International Journal of Creative and Open Research in Engineering and Management
ISSN: 3108-1754 (Online)
Volume 02 Issue 09 September-2026 | Impact Factor: 3.5

IOT-Enabled Battery Management Systems for Real-Time Monitoring and Protection: A Review of ESP32-Based Architectures

Pritesh Mhaiskar[1], Rushikesh A. Sonarkhan[2], Pratik G. Bhujbale[3], Akhil S. Pazare[4], Diksha U. Rathod[5], Sayali P. Lagdutkar[6]

[1–6] Department of Electrical Engineering, Tulsiramji Gaikwad Patil College of Engineering and Technology, Nagpur, Maharashtra, India

Guide Email: priteshmhaiskar1@gmail.com

How to Cite this Article:

Mhaiskar, P., Sonarkhan, R. A., Bhujbale, P. G., Pazare, A. S., Rathod, D. U. & Lagdutkar, S. P. (2026). IOT-Enabled Battery Management Systems for Real-Time Monitoring and Protection: A Review of ESP32-Based Architectures. International Journal of Creative and Open Research in Engineering and Management, 02(9), 1-9. https://doi.org/10.55041/ijcope.v2i9.147

https://doi.org/10.55041/ijcope.v2i9.147

Abstract

Battery Management Systems (BMSs) are essential for the safe, reliable, and efficient operation of rechargeable batteries because they supervise electrical, thermal, and operational conditions and support functions such as state-of-charge (SOC) estimation, protection, cell balancing, and diagnostics. The rapid adoption of Internet of Things (IoT) technologies has extended conventional BMS functionality by enabling wireless data transmission, cloud visualization, remote supervision, and event notification. This paper presents a systematic review of BMS architectures with emphasis on low-cost IoT-enabled implementations based on ESP32-class microcontrollers. The review covers voltage, current, and temperature acquisition; SOC and state-of-health (SOH) estimation; thermal protection; cell balancing; communication; cloud monitoring; and fault handling. Particular attention is given to an ESP32-based architecture using an INA219 voltage/current sensor, a DS18B20 temperature sensor, a local OLED display, a relay, a buzzer, and a Wi-Fi cloud interface. The reviewed literature indicates that ESP32-based systems can provide an economical platform for real-time monitoring, but a low-cost monitoring prototype should not be considered equivalent to a full multi-cell automotive BMS unless cell-level measurement, balancing, validated state estimation, robust protection circuitry, and safety-oriented communication are implemented. The paper identifies research gaps in accurate SOC/SOH estimation, multi-cell scalability, cybersecurity, fault diagnosis, and validation under dynamic operating conditions, and discusses directions toward predictive and intelligent BMSs.

Keywords— Battery Management System; Internet of Things; ESP32; State of Charge; Cloud Monitoring; Battery Protection.

I. INTRODUCTION

Rechargeable batteries are increasingly used in portable electronics, backup power systems, renewable-energy storage, and electric mobility. Their useful operation depends not only on electrochemical characteristics but also on appropriate monitoring and control. A BMS supervises important variables such as cell voltage, pack current, and temperature and uses these measurements to protect the battery and estimate operational states. Earlier BMS literature emphasizes the need for accurate SOC and SOH estimation together with protection against deep discharge, overcharge, overcurrent, and unsafe thermal conditions [1].

Modern BMS research has expanded from basic protection toward intelligent, connected, and data-driven systems. Contemporary architectures may include state estimation, cell balancing, thermal

management, fault diagnosis, data logging, and communication with higher-level controllers. Reviews have also identified IoT and cloud-based BMS architectures as important directions for improving remote supervision and battery safety [2], [3].

The supplied project material describes a low-cost IoT-enabled BMS centered on an ESP32 for a nominal 12-V battery. It combines INA219 voltage/current sensing, DS18B20 temperature sensing, local OLED indication, Wi-Fi communication, cloud visualization, relay isolation, and audible alarm generation. The system calculates derived parameters including SOC, power, and cumulative energy and can disconnect the battery when configured limits are exceeded. The original prototype document reports functional bench testing of monitoring and fault-trigger behavior but does not provide a statistically validated accuracy study or a complete multi-cell balancing implementation [4].

The objective of this review is therefore not to claim that the prototype replaces a commercial BMS, but to position the architecture within the broader BMS literature, compare its functions with established BMS requirements, identify limitations, and define a technically appropriate path toward a more capable IoT-enabled BMS.

II. FUNDAMENTALS AND FUNCTIONS OF A BMS

A. Battery Parameter Monitoring

Voltage, current, and temperature are fundamental measurable variables in a BMS. Voltage monitoring identifies abnormal cell or pack conditions, current measurement supports charge/discharge supervision and energy calculations, and temperature monitoring helps identify thermal stress. Modern BMS reviews consistently treat these measurements as the foundation for protection and state estimation [1], [2].

B. State-of-Charge Estimation

SOC represents the available charge relative to the usable battery capacity. A simple voltage-based estimate can be implemented at low computational cost, whereas current integration (coulomb counting) uses measured current over time. More advanced methods employ equivalent-circuit models, observers, Kalman filters, and machine-learning approaches. OCV-based methods require appropriate battery characterization because temperature, aging, cell variation, and modeling assumptions influence the voltage-SOC relationship [5].

For a simplified coulomb-counting implementation, SOC can be expressed conceptually as $SOC(t) = SOC(t_0) - (1/C_n)\int I(t)dt$, with the sign convention selected according to charging or discharging current. In practical BMS implementations, sensor offset, capacity variation, temperature dependence, and accumulated integration error require calibration and periodic correction.

C. State-of-Health Estimation

SOH describes the battery's condition relative to a reference condition, commonly associated with capacity or resistance degradation. Unlike instantaneous voltage/current monitoring, SOH estimation generally requires historical data and a model or diagnostic algorithm. The supplied prototype estimates SOC but does not report a validated SOH algorithm; therefore SOH should be considered a future enhancement rather than a demonstrated function [4].

D. Thermal Management and Protection

Temperature is a critical safety variable. BMSs monitor temperature and may activate cooling, heating, charging restrictions, or emergency disconnection. Literature identifies thermal management as an important subsystem because excessive temperature can accelerate degradation and create safety risks [2]. A simple prototype can use a temperature threshold to trigger a relay and alarm, but a complete EV-grade BMS normally requires distributed sensing, thermal control, fault diagnosis, and coordinated protection.

E. Cell Balancing

Cell balancing is required for series-connected battery packs because individual cells do not remain perfectly identical during operation. Passive balancing dissipates excess energy, whereas active methods transfer energy between cells. The choice involves trade-offs among cost, efficiency, circuit complexity, and control requirements [6]. The reviewed ESP32 prototype is centered on a 12-V battery and does not implement per-cell balancing; consequently, it should not be represented as a complete multi-cell lithium-ion BMS.

III. IoT-BASED BATTERY MONITORING

IoT integration adds a communication layer to the measurement and protection functions of a BMS. A typical architecture consists of sensors, a microcontroller, a communication interface, a cloud or local server, and a user interface. Data can be transmitted through Wi-Fi or other wireless technologies to support remote visualization and alerts.

Recent IEEE literature demonstrates the continuing movement toward connected BMS architectures. An IEEE 2025 SmartBMS publication describes IoT-based remote supervision of battery assets and includes monitoring of voltage, current, temperature, SOC, SOH, diagnostics, and fault protection [7]. Another IEEE study describes wireless real-time BMS data transmission using an ESP32 and cloud server, emphasizing remote monitoring of battery parameters and safety functions [8].

Cloud connectivity does not itself improve measurement accuracy or battery protection. Instead, it extends visibility and enables historical data analysis, alarms, and remote decision support. Protection-critical functions should therefore remain locally available even when Wi-Fi or cloud services are unavailable.

IV. REVIEW OF ESP32-BASED AND RELATED APPROACHES

The supplied project references an ESP32-based EV battery monitoring study, a Wi-Fi-enabled LiFePO4 monitoring implementation, an IEEE P2668-compatible smart-BMS evaluation strategy, an IoT-enhanced STM32-based system, and low-cost ESP32 data-acquisition work [4]. These studies collectively illustrate a progression from parameter acquisition toward connected monitoring and intelligent battery management. Table I summarizes the principal characteristics of these approaches.

Table I: Comparison of Reviewed BMS Approaches

Approach	Measurements / functions	Connectivity	Main contribution / limitation
Conventional BMS	Voltage, current, temperature; protection; SOC/SOH; balancing	Local / vehicle network	Strong local control; remote visibility depends on communication layer [1], [2].
ESP32 IoT monitoring	Voltage, current, temperature; SOC/monitoring	Wi-Fi / cloud	Low-cost and flexible; validation and multi-cell capability depend on implementation [4], [8].
Cloud-connected cell monitoring	Individual cell status; SOC estimation; display, cloud	Wi-Fi / IoT	Enables remote cell-level visibility; requires scalable cell acquisition [9].
Smart / IoT BMS	Monitoring, estimation, diagnostics, remote supervision	IoT / cloud	Broader functionality but greater software, communication, and cybersecurity requirements [7].
Proposed reviewed architecture	Voltage/current, temperature, SOC, power, energy, alarms, relay	ESP32 Wi-Fi + cloud	Low-cost prototype architecture; no demonstrated per-cell balancing or validated SOH [4].

The comparison shows that the ESP32 is attractive as a controller because it combines processing capability, wireless connectivity, and a broad development ecosystem. However, the microcontroller alone does not define BMS completeness. The sensing topology, isolation, protection hardware, estimation algorithms, balancing circuitry, communication reliability, and validation methodology determine the actual capability of the system.

V. REVIEWED ESP32-BASED IoT BMS ARCHITECTURE

The project architecture uses the ESP32 as the central controller. The INA219 measures voltage and current through an I2C interface, while the DS18B20 measures temperature through a OneWire interface. A buck converter provides a lower-voltage supply for the controller and peripherals. An OLED provides local information, while status LEDs and a buzzer provide visual and audible indications. A relay is used as a disconnect element during detected fault conditions. Wi-Fi transfers measurements to a cloud dashboard such as ThingSpeak or Blynk [4].

This architecture can be represented as five functional layers: (1) battery and load layer, (2) sensing layer, (3) embedded processing and protection layer, (4) communication/cloud layer, and (5) user-interface layer. Separating these functions is useful because protection and control must continue locally even if the communication layer fails. Fig. 1 shows the assembled hardware prototype, and Fig. 2 shows the corresponding system block diagram.

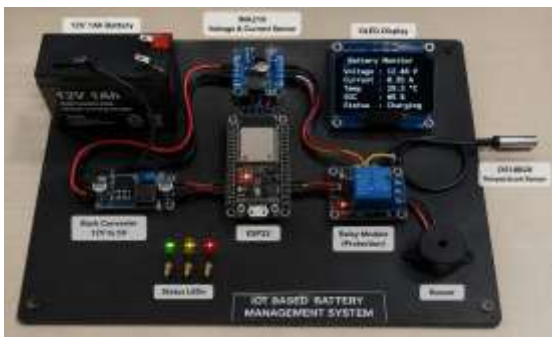

Fig. 1. Assembled IoT-based BMS hardware prototype.

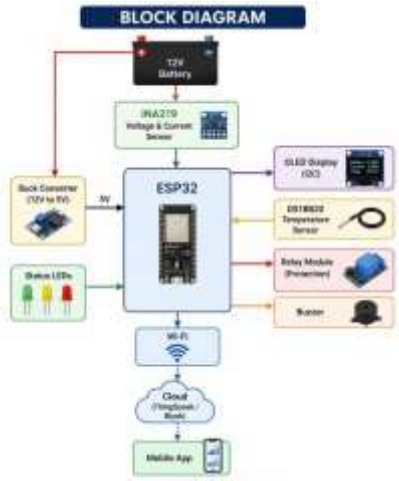

Fig. 2. System block diagram of the reviewed BMS architecture.

VI. WORKING PRINCIPLE

The supplied prototype operates as a continuous measurement-and-protection loop [4]. First, the battery supplies the load and monitoring circuit. The INA219 acquires instantaneous electrical measurements and the DS18B20 acquires temperature. The ESP32 reads the sensors, calculates derived parameters, updates the local display, and transmits the data through Wi-Fi. The controller then compares measured values with configured limits. When a monitored parameter crosses its protection threshold, the relay is commanded to disconnect the battery and the buzzer is activated.

- Battery voltage and current are acquired through the INA219 interface.
- Battery temperature is measured using the DS18B20.
- ESP32 firmware processes the acquired data and calculates SOC, power, and cumulative energy.
- An OLED provides local real-time information.
- Wi-Fi transfers selected measurements to a cloud/mobile dashboard.
- Threshold logic checks for abnormal voltage, current, and temperature conditions.
- A relay provides physical disconnection and a buzzer provides an audible fault indication.

VII. HARDWARE AND SOFTWARE COMPONENTS

Table II lists the principal hardware and software elements of the reviewed architecture and the role of each element within the system.

Table II: Hardware and Software Components of the Reviewed Architecture

Component	Function	Role in reviewed architecture
ESP32 development board	Embedded processing and Wi-Fi	Main controller and communication node
INA219	Voltage/current/power monitoring	Electrical measurement
DS18B20	Digital temperature sensing	Thermal monitoring
Relay module	Electrical disconnection	Fault protection
Buck converter	Voltage reduction	Controller/peripheral supply
OLED / LCD	Local display	Local monitoring
Buzzer + LEDs	Alarm/status indication	User notification
Fuse	Overcurrent safety	Additional hardware protection
Arduino IDE + libraries	Firmware development	Software implementation
ThingSpeak/Blynk or MQTT	Remote data service	Cloud/remote visualization

The original project document lists these components and identifies Arduino IDE, WiFi.h, Wire.h, INA219 libraries, OneWire/DallasTemperature libraries, and Blynk/MQTT-related connectivity as the software environment [4].

VIII. MONITORING, PROTECTION, AND DATA PROCESSING

The principal monitoring variables are voltage, current, and temperature. Electrical power can be obtained from $P = VI$. Cumulative energy can be

estimated by numerical integration of power over time, for example $E \approx \Sigma P(k)\Delta t$, provided the sampling interval and units are handled consistently. These calculations are suitable for embedded monitoring, although accuracy depends on sensor calibration and sampling methodology.

The prototype's protection concept uses threshold-based decisions. This is appropriate for a basic demonstration because it is computationally simple and predictable. Nevertheless, a production BMS should incorporate validated limits specific to battery chemistry, cell configuration, temperature, charge/discharge direction, and operating conditions. Generic threshold values should not be transferred between lead-acid and lithium-ion batteries without appropriate engineering validation.

A key distinction is therefore made between monitoring and safety certification. The prototype demonstrates monitoring and a relay-based response to deliberately induced abnormal conditions, but the supplied evidence does not establish compliance with automotive or battery safety standards, nor does it establish a complete

short-circuit protection strategy. These claims should not be inferred from the prototype demonstration [4].

IX. RESEARCH GAP AND CHALLENGES

The literature indicates several areas in which low-cost IoT BMS implementations require further development.

- Accurate SOC estimation: voltage-only or simplified estimation can be sensitive to load, temperature, aging, and battery chemistry [5].
- SOH and remaining-life estimation: long-term degradation requires historical data and validated diagnostic models.
- Multi-cell scalability: a series battery pack requires cell-level voltage and temperature acquisition and balancing [6].
- Fault diagnosis: threshold alarms identify some abnormal states but do not necessarily distinguish root causes.
- Communication dependence: cloud connectivity can fail, so protection-critical decisions must remain local.
- Cybersecurity and data integrity: connected BMSs introduce communication and data-security considerations [3], [7].
- Experimental validation: sensor accuracy, SOC error, response time, energy error, and protection repeatability should be quantified under controlled conditions.
- Battery-chemistry dependence: protection thresholds and estimation models must be adapted to the chemistry and cell configuration.

These gaps are important when positioning the ESP32 architecture. The system is well suited to a low-cost academic prototype and remote monitoring demonstrator, while further hardware and algorithm development is needed for a high-reliability multi-cell BMS.

X. FUTURE SCOPE

The supplied project identifies several extensions, including machine-learning-based battery-health prediction, remaining backup-time and charging-time estimation, automatic load shedding, solar/battery/grid source switching, theft or disconnection alerts, GPS tracking, smoke/gas sensing, per-cell balancing, and a digital-twin dashboard [4].

Among these directions, three are particularly suitable for a systematic next phase. First, a multi-cell sensing and balancing layer can transform the single-pack monitoring concept into a more complete BMS architecture. Second, calibrated SOC/SOH estimation can be evaluated using controlled charge-discharge cycles and reference measurements. Third, historical IoT data can be used for anomaly detection and predictive maintenance, provided that sufficient high-quality datasets are collected.

Future work should also evaluate communication failure behavior, sensor faults, relay failure, abnormal temperature rise, and data integrity. Local fail-safe behavior should remain effective when the cloud dashboard is inaccessible.

XI. CONCLUSION

This review examined the role of IoT in modern

Battery Management Systems and positioned an ESP32-based low-cost architecture within the broader BMS research landscape. The reviewed architecture combines an ESP32 controller with INA219 electrical sensing, DS18B20 temperature sensing, local display, relay-based protection, audible indication, and Wi-Fi/cloud monitoring. The supplied prototype demonstrates the feasibility of real-time measurement, SOC-related processing, remote visualization, and threshold-based fault response [4].

The literature confirms that a complete BMS involves more than monitoring voltage, current, and temperature. Accurate state estimation, cell balancing, thermal management, diagnostics, protection, communication, and validation are essential for advanced battery applications [1]-[3], [6]. Therefore, the reviewed ESP32 system should be regarded as a practical low-cost IoT monitoring and protection platform rather than a direct replacement for a fully validated multi-cell automotive BMS. Its principal value is its modularity: additional cell-level sensing, balancing, SOH estimation, predictive analytics, and robust communication can be added progressively. This makes the architecture relevant to academic prototypes and small-scale battery monitoring while providing a foundation for future intelligent and connected BMS research.

ACKNOWLEDGMENT

The authors thank the Department of Electrical Engineering, Tulsiramji Gaikwad Patil College of Engineering and Technology, Nagpur, for the guidance and laboratory facilities provided during this work.

REFERENCES

[1] H. Rahimi-Eichi, U. Ojha, F. Baronti, and M.-Y. Chow, "Battery Management System: An Overview of Its Application in the Smart Grid and Electric Vehicles," IEEE Industrial Electronics Magazine, vol. 7, no. 2, pp. 4-16, Jun. 2013, doi: 10.1109/MIE.2013.2250351.

[2] M. A. Hannan et al., "Towards Safer and Smarter Design for Lithium-Ion-Battery-Powered Electric Vehicles: A Comprehensive Review on Control Strategy Architecture of Battery Management System," Energies, vol. 15, no. 12, 4227, 2022.

[3] H. Wang, K. F. Tsang, C. K. Wu, Y. Wei, Y. Liu, and C. S. Lai, "IEEE P2668 Compatible Evaluation Strategy for Smart Battery Management Systems," Sensors, vol. 22, no. 16, 6057, 2022, doi: 10.3390/s22166057.

[4] "IoT-Based Smart Battery Management System Using ESP32 for Real-Time 12V Battery Monitoring and Protection," supplied project manuscript, 2026.

[5] "Open-Circuit Voltage Models for Battery Management Systems: A Review," Energies, vol. 15, no. 18, 6803, 2022.

[6] "Review of Cell-Balancing Schemes for Electric Vehicle Battery Management Systems," Energies, vol. 17, no. 6, 1271, 2024.

[7] "A Smart Battery Management System (BMS) for Electric Vehicles Using Internet of Things Technology," in Proc. 10th Int. Conf. Communication and Electronics Systems (ICCES), Coimbatore, India, 2025, doi: 10.1109/ICCES67310.2025.11337202.

[8] "Smart Wireless Battery Management System for Real-Time Data Transmission in EVs," in Proc. IEEE 5th Int. Conf. Sustainable Energy and Future Electric Transportation (SEFET), Jaipur, India, 2025, doi: 10.1109/SEFET65155.2025.11255284.

[9] "Series Connected Cell Monitoring System of Li-ion Battery with I2C and Cloud Interfacing," in Proc. IECON 2024 - 50th Annual Conference of the IEEE Industrial Electronics Society, 2024, doi: 10.1109/IECON55916.2024.10905926.

[10] Texas Instruments, "INA219 Zero-Drift, Bidirectional Current/Power Monitor with I2C Interface," Datasheet.

[11] Analog Devices/Maxim Integrated, "DS18B20 Programmable Resolution 1-Wire Digital Thermometer," Datasheet.

[12] Espressif Systems, "ESP32 Series Datasheet," Datasheet.