2025 TEEP Progress Report Week-21

Single-Cell Li-ion Battery Monitoring System Using Flask Server Report

1. Introduction

This project focuses on the design and implementation of a **single-cell Li-ion battery monitoring system** using an ESP32 microcontroller, an INA219 current/voltage sensor, and a TP4056 charging module. The objective is to accurately monitor battery voltage, estimate the state of charge (SoC), detect charging status, and visualize the data in real time using a web-based dashboard built with Flask.

The system is designed as a foundation for a Battery Management System (BMS) suitable for low-power embedded and IoT applications. The mobile application component is planned for future work and is **not included in this phase** of the project.

---

2. System Objectives

The main objectives of this project are:

- To measure the battery voltage accurately using the INA219 sensor.
- To estimate battery percentage (state of charge) based on voltage levels.
- To monitor charging behavior through the TP4056 charging module.
- To transmit battery data wirelessly using Wi-Fi.
- To display battery information on a professional, responsive Flask-based dashboard.
- To generate system logs for debugging and monitoring purposes.

3. Hardware Components

3.1 ESP32 Microcontroller

The ESP32 is used as the main controller due to its:

- Built-in Wi-Fi capability
- Low power consumption
- Compatibility with MicroPython
- Sufficient GPIO and I2C support

3.2 INA219 Current and Voltage Sensor

The INA219 is a high-side current and voltage monitoring IC. In this project, it is used to:

- Measure the battery voltage
- Provide the basis for current and power measurement (future enhancement)

The sensor communicates with the ESP32 via the I2C protocol.

3.3 TP4056 Li-ion Charging Module

The TP4056 module is responsible for safely charging the single-cell Li-ion battery. It provides:

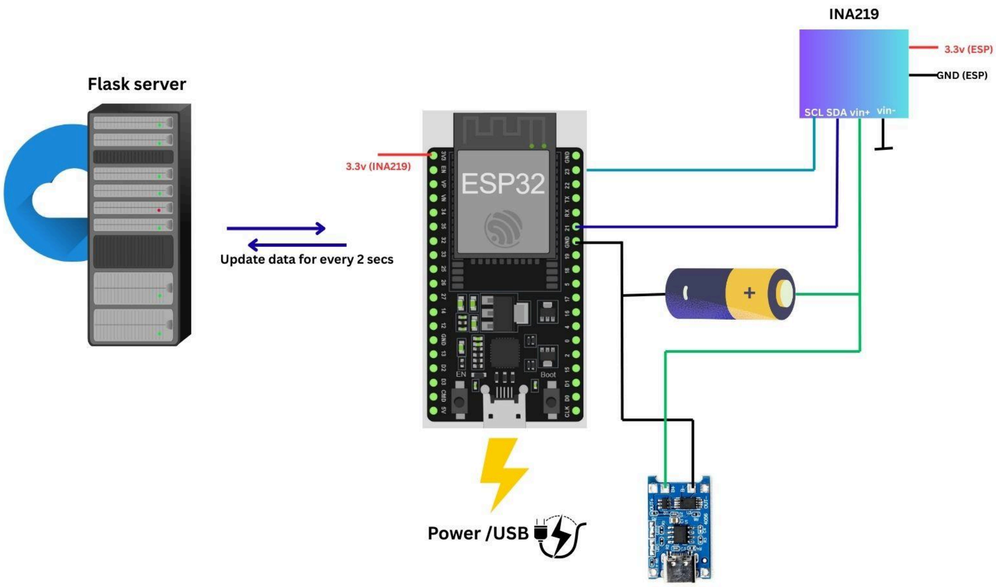
- Constant-current / constant-voltage (CC/CV) charging
- Overcharge protection
- Status indication (charging / full)

3.4 Li-ion Battery

A single-cell 3.7V Li-ion battery is used. The typical operating voltage range considered is:

- Minimum voltage: 3.0 V
- Maximum voltage: 4.2 V

4. Hardware Connections



4.1 INA219 Connections

- **VCC** → ESP32 3.3 V
- **GND** → ESP32 GND and Battery Negative (BAT−)
- **SDA** → ESP32 GPIO 21
- **SCL** → ESP32 GPIO 22
- **VIN+ (V+)** → Battery Positive (BAT+)

4.2 TP4056 Connections

- **B+** → Battery Positive (BAT+)
- **B−** → Battery Negative (BAT−)
- **OUT+ / OUT−** → Load / system power path

This configuration allows safe charging while enabling accurate voltage monitoring.

5. Software Architecture

The software system consists of two main parts:

1. **ESP32 firmware (MicroPython)**
2. **Flask web server and dashboard**

---

6. ESP32 Firmware Design

6.1 Development Environment

- Language: MicroPython
- IDE: Thonny
- Communication protocol: HTTP (JSON payload)

6.2 INA219 Driver

A custom MicroPython-compatible INA219 driver is used. The driver was modified to remove unsupported modules (such as `logging`) to ensure full compatibility with the ESP32 environment.

6.3 Voltage Sampling and Averaging

To reduce noise and improve stability:

- 20 voltage samples are collected per cycle
- A short delay is used between samples
- The average voltage is calculated and used for further processing

6.4 Battery Percentage Calculation

Battery percentage is estimated using a linear voltage-based model:

- 0% → 3.0 V
- 100% → 4.2 V

The calculated percentage is clamped between 0% and 100% to ensure safe values.

6.5 Wi-Fi Communication

The ESP32 connects to a local Wi-Fi network and sends battery data periodically to the Flask server using HTTP POST requests.

6.6 Serial Monitoring

All critical values such as voltage, percentage, and connection status are printed to the Thonny console for debugging and validation.

---

7. Flask Server and Dashboard

7.1 Flask Backend

The Flask server receives JSON data from the ESP32, including:

- Battery voltage
- Battery percentage
- Charging status (basic implementation)

The server processes this data and makes it available to the dashboard in real time.

7.2 Dashboard Features

The web-based dashboard includes:

- Real-time battery percentage display
- Battery voltage graph
- Charging curve visualization (0–100%)
- System command log (ESP32 → server messages)
- Battery information widget explaining Li-ion battery behavior
- Responsive layout for desktop and tablet screens

7.3 Alert System

The dashboard generates visual alerts at predefined battery levels:

- 25%, 50%, 75% → informational pop-ups
- 90% → warning message (remove charging soon)
- 95% → critical alert (disconnect charger)

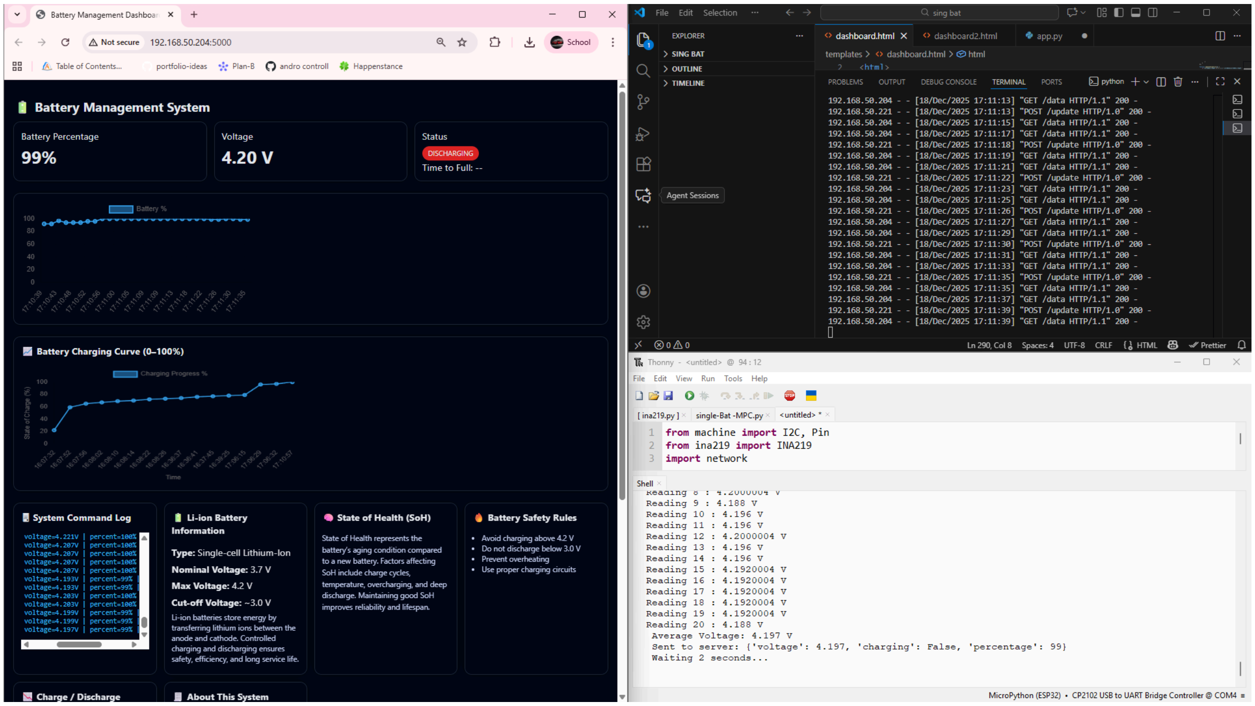
8. Code Files

Click bellow link and get code files :

- [Battery monitoring system using flask server code files Github]

8. Testing and Results

The system was tested under charging and discharging conditions. Observations include:



- Stable voltage readings after averaging
- Reliable Wi-Fi communication
- Correct percentage estimation within expected voltage ranges
- Real-time updates on the dashboard with minimal latency

Click the below link to watch the output video

- https://youtu.be/0CwqLIPdDpg

---

11. Conclusion

This project successfully demonstrates a **single-cell Li-ion battery monitoring system** using the ESP32, INA219, and TP4056 modules. The system provides reliable voltage monitoring, battery percentage estimation, wireless data transmission, and real-time visualization through a professional Flask dashboard.

https://www.notion.so/Battery-Manabement-System-2d3f865118d48057995ac57cb65b382b?source=copy_link