BMS With CAN Communication – 2s2p Battery Pack Simulation

Candidate: K M Vaishnavi Gowda
Assignment: Q1 – BMS with CAN Communication
Organization: Arys Garage – November 2025 Assignment

1. Abstract

This project implements a complete Battery Management System (BMS) simulation for a 2s2p lithium-ion battery pack including electrical modeling, SOC estimation, thermal modeling, fault detection, balancing, cooling logic, and CAN communication. The objective is to build a realistic BMS flow that monitors per-cell parameters such as voltage, temperature, current, and state of charge (SOC), while identifying safety-critical conditions like over-voltage, under-voltage, over-temperature, under-temperature, imbalance, and over-current.

The system also simulates real-time BMS behavior under a dynamic load profile, using coulomb counting combined with voltage-based correction to achieve stable SOC estimation. A thermal model tracks cell temperature rise due to internal resistance losses. Fault actions include cooling activation and shutdown during thermal emergencies.

CAN-like messages are encoded and logged into CSV files to represent pack-level and cell-level telemetry. The simulation produces graphs for voltage, SOC, and temperature evolution over time. Overall, this project demonstrates a full workflow of a BMS with safety logic, pack simulation, CAN logging, and visualization, aligned with automotive EV BMS requirements.

2. Tools & AI Usage

Tools Used

- **Python 3.12**
- **NumPy** – mathematical modeling
- **Matplotlib** – visualization
- **Pandas** – CAN log viewing
- **python-can** style logging (simulated)
- **Git** + GitHub for version control

AI Usage (As required by assignment)

AI tools were used to:

- Generate base code structure for the BMS components
- Debug module import errors and improve architecture
- Create the battery thermal model and SOC correction logic

3. Design & Methodology

3.1 Battery Pack Topology – 2s2p

- Four Li-ion cells
- Two cells in parallel → two parallel groups
- Two groups in series → pack voltage = $V_1 + V_2$
- Each parallel cell receives **I/2** current

3.2 Cell Model

Each cell is represented using:

- **Open Circuit Voltage (OCV) vs SOC curve** (linear approximation 3.0–4.2 V)
- **Internal resistance model:**
- V=OCV(SOC)−I · Rinternal
- **Thermal model:**
  Heat generated by I2RI^2RI2R and cooled by convection.

3.3 SOC Estimation

SOC is computed using:

1. **Coulomb counting** (primary)
2. **Voltage-based correction** to improve long-term drift
3. Per-cell SOC maintained independently

3.4 Fault Detection Logic

Faults monitored:

- Over-voltage (>4.25 V)
- Under-voltage (<2.7 V)
- Over-temperature (>60°C)
- Under-temperature (<0°C)
- Over-current (>10 A)
- Cell imbalance (>0.02 SOC difference)

A **persistence counter** ensures faults only trigger after 3 consecutive detections.

3.5 Cooling & Shutdown Logic

Cooling states:

- **COOLING_OFF** → normal
- **COOLING_ON** → temps >45°C
- **EMERGENCY_SHUTDOWN** → temps >60°C

3.6 Balancing

- Simple passive balancing
- When a cell SOC exceeds pack average by 0.02 → BLEED

3.7 CAN Simulation

- CAN messages encoded every 5 seconds
- Includes pack voltage, average SOC, max temperature
- Per-cell telemetry: voltage, SOC, temperature
- Logs saved under:
- results/can_logs/

4. Implementation Details

Directory structure:

src/

│ ─── battery_model.py

│ ─── soc_estimator.py

│ ─── bms_controller.py

│ ─── can_simulator.py

│ ─── simulate.py

│ ─── utils.py

Key Modules

battery_model.py

- Defines Cell and Pack2s2p
- Implements electrical + thermal behaviors

soc_estimator.py

- Performs coulomb counting
- Applies voltage correction to reduce drift

bms_controller.py

- Fault detection logic
- Balancing
- Cooling & shutdown logic

can_simulator.py

- CSV-based CAN logger
- Encodes compact BMS messages

simulate.py

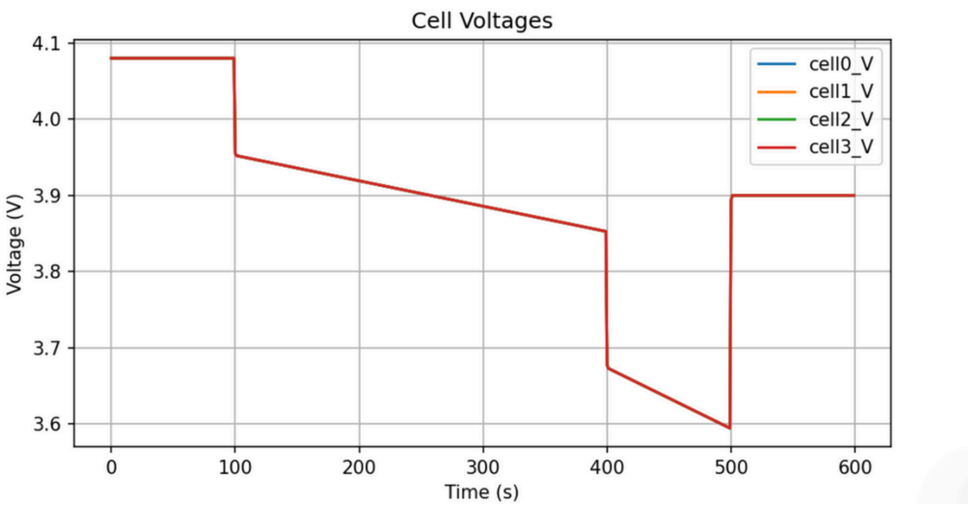
- Integrates all modules
- Runs load profile simulation
- Generates plots and CAN logs

Load profile:

- 0–100s → idle
- 100–400s → 5A discharge
- 400–500s → 12A over-current stress
- 500–600s → rest

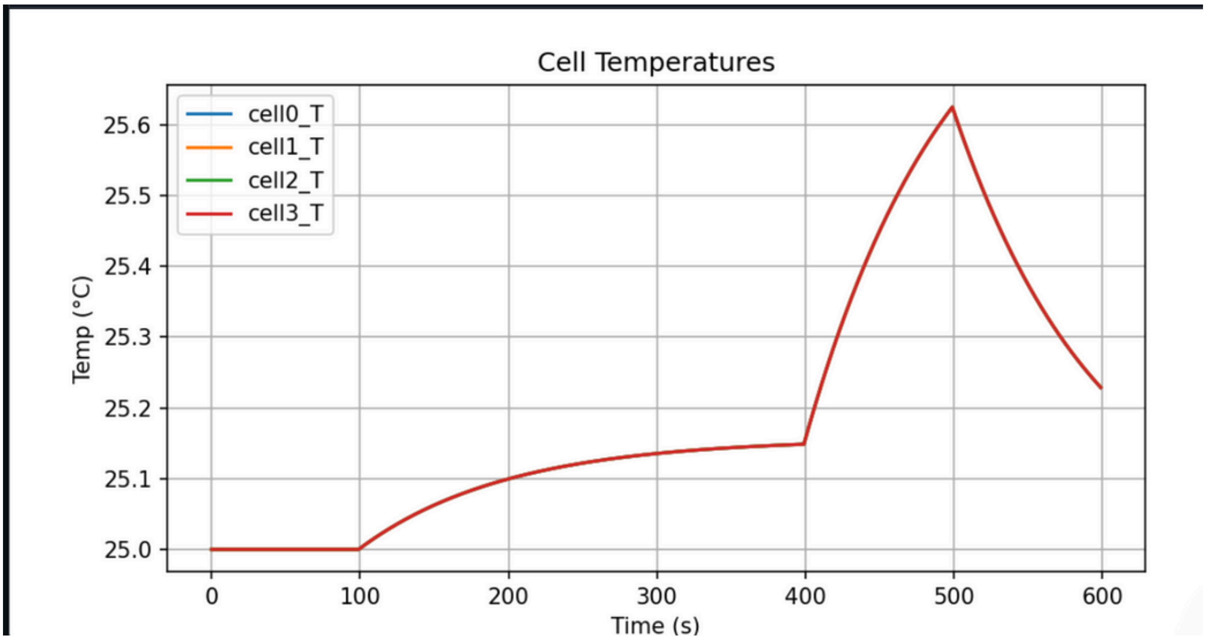
5. Results

5.1 Cell Voltage Plot



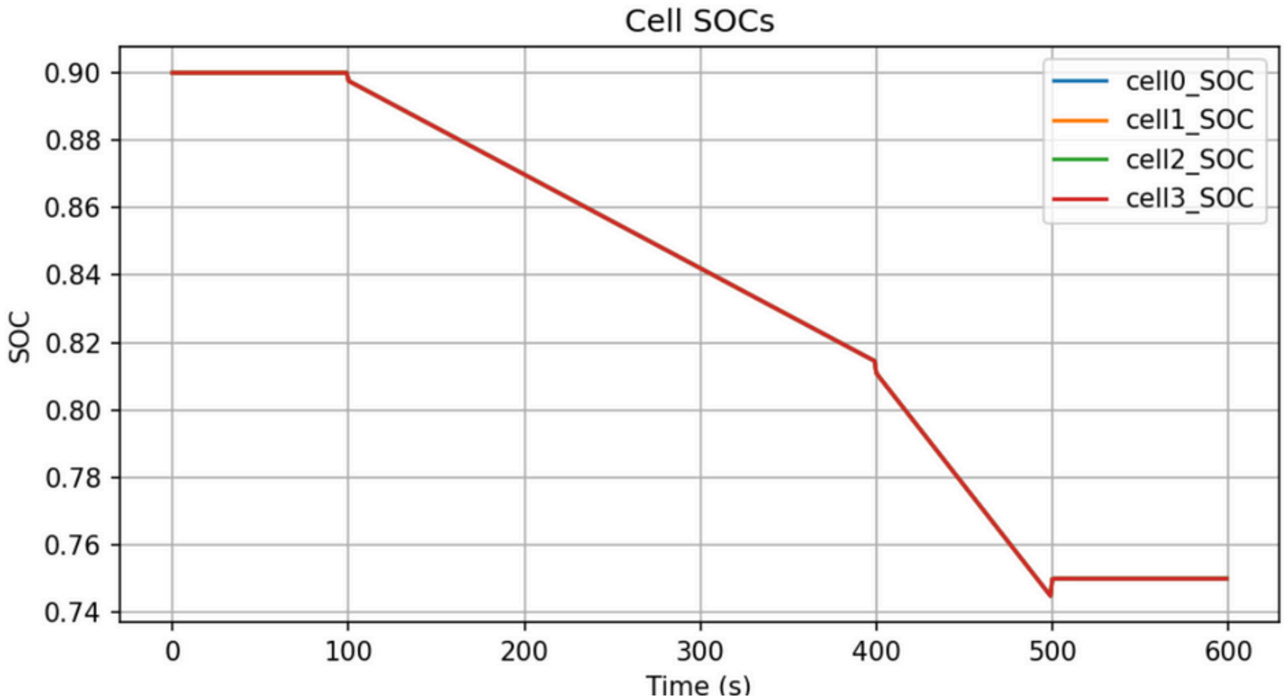
5.2 Temperature Plot

Temperature increases during high-current phases.
Cooling activates after 45°C.
Emergency shutdown if >60°C.



5.3 SOC Plot

SOC decreases linearly during discharge.
Voltage correction stabilizes SOC at rest.



5.4 CAN Log Output

timestamp	id	pack_volt	avg_soc	max_temp	v0	soc0	t0	v1	soc1	t1	v2	soc2	t2	v3	soc3	t3
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25
2025-11-2	0x100	8.16	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25	4.08	0.9	25

6. Challenges & Limitations

- OCV curve is a simplified linear model (not chemistry-accurate)
- Thermal model is lumped and does not model conduction between cells
- Balancing is passive; active balancing not implemented
- Real CAN bus hardware not used (simulation via CSV logs)
- Over-current event generates large heat due to simplified R_internal value

Possible improvements:

- Use nonlinear OCV-SOC curve
- Implement Kalman Filter SOC estimation
- Add pack-level contactor simulation
- Integrate real CAN hardware using python-can

8. References

- Lithium-ion Cell Modeling Literature
- python-can documentation
- NumPy and Matplotlib documentation