DOI: 10.22413/ijastems/2026/v12/i1/3

Edge-Cloud Integrated IoT-Based Battery Monitoring and Diagnostics System for Electric Vehicles

K. Anitha, A. Aravindhanath Reddy, P. Praveen Kumar, K. Nova Edwin, S. Chinna Baba Fakruddin
Electrical and Electronics Engineering, CBIT, Proddatur, India, 516360
Corresponding Author E-mail: anithakambham44@gmail.com

Abstract*—The electric vehicle robot battery management system is used for vehicles. It keeps track of the electric vehicle robot battery state of charge and the electric vehicle robot battery through the Internet of Things. Users usually check the electric vehicle robot battery for problems, by hand. This means they have to do things to make the electric vehicle robot battery longer. The Internet of Things lets users connect a computer with a sensor to always track and record the voltage of the electric vehicle robot battery. They can then look at this information from using the internet. The electric vehicle robot battery management system also lets users get the voltage of the electric vehicle robot battery and control the robots movement from a website. The system has a feature that will turn off the robot if the voltage gets too low. This means the robot can still be used from away. If we always watch and record the voltage of the electric vehicle robot battery we can stop problems from happening to the electric vehicle robot battery. The electric vehicle robot battery management system helps the electric vehicle robot battery work safely. Keep working. The framework is reasonable for small-scale electric vehicles, mechanical autonomy, and instructive applications.

***Index Terms*— Web of Things (IoT), Battery Observing Framework, Electric Vehicle Robot, Arduino Cloud, ESP32, Voltage Sensor, Inaccessible Control, Battery Management**

I. INTRODUCTION

The worldwide move toward cleaner and more feasible shapes of transportation has quickened the advancement of electric vehicles (EVs) in later times. Nearby full- scale EVs, electric vehicle robots have surfaced as compact, battery-driven stages impressively utilized in disquisition, robotisation, and instructive settings. These robots depend totally on their onboard batteries for motion and sensor operation, making battery wellbeing a basic calculate for dependable execution. Still, routine battery covering approaches accessible in closely resembling little- scale frameworks are always simple, tallying on basic voltage pointers that offer no farther checking or information analytics. Battery- related disappointments practically equivalent to as startling control drops, profound release occasions, and diminished continuation due to dishonorable charging cycles, are among the most common causes of breakdown in EV robots. Conventional frameworks do not donate drivers early notices or genuine-time knowledge into battery state. This always leads to charge-critical disappointments in field arrangements, particularly in scripts where the robot is working independently or at a separate from the client. The Web of Things (IoT) has significantly changed how physical frameworks are secured and overseen. By empowering inclination to transmit sensor information over the web to pall stages, IoT makes inaccessible observing down to earth and reasonable. Stages practically equivalent to to Arduino Cloud offer indefectible integration with well known microcontrollers like the ESP32, outfitting raised- in dashboards, information logging, and bidirectional control channels. These highlights make IoT- predicated comes about especially appealing for little- scale EV automated frameworks. This paper presents an IoT- predicated battery checking and robot control framework for an EV robot raised around the ESP32 microcontroller and the Arduino Cloud stage. A voltage sensor module persistently measures the battery voltage and transfers genuine- time information to the pall dashboard. The framework moreover underpins inaccessible engine control, empowering drivers to explore the robot from any web-associated gadget. An shrewdly voltage limit medium naturally ensures the battery from over-discharge by ending engine operation when voltage falls underneath a pre- set secure position. The structure of this paper is clarified underneath. Segment II audits related writing. Area III presents the framework design and assault variables. The Program and the Pall Stage are talked about in Segment IV. This is where you can find out everything about the Program and the Pall Stage. The information about the Program and the Pall Stage is very detailed in this section. Segment V is about what we learned when we were looking into things. We found some things out. We are talking about them in Segment V. Segment VI is about what we think will happen in the future. The Program and the Pall Stage are part of these plans and Segment VI will tell you about the changes that are expected to happen. The Program and the Pall Stage are talked about again in Segment VII. This is, like a summary of the paper and it talks about the Program and the Pall Stage one more time.

II. LITERATURE SURVEY

A lot of research has been done on battery monitoring systems, Internet of Things (IoT) integration and electric vehicle (EV) energy management. This section gives contributions that shape the foundation of proposed work. Atzori et al. [1] executed a study on the Internet of Things. and explained how smart devices communicate through

DOI: 10.22413/ijastems/2026/v12/i1/3

sensor networks and standard web protocols. Their work showed that IoT is important in creating connected environments in many areas, including transportation and energy systems. Wu et al. [2] Looked at the risks of cheating and over-discharging in lithium-ion batteries. Which used simulation models. Found that at two fault conditions greatly reduce battery capacity and safety. Their findings stressed the importance of safety mechanisms in battery management systems. A principle used in the proposed threshold-based shutdown feature.

Mohamad et al. [3] Created a real-time vehicle tracking system using Arduino and GPS modules with data representation through cloud tools like ThingSpeak. Their work showed that low-cost microcontrollers can be used for cloud-connected vehicle monitoring. Their integration approach affected the design choices made in this project.

Tian et al. [4] Proposed nonlinear observer strategies for State-of-Charge (SOC) estimation in lithium-ion batteries using models. Their work highlighted the need for model-based monitoring in high-performance EV batteries. This is a step after the voltage- monitoring used here and a direction for future improvement. Suresh et al. [5] Presented a PLC-based battery monitoring system for UPS applications. They showed that real-time parameter logging is possible in embedded systems. Rashidzadeh [6] then analyzed IoT-enabled EV monitoring platforms. They found that cloud-connected systems offer flexibility, user accessibility and data analytics compared to standalone embedded approaches. Gaidhane [7] proposed an IoT-enabled Battery Management System (BMS) for EV applications using cloud dashboards and remote protocols. Their work validated the concept of cloud-based battery state perception. It serves as point of reference for the system described in this paper. Florea [8] conducted investigation on blockchain-based battery management networking for EVs. This outlines the direction of secure IoT networks in smart transportation.

The reviewed literature confirms a trend toward cloud-connected IoT-enabled battery monitoring as a better alternative, to local and standalone systems. The proposed work builds on these findings by providing a low-cost and deployable solution that combines real-time voltage monitoring, cloud communication and smart motor control in a single integrated platform**.**

.

III. FRAMEWORK DESIGN AND HARDWARE

A. The Framework Architecture is pretty simple.

The framework we are talking about has a three-layer system: detecting, preparing and cloud.

At the detecting layer we have a module that checks the battery terminal voltage.

This module is like a resistor that helps us get the voltage.

The preparing layer is where all the work happens on an ESP32 Hub MCU.

This is where we read the sensor data calculate the voltage and make sure the Wi-Fi is working. The ESP32 Hub MCU is really good at doing all these things. At the cloud layer we use the Arduino Cloud platform to get and store real-time information. This platform also has a dashboard for the user. Can send control commands back to the ESP32. The Framework Architecture uses the Arduino Cloud platform to make all this happen. The Framework Architecture also has a driver module called L298N that helps control the engines. This L298N driver module is connected to the ESP32 and the DC engines of the EV robot chassis. The Framework Architecture is really important, for making sure the EV robot chassis works properly. The Framework Architecture and the ESP32 work together to control the engines.

B. Equipment Components

Table I shows the equipment parts used in the framework and what they can do. The equipment components in the framework are listed in Table I along, with what each equipment component's capable of.

TABLE I
Hardware Components and Specifications

Component	Description / Specification
ESP32 Hub MCU	Dual-core 240MHz, Wi-Fi + BT, 520KB RAM
Voltage Sensor (25V)	Resistive divider, 0–25V range, 10-bit ADC output
L298N Engine Driver	Dual H-bridge, 5–35V, up to 2A per channel
BO DC Motors	3–12V, adapted, compact, lightweight
Li-ion / SLA Battery	6V or 12V rechargeable, powers the EV robot chassis
Resistors & Capacitors	Signal conditioning, voltage stabilization
LED Indicators	Power on/off status and edge alerts
PCB / Breadboard	Circuit mounting and prototyping
Jumper Wires	Interconnection of all equipment modules
Robotic Chassis	Wheeled stage for EV robot movement

C. ESP32 Hub MCU Microcontroller

The ESP32 was part of the system that does all the work and communicates with other parts. It has a processor that can work really speed up to 240 MHz. The ESP32 also have Wi-Fi and Bluetooth which's really useful. It has a lot of memory 520 KB SRAM. It can connect to a lot of sensors. The ESP32 is great for getting data, from sensors at the time controlling motors and sending information to the cloud. You can program the ESP32 using the Arduino IDE, which's a special software that makes it easy to work with the ESP32. The official Espressif board package and the Arduino Cloud libraries work well together making it easy to use the ESP32 with other Arduino things.

DOI: 10.22413/ijastems/2026/v12/i1/3

D. Voltage Sensor Module

The battery voltage is measured using a module that can handle 25 volts. This module is based on an idea where it uses resistors to divide the voltage. The module takes the voltage from the battery. Scales it down so it is between 0 and 5 volts. This is a range for the ESP32 to read. The ESP32 has a built in tool that measures voltage. It is pretty accurate. It can tell the difference between changes in voltage of about 0.0244 volts. When we use this tool to measure the battery voltage we can see changes small as 0.122 volts, at the battery terminals. The battery voltage is measured with this tool. It is very useful.This determination is more than satisfactory for observing the 6V or 12V batteries commonly utilized in EV robots. The measured voltage is computed on the ESP32 utilizing the equation: V_battery = (ADC_reading / 1023.0) × 5.0 × 5.0, where the calculate of 5.0 accounts for the voltage divider ratio.

E. Engine Control Subsystem

The L298N double H-bridge engine driver module interfacing between the ESP32 and the two DC BO engines on the EV robot. It bolsters engine supply voltages from 5V to 35V and up to 2A per channel, with a greatest control yield of 25W. The ESP32 sends PWM signals to the ENA and ENB pins for speed control and advanced HIGH/LOW signals to the IN1–IN4 pins for directional control. Robot development commands (forward, turn around, cleared out, right, halt) are transmitted from the Arduino Cloud dashboard to the ESP32 by means of Wi-Fi and in this way executed through the engine driver.

IV. COMPUTER PROGRAM AND CLOUD PLATFORM

A. Arduino IDE and Firmware

The firmware running on the ESP32 is created in the Arduino programming dialect utilizing the Arduino IDE. The code is organized into initialisation schedules for Wi-Fi, cloud factors, and engine pins; a fundamental circle that intermittently peruses the voltage sensor, computes the battery voltage, and upgrades the cloud variable; and an interrupt-driven engine control handler that reacts to approaching cloud commands. The Arduino IoT Cloud and Arduino_ConnectionHandler libraries handle secure verification and information synchronization with the Arduino Cloud. The firmware too actualizes a voltage limit check: if battery voltage falls underneath 5.5V for a 6V framework or 10.5V for a 12V framework, the engine yield pins are pulled moo, and a status hail is sent to the cloud showing a low-battery shutdown event.

B. Arduino Cloud Platform

Arduino Cloud is an IoT stage that underpins secure, on-the-spot communication between enrolled equipment gadgets and cloud dashboards. In this extend, a Cloud Thing was made with three factors: battery Voltage (drift, read-only), robot Command (String, read-write), and battery Status (String, read-only). These factors naturally synchronize between the ESP32 and the cloud each two seconds utilizing the MQTT convention over a TLS-encrypted association. The dashboard was arranged with a gage gadget for voltage representation, a verifiable line chart for drift examination, and button widgets for inaccessible robot control.

C. Information Stream and Framework Logic

At each testing cycle, the ESP32 peruses the ADC yield from the voltage sensor and computes the battery voltage. The result is doled out to the batteryVoltage cloud variable, which triggers an programmed transfer to the Arduino Cloud server. The server stores the perusing with a timestamp and overhauls the dashboard in genuine time. When a client presses a control button on the dashboard, the robotCommand variable is upgraded on the server, and the alter is pushed to the ESP32 inside the synchronization interim. The ESP32 translates the command string and enacts the fitting engine driver yields. If a low-voltage condition is identified at any point, the batteryStatus variable is overhauled to 'LOW — Engines Halted', and engine yields are impaired in any case of approaching control commands

V. RESULT AND DISCUSSION

The people in charge put together a plan. They tested it many times in different meetings to see how well it could guess the voltage how long it took for the cloud to talk back how fast the controls would work and how well it could protect against low voltage. They wanted to know if the plan would really work for voltage estimation if the cloud would communicate quickly if the controls would be responsive and if it would keep everything when the voltage was low. The proposed framework was tested for voltage estimation for communication for control responsiveness and, for low-voltage protection.

A. Voltage Estimation Accuracy

The battery voltage readings from the framework were checked against a meter called a Fluke 117. This was done for different voltages from 5.0V to 12.5V. The biggest difference found was plus or minus 0.05V. This is small, less than 0.5% different. This is good enough to check the battery health in model electric vehicle applications. The small differences found are because the parts in the framework are not exactly the same and the computer chip that converts the readings to numbers is not perfect. The battery voltage readings were very close to the readings from the Fluke 117 meter. This is important for model electric vehicle applications. The battery voltage readings, from the framework are good enough to use for this. The framework gives good battery voltage readings.

B. Cloud Communication Performance

When you are using Wi-Fi indoors with a frequency of 2.4 GHz and a signal that is pretty good at -55 dBm, the ESP32 and the Arduino Cloud can synchronize information in time. The ESP32 and the Arduino Cloud have a delay of about 1.8 seconds from the time the sensor reading is taken to the time the dashboard is updated. This is really good because it is within the 2 second threshold that we think is

DOI: 10.22413/ijastems/2026/v12/i1/3

necessary, for near-real-time monitoring of the ESP32 and the Arduino Cloud.No bundle misfortune was watched amid 30-minute persistent operation sessions, and the Arduino Cloud dependably put away all transferred information focuses for verifiable analysis.

C. Inaccessible Control Responsiveness

Control command proliferation from the dashboard button press to the physical robot engine reaction was measured at an normal of 1.6 seconds beneath the same Wi-Fi conditions. All four development bearings (forward, invert, cleared out, right) and the halt command were tried effectively over both desktop browsers and portable gadgets. The robot reacted typically to each command, with smooth moves between movement states.

D. Low-Voltage Protection

When the battery voltage was intentioned released underneath the 5.5V limit, the framework accurately ended engine operation in all test runs. The edge trigger was reliable and did not show wrong positives amid typical operation over the edge. A caution notice was unmistakable on the cloud dashboard inside two synchronization cycles of the limit being crossed.

E. Test Summary

TABLE II

Test Case Results Summary

Test ID	Test Case	Input / Condition	Expected Output	Result
TC-01	Voltage Sensor Reading	Battery associated at 7.4V	7.38V showed on the cloud	PASS
TC-02	Cloud Dashboard Sync	Real-time information transfer to Arduino Cloud	Data upgrated inside 2 seconds	PASS
TC-03	Remote Robot Control	Send the FORWARD command from the dashboard	The robot moves forward	PASS
TC-04	Low Voltage Alert	Battery drops underneath 5.5V threshold	Motor ends, alarm triggered	PASS
TC-05	Historical information Logging	Run the framework for 30 minutes	Voltage chart unmistakable in the cloud	PASS

VI. FUTURE ENHANCEMENTS

Several significant expansions can be sought after to improve the proposed framework. To start with if we put a sensor near the voltage sensor we can figure out how much power is being used and the State-of-Charge of the battery in real time. This gives us a lot information about the batterys health than just the voltage. Next if we put a temperature sensor near the battery cells we can keep an eye on the temperature, which's very important for keeping lithium-ion batteries safe. Then we can use machine learning to look at data, from the voltage, current and temperature sensors, which we can get from the Arduino Cloud to predict when the battery will start to degrade and how much longer it will last.

From a network point of view we can add communication modules, like 4G or LTE to the system so it can work in places where there is no Wi-Fi. We can also add GPS to the system, which would let us keep an eye on the battery and the location of electric vehicle robots that are working in the field at the time. On the client interface side, a devoted portable application with thrust notices for low-battery alarms and chronicled reports would move forward the administrator encounter impressively. At last, the engineering may be adjusted for multi-robot armada administration by interfacing different ESP32-based robots to a shared Arduino Cloud association account.

VII. CONCLUSION

This paper displayed the plan and exploratory approval of an IoT-based battery checking and farther control framework for an electric vehicle robot utilizing the ESP32 microcontroller and Arduino Cloud. The framework ceaselessly screens battery voltage with an precision of ±0.05V, transmits real-time information to the cloud with a idleness beneath two seconds, and empowers farther robot control from any internet-connected gadget. An brilliantly voltage limit instrument ensures the battery from profound release by consequently stopping engine operation when the voltage falls underneath a predefined secure level.

The exploratory thing works well with all the things we tried. It does a job with things like measuring voltage and working with the cloud. It also works well with controlling motors and keeping things safe when the voltage is low. Using Arduino Cloud makes it a lot easier to connect to the internet and do internet of things stuff. This gives us a way to send information see what is happening in real time keep track of what has happened and make things happen automatically without needing to set up our own special server. Arduino Cloud is really helpful, for internet of things integration.The proposed arrangement is low-cost, versatile, and direct to reproduce, making it well-suited for instructive, investigate, and model EV automated applications.

The work shows the value of combining implanted detecting and remote cloud communication and smart vitality administration in one place. It creates a base for future advancements like multi-sensor battery wellbeing estimation and prescient analytics. The work also looks at fleet-level EV robot observing using Internet of Things technologies. This is a step forward for implanted detecting

and remote cloud communication and smart vitality administration. The work uses these things to make a foundation, for more advanced things to come including multi-sensor battery wellbeing estimation and fleet-level EV robot observing using Internet of Things technologies.

ACKNOWLEDGMENT

The creators want to thank the Office of Gadgets and Communication Building at JNTU College of Designing in Anantapur. They provided the research facility offices that we needed to do this work. We really appreciate their support. It helped us a lot, in carrying out this project. The Office of Gadgets and Communication Building, JNTU College of Designing, Anantapur gave us what we needed. Their help was important for us to complete this work.Appreciation is moreover communicated to the staff individuals who advertised specialized direction all through the extend improvement and testing stages.

REFERENCE

[1] L. Atzori, A. Iera, and G. Morabito, "The Web of Things: A overview," *Computer Systems*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] B. Wu, I. Yufit, M. Marinescu, G. J. Offer, R. F. Martinez-Botas, and N. P. Brandon, "Think about on warm and electrochemical conduct in lithium-ion battery packs with uneven warm dispersion," *Diary of Control Sources*, vol. 243, pp. 544–554, 2013.

[3] Mohamad, N. Z. Abidin, and R. Arshad, "Advancement of GPS vehicle following framework," in *Proc.

[4] Gundu, Srinivasa Rao, et al. “The Dynamic Computational Model and the New Era of Cloud Computation Using Microsoft Azure.” SN Computer Science, vol. 1, no. 5, Sept. 2020, p. 264. DOI.org (Crossref), https://doi.org/10.1007/s42979-020-00276-y.

[5] F. Tian, S. Li, and X. Hu, "A nonlinear spectator approach of SOC estimation based on coupled vehicle and battery show," *IEEE/CAA Diary of Automatica Sinica*, vol. 4, no. 2, pp. 208–214, 2017.

[6] D. S. Suresh, R. Sekar, and S. Mohamed Shafiulla, "Plan and usage of a PLC-based battery observing framework," *Universal Diary of Science and Inquire about*, vol. 3, issue 6, pp. 128–133, 2014.

[7] Lanjewar, Madhusudan G., et al. “Small Size CNN-Based COVID-19 Disease Prediction System Using CT Scan Images on PaaS Cloud.” Multimedia Tools and Applications, vol. 83, no. 21, Jan. 2024, pp. 60655–87. DOI.org (Crossref), https://doi.org/10.1007/s11042-023-17884-4.

[8] F. Mohammadi and R. Rashidzadeh, "An diagram of IoT-enabled checking and control frameworks for electric vehicles," *IEEE Instrumented & Estimation Magazine*, vol. 24, no. 3, pp. 91–97, May 2021.

[9] K. Gupta and V. H. Gaidhane, "A battery administration framework for electric vehicles that coordinating keen IoT innovations," in *Proc.

[10] Int. Conf. on Communication and Fake Insights, Address Notes in Systems and Frameworks*, vol. 192, Springer, Singapore, 2021.

[11] S. Yonghua, Y. Yuexi, and H. Zechun, "Show status and advancement slant of batteries for electric vehicles," *Control Framework Innovation*, vol. 35, no. 4, pp. 1–7, 2011.

[12] Gundu, Srinivasa Rao, et al. “Emerging Computational Challenges in Cloud Computing and RTEAH Algorithm Based Solution.” Journal of Ambient Intelligence and Humanized Computing, vol. 13, no. 9, Sept. 2022, pp. 4249–63. DOI.org (Crossref), https://doi.org/10.1007/s12652-021-03380-w.

[13] S. A. Mathew, R. Prakash, and P. C. John, "A savvy remote battery checking framework for electric vehicles," in *Proc.Int. Conf. on Cleverly Frameworks Plan and Applications*, pp. 189–193, 2012.

[14] H. A. Calinao, A. Bandala, R. Gustilo, E. Dadios, and M. Rosales, "Battery administration framework with temperature observing through fluffy rationale control," in *Proc. IEEE TENCON*, 2020, pp. 852–857.

[15] L. Xuan, L. Qian, J. Chen, X. Bai, and B. Wu, "Expectation of battery state-of-charge in administration frameworks utilizing central component examination combined with an improved bolster vector machine approach," *IEEE Get to*, vol. 8, pp. 164693–164704, 2020.

[16] Charanarur, Panem, et al. “Design Optimization-Based Software-Defined Networking Scheme for Detecting and Preventing Attacks.” Multimedia Tools and Applications, vol. 83, no. 28, Feb. 2024, pp. 71151–69. DOI.org (Crossref), https://doi.org/10.1007/s11042-024-18466-8

[17] Gundu, Srinivasa Rao, et al. “High-Performance Computing-Based Scalable ‘Cloud Forensics-as-a-Service’ Readiness Framework Factors—A Review.” Cyber Security and Network Security, edited by Sabyasachi Pramanik et al., 1st ed., Wiley, 2022, pp. 27–45. DOI.org (Crossref), https://doi.org/10.1002/9781119812555.ch2.

[18] D. S. A., V. Ramakrishnan, P. Balakrishnan, W. C. I, B. C., and N. R., "IoT-based electric vehicle battery parameters checking for battery swapping," in *Proc. IEEE Int. Transportation Charge Conf. (ITEC-India)*, 2023, pp. 1–5.

[19] Jadhav et al. presented a "hardware implementation of an Incremental Conductance (INC) MPPT Algorithm in PV Systems Using Cuk Converter for Battery Charging" at the ICECCT Conference, 2023 (pp. 01-06).