**Li-ion Battery Test Bench**

**Arduino Controlled and mobile App monitored Test Bench.**

******

***Submitted by:***

Hammad Khan 780618

Hamza Abbas 780615

Jawad Maqsud 780579

Mohamed Mahmoud 780611

Team Project 13576

Hochschule Esslingen

University

*Submission Date: 1st August 2025*

Supervisors:

***Prof. Dipl.-Ing Georg Mallebrein***

***Prof. Dipl.-Ing Mathias Oberhauser***

External Supervisors:

***M.Eng. Jochen Buck (Robert Bosch GmbH)***

Certificate of Authenticity

We hereby declare that we have written the attached work alone and without any other reference works than those mentioned. All thoughts and quotations taken directly or indirectly from other sources have been noted as such. Furthermore, we have not used this work, parts of this work or basic ideas from this work to achieve credits in any academic course at any time.

Esslingen am Neckar, 25/06/2025

*Hammad Khan*

*Hamza Abbas*

*Jawad Maqsud*

*Mohamed Mahmoud*

Acknowledgement

We would like to express our sincere gratitude to **Prof. M. Eng. Jochen Buck and Prof. M. Eng. GeorgMallebrein** for his invaluable guidance, continuous support, and technical insight throughout the duration of this project. His expertise in battery systems, embedded programming, and safety considerations greatly contributed to the successful development of our small battery test bench.

We would also like to thank **Mr. Schmidt** at Hochschule Esslingen for his assistance during the planning and construction phases, especially in the procurement of necessary components.

Our appreciation also extends to all team members for their collaborative spirit and dedication, as well as to the creators of open-source tools like **Arduino** and **MIT App Inventor**, which played a critical role in the implementation of the control interface.

Lastly, we acknowledge the earlier work of previous student groups, whose concepts and documentation laid a foundation for our improvements and adaptations.

*Hammad Khan*

*Hamza Abbas*

*Jawad Maqsud*

*Mohamed Kame*

Table of content:

Contents

[Introduction 1](#_Toc205156967)

[1.1 Background and Motivation 1](#_Toc205156968)

[1.2 Purpose of Test Bench 1](#_Toc205156969)

[1.3 Structure of the Report 2](#_Toc205156970)

[2 System Requirements 4](#_Toc205156971)

[2.1 Overview of functional and safety requirements 4](#_Toc205156972)

[2.2 Functional Design/ Solution / Schematic: 6](#_Toc205156973)

[3 Hardware Architecture 8](#_Toc205156974)

[3.1 Description main components: 8](#_Toc205156975)

[3.1.1 Battery: 8](#_Toc205156976)

[3.1.2 Arduino Nano (ESP32): 9](#_Toc205156977)

[3.1.3 TIP 145 Darlington: 9](#_Toc205156978)

[3.1.4 TIP 140 Darlington: 10](#_Toc205156979)

[3.1.5 BC-337 Transistor: 11](#_Toc205156980)

[3.1.6 ADS1115 ADC: 11](#_Toc205156981)

[3.1.7 Power resistor: 12](#_Toc205156982)

[3.1.8 Relays: 12](#_Toc205156983)

[3.1.9 Fuse: 13](#_Toc205156984)

[3.1.10 LEDs: 13](#_Toc205156985)

[3.1.11 DS18B20 Temperature Sensor: 14](#_Toc205156986)

[3.1.12 M74HC27B1 logic IC: 14](#_Toc205156987)

[3.2 Simulink Model and Simulation Results: 15](#_Toc205156988)

[4 SUB MODULES OF SIMULINK DIAGRAM: 15](#_Toc205156989)

[4.1 Charging Module: 16](#_Toc205156990)

[4.2 Discharging Module 16](#_Toc205156991)

[4.3 Battery (Load): 17](#_Toc205156992)

[4.4 External ADC: 17](#_Toc205156993)

[4.5 PI Controller with Limitations: 17](#_Toc205156994)

[4.5.1 Voltage Limitation: 18](#_Toc205156995)

[4.5.2 Low-Pass Filter (After Voltage Measurement): 18](#_Toc205156996)

[4.5.3 Current Limitation using Differential ADCs: 18](#_Toc205156997)

[4.5.4 PI Controller with Anti-Windup: 18](#_Toc205156998)

[5 Charging and Discharging Mode Experiments: 19](#_Toc205156999)

[5.1 PWM, Current, and ADC Voltages at 2A 20](#_Toc205157000)

[6 Circuit Diagram: 21](#_Toc205157001)

[6.1 PCB Layout: 23](#_Toc205157002)

[6.2 PCB 3D Rendered View: 23](#_Toc205157003)

[6.3 Creating a PCB process using blocks diagram 24](#_Toc205157004)

[***Detailed Explanation of Each Step*** 24](#_Toc205157005)

[**1. Create Schematic** 24](#_Toc205157006)

[**2. Assign Footprints** 24](#_Toc205157007)

[**3. Import Schematic to PCB Layout** 25](#_Toc205157008)

[**4. Define Board Outline and Place Components** 25](#_Toc205157009)

[**5. Route Manually or Export to FreeRouting** 25](#_Toc205157010)

[**6. Import Routed File Back to PCB Tool** 25](#_Toc205157011)

[**7. Check Design Rules** 26](#_Toc205157012)

[**8. Export Gerber Files** 26](#_Toc205157013)

[**9. Send to Manufacturer.** 26](#_Toc205157014)

[***Considerations regarding assembly:*** 26](#_Toc205157015)

[Leave their outputs unconnected. 27](#_Toc205157016)

[7 Software & Control Logic 28](#_Toc205157017)

[7.1 Charging Algorithm (CC/CV) 28](#_Toc205157018)

[7.2 Charging/Discharging flow chart: 29](#_Toc205157019)

[7.3 Use Cases 30](#_Toc205157020)

[7.4 Module Responsibility Map 33](#_Toc205157021)

[7.5 Module Dependency Diagram 34](#_Toc205157022)

[7.6 Run-Time Flow (sequence diagram) 35](#_Toc205157023)

[7.7 Finite-State Machine (operation logic) 36](#_Toc205157024)

[7.8 Key Software Assets 37](#_Toc205157025)

[7.9 Further Software Improvements 37](#_Toc205157026)

[8 Mobile Application 38](#_Toc205157027)

[9 Testing and Results: 45](#_Toc205157028)

[9.1 Pre-Testing Setup 45](#_Toc205157029)

[9.2 Testing Results and Curves for Charging Mode: 47](#_Toc205157030)

[9.3 Testing Results and Curves for Discharging Mode: 48](#_Toc205157031)

[10 Conclusion 50](#_Toc205157032)

[10.1 Project Deliverables 50](#_Toc205157033)

[Utility and Scientific Relevance 50](#_Toc205157034)

[Overall Assessment 50](#_Toc205157035)

[11 Future Scope and Improvements 51](#_Toc205157036)

[12 References 52](#_Toc205157037)

[13 Appendices 54](#_Toc205157038)

List of Tables

[Table 1 Requirements Table 5](#_Toc205156689)

[Table 2 Battery Specifications 8](#_Toc205156690)

[Table 3 Use Case battery charging 32](#_Toc205156691)

[Table 4 Use Case battery discharging 33](#_Toc205156692)

[Table 5 Module Responsibility Map 34](#_Toc205156693)

[Table 6 Key Software Assets 37](#_Toc205156694)

[Table 7 Mobile Application Manual 45](#_Toc205156695)

[Table 8 Li-ion battery used 54](#_Toc205156696)

List of Figures

[Figure 1: Functional Design/ solution/ Schemtic 6](#_Toc205156653)

[Figure 2: Functional architecture 7](#_Toc205156654)

[Figure 3: Screenshot from Arduino Nano ESP32 Cheat Sheet (Arduino, 2025 9](#_Toc205156655)

[Figure 4: TIP 145 Darlington Transistor 10](#_Toc205156656)

[Figure 5: TIP 140 Darlington Transistor 10](#_Toc205156657)

[Figure 6: BC-337 Transistor 11](#_Toc205156658)

[Figure 7: ADS1115 ADC 11](#_Toc205156659)

[Figure 8: Power resistor 12](#_Toc205156660)

[Figure 9: SPDT Electro-Mechanical relay 12](#_Toc205156661)

[Figure 10: Fuse 13](#_Toc205156662)

[Figure 11: LED 13](#_Toc205156663)

[Figure 12: Dallas DS18B20 Temperature sensor 14](#_Toc205156664)

[Figure 13: M74HC27B1 IC 14](#_Toc205156665)

[Figure 14: Test Bench Simulink/Sims cape model 15](#_Toc205156666)

[Figure 15: Charging Module 16](#_Toc205156667)

[Figure 16: Discharging Module 17](#_Toc205156668)

[Figure 17: External ADC 17](#_Toc205156669)

[Figure 18: PI Controller and Switch 19](#_Toc205156670)

[Figure 19: PI Controller Detailed 19](#_Toc205156671)

[Figure 20: ADC Voltages 20](#_Toc205156672)

[Figure 21: PWM Controller for Discharging 21](#_Toc205156673)

[Figure 22 Full Circuit Diagram 22](#_Toc205156674)

[Figure 23 PCB layout routed 23](#_Toc205156675)

[Figure 24 PCB 3D Rendered view 23](#_Toc205156676)

[Figure 25 PCB life cycle 24](#_Toc205156677)

[Figure 26 3 input NOR gate from the 74HCT27D 27](#_Toc205156678)

[Figure 27 Screenshot from How to Build a CC/CV Battery Charger (Power Electronics News,2025) 28](#_Toc205156679)

[Figure 28 Charging/Discharging flow chart 29](#_Toc205156680)

[Figure 29 Module Dependency Diagram 35](#_Toc205156681)

[Figure 30 illustrates the 5 ms control loop executed on the ESP32 once the user has selected a mode via the smartphone application. 35](#_Toc205156682)

[Figure 31 Finite-State Machine 36](#_Toc205156683)

[Figure 32: Battery charging curve 47](#_Toc205156684)

[Figure 33: Voltage vs Soc Figure 34: Current 47](#_Toc205156685)

[Figure 35: Battery discharging curve 48](#_Toc205156686)

[Figure 36: Voltage vs Soc for discharging Figure 37: Discharging current 49](#_Toc205156687)

[Figure 38 Screenshot from Arduino Nano ESP32 Cheat Sheet (Arduino, 2025) 55](#_Toc205156688)

List of Abbreviations:

|  |  |
| --- | --- |
| **SPI** | serial prepheral interface |
| **ADC** | Analog to digital converter |
| **CC** | constant current |
| **CV** | constant voltage |
| **DC** | direct current |
| **HVP** | high voltage protection |
| **I/O** | input/ output |
| **Li-Ion** | Lithium ion |
| **LVP** | low voltage protection |
| **NCA** | (Nickel Cobalt Aluminum Oxide) |
| **Ni-MH** | nickel-metal hydride |
| **PC** | personal computer |
| **PWM** | pulse width modulation |
| **SOC** | state of charge |
| **SOH** | state of health |
| **SRAM** | Static Random Access Memory |
| **TB** | test bench |
| **UART** | Universal Asynchronous Receiver/Transmitter |

# 

# Introduction

## Background and Motivation

With the increasing importance of efficient and safe energy storage systems, the use of Lithium-Ion (Li-Ion) batteries has become widespread across various applications, including electric mobility, portable electronics, and energy management systems. However, testing and characterizing Li-Ion batteries requires careful attention to safety standards and precise measurement techniques due to the potential risks associated with battery failures, such as thermal runaway and fire hazards.

In 2023, a prototype for a bicycle power management system was developed at Esslingen University. Initially, a Nickel-Metal Hydride (Ni-MH) battery was used for testing purposes. However, the long-term intention was to utilize a Li-Ion battery due to its higher energy density and better performance characteristics. Unfortunately, the absence of an appropriate battery test facility in the lab limited the possibility of safely testing Li-Ion batteries at that time.

This gap in infrastructure highlighted the need for a compact, reliable, and safe battery testbench tailored for Li-Ion batteries. The current project aims to develop such a small-scale battery testbench, capable of performing necessary measurements, safety verifications, and functional tests under controlled conditions. An Arduino microcontroller integrated with Bluetooth communication will serve as the core control unit, enabling remote monitoring and management via a smartphone application.

The motivation behind this project is the need of a safe testing environment for high-energy battery systems, to support the further development of mobile energy solutions, and to provide a flexible, educational platform for students to deepen their understanding of electrical measurement techniques, embedded system programming, and battery management.

## Purpose of Test Bench

The primary objective of this project is to design, develop, and validate a small battery testbench capable of safely testing and evaluating Li-Ion batteries. The developed testbench must fulfill both safety and functional requirements, ensuring reliable operation during the testing process.

**Specific objectives of the project include:**

* **Conducting a theoretical analysis of Li-Ion battery characteristics and their implications for safe testing and management.**
* **Reviewing and adapting an existing battery testbench design to meet current project needs.**
* **Planning and constructing an electrical and mechanical setup for the testbench, including part selection and facility integration.**
* **Developing embedded software for an Arduino-based control unit, enabling communication with electric loads, power supplies, smartphones, and measurement PCs.**
* **Ensuring comprehensive testing of the system’s safety measures and operational functionalities with various types of Li-Ion batteries.**
* **Creating a simulation model based on experimental test results for further analysis and parametrization.**
* **Establishing a wireless user interface via smartphone connection to monitor and control the testbench remotely.**
* **Optionally, integrating the system with a bicycle-dynamo project for extended application testing.**
* **Implement automatic shutdown of system when voltage exceeds the high voltage protection threshold (HVP) to prevent thermal runaway or cell damage.**
* **Implement automatic shutdown of system when voltage falls below the LVP threshold, avoiding cell over-discharge and potential degradation.**
* **To monitor battery temperature continuously. Suspend operations when temperature exceeds the safe operating limit to prevent overheating, fire hazards, or component failure.**
* **Integrate current sensing mechanisms to detect overcurrent conditions during both charging and discharging.**
* **Alert the user in case of any emergency shutdown via mobile app.**

Through these objectives, the project aims to deliver a compact, cost-effective, and educationally valuable test environment, fostering deeper understanding and skill development in battery management, embedded programming, and system integration.

## Structure of the Report

This report is structured to systematically present the development process, design decisions, and validation of the small battery testbench project.

**Chapter 2** provides a theoretical background on Li-Ion battery technology, including characteristics, operational principles, and safety considerations necessary for the design of a reliable test environment.

**Chapter 3** details the system requirements and the initial analysis of existing solutions, leading to the definition of the specific needs for this project.

**Chapter 4** describes the conceptual design and system architecture. The small battery testbench consists of an Arduino-based control unit, communication interfaces for electric load, power supply, smartphone app integration via Bluetooth, and measurement systems for monitoring and evaluation.

**Chapter 5** outlines the practical implementation, including the electrical and mechanical construction of the testbench hardware, software development for the control unit, and integration with external devices.

**Chapter 6** covers the testing and validation phase, presenting the procedures used to verify the functionality, safety, and performance of the testbench. This chapter also includes results from simulation model parametrization and real-world battery tests.

**Chapter 7** summarizes the project outcomes, discusses encountered challenges, and provides recommendations for further improvements or extensions, such as integration with additional energy sources like a bicycle-dynamo.

# System Requirements

## Overview of functional and safety requirements

|  |  |  |
| --- | --- | --- |
| **S.No:** | **Category** | **Requirements** |
| 1 | Functional | TB must charge the battery by using JTPS1440-DC power supply. |
| 2 | Functional | The DC power supply must be controlled by **Arduino nano (ESP32)** |
| 3 | Functional | TB should charge the battery using Constant Current/Constant Voltage method |
| 4 | Functional | TB must discharge the battery. |
| 5 | Functional | TB should receive an input for discharge rate from the user via mobile app |
| 6 | Functional | TB should measure the basic parameters of battery |
| 7 | Functional | TB must measure the terminal voltage of battery voltage sensor |
| 8 | Functional | TB must measure the current of battery current sensor |
| 9 | Functional | TB must measure the Temperature of battery by temperature sensor |
| 10 | Functional | TB must measure the SOC of battery |
| 11 | Functional | TB must measure the SOH of battery |
| 12 | Safety | TB must operate under the defined safety conditions |
| 13 | Safety | TB must stop the charging when its **voltage** exceeds the **High Voltage Protection Level (HVP)** |
| 14 | Safety | TB must stop the discharging when its **voltage** reaches the **Low Voltage Protection Level (LVP)** |
| 15 | Safety | TB must stop its operation when its **temperature** reaches the value of **certain temperature.** |
| 16 | Safety | TB must stop its operation when its **Current** reaches beyond the Maximum **current value.** |
| 17 | Functional | TB must communicate with mobile app MIT Inventor |
| 18 | Functional | TB mobile app must receive the measured **Voltage, Current, Temperature, SOC, SOH** in relatively small intervals |
| 19 | Functional | TB mobile app must enable displaying Voltage, Current, Temperature, SOC, SOH as graphical representations |
| 20 | Functional | TB mobile app must enable displaying Voltage, Current, Temperature, SOC, SOH as real time values |
| 21 | Functional | TB mobile app must save the received Voltage, Current, Temperature, SOC, SOH |
| 22 | Safety | TB mobile app must alert the user in any case of **noncompliance of safety** |
| 23 | Functional | User must be able to use charging and discharging option |
| 24 | Functional | TB mobile app must control the charging/discharging mode switching |
| 25 | Functional | App must display real-time values of Voltage, Current, Temp, SOC, SOH |
| 26 | Functional | App must show data in graphical form |
| 27 | Non-Functional | App must save telemetry data |
| 28 | Non-Functional | App must save data in CSV or similar file format |
| 29 | Functional | TB must allow configuration of voltage/current thresholds via app |
| 30 | Non-Functional | Mobile app must run smoothly on **Android 8+ devices** |
| 31 | Non-Functional | System should be **modular** in design for easy replacement of sensors or microcontroller |
| 32 | Functional | The system should be able to perform different charging and discharging sequences. |
| 33 | Non-Functional | The system should be able to make charging and discharging curves. |

Table 1 Requirements Table

## Functional Design/ Solution / Schematic:

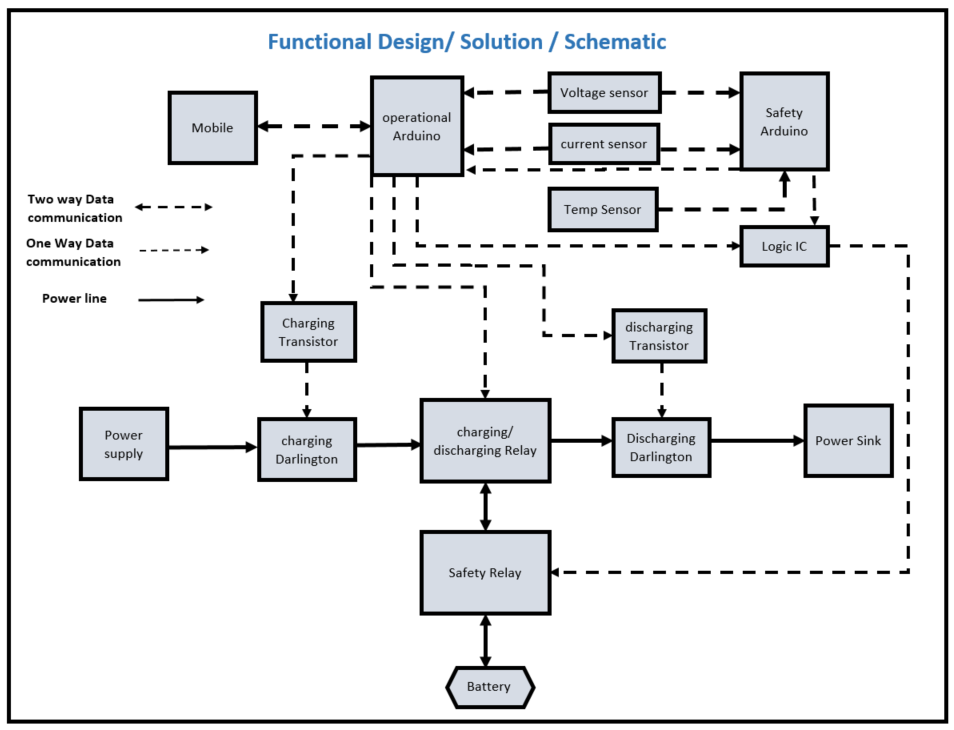


Figure 1: Functional Design/ solution/ Schemtic

The user specifies the desired mode of operation, current, and charging/discharging duration via a mobile application. This data is transmitted to the operational Arduino. Simultaneously, the Arduino receives input from three sensors and verifies the validity of the selected operation mode based on real-time sensor readings. Upon successful validation, the Arduino activates the safety relay to enable the circuit and triggers the charging/discharging switching relay to select the appropriate mode of operation. In case of error, both Arduinos send signal to safety relay to turn it off. (Figure 1)

In **charging mode**, the Arduino sends a control signal to the charging transistor, which regulates the current flow to the charging Darlington transistor. The Darlington transistor then manages the current supplied from the power source to the battery.

In **discharging mode**, the Arduino sends a control signal to the discharging transistor, which governs the current flow to the discharging Darlington transistor. This transistor subsequently controls the current flow from the battery to the power sink.

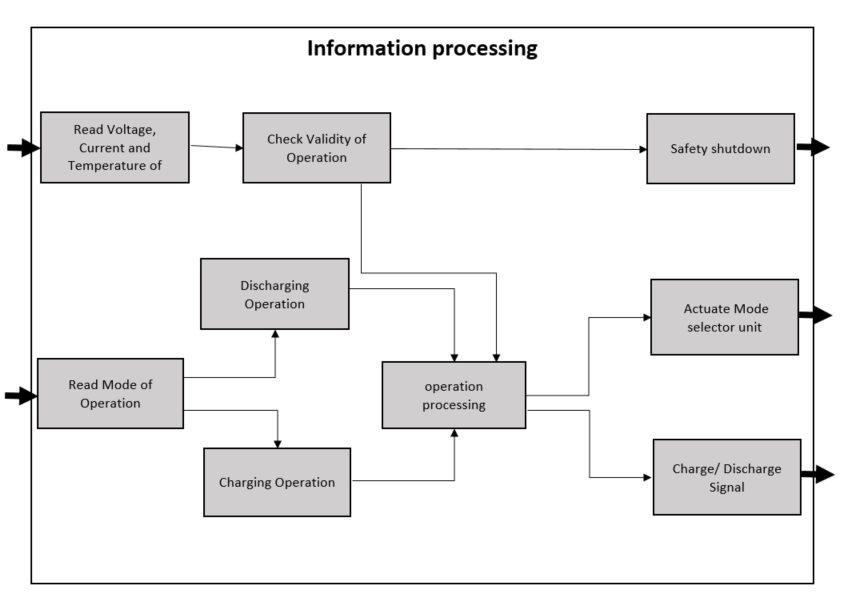


Figure 2: Functional architecture

The system continuously monitors the battery’s voltage, current, and temperature, along with the user-selected mode of operation—either charging or discharging. Based on the selected mode, it initiates the corresponding process only after verifying that all measured parameters are within defined safe operating limits. If the operation is deemed valid, the system performs necessary calculations and sends appropriate signals to the mode selector unit to switch to the correct mode, while also activating the charging or discharging mechanism accordingly. If any parameter falls outside the safe range, the system aborts the process and initiates a shutdown to ensure safety. (Figure 2)

# Hardware Architecture

## Description main components:

Sensors (Voltage, Current, Temperature, SOC, SOH), Charging Circuit

Justification of component selection

### Battery:

|  |  |
| --- | --- |
| Model: | INR21700-50E |
| Cell Type: | Lithium-ion (Li-ion), cylindrical Size: 21700 (21mm diameter × 70mm length) |
| Nominal Voltage: | 3.6V |
| Max Charge Voltage: | 4.2V |
| Capacity (Nominal): | 5000mAh (5.0Ah) |
| Minimum Capacity: | ~4900mAh |
| Standard Charge Current: | 2.45A |
| Max Continuous Discharge Current: | 9.8A |
| Charging Temperature Range: | 0°C to 50°C |
| Discharging Temperature Range: | -20°C to 60°C |
| Chemistry: | NCA (Nickel Cobalt Aluminum Oxide) |
| Protection Circuit: | None |

Table 2 Battery Specifications

for more Nominal Specifications refer to *Figure 2*.

### Arduino Nano (ESP32):

The Arduino Nano ESP32 (Figure 3) is a compact development board based on the ESP32-S3 microcontroller, featuring a 32-bit dual-core Xtensa LX7 processor. It operates at **3.3V** and supports both **Wi-Fi** (802.11 b/g/n) and **Bluetooth Low Energy 5.0**, making it suitable for IoT applications. The board includes **16MB of flash memory and 512KB of SRAM**, providing sample space for complex firmware. It connects via a **USB-C** port and is powered by 5V through USB. Despite its small footprint—roughly 18 by 45 millimeters—it offers **22** general-purpose **I/O pins**, **8 analog inputs with 12-bit ADC resolution**, and support for multiple communication protocols including **UART, SPI, I2C, I2S, and CAN**. The board is designed to operate in a wide temperature range from **-40°C to 85°C**.

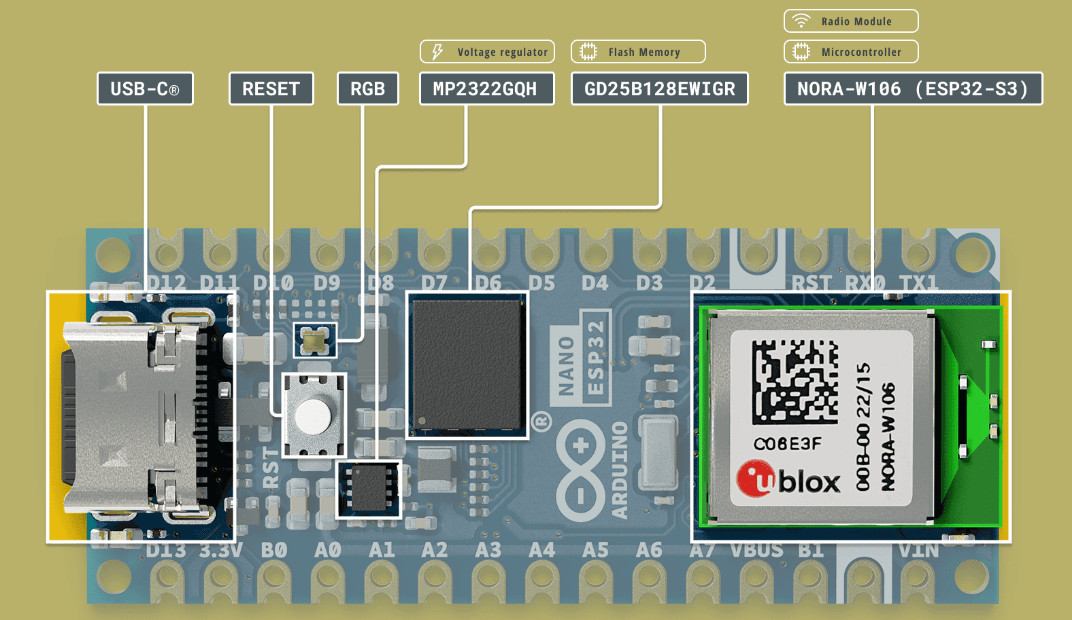


Figure 3: Screenshot from Arduino Nano ESP32 Cheat Sheet (Arduino, 2025

### TIP 145 Darlington:

The **TIP145** is a silicon **PNP Darlington power transistor** designed for high current and voltage applications. As a **monolithic device**, it integrates two transistors within a single package, offering high current gain. It also includes built-in base-emitter shunt resistors, which enhance thermal stability and performance. (Figure 4)

The Data sheet for the TIP 145 Darlington Transistor can be accessed here:  
<https://www.farnell.com/datasheets/2920591.pdf>

In practical applications, the TIP145 can be used for **battery charging circuits**, where it controls the flow of current into the battery. As the **base current** increases, the transistor allows a proportionally larger **collector current** to pass, thereby regulating the charging current delivered to the battery.

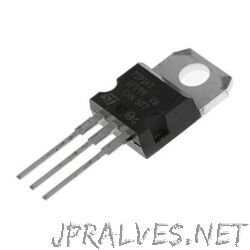
 .

Figure 4: TIP 145 Darlington Transistor

### TIP 140 Darlington:

The **TIP140** is a 60V **NPN Darlington power transistor** designed for general-purpose amplification and low-speed switching applications. It is a **monolithic device**, incorporating two transistors in a Darlington configuration, along with built-in base-emitter shunt resistors to improve switching performance and thermal stability.(Figure 5)

The data sheet of TIP 140 Darlington transistor can be accessed here:  
<https://www.farnell.com/datasheets/2864209.pdf>

In practical applications, the TIP140 can be used in **battery discharging circuits**, where it controls the flow of current from the battery to a load. As the **base current** increases, the transistor allows a larger **collector current** to flow, thereby regulating the discharge current from the battery.

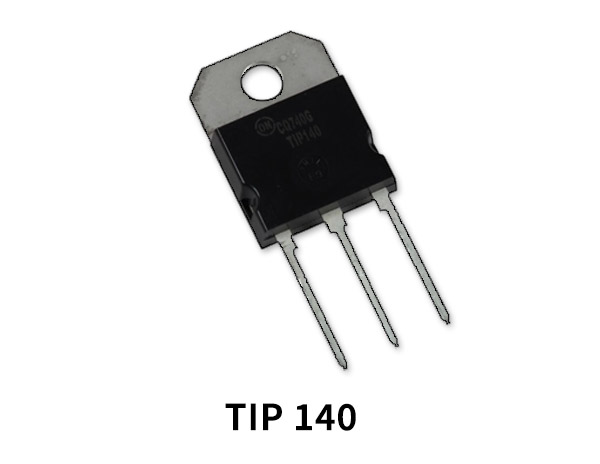


Figure 5: TIP 140 Darlington Transistor

### BC-337 Transistor:

The **BC337** is a popular **NPN bipolar junction transistor (BJT)** often used in everyday electronic circuits. It's commonly chosen for **general-purpose amplification and switching** tasks. Thanks to its ability to handle moderate levels of current and power, the BC337 works well for driving small loads or functioning as a switch that's controlled by a low-power signal—like one from a microcontroller or sensor. (Figure 6)

In this project, the BC337 is used to control both the TIP140 and TIP145 Darlington transistors, acting as a low-power driver to switch these high-current devices. Additionally, it is also used to control relays, making it a versatile component for managing both solid-state and electromechanical switching operations.

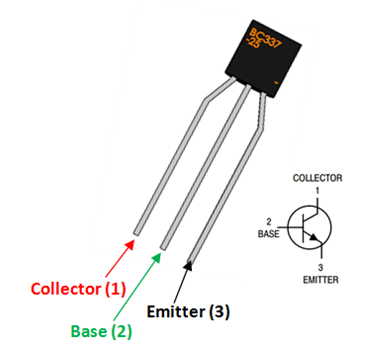


Figure 6: BC-337 Transistor

### ADS1115 ADC:

The ADS1115 is a 16 Bit ADC with I2C interface, designed for precision, low-power, and space constraint sensor measurement application. It allows user to convert analog signal to digital values with high accuracy, suitable for applications like sensor data processing and battery management. The ADS1115 can handle up to four single-ended or two differential input measurements.(Figure 7)

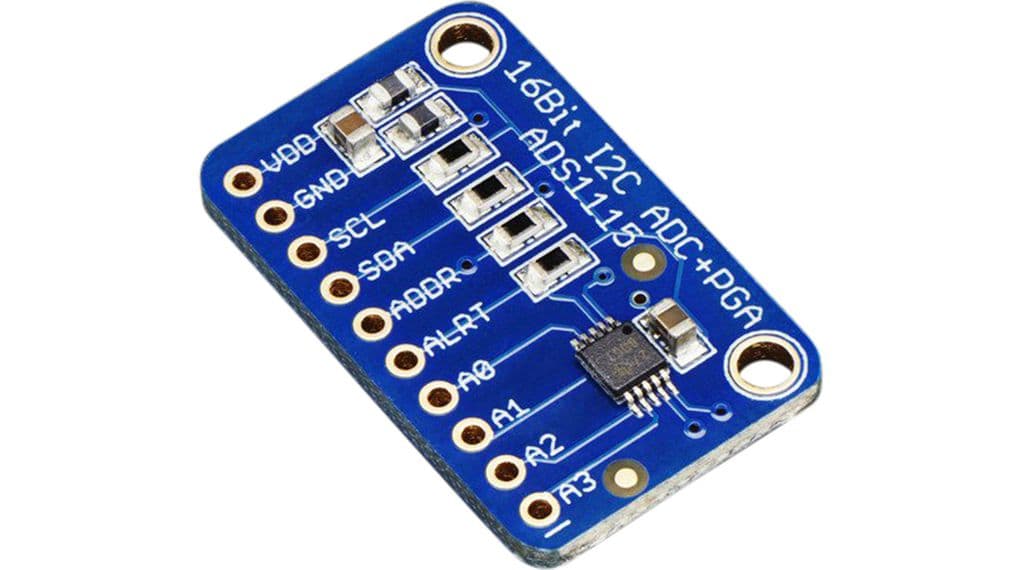


Figure 7: ADS1115 ADC

### Power resistor:

Power resistor are a type of resistor designed to handle a large amount of electrical power, converting it into heat. They are built to withstand higher temperatures and dissipate significant amount of heat without damage, making them suitable for applications where energy needs to be safely dissipated as heat.(Figure 8)

In our project, we have used **three power resistors**, each serving a specific function: one is used in the **charging circuit**, another in the **discharging circuit**, and the third acts as a **shunt resistor** for measuring current indirectly through the voltage drop across it.

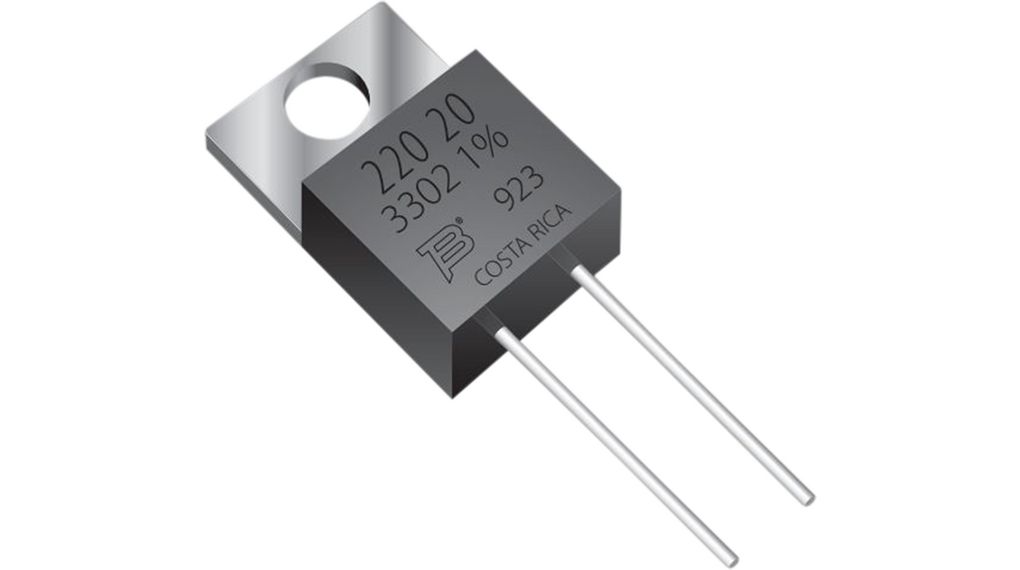


Figure 8: Power resistor

### Relays:

Relays are electrically operated switches that uses a small electrical signal to control a much larger electrical current. They act as a bridge between two circuits, allowing a low-power control signal to switch a high-power circuit on or off. (Figure 9)

There are two main types of relays: **electro-mechanical relays** and **solid-state relays**. In our project, we have used **two electro-mechanical relays**. One relay functions as a **safety switch**, allowing the battery to be connected to or disconnected from the circuit as needed. The second relay is used to **switch the operating mode** of the circuit, toggling between **charging and discharging** modes.



Figure 9: SPDT Electro-Mechanical relay

### Fuse:

A fuse is a safety device in an electrical circuit that protects against excessive current flow. It contains a thin wire or a metal strip that melts and breaks when the current exceeds a safe level, preventing damage to the device and wiring. (Figure 10)



Figure 10: Fuse

### LEDs:

LED stands for Light Emitting Diode. It's a semiconductor device that emits light when an electrical current passes through it. LEDs are known for their energy efficiency, long lifespan, and ability to produce various colors of light. (Figure 11)

**Different colored LEDs** have been used in the project to indicate the **status of the circuit** (ON or OFF) as well as to represent the **different modes of operation**, such as charging or discharging. This provides a simple and effective **visual indication** of the system’s current state.



Figure 11: LED

### DS18B20 Temperature Sensor:

The DS18B20 is a popular, digital temperature sensor that communicates using the **1-Wire protocol**. It's known for its ease of use, accuracy, and affordability, making it suitable for various applications including data logging and temperature control. The sensor can measure temperatures from -55°C to +125°C and has a typical accuracy of ±0.5°C within the range of -10°C to +85°C. (Figure 12)

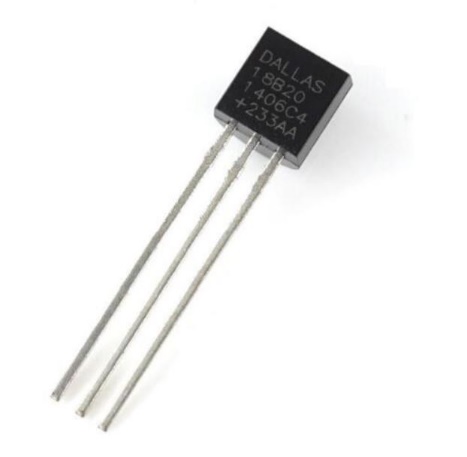


Figure 12: Dallas DS18B20 Temperature sensor

### M74HC27B1 logic IC:

The M74HC27B1 is an High speed CMOS triple 3-input NOR gate IC. A NOR gate produces a high output only when **all its inputs are low**. If **any input is high**, the output goes low.

This IC is used to trigger the safety relay. The ic is connected to both operational and safety Arduino. (Figure 13)

This **M74HC27B1 IC** is used to **trigger the safety relay** in the system. It is connected to both the **operational Arduino** and the **safety Arduino**, allowing it to process logic signals from both sources. The relay is activated only under specific logic conditions, ensuring that the battery is connected or disconnected safely based on the system's current state.

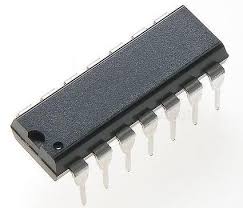
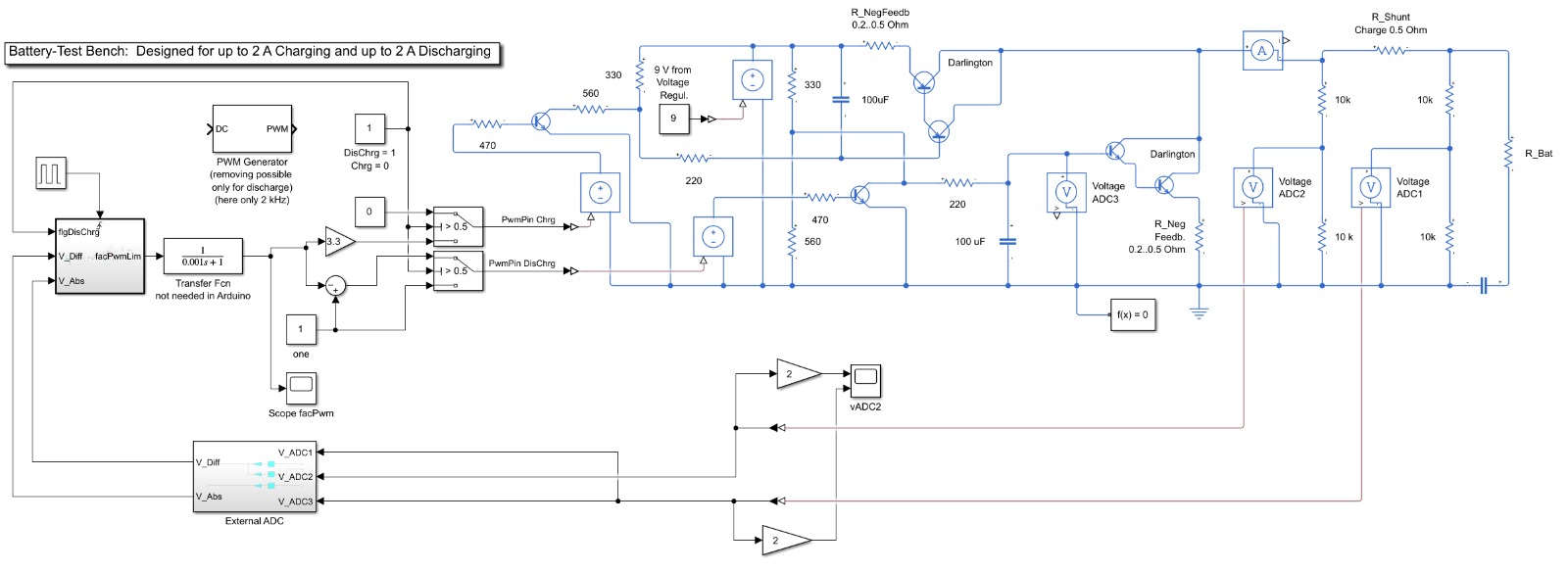


Figure 13: M74HC27B1 IC

## Simulink Model and Simulation Results:

The aim of this simulation study was to analyze the effects of different electrical and control parameters on a battery test bench during charging and discharging operations. Using a Simulink-based model, various system-level components such as PWM generators, current sensors, ADC resolutions, controller gains, and passive components were varied to understand their influence. The primary goal was to finalize the key parameters for robust and stable control of battery current under different operating conditions. (Figure 13)



Charging Module

Battery (Load)

Controller

Measurement Module

Discharging Module

External ADC

Figure 14: Test Bench Simulink/Sims cape model

# SUB MODULES OF SIMULINK DIAGRAM:

## Charging Module:

In this sub-circuit of the charging module, a Darlington transistor is used to supply a controlled charging current to the battery. The Darlington pair is driven by a BC337 transistor, which in turn is controlled via a PWM (Pulse Width Modulation) signal generated by an Arduino microcontroller. An RC (Resistor-Capacitor) filter is included in the circuit to smooth out the PWM signal.

PWM controls the BC337 transistor's base current, which adjusts the drive to the Darlington pair. Since Darlington transistors provide high current gain, this configuration allows for efficient control of a larger charging current using a relatively low-power signal from the Arduino. (Figure 14)

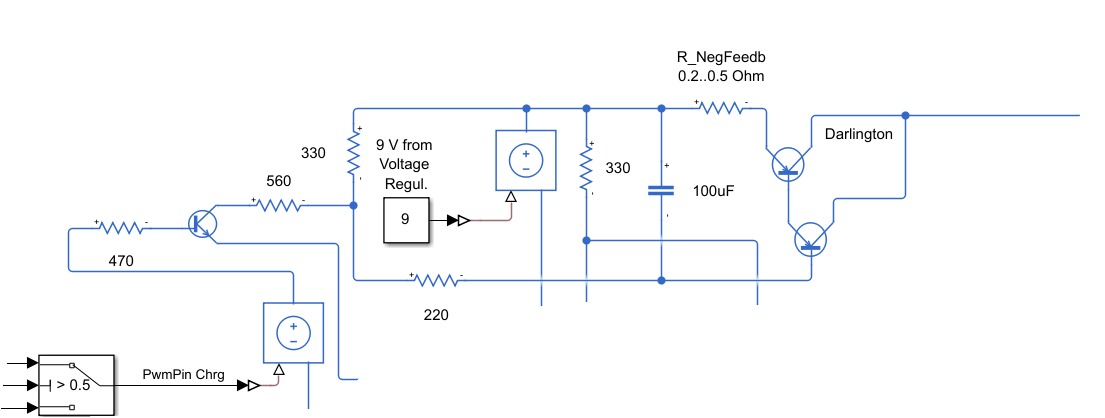


Figure 15: Charging Module

## Discharging Module

In the discharging (load) sub-circuit, a Darlington transistor is used to sink current from the battery, effectively allowing it to discharge in a controlled manner. A BC337 transistor acts as the driver for the Darlington pair and is controlled using a PWM signal from the Arduino. To ensure proper operation and smooth control, a RC filter is used to convert the PWM signal into a stable analog voltage. PWM controls the base of the BC337 transistor, regulating how much the Darlington conducts.

After the Darlington transistor, a power resistor is connected to dissipate the current drawn from the battery, acting as the actual load. This resistor converts the electrical energy into heat, ensuring safe and measurable battery discharge. power supply is provided to bias and operate the BC337 transistor properly, ensuring that the control circuit is electrically isolated or stable regardless of the battery's state of charge. (Figure 15)

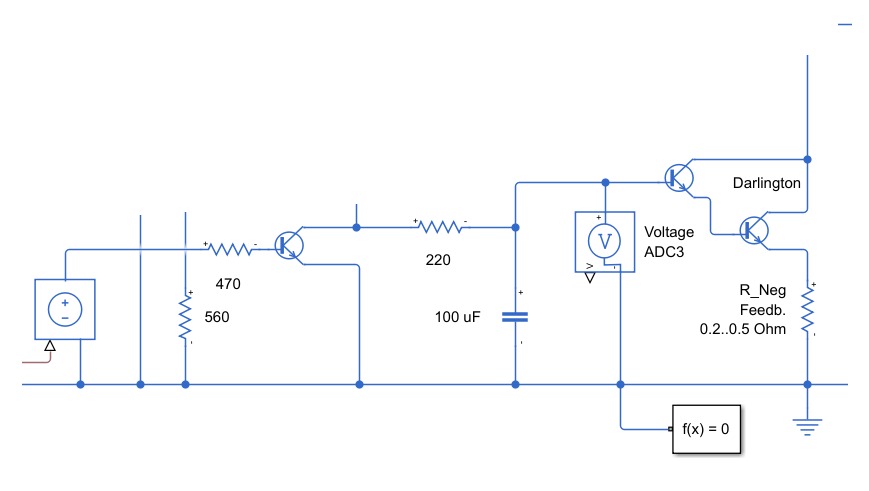


Figure 16: Discharging Module

## Battery (Load):

Resistor and Capacitor in the circuit act as a simulated battery, cell, or load, forming a basic energy storage and dissipation element. This setup allows the circuit to both charge and discharge, enabling testing and simulation within the system.

## External ADC:

To measure current accurately in the circuit, we use external 16-bit ADCs instead of the built-in 12-bit ADC of the Arduino, which provides higher resolution and precision for small voltage measurements. We measure voltage at two points around a shunt resistor placed in the discharge path. ADC1 and ADC3 measure the voltage before the shunt resistor (i.e., at the battery side). ADC2 measures the voltage after the shunt resistor.The difference between the voltage before and after the shunt resistor gives us a differential voltage, which is used to calculate the current flowing through the resistor using Ohm’s Law.The reason for using an external 16-bit ADC is to achieve more accurate and stable readings of small voltage differences, which the Arduino’s built-in 12-bit ADC may not capture reliably due to its lower resolution. (Figure 16)

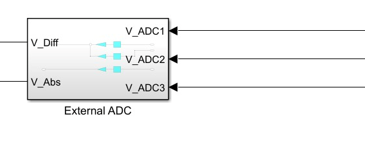


Figure 17: External ADC

## PI Controller with Limitations:

**In the controller section** of the circuit, a **Proportional-Integral (PI) controller** is implemented to regulate charging and discharging behavior. To ensure safe and stable operation, both **voltage and current limitations** are applied. The controller is designed to output a **PWM signal**, which controls the BC337 transistor and, ultimately, the Darlington pair for battery current regulation. (Figure 18)

### Voltage Limitation:

The system monitors the battery voltage and enforces upper and lower voltage limits — for example, 80V as maximum and 20V as minimum, which correspond to 100% and 0% State of Charge (SoC) respectively. If the battery reaches the maximum voltage limit, the system automatically switches from charging to discharging mode to prevent overcharging and protect the battery.Similarly, if the voltage drops below the minimum, discharging stops to avoid deep discharge and damage.This ensures battery health and keeping operation within a safe voltage range.

### Low-Pass Filter (After Voltage Measurement):

A low-pass filter is applied to the measured voltage signal before feeding it into the controller.

This filter removes high-frequency noise and smooths the signal, ensuring that the PI controller receives a clean and stable voltage value.Without filtering, rapid voltage fluctuations or electrical noise could cause the controller to react unnecessarily, leading to instability or oscillations in PWM output.

### Current Limitation using Differential ADCs:

To control the amount of current during charging and discharging. The circuit uses external 16-bit differential ADCs to measure the voltage drop across a precision shunt resistor.This differential voltage is divided by the known resistance value (0.2 Ohm) to calculate the actual battery current.The measured current is then compared with user-defined current limits for both charging and discharging.

### PI Controller with Anti-Windup:

The PI controller processes the difference between the desired and actual current to generate the control output as a PWM signal. Anti-windup protection is implemented to prevent the integral term from accumulating excessively when the actuator (PWM signal) is saturated — for example, when the system hits voltage or current limits. Without anti-windup, the integral term could overshoot, leading to slow recovery, instability, or overshooting when conditions return to normal.

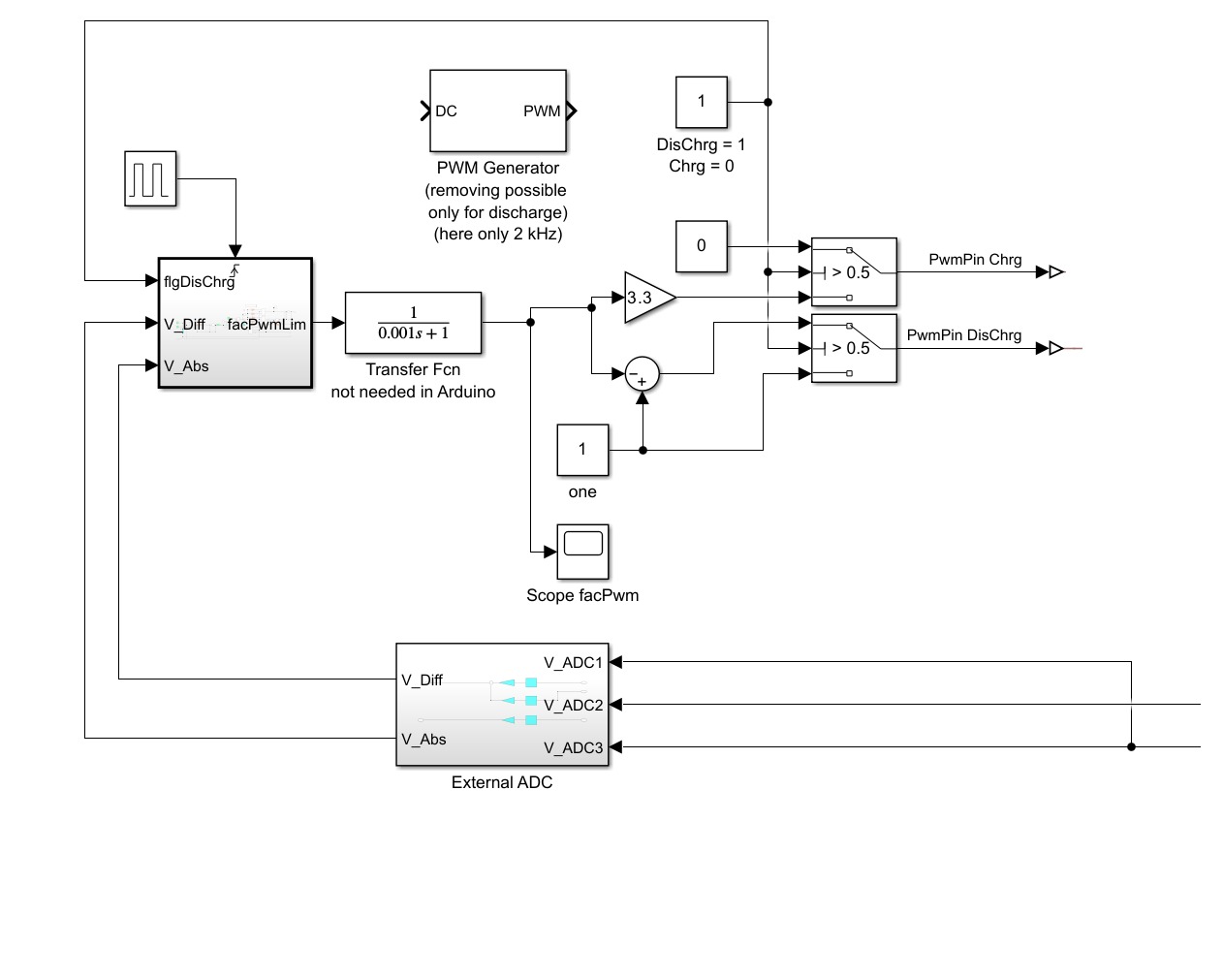
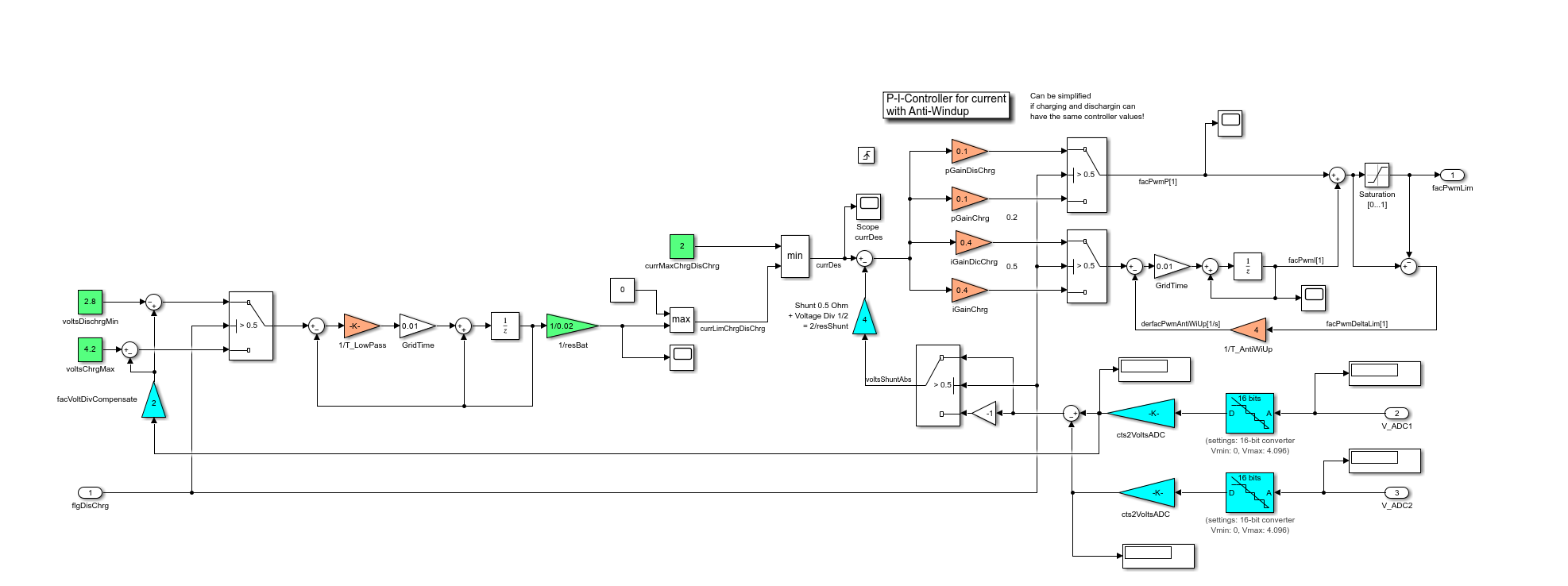


Figure 18: PI Controller and Switch



PI Controller with Anti windup

Current Limitation by Diff ADCs

Voltage Limitation

Low Pass Filter

Figure 19: PI Controller Detailed

# Charging and Discharging Mode Experiments:

## PWM, Current, and ADC Voltages at 2A

**Observations:**

**For Charging Mode:**

At the beginning of the charging process, the battery is at a low state of charge (SoC), so the system allows the maximum possible current to charge it efficiently. This is achieved by setting the PWM (Pulse Width Modulation) signal to its maximum value. As the charging progresses, the controller continuously monitors the voltage difference between the battery and the power supply. Based on this difference, it adjusts the PWM duty cycle accordingly. After the initial phase, the PWM stabilizes for a period to maintain a consistent charging current. As the battery approaches its fully charged state, the PWM gradually decreases. Finally, when the battery reaches its maximum voltage threshold, the PWM is reduced to zero, effectively stopping the charging process to prevent overcharging.

**Discharging Mode**

The discharging process operates on a similar principle but behaves slightly differently due to the characteristics of the discharging Darlington transistor. At the start, a low PWM value allows for a high discharge current, resulting in a faster discharging rate. As the PWM value increases, the discharging current decreases, slowing down the discharge process. When the battery nears its minimum allowable voltage, the PWM reaches 100%, effectively stopping further current flow to prevent deep discharge and protect the battery. (Figure 20)

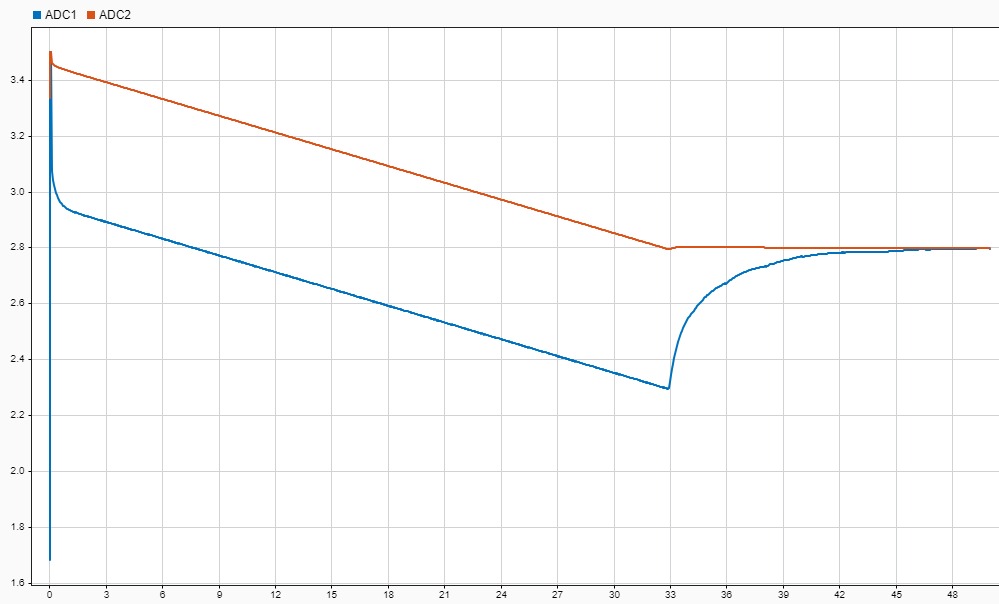


Figure 20: ADC Voltages

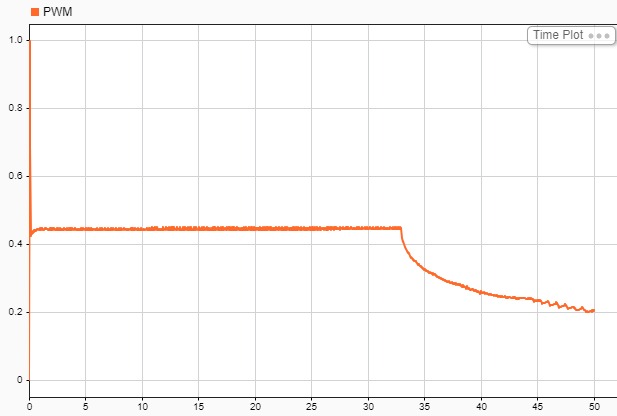


Figure 21: PWM Controller for Discharging

# Circuit Diagram:

12 Bit ADCs

14 Bit ADCs

The circuit presented was developed as part of a custom two-layer PCB design to accommodate a power electronics application. The entire hardware design process was carried out using **KiCad**, an open-source EDA tool, which facilitated schematic capture, footprint assignment, **PCB layout**, and **3D visualization**.

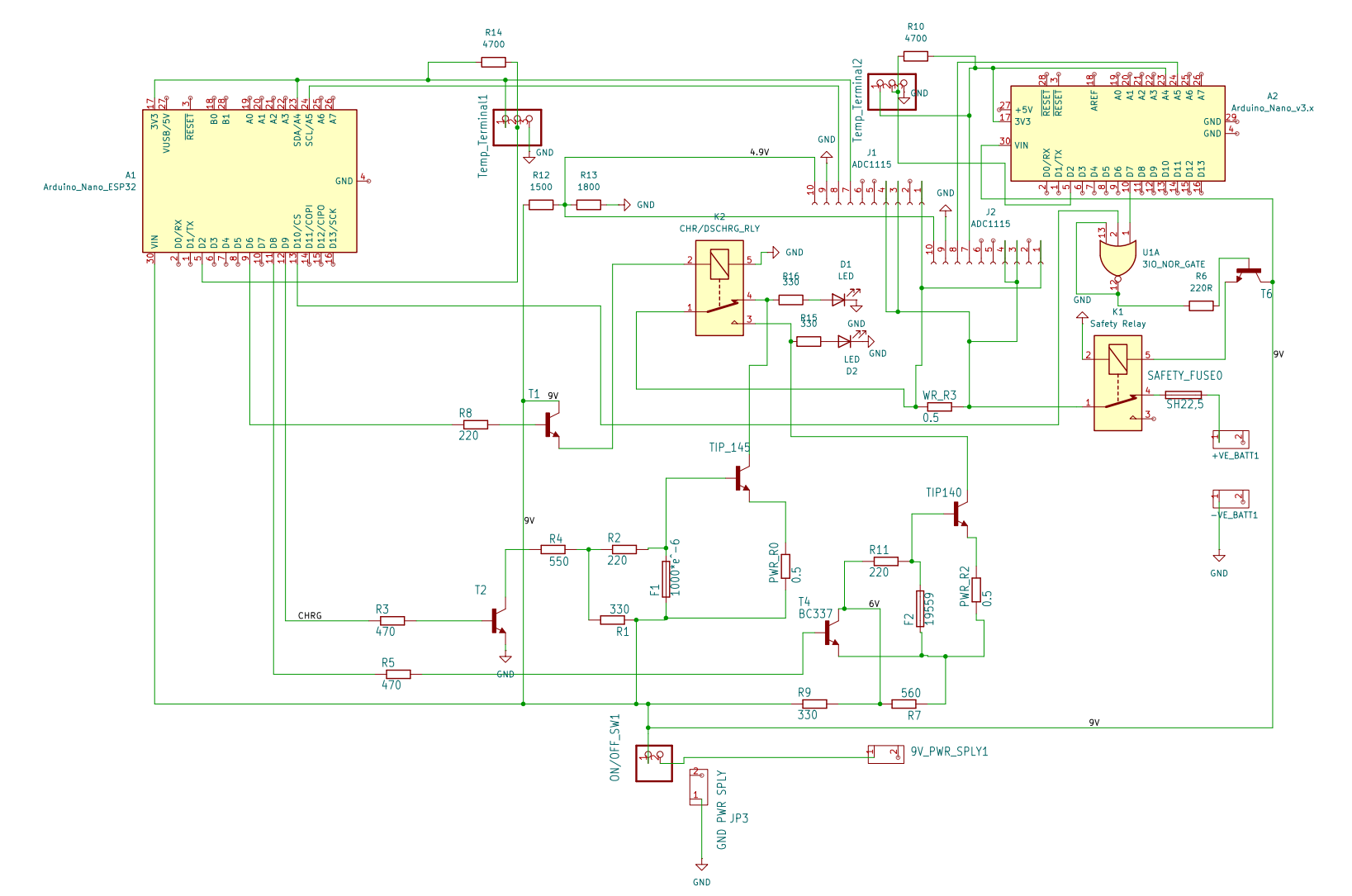
The hardware development procedure began with drafting the circuit schematic, followed by validating component ratings and net connectivity. Special attention was given to the **arrangement of components** on the board to ensure minimal thermal interference and short, low-impedance paths for high-current traces.

The design incorporates both **top and bottom layers**, where the top layer primarily handles the power stage and key signal routing, while the bottom layer was utilized for return paths and signal integrity.

Notably, **custom footprints were designed manually** for the **TIP140 and TIP145 Darlington transistors** as well as the **banana sockets** used for external connectivity. These components had non-standard pin spacing and mounting dimensions, requiring careful measurement and footprint modeling in KiCad’s footprint editor. Proper silkscreen markings and pad sizing were also considered to simplify assembly and maintenance.

Overall, the design choices in layer management, component placement, and thermal considerations reflect a practical approach toward **manufacturability** and performance, aligned with **industry-standard** hardware development procedures.

Figure 22 Full Circuit Diagram

****

## PCB Layout:

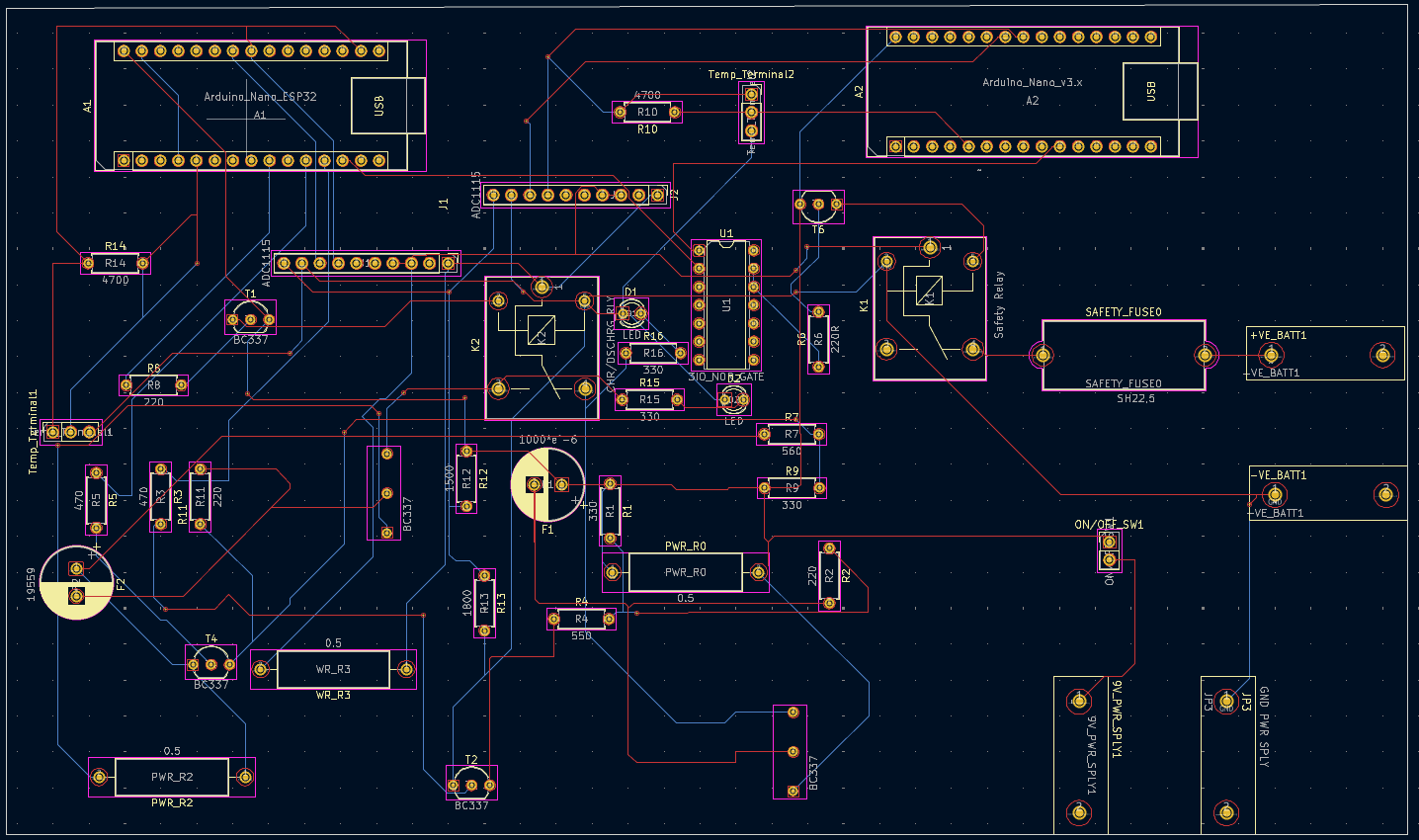


Figure 23 PCB layout routed

## 

## PCB 3D Rendered View:

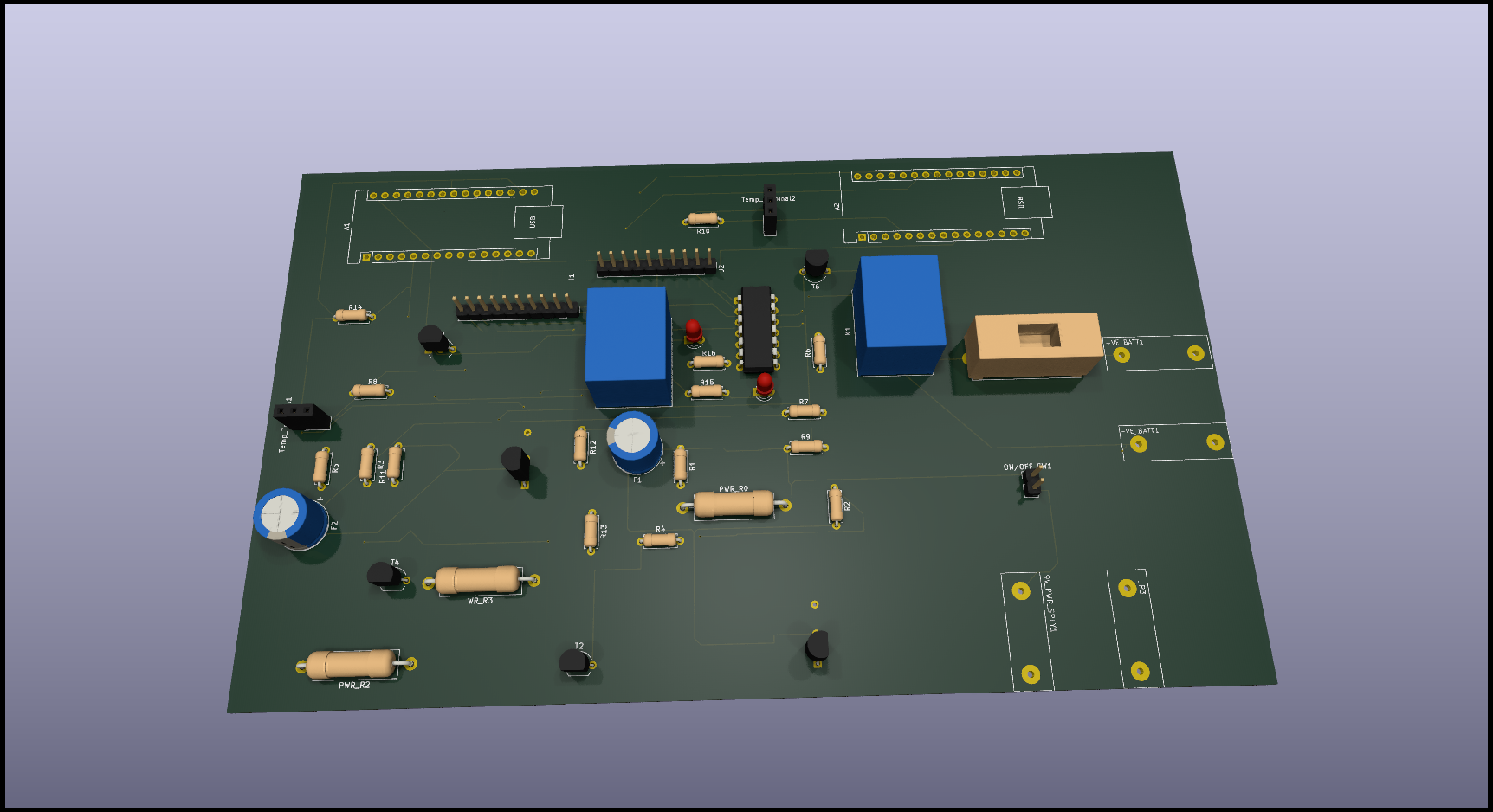


Figure 24 PCB 3D Rendered view

## 

## Creating a PCB process using blocks diagram

Figure 25 PCB life cycle

**Simplified Process Char**t

1. **Create Schematic**
2. **Assign Footprints**
3. **Import Schematic to PCB Layout**
4. **Define Board Outline and Place Components**
5. **Route Manually or Export to FreeRouting**
6. **Import Routed File Back to PCB Tool**
7. **Check Design Rules**
8. **Export Gerber Files**
9. **Send to Manufacturer**

***Detailed Explanation of Each Step***

**1. Create Schematic**

**Start:** Use a schematic editor - KiCad

* **What to do:**
  + Place your components (ICs, resistors, etc.)
  + Connect them with nets (wires)
  + Add power and ground symbols
  + Annotate components (e.g., R1, U1)

**Transfer:** Once schematic is complete, run **Electrical Rules Check (ERC)** and then **assign footprints**.

**2. Assign Footprints**

**Start:** Use KiCad tool

* **What to do:**
  + Assign a physical footprint (e.g., SOIC-14, 0603) to each schematic symbol.
  + Custom footprints can be created if needed.

**Transfer:** Save and run **"Generate Netlist"** or **"Update PCB from Schematic"** depending on the tool.

**3. Import Schematic to PCB Layout**

**Start:** Open your PCB layout tool (e.g., KiCad).

* **What to do:**
  + Load the netlist or press **"Update PCB from Schematic"**.
  + All components will appear as a cluster ready to be placed.

**Transfer:** You can now start laying out the board.

**4. Define Board Outline and Place Components**

**Start:** Define the board shape using **edge cuts** layer.

* **What to do:**
  + Draw the board outline (rectangular, custom).
  + Place components logically (group by function, minimize wire lengths).
  + Set proper clearances and spacing.

**Transfer:** Save and prepare to route.

**5. Route Manually or Export to FreeRouting**

**Start:** Choose routing method.

* **Option A – Manual:**
  + Use PCB tool’s routing tools to route traces.
* **Option B – FreeRouting:**
  + Export **.DSN file** from PCB tool.
  + Open it in **FreeRouting**.
  + Run auto-router → Export **.SES file**.

**Transfer:** If using FreeRouting, next step is importing SES back.

**6. Import Routed File Back to PCB Tool**

**Start:** In your PCB layout tool.

* **What to do:**
  + Import the **.SES file**.
  + Verify that all traces are correct and match the design.
  + Make manual adjustments if needed.

**Transfer:** Save and prepare for design rule checking.

**7. Check Design Rules**

**Start:** Use DRC (Design Rule Check) tool.

* **What to do:**
  + Check for clearance errors, unconnected pins, via problems, etc.
  + Fix all violations.

**Transfer:** Once DRC passes, you’re ready to generate output files.

**8. Export Gerber Files**

**Start:** Use **Plot** and **Generate Drill Files**.

* **What to do:**
  + Export layers: top copper, bottom copper, solder mask, silkscreen, edge cuts.
  + Export **.drl** file (drill file).
  + Use **Gerber Viewer** to verify output.

**Transfer:** Zip all Gerber and drill files together.

**9. Send to Manufacturer.**

* **What to do:**
  + Upload the zipped Gerber folder.
  + Review preview.
  + Choose specs (thickness, finish, color).
  + Place the order.

***Considerations regarding assembly:***

74HCT27D consideration:

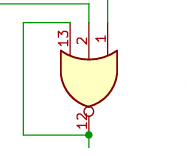


Figure 26 3 input NOR gate from the 74HCT27D

**Best Practices from Datasheet:**

**Unused INPUT pins**:

* + **Must not be left floating**.
  + Floating inputs can pick up noise, cause internal oscillation, increase power consumption, and may lead to unpredictable behavior or even damage in some cases.
  + To avoid unnecessary power consumption and possible oscillation, unused inputs should be tied to an appropriate logic level.

In our Case: We are using only 1 out of 3 NOR gates, so for the other 2 unused NOR gates:

* Tie all 3 inputs of each gate to GND (or VCC).

## Leave their outputs unconnected.

# Software & Control Logic

Arduino control structure, charging algorithm (CC/CV), Discharging logic and input handling, logic flow chart or similar diagrams, and Communication with MIT App Inventor.

## Charging Algorithm (CC/CV)

## 

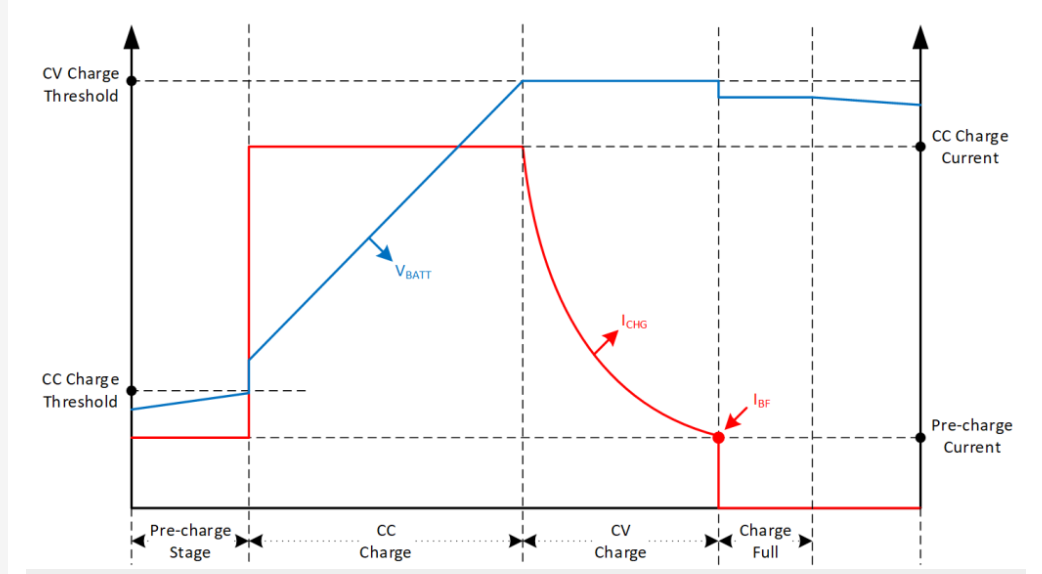


Figure 27 Screenshot from How to Build a CC/CV Battery Charger (Power Electronics News,2025)

**Pre-charge stage:** This stage is used to safely awaken deeply discharged batteries. A **low constant current** is applied to minimize stress and avoid damage. The charger monitors voltage until it reaches a defined threshold — the **CC Charge Threshold** — before switching to the next stage. Battery health diagnostics are typically performed during this phase.

**Constant Current (CC**): In this stage, the charger aims to **rapidly charge the battery** while keeping the current at a predefined maximum (safe) level — the **CC Charge Current**.  
It’s important to note that **the charger does not directly enforce constant current**. Instead, it **continuously increases the output voltage** in response to the battery's rising voltage and internal resistance. This dynamic voltage adjustment, combined with current sensing, **results in a constant charging current** due to Ohm’s law. The system continuously monitors the current and adjusts the voltage to maintain it near the CC limit.

**Constant Voltage (CV):** Once the battery reaches its target voltage (e.g., 4.2 V for Li-ion), the charger enters the CV phase, where the **voltage is held constant**. During this time, the **charging current naturally begins to decrease** as the battery becomes more saturated. The tapering of current must be monitored to determine when the battery is considered full.

**Charge Full**: The CV phase continues until the current drops below a predefined **cutoff threshold** (I<sub>cutoff</sub> or I<sub>BFK</sub>). When this condition is met, the battery is declared fully charged, and charging stops automatically to prevent overcharge and ensure battery longevity.

## Charging/Discharging flow chart:

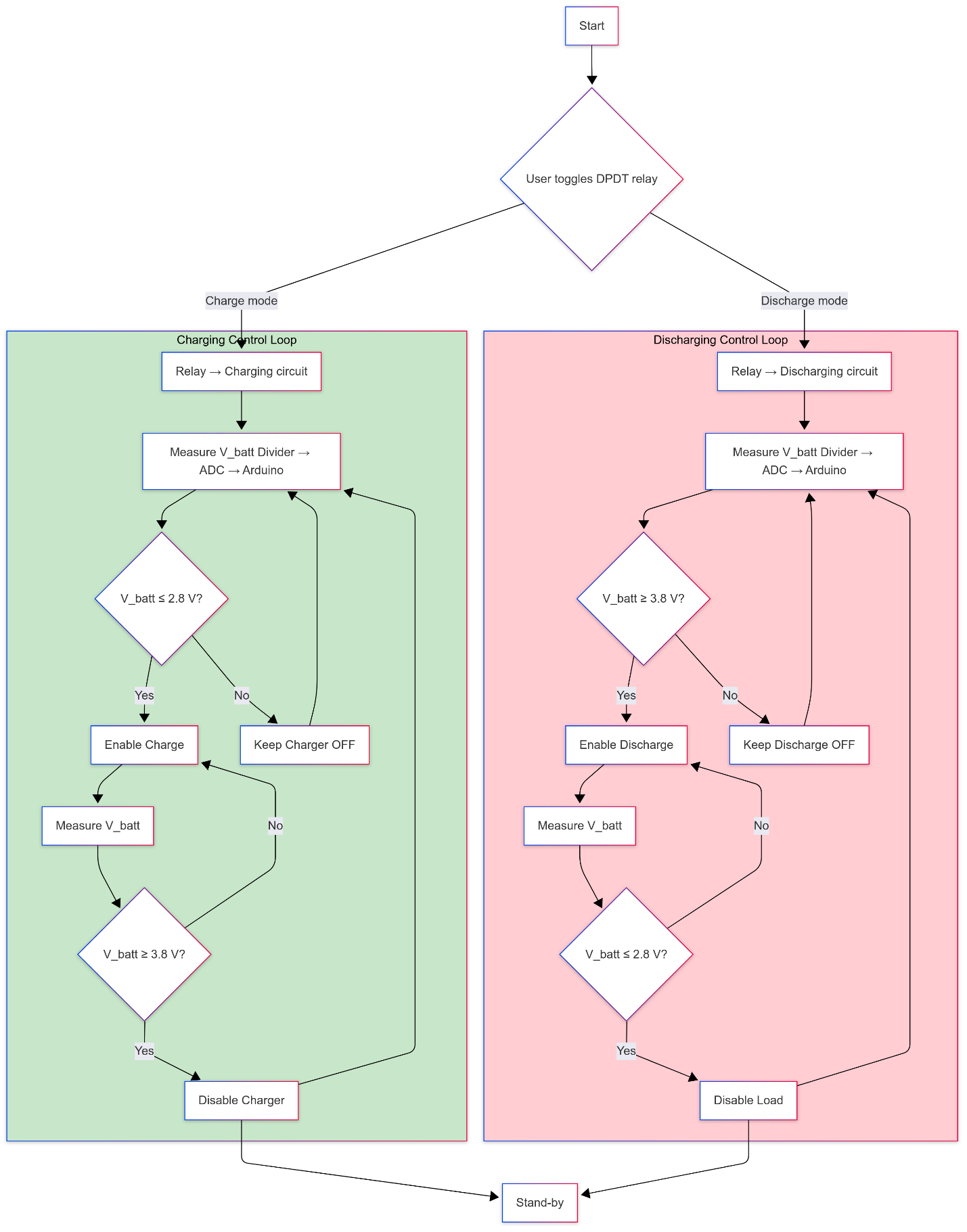


Figure 28 Charging/Discharging flow chart

## Use Cases

In this chapter, we describe the key user interactions with the battery testbench system through structured **use cases**. A use case helps to visualize the interaction between the user and the system to accomplish a specific task. The following terms are used throughout the use case tables:

* **Use Case**: The name of the task the user wants to perform.
* **Context of Use**: The situation in which the task is carried out, including the user’s goal.
* **Scope**: The parts of the system involved in fulfilling the task (e.g., hardware, software).
* **Level**: Indicates whether the use case is a high-level summary or a primary task.
* **Primary Actor**: The main user or system component that initiates the task.
* **Stakeholders & Interests**: People or roles affected by the system and what they expect from it.
* **Preconditions**: What must be true or set up before the task starts.
* **Success End Condition**: The desired final state after the task is completed successfully.
* **Failed End Condition**: Conditions that could lead to the task being aborted or failing.
* **Trigger**: What initiates the task (user input or automatic event).
* **Description**: A step-by-step explanation of how the task is carried out.

|  |  |
| --- | --- |
| **Use Case** | Start Battery Charging Cycle |
| **Context of use** | A user wants to safely charge a Li-Ion battery using the small battery testbench system with automatic voltage monitoring and control. |
| **Scope** | Battery Testbench System (Hardware + Arduino firmware + Smartphone interface) |
| **Level** | Primary Task |
| **Primary actor** | User (operator of the testbench) |
| **Stakeholder & Interests** | **User** – wants to safely test and charge batteries without risk.  **Supervisor** – expects safe, functional implementation for evaluation. |
| **Preconditions** | Battery is connected, system is powered, and the charging mode is selected through the UI or hardware switch. |
| **Success End Condition** | Battery is charged to max\_chrg\_volt and charger is safely turned off. |
| **Failed End Condition** | Charging is aborted due to voltage anomalies, user intervention, or hardware failure. |
| **Trigger** | User toggles mode switch or selects “Charging” on the Smartphone App. |
| **DESCRIPTION** |  |
| **Step** | **Action** |
| **1** | User toggles the relay switch or selects “Charging Mode” in the app. |
| **2** | Microcontroller checks battery voltage via ADC from the voltage divider. |
| **3** | If voltage ≤ 2.8V, charging circuit is enabled. |
| **4** | Voltage is monitored continuously during charging. |
| **5** | Once voltage ≥ 3.8V, charging circuit is turned off. |
| **6** | System goes to standby, waiting for next command. |
| **EXTENSIONS** |  |
| **3a** | If voltage > 2.8V on startup: Charging does not start. System waits and monitors. |
| **4a** | If voltage exceeds safety threshold (>4.2V): Emergency stop is triggered. |
| **VARIATIONS** | **Branching Action** |
|  | |  | | --- | | Battery not connected → Error displayed on UI; system prevents charging. | |  | |
|  | User aborts charging → System immediately disables charger. |

Table 3 Use Case battery charging

|  |  |
| --- | --- |
| **Use Case** | Start Battery Discharging Cycle |
| **Context of use** | A user wants to safely discharge a Li-Ion battery using the small battery testbench system with automatic voltage monitoring and cut-off control. |
| **Scope** | Battery Testbench System (Hardware + Arduino firmware + Smartphone interface) |
| **Level** | Primary Task |
| **Primary actor** | User (operator of the testbench) |
| **Stakeholder & Interests** | **User** – wants to safely test and charge batteries without risk.  **Supervisor** – expects safe, functional implementation for evaluation. |
| **Preconditions** | Battery is connected, system is powered, and the discharging mode is selected through the UI or hardware switch. |
| **Success End Condition** | Battery discharges to 2.8 V and load is safely disconnected. |
| **Failed End Condition** | Discharging fails due to undervoltage, disconnection, or user abortion. |
| **Trigger** | User toggles relay or selects "Discharging Mode" in smartphone app. |
| **DESCRIPTION** |  |
| **Step** | **Action** |
| **1** | |  | | --- | | User initiates discharging mode (switch or app). |  |  |  | | --- | --- | |  |  | |
| **2** | Arduino measures battery voltage via voltage divider and ADC. |
| **3** | If voltage ≥ 3.8 V → enable discharge load. |
| **4** | Continuously monitor voltage during discharge |
| **5** | If voltage ≤ 2.8 V → stop discharge |
| **6** | System returns to standby. |
| **EXTENSIONS** |  |
| **3a** | If voltage < 3.8 V on start → do not enable discharge; monitor periodically. |
| **4a** | If voltage drops below safety threshold (<2.5 V) → trigger emergency stop. |
| **VARIATIONS** | **Branching Action** |
|  | |  | | --- | | Battery not connected → Show error, disable discharging. | |  | |
|  | User aborts discharging → Stop discharging immediately and go to standby. |

Table 4 Use Case battery discharging

## Module Responsibility Map

## 

These are the modules for the Main Arduino ESP23 and their descriptions. Also following the table, a module dependency diagram is provided for further understanding

| **Module / File(s)** | **Key Responsibilities** | **Exposed APIs** |
| --- | --- | --- |
| **bluetooth\_controller.(h/cpp)** | Initialize BLE server, Send strings through the server, check connection state | setupBLE(…); sendBLEString(…);  isBLEConnected(); |
| **circuit\_control.(h/cpp)** | Issue control and state change commands to the circuit, parse control strings which are received from the mobile App, read and set the circuit mode | getBoundarySettingsString();  parseControlString(…);  getCircuitMode();  setCircuitMode(…); |
| **circuit\_measurements.(h/cpp)** | Read battery voltage & shunt voltage via **ADS1115**, read temperature forwarded by the Nano, convert to SI units. | getVoltageAndCurrent(…), getRealtimeDataString(…) |
| **circuit\_operation.(h/cpp)** | The lowest-level module to the circuit which controls the signals sent from arduino to control it. sets PWM duty for charge or discharge, flips the DPDT relay, enforces pulse-mode, set pulse timers, get current pulse state, trigger safety relay | circuitOperationSetup((…);  operateCircuit(…);  getCircuitVoltage();  getCircuitCurrent();  setPulseOnTime(…);  setPulseOffTime(…);  setDesiredCurrent(…);  getPulseState();  triggerSafetyRelay(); |
| **circuit\_safety.(h/cpp)** | Validate every sample against the limits in **circuit\_config.h** and return safety codes, check if circuit is safe with input measurements, get error messages from the returned safety code from checkCircuitSafety | checkCircuitSafety(…);  isTemperatureSafe(…);  isCurrentSafe(…);  isVoltageSafe(…);  getSafetyCodeString(…); |
| **custom\_types.h** | Enumerations for circuit states (c, d, o) and safety codes. | - |
| **circuit\_config.h** | A file which includes general definitions for the Arduino configuration such as Pin mapping, BLE UUID, hard safety limits, default settings. | - |

Table 5 Module Responsibility Map

## Module Dependency Diagram

The following is a simplified diagram to describe how the modules in the code for The main Arduino ESP23 access each other.

to read the diagram, start with the module1 name going to the arrow direction to module2 , saying the module1 accesses module2

Examples:

main **accesses** Bluetooth\_controller

circuit\_control **accesses** circuit\_operation

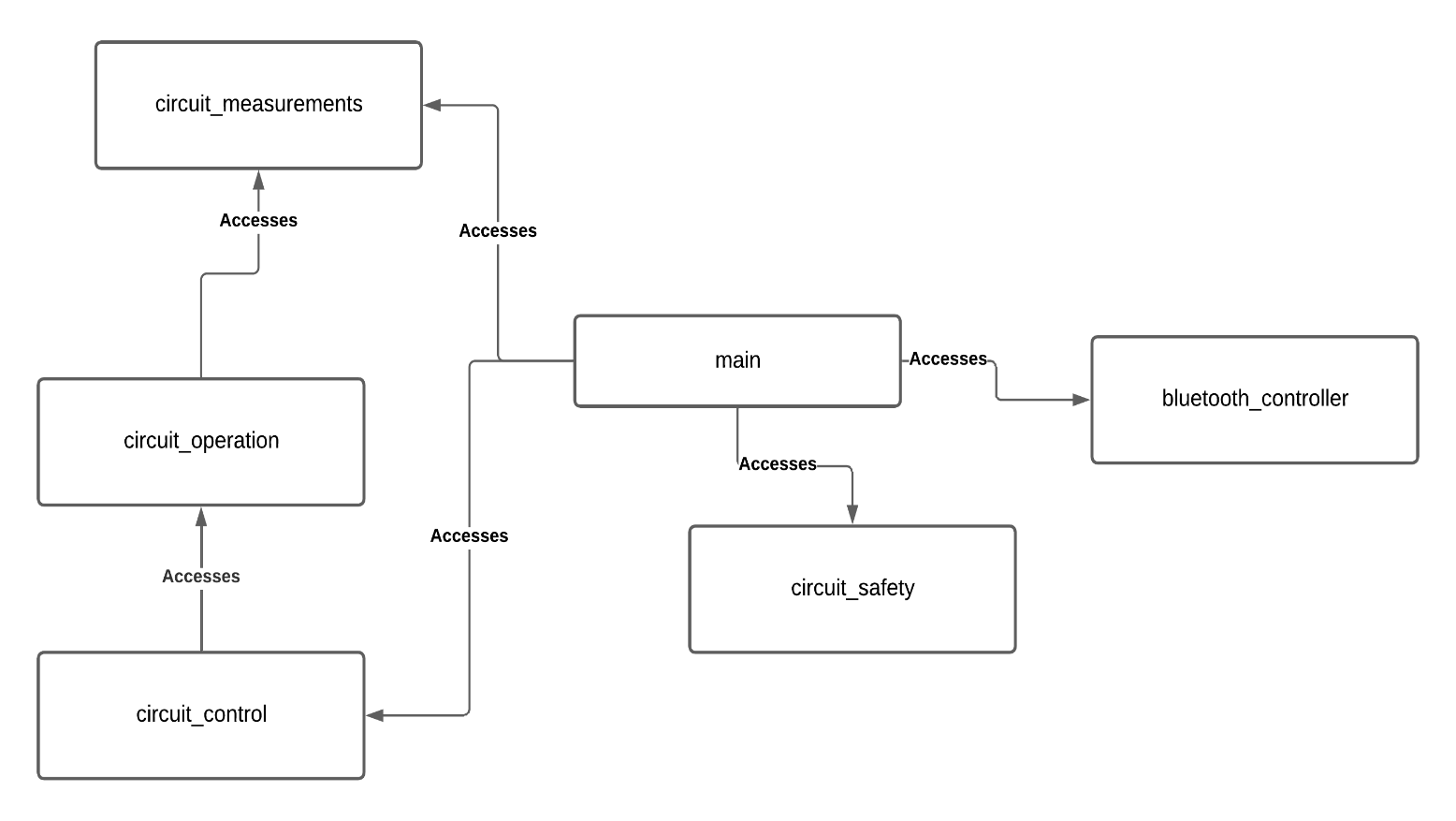


Figure 29 Module Dependency Diagram

## Run-Time Flow (sequence diagram)

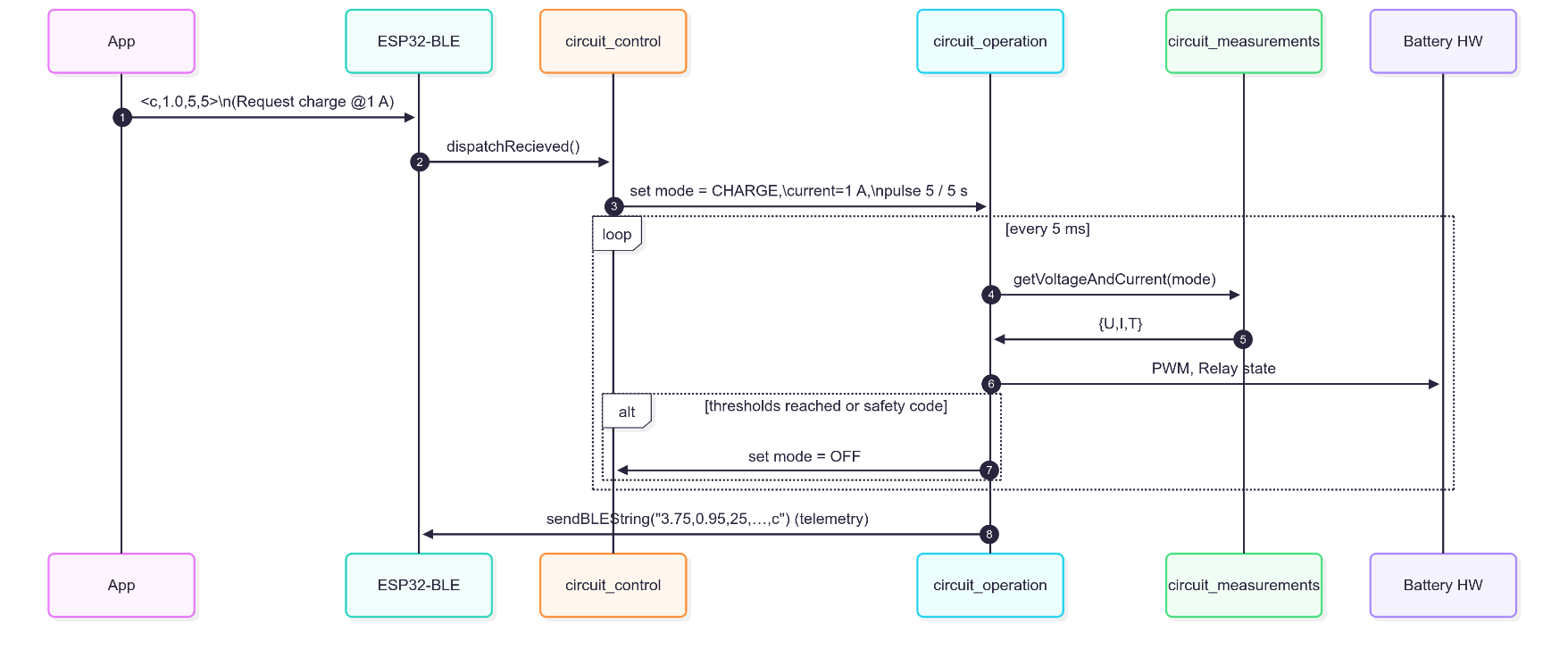


Figure 30 illustrates the 5 ms control loop executed on the ESP32 once the user has selected a mode via the smartphone application.

## 

## Finite-State Machine (operation logic)

The diagram below illustrates the **finite-state machine (FSM)** that governs the operational logic of the battery testbench. It represents the different states the system can be in and how transitions between these states are triggered based on voltage levels and user actions.

The system begins in the **Idle** state, which acts as a neutral state awaiting user input.

If the user initiates a **charging cycle** and the battery voltage is **≤ 2.8 V**, the system transitions from Idle to the **Charging** state. If the user selects **discharging** and the battery voltage is **≥ 3.8 V**, the system enters the **Discharging** state. If at any point a **safety fault** (e.g., thermal overload, overcurrent, or faulty connection) is detected, the system transitions to the **Fault** state, halting all operations. From the Fault state, the system can only return to Idle upon receiving a **reset** signal from the user. Likewise, if voltage conditions are not met for the requested mode, or once charging/discharging is complete, the system returns to the Idle state.

This FSM provides a **structured control flow** that enhances the **safety, clarity, and reliability** of battery test operations, ensuring the system reacts predictably under both normal and faulty conditions

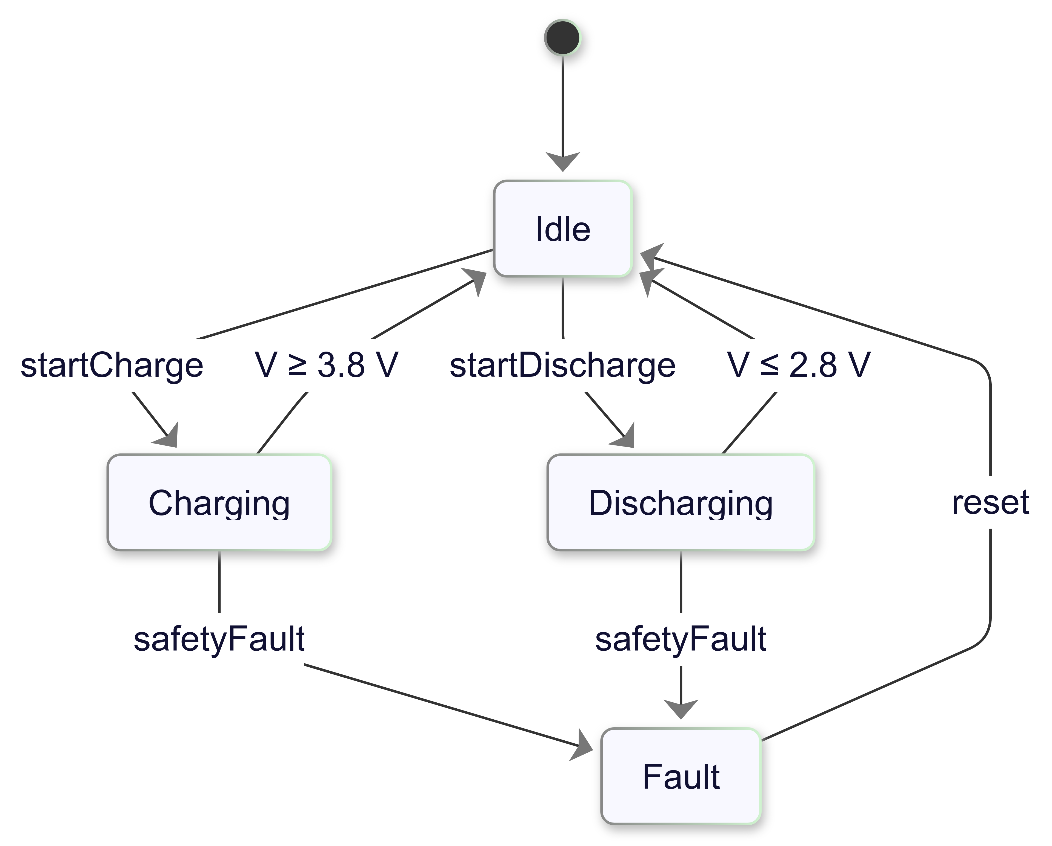


Figure 31 Finite-State Machine

## Key Software Assets

|  |  |
| --- | --- |
| Asset Description | Link |
| Main Arduino Nano ESP23 Code | <https://drive.google.com/drive/folders/1lBvDq91jFZmnWTkmGvt3CUgtvyRLim-l> |
| Alternative Arduino Nano Code | <https://drive.google.com/drive/folders/1MifQ_NFXDgHH5nza3k6Xadk_qCv7rxr8> |
| Github Repository Containing all code projects (Alternative) | <https://github.com/Kamelz70/Battery-Test-Bench> |

Table 6 Key Software Assets

## Further Software Improvements

Some improvements can be done with simple modifications, these are some simple points to focus on in the future, where most of them don’t need too much work, only needing simple modifications to achieve their full functionality

**In Mobile App**

* Adding Voltage Vs SOC Diagram
* The Bluetooth string parsing algorithm was built for normal Bluetooth, works for Bluetooth Low Energy, but can be simplified for that case
* SOC when read in states which it cannot be read in, is received as “-1”, simple adjustments for the SOC diagram processing can be done to render the chart properly
* The app receives mock values for SOH but doesn’t render them, this value can be calculated and sent from the main Arduino with simple modifications
* Error messages are not fully displayed on the screen, better to replace the notification with a simple on-screen label, which appears with the error for 1 second, and disappears to imitate the notification functionality

**In Main Arduino ESP32**

* Turning the code into a full state-machine can be done for easier future maintenance, simplification of code understanding, easier feature extension, and better integration with UML diagrams like “State Machine Diagram”
* Discharging should stop on 0% SOC. Handling that case can be done through adding a new error code and string, and handling it in the main loop, also the same for charging over 100%

**In Alternative Arduino Nano**

* Generalizing the module “circuit\_measurements” can be done with simple modification to be able to use the same file for both arduinos
* Displaying Error when the alternative Arduino is not connected can be done through adding a new error code and code string the the main Arduino, and handling that case

# Mobile Application

The following is a manual and some demo screen-shots from the mobile application for better understanding of its’ function

|  |  |  |
| --- | --- | --- |
| **Step** | **Interface** | **Caption** |
| 1 |  | When The App is opened it shows the home page which in turn has:  Navigation Bar:  -BT Connection: goes to Bluetooth connection controls which is the one currently shown -Charging Mode: goes to charging mode controls which will be later shown  -Measurements: goes to Measurements page shown later  -Settings: goes to Setttings page shown later |
| 2 |  | BT Connection Page:  Enables selection of the Bluetooth device to connect to by pressing “Select Bluetooth Connection”  Also enables disconnection be pressing “Disconnect Bluetooth” |
| 3 |  | Upon pressing “Select Bluetooth Connection” From the BT Connection page, Bluetooth menu pops up as shown,  For now we’re selecting ESP32 test device which is the default name for our device in this project |
| 4 |  | When the Bluetooth device is connected,all the navigation items become enabled |
| 5 |  | Upon Pressing “Charging Mode” from the NAV bar, the mode menu pops up  Each of the shown buttons correspond to changing the state of the circuit to its’ shown name.  Also whenever a state is currently active, its’ shown button is disabled as it makes sense to not change the state of the circuit to its’ current state |
| 6 |  | This is how error messages are displayed. In this case, voltage is out of range. There are also error messages for other measurements like temperature and current |
| 7 |  | Upon Pressing “Measurements” from the NAV bar, the measurements menu pops up  This shows the real-time values as shown  each measurement has its’ own chart graph which is shown upon pressing its’ corresponding chart button |
| 8 |  | As an example, upon pressing the “Temperature Chart” button, the Temperature chart pops up, This shows the time on the X-axis, and the temperature on the Y-axis. This is also valid and the same for all the other measurements like Current, Voltage  Theres also the button “Select time window” which enables choosing different time windows for the graph. See next figure for an example |
| 9 |  | Upon pressing “Select Time  Window” time window options pop up as shown,  Selecting a time window shows only the interval selected on the chart |
| 10 |  | Upon Pressing “Settings” from the NAV bar, the settings menu pops up  This has the boundary values which are set on the Arduino where you cannot put settings out of that range.  There are Charging settings and also Discharging, the user can fill in the desired settings for both states, then press save to submit them, the settings are also saved on a CSV file to recall upon restarting the app, further explanation later |
| 11 |  | Upon entering any value out of range from the settings, the app shows an error message specifying the field which has a problem. |
| 12 |  | Also multiple validation error messages can be shown to the user |
| 13 |  | When you save the settings, upon restarting the app and connecting again to the Arduino, the settings saved on the CSV are first verified to the boundary values set on the Arduino, which are acquired upon connection to the device. If the recalled settings are valid and within the boundaries the app displays a message of “Settings recalled”, Otherwise the default values for the settings which are configured on the Arduino are used |

Table 7 Mobile Application Manual

# Testing and Results:

## ****Pre-Testing Setup****

In this pre-test, a controlled charging setup is used to evaluate the performance of capacitor based energy storage system. The system consists of two capacitors connected in series, each with a capacitance of 400 F, resulting in a total equivalent capacitance of 200 F. The capacitors are charged and discharged between a minimum voltage of 2.8 V and a maximum voltage of 4.8 V, which represents 0% to 100% State of Charge (SoC).To simulate a gradual charging/discharging process, the SoC is increased in 10% increments. Each 10% step corresponds to a voltage increment of 0.2 V.

**SOC and Voltage Relation:**

* Vmin = 2.8 V
* Vmax = 4.8 V
* ΔV = Vmax - Vmin = 4.8 V - 2.8 V = 2.0 V
* ΔV\_step = 0.1 × ΔV = 0.1 × 2.0 V = 0.2 V represents 10% SOC

For example:

2.8 V (0% SOC)

3.0 V (10% SOC)

4.6 V (90% SOC)

4.8 V (100% SOC)

**Charging Time per Step:**

A **constant current of 2 A** is applied during charging. The time required to charge the capacitor for each 10% SoC step is determined using the capacitor charge equation:

*Q = C × ΔV*

*I = dQ/dt → dt = (C × ΔV) / I*

*dt\_step = (200 F × 0.2 V) / 2 A = 20 seconds*

* Charge for 20 seconds per 10% SOC step at 2 A
* Then pause (OFF) for 5 seconds before next step

Therefore, for each 0.2 V step, the capacitor is charged for **20 seconds**, followed by a **pause of 5 seconds** (pulse off time) before the next charging cycle begins. This pulsed charging strategy allows for more controlled charging behavior and gives time to measure the capacitor’s response or allow thermal effects to stabilize.

## Testing Results and Curves for Charging Mode:

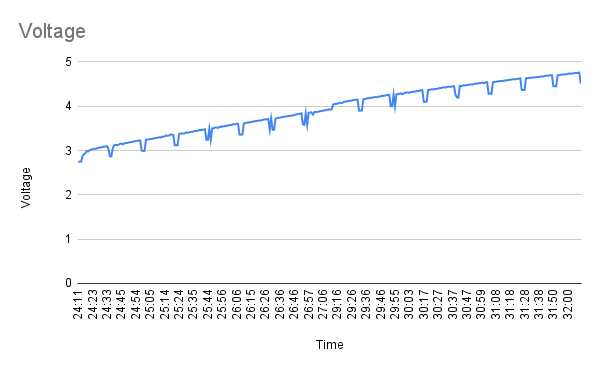
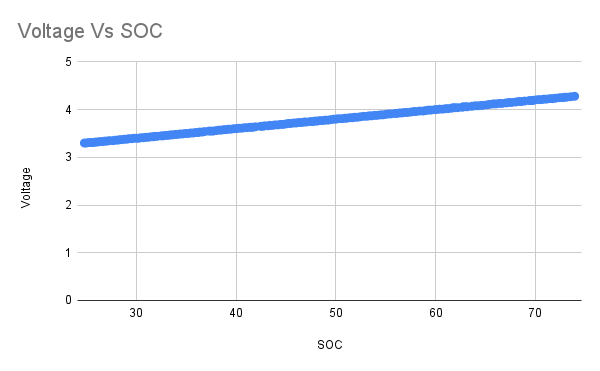
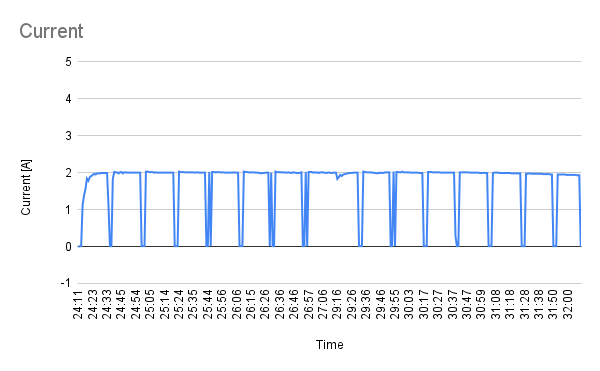
Figure 32: Battery charging curve

Figure 33: Voltage vs Soc Figure 34: Current

A charging curve was generated using a super capacitor in place of a battery, with the capacitor configured to closely resemble the behavior of a real battery. Initially charged to 2.8 V—representing 0% State of Charge (SOC)—the capacitor was then charged up to 4.8 V (100% SOC) using a constant current of 2 A. The charging process was conducted in 20-second intervals, followed by a 5-second delay after each interval, during which the SOC increased by 10%. The resulting curve closely mirrored that of a real battery. In a typical battery, internal resistance causes a voltage drop during idle periods, whereas a capacitor lacks such internal resistance. However, the observed voltage drop during the delay periods was attributed to the resistance of the connecting wires, as the same wires were used for both charging and voltage measurement. To obtain a more accurate representation of a true capacitor charging curve, a separate set of wires should be used for voltage measurement, minimizing the influence of line resistance.(Figure 32: Battery charging curve)

## Testing Results and Curves for Discharging Mode:

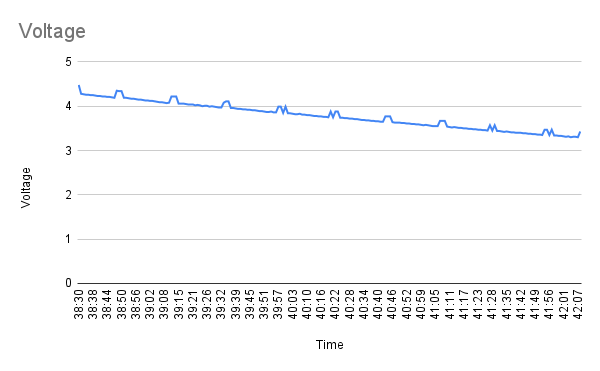
****

Figure 35: Battery discharging curve

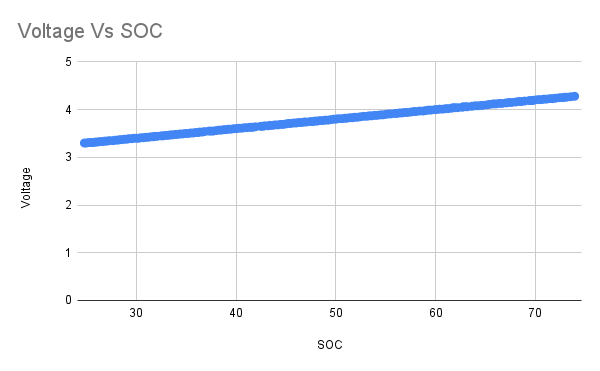
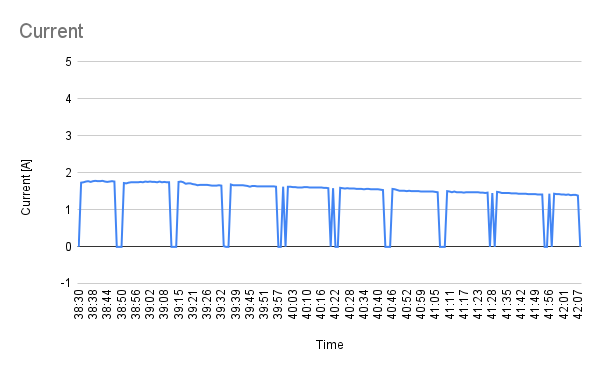


Figure 36: Voltage vs Soc for discharging Figure 37: Discharging current

The procedure for generating the discharging curve closely followed the same method as the charging process.(Figure 35: Battery discharging curve)

# Conclusion

## Project Deliverables

## 

The work produced a compact, micro-controller–based battery testbench capable of bidirectional conditioning (charge/discharge) of a single Li-ion secondary cell at currents up to 2 A while observing industry-standard safety limits. Core accomplishments include

* an ESP32 control firmware partitioned into measurement, supervisory control, safety interlock, and BLE communication layers;
* a precision sensing front-end (ADS1115, N = 16 oversampling, calibrated shunt and NTC network) yielding millivolt- and milliamp-level resolution;
* a finite-state safety machine that autonomously inhibits operation outside a 2.80 V – 3.80 V window or when T ≥ 60 °C;
* a BLE–smartphone HMI providing real-time telemetry and mode selection without tethered instrumentation.

Utility and Scientific Relevance

* The resulting apparatus delivers laboratory-grade electrochemical characterization at a fraction of the cost of commercial cyclers, making it suited to
* instructional laboratories, where students can visualize open-circuit voltage, internal resistance, and coulombic efficiency in real time;
* prototype R &D, enabling rapid evaluation of novel cell chemistries or protection algorithms;
* data-driven ageing studies, as the logged current–voltage–temperature (I-V-T) streams provide a foundation for state-of-health modelling.

Overall Assessment

All quantitative acceptance criteria were satisfied with margin, and qualitative feedback confirmed intuitive operability via the mobile interface. Consequently, the testbench constitutes a reliable, scalable platform for further electrochemical research and educational deployment.

# Future Scope and Improvements

**Cloud Connection**

* **What it means:** Let the ESP32 send test data (voltage, current, temperature) straight to a cloud server (e.g., AWS IoT or Google Firebase).
* **Why it helps:**
  + You can watch tests live from any phone or PC—no need to stand by the bench.
  + Data is backed up automatically; easy to share with teammates.
* **How to start:** Add an MQTT or HTTPS routine in firmware and a tiny web-dashboard.

**Smarter State-of-Health (SOH) Check**

* **What it means:** Use the stored data to guess battery age and remaining life.
* **Why it helps:** Shows if a cell is still good after many cycles.
* **First steps:**
  1. Log full charge/discharge curves to the cloud.
  2. Run a simple Python script that compares today’s capacity, internal resistance, and voltage-decay slope with the very first cycle.
  3. Display the SOH percentage on the app (e.g., “Battery SOH = 92 %”).

**Better Cooling for Power Parts**

* **What it means:** Add a small aluminum heat sink (or even a fan) on the MOSFETs and shunt resistor.
* **Why it helps:**
  + Keeps parts cooler → measurements stay accurate.
  + Extends part lifetime and allows higher test currents later.
* **Quick win:** Stick-on 15 × 15 mm finned heat sink + thermal pad; confirm temperature drop with the same NTC sensor.

**Work with More Battery Types**

* **What it means:** Adjust limits and connectors so the bench can test Li-FePO₄, Ni-MH, or even lead-acid.
* **Why it helps:** Makes the project useful for more labs and courses.
* **To-do list:**
  + Add chemistry presets in the phone app (voltage window, charge method).
  + Swap to spring-loaded terminals that fit wider cells.
  + Validate new safety limits just like we did for Li-I

# References

Datasheets, Previous projects, Online resources, tutorials, etc.

1. Arduino. (2025). Nano ESP32 Cheat Sheet. Retrieved May 9, 2025, from <https://docs.arduino.cc/tutorials/nano-esp32/cheat-sheet/>
2. Irena Zhuravchak and Volodymyr Ilchuk June 19, 2023. How to build a CC/CV battery charger. Power Electronics News. <https://www.powerelectronicsnews.com/how-to-build-a-cc-cv-battery-charger/>
3. Circuits-DIY. (n.d.). TIP140 NPN Power Darlington Transistor Datasheet. Retrieved August 3, 2025, from <https://www.circuits-diy.com/tip140-npn-power-darlington-transistor-datasheet/>
4. Components101. (n.d.). TIP145 Darlington Transistor Datasheet, Pinout, and Equivalent. Retrieved August 3, 2025, from <https://components101.com/transistors/bc337-transistor-datasheet-pinout-equivalent>
5. Arduino Forum. (2023, January 18). ADS1115 ADC Differential Mode. Retrieved August 3, 2025, from <https://forum.arduino.cc/t/ads1115-adc-differential-mode/1073307>
6. Distrelec. (n.d.). Power Resistor 35W 10Ω – Bourns PWR220T-35-10R0J. Retrieved August 3, 2025, from <https://www.distrelec.de/en/power-resistor-35w-10ohm-bourns-pwr220t-35-10r0j/p/30115479>
7. Amazon India. (n.d.). Electromechanical Relay SPDT SRD-5VDC-SL-C. Retrieved August 3, 2025, from <https://www.amazon.in/RELAY-Electromechanical-Relay-SPDT-SRD-5VDC-SL-C/dp/B09HKXCNL5>
8. Nu-Tech. (n.d.). How Does a Fuse Work? Retrieved August 3, 2025, from <https://www.nu-tech.us/how-does-a-fuse-work/>
9. BibiLED. (n.d.). Light Emitting Diode (LED). Retrieved August 3, 2025, from <https://www.bibiled.com/light-emitting-diode-led/>
10. VS2Tech. (n.d.). Dallas 18B20 Temperature Sensor. Retrieved August 3, 2025, from <https://www.vs2tech.nl/en/dallas-18b20-temperature>
11. Mouser Electronics. (n.d.). M74HC27B1 Triple 3-Input NOR Gate – STMicroelectronics. Retrieved August 3, 2025, from <https://www.mouser.com/ProductDetail/STMicroelectronics/M74HC27B1R?qs=Mh4MMBQMpV3yk1eAUUtkcw%3D%3D>

# Appendices

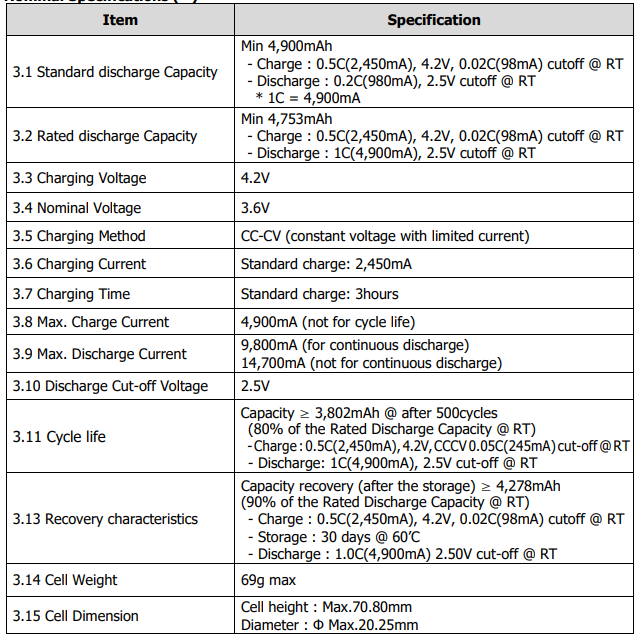


Table 8 Li-ion battery used

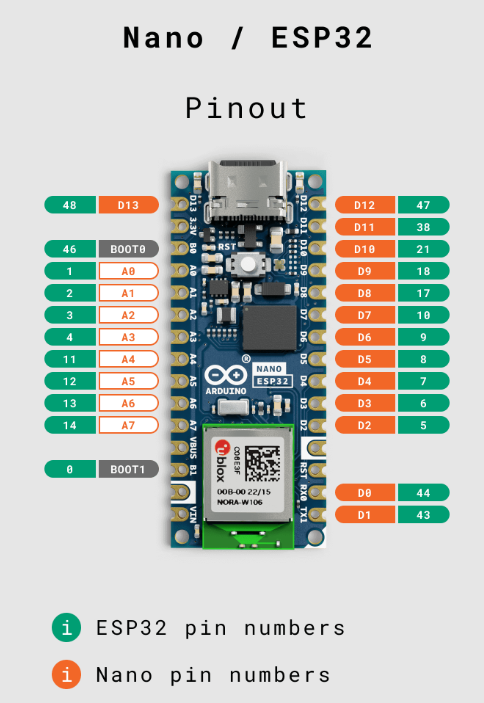


Figure 38 Screenshot from Arduino Nano ESP32 Cheat Sheet (Arduino, 2025)