ijamrsd.com
ISSN: 2583-6404
Journal of Advanced Multidisciplinary Research Studies and Development
Peer Reviewed Referred Journal | Bi-monthly | Open Access

AN IOT ENABLED SMART BATTERY MANAGEMENT SYSTEM USING RASPBERRY PI AND CLOUD ANALYTICS

Ms. Jadhav Vinaya Suresh[1], Mr. Shaikh Aslam Amir[2], Dr. Patil Rahul Keru[3]
[1,3] E& TC, SVPM College of Engineering, Malegaon-Bk Baramati.
Email: jadhavvinaya18@gmail.com, rkpatil@engg.svpm.org.in
[2] E&TC, BPET Kalamb Walchandnagar, Indapur.
Email: Shaikhaslam0359@gmail.com

Abstract
Battery-powered systems like electric vehicles, renewable power storage systems, and portable electronic gadgets need proper monitoring and protection schemes to guarantee safe and efficient operation. Existing Battery Management Systems (BMS) may not offer remote monitoring, real-time analysis, and safety features. This paper proposes an IoT-based Smart Battery Management System using a Raspberry Pi device for real-time monitoring and protection of Lithium Ion batteries. In the proposed system, an INA219 sensor is used to measure voltage and current levels, and a DS18B20 sensor is used to measure temperature levels. The data is processed locally using the Raspberry Pi device and uploaded to the ThingSpeak cloud platform.From the experimental evaluation, it is clear that the system maintains a stable monitoring performance under normal operating conditions with voltage levels ranging from 12.65–12.80 V, current levels ranging from 1.95–2.18 A, and temperatures ranging from 30.4–33.5°C. It is also clear from the experiment that the system is capable of detecting abnormal operating conditions such as overvoltage (15.20 V), overcurrent (12.50 A), and overheating (60.3°C), disconnecting the battery using a relay protection mechanism. Analysis of the performance of the system indicates that it maintains a high level of accuracy in measurement, with voltage measurement accuracy being within ±1%, current measurement accuracy being within ±2%, and temperature measurement accuracy being within ±0.5°C, with a cloud data transmission delay of 1–2 seconds and a relay protection response time of less than 50 ms.

Keywords: Battery Management System, Internet of Things, Raspberry Pi, Relay, things Speak.

► *Corresponding Author: Ms. Jadhav Vinaya Suresh*

I. Introduction

Lithium-ion batteries have become the dominant energy storage technology for modern applications such as electric vehicles, renewable energy storage, and portable electronic devices due to their high energy density, long cycle life, and superior performance characteristics [1], [3]. Despite these advantages, lithium-ion batteries are highly sensitive to operating conditions including over-voltage, over-current, and excessive temperature. These conditions can significantly degrade battery performance, shorten lifespan, and in extreme cases lead to hazardous situations such as thermal runaway [2], [5]. Therefore, the implementation of an efficient Battery Management System (BMS) is essential to ensure safe operation, reliability, and optimal performance of battery packs.

A conventional battery management system typically performs basic tasks such as voltage monitoring, current measurement, and protection against abnormal operating conditions [6]. However, traditional BMS architectures are often limited to local monitoring and lack advanced capabilities such as remote supervision, intelligent data analysis, and predictive diagnostics [7]. As energy storage systems become more complex and widely deployed, there is a growing demand for smart battery monitoring solutions that provide real-time data acquisition, cloud connectivity, and automated safety mechanisms.

The rapid development of the Internet of Things (IoT) has enabled new possibilities for intelligent battery monitoring and management. IoT technologies allow embedded devices and sensors to communicate through network infrastructures, enabling remote monitoring and control of physical systems [11]. Integrating IoT with battery management systems allows real-time data collection, remote monitoring, and cloud-based analytics, significantly improving system efficiency and reliability [13], [14]. Recent studies have demonstrated the use of IoT platforms for monitoring battery parameters such as voltage, current, and temperature in energy storage systems [16], [18].

Low-cost embedded computing platforms such as the Raspberry Pi have become widely adopted in IoT-based monitoring applications due to their high computational capability, flexible communication interfaces, and compatibility with various sensors [15]. These devices enable the implementation of edge computing techniques where sensor data is processed locally before being transmitted to cloud platforms for further analysis. Sensors such as the INA219 current and voltage monitoring sensor and the DS18B20 digital temperature sensor provide accurate real-time measurement of battery parameters through digital communication protocols such as I²C and 1-Wire interfaces [26], [27].

Cloud platforms such as ThingSpeak provide powerful tools for real-time data visualization, storage, and advanced analytics using integrated MATLAB tools [29]. By transmitting battery parameters to the cloud, it becomes possible to monitor system health remotely and detect abnormal operating conditions. Furthermore, automated control mechanisms such as relay-based protection circuits can be implemented to disconnect the battery from the load when unsafe conditions are detected, thereby improving system safety and reliability.

Several researchers have proposed IoT-based battery monitoring frameworks for renewable energy systems and electric vehicles. However, many existing systems focus primarily on data acquisition and lack integrated protection mechanisms or scalable architectures capable of combining sensing, edge processing, cloud analytics, and safety actuation [17], [19]. Therefore, there is a need for a comprehensive and intelligent battery monitoring architecture that integrates sensing, processing, communication, and automated protection within a unified IoT framework.

To address these challenges, this paper proposes an IoT-Enabled Smart Battery Management System using Raspberry Pi and cloud analytics. The proposed system integrates voltage, current, and temperature sensors with an edge computing platform to continuously monitor battery parameters. The system transmits real-time data to the ThingSpeak cloud platform for visualization and analysis while implementing an automated relay-based protection mechanism to disconnect the battery during abnormal conditions. The proposed architecture aims to provide a low-cost, scalable, and intelligent solution for real-time battery monitoring and protection in modern energy storage systems.

II. Related Work

Battery Management Systems (BMS) have been widely studied to improve the safety, reliability, and performance of lithium-ion batteries used in electric vehicles, renewable energy storage

systems, and portable electronics. Early research in this field primarily focused on battery modeling and state-of-charge (SOC) estimation techniques. For example, Plett proposed the use of extended Kalman filtering to estimate battery SOC accurately, which significantly improved the reliability of battery monitoring systems [1]. Similarly, Chen and Rincon-Mora developed an accurate electrical battery model capable of predicting battery runtime and performance characteristics, enabling improved battery monitoring and management strategies [6]. These foundational studies laid the groundwork for modern battery management architectures.

Subsequent research focused on improving battery health estimation and energy management in electric vehicle applications. Khaligh and Li presented a comprehensive overview of hybrid energy storage technologies for electric vehicles, highlighting the importance of efficient BMS for battery safety and performance optimization [7]. Hannan et al. conducted a detailed review of energy storage systems used in electric vehicles and emphasized the need for advanced battery monitoring mechanisms to extend battery life and improve system efficiency [3]. Furthermore, Santhanagopalan and White investigated online estimation techniques for lithium-ion battery SOC, providing methods to monitor battery behavior during real-time operation [5].

With the rapid advancement of Internet of Things (IoT) technologies, researchers have begun integrating IoT frameworks with battery management systems to enable remote monitoring and cloud-based analytics. IoT technologies provide distributed sensing, communication, and data processing capabilities, enabling real-time monitoring of battery parameters such as voltage, current, and temperature [11]. Bellavista et al. highlighted the role of IoT networks and wireless sensor systems in enabling smart monitoring environments where multiple devices can communicate efficiently in distributed applications [12].

Several recent studies have explored IoT-based battery monitoring architectures for energy storage and electric vehicles. Burgio et al. proposed an IoT-enabled solution for monitoring and controlling residential and commercial battery storage systems using cloud-connected devices and gateways, enabling remote supervision and management of energy storage units [13]. Similarly, Zhang et al. developed an IoT-based real-time battery monitoring system that collects sensor data and transmits it to cloud servers for remote analysis and visualization [14]. These approaches demonstrate the effectiveness of integrating IoT technologies with battery monitoring systems to enhance data accessibility and system reliability.

Raspberry Pi and other low-cost embedded computing platforms have become popular for implementing IoT-based monitoring systems due to their flexibility, computational capability, and support for multiple communication interfaces. Research has shown that Raspberry Pi-based monitoring platforms can efficiently collect sensor data and transmit it to cloud platforms for analysis [17]. For instance, Hong et al. developed a Raspberry Pi-based platform for real-time battery state-of-charge estimation, demonstrating the feasibility of implementing battery monitoring systems using low-cost embedded devices [15]. Similarly, Rao et al. proposed a real-time battery monitoring system using Raspberry Pi and IoT technologies that continuously measures voltage, current, and temperature parameters of lithium-ion cells [2].

Recent studies have also investigated the use of advanced analytics and machine learning techniques for battery health prediction. Zhao et al. reviewed various machine learning methods for estimating battery state-of-charge and state-of-health, highlighting their potential for improving battery diagnostics and predictive maintenance [10]. Additionally, recent research has explored the use of deep learning algorithms such as LSTM models to predict the remaining useful life of batteries based on real-time sensor data collected through IoT devices [4]. These approaches enable intelligent battery monitoring systems capable of predicting failures before they occur.

IoT-based battery monitoring systems have also been applied in renewable energy storage and smart grid environments. Studies have shown that integrating IoT technologies with energy management systems can improve energy utilization, optimize battery charging cycles, and enable remote monitoring of distributed energy resources [19]. Furthermore, modern IoT-enabled battery management systems incorporate sensors and microcontrollers to continuously measure battery parameters and provide real-time alerts when abnormal conditions are detected [13].

Despite these advancements, several limitations remain in existing battery monitoring systems. Many existing systems focus primarily on data acquisition without integrating real-time protection mechanisms capable of disconnecting the battery under hazardous conditions [20]. Additionally, some systems rely heavily on cloud connectivity, which may introduce latency and reliability issues during critical fault conditions.

Therefore, there is a need for a **comprehensive IoT-enabled battery monitoring architecture that integrates sensing, edge computing, cloud analytics, and automated protection mechanisms** within a unified framework. The proposed system in this research addresses these challenges by combining sensor-based monitoring, Raspberry Pi edge processing, cloud-based data visualization using ThingSpeak, and relay-based safety protection to ensure reliable and intelligent battery management.

III. Proposed Methodology

The proposed system presents an **IoT-enabled Smart Battery Management System (Smart BMS)** designed to monitor critical battery parameters in real time and ensure safe operation through automated protection mechanisms. The methodology integrates **sensor-based monitoring, edge computing, wireless communication, and cloud-based analytics** to create a reliable and scalable battery monitoring framework. The overall system follows a **sense process protect–communicate architecture**, where sensor data is collected from the battery, processed by an embedded computing platform, and transmitted to a cloud platform for remote monitoring and analysis.

The proposed methodology consists of four major stages: **data acquisition, edge processing, safety control, and cloud communication**. Proposed Methodology shows in figure 1.

3.1 Data Acquisition Layer

The first stage of the system involves **real-time acquisition of battery parameters** using dedicated sensors. The system measures three critical battery parameters: **voltage, current, and temperature**, which are essential indicators of battery health and operating conditions.A**voltage and current monitoring sensor (INA219)** is used to measure the electrical parameters of the battery. The INA219 sensor utilizes a precision shunt resistor to detect current flow and measure bus voltage. The sensor communicates with the processing unit using the **I²C communication protocol**, allowing accurate digital measurements with minimal noise.In addition to electrical measurements, the **DS18B20 digital temperature sensor** is used to monitor the battery temperature. The sensor operates using the **1-Wire communication protocol**, which enables efficient data transmission using a single data line. Monitoring battery temperature is critical for preventing overheating and thermal runaway conditions.These sensors continuously collect real-time data from the battery pack and transmit the measurements to the processing unit for further analysis.

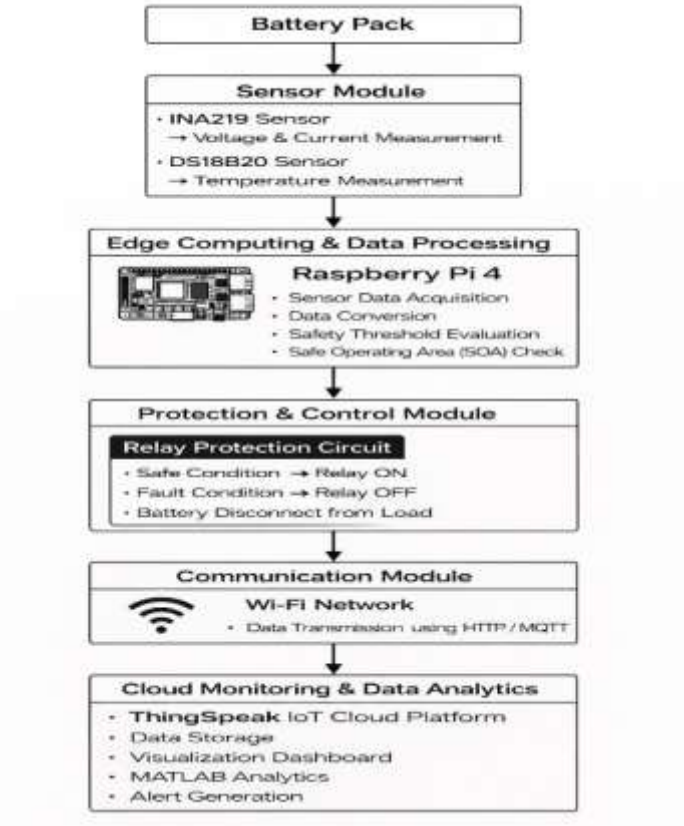

Fig.1 Proposed Methodology

3.2 Edge Computing and Data Processing

The second stage of the methodology involves **edge computing using a Raspberry Pi 4**, which acts as the central controller of the Smart BMS. The Raspberry Pi receives sensor data from the INA219 and DS18B20 sensors and processes the information using embedded software written in Python.

The processing unit performs several tasks:

1. Conversion of raw sensor readings into engineering units such as volts, amperes, and degrees Celsius.
2. Continuous monitoring of battery parameters.
3. Evaluation of the battery's operating conditions using predefined safety thresholds.

The system defines a **Safe Operating Area (SOA)** for the battery, which represents the acceptable operating limits for voltage, current, and temperature. If the measured parameters fall within the defined safe range, the system continues normal operation and transmits the data to the cloud platform for monitoring.

The use of edge computing ensures that critical decisions related to battery safety are performed locally, reducing response time and minimizing dependence on network connectivity.

3.3 Protection and Control Mechanism

The third stage of the system implements a **hardware protection mechanism using a relay module**. The relay acts as an electrical switch that controls the connection between the battery and the load.

The relay module is controlled by the Raspberry Pi through a **General Purpose Input Output (GPIO) pin**. Under normal operating conditions, the relay remains activated, allowing power to flow from the battery to the load. However, if the system detects abnormal conditions such as **over-voltage, over-current, or excessive temperature**, the Raspberry Pi immediately sends a control signal to deactivate the relay.

Deactivating the relay disconnects the battery from the load, preventing potential damage to the battery and connected devices. This **automatic protection mechanism** ensures rapid response during unsafe operating conditions and enhances the overall reliability of the system.

3.4 Cloud Communication and Remote Monitoring

The final stage of the methodology involves **cloud integration for remote monitoring and data analytics**. The processed sensor data is transmitted from the Raspberry Pi to the **ThingSpeak IoT cloud platform** using wireless communication through the Raspberry Pi's built-in Wi-Fi module. The system uses **HTTP or MQTT communication protocols** to send sensor data to the cloud server. Each data packet contains key battery parameters, including voltage, current, temperature, and system status.The cloud platform stores and visualizes this information through real-time dashboards, allowing users to monitor battery performance remotely. Additionally, the ThingSpeak platform integrates MATLAB analytics tools that enable further data analysis, trend visualization, and fault detection.By combining cloud connectivity with local edge processing, the proposed system provides both **real-time monitoring and long-term battery performance analysis**.

3.5 Overall System Architecture

The overall architecture of the proposed Smart BMS follows a **layered IoT architecture consisting of five main layers**:

1. Perception Layer – responsible for sensing battery parameters using voltage, current, and temperature sensors.

2. Edge Computing Layer – processes sensor data using the Raspberry Pi and executes decision-making algorithms.

3. Control Layer – implements safety protection using a relay module that disconnects the battery under abnormal conditions.

4. Communication Layer – transmits processed data to the cloud using wireless communication protocols.

5. Application Layer – provides cloud-based visualization, data storage, and advanced analytics using the ThingSpeak platform.

This layered architecture ensures efficient data acquisition, rapid fault detection, and reliable remote monitoring of battery systems.

3.6 Operational Workflow

The operational workflow of the system follows a continuous monitoring cycle:

1. Sensors measure battery voltage, current, and temperature.
2. Sensor data is transmitted to the Raspberry Pi controller.
3. The Raspberry Pi processes the data and evaluates safety thresholds.
4. If parameters remain within safe limits, the system continues normal operation.
5. If abnormal conditions are detected, the relay disconnects the battery from the load.
6. All system parameters are transmitted to the cloud platform for visualization and analysis.

This continuous monitoring cycle ensures **real-time battery supervision and automatic protection**, making the system suitable for modern energy storage applications.

IV. System Work Flow

The operational logic of the Smart BMS follows a deterministic "Sense-Think-Act" closed-loop cycle. This ensures that the system doesn't just record data but proactively manages the battery's physical state. The workflow is designed with a priority-interrupt structure to ensure safety-critical actions take precedence over telemetry tasks.

Workflow Description and Control Logic:

1. Initialization and Handshaking: Upon system boot, the Raspberry Pi initializes the I2C bus for the INA219 power monitor and searches the 1-Wire directory for the unique 64-bit serial number of the DS18B20 temperature probe. This ensures all peripheral hardware is responsive before the battery circuit is energized.

2. Synchronous Data Acquisition: The system executes a high-frequency polling loop. It initiates a request to the INA219, which returns a 16-bit digital word representing the shunt voltage and bus voltage. Simultaneously, the Pi reads the temperature bitstream. These raw digital values are then scaled into engineering units (V, A, and °C) using calibrated conversion factors programmed into the local Python environment.

3. Real-Time Threshold Logic (The "Safe Zone"): The converted values are immediately passed to a comparator logic block. We define the "Safe Zone" based on the specific chemistry of the battery pack (e.g., $11.0V \leq V \leq 14.6V$ and $T \leq 50°C$).

Normal Path: If the metrics fall within the Safe Zone, the Pi encapsulates the data into a JSON structure. This packet is pushed via an HTTP POST request to the ThingSpeak server for long-term logging and MATLAB-based health trend analysis.

4. Asynchronous Interrupt (The "Critical Zone"): If any single parameter violates the safety bounds, the software enters an Exception Handling state. The local logic treats this as a high-priority interrupt.

Immediate Actuation: The Pi sets GPIO 17 to a LOW state, de-energizing the relay coil. This mechanical isolation happens in approximately 30-50ms, physically disconnecting the battery from the load or charger to prevent thermal runaway or cell swelling.

Post-Fault Communication: Only after the relay has been successfully tripped does the system attempt to broadcast the "FAULT" status to the cloud and trigger SMS/Email notifications. This sequence ensures that hardware safety is never delayed by network latency or Wi-Fi connectivity issues.

5. Autonomous Recovery: The system remains in a "Lockout" state until the parameters return to a hysteresis-defined safe range, at which point it logs a recovery event and awaits manual or automated reset.

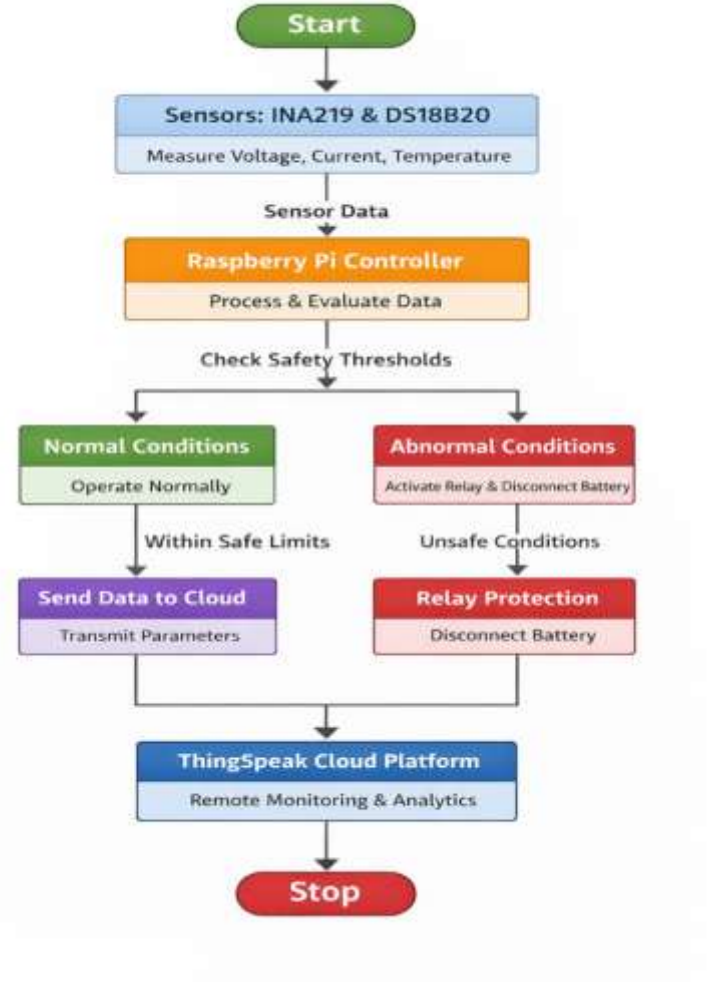

Fig.2 Smart BMS Workflow

V. Hardware Implementation

The physical prototype was assembled on a breadboard, integrating the battery pack with the sensing and control modules.

1. INA219 Interfacing: Connected to the Raspberry Pi via the I²C protocol utilizing GPIO 2 (SDA) and GPIO 3 (SCL).

2. DS18B20 Interfacing: Connected via the 1-Wire interface to GPIO 4, utilizing a 4.7kΩ pull-up resistor between the DQ and VCC pins.

3. Relay Interfacing: A 5V relay module is controlled via GPIO 17. The Common (COM) pin connects to the battery output, while the Normally Open (NO) pin connects to the load.

The software environment utilizes Python 3.9 on the Raspbian OS. Specific libraries such as smbus2 (for I²C), w1thermsensor (for temperature), and paho-mqtt are used for data acquisition and transmission.

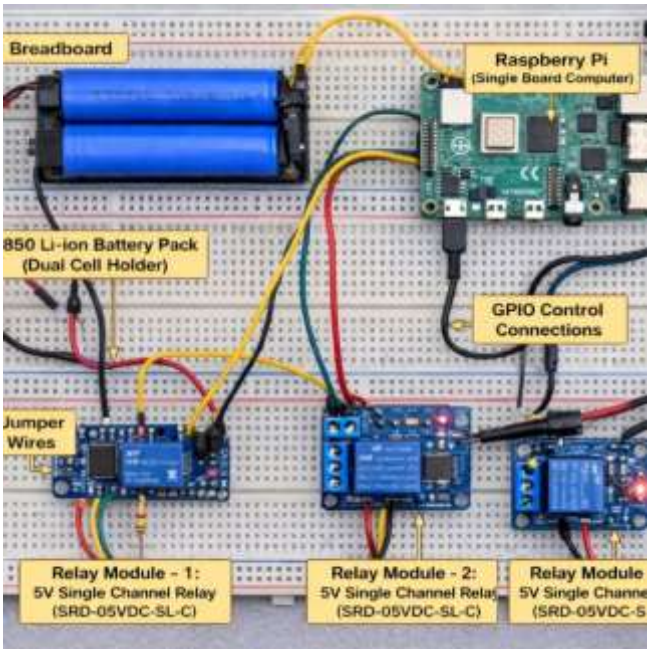

Fig.3 Hardware setup

The hardware architecture of the Smart Battery Management System (Smart BMS) integrates sensing modules, a processing unit, and a control mechanism to ensure safe battery operation. The system continuously monitors critical battery parameters such as voltage, current, and temperature, processes the data using the Raspberry Pi, and takes protective actions when abnormal conditions are detected.The major components of the system include the Raspberry Pi 4 controller, INA219 voltage/current sensor, DS18B20 temperature sensor, relay module, battery pack, and load device. These components are interconnected through communication interfaces such as I²C, 1-Wire protocol, and GPIO control signals.

1. Raspberry Pi 4

The Raspberry Pi 4 serves as the central processing unit of the Smart BMS. It is responsible for collecting sensor data, executing safety algorithms, controlling the relay module, and transmitting data to the cloud platform.

The Raspberry Pi communicates with different hardware components using multiple interfaces:

1. I²C interface for the INA219 sensor
2. 1-Wire interface for the DS18B20 temperature sensor
3. GPIO control pin for operating the relay module

The Raspberry Pi processes all incoming sensor data using Python scripts and determines whether the battery is operating within safe limits.

2. INA219 Voltage and Current Sensor

The INA219 module is used to monitor the electrical parameters of the battery, including voltage and current.

The sensor is connected to the Raspberry Pi using the I²C communication protocol, which requires two data lines:

1. SDA (Serial Data Line) connected to GPIO 2
2. SCL (Serial Clock Line) connected to GPIO 3

The INA219 measures the voltage across the battery and the current flowing through the circuit using an internal shunt resistor. The sensor converts these measurements into digital signals and transmits them to the Raspberry Pi for processing.
These readings help the system determine whether the battery is operating within the defined voltage and current limits.

3. DS18B20 Temperature Sensor

The DS18B20 temperature sensor is used to monitor the temperature of the battery pack to prevent overheating or thermal runaway conditions.
The sensor communicates with the Raspberry Pi through the 1-Wire communication protocol, which requires only one data line. The sensor is connected to:

- GPIO 4 of the Raspberry Pi.

A 4.7 kΩ pull-up resistor is placed between the data line (DQ) and the power supply (VCC) to ensure reliable communication.
The DS18B20 provides high-resolution temperature readings, which are read by the Raspberry Pi using the w1thermsensor Python library.

4. Relay Module (Protection Mechanism)

The 5V relay module acts as a hardware switching device that controls the connection between the battery and the load.
The relay is controlled by the Raspberry Pi using GPIO 17.
When the system operates under normal conditions:

1. The relay remains activated
2. The battery supplies power to the load device

If any abnormal condition is detected, such as:

1. Over-voltage
2. Over-current
3. High temperature

the Raspberry Pi sends a signal to GPIO 17, which deactivates the relay and disconnects the battery from the load, thereby preventing potential damage or safety hazards.

5. Lithium-Ion Battery Pack

The Li-ion battery pack serves as the primary energy source of the system. The battery is connected to the INA219 sensor for monitoring voltage and current parameters.
The output of the battery is also connected to the relay module, which determines whether power should be delivered to the load or disconnected during fault conditions.

6. Load Device

The load device represents the electrical equipment powered by the battery. The load receives power only when the relay is in the normally open (NO) closed state.
When a fault condition occurs, the relay opens the circuit, isolating the load from the battery.

7. Overall System Operation

The Smart BMS hardware operates through the following process:

1. The battery supplies power to the circuit.
2. The INA219 sensor continuously measures battery voltage and current.
3. The DS18B20 sensor monitors the battery temperature.
4. Sensor data is transmitted to the Raspberry Pi.
5. The Raspberry Pi analyzes the parameters using predefined safety thresholds.
6. If the parameters are within safe limits, the relay allows the battery to power the load.

7. If unsafe conditions are detected, the Raspberry Pi deactivates the relay and disconnects the battery from the load.

This architecture ensures real-time monitoring, fault detection, and automatic battery protection, making the system suitable for intelligent battery management applications.

VI. Experimental Results

The proposed IoT-enabled Smart Battery Management System (Smart BMS) was experimentally evaluated to verify the accuracy of sensor measurements, system reliability, protection response time, and cloud communication performance. The prototype system was implemented using a Raspberry Pi 4, INA219 voltage/current sensor, DS18B20 temperature sensor, and a relay-based protection mechanism. Experimental tests were conducted under different operating conditions including normal operation and simulated fault scenarios such as overvoltage, overcurrent, and overheating.

The collected sensor data was transmitted to the ThingSpeak cloud platform, where the battery parameters were visualized and logged for analysis.

6.1 Sensor Measurement Results

The system continuously monitored battery voltage, current, and temperature under normal operating conditions. Table 1 presents sample measurements collected during standard operation.

Table 1: Real-Time Battery Monitoring Data

Test No	Voltage (V)	Current (A)	Temperature (°C)	System Status
1	12.65	1.95	30.4	Normal
2	12.72	2.10	31.2	Normal
3	12.80	2.05	32.1	Normal
4	12.75	2.18	33.5	Normal
5	12.70	2.00	32.8	Normal

Table 1 presents the real-time measurements of key battery parameters including voltage, current, and temperature during normal operating conditions. The recorded voltage values range from 12.65 V to 12.80 V, while the current values vary between 1.95 A and 2.18 A, indicating stable power delivery to the load. The battery temperature remains within the safe operating range of 30.4°C to 33.5°C, confirming that the system operates under normal thermal conditions. These results demonstrate that the proposed Smart BMS successfully monitors battery parameters in real time and maintains operation within the predefined Safe Operating Area (SOA).

6.2 Fault Detection and Protection Results

To validate the system's protection capability, different fault conditions were intentionally introduced. The Smart BMS successfully detected abnormal parameters and activated the relay protection mechanism.

Table 2: Fault Detection and Protection Performance

Test Case	Voltage (V)	Current (A)	Temperature (°C)	Detected Fault	Relay Action
Overvoltage	15.20	2.10	32.0	Overvoltage	Battery Disconnected
Overcurrent	12.60	12.50	34.2	Overcurrent	Battery Disconnected

Overtemperature	12.55	2.05	60.3	Overtemperature	Battery Disconnected

Table 2 illustrates the system's response under different fault conditions including overvoltage, overcurrent, and overtemperature scenarios. When the voltage increased to 15.20 V, the system detected an overvoltage condition and immediately disconnected the battery using the relay mechanism. Similarly, an overcurrent condition of 12.50 A and an overheating condition of 60.3°C were successfully identified by the system. In all cases, the relay protection mechanism isolated the battery from the load to prevent potential damage. These results confirm the effectiveness of the proposed Smart BMS in detecting abnormal operating conditions and activating automatic protection.

6.3 System Performance Evaluation

Several performance metrics were analyzed to evaluate the efficiency of the proposed system.

Table 3: System Performance Metrics

Parameter	**Measured Value**
Voltage Measurement Accuracy	±1%
Current Measurement Accuracy	±2%
Temperature Sensor Accuracy	±0.5°C
Cloud Data Transmission Delay	1 – 2 seconds
Relay Protection Response Time	< 50 ms
Data Sampling Interval	2 seconds

Table 3 summarizes the overall performance evaluation of the proposed Smart BMS. The measurement accuracy of the voltage sensor was approximately ±1%, while the current sensor achieved an accuracy of ±2%. The DS18B20 temperature sensor demonstrated high precision with an accuracy of ±0.5°C. The system transmitted sensor data to the cloud platform with an average delay of 1–2 seconds, enabling near real-time monitoring. Additionally, the relay-based protection mechanism responded in less than 50 milliseconds, ensuring rapid disconnection of the battery during unsafe conditions. These performance metrics validate the reliability and efficiency of the proposed monitoring system.

6.4 Cloud Monitoring Results

The system successfully transmitted sensor data to the **ThingSpeak cloud platform,** where real-time dashboards displayed the following parameters:

1. Battery Voltage
2. Battery Current
3. Battery Temperature
4. System Status

The cloud interface enabled **remote monitoring and historical data analysis**, allowing users to observe battery behavior over time and detect potential faults early.

6.5 System Reliability Analysis

The overall performance of the Smart BMS demonstrated high reliability during experimental testing. The system maintained stable monitoring under normal conditions while rapidly detecting unsafe situations. The relay protection mechanism effectively isolated the battery from the load during fault conditions, ensuring safe operation of the energy storage system.The integration of **edge computing and cloud analytics** enabled both immediate safety response and long-term data

monitoring, making the proposed system suitable for applications such as renewable energy storage, electric vehicles, and smart grid systems.

VII. Conclusion

The proposed IoT-based Smart Battery Management System is successfully validated for the efficient real-time monitoring and protection of the battery system through the use of Raspberry Pi and cloud analytics. The proposed system is able to accurately sense the voltage, current, and temperature of the battery system through the use of INA219 and DS18B20 sensors. The proposed system is found to operate stably in the safe range of 12.65–12.80 V, 1.95–2.18 A, and 30–33°C. The proposed system is able to accurately detect the faults of overvoltage, overcurrent, and overheating. The proposed system is able to disconnect the battery through the use of the relay protection method in 50 ms. The proposed system is able to achieve the accuracy of ±1–2% along with the communication delay of 1–2 seconds through the use of the ThingSpeak cloud. The proposed system is found to be efficient in the real-time monitoring of the battery system.

References

1. G. L. Plett, “Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs: Part 1. Background,” *J. Power Sources*, vol. 134, no. 2, pp. 252–261, Aug. 2004.
2. J. Jiang and C. Zhang, “Fundamentals and control strategies of lithium-ion battery management systems,” *IEEE Ind. Electron. Mag.*, vol. 10, no. 2, pp. 36–46, Jun. 2016.
3. M. A. Hannan, M. M. Hoque, A. Mohamed, and A. Ayob, “Review of energy storage systems for electric vehicle applications,” *Renew. Sustain. Energy Rev.*, vol. 69, pp. 771–789, Mar. 2017.
4. W. Y. Chang, “The state of charge estimating methods for battery: A review,” *ISRN Appl. Math.*, vol. 2013, pp. 1–7, 2013.
5. S. Santhanagopalan and R. E. White, “Online estimation of the state of charge of a lithium-ion cell,” *J. Power Sources*, vol. 161, no. 2, pp. 1346–1355, Oct. 2006.
6. M. Chen and G. A. Rincon-Mora, “Accurate electrical battery model capable of predicting runtime and I–V performance,” *IEEE Trans. Energy Convers.*, vol. 21, no. 2, pp. 504–511, Jun. 2006.
7. A. Khaligh and Z. Li, “Battery, ultracapacitor, fuel cell, and hybrid energy storage systems for electric, hybrid electric, fuel cell, and plug-in hybrid electric vehicles: State of the art,” *IEEE Trans. Veh. Technol.*, vol. 59, no. 6, pp. 2806–2814, Jul. 2010.
8. J. Jaguemont, L. Boulon, and Y. Dubé, “A comprehensive review of lithium-ion batteries used in hybrid and electric vehicles,” *J. Power Sources*, vol. 287, pp. 93–105, Jul. 2015.
9. H. He, R. Xiong, and J. Fan, “Evaluation of lithium-ion battery equivalent circuit models for state of charge estimation,” *Appl. Energy*, vol. 87, no. 1, pp. 239–248, Jan. 2010.
10. F. Zhao, X. Liu, and Y. Wang, “A review of lithium-ion battery state-of-charge estimation using machine learning,” *World Electr. Veh. J.*, vol. 15, no. 4, p. 131, 2024.
11. L. Da Xu, W. He, and S. Li, “Internet of things in industries: A survey,” *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.
12. P. Bellavista, G. Cardone, A. Corradi, and L. Foschini, “Convergence of MANET and WSN in IoT urban scenarios,” *IEEE Sensors J.*, vol. 13, no. 10, pp. 3558–3567, Oct. 2013.
13. A. Burgio, D. Cimmino, A. Nappo, L. Smarrazzo, and G. Donatiello, “An IoT-based solution for monitoring and controlling battery energy storage systems at residential and commercial levels,” *Energies*, vol. 16, no. 7, p. 3140, 2023.

14. K. Zhang, Y. Li, and J. Deng, "IoT-based real-time battery monitoring system for smart energy applications," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7263–7272, Aug. 2020.
15. S. Hong, H. Kim, and J. Park, "Raspberry Pi-based real-time SOC estimation platform for battery monitoring," *Electronics*, vol. 11, no. 13, p. 2010, Jul. 2022.
16. Y. Zhang, H. Wang, and L. Chen, "IoT-based monitoring system for lithium-ion battery aging," *Energy Rep.*, vol. 8, pp. 450–462, Apr. 2022.
17. H. A. Gabbar and K. Abdussami, "Review of battery management systems development and industrial standards," *Technologies*, vol. 9, no. 2, p. 28, 2021.
18. J. Lee, H. Kim, and S. Park, "Advanced battery monitoring system using IoT for renewable energy storage," *Renew. Energy*, vol. 187, pp. 1152–1163, 2022.
19. M. Patel, K. Desai, and R. Shah, "IoT-based battery health monitoring system for industrial applications," *IEEE Trans. Ind. Electron.*, vol. 68, no. 9, pp. 8974–8983, Sep. 2021.
20. I. Syafii, R. Rahim, and M. Syahril, "Battery state-of-charge monitoring and control system using coulomb counting method," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 33, no. 2, pp. 780–789, 2024.
21. A. Khaligh and O. C. Onar, *Energy Harvesting: Solar, Wind, and Ocean Energy Conversion Systems*. Boca Raton, FL, USA: CRC Press, 2017.
22. S. Vazquez et al., "Energy storage systems for transport and grid applications," *IEEE Trans. Ind. Electron.*, vol. 57, no. 12, pp. 3881–3895, Dec. 2010.
23. D. Andrea, *Battery Management Systems for Large Lithium-Ion Battery Packs*. Boston, MA, USA: Artech House, 2010.
24. A. Emadi, S. S. Williamson, and A. Khaligh, "Power electronics intensive solutions for advanced electric, hybrid electric, and fuel cell vehicular power systems," *IEEE Trans. Power Electron.*, vol. 21, no. 3, pp. 567–577, May 2006.
25. M. Dubarry and B. Y. Liaw, "Identify capacity fading mechanism in lithium-ion cells," *J. Power Sources*, vol. 194, no. 1, pp. 541–549, Oct. 2009.
26. Texas Instruments, "INA219 Zerø-drift, bidirectional current/power monitor with I²C interface," Datasheet, 2022.
27. Maxim Integrated, "DS18B20 programmable resolution 1-wire digital thermometer," Datasheet, 2021.
28. Raspberry Pi Foundation, "Raspberry Pi hardware documentation," Raspberry Pi Foundation, Cambridge, U.K., 2025.
29. MathWorks, "ThingSpeak IoT analytics platform documentation," MathWorks Inc., Natick, MA, USA, 2025.
30. OASIS, "MQTT version 5.0 specification," OASIS Standard, 2019.