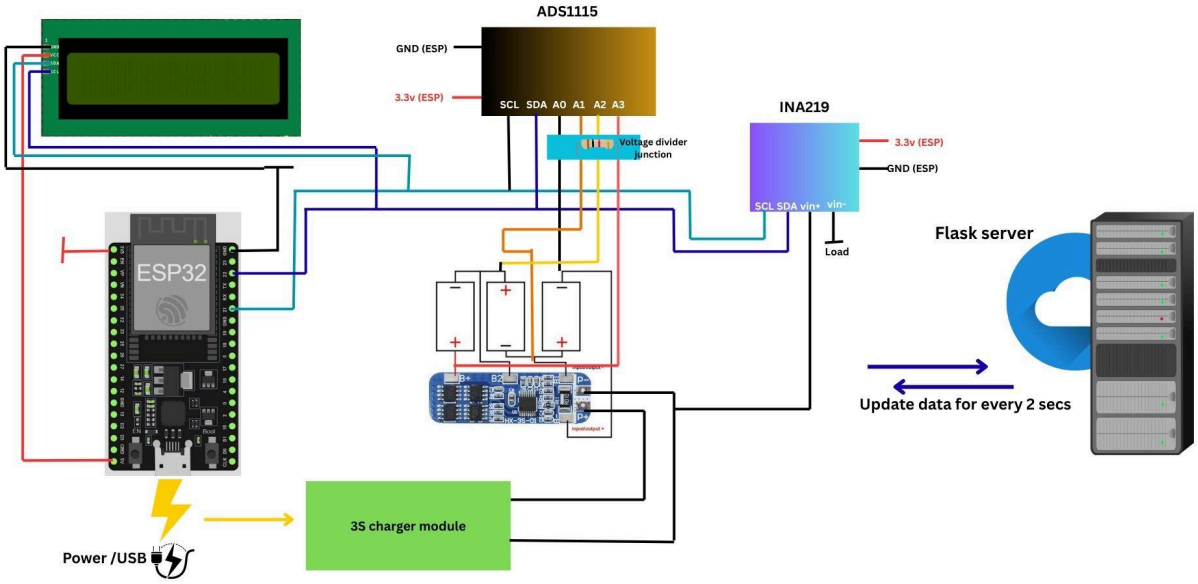
2025 TEEP Progress Report Week-20

This project focuses on optimizing the single-cell battery charging monitoring system completed last week, with the goal of improving 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 completeness and traceability of the research process.



Connections :

Battery Setup:

1. Three Li-Ion cells connected in series to form a 3S pack (≈12.6 V fully charged).
2. BMS module connected to balance and protect the cells during charging and discharging.

Sensor Connections:

1. INA219 placed between the pack's output (BMS P− terminal) and the system ground to measure charging/discharging current.

2. ADS1115 connected via I²C to ESP32 to measure voltages through voltage dividers:
    - A0 → Cell 1 (3.7 V max)
    - A1 → Cell 1 + 2 (≈7.4 V max)
    - A2 → Total pack (≈12.6 V max)

ESP32 and Peripherals:

- ESP32 connected to I²C devices:
    - ADS1115 (SDA = GPIO21, SCL = GPIO22)
    - INA219 (same I²C bus)
    - 16x2 LCD (I²C address = 0x27)
- Wi-Fi credentials configured to enable data transfer to the Flask server.

Programming:

- ESP32 programmed using Thonny (MicroPython).
- Code reads:
    - Individual cell voltages from ADS1115
    - Current and pack voltage from INA219
- Microcontroller calculates:
    - Each cell's voltage
    - Total pack voltage
    - Charging/discharging current
    - Battery percentage (SoC) and status
- EMA smoothing applied to reduce fluctuations in voltage and SoC readings.

Data Display and Communication:

- 16x2 LCD displays pack voltage and battery percentage in real time.
- ESP32 uploads the smoothed data to a Flask web server via Wi-Fi for remote monitoring.
- Flask dashboard displays live data and voltage trends graphically, providing stable and readable charts thanks to the smoothing algorithm.

Source code :

main.py

```
from machine import Pin, I2C
import network, urequests, time

# I2C Configuration
i2c = I2C(0, scl=Pin(22), sda=Pin(21), freq=100000)
ADS_ADDR, INA_ADDR, LCD_ADDR = 0x48, 0x40, 0x27
SSID, PASSWORD = "ISILAB CR", "isilab.ncut.CR"
URL = "http://192.168.50.205:5000/update"
```

```
# Wi-Fi Connection
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(SSID, PASSWORD)
while not wifi.isconnected():
    time.sleep(0.5)
print("Connected:", wifi.ifconfig())

# LCD Driver
class I2CLcd:
    def __init__(self, i2c, addr, rows=2, cols=16):
        self.i2c, self.addr = i2c, addr
        self.backlight = 0x08
        self._init_lcd()
    def _cmd(self, c):
        self.i2c.writeto(self.addr, bytes([c & 0xF0 | self.backlight]))
        self.i2c.writeto(self.addr, bytes([(c << 4) & 0xF0 | self.backlight]))
        time.sleep_ms(2)
    def _char(self, c):
        self.i2c.writeto(self.addr, bytes([c & 0xF0 | 0x01 | self.backlight]))
        self.i2c.writeto(self.addr, bytes([(c << 4) & 0xF0 | 0x01 | self.backlight]))
        time.sleep_ms(1)
    def _init_lcd(self):
        for c in [0x33,0x32,0x06,0x0C,0x28,0x01]: self._cmd(c)
    def clear(self): self._cmd(0x01)
    def move_to(self, r,c): self._cmd(0x80 | (c + 0x40*r))
    def putstr(self, s): [self._char(ord(ch)) for ch in s]

lcd = I2CLcd(i2c, LCD_ADDR)

# ADS1115 Voltage Reading
def read_ads(ch):
    config = 0x8400 | (ch << 12) | 0x0083
    i2c.writeto_mem(ADS_ADDR, 0x01, config.to_bytes(2,'big'))
    time.sleep(0.01)
    raw = int.from_bytes(i2c.readfrom_mem(ADS_ADDR, 0x00, 2),'big')
    if raw > 32767: raw -= 65536
    return raw * 4.096 / 32768

def get_cells():
    v1 = read_ads(0)*2
    v12 = read_ads(1)*3
    v123 = read_ads(2)*4
    return v1, v12-v1, v123-(v12), v123
```

```
# INA219 Current Reading
def read_ina219():
    i2c.writeto_mem(INA_ADDR, 0x05, (4096).to_bytes(2,'big'))
    bus_v = (int.from_bytes(i2c.readfrom_mem(INA_ADDR,0x02,2),'big')>>3)*0.004
    sh = int.from_bytes(i2c.readfrom_mem(INA_ADDR,0x01,2),'big')
    if sh > 32767: sh -= 65536
    cur = (sh*0.01/1000)/0.1
    return bus_v, cur
alpha=0.2
def ema(prev,new): return new if prev is None else prev*(1-alpha)+new*alpha
ema_c1=ema_c2=ema_c3=ema_pack=ema_soc=ema_cur=None

def soc_calc(v): return max(0,min(100,int((v-9)/(12.6-9)*100)))

while True:
    try:
        c1,c2,c3,pack = get_cells()
        vbus,cur = read_ina219()
        soc = soc_calc(pack)
        ema_c1,ema_c2,ema_c3 = ema(ema_c1,c1),ema(ema_c2,c2),ema(ema_c3,c3)
        ema_pack,ema_soc,ema_cur =
ema(ema_pack,pack),ema(ema_soc,soc),ema(ema_cur,cur)
        data={"cell1":round(ema_c1,2),"cell2":round(ema_c2,2),"cell3":round(ema_c3,2),
              "pack":round(ema_pack,2),"current":round(ema_cur,2),"soc":int(ema_soc)}
        urequests.post(URL,json=data)
        lcd.clear()
        lcd.move_to(0,0); lcd.putstr("Pack:{:.2f}V".format(ema_pack))
        lcd.move_to(1,0); lcd.putstr("SoC:{}%".format(int(ema_soc)))
        time.sleep(2)
    except Exception as e:
        print("Error:", e)
        time.sleep(2)
```

1. Description

This week's work focused on studying and understanding the BQ25792RQMR advanced buck-boost battery charger IC from Texas Instruments. The objective was to gain a strong theoretical foundation of its working principles, features, and application relevance for multi-cell Li-ion battery management systems. The study primarily involved analyzing the datasheet, functional block diagrams, and control mechanisms to prepare for future hardware integration and firmware development.

2. Learning Objectives

- Understand the purpose and application of the BQ25792RQMR
- Study the buck-boost charging architecture
- Learn how the IC manages multi-cell Li-ion batteries
- Analyze power path management, charging modes, and protections
- Understand I²C-based configuration and monitoring

3. Technical Study and Observations

3.1 Overview of BQ25792RQMR

The BQ25792RQMR is a highly integrated, NVDC buck-boost battery charger controller designed for 2S to 5S Li-ion battery packs. It supports both charging and system power regulation, making it suitable for portable and embedded power applications.

3.2 Buck-Boost Charging Operation

- The IC can operate in buck mode, boost mode, or buck-boost mode depending on the adapter and battery voltage.
- This allows stable charging even when the input voltage is higher or lower than the battery pack voltage.
- This feature improves flexibility and efficiency in real-world power conditions.

3.3 Power Path Management

- The BQ25792RQMR supports NVDC (Narrow Voltage DC) architecture.
- It intelligently distributes power between the system load and battery, ensuring uninterrupted system operation.
- System voltage is regulated independently of battery voltage, improving system stability.

3.4 Battery Charging Features

- Programmable charge current and charge voltage
- Supports pre-charge, fast charge, and termination modes

- Compatible with Li-ion and Li-polymer batteries
- Supports cell balancing preparation through accurate voltage control

3.5 Communication and Control

- Uses I²C communication for configuration and monitoring.
- Enables real-time monitoring of:
    - **Battery voltage**
    - **Charge current**
    - **Input current**
    - **System status and fault conditions**
- **This makes it suitable for microcontroller-based battery management systems.**

3.6 Protection and Safety Features

- Input over-voltage and under-voltage protection
- Battery over-voltage protection
- Thermal regulation and thermal shutdown
- Short-circuit and over-current protection
  These built-in protections enhance system safety and reliability.