# 2025 TEEP Progress Report Week-10

Used Wokwi to complete the tasks for Labs 1 and 2

(Before creating the physical project, please use **Wokwi** for online simulation to complete the initial design of the circuit and code. Then, use **Thonny** together with the **ESP32** development board to build the actual project. Please **record a video of the project** in operation, upload it to YouTube, and provide the video link(LAB7) so we can better understand your learning progress.)

## Lab1 – On ESP32, complete the following tasks:

This project focuses on optimizing the 12V battery charging monitoring system, which was completed last week using three 3.7V Li-Ion cells connected in series. The aim is to enhance the overall stability of measurement and display. Each step of the experimental process must be thoroughly documented, compiled into the project report, and also presented in video format to ensure the completeness and traceability of the research process.

Please record the experimental process in the Experimental Record, which should include the following:

**Date:** 02 / oct / 2025

**Course/Project Title: Development of IOT Situation room for elevator operation monitoring and health diagnosis.**

**1. Experiment Title :** Battery monitoring and real-time data visualization using ESP32, multi-cell Li-Ion pack, external ADC, and Flask server.

### 2. Objective

The purpose of this experiment is to design and implement a real-time battery monitoring system capable of:

- Measuring individual Li-Ion cell voltages and total pack voltage using ADS1115.

- Measuring charging/discharging current using INA219.
- Calculating battery state-of-charge (SoC) and monitoring pack status.
- Uploading live data to a custom Flask server and visualizing it on a web dashboard.
- Investigating connectivity and measurement accuracy issues during implementation.

## 3. Materials and Equipment

### Hardware:

- ESP32 module
- 3 × 3.7V Li-Ion cells in series (3S configuration)
- BMS module for 3S pack protection
- 3S Li-Ion charging module
- ADS1115 16-bit external ADC
- INA219 current sensor
- Connecting wires, resistors for voltage dividers

### Software/Tools:

- MicroPython firmware on ESP32
- Thonny IDE
- Python 3.13 on Windows 10
- Flask 3.1.2 framework for web dashboard
- Web browser for dashboard visualization

## 4. Connections and wiring :

### 1. Battery Pack and BMS

- Connect three 3.7V Li-Ion cells in series to form a 3S battery pack.
- Connect the battery pack to the BMS input terminals.
- Connect BMS output terminals to the voltage dividers, INA219, and ESP32 power supply.
- Connect the 3S charging module to the BMS-protected battery pack for safe charging.

### 2. Voltage Dividers (for ADS1115 ADC)

- Cell1 voltage divider output → ADS1115 channel A0.
- Cell1 + Cell2 voltage divider output → ADS1115 channel A1.
- Full pack voltage divider output (Cell1 + Cell2 + Cell3) → ADS1115 channel A2.
- Ensure the divider resistors reduce the voltage to be within the 0–4.096V range for ADS1115 input.
- Connect the common GND of voltage dividers to ESP32 GND.

### 3. ADS1115 Connections

- VDD → 3.3V from ESP32.
- GND → ESP32 GND.
- SDA → ESP32 GPIO21.
- SCL → ESP32 GPIO22.
- Connect A0, A1, A2 to respective voltage divider outputs.

### 4. INA219 Current Sensor

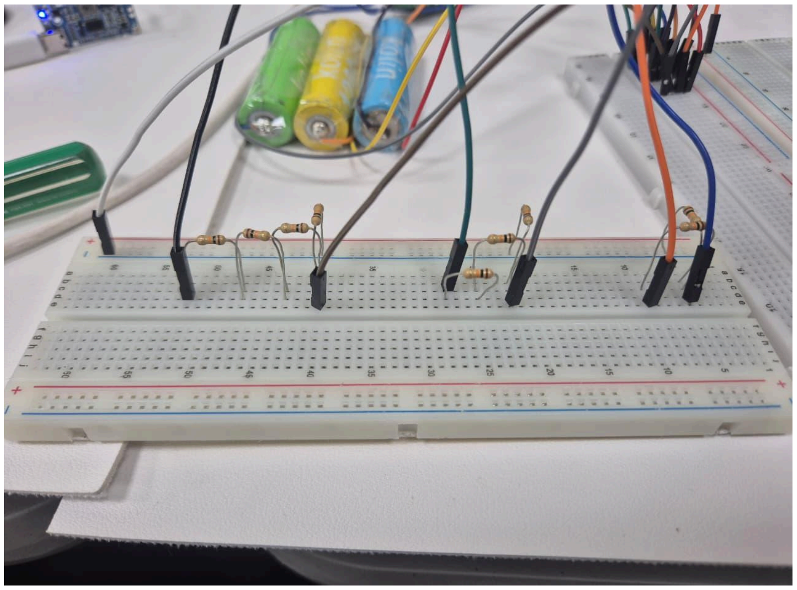
- VCC → 3.3V from ESP32.
- GND → ESP32 GND.
- SDA → ESP32 GPIO21 (same I²C bus as ADS1115).
- SCL → ESP32 GPIO22 (same I²C bus as ADS1115).
- Place the INA219 in series with the positive terminal of the battery pack or charger to measure current.

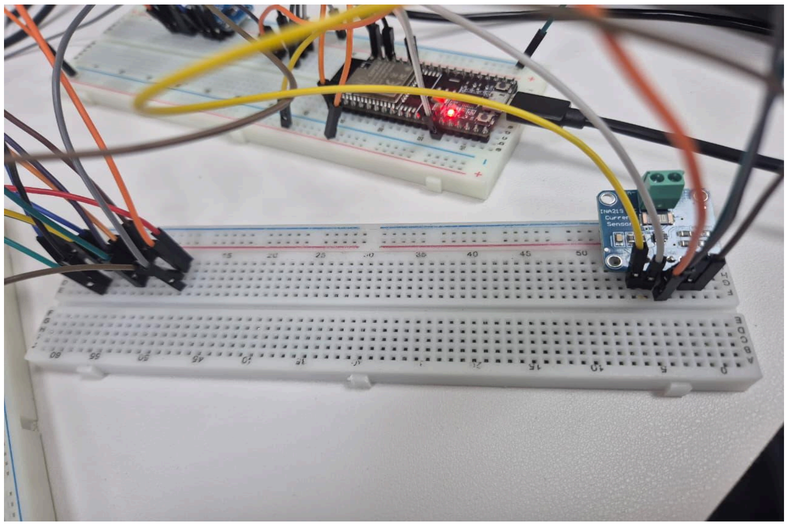
### 5. ESP32 Power

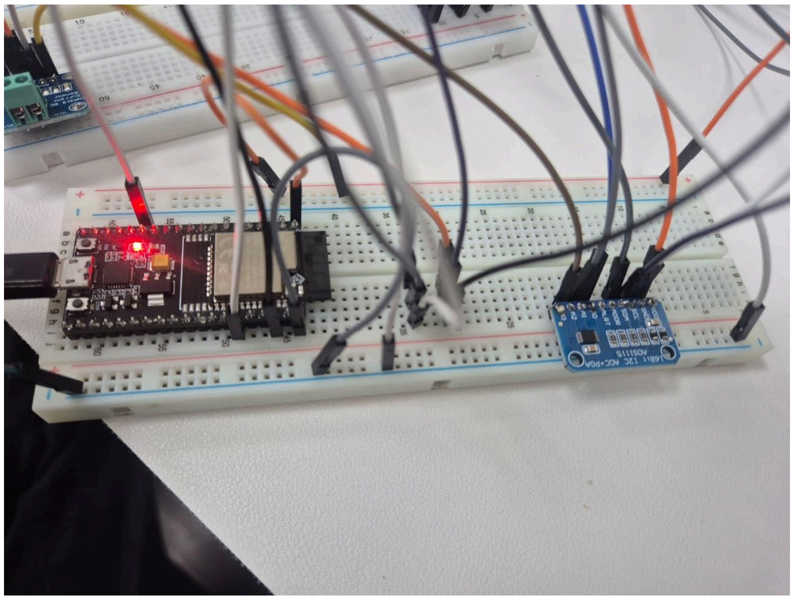
- ESP32 powered via 3.3V regulated supply from the BMS-protected pack or USB connection.
- Connect ESP32 GND to common ground shared with voltage dividers, ADS1115, and INA219.

### 6. Wi-Fi and Flask Server

- ESP32 connects to local Wi-Fi to send JSON data.
- Flask server runs on PC connected to the same network to receive data and display on web dashboard.







## 5. Procedure

1. **Power system setup**
    - Connected three Li-Ion cells in series.
    - BMS installed to protect against overcharge, over-discharge, and cell imbalance.
    - Charger module connected for safe charging.
2. **Voltage sensing**
    - Constructed voltage dividers to measure:
        - Cell1 → ADS1115 A0
        - Cell1+Cell2 → ADS1115 A1
        - Full pack → ADS1115 A2

    - Connected ADS1115 to ESP32 via I²C (SDA=GPIO21, SCL=GPIO22).

3. **Current sensing**
    - INA219 placed in series with pack/charger to measure current.
    - Connected INA219 to same I²C bus.

4. **ESP32 programming**
    - Wrote MicroPython code to read ADS1115 and INA219 values directly over I²C.
    - Calculated individual cell voltages, pack voltage, current, and battery SoC.
    - Configured ESP32 to connect to Wi-Fi and send JSON payloads to Flask server every 2 seconds.

5. **Flask server setup**
    - Installed Flask 3.1.2 on Windows.
    - Created `app.py` with endpoints to receive JSON data and render web dashboard.
    - Created `dashboard.html` template to display real-time voltages, current, and SoC.

6. **Data transmission and monitoring**
    - Ran ESP32 code and Flask server simultaneously.
    - Observed data being sent but with errors (`ECONNABORTED`) and fluctuating/incorrect voltage readings.

## 6. Results

ESP32 successfully connected to Wi-Fi:

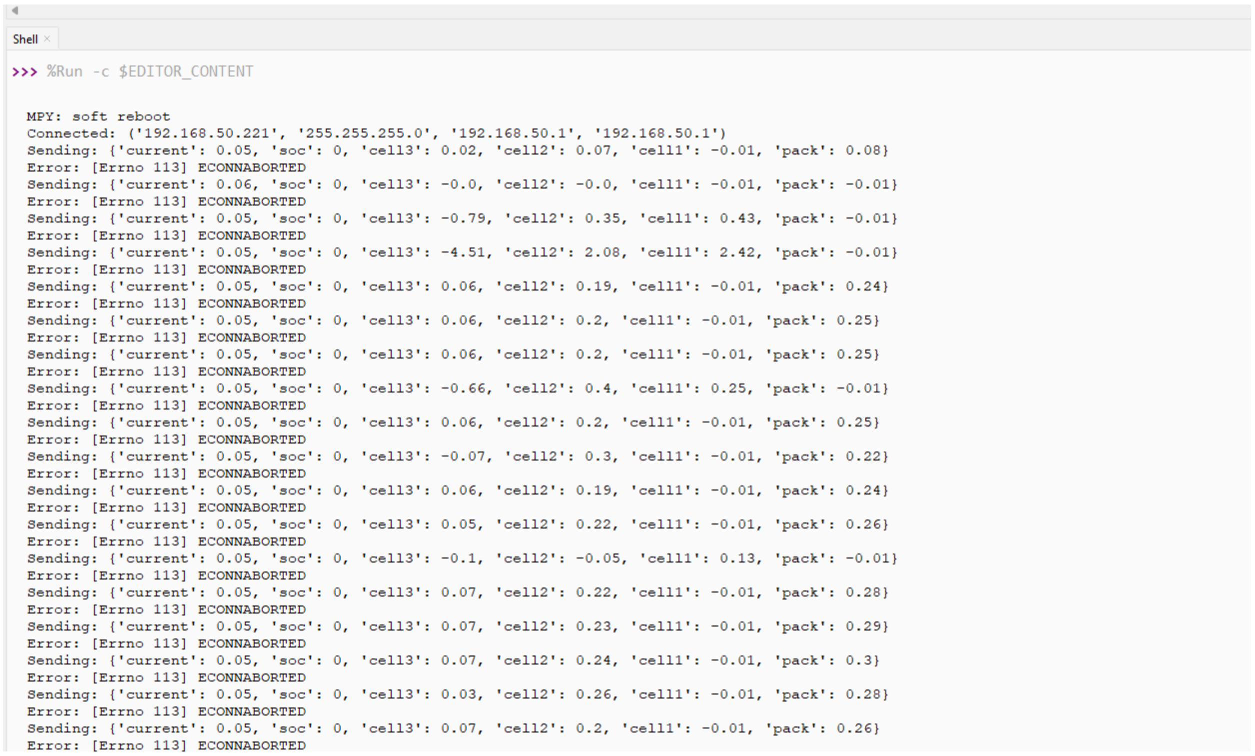
```
Connected: ('192.168.50.221', '255.255.255.0', '192.168.50.1',
'192.168.50.1')
```

- Attempted JSON transmissions from ESP32:

```
Sending: {'current': 0.05, 'soc': 0, 'cell3': 0.02, 'cell2': 0.07,
'cell1': -0.01, 'pack': 0.08}
Error: [Errno 113] ECONNABORTED
...
Sending: {'current': 0.05, 'soc': 0, 'cell3': -0.1, 'cell2': -0.05,
'cell1': 0.13, 'pack': -0.01}
```

- Observed:
    - Individual cell voltage readings are inaccurate, sometimes negative or fluctuating.

- SoC remains at 0% due to voltage calculation errors.
- Connection errors (`ECONNABORTED`) occurred when ESP32 attempted to post to Flask server.



Shell

```
>>> %Run -c $EDITOR_CONTENT

  MPY: soft reboot
  Connected: ('192.168.50.221', '255.255.255.0', '192.168.50.1', '192.168.50.1')
  Sending: {'current': 0.05, 'soc': 0, 'cell3': 0.02, 'cell2': 0.07, 'cell1': -0.01, 'pack': 0.08}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.06, 'soc': 0, 'cell3': -0.0, 'cell2': -0.0, 'cell1': -0.01, 'pack': -0.01}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.05, 'soc': 0, 'cell3': -0.79, 'cell2': 0.35, 'cell1': 0.43, 'pack': -0.01}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.05, 'soc': 0, 'cell3': -4.51, 'cell2': 2.08, 'cell1': 2.42, 'pack': -0.01}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.05, 'soc': 0, 'cell3': 0.06, 'cell2': 0.19, 'cell1': -0.01, 'pack': 0.24}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.05, 'soc': 0, 'cell3': 0.06, 'cell2': 0.2, 'cell1': -0.01, 'pack': 0.25}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.05, 'soc': 0, 'cell3': 0.06, 'cell2': 0.2, 'cell1': -0.01, 'pack': 0.25}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.05, 'soc': 0, 'cell3': -0.66, 'cell2': 0.4, 'cell1': 0.25, 'pack': -0.01}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.05, 'soc': 0, 'cell3': 0.06, 'cell2': 0.2, 'cell1': -0.01, 'pack': 0.25}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.05, 'soc': 0, 'cell3': -0.07, 'cell2': 0.3, 'cell1': -0.01, 'pack': 0.22}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.05, 'soc': 0, 'cell3': 0.06, 'cell2': 0.19, 'cell1': -0.01, 'pack': 0.24}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.05, 'soc': 0, 'cell3': 0.05, 'cell2': 0.22, 'cell1': -0.01, 'pack': 0.26}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.05, 'soc': 0, 'cell3': -0.1, 'cell2': -0.05, 'cell1': 0.13, 'pack': -0.01}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.05, 'soc': 0, 'cell3': 0.07, 'cell2': 0.22, 'cell1': -0.01, 'pack': 0.28}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.05, 'soc': 0, 'cell3': 0.07, 'cell2': 0.23, 'cell1': -0.01, 'pack': 0.29}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.05, 'soc': 0, 'cell3': 0.07, 'cell2': 0.24, 'cell1': -0.01, 'pack': 0.3}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.05, 'soc': 0, 'cell3': 0.03, 'cell2': 0.26, 'cell1': -0.01, 'pack': 0.28}
  Error: [Errno 113] ECONNABORTED
  Sending: {'current': 0.05, 'soc': 0, 'cell3': 0.07, 'cell2': 0.2, 'cell1': -0.01, 'pack': 0.26}
  Error: [Errno 113] ECONNABORTED
```

## 7. Discussion/Analysis

- **Current issues identified:**
  1. Voltage dividers may need adjustment or recalibration to match ADC input range.
  2. Direct ADS1115 and INA219 I²C reads without external drivers may introduce timing errors or incorrect conversions.
  3. Flask server received few packets successfully; network errors may be caused by Wi-Fi instability or ESP32 post timing.

- **Next steps for improvement:**
  1. Calibrate voltage dividers precisely and verify ADS1115 conversion factors.
  2. Implement proper signed conversion for ADS1115 and INA219 readings.
  3. Increase delay between ESP32 posts to Flask server to reduce `ECONNABORTED` errors.
  4. Continue developing Flask dashboard with real-time plotting of cell voltages, pack voltage, current, and SoC trends.
  5. Implement averaging of sensor readings to improve stability and accuracy.

## 8. Conclusion

- The battery monitoring system hardware is connected and functional.
- ESP32 can read ADC and current sensor data and attempt transmission to the Flask server.
- Measurement accuracy and network stability are not yet satisfactory.
- Work is ongoing to calibrate sensors, fix connection errors, and complete the Flask web dashboard for live visualization.

## 9. References

1. MicroPython documentation: https://docs.micropython.org
2. ADS1115 datasheet: https://www.ti.com/lit/ds/symlink/ads1115.pdf
3. INA219 datasheet: https://www.ti.com/lit/ds/symlink/ina219.pdf

## Lab-2 : Learning Reflection / Comments

This week, the battery monitoring project has progressed significantly. The ESP32 successfully connects to Wi-Fi and reads raw voltage and current data from ADS1115 and INA219 sensors. Data transmission to the Flask server is working, though some connection errors and voltage fluctuations remain. The web dashboard framework has been set up, ready for real-time visualization once sensor readings are stabilized. Overall, the experiment was challenging due to sensor calibration and network communication issues, but it was highly engaging and informative. The complete work has advanced up to this stage; Lab 2 and Lab 3 are not included in this week's progress.

**Note:** Lab 3 is not included in this week's progress; this report focuses on the advancements made by the team up to the current stage of the project.