



## IoT-Enabled Smart Battery Management System

**V.G. Umale<sup>1</sup>, Ashray Babhare<sup>2</sup>, Kashish Gajbhiye<sup>3</sup>, Pranjal Kale<sup>4</sup>, Riya Narnaware<sup>5</sup>, Vaishnavi Thakare<sup>6</sup>**

Department of Electrical Engineering  
Priyadarshini College of Engineering, Nagpur, Maharashtra, India

### ABSTRACT—

Most battery management systems (BMS) in electric vehicles (EVs) reach the user only as a single pack-level number, which hides what individual cells are doing and leaves the driver with little warning before a thermal event. This paper presents the design and prototype testing of a low-cost, IoT-enabled BMS built around an ESP32 controller for a 4S lithium-ion pack (14.8 V nominal, 16.8 V maximum). The prototype measures the voltage at every cell tap through resistor dividers, records load current and power with an INA219 sensor, and tracks three temperature points with DS18B20 probes. A four-mode thermal strategy (normal, fan, max cool and shutdown) switches a fan, a Peltier module and a protection relay according to user-configurable temperature limits, and a buzzer signals the shutdown state. Readings are shown on a local 2.8-inch touchscreen and on a web dashboard served by the ESP32 itself over its own Wi-Fi access point, together with an event log and settings that survive a restart. A brushless DC motor driven through an electronic speed controller acts as an adjustable load during testing. Bench tests covering fourteen functions confirmed that measurement, cooling escalation, relay cut-off and both user interfaces behaved as designed. The paper also states plainly what the prototype does not yet do, mainly capacity-based State of Health estimation, coulomb counting and cell balancing, and outlines how these can be added.

**Index Terms**—battery management system, electric vehicle, ESP32, lithium-ion battery, thermal management, Peltier cooling, cell monitoring, IoT dashboard.

### INTRODUCTION

The shift towards electric mobility has made the lithium-ion battery the most important and the most scrutinised component of a modern vehicle. Cells store a great deal of energy in a small volume, but they tolerate only a narrow window of voltage and temperature. Charging a cell too far, draining it too deeply, or letting it run hot shortens its life and, in the worst case, can start a self-sustaining thermal runaway [1], [2]. The battery management system is the electronic layer that keeps the pack inside that window [3], [4].

A working BMS has to do several jobs at once: measure voltage, current and temperature, estimate the remaining charge and the ageing of the pack, protect it from abuse, and share its condition with the rest of the vehicle [5], [6]. In practice, however, the information that reaches the driver is heavily summarised. A single battery percentage and a warning lamp say very little about whether one weak cell is dragging the pack down or whether the pack is slowly heating up. Reviews of thermal-runaway research also point out that protection in commercial systems is largely reactive, acting only after a limit has been crossed, and argue for earlier, more proactive intervention [1], [7].

This work addresses that gap at prototype scale. We built a BMS that exposes cell-level voltages, current, power and three temperatures in real time, reacts to rising temperature in stages rather than with a single hard cut, and shares everything through both a local touchscreen and a self-hosted web dashboard that needs no internet connection. The main contributions are as follows.

- A tap-based voltage measurement scheme that recovers the voltage of each of the four series cells on a single ESP32.
- A four-mode, threshold-driven thermal strategy that escalates from fan cooling to Peltier cooling and finally to relay cut-off, with all limits adjustable at run time.
- A dual user interface: an ILI9341 touchscreen with seven pages, and a Wi-Fi dashboard hosted directly on the microcontroller.
- Non-volatile storage of settings and a time-stamped event log to support troubleshooting.
- A repeatable load-test arrangement using a brushless motor, an ESC and a servo tester.

The remainder of the paper is organised as follows. Section II reviews related work, Section III describes the system architecture, Sections IV and V cover the hardware and software, Section VI gives the mathematical relations, Section VII explains the thermal strategy, Section VIII reports the tests, and Sections IX and X discuss limitations and conclude.

### RELATED WORK

#### A. BMS architectures and state estimation

Hardware concepts for BMS design range from simple centralised boards to modular and distributed layouts, each trading cost against wiring complexity and measurement quality [4]. Plett's treatment of battery modelling remains a standard foundation for the estimation methods used in these systems [3]. For State of Charge (SoC), the literature describes open-circuit-voltage lookup, coulomb counting, model-based observers and data-driven

techniques, with the simplest methods being cheap but sensitive to load and temperature [6], [8]. State of Health (SoH) is commonly defined through the ratio of present usable capacity to rated capacity, and estimation methods are reviewed in [9] and [10]. Our prototype relies on the simplest end of this spectrum for SoC, which is one of its declared limitations.

### B. Thermal behaviour and cooling

Heat generation during charge and discharge, and the way a temperature rise can escalate into thermal runaway, are described in [1] and [2]. Rao and Wang survey the cooling routes available for power batteries, including air, liquid, phase-change and heat-pipe approaches [11]. Thermoelectric (Peltier) devices are attractive for small packs because they are compact and can be switched electronically, although their efficiency is modest, as summarised in a recent review [12]. This is the reason the present design uses the Peltier element only as a second cooling stage after the fan.

### C. IoT-based and intelligent BMS

Swamy et al. analyse thermal-runaway models and recommend a proactive battery safety management layer that uses voltage, current, temperature and SoH for early fault detection [7]. Suddepong et al. describe a low-cost IoT BMS for a 72 V, 20-cell NMC pack in which an ESP32-S3 helps detect abnormal cell-voltage trends before failure [13]. Shaout and Brauchler apply fuzzy logic to charging and balancing so that the pack performs better than with conventional fixed-rule control [14]. Our project follows the same direction of travel, visibility and early action, but uses a much simpler hardware set so that it can be reproduced in a student laboratory.

## METHODOLOGY

### 1. SYSTEM ARCHITECTURE

Fig. 1 shows the overall arrangement. The ESP32 sits at the centre. On its input side it receives the divided tap voltages, the INA219 readings (I<sup>2</sup>C) and the three DS18B20 temperatures (1-Wire). On its output side it drives the fan, the Peltier module, the protection relay and the buzzer, refreshes the touchscreen and serves the dashboard. The motor and its ESC are connected on the load side of the relay, so that opening the relay removes the load from the battery.

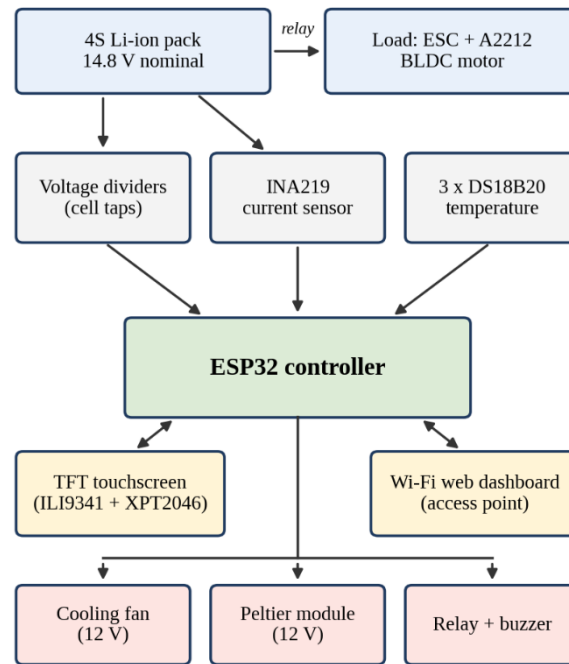


**Fig. 1.** Block diagram of the proposed IoT-enabled BMS.

A LM2596 buck converter provides the regulated supply for the low-voltage electronics from the pack. Keeping the control electronics on their own regulated rail avoids feeding motor switching noise into the analogue measurements.

### 2. HARDWARE IMPLEMENTATION

#### A. Components

Table I lists the parts used and the role each one plays. All parts are inexpensive and easy to source, which was a deliberate choice; their electrical details follow the manufacturer datasheets [15]–[17].

**TABLE I** COMPONENTS AND THEIR FUNCTIONS

| Component               | Specification                            | Function                |
|-------------------------|------------------------------------------|-------------------------|
| ESP32 development board | 32-bit, Wi-Fi/Bluetooth, 3.3 V logic     | Main controller         |
| Li-ion pack             | 4S, 14.8 V nominal, 16.8 V max, 2200 mAh | Energy source           |
| INA219                  | I <sup>2</sup> C voltage/current sensor  | Voltage, current, power |

| Component        | Specification                     | Function                          |
|------------------|-----------------------------------|-----------------------------------|
| DS18B20 (×3)     | Digital 1-Wire temperature sensor | Battery, BMS and test temperature |
| TFT display      | 2.8 in, ILI9341, SPI              | Local display                     |
| XPT2046          | SPI touch controller              | Touch input                       |
| Voltage dividers | Resistor networks                 | Scale tap voltage for ADC         |
| Relay module     | 5 V DC relay                      | Load disconnect                   |
| Cooling fan      | 12 V DC                           | Forced-air cooling                |
| Peltier module   | 12 V thermoelectric               | Additional cooling                |
| Buzzer           | 5 V DC                            | Audible alert                     |
| LM2596           | Adjustable buck converter         | Regulated supply                  |
| BLDC motor + ESC | A2212 1000 KV, 150 W; 30 A ESC    | Variable electrical load          |
| Servo tester     | PWM signal generator              | Sets motor speed                  |

### B. Pin allocation

The ESP32 pin assignment used in the prototype is summarised in Table II. The display shares the SPI bus with the touch controller, each with its own chip-select line.

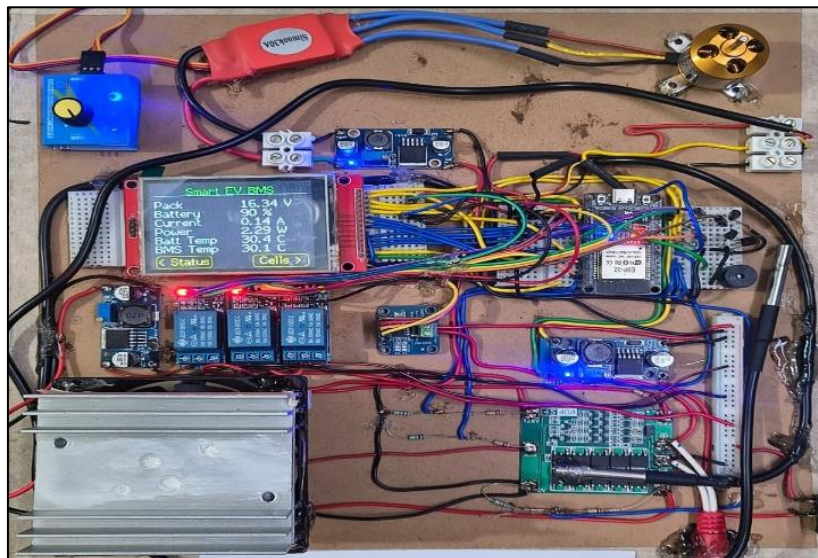
TABLE II ESP32 PIN ALLOCATION

| Signal           | GPIO    | Interface        |
|------------------|---------|------------------|
| INA219 SDA / SCL | 21 / 22 | I <sup>2</sup> C |
| DS18B20 data     | 25      | 1-Wire           |
| Protection relay | 26      | Digital output   |
| Cooling fan      | 13      | Digital output   |
| Peltier module   | 12      | Digital output   |
| Buzzer           | 5       | Digital output   |
| Touch CS / IRQ   | 14 / 27 | SPI / interrupt  |

### C. Prototype photographs

Photographs of the assembled prototype are placed in Figs. 2 to 6. The frames below are left blank for the team to insert its own images.

Fig. 2. Complete assembled hardware prototype (top view).



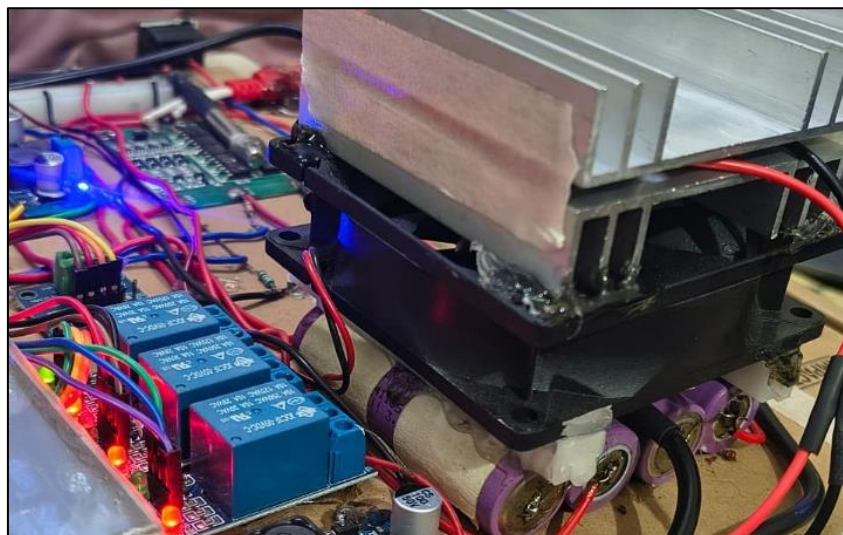


Fig. 3. Fan and Peltier cooling assembly with relay module.



Fig. 4. BLDC motor

## SOFTWARE AND USER INTERFACE

The firmware is written in C/C++ with the Arduino IDE and the ESP32 Arduino framework. The TFT\_eSPI library drives the display, the Adafruit INA219 library reads the current sensor, and the OneWire and DallasTemperature libraries handle the temperature probes. The web dashboard uses the WiFi and WebServer libraries, with the page itself written in HTML, CSS and JavaScript and refreshed with JSON data. Settings are held in flash through the Preferences library.

At start-up the controller initialises the sensors, display, relay, cooling outputs and Wi-Fi, then loads the saved limits. The main loop then repeats the same sequence: read the tap voltages, derive cell voltages, read current and power, read the three temperatures, estimate battery percentage and SoH, compare the cell spread and temperatures with their limits, drive the outputs, refresh the display and dashboard, and write any change of state to the event log.

### A. Touchscreen

The touchscreen has seven pages: Dashboard, Cell Monitor, Battery Analytics, Cooling, Configuration, System Status and Event Log. The Configuration page lets the user change the fan, Peltier and cut-off temperatures without reprogramming the board.

### B. Web dashboard

The ESP32 creates its own Wi-Fi network named "Smart EV BMS". A phone or laptop that joins this network can open the dashboard at 192.168.4.1 and see battery voltage, percentage, current, power, cell voltages, the three temperatures, the fan, Peltier and relay states, the cooling mode, the current settings and the event log. Because the page is served by the controller itself, no router or internet service is needed, which suits a moving vehicle or a workshop bench.

### C. Event log

A new log entry is written only when something changes: a fan, Peltier or relay state, a cooling mode, or a saved setting. Recording changes instead of samples keeps the log short and readable, and makes it easy to see in what order events happened during a test.

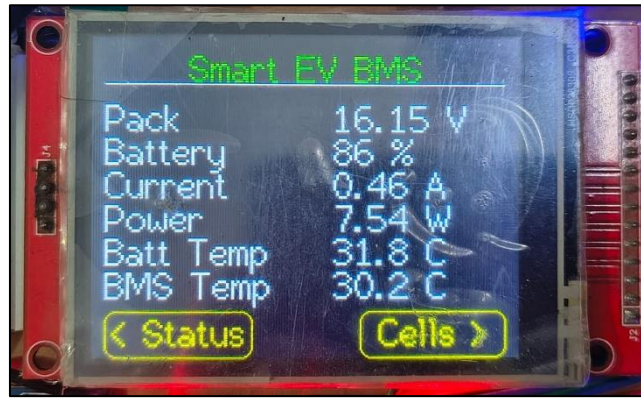


Fig. 5. TFT touchscreen

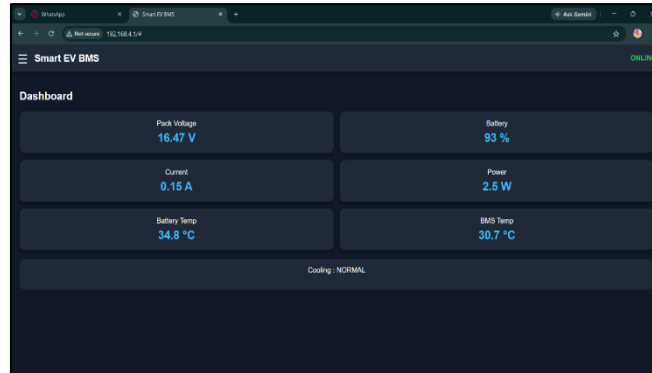


Fig. 6. Web dashboard opened on a Laptop.

## 4. MATHEMATICAL FORMULATION

### A. Voltage scaling

The tap voltages are higher than the ESP32 ADC can accept, so each tap passes through a resistor divider. The voltage seen by the ADC and the value recovered in software are

$$V_{out} = V_{in} R_2 / (R_1 + R_2) \quad (1)$$

$$V_{in} = V_{out} (R_1 + R_2) / R_2 \quad (2)$$

where  $R_1$  is the upper and  $R_2$  the lower resistor. Accuracy depends on resistor tolerance and ADC calibration.

### B. Individual cell voltages

Because the cells are in series, each tap carries the sum of the cells beneath it. The four cell voltages are therefore obtained by subtraction:

$$V_{c1} = V_{T1} \quad (3)$$

$$V_{c2} = V_{T2} - V_{T1} \quad (4)$$

$$V_{c3} = V_{T3} - V_{T2} \quad (5)$$

$$V_{c4} = V_{pack} - V_{T3} \quad (6)$$

and the imbalance indicator is the spread between the highest and lowest cell:

$$\Delta V_{cell} = V_{max} - V_{min} \quad (7)$$

### C. Power and battery percentage

$$P = V \cdot I \quad (8)$$

For instance, 14.8 V at 2 A corresponds to 29.6 W. The battery percentage is estimated linearly between 12.0 V and 16.8 V and clipped to the range 0–100 %:

$$SoC = (V_{batt} - 12.0) / (16.8 - 12.0) \times 100\% \quad (9)$$

### D. State of Health

The reference definition of SoH used in this project compares the capacity that can currently be delivered with the rated capacity, where the delivered capacity is obtained by integrating current over a discharge:

$$SoH = (Q_{current} / Q_{rated}) \times 100\% \quad (10)$$

$$Q_{current} = \int I(t) dt \quad (11)$$

In the present firmware, however, the value on the display is a simplified indicator derived from the estimated charge level,  $SoH = 98 + (SoC/100) \times 2$ , which stays between 98 % and 100 %. It demonstrates how an SoH figure travels from the controller to the screens, but it does not measure ageing. Replacing it with the capacity-based form above is the first item of future work.

#### E. Motor load

The A2212 motor has a rating of about 1000 rpm per volt, so its theoretical no-load speed on a fully charged 4S pack is roughly  $1000 \times 16.8 \approx 16,800$  rpm. The real speed is lower under load and depends on the ESC, battery voltage and losses.

## THERMAL MANAGEMENT AND PROTECTION

Instead of cutting power the moment a single limit is crossed, the controller escalates through four modes using three user-set temperatures:  $T_{fan}$ ,  $T_{peltier}$  and  $T_{cutoff}$ . The decision is made on the test-point temperature  $T_{test}$ , and the resulting output states are listed in Table III and drawn in Fig. 9.

**TABLE III** COOLING MODES AND OUTPUT STATES

| Mode     | Condition                                | Fan | Peltier | Relay | Buzzer |
|----------|------------------------------------------|-----|---------|-------|--------|
| NORMAL   | $T_{test} < T_{fan}$                     | OFF | OFF     | ON    | Off    |
| FAN      | $T_{fan} \leq T_{test} < T_{peltier}$    | ON  | OFF     | ON    | Off    |
| MAX COOL | $T_{peltier} \leq T_{test} < T_{cutoff}$ | ON  | ON      | ON    | Off    |
| SHUTDOWN | $T_{test} \geq T_{cutoff}$               | ON  | ON      | OFF   | On     |

In SHUTDOWN the relay opens and removes the load while the fan and Peltier keep running, so the pack continues to be cooled after it has been isolated. The relay closes again once the temperature returns below the limit, which lets the system recover without manual intervention. The staged approach gives the pack a chance to cool while still delivering power, and the buzzer and dashboard make the shutdown state obvious to the user.

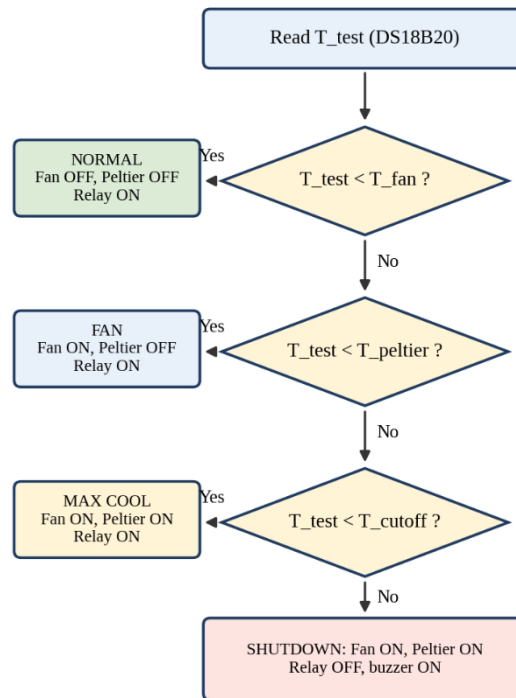


**Fig. 7.** Flowchart of the temperature-based cooling and protection logic.

## TESTING AND RESULTS

#### A. Test set-up

Testing was carried out on the bench with the complete prototype connected. The BLDC motor, connected through the 30 A ESC, was run at different throttle settings from the servo tester so that the battery current and power changed in a controlled way. Temperature behaviour was checked by raising the temperature at the test sensor until each threshold was reached. Displayed values were compared with the physical state of the hardware, since a correct software state

does not by itself prove that the switched device is really operating.

## B. Observations

Battery voltage and the four cell voltages were displayed correctly on both the touchscreen and the dashboard. The INA219 followed the motor load: current and power changed with throttle and were visible on the Battery Analytics page. The three DS18B20 probes responded to temperature changes and fed the control logic.

For cooling, the fan stayed off below the fan limit and came on when the limit was reached. Raising the temperature further activated the Peltier module and produced the MAX COOL mode. At the cut-off limit the controller entered SHUTDOWN and switched the relay off. Changed limits were used by the algorithm straight away and were still present after the ESP32 was restarted. The event log recorded the mode and output changes with time information. Table IV summarises all fourteen tests.

**TABLE IV SUMMARY OF FUNCTIONAL TESTS**

| No. | Function tested                   | Outcome                                |
|-----|-----------------------------------|----------------------------------------|
| 1   | Battery voltage measurement       | Monitored successfully                 |
| 2   | Individual cell/tap voltages      | Displayed successfully                 |
| 3   | Current measurement (INA219)      | Followed the connected load            |
| 4   | Power monitoring                  | Calculated and displayed               |
| 5   | Temperature measurement (DS18B20) | Three points monitored                 |
| 6   | Fan control                       | Switched at the fan threshold          |
| 7   | Peltier control                   | Switched at the Peltier threshold      |
| 8   | Thermal cut-off                   | Relay opened at the cut-off condition  |
| 9   | TFT display pages                 | All seven pages operated               |
| 10  | Touchscreen                       | Navigation and controls worked         |
| 11  | Configuration storage             | Values kept after restart              |
| 12  | Wi-Fi dashboard                   | Opened through the ESP32 access point  |
| 13  | Event logging                     | State changes recorded                 |
| 14  | BLDC load operation               | Motor ran through ESC and servo tester |

The photographs in Figs. 8 to 10 record the three cooling states during testing.

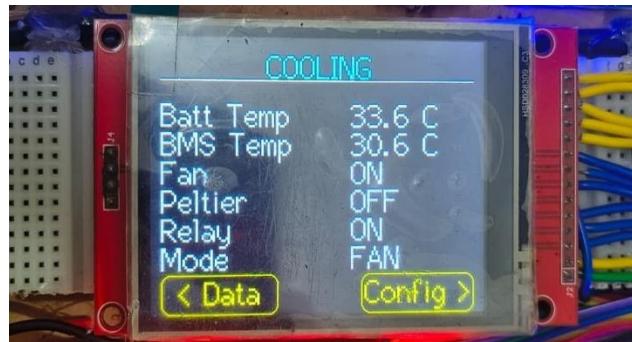


**Fig. 8. System in FAN mode (fan ON, Peltier OFF).**



**Fig. 9. System in MAX COOL mode (fan and Peltier ON).**



**Fig. 10. System in SHUTDOWN mode (relay OFF, buzzer active).**

### C. Note on quantitative results

The tests reported here are functional: each function either behaved as designed or it did not. Numerical figures such as measurement error against a reference meter, temperature-versus-time curves and response delays have not yet been collected. They would strengthen the evidence and are listed as planned work in Section X.

## DISCUSSION AND LIMITATIONS

The prototype shows that one inexpensive microcontroller can combine cell-level monitoring, staged thermal control, a local interface and wireless monitoring. Seeing every cell voltage matters because a normal-looking pack voltage can hide one weak cell. Staged cooling gives the pack a chance to recover before power is removed, and a self-hosted dashboard keeps the system usable without any network infrastructure.

The limitations are equally important. First, the battery percentage is derived from pack voltage alone, so it shifts with load and temperature and is only approximate. Second, the SoH shown is a simplified indicator and not a measurement of ageing. Third, the design has no cell-balancing circuit. Fourth, protection is temperature-based; a commercial system would also cover over-voltage, under-voltage, over-current, short-circuit and insulation monitoring. Fifth, voltage accuracy depends on divider tolerance and calibration. Finally, the protection decision runs on the same controller as everything else, so a controller fault could affect it. For this reason the prototype should be treated as an educational and experimental platform and not as a replacement for a certified automotive BMS.

## CONCLUSION AND FUTURE SCOPE

An ESP32-based smart BMS for a 4S lithium-ion pack was designed, built and bench-tested. It measures pack and cell voltages, current, power and three temperatures, applies a four-mode cooling and cut-off strategy through a fan, a Peltier module and a relay, and presents the results on a touchscreen and on a web dashboard, with saved settings and an event log. All fourteen planned functional tests were completed successfully, and a motor-and-ESC arrangement provided a practical load.

Future work will proceed in five directions. (1) Capacity-based SoH estimation using measured discharge capacity, internal resistance and charge history, together with coulomb-counting SoC. (2) An independent secondary supervisory controller with its own hardware cut-off path, so that protection no longer depends on a single processor, together with a driver-facing rest advisory issued before temperatures become critical. (3) Passive or active cell balancing and additional protections such as over-current and short-circuit detection. (4) Hardware-in-the-loop simulation and quantitative tests, including calibration against a reference meter and logged temperature curves. (5) CAN, Bluetooth and cloud connectivity, with an improved enclosure and electrical isolation.

## REFERENCES

- [1] X. Feng, M. Ouyang, X. Liu, L. Lu, Y. Xia, and X. He, "Thermal runaway mechanism of lithium ion battery for electric vehicles: A review," *Energy Storage Mater.*, vol. 10, pp. 246–267, 2018.
- [2] Q. Wang, P. Ping, X. Zhao, G. Chu, J. Sun, and C. Chen, "Thermal runaway caused fire and explosion of lithium ion battery," *J. Power Sources*, vol. 208, pp. 210–224, 2012.
- [3] G. L. Plett, *Battery Management Systems, Volume I: Battery Modeling*. Norwood, MA, USA: Artech House, 2015.
- [4] M. Lelie et al., "Battery management system hardware concepts: An overview," *Applied Sciences*, vol. 8, no. 4, Art. no. 534, 2018.
- [5] K. W. E. Cheng, B. P. Divakar, H. Wu, K. Ding, and H. F. Ho, "Battery-management system (BMS) and SOC development for electrical vehicles," *IEEE Trans. Veh. Technol.*, vol. 60, no. 1, pp. 76–88, Jan. 2011.
- [6] M. A. Hannan, M. S. H. Lipu, A. Hussain, and A. Mohamed, "A review of lithium-ion battery state of charge estimation and management system in electric vehicle applications: Challenges and recommendations," *Renew. Sustain. Energy Rev.*, vol. 78, pp. 834–854, 2017.
- [7] D. Swamy, S. Sarkar, M. T. Amin, S. Vaddiraju, and F. Khan, "Analysis of lithium-ion battery thermal runaway models to design battery safety management systems," *ScienceDirect*, vol. 209, Jun. 2026.
- [8] R. Xiong, J. Cao, Q. Yu, H. He, and F. Sun, "Critical review on the battery state of charge estimation methods for electric vehicles," *IEEE Access*, vol. 6, pp. 1832–1843, 2018.
- [9] R. Xiong, L. Li, and J. Tian, "Towards a smarter battery management system: A critical review on battery state of health monitoring methods," *J. Power Sources*, vol. 405, pp. 18–29, 2018.

- 
- [10] M. Berecibar et al., "Critical review of state of health estimation methods of Li-ion batteries for real applications," *Renew. Sustain. Energy Rev.*, vol. 56, pp. 572–587, 2016.
  - [11] Z. Rao and S. Wang, "A review of power battery thermal energy management," *Renew. Sustain. Energy Rev.*, vol. 15, no. 9, pp. 4554–4571, 2011.
  - [12] M. K. Mahek, M. Ramadan, S. S. bin Dol, M. Ghazal, and M. Alkhdher, "A comprehensive review of thermoelectric cooling technologies for enhanced thermal management in lithium-ion battery systems," *Heliyon*, 2024, doi: 10.1016/j.heliyon.2024.e40649.
  - [13] C. Suddeepong, S. Nuchkum, N. Donjaroenon, and U. Leeton, "A battery management system capable of analyzing abnormal cell trends," *Energies (MDPI)*, 2026.
  - [14] A. K. Shaout and Z. Brauchler, "Fuzzy Battery Manager: Charging and balancing rechargeable battery cells with fuzzy logic," *Electronics*, vol. 14, no. 7, Art. no. 1470, 2025, doi: 10.3390/electronics14071470.
  - [15] Texas Instruments, "INA219 zero-drift, bidirectional current/power monitor with I<sup>2</sup>C interface," Datasheet SBOS448.
  - [16] Maxim Integrated, "DS18B20 programmable resolution 1-Wire digital thermometer," Datasheet.
  - [17] Espressif Systems, "ESP32 series datasheet," Espressif Systems, Shanghai, China.