Int. J. Sci. R. Tech., 2026 3(11)
A Multidisciplinary peer-reviewed Journal
www.ijsrtjournal.com [ISSN: 2394-7063]

Original Article

Smart Battery Management System Using AI and Can Protocol

Aishwarya Mali*, Divya Galbale, Vedang Jadhav, Bhausaheb Eknath. Shinde

Department of Electronics and Telecommunication Engineering, Dhole Patil College of Engineering, Pune, Maharashtra, India.

ABSTRACT

Lithium-ion batteries power electric vehicles, energy storage and portable electronics, but they are sensitive to overvoltage, deep discharge, overcurrent and elevated temperature. This paper presents the design and implementation of a low-cost smart Battery Management System (BMS) for a 3S lithium-ion pack built around the ESP32 microcontroller. The system measures pack voltage, current and temperature, estimates State of Charge (SOC) using [method: Coulomb counting / AI model], detects overvoltage, undervoltage, overcurrent and overtemperature faults, and isolates the load through a relay. Measurements and fault states are shared with other nodes over a Controller Area Network (CAN) bus using an MCP2515 controller, and a dashboard displays live data. An AI component is used for [SOC/SOH estimation / anomaly detection] to improve on fixed-threshold decisions. Experimental results show [SOC error of X %, fault response time of Y ms, CAN message rate of Z Hz]. The system offers an inexpensive and extensible platform for battery monitoring and protection.

Keywords: Battery Management System, ESP32, CAN bus, State of Charge, lithium-ion, artificial intelligence, embedded systems

INTRODUCTION

The growing use of lithium-ion batteries in electric vehicles, renewable energy storage and backup power has made battery safety and lifetime a central engineering concern. Operating a cell outside its safe voltage, current or temperature window accelerates degradation and can lead to thermal runaway. A BMS is therefore required to monitor the pack, estimate its state, and protect it from unsafe conditions.

Conventional low-cost BMS designs rely on fixed thresholds and monitor a limited set of parameters, with little or no communication to other system components. Recent work moves toward data-driven state estimation, IoT monitoring and automotive communication such as CAN. This work combines these directions in a low-cost platform.

The main contributions of this paper are:

1. A hardware design of a 3S Li-ion BMS using an ESP32, resistive voltage sensing, an ACS712 current sensor and an LM35 temperature sensor.
2. Relay-based automatic load isolation on overvoltage, undervoltage, overcurrent and overtemperature faults.
3. CAN bus communication of battery data using an MCP2515 controller.
4. An AI-based [SOC/SOH estimation or anomaly detection] module: [one-line description of what you built].
5. An experimental evaluation of [accuracy, response time, communication reliability].

The rest of the paper is organized as follows. Section II reviews related work, Section III describes the system architecture, Section IV covers implementation, Section V presents results, and Section VI concludes.

II. Related Work

Accurate SOC estimation is a core BMS function, and deep-learning methods are increasingly replacing traditional estimators [1]. Kumar et al. proposed a five-layer AI-assisted IoT BMS integrating a battery model, AI engine, PID controller, Peltier cooling and cloud monitoring, targeting SOC/SOH prediction and thermal management during fast charging [2].

Relevant conflicts of interest/financial disclosures: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Mulpuri et al. presented a hybrid edge–cloud BMS for second-life EV batteries with monitoring, cell balancing, fault detection and AI analytics [3].

Low-cost IoT designs have also been reported. Rohith et al. built a battery charging and monitoring system using NodeMCU/ESP8266, TP4056, ACS712 and an NTC thermistor, with a web dashboard [4]. Jaibhai et al. proposed a microcontroller-based EV BMS focused on overcharge, over-discharge, thermal runaway and fire protection using relay disconnection [6].

On the automotive side, Grini et al. discuss modern BMS architectures featuring edge-processing controllers, wireless BMS and CAN-FD/Ethernet communication [5]. Saidulu et al. developed a hardware–software BMS with cell-level monitoring, active/passive balancing, CAN communication and an Extended Kalman Filter for state estimation [9]. For health prediction, Alhazmi combined SVM-RFE feature selection with a deep recurrent model for SOH and remaining-useful-life estimation [7], and Choudhari et al. reviewed EV BMS challenges including SOC/SOH estimation, thermal management and cybersecurity [8].

Existing low-cost systems often monitor few parameters, lack fast automatic cutoff, or omit vehicle-grade communication, while AI-capable systems are typically simulation-based or computationally heavy. This work targets the gap between them: a low-cost, hardware-implemented BMS that adds CAN communication and an embedded AI function. [State precisely how your system differs from [2], [4] and [9].]

III. System Architecture

The system comprises a 3S lithium-ion pack (nominal 11.1 V, 12.6 V fully charged), an LM2596 buck converter producing a regulated 5 V rail, sensing blocks for voltage, current and temperature, an ESP32 controller, an MCP2515 CAN module, a relay, a buzzer and LED indicators, and a fan acting as the load. [Insert Fig. 1: system block diagram.]

A. Power Supply

The LM2596 steps the pack voltage down to 5 V to power the ESP32, sensors, relay and display, remaining stable as the pack voltage varies between roughly 9 V and 12.6 V.

B. Voltage Sensing

A divider of R1 = 100 kΩ and R2 = 10 kΩ scales the pack voltage by 1/11, so a 12.6 V pack produces about 1.15 V at the ESP32 ADC, within the 3.3 V input limit. The pack voltage is recovered as V_batt = V_adc × (R1 + R2) / R2. [State ADC calibration method; the ESP32 ADC is non-linear.]

C. Current Sensing

An ACS712 Hall-effect sensor in series with the load gives an analog output proportional to current. [Specify the variant (5 A/20 A/30 A), sensitivity, and how its output is scaled to the ESP32's 3.3 V ADC range.]

D. Temperature Sensing

An LM35 sensor attached to the pack provides 10 mV/°C output read by the ADC.

E. Controller and Protection

The ESP32 samples all channels, computes SOC, and compares readings against thresholds. On a fault it opens the relay, sounds the buzzer and flags the condition. Table I lists the thresholds.

Table I: Protection Thresholds

Parameter	Threshold
Overvoltage	[e.g. 12.6 V]
Undervoltage	[e.g. 9.0 V]
Overcurrent	[value] A
Overtemperature	[value] °C

F. CAN Communication

The MCP2515 provides CAN framing, filtering and error handling, and a transceiver converts logic levels to differential CANH/CANL. The ESP32 communicates with it over SPI and transmits battery data at [bit rate, e.g. 500 kbit/s]. **Table II** defines the message set.

Table II: Can Message Format

CAN ID	Content	Bytes	Rate
[0x...]	Voltage, current	[4]	[10hz]
[0x...]	Temperature, SOC	[3]	[10hz]
[0x...]	Fault flags	[1]	[1hz]

G. AI Module

[Describe exactly what the AI does: the task (SOC estimation, SOH estimation, anomaly or fault prediction), model type and size, input features, training data (own measurements or a public dataset such as NASA or CALCE), where it runs (ESP32 via TensorFlow Lite Micro, or on a PC/cloud), and the inference time.]

IV. Implementation

The circuit was built around the ESP32 with the sensors, relay module, 16×2 I2C LCD, buzzer, LED and MCP2515 connected as shown in [Fig. 2: circuit diagram] and a PCB was designed [Fig. 3: PCB layout]. Firmware is written in [Arduino C++/ESP-IDF]. The control loop executes the following steps each cycle: sample voltage, current and temperature; filter the samples; update SOC; run the AI module; evaluate fault conditions; drive the relay and indicators; transmit CAN frames; update the display and dashboard. SOC is estimated by [Coulomb counting with voltage correction / the AI model]. [Add a flowchart.]

RESULTS AND DISCUSSION

[This section is mandatory for publication. Suggested experiments:]

1. **Sensor accuracy:** compare ESP32 voltage, current and temperature readings with a calibrated multimeter and thermometer. Report mean error. [Table III]
2. **SOC estimation:** compare estimated SOC with a reference over a full discharge, and report MAE/RMSE for the threshold-only baseline versus the AI method. [Fig. 4]
3. **Fault response:** inject overvoltage, undervoltage, overcurrent and overtemperature conditions and report the time from fault to relay opening. [Table IV]
4. **CAN performance:** report message rate, latency, and frame loss measured with a second CAN node or analyzer. [Table V]
5. **Cost:** bill of materials compared with commercial BMS modules.

VI. Conclusion and Future Work

This paper presented a low-cost smart BMS using the ESP32 that monitors voltage, current and temperature, protects a 3S Li-ion pack through relay isolation, and exchanges data over CAN with an AI-based [function]. Results show [key quantitative findings]. Future work includes per-cell voltage monitoring and balancing, SOH and remaining-useful-life prediction, wireless or cloud monitoring, CAN-FD support, and testing on a larger pack or EV platform.

REFERENCES

1. *(Verify every entry against the original publication and add journal/conference name, volume, pages and DOI before submitting.)*
2. J. Yao, [title], 2026. [Complete details; study on deep-learning SOC estimation.]
3. Kumar et al., "AI-Assisted IoT Smart Battery Management System with Peltier-Based Thermal Control," 2026.
4. Mulpuri et al., "Cloud-Based Battery Management System for Energy Storage Applications: Enabling Second-Life Batteries for Smart Mobility and Smart Homes," 2025.
5. Rohith et al., "Design and Implementation of an IoT-Enabled Smart Battery Charging and Monitoring System," 2025.
6. Grini et al., "Designing Safer, Smarter and More Connected Battery Management Systems," 2025.
7. Jaibhai et al., "Electric Vehicle BMS (Battery Management System) – Charge Monitoring and Fire Protection," 2024.
8. Alhazmi, "State of Health Prediction in Electric Vehicle Batteries Using a Deep Learning Model," 2024.
9. Choudhari et al., "Battery Management Systems for Electric Vehicles: A Review of State-of-the-Art Technologies," 2025.

10. Saidulu et al., "A Smart Battery Management System (BMS) Development for Electric Vehicles to Improve Energy and Power Management Structure," 2024. S.No Name of Author Designation Department 1 Vedang Santosh Jadhav Student E&TC Engineering 2 Divya Mohan Galbale Student E&TC Engineering 3 Aishwarya Ganesh Mali Student E&TC Engineering 4 Prof.Dr.Bhausaheb Eknath Shinde Dean Research & Development, Assistant Professor E&TC Engineering

HOW TO CITE: Aishwarya Mali, Divya Galbale, Vedang Jadhav, Bhausaheb Eknath. Shinde, Smart Battery Management System Using AI and Can Protocol, Int. J. Sci. R. Tech., 2026, 3 (11), 92-95. https://doi.org/10.5281/zenodo.23913757