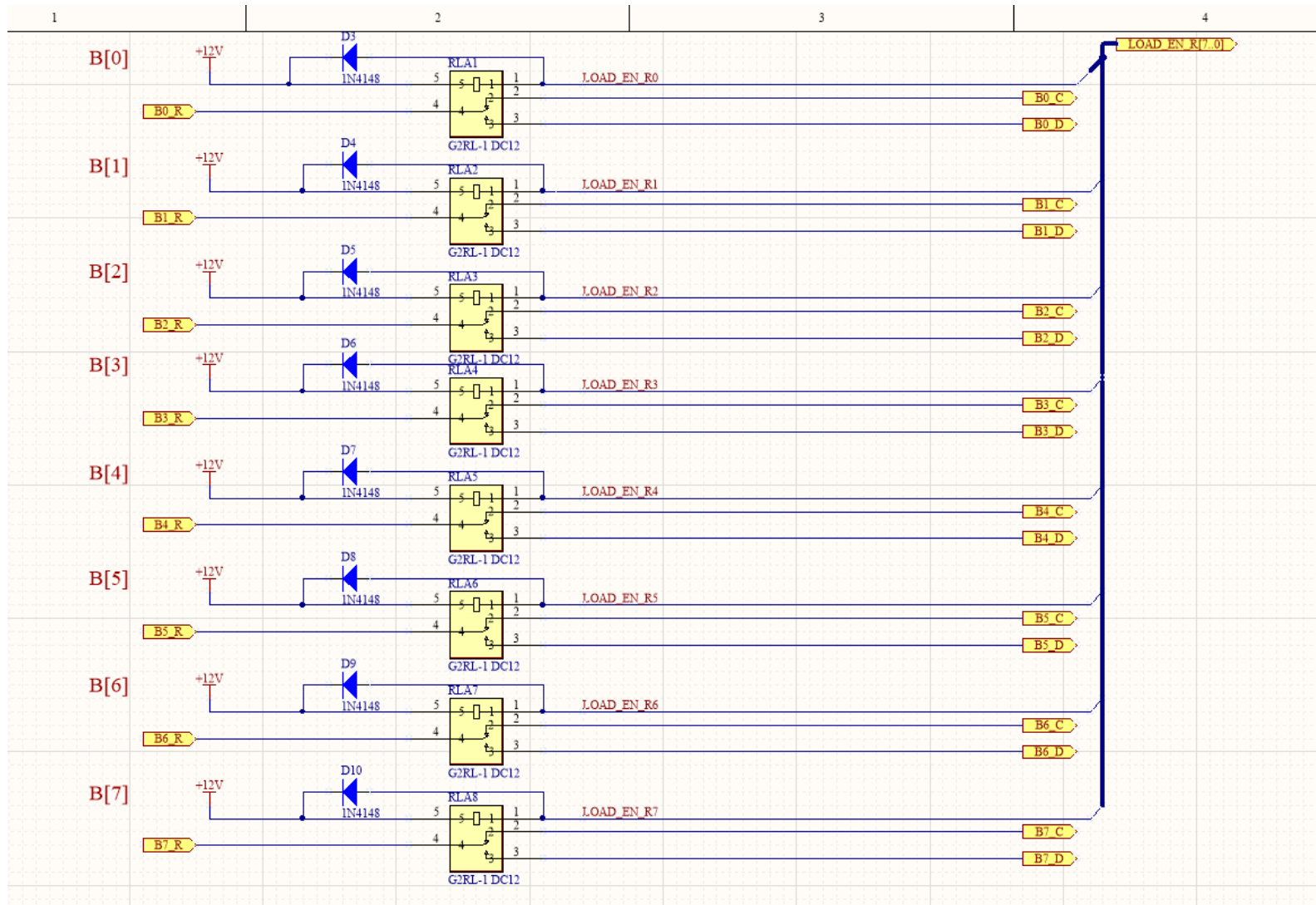


Figure 34: Relay Circuit Schematic

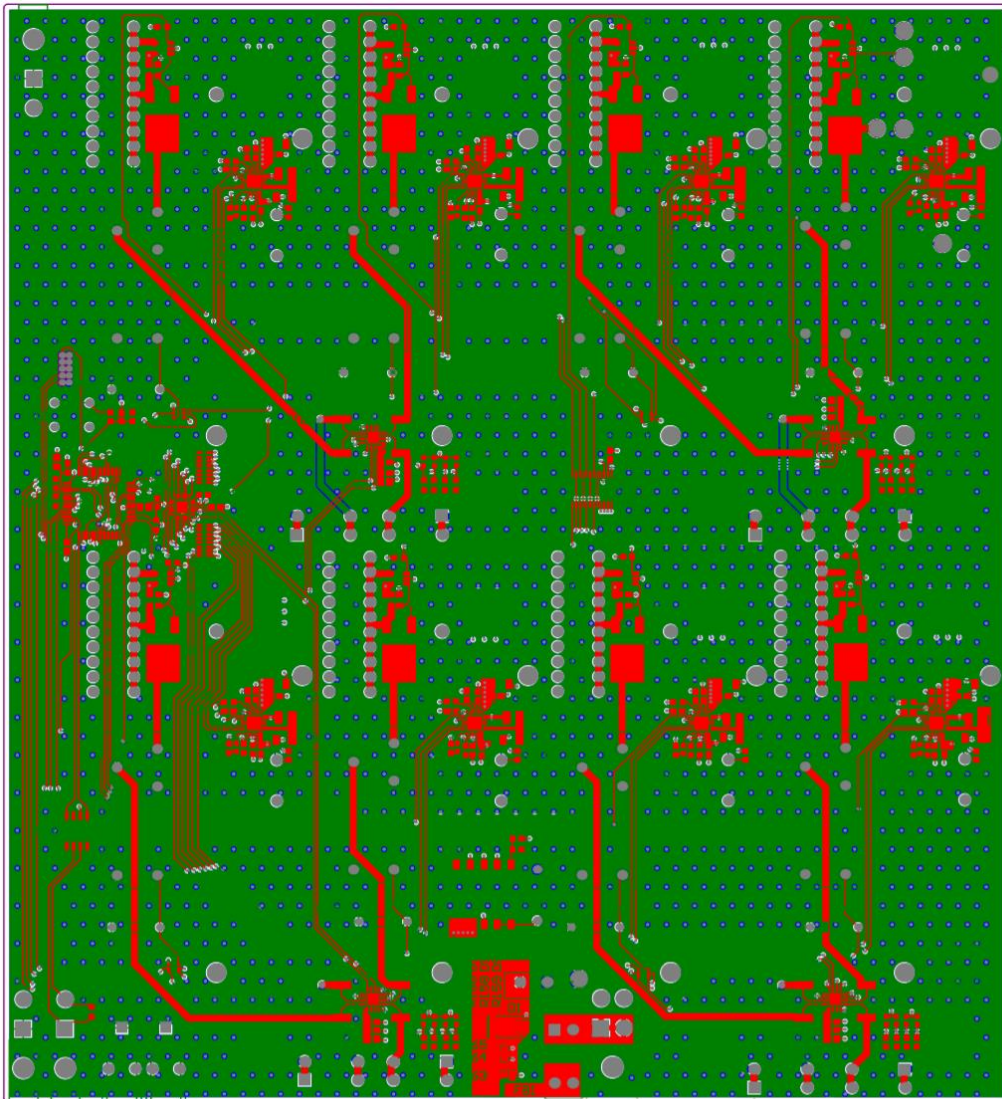


Figure 35: Main Board PCB Layout

## Appendix B: Constant Load Circuit Eval Schematic and PCB

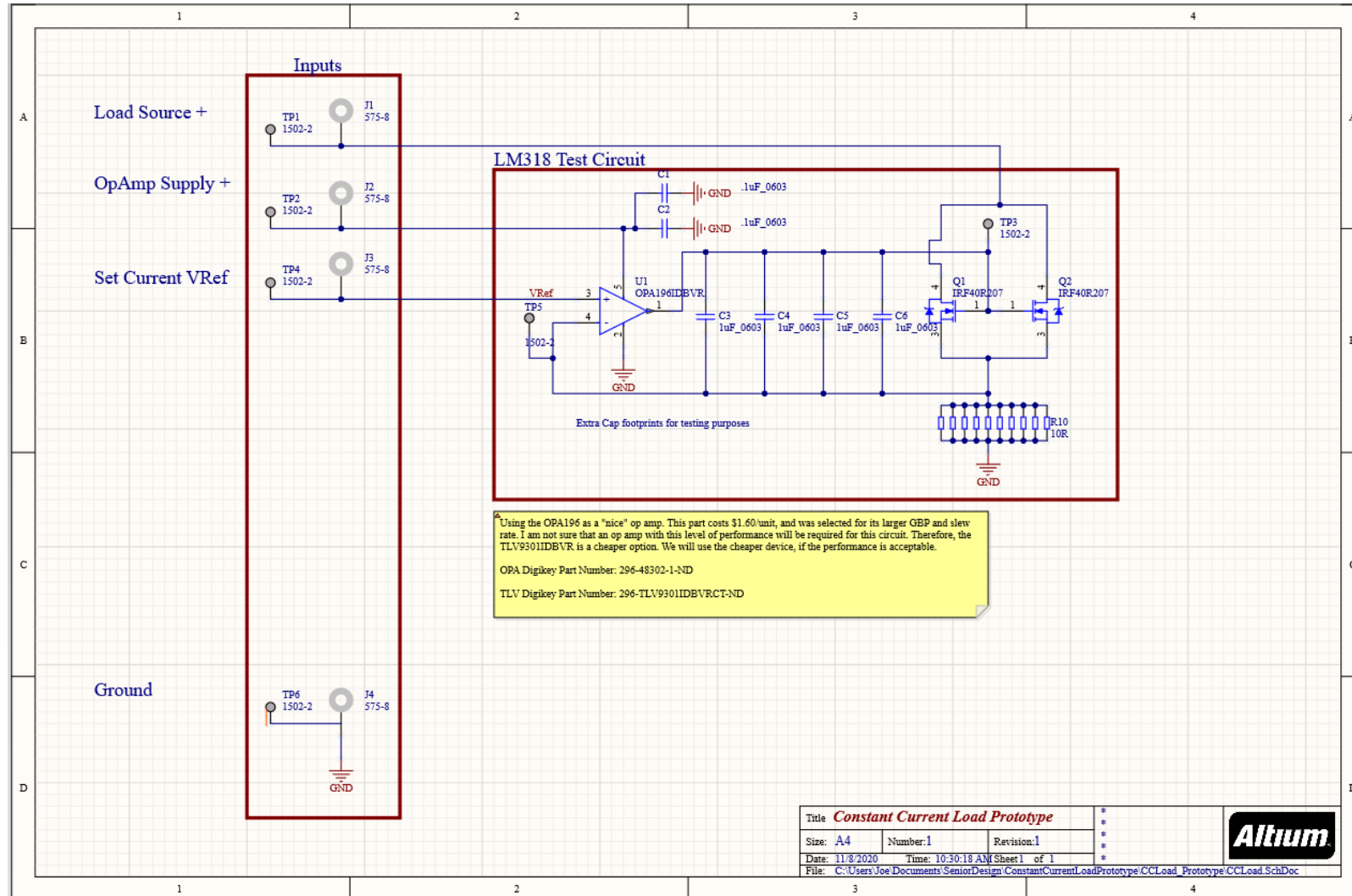


Figure 36: Constant Load Circuit Eval Board Schematic

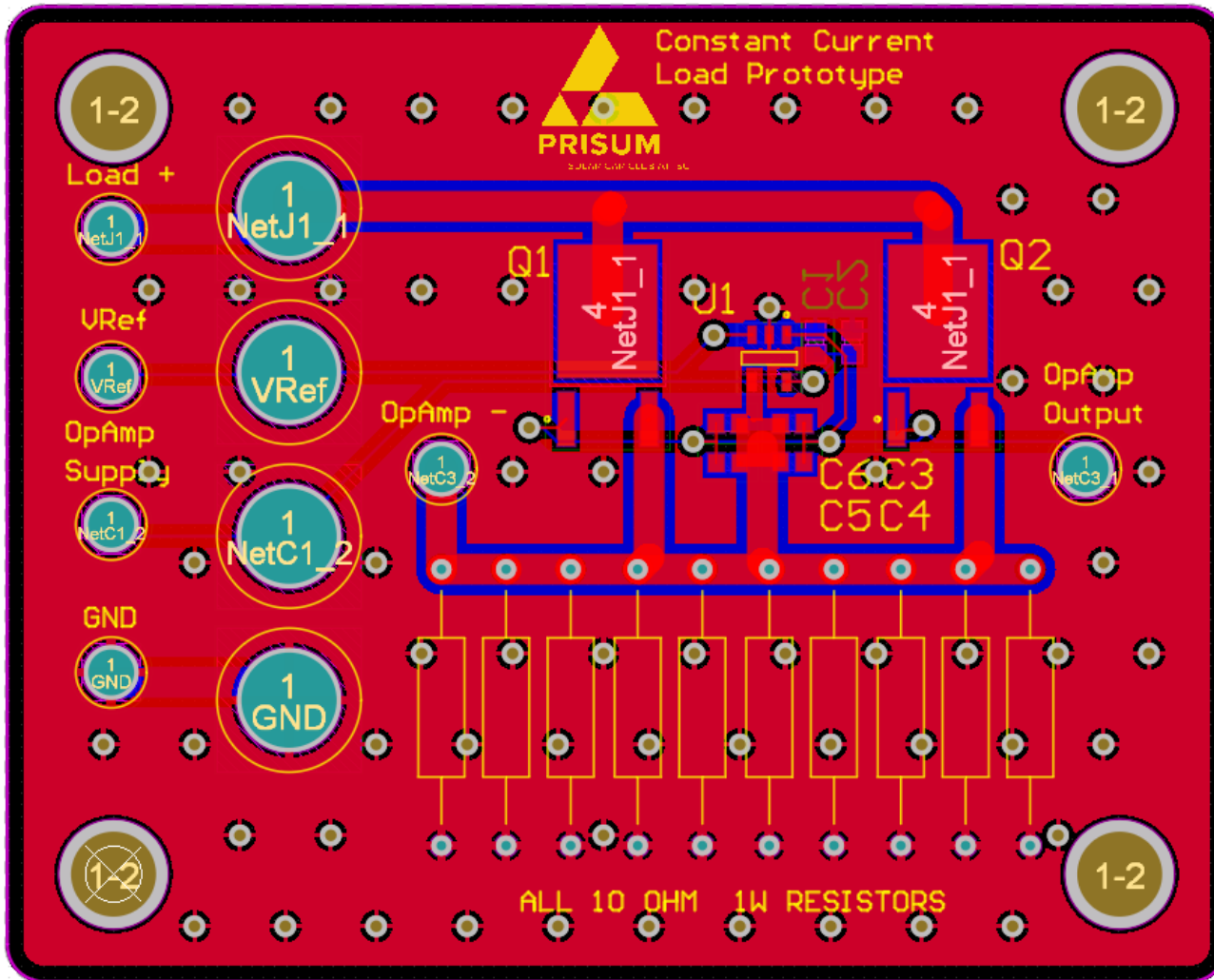


Figure 37: Constant Current Load Eval PCB layout



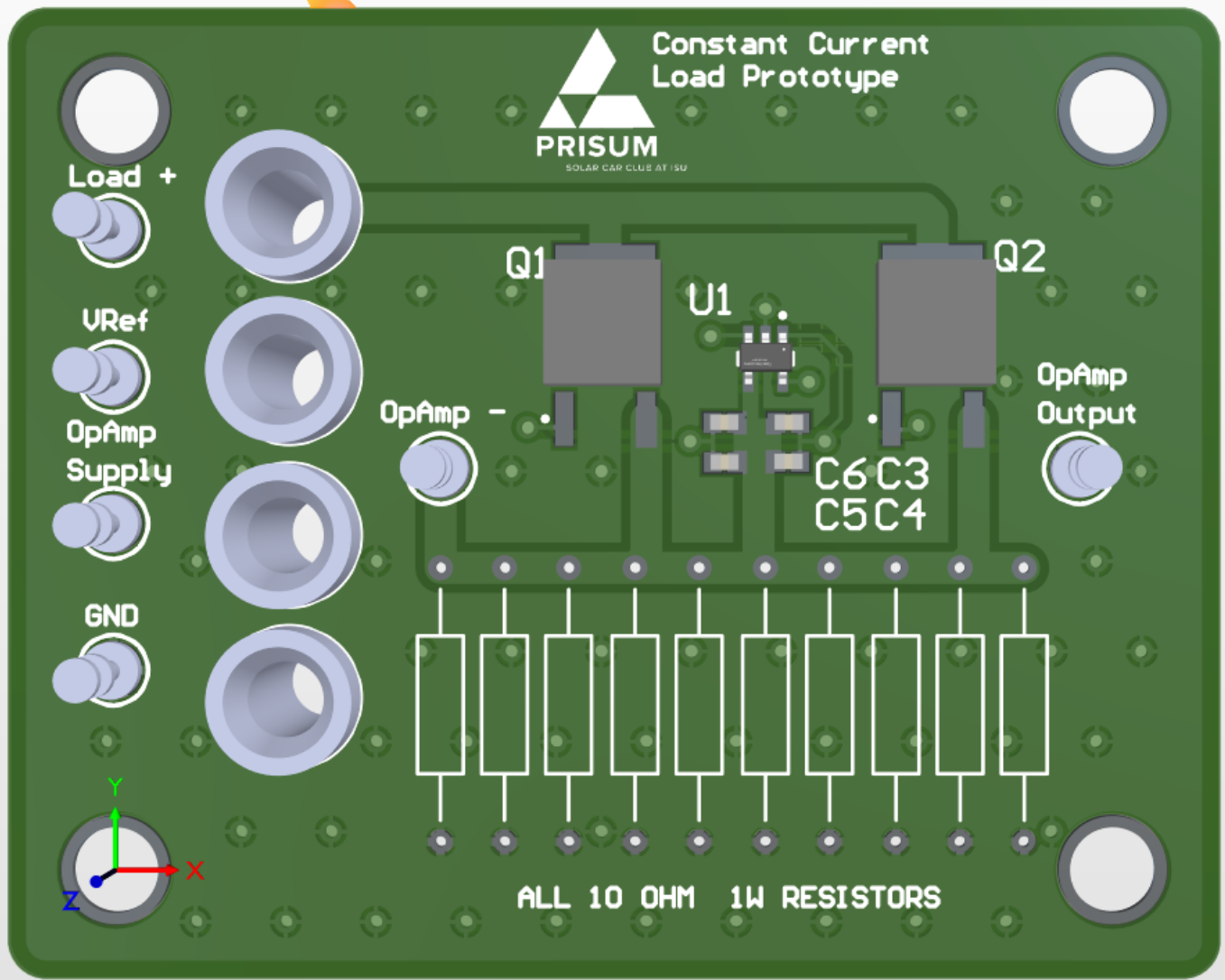


Figure 38: Constant Current Load Eval Board Physical PCB

## Appendix C: Microcontroller Eval Schematic and PCB

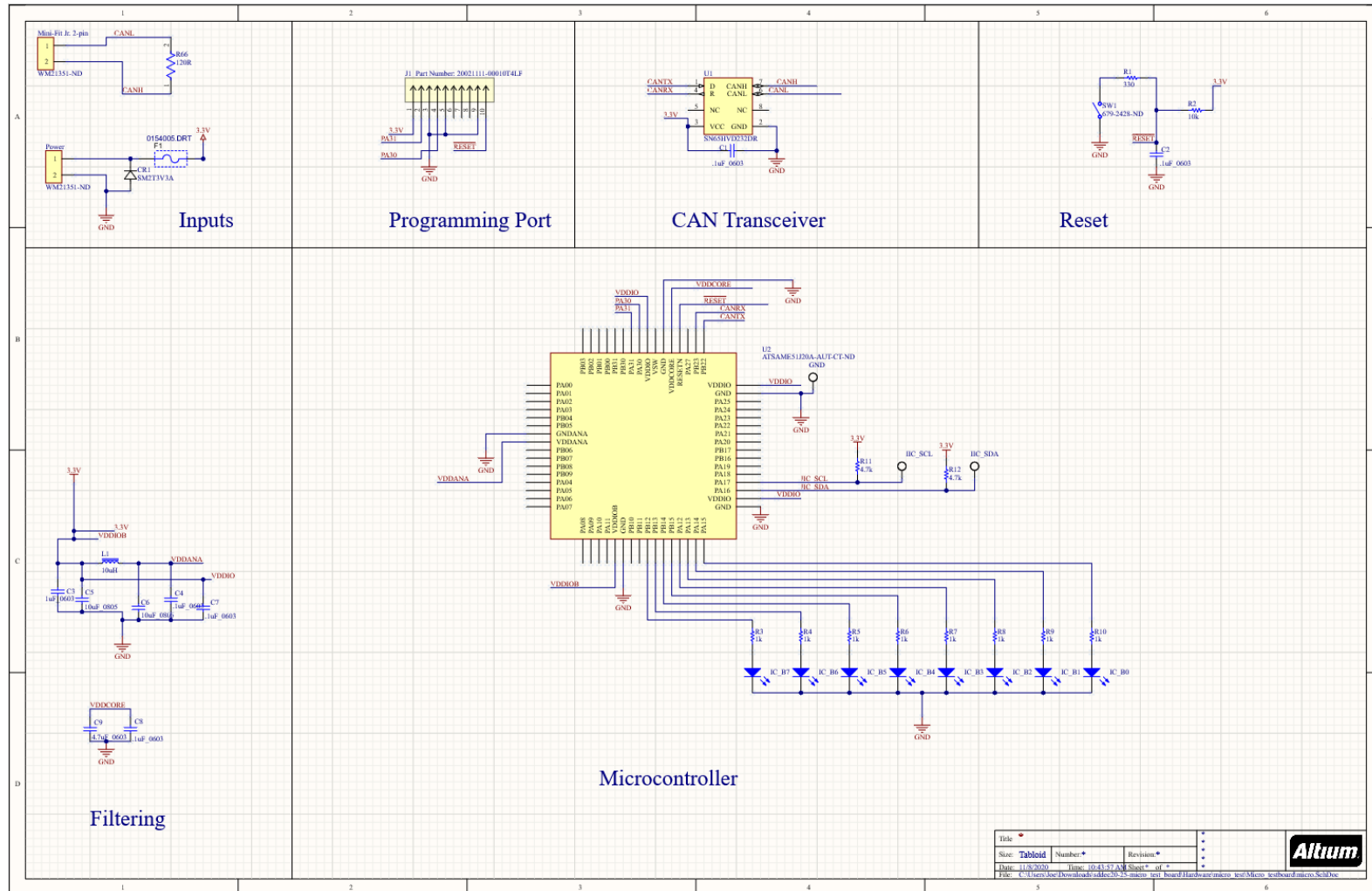


Figure 39: Microcontroller Eval Board Schematic

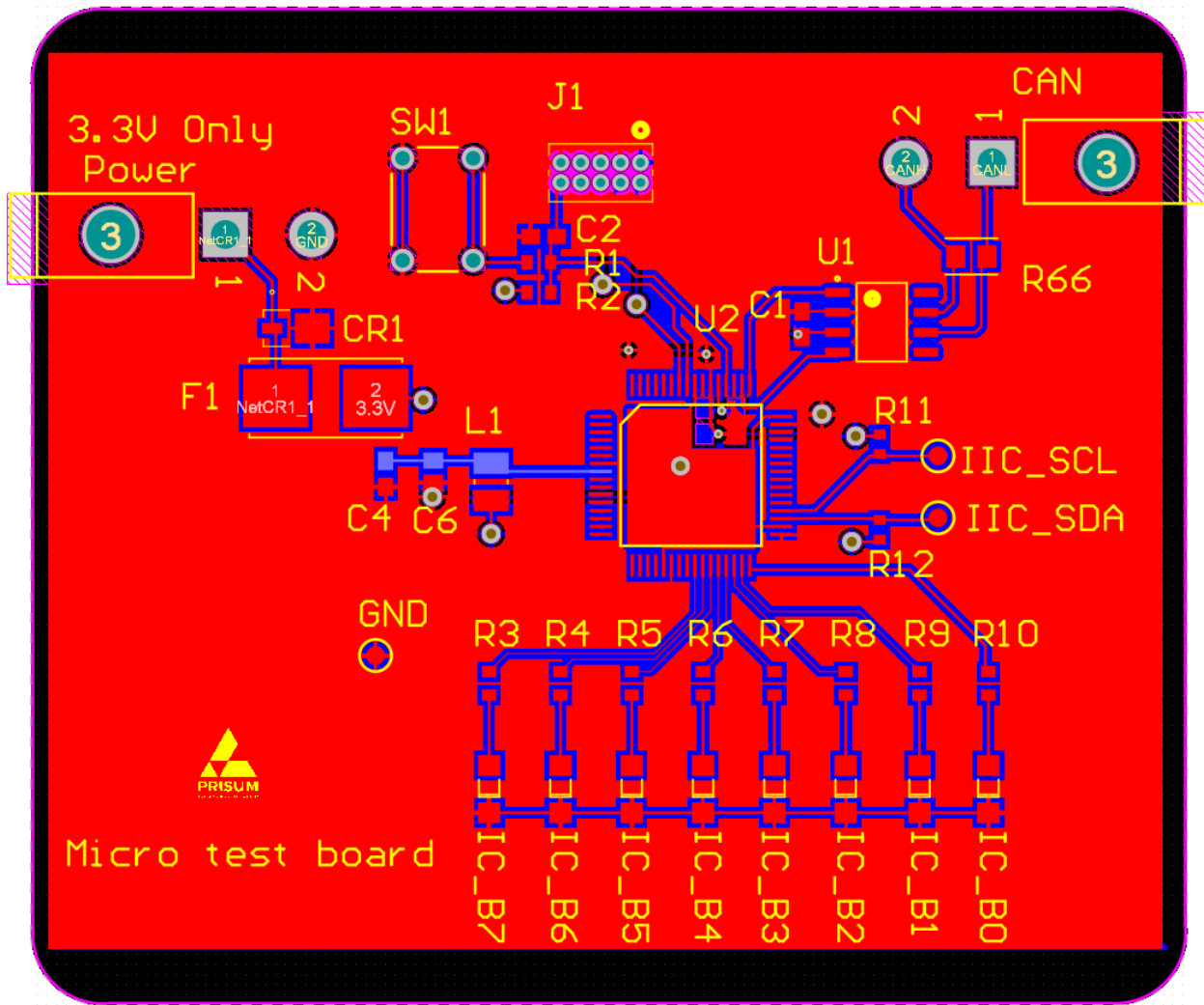


Figure 40: Microcontroller Eval Board PCB Layout

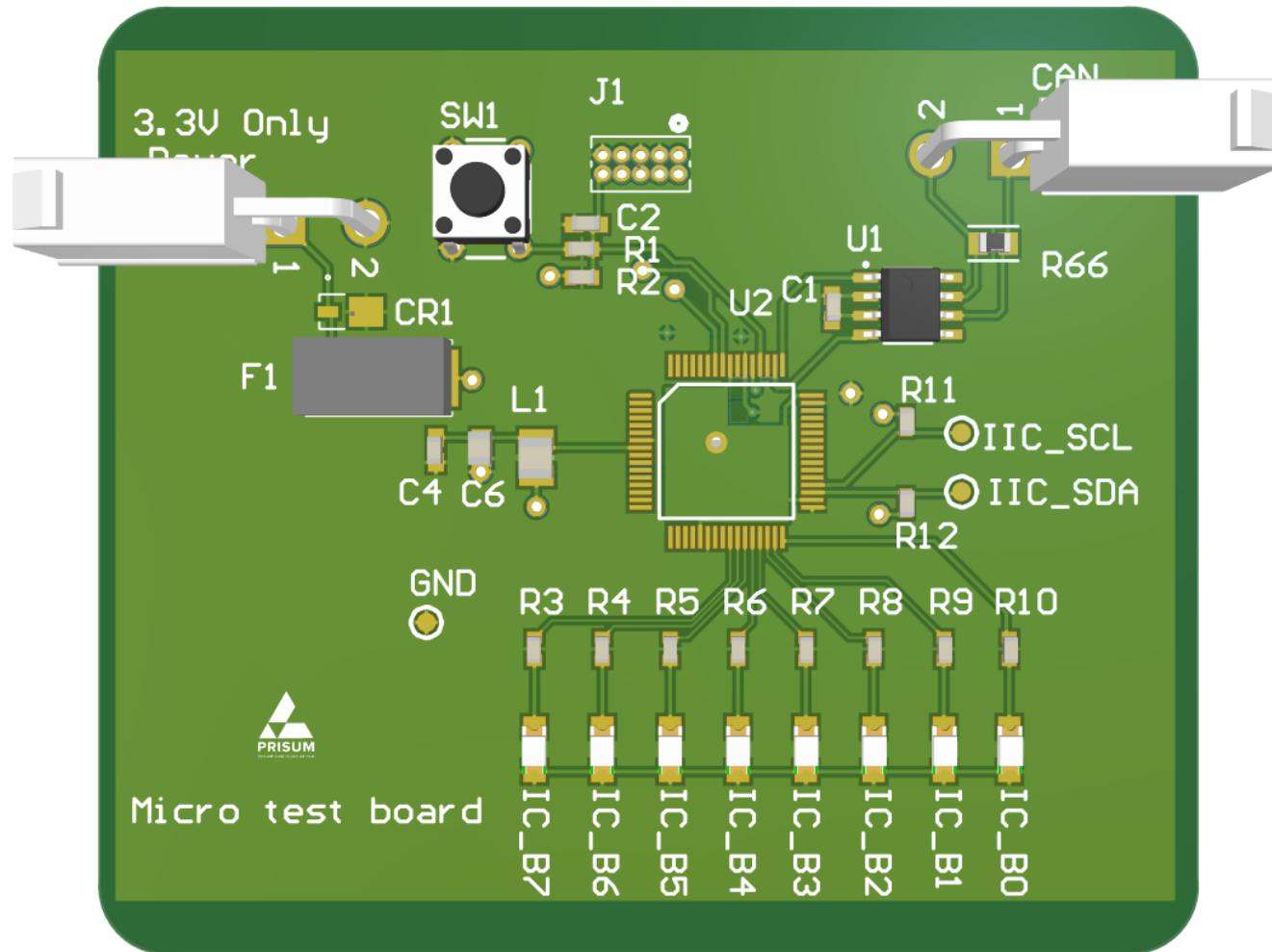


Figure 41: Microcontroller Physical PCB

## Appendix D: Bill of Materials for Main Board

Table 6: Bill of Materials for Main Board

Comment	Description	Designator	Quantity	Part Number	Cost (total)
<b>33nF_50V_0603</b>		C1-3,17-19, 32,33	8	CGA3E2X5R1H333M080AA	\$.80
<b>1uF_10V_0603</b>		C4-35, 38-41, 73, 76, 79,82, 85	29	CL10A105KP8NNNC	\$2.90
<b>22uF_10V_1206</b>		C7-9, 23-25, 36, 37	8	CC1206MKX5R6BB226	\$3.04
<b>.1uF_25V_0603</b>		C16, 42,43,46, 49, 50, 56, 59, 60, 65	10	CL10A104KA8NNNC	\$1.00
<b>1uF_25V_0603</b>		C44,45, 51, 54, 55, 58, 61, 62, 64, 88, 90, 92, 94, 99, 101, 103, 105	17	CC0603KRX5R8BB105	\$1.87
<b>.1uF_10V_0603</b>		C57, 69, 72, 74, 77, 80, 83,86	8	CC0603KRX7R6BB104	\$.88
<b>10uF_25V_1206</b>		C63, 87, 89, 91, 93, 95, 98, 100, 102, 104	10	C1206C106M3PACTU	\$2.70
<b>.1uF_0603</b>		C66-67	2	06035C104K4T4A	\$.20
<b>10uF_10V_0805</b>		C68, 70,71, 75, 78, 81, 84	7	C0805C106K8PACTU	\$.98
<b>STPS20L15D</b>	DIODE SCHOTTKY 15V 20A TO220AC	CR1	1	STPS20L15D	\$1.70
<b>SMBJ12A</b>	TVS DIODE 12V 19.9V DO214AA	D1	1	SMBJ12A	\$.39
<b>1N4001</b>	DIODE GEN PURP 50V 1A DO41	D2	1	1N4001-E3/53	\$.40
<b>1N4148</b>	High-speed Diode, 100 V, 450 mA, 2-Pin SOD27, RoHS	D3-10	8	1N4148	\$.80
<b>HSMG-C190</b>	Green LED	DS1- 24	24	HSMG-C190	\$12.24
<b>Fuse</b>	8A	F1	1	029707.5H	\$.34
<b>Fuse</b>	4A	F2-9	8	0297004.L	\$3.20
<b>Fuse Holder</b>	FUSE HOLDER	F1-9	9	3568	10.44

	BLADE				
<b>FerriteBead_12A</b>		FB1	1	BLM31SN500SN1L	\$ .32
<b>VES150_DC_CONN</b>	CONN JACK FEMALE 4POS TIN SOLDER	J1	1	KPJX-4S-S	\$2.23
<b>BatteryConnector</b>		J2-9	8	1042P	\$31.44
<b>10pin_connector</b>	CONN HEADER VERT 10POS 1.27MM	J10	1	20021111-00010T4LF	\$.69
<b>FAN_Connector</b>	Mini-Fit Jr. 2- pin Right Angle	J11- 12,CAN	3	0039300020	\$2.19
<b>Power_Connector</b>	Connector TeStraight Probe RCP 3 Position Solder Right Angle Thru-Hole 1 Port	J13-14	2	350180	\$12.70
<b>Inductor</b>	FIXED IND 1UH 1.6A 69 MOHM SMD	L1-8	8	LQM2HPN1R0MG0L	\$2.48
<b>IRF40R207</b>	MOSFET N-CH 40V 56A DPAK	Q1-8	8	IRF40R207	\$6.16
<b>SMUN5233DW1T1G</b>	TRANS 2NPN PREBIAS 0.187W SOT363	Q9-12	4	SMUN5233DW1T1G	\$1.72
<b>Res</b>	10k resistor	R1-9,22- 24,28- 30,40- 45,52- 53,76,87- 88,109,120- 121,143- 161,172,173	45	RC0603FR-0710KL	\$4.50
<b>Res</b>	1k resistor	R4-6,21,25- 27,42- 43,74,75	11	RC0603FR-071KL	\$1.10
<b>Res</b>	THERMISTOR NTC 4KOHM 3553K DO35	R10-12,31- 33,46,47	8	AL03006-2463-76-G1	\$21.28
<b>180R_0603</b>		R13-15,34- 36,48,49	8	RC0603FR-07180RL	\$.80
<b>249R_0603</b>		R16-18,37- 38,50,51	7	RC0603FR-07249RL	\$.70
<b>742C163102JP</b>	RES ARRAY 8 RES 1K OHM 2506	R19-20	2	742C163102JP	\$1.06
<b>300R</b>		R39	1	RC0603FR-07300RL	\$.10
<b>10R</b>		R54-141	80	KNP100FR-73-10R	\$40.80
<b>330R</b>	ERJ3EKF3300V	R142	1	ERJ-3EKF3300V	\$.10
<b>100mR 1%</b>	2512 package precision current	R162-169	8	WSL2512R1000FEA	\$8.40

	shunt resistor.				
<b>120R_0603</b>		R266	1	RC0603FR-07120RL	\$ .10
<b>G2RL-1 DC12</b>	RELAY GEN PURPOSE SPDT 12A 12V	RLA1-8	8	G2RL-1 DC12	\$21.52
<b>Reset Button</b>		SW1	1	MJTP1230	\$ .10
<b>1502-2</b>	Terminal DBL Turret, Through Hole, RoHS	TP1-5,8	6	1502-2	\$2.16
<b>INA3221</b>		U1-4	4	INA3221AIRGVR	\$15.92
<b>BQ24250RGER</b>		U5-13	8	BQ24250RGER	\$23.36
<b>I2C Multiplexer</b>		U11	1	TCA9548ARGERQ1	\$1.62
<b>TLV9301</b>		U14- 15,17,19- 20,22-24	8	TLV9301IDBVR	\$5.28
<b>ADG715BRUZ- REEL7</b>	IC SWITCH OCTAL SPST 24TSSOP	U21	1	ADG715BRUZ-REEL7	\$6.12
<b>SN65HVD232DR</b>	CAN transceiver	U25	1	SN65HVD232DR	\$1.95
<b>ATSAMC21J18</b>	IC MCU 32BIT 256KB FLASH 64TQFP	U26	1	ATSAMC21J18A-AUT	\$3.45
<b>Switching Regulator</b>	1A DC/DC- Converter	U31	1	R-78AA5.0-1.0SMD-R	\$8.66
<b>Total Cost</b>					\$272.90

## Appendix E: Bill of Materials for Constant Load Circuit Eval Board

Table 7: Bill of Materials for Constant Load Circuit Eval Board

Comment	Description	Designator	Quantity	Part Number
.1uF_0603	06035C104K4T4A	C1-2	2	06035C104K4T4A
1uF_0603		C3-6	4	CL10A105KP8NNNC
575-8	Non-Insulated Jack, Through Hole, RoHS, Bulk	J1-4	4	575-8
IRF40R207	MOSFET N-CH 40V 56A DPAK	Q1-2	2	IRF40R207
10R		R1-6	10	KNP100FR-73-10R
1502-2	Terminal DBL Turret, Through Hole, RoHS	TP1-6	6	1502-2
OPA196IDBVR	LOW-POWER 36-V PRECISION CMOS OP	U1	1	OPA196IDBVR
TLV9301IDBVR	Alternative for OPA196IDBVR	U1A	1	TLV9301IDBVR



## Appendix F: Bill of Materials for Microcontroller Test Eval Board

Table 8: Bill of Materials for Microcontroller Eval Board

Comment	Description	Designator	Quantity	Part Number
ATSAMC21J18	IC MCU 32BIT 256KB FLASH 64TQFP	U2	1	ATSAMC21J18A-AUT
Reset Switch	Tactile Switch SPST-NO Top Actuated Through Hole	SW1	1	MJTP1230
2-Pin connector	CONN HEADER R/A 2POS	CAN, POWER	2	0039300020
Programming Port	Connector Header Through Hole 10 position 0.050" (1.27mm)	J1	1	20021111-00010T4LF
Resistor 120Ω	CAN terminating resistor	R66	1	CRG0805F120R
Resistor 1kΩ		R3-10	8	RC0603FR-071KL
CAN Transceiver	IC TRANSCEIVER HALF 1/1 8SOIC	U1	1	SN65HVD232DR
Inductor 10μH		L1	1	1239AS-H-100M=P2
Fuse Holder	5A 125V AC 125V DC Fuse Board Mount (Cartridge Style Excluded) Holder, Surface Mount 2-SMD, Square End Block with Holder	F1	1	0154005.DRT
Fuse	FUSE BOARD MNT 500MA 125VAC/VDC	F1	1	SST 500
LED Orange		IC_B0-7	8	150120AS75000
TVS Diode	6.8V Clamp 30A Ipp Tvs Diode Surface Mount DO- 216AA	CR1	1	SM2T3V3A
Capacitor 4.7μF		C8-9	2	C0603C475K9PACTU
Capacitor 10μF		C5-6	2	C0805C106K8PACTU
Capacitor .1μF		C1-2,4,7	4	06035C104K4T4A
Capacitor 1μF		C3	1	GRT188C8YA105KE13D
Resistor 4.7kΩ		R11-12	2	ERJ-3EKF4701V
Resistor 330Ω		R1	1	ERJ-3EKF3300V

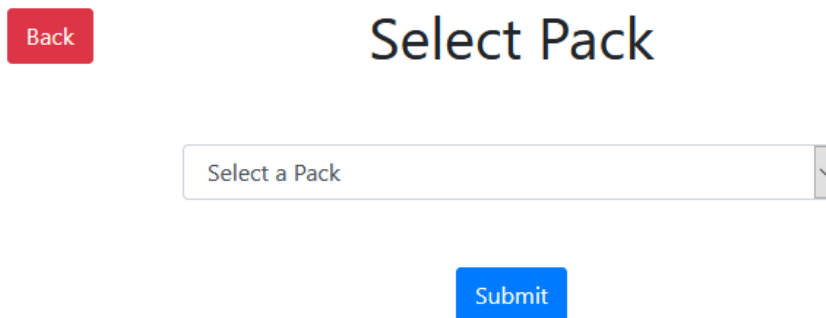
## Appendix G: User Manual

1. Connect the CAN cable between the raspberry pi and the testing module
2. Connect the power cable to the test module
3. Connect the power cable to the raspberry pi
4. As the system boots up, you should see the bootup screen on the raspberry pi's display, as shown in Figure 42
5. Pushing the "Set Up Test" button takes you to the pack selection screen, shown in Figure 43. This screen allows you to select which battery pack to save the battery characteristic data into. You can select from existing battery packs, or you can receive a prompt for how to use the desktop interface for creating a new battery pack (described below).

# Automated Li-Ion Battery Characterizer

A blue rectangular button with the text "Set Up Test" in white.

Figure 42: User Interface the Pi Boot onto

The "Select Pack" screen features a red "Back" button on the left, a large "Select Pack" title in the center, a dropdown menu with "Select a Pack" and a downward arrow, and a blue "Submit" button at the bottom.

Back

## Select Pack

Select a Pack

Submit

Figure 43: User Interface for selecting the active battery pack

6. Once you have selected the battery pack and hit “Submit”, you are taken to the battery labeling page, which is shown in Figure 44. Once all of your batteries are connected to the test module, the raspberry pi will provide unique labels for each of the batteries connected to the device. You should label the batteries with a permanent marker, a printed label, or other marking device.
7. After all batteries under test are labeled, press “Begin Test” to start the test. This will take you to the test in progress page, as shown in Figure 45
8. The test in progress page will display voltage and current information for each battery, and an average temperature of the system. The test does not require any further input during the test.
9. If desired, the test can be aborted at any time by pressing the “Stop Test” button. If the test is aborted, the battery labels and the test data is discarded.
10. If the test module aborts the test before completion, the raspberry pi will attempt to display what went wrong with a display similar to Figure 46. If the test is aborted, the battery labels and the test data is discarded.

## Connected Battery Labels

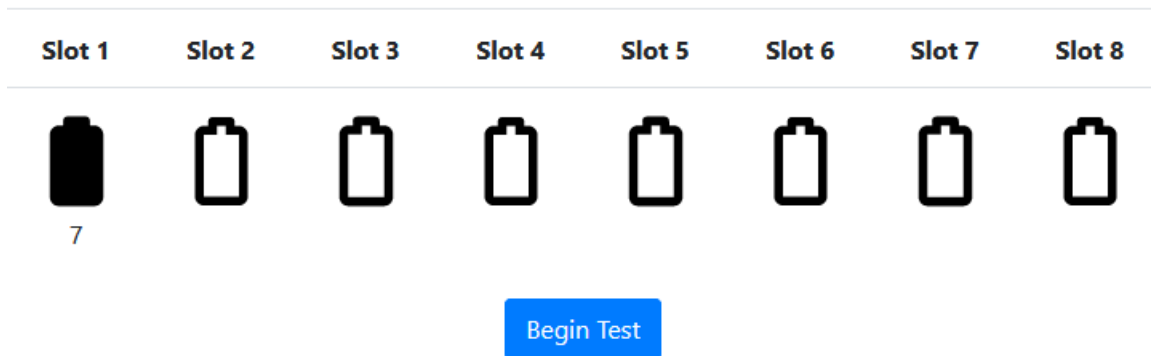


Figure 44: User Interface defining what to label each cell

# Test in Progress...

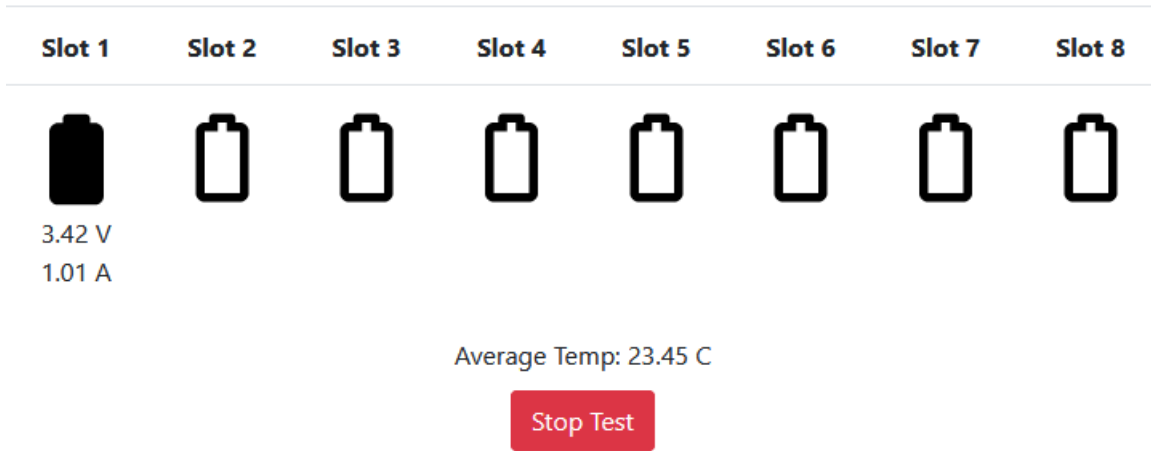


Figure 45: User interface for test in progress page

Failed!  
Failure reason: Temperature Fault

Finish

Figure 46: User Interface for failed test

11. If the test completes successfully, all data is saved to the database and the raspberry pi will display a success screen, as shown in Figure 47.

Success!  
Data is available on the website!

Finish

Figure 47: User interface for successful test

12. Pressing the “Finish” button on either the failure or success screen will take you back to the bootup screen, where you can start a new test.

#### APPENDIX G.A: CONNECTING TO THE WEBSITE

1. Connect the CAN cable between the raspberry pi and the testing module
2. Connect the power cable to the test module
3. Connect the power cable to the raspberry pi
4. Navigate to the raspberry pi’s IP address in your web browser. You should see the display in Figure 48.

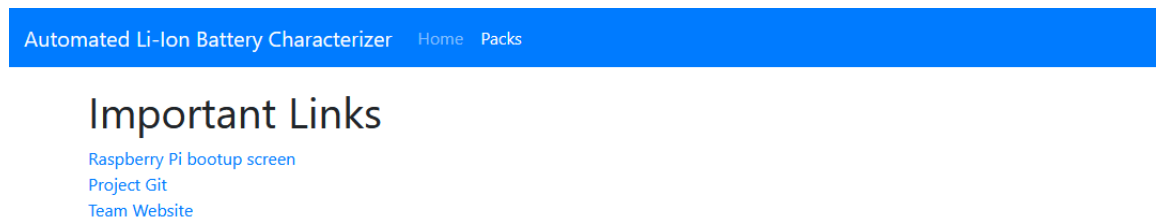


Figure 48: User interface for desktop home screen

#### APPENDIX G.B: VIEW TEST DATA

1. Connect to the website by following the steps in Appendix G.A: Connecting to the website
2. At the top of the page, click the “Packs” button to go to the page listing the created packs, shown in Figure 48.

- a. If your display is fairly narrow, the “Packs” button will be within the hamburger menu in the top right corner of the page, similar to the one in Figure 49.
3. Click the name of the battery pack you would like to see data for, this will take you to the battery pack details page, which is shown in Figure 50.
4. Click the label of a battery that you would like to see test data for, which will take you to the battery cell details page, which is shown in Figure 51.
5. If desired, you can press the “Download Data” button to download as Excel Spreadsheet with all of the data for that battery in it.

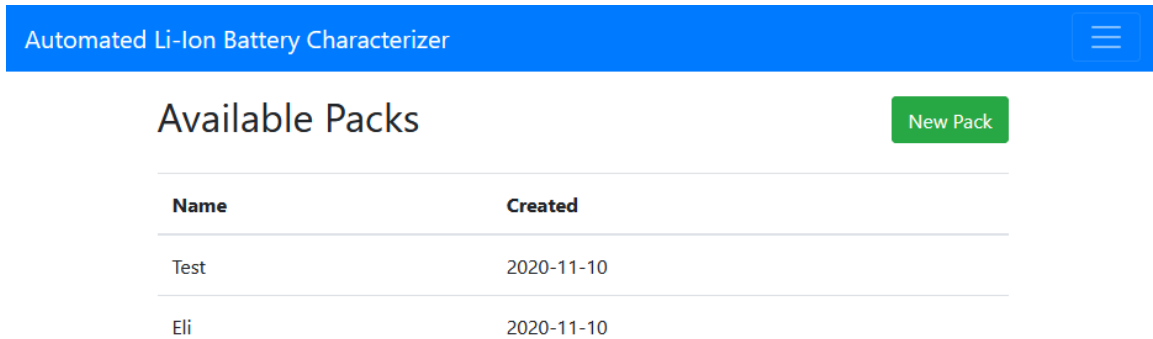


Figure 49: User interface for seeing all battery packs that have been created



## Pack: Test

35 Modules

20 Batteries per Module

---

<b>Batteries</b>
1
2
3
4
5

*Figure 50: User interface for seeing information about a battery pack*

## Test: Battery #1

[Download Data](#)

Sample Number	Voltage (V)	Current (A)	Temp (C)
1	3.0	0.0	0.0
2	3.0	0.0	0.0
3	3.0	0.0	0.0
4	3.0	0.0	0.0
5	3.0	0.0	0.0
6	3.0	0.0	0.0
7	3.0	0.0	0.0
8	3.0	1.0	23.4
9	3.0	1.0	23.4
10	3.0	1.0	23.4
11	3.0	1.0	23.4
12	3.0	1.0	23.4
13	3.0	1.0	23.4
14	3.0	1.0	23.4

Figure 51: User interface for seeing battery test data (Using fake data)

#### APPENDIX G.C: CREATE NEW BATTERY PACK

1. Connect to the website by following the steps in Appendix G.A: Connecting to the website
2. At the top of the page, click the “Packs” button to go to the page listing the created packs, shown in Figure 49.
  - a. If your display is fairly narrow, the “Packs” button will be within the hamburger menu in the top right corner of the page, similar to the one in Figure 49
3. Click the “New Pack” button, which will take you to the pack creation page, which is shown in Figure 52.
4. Enter the name of the battery pack, the number of modules to go in the battery pack, and the desired number of batteries in each module for a battery pack.
5. Click “Submit” to create the new battery pack, you will be sent back to the list of created battery packs.





## Create Pack

Pack Name

Number of Modules:

Number of Batteries (For each module):

*Figure 52: User interface for creating a new battery pack*