MicroPython IoT Weather Station Documentation

1. Project Overview

The MicroPython IoT Weather Station is a battery-powered, Wi-Fi-enabled environmental monitoring system built on an ESP32 microcontroller and programmed in **MicroPython**. It collects local environmental data (temperature, humidity, pressure, altitude) using a BME280 sensor, monitors battery voltage for a 2000mAh 18650 battery, fetches weather data from the Open-Meteo API, uploads data to ThingSpeak, and reads wind speed via Modbus RTU. Data is displayed on an SH1106 OLED display, with a button for navigation and sensor recalibration. The system uses deep sleep to conserve power, waking on button press.

Key Features

- **Environmental Monitoring**: BME280 sensor for temperature, humidity, pressure, and altitude.
- **Battery Monitoring**: Voltage divider to measure 18650 battery voltage and estimate percentage.
- **Display**: SH1106 OLED (128x64) to show local sensor data, Open-Meteo weather, Modbus wind speed, and battery status.
- **Connectivity**: Wi-Fi for Open-Meteo API (weather data) and ThingSpeak (data logging).
- **Modbus RTU**: Reads wind speed and sensor status via UART from a Modbus device.
- **Power Management**: Deep sleep mode with wake on button press to extend battery life.
- **User Interaction**: Button to switch display pages and recalibrate sensors.

2. Hardware Requirements

- **ESP32 Development Board** (e.g., ESP32-WROOM-32)
    - MicroPython-compatible with Wi-Fi and UART.

- **BME280 Sensor**
    - I2C sensor for temperature, humidity, pressure, and altitude (address: 0x76).
- **SH1106 OLED Display**
    - 128x64, SPI interface.
- **18650 Battery (2000mAh)**
    - Rechargeable 3.7V Li-ion battery.
- **Power Bank Module**
    - TP4056 or similar with protection circuit for charging the 18650 battery.
- **Voltage Divider**
    - Two 100kΩ resistors to scale battery voltage (2.5V–4.2V) for ADC input.
- **Push Button**
    - For navigation and recalibration (pull-up, active low).
- **RS485 to TTL Module**
    - For Modbus RTU communication via UART.
- **Modbus RTU Device**
    - Provides wind speed and sensor status data.
- **Jumper Wires and Breadboard**
    - For prototyping.
- **Wi-Fi Network**
    - 2.4 GHz for internet access.

Pin Configuration

- **I2C (BME280)**:
    - SCL: GPIO 19
    - SDA: GPIO 21
- **SPI (SH1106 OLED)**:
    - SCK: GPIO 18
    - MOSI: GPIO 23
    - DC: GPIO 12
    - RST: GPIO 13
    - CS: GPIO 5
- **Button**: GPIO 14 (pull-up, active low)
- **Modbus UART**:
    - TX: GPIO 16

    - RX: GPIO 17
- **Battery ADC**: GPIO 34

3. Software Requirements

- **Thonny IDE**
    - Download: https://thonny.org/
- **MicroPython Firmware**
    - Version 1.20 or later for ESP32: https://micropython.org/download/esp32/
- **MicroPython Libraries**:
    - **Included**:
        - `sh1106.py` (for SH1106 OLED)
        - `bme280.py` (for BME280 sensor)
    - **External**:
        - `umodbus` (for Modbus RTU): https://github.com/brainelectronics/micropython-modbus
    - **Standard**:
        - `machine`, `network`, `urequests`, `ujson`, `ntptime`
- **External Services**:
    - **Open-Meteo API**: https://open-meteo.com/ (no API key required)
    - **ThingSpeak Account**: https://thingspeak.com/ (requires API key)

4. System Architecture

1. **Sensor Data Collection**: BME280 measures environmental data via I2C.
2. **Battery Monitoring**: Voltage divider scales battery voltage, read via ADC.
3. **Modbus RTU**: Reads wind speed and status via UART.
4. **Online Weather**: Fetches data from Open-Meteo via Wi-Fi.
5. **Data Upload**: Sends local and Modbus data to ThingSpeak.
6. **Display**: SH1106 OLED shows sensor data, battery status, and online weather.
7. **User Interaction**: Button switches display pages or triggers recalibration.
8. **Power Management**: Deep sleep with wake on button press.

Block Flow Diagram

```
graph TD
    A[BME280 Sensor] --> B[ESP32 Microcontroller]
    C[18650 Battery] --> D[Power Bank Module]
    C --> E[Voltage Divider]
    E --> B
    D --> B
    F[Modbus RTU Device] --> G[RS485 to TTL Module]
    G --> B
    H[Button] --> B
    B --> I[Read Sensors, Battery, Modbus]
    I --> J[Fetch Open-Meteo Data]
    J --> K[Process Data]
    K --> L[Display on SH1106 OLED]
    K --> M[Upload to ThingSpeak]
    K --> N[Handle Button Input]
    I --> O[Enter Deep Sleep]
    H --> O[Wake from Sleep]
```

5. Assembly Instructions

Step 1: Connect the BME280 Sensor

- **I2C Connections**:
  - VCC → ESP32 3.3V
  - GND → ESP32 GND
  - SDA → GPIO 21
  - SCL → GPIO 19

Step 2: Connect the SH1106 OLED Display

- **SPI Connections**:
  - VCC → ESP32 3.3V
  - GND → ESP32 GND
  - SCK → GPIO 18
  - MOSI → GPIO 23

    - DC → GPIO 12
    - RST → GPIO 13
    - CS → GPIO 5

Step 3: Connect the Power Bank Module and 18650 Battery

- Battery to power bank module (e.g., TP4056):
    - Positive (+) → B+
    - Negative (-) → B-
- Module output to ESP32:
    - OUT+ → ESP32 3.3V or 5V (check module specs)
    - OUT- → ESP32 GND
- Ensure protection circuit is active.

Step 4: Build the Voltage Divider

- Use two 100kΩ resistors (R1, R2):
    - Battery positive → R1 → GPIO 34 → R2 → GND
    - Battery negative → GND
- Output voltage: `Vout = Vbat / 2` (e.g., 4.2V → 2.1V)

Step 5: Connect the Button

- Button to GPIO 14:
    - One pin → GPIO 14
    - Other pin → GND
    - Enable internal pull-up in code

Step 6: Connect the Modbus RTU Device

- RS485 to TTL module:
    - VCC → ESP32 3.3V or 5V
    - GND → ESP32 GND
    - TX → GPIO 16 (ESP32 RX)
    - RX → GPIO 17 (ESP32 TX)
- Connect module to Modbus device per its specs.

6. Software Setup

Step 1: Install MicroPython Firmware

1. Download firmware (v1.20+): https://micropython.org/download/esp32/
2. In Thonny:
   - `Tools > Options > Interpreter`
   - Select `MicroPython (ESP32)` and port
   - Install firmware using “Install or update firmware”
3. Or use `esptool.py`:

```
 esptool.py --chip esp32 --port /dev/ttyUSB0 eraseflash
esptool.py --chip esp32 --port /dev/ttyUSB0 writeflash -z 0
x1000 esp32-xxxx.bin
```

Step 2: Configure Thonny

- Select `MicroPython (ESP32)` interpreter and port.
- Test:

```
 import machine
print(machine.freq())
```

Step 3: Install Libraries

1. Download:
   - `sh1106.py`, `bme280.py` (from repository)
   - `umodbus` (from repository)
2. Upload to ESP32 using Thonny’s File Explorer (`View > Files`).

Step 4: Upload and Configure Code

1. Clone/download: https://github.com/ShariarIman/Weather-Station-Mini-ESP32
2. Open `main.py` in Thonny and update:
   - Wi-Fi: `SSID`, `PASSWORD`
   - ThingSpeak: `API_KEY`

    - Open-Meteo: Latitude, longitude
    - Modbus: Device address, register
    - Voltage Divider: R1, R2, VMAX, VMIN
3. Save as `main.py` on ESP32.

7. Configuration

Wi-Fi

```
SSID = "MyNetwork"
PASSWORD = "MyPassword123"
```

ThingSpeak

- Create a channel on https://thingspeak.com/ and get the API key.

```
THINGSPEAK_API_KEY = "your-api-key"
```

Open-Meteo

- Set your location:

```
LATITUDE = 51.5074  # e.g., London
LONGITUDE = -0.1278
```

Modbus

- Configure device address and register:

```
MODBUS_ADDRESS = 1
MODBUS_REGISTER = 0
```

Battery

```
R1 = 100000
R2 = 100000
```

```
VMAX = 4.2
VMIN = 2.5
```

8. Usage

1. Power via 18650 battery or USB.
2. ESP32 runs `main.py`, connects to Wi-Fi, and displays data.
3. Press button to cycle display pages (sensors, online weather, Modbus).
4. Long press (if implemented) recalibrates BME280.
5. Data uploads to ThingSpeak every minute.
6. Device sleeps for 5 minutes, wakes on button press.

9. Troubleshooting

- **Wi-Fi Failure**: Verify `SSID`, `PASSWORD`, 2.4 GHz network.
- **BME280 Issues**: Check I2C wiring, address (`i2c.scan()`).
- **OLED Blank**: Verify SPI pins, `sh1106.py` version.
- **Modbus Errors**: Check UART wiring, baud rate, device address.
- **Battery Reading Off**: Measure voltage divider output, adjust `VMAX`, `VMIN`.
- **No ThingSpeak Data**: Validate API key, internet connection.

10. Extending the Project

- Add sensors (e.g., UV).
- Log to filesystem:

```
with open("weather.txt", "a") as f:
    f.write(f"{time.time()},{temp},{hum},{press},{battery}\n")
```

- Enhance button for recalibration:

```
def recalibrate():
    bme = BME280(i2c=i2c, address=0x76)  # Reset sensor
```

11. License

Check repository’s `LICENSE` file (likely MIT).

12. References

- MicroPython Docs
- BME280 Library
- SH1106 Library
- uModbus
- Open-Meteo
- ThingSpeak

13. Contact

- Repository: https://github.com/ShariarIman/Weather-Station-Mini-ESP32
- Contact: ShariarIman via GitHub

Last Updated: May 19, 2025