**Battery IOT Documentation**

* **Revision Number:**  
  Rev. 2.0
* **Date of Creation or Update:**  
  14/12/2024
* **Prepared By:**  
  Muhammad .Sulaiman Shahid  
  Sabahat Akhter
* **Reviewed By:**
* **Approved By**

**Table of Contents**

1. Introduction
2. Working Flow Diagram
3. Battery IOT Details.

* Power supply
* Data acquisition and processing
* Data Communication

1. Battery IOT circuit Diagram
2. Troubleshooting the Battery IoT Circuit

**1. Introduction**

This Battery IoT Circuit is designed to interface with the Battery Management System (BMS) using UART communication to retrieve critical battery data. The retrieved data is then processed and transmitted to the Bike IoT system via CAN communication, ensuring seamless integration and real-time monitoring

**2. Working Flow Diagram:**

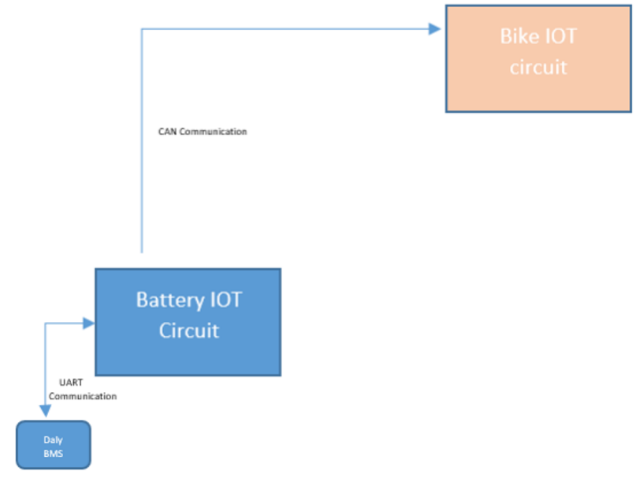
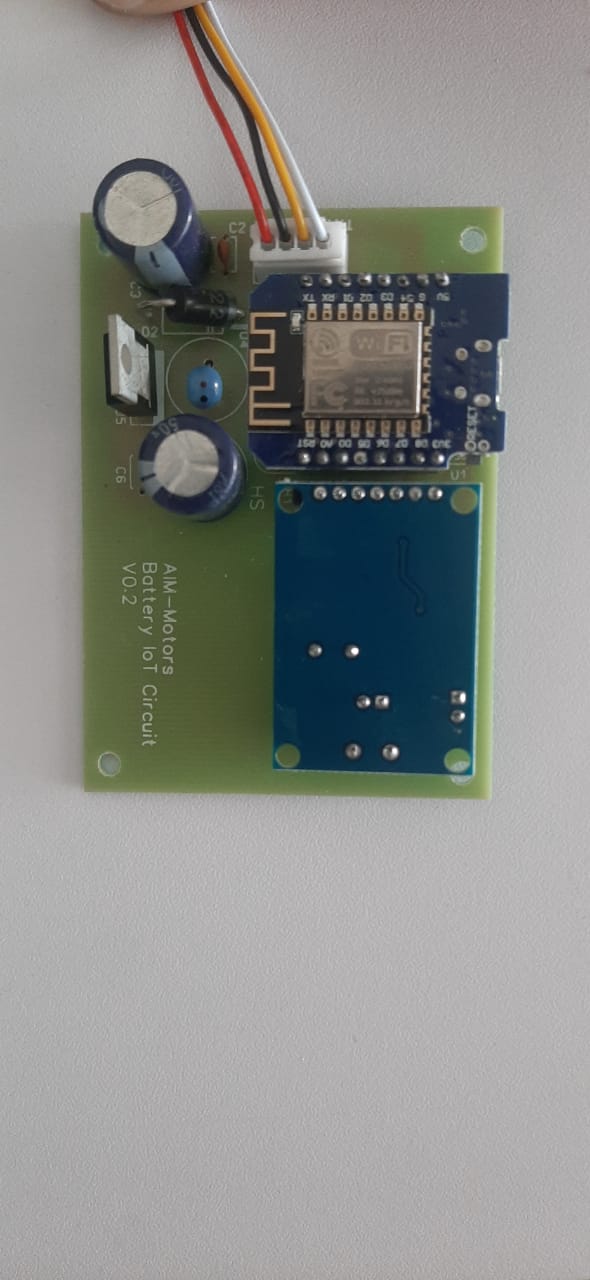
****

Figure 1: Flow diagram of Battery Iot

**3. Battery IOT Details**



**Battery IoT Circuit** that retrieves data from the Battery Management System (BMS) using UART communication, processes the data, and transmits it to the Bike IoT system via CAN communication. The circuit is divided into three main portions as follows:

* **Power supply.**
* **Data acquisition and processing.**
* **Data Communication**

Figure 2: Battery Iot PCB

**1-Power supply**   
The **LM2576T** is a popular step-down (buck) voltage regulator IC that provides an efficient and reliable way to design a regulated DC power supply. Below are the details for creating a power supply using this IC:

**Features of LM2576T**

* Output current capability: 3A.
* Adjustable or fixed output voltages: 3.3V, 5V, 12V, 15V, or adjustable (via external resistors).
* Input voltage range: Up to 40V.
* High efficiency (up to 88%).
* Internal thermal shutdown and current limit protection.

**Components Required**

1. **LM2576T Voltage Regulator IC**
2. Input capacitor: Electrolytic capacitor (100 µF to 470 µF, 50V).
3. Output capacitor: Electrolytic capacitor (330 µF, 25V).
4. Inductor: 100 µH (rated for 3A or higher).
5. Schottky diode: 1N5822 (for flyback protection).
6. Optional components: Resistors for adjustable voltage configurations.
7. PCB or breadboard for assembly.

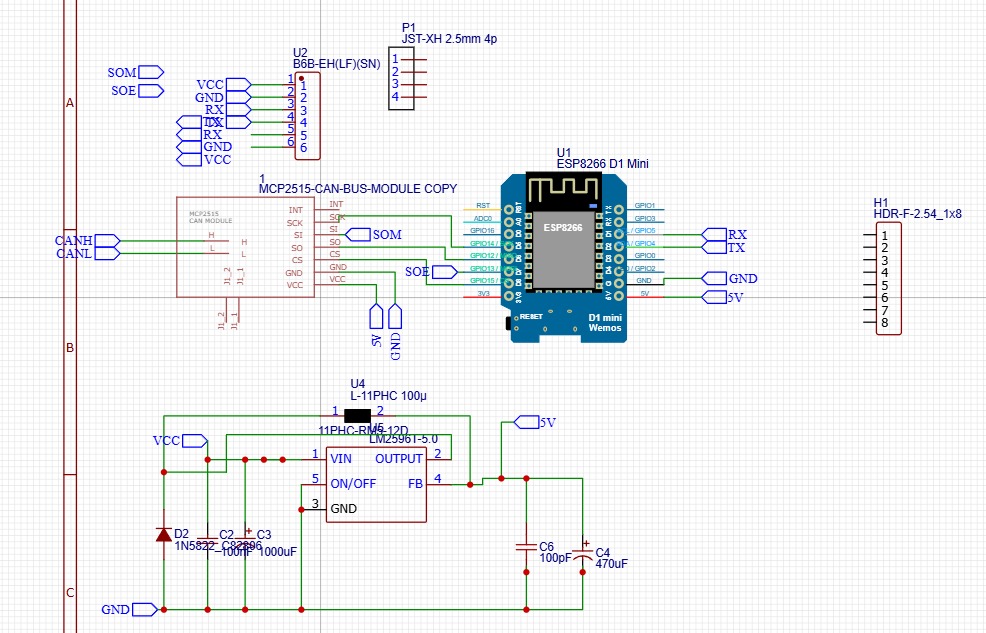
**Circuit Diagram**

Figure 3: Schematic diagram of Power supply

**Basic Circuit Design (Fixed Output Voltage):**

* **Input Voltage (Vin)**: Connect a DC input source between 7V and 40V.
* **Inductor**: Place a 100 µH inductor between the output pin of the LM2576T and the output capacitor.
* **Diode**: Connect the Schottky diode between the inductor and ground (cathode to the inductor, anode to ground).
* **Capacitors**: Place a 470 µF electrolytic capacitor on the input side and a 330 µF capacitor on the output side to filter noise and improve stability.

**Adjustable Voltage Configuration:**

* Connect a resistor divider network (R1 and R2) between the output pin and the feedback pin of the IC.

**Step-by-Step Assembly**

1. **Input Side**:
   * Connect the input voltage source (+) to the Vin pin of the LM2576T and ground (-) to the ground pin.
   * Place the input capacitor (470 µF) between the input and ground to stabilize the input voltage.
2. **Output Side**:
   * Connect the inductor to the Vout pin of the IC.
   * Place the Schottky diode between the inductor and ground (cathode facing the inductor).
   * Attach the output capacitor (330 µF) between the Vout pin and ground to smoothen the output voltage.
3. **Feedback (for Adjustable Output)**:
   * Connect a resistor divider network between Vout, feedback pin, and ground to set the desired output voltage.
4. **Testing**:
   * Power the circuit with the input voltage and measure the output using a multimeter.
   * Ensure the output voltage matches the expected value (fixed or adjustable configuration).

**Applications**

* Powering microcontrollers or sensors in IoT circuits.
* DC-DC step-down converters in battery-powered systems.
* General-purpose regulated power supplies for embedded systems.

**2-Data acquisition and processing**

This component involves the collection of data from various sources, such as sensors, instruments, or external systems, and its subsequent processing for analysis or decision-making.  
We use **ESP8266 module** for data acquisition

The ESP8266 is a low-cost, Wi-Fi-enabled microcontroller module widely used for IoT (Internet of Things) applications. Its key functionalities include:

**Wi-Fi Connectivity**: Supports 2.4 GHz Wi-Fi, enabling devices to connect to wireless networks and access the internet or local networks.

**Microcontroller Capabilities**: Features a built-in 32-bit microcontroller, capable of running user-defined programs.

**GPIO Pins**: Provides several General Purpose Input/output (GPIO) pins for interfacing with external sensors, actuators, and other peripherals.

**Serial Communication**: Supports communication protocols like UART, SPI, and I2C for data exchange with other devices.

**Low Power Consumption**: Includes sleep modes, making it suitable for battery-powered applications.  
**Web Server and Client**: Can act as a web server, hosting webpages for controlling or monitoring connected devices. It can also operate as a client, sending data to cloud servers.

**Programming Support**: Programmable using various platforms, including Arduino IDE, Lua, MicroPython, and more.

**OTA Updates**: Supports Over-the-Air (OTA) firmware updates, allowing users to update software without physical access to the device.

**Flexible Networking Modes**: Operates in station (STA), access point (AP), or both modes simultaneously for versatile networking configurations.

The ESP8266 is widely used in applications like **smart home automation, remote monitoring**, **industrial control systems**, and **wearable devices** due to its versatility and affordability.

**Circuit Diagram**

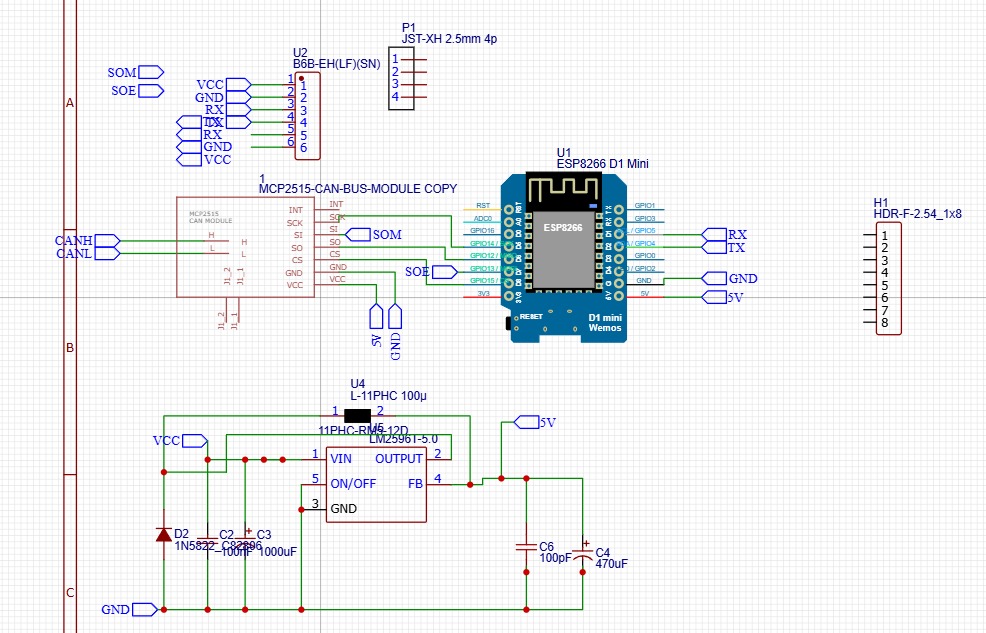


Figure 4: Schematic diagram of ESP 8266 weMOs D1 min   
**3- Data Communication**

Data communication refers to the transmission of data between various components, devices, or systems. In this circuit, we utilize the **MCP2515 CAN module** and the **UART port** to facilitate data communication.

The **MCP2515 CAN module** implements the Controller Area Network (CAN) protocol, allowing efficient and reliable communication between multiple devices in a network. It is well-suited for applications requiring real-time data exchange and robust error-handling capabilities. The MCP2515 is a high-speed CAN (Controller Area Network) transceiver that enables reliable and efficient communication between electronic control units (ECUs) in automotive and industrial applications. It offers a cost-effective and flexible solution for implementing CAN bus networks.

**Working Principle:** The MCP2515 operates on the CAN bus protocol, a robust and reliable communication protocol that allows multiple devices to communicate with each other without a central controller. It employs a carrier-sense multiple access with collision detection (CSMA/CD) mechanism to ensure reliable data transmission.

**Functionality:** When the system is powered on, the MCP2515 initiates communication with the BMS over the CAN bus. This high-speed communication protocol allows for efficient and reliable data exchange between the two devices.

The **UART (Universal Asynchronous Receiver/Transmitter) port**, on the other hand, enables point-to-point communication between two devices. This protocol is simple and effective for direct data transmission, making it ideal for low-speed communication or debugging purposes.

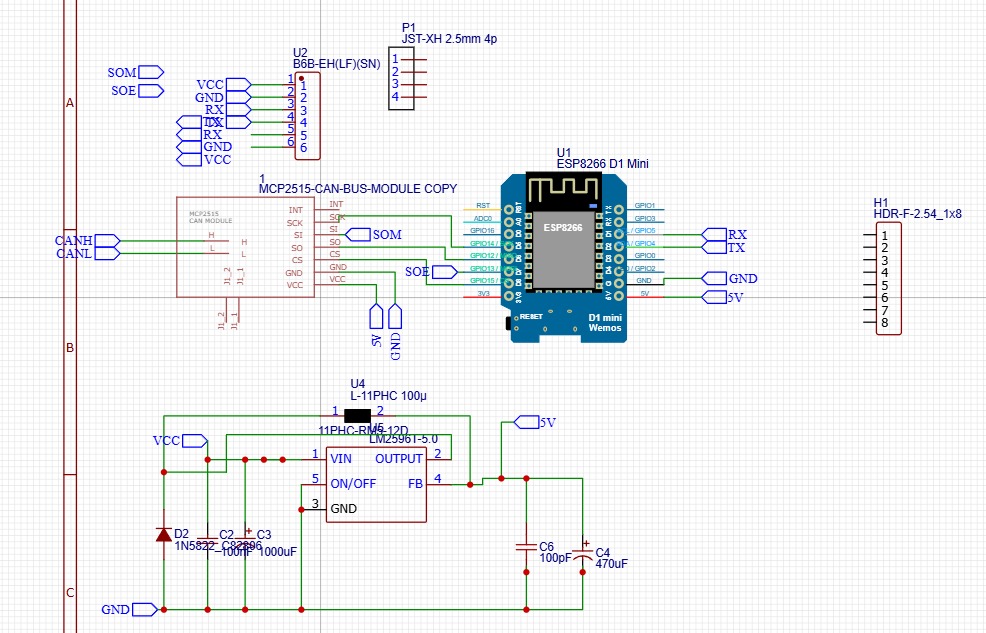
**Working Principle**

Its working principle involves converting parallel data from the transmitting device into serial data for transmission and then reconverting it back into parallel data at the receiving device. UART operates asynchronously, meaning it does not require a shared clock signal between devices; instead, both devices must agree on a common baud rate. Data is transmitted in a frame format, typically consisting of a start bit, data bits, an optional parity bit for error checking, and one or more stop bits to signal the end of transmission.

**Functionality:**  
 **Data Transmission**: Sends data serially bit by bit, making it efficient for communication between microcontrollers and peripheral devices like sensors, modules, and debugging tools.

 Error **Checking**: Offers an optional parity bit to detect errors during data transmission.

 Simple **Connectivity**: Requires only two wires for transmission (TX) and reception (RX), making it lightweight and easy to implement.

**4. Battery Iot Circuit Diagram:**Figure 4: Schematic diagram of ESP 8266 weMOs D1 min   
 Figure 5: Schematic diagram Battery Iot

**5. Troubleshooting the Battery IoT Circuit**

If issues arise with the Battery IoT Circuit, follow these steps to identify and resolve potential problems in each portion of the circuit:

### ****1. Power Supply Issues****

**Problem**: No power or unstable voltage output.  
**Solution**:

* **Check the Input Voltage**: Ensure the input voltage is within the acceptable range (7V–40V) for the LM2576T.
* **Verify Capacitors**: Inspect the input and output capacitors for damage or incorrect polarity. Replace if necessary.
* **Check the Inductor**: Ensure the inductor is correctly rated (100 µH, 3A or higher) and properly connected.
* **Inspect the Schottky Diode**: Verify the diode's polarity and functionality using a multimeter. Replace a faulty diode.
* **Thermal Shutdown**: If the LM2576T is overheating, check for short circuits or overload conditions on the output.

### ****5.2. Data Acquisition and Processing Issues (ESP8266)****

**Problem**: ESP8266 fails to acquire or process data.  
**Solution**:

* **Wi-Fi Connection**: Confirm that the ESP8266 is connected to a stable Wi-Fi network. Check the credentials and signal strength.
* **Check UART Connections**: Ensure the RX and TX pins are correctly wired and match the BMS's UART pins. Verify the baud rate settings.
* **Power Supply**: Ensure the ESP8266 receives a stable 3.3V power supply.
* **Firmware Issues**: Reflash the firmware or code onto the ESP8266 using the Arduino IDE or another compatible platform.
* **Restart Module**: Perform a hard reset of the ESP8266 to clear temporary issues.

### ****5.3. Data Communication Issues (CAN Communication)****

**Problem**: Data is not transmitted or received correctly over the CAN bus.  
**Solution**:

* **MCP2515 Configuration**: Verify the MCP2515 is initialized with the correct CAN bus settings (baud rate, filters, etc.).
* **Check CAN Wires**: Ensure the CAN-H and CAN-L wires are properly connected to the bike IoT system and are not reversed.
* **Termination Resistor**: Verify the presence of a 120-ohm termination resistor at both ends of the CAN bus.
* **Power Supply**: Confirm that the MCP2515 and CAN transceiver receive a stable 5V power supply.
* **Inspect Connections**: Check the SPI connection between the MCP2515 and the microcontroller for loose or incorrect wiring.
* **Debug Communication**: Use a CAN analyzer or serial monitor to check if messages are being sent and received on the bus.

### ****5.4. General Troubleshooting Steps****

* **Check Ground Connections**: Ensure all components share a common ground to avoid communication or power issues.
* **Inspect Solder Joints**: Look for cold or broken solder joints on the PCB. Resolder if necessary.
* **Test Components**: Use a multimeter to verify the functionality of individual components like resistors, capacitors, diodes, and ICs.
* **Verify Circuit Diagram**: Double-check all connections against the circuit diagram to ensure no wiring errors.
* **Error Codes or Logs**: Check for error messages or logs in the firmware to pinpoint the issue.

**5.5. Common Issues and Fixes**

|  |  |  |
| --- | --- | --- |
| Issue | Possible Cause | Solution |
| No output voltage from power supply | Faulty LM2576T, incorrect connections | Replace LM2576T, verify connections |
| ESP8266 not responding | Incorrect baud rate or firmware issues | Check baud rate, reflash firmware |
| No CAN communication | Faulty CAN-H or CAN-L connections | Verify and correct CAN wiring |
| Overheating components | Excessive load or short circuits | Check for short circuits, reduce load |
| Data mismatch between system | UART or CAN baud rate mismatch | Match UART and CAN baud rates |