Battery Management System for SoC Estimation

1. Introduction

This DBR documents the design basis for a compact Battery Management System (BMS) whose primary output is the *State-of-Charge* (SoC). The system estimates SoC from three primary inputs: pack **voltage**, pack **current**, and elapsed **time**. The controller is a Raspberry Pi Zero 2 W running CPython and scientific libraries (e.g., `numpy`) to implement Coulomb-counting and Thevenin-equivalent model corrections. Target system parameters: nominal voltage 12 V and peak continuous currents up to 150 A.

2. System Overview

2.1. Scope

The system provides real-time SoC estimation, on-board display of telemetry (OLED), local logging, and basic safety monitoring. It is intended as a prototyping and validation platform for SoC algorithms with the following constraints:

- Pack voltage: up to 25 V sensing headroom
- Continuous currents: up to 150 A (measured with a 200 A Hall-effect sensor)
- Host: Raspberry Pi Zero 2 W running Python 3.x

2.2. High-level architecture

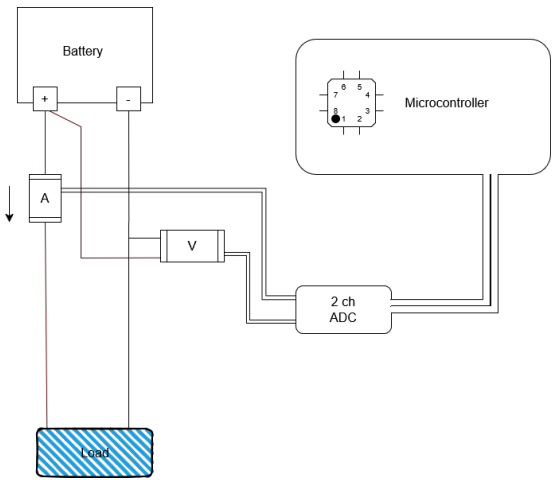

Figure 1: Schematic Diagram

3. Design Inputs and Constraints

- Battery chemistry: Lithium-based (assumed for OCV–SoC mapping)
- Nominal pack voltage: 12 V
- Maximum expected continuous current: 150 A
- Measurement interfaces: I^2C / SPI ADC for analog sensors
- Operating environment: laboratory bench (0 °C to 45 °C)

4. Components and Bill of Materials (BoM)

Item	Description / Notes	Quantity
Raspberry Pi Zero 2 W	Host controller, CPython, WiFi/Bluetooth (optional)	1
Micro SD card	16 GB–128 GB, class 10	1
OLED display	I^2C or SPI, 0.96ör 1.3¨	1
High-precision ADC	16–24 bit (e.g., ADS1115/ADS1256)	1
Voltage sensor	Voltage divider / module, $\leq 25\,\mathrm{V} rating$	1
Hall-effect current sensor	200 A range, analog output	1
5 V regulator	Buck converter for Pi power (from 12 V)	1
Protective elements	TVS, fuses, reverse-polarity diode	as needed
Wiring and connectors	AWG/busbars rated for 150 A	as needed
Enclosure	Non-conductive or properly insulated	1

Table 1: Bill of Materials (high-level).

5. Functional Requirements

5.1. Measurement

- Voltage measurement: 0 V–25 V (ADC input scaled appropriately)
- Current measurement: 0 A–200 A (Hall-effect sensor)
- Sampling rates: Voltage 1 Hz–10 Hz; Current 10 Hz–100 Hz
- ADC resolution: recommended 16 bit to 18 bit for low-noise SoC estimation

5.2. Computation and Estimation

Primary SoC estimation is via Coulomb counting with periodic correction using a Thevenin-equivalent model. The Coulomb counting integral is expressed as:

$$\mathrm{SoC}(t) = \mathrm{SoC}(t_0) + \frac{1}{C_{\mathrm{nom}}} \int_{t_0}^{t} I(\tau)\,\mathrm{d}\tau \tag{1}$$

where C_{nom} is nominal capacity in ampere-hours (Ah) and $I(\tau)$ is the instantaneous pack current (positive for charge).

5.3. Display and Logging

- Real-time display of SoC (%), instantaneous pack voltage (V), instantaneous current (A), and status (Charging / Discharging / Idle)
- Local logging of timestamped telemetry (CSV or JSON)
- Configurable logging interval; safe shutdown handling to prevent corruption

6. Hardware Selection Justification

6.1. Raspberry Pi Zero 2 W

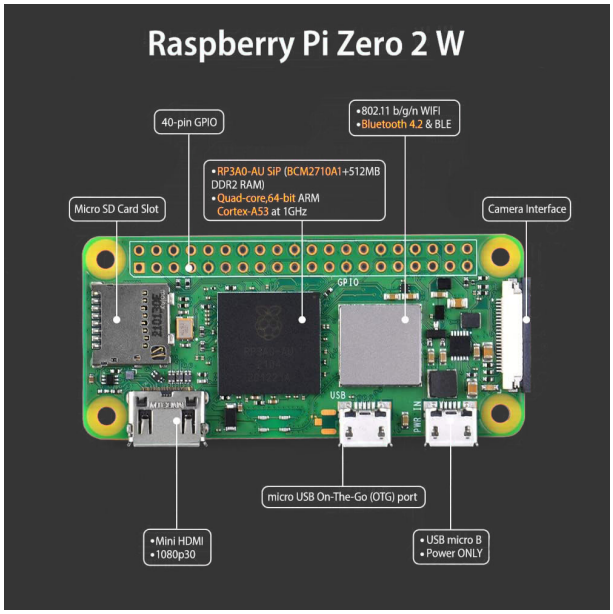

Figure 2: Raspberry Pi Zero 2w

The Raspberry Pi Zero 2 W is chosen for the following design-driven reasons:

1. **Library availability:** Full CPython environment allows direct use of `numpy`, `scipy` (if required), and domain-specific libraries (e.g., a Thevenin package), which are difficult to port or use on MicroPython-based microcontrollers.
2. **Computational headroom:** The quad-core ARM CPU can perform numerical integration, filtering, and model-based corrections at the sampling rates defined above without falling behind in realtime.
3. **Compactness:** Compared to larger Pi models, the Zero 2 W maintains a minimal physical footprint while providing required I/O (I^2C, SPI, UART, GPIO).
4. **Ecosystem and driver support:** Wide availability of ADC and display drivers reduces development time.

6.2. Sensor and ADC choices

Sensors were chosen to provide safe measurement margins (25 V voltage headroom and 200 A current sensor) and galvanic isolation (Hall-effect) for operator safety during high-current testing. ADC selection emphasizes resolution and low noise; an external ADC is recommended over Pi's internal peripherals.

7. Software and Algorithms

- Language: Python 3.x
- Required libraries: `numpy`, `math`, Thevenin-model library (pip-installable), I^2C/SPI driver packages
- Key modules: data-acquisition, SoC-estimator (Coulomb + Thevenin correction), display/GUI, logger, calibration utilities
- Calibration: zero-offset for current sensor, gain calibration for voltage divider, identification tests to extract Thevenin parameters

8. Safety, Protection and EMC Considerations

- Use TVS diodes and slow-blow fuses on the measurement inputs where appropriate.
- Provide reverse-polarity protection on the Pi power rail; use a stable 5 V buck converter rated for the Pi and peripherals.
- Ensure wiring and busbars are rated for 150 A; use appropriate crimping and thermal management.
- Follow grounding and shielding practices to minimize ADC measurement noise; consider differential measurement or isolated ADC front-ends if necessary.

9. Validation, Testing and Calibration Plan

1. **Hardware-in-the-loop tests:** Validate ADC scaling and noise at expected voltage/current ranges.
2. **Calibration:** Perform zero-current and gain calibration for current and voltage sensors using precision loads and calibrated references.
3. **Parameter identification:** Run pulse tests to fit Thevenin parameters (R_0, R_{th}, C_{th}).
4. **SoC validation:** Controlled charge/discharge cycles against a reference coulomb-meter or known capacity cell to verify SoC accuracy (target ±2–5%).
5. **Stress testing:** Long-duration logging and high-current operation to verify thermal and electrical stability.

10. Deliverables

- Prototype hardware and wiring diagrams
- Python codebase (DAQ, estimator, logger, UI)
- Calibration and validation report with datasets
- Final DBR and user guide for commissioning

11. Conclusion

The proposed BMS DBR outlines a compact, Python-based SoC estimation platform using Raspberry Pi Zero 2 W and external sensing hardware. The design balances measurement accuracy, safety, and developer productivity by leveraging CPython numerical libraries and robust sensor headroom for 12 V/150 A systems. The next step is procurement of specific ADC and sensor models followed by sensor-chain prototyping and parameter identification tests.

References

[1] Shubha Rao K., Nimisha T., Sai Rashmi B. J., Sanjana Alladwar, "Real Time Monitoring System for Lithium-Ion Cell Using IoT," BNM Institute of Technology, Bengaluru, Karnataka, India.

[2] Taufiq Alif Kurniawan, Aldyan Natajaya, Purnomo Sidi Priambodo, Gunawan Wibisono, "Real time monitoring state-of-charge battery using internal resistance measurements for remote applications," *Journal of Physics: Conference Series*, Volume 1528, 4th International Seminar on Sensors, Instrumentation, Measurement and Metrology, 14 November 2019, Padang, Indonesia. DOI: 10.1088/1742-6596/1528/1/012034.