2025 TEEP Progress Report Week- 5

Used Wokwi to complete the tasks for Labs 1 through 6

(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 – Short Answer Questions:

1. What is the approximate temperature measurement range of the DHT11, and what is its resolution?

Ans: The DHT11 can measure temperatures from **0°C to 50°C** with a resolution of **1°C**.

---

2. What is the approximate humidity measurement range of the DHT11, and what is its resolution?

Ans: The DHT11 can measure relative humidity from **20% to 90% RH** with a resolution of **1% RH**.

---

3. What type of data transmission method does the DHT11 use?

Ans: The DHT11 uses a **single-wire digital serial communication protocol**.

---

4. Between DHT11 and DHT22, which one has higher accuracy?

Ans: The **DHT22** has higher accuracy and a wider measurement range compared to the DHT11.

---

5. When using the DHT11, what is the typical time interval required between readings?

Ans: A minimum interval of about **1 second** is required between readings.

---

6. In what kinds of applications is the DHT11 commonly used?

Ans: The DHT11 is commonly used in **weather monitoring systems, indoor climate control, greenhouse monitoring, smart home systems, and educational projects**.

---

7. If abnormal data is read from the DHT11, what could be the possible causes?

Ans: Possible causes include:

- **Poor electrical connections** (loose wires, bad soldering)
- **Incorrect wiring** of VCC, GND, or data pin
- **Insufficient power supply**
- **Excessive sampling rate** (reading too fast)
- **Environmental factors** (condensation, extreme temperature outside sensor's range)

Overview :

The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin (no analog input pins needed). It's fairly simple to use but requires careful timing to grab data.

Lab2- On Wokwi , complete the following tasks:

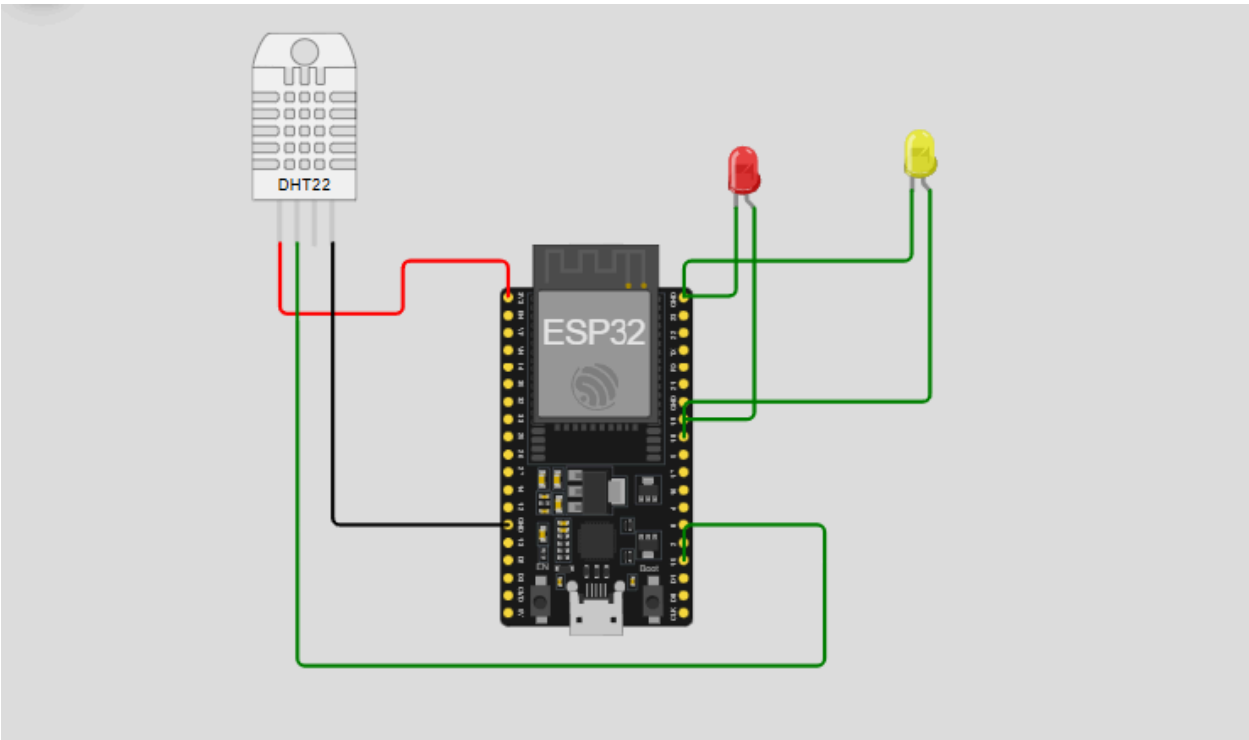
Design a circuit on the Wokwi that connects a DHT22, a yellow LED (LED_Hum), and a red LED (LED_Temp). Complete the following steps:

1. Connect the floating pin of the variable resistor to DHT22XX (GPIO XX) , the yellow LED (LED_Hum) to GPIOXX and ed LED (LED_Temp) to GPIOXX。

    - **Function 1**: Get the temperature and humidity data of DHT22 every 3 seconds and display the average value in Shell after getting 5 records.
    - **Function 2**:When the temperature is above 30 degrees, LED_Temp lights up.
    - **Function 3:**When the humidity is higher than 60%, LED_Hum lights up.

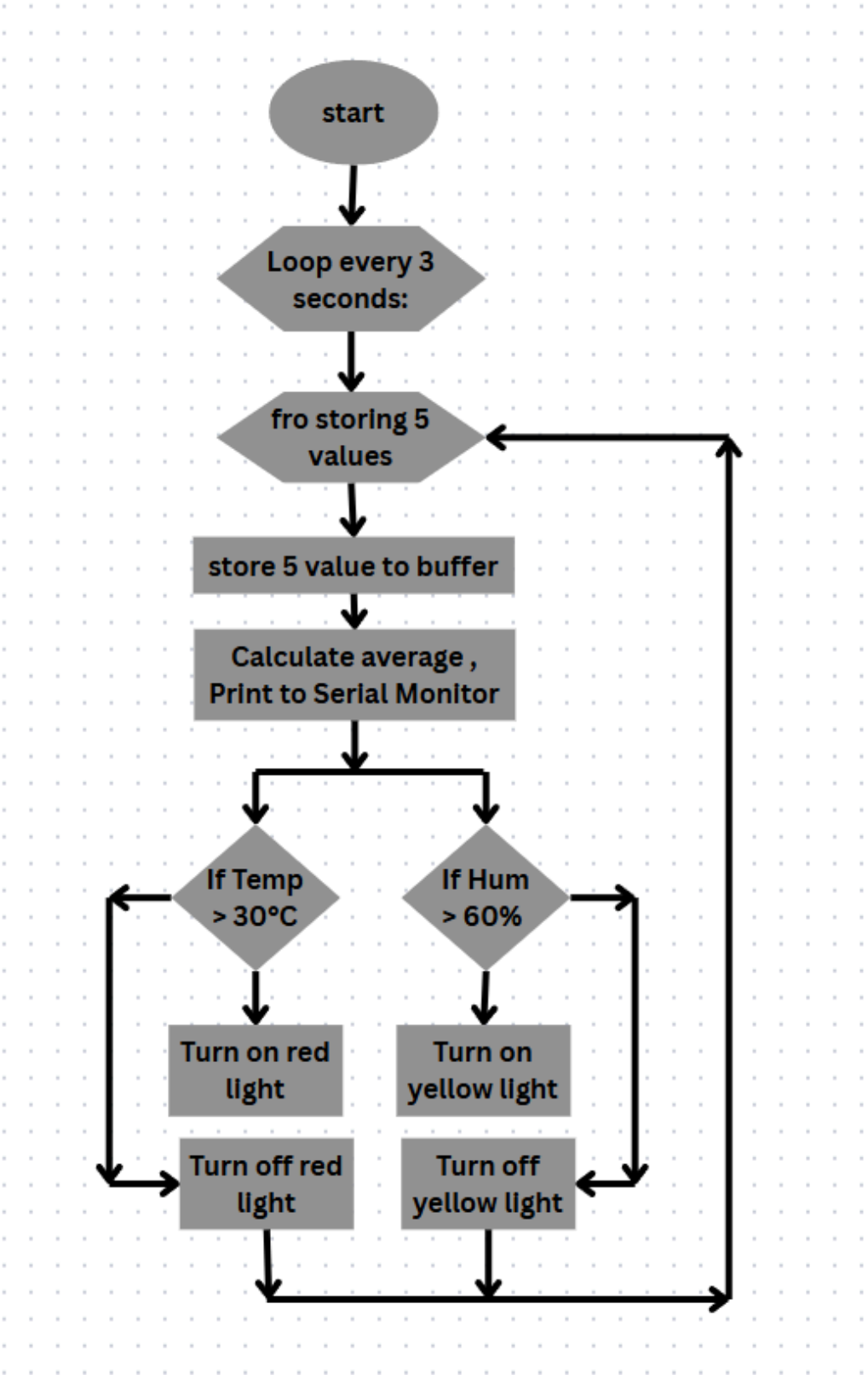
Connections:

- **DHT22** → Connect `VCC` to `3.3V`, `GND` to `GND`, and `DATA` to **GPIO 15**
- **Yellow LED (LED_Hum) A** → **GPIO 18** ,and **C** → GND.
- **Red LED (LED_Temp) A** → **GPIO 19** ,and **C** → GND.



Note : In wokwi i use the DHT22 instead of DHT11 , because of DHT11 isn't available

Flow Chart :


start
Loop every 3 seconds:
fro storing 5 values
store 5 value to buffer
Calculate average , Print to Serial Monitor
If Temp > 30°C
If Hum > 60%
Turn on red light
Turn on yellow light
Turn off red light
Turn off yellow light

Souce code :

```
#include "DHT.h"

#define DHTPIN 15       // DHT22 data pin connected to GPIO 15
#define DHTTYPE DHT22   // Define sensor type DHT22

#define LED_HUM 18      // Yellow LED pin
#define LED_TEMP 19     // Red LED pin

DHT dht(DHTPIN, DHTTYPE);

float tempBuffer[5];    // To store last 5 temp readings
float humBuffer[5];     // To store last 5 humidity readings
int index = 0;          // Current index in buffer

void setup() {
  Serial.begin(115200);
  dht.begin();
  pinMode(LED_HUM, OUTPUT);
  pinMode(LED_TEMP, OUTPUT);
  Serial.println("DHT22 Sensor Test Started...");
}

void loop() {
  delay(3000); // Wait 3 seconds between readings

  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  // Check if readings are valid
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT22 sensor!");
    return;
  }

  // Save to buffer
  tempBuffer[index] = temperature;
  humBuffer[index] = humidity;
  index++;

  // When 5 records are collected
  if (index == 5) {
    float avgTemp = 0, avgHum = 0;
    for (int i = 0; i < 5; i++) {
```

```
      avgTemp += tempBuffer[i];
      avgHum += humBuffer[i];
    }
    avgTemp /= 5;
    avgHum /= 5;

    Serial.print("Average Temp (last 5 readings): ");
    Serial.print(avgTemp);
    Serial.println(" °C");

    Serial.print("Average Humidity (last 5 readings): ");
    Serial.print(avgHum);
    Serial.println(" %");

    index = 0; // Reset buffer
  }

  // Function 2: Temp > 30°C → Red LED ON
  if (temperature > 30) {
    digitalWrite(LED_TEMP, HIGH);
  } else {
    digitalWrite(LED_TEMP, LOW);
  }

  // Function 3: Hum > 60% → Yellow LED ON
  if (humidity > 60) {
    digitalWrite(LED_HUM, HIGH);
  } else {
    digitalWrite(LED_HUM, LOW);
  }
}
```

Test Record :

```
ets Jul 29 2019 12:21:46

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:4728
load:0x40078000,len:14888
load:0x40080400,len:3368
entry 0x400805cc
Average Temp (last 5): 24.00 °C
Average Humidity (last 5): 40.00 %
Average Temp (last 5): 24.00 °C
Average Humidity (last 5): 40.00 %
Average Temp (last 5): 24.00 °C
Average Humidity (last 5): 40.00 %
Average Temp (last 5): 24.00 °C
Average Humidity (last 5): 40.00 %
```

Shell ×

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

  MPY: soft reboot
  Average Temp (last 5): 691.60 °C
  Average Humidity (last 5): 1192.96 %
  Average Temp (last 5): 691.50 °C
  Average Humidity (last 5): 1203.20 %
  Average Temp (last 5): 691.42 °C
  Average Humidity (last 5): 1203.20 %
  Average Temp (last 5): 691.38 °C
  Average Humidity (last 5): 1198.08 %
  Average Temp (last 5): 691.40 °C
  Average Humidity (last 5): 1198.08 %
  Average Temp (last 5): 691.42 °C
  Average Humidity (last 5): 1198.08 %
  Average Temp (last 5): 691.60 °C
  Average Humidity (last 5): 1198.08 %
  Average Temp (last 5): 691.66 °C
  Average Humidity (last 5): 1198.08 %
  Average Temp (last 5): 691.68 °C
  Average Humidity (last 5): 1213.44 %
  Average Temp (last 5): 691.62 °C
  Average Humidity (last 5): 1228.80 %
  ets Jul 29 2019 12:21:46
```

Lab3 – On ESP32, complete the following tasks:

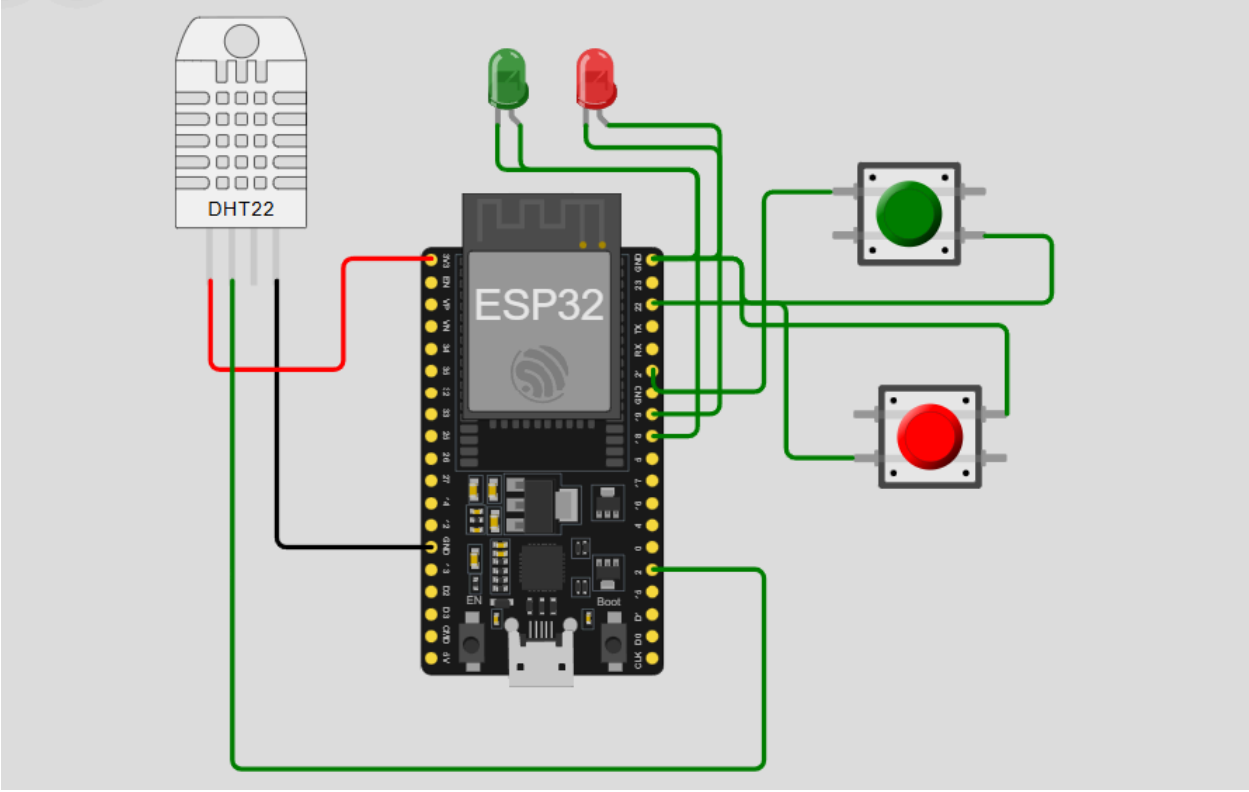
Design a circuit on the ESP32 that connects a DHT11, a yellow LED (LED_Hum), and a red LED (LED_Temp). Complete the following steps:Connect SW1 to GPIOXX, SW2 to GPIOXX, and the LED to GPIOXX.

1. Connect the floating pin of the variable resistor to DHT11XX (GPIO 2) , the yellow LED (LED_Hum) to GPIOXX and ed LED (LED_Temp) to GPIOXX。

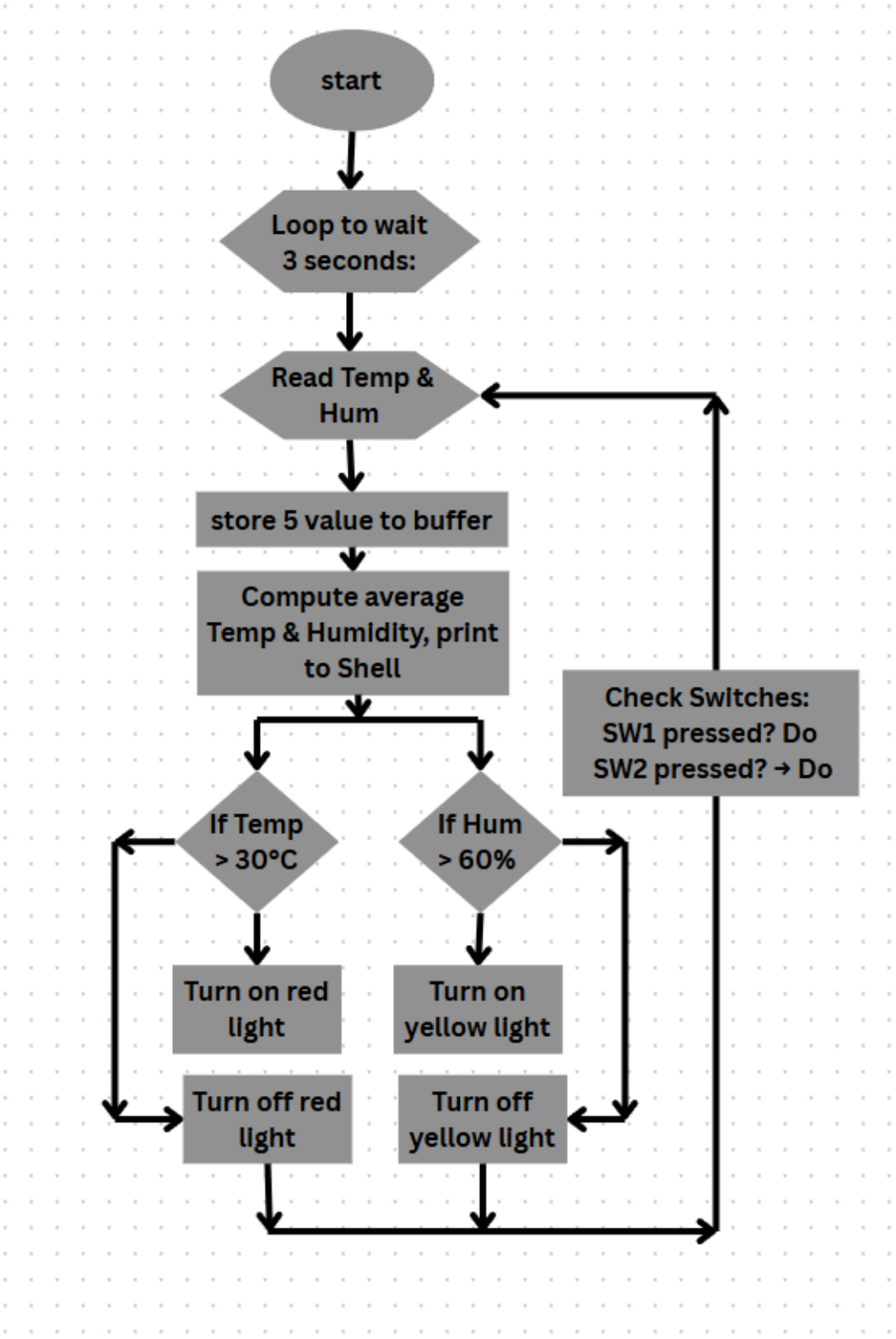
    - Function 1 : Get the temperature and humidity data of DHT22 every 3 seconds and display the average value in Shell after getting 5 records.
    - Function 2 : When the temperature is above 30 degrees, LED_Temp lights up.
    - Function 3 : When the humidity is higher than 60%, LED_Hum lights up.

Connections :

- DHT11 Data → GPIO 2
- LED_Hum (Yellow) → GPIO 18
- LED_Temp (Red) → GPIO 19
- SW1 → GPIO 21
- SW2 → GPIO 22



Flow chart :


start
Loop to wait 3 seconds:
Read Temp & Hum
store 5 value to buffer
Compute average Temp & Humidity, print to Shell
If Temp > 30°C
If Hum > 60%
Turn on red light
Turn on yellow light
Turn off red light
Turn off yellow light
Check Switches: SW1 pressed? Do SW2 pressed? → Do

Source Code :

```
import machine
import time
import dht

# --- Pin setup ---
dht_sensor = dht.DHT11(machine.Pin(2))   # DHT11 on GPIO2
led_hum = machine.Pin(18, machine.Pin.OUT)  # Yellow LED
led_temp = machine.Pin(19, machine.Pin.OUT) # Red LED
sw1 = machine.Pin(21, machine.Pin.IN, machine.Pin.PULL_UP) # Switch 1
sw2 = machine.Pin(22, machine.Pin.IN, machine.Pin.PULL_UP) # Switch 2

# Buffers
temp_buffer = []
hum_buffer = []
logging_enabled = True

while True:
    time.sleep(3)  # 3 second delay

    # Check switches
    if sw1.value() == 0:
        logging_enabled = True
        print("SW1 pressed → Logging ENABLED")
    if sw2.value() == 0:
        logging_enabled = False
        temp_buffer.clear()
        hum_buffer.clear()
        print("SW2 pressed → Logging DISABLED & buffers reset")

    if not logging_enabled:
        continue

    try:
        dht_sensor.measure()
        temperature = dht_sensor.temperature()
        humidity = dht_sensor.humidity()

        # Save readings
        temp_buffer.append(temperature)
        hum_buffer.append(humidity)

        # Keep only last 5 samples
        if len(temp_buffer) > 5:
            temp_buffer.pop(0)
            hum_buffer.pop(0)

        # Print averages when 5 samples collected
        if len(temp_buffer) == 5:
            avg_temp = sum(temp_buffer) / 5
```

```
            avg_hum = sum(hum_buffer) / 5
            print(" Avg Temp (last 5): {:.2f} °C".format(avg_temp))
            print(" Avg Hum (last 5): {:.2f} %".format(avg_hum))

        # LED controls
        led_temp.value(1 if temperature > 30 else 0)
        led_hum.value(1 if humidity > 60 else 0)

    except Exception as e:
        print("Sensor read failed:", e)
```

Test Records :

```
ets Jul 29 2019 12:21:46

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:4728
load:0x40078000,len:14888
load:0x40080400,len:3368
entry 0x400805cc
 Avg Temp (last 5): 0.00 °C
 Avg Hum (last 5): 1.00 %
 Avg Temp (last 5): 0.00 °C
 Avg Hum (last 5): 1.00 %
 Avg Temp (last 5): 0.00 °C
 Avg Hum (last 5): 1.00 %
SW2 pressed → Logging DISABLED & buffers reset
```

Shell

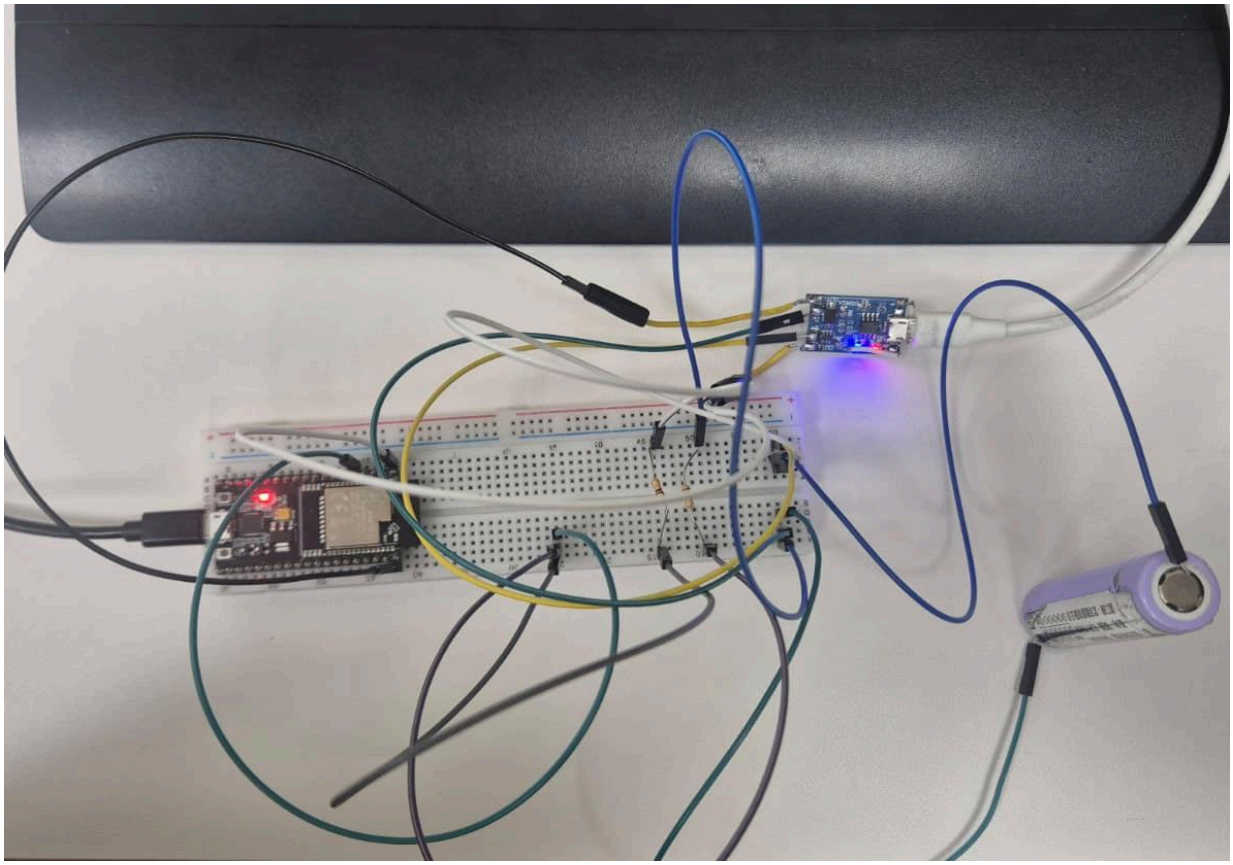
```
>>> %Run -c $EDITOR_CONTENT

 MPY: soft reboot
 SW1 pressed → Logging ENABLED
  Avg Temp (last 5): 27.00 °C
  Avg Hum (last 5): 46.20 %
  Avg Temp (last 5): 27.00 °C
  Avg Hum (last 5): 46.00 %
  Avg Temp (last 5): 27.00 °C
  Avg Hum (last 5): 46.00 %
  Avg Temp (last 5): 27.00 °C
  Avg Hum (last 5): 46.00 %
  Avg Temp (last 5): 27.00 °C
  Avg Hum (last 5): 46.00 %
  Avg Temp (last 5): 27.00 °C
  Avg Hum (last 5): 46.00 %
  Avg Temp (last 5): 27.00 °C
  Avg Hum (last 5): 46.00 %
 ets Jul 29 2019 12:21:46

 rst:0x1 (POWERON_RESET),boot:0x1b (SPI_FAST_FLASH_BOOT)
 configsip: 0, SPIWP:0xee
 clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
 mode:DIO, clock div:2
 load:0x3fff0030,len:4112
 load:0x40078000,len:15072
 load:0x40080400,len:4
 load:0x40080404,len:3332
 entry 0x400805ac
```

Lab4 – On ESP32, complete the following tasks:

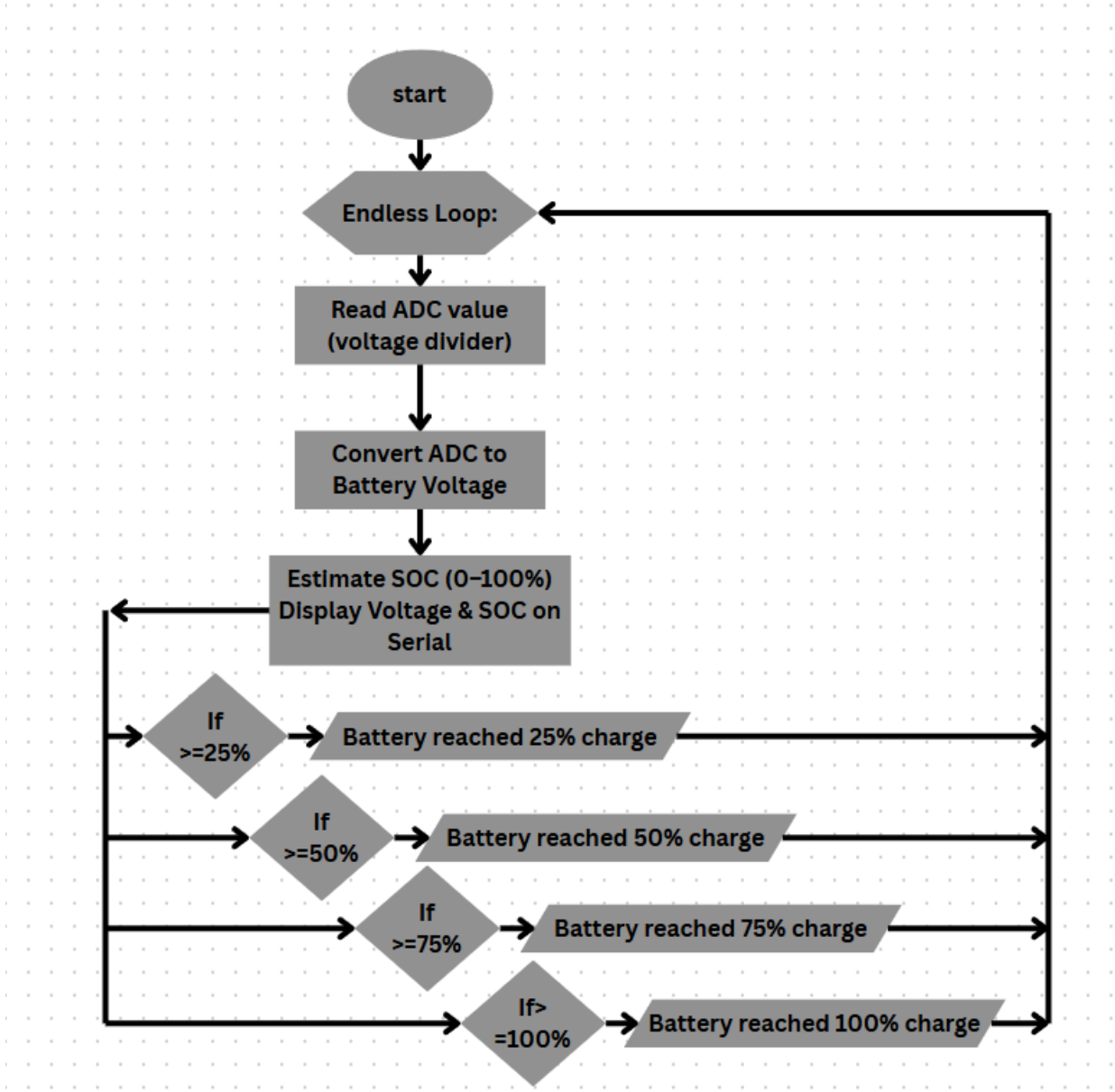
Using a TP4056 lithium battery charging module and an ESP32, build a charging measurement system for a single 3.7V lithium battery (such as an 18650 cell). Utilize the ESP32's built-in ADC to read the battery terminal voltage and estimate the battery's State of Charge (SOC). During the charging process, when the estimated SOC reaches 25%, 50%, 75%, and 100%, the system must output notification messages in the Shell (Serial Console).

Connections:

- Battery + → TP4056 B+
- Battery – → TP4056 B-
- TP4056 OUT+ → ESP32 Vin (for powering ESP32)
- TP4056 OUT– → ESP32 GND
- Battery + also connected to a **voltage divider** → **GPIO34 (ADC)**



Flow Chart :

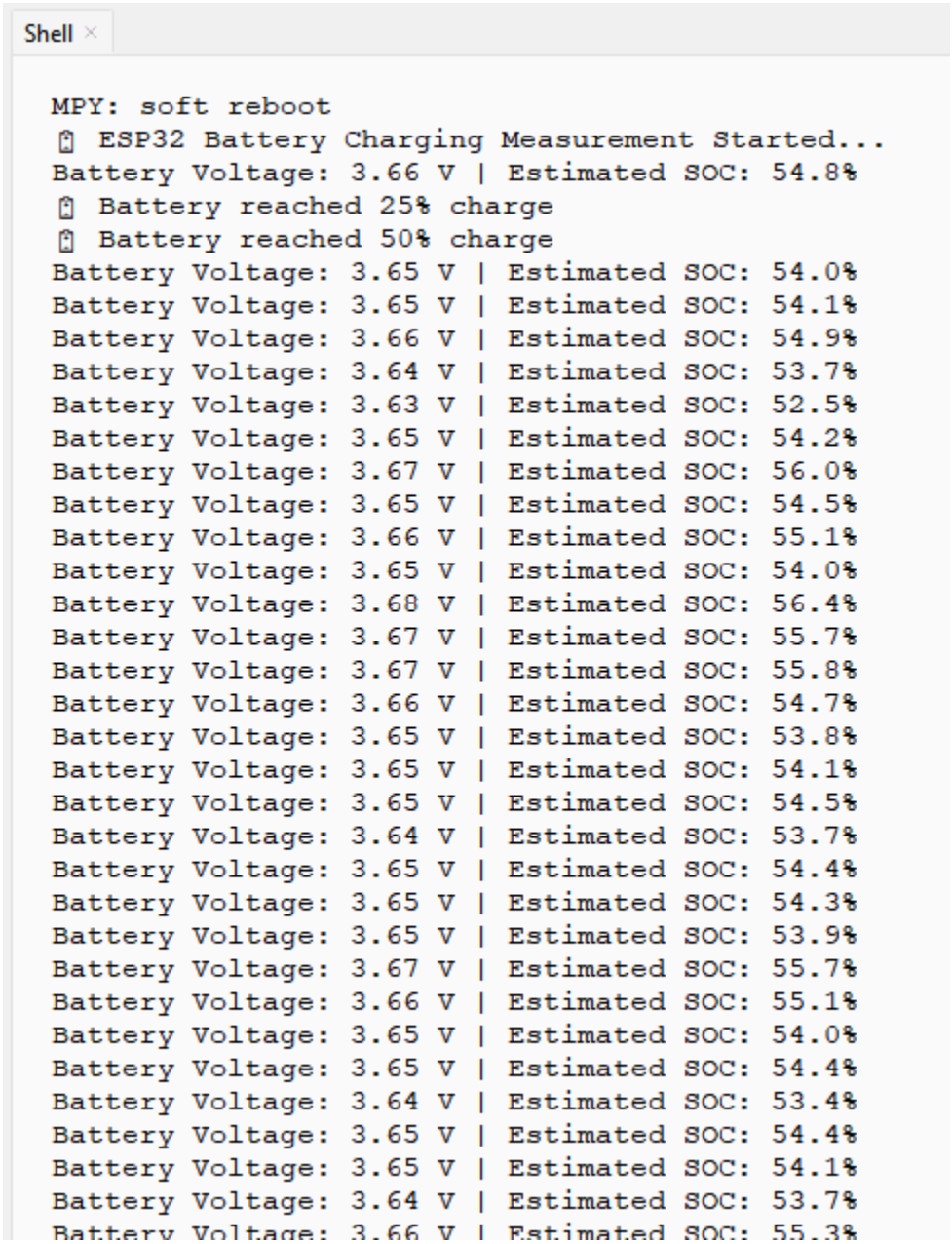

start
Endless Loop:
Read ADC value (voltage divider)
Convert ADC to Battery Voltage
Estimate SOC (0–100%) Display Voltage & SOC on Serial
If >=25%
Battery reached 25% charge
If >=50%
Battery reached 50% charge
If >=75%
Battery reached 75% charge
If> =100%
Battery reached 100% charge

Source code :

```
from machine import ADC, Pin
import time

# --- CONFIGURATION ---
ADC_PIN = 34      # ADC pin connected to voltage divider
R1 = 100000.0     # Resistor R1 value (100k)
R2 = 100000.0     # Resistor R2 value (100k)
VREF = 3.3        # ESP32 ADC reference voltage
ADC_RES = 4095.0  # 12-bit ADC resolution

last_notified = 0  # To avoid repeating notifications


def read_battery_voltage():
    adc = ADC(Pin(ADC_PIN))
    adc.atten(ADC.ATTN_11DB)     # Allows ~3.3V input
    adc.width(ADC.WIDTH_12BIT)   # 12-bit resolution

    raw = adc.read()
    voltage_at_adc = (raw / ADC_RES) * VREF
    battery_voltage = voltage_at_adc * ((R1 + R2) / R2)  # Reverse divider calc
    return battery_voltage


def estimate_soc(voltage):
    # Simple linear SOC estimation between 3.0V (0%) and 4.2V (100%)
    soc = (voltage - 3.0) / (4.2 - 3.0) * 100
    if soc < 0:
        soc = 0
    if soc > 100:
        soc = 100
    return soc


print(" ESP32 Battery Charging Measurement Started...")

while True:
    vbat = read_battery_voltage()
    soc = estimate_soc(vbat)

    print("Battery Voltage: {:.2f} V | Estimated SOC: {:.1f}%".format(vbat, soc))

    # Notifications at milestones
    global last_notified
    if soc >= 25 and last_notified < 25:
        print(" Battery reached 25% charge")
        last_notified = 25
    if soc >= 50 and last_notified < 50:
        print(" Battery reached 50% charge")
```

```
        last_notified = 50
    if soc >= 75 and last_notified < 75:
        print(" Battery reached 75% charge")
        last_notified = 75
    if soc >= 100 and last_notified < 100:
        print(" Battery fully charged (100%)")
        last_notified = 100

    time.sleep(2)  # Read every 2s
```

Test Record :



Lab5 learning reflection or comments(100 words)

The content of Learning reflection or comments can include your thoughts on the course, such as whether it was too difficult or too easy. You may also share something memorable that happened during the week. Overall, it is similar to a weekly journal, focusing on your learning experiences and personal feelings.

learning reflection or comments :

This week's labs were both challenging and rewarding, pushing me to apply my knowledge of sensors, microcontrollers, and programming in practical scenarios. Working with the DHT11 sensor on the ESP32 was initially straightforward, but I quickly realized that accurately reading and interpreting temperature and humidity data required attention to timing, averaging, and understanding how the sensors behave in different conditions. It was satisfying to see the LEDs respond to thresholds I had programmed, as it made the link between code and physical hardware tangible and motivating.

However, the battery charging measurement task was particularly challenging. I failed many times while trying to read the battery voltage accurately. Initially, I did not use a proper voltage divider, which caused the ADC readings to be unstable or exceed the ESP32's safe input range. Even after wiring the divider correctly, I experienced random fluctuations and incorrect SOC calculations, which was frustrating. These repeated failures taught me the importance of careful circuit design, proper grounding, and signal stabilization, such as averaging multiple ADC readings to reduce noise. It also highlighted how simulations or real hardware behave differently and require critical thinking and patience.

Despite the difficulties, I found these experiences extremely valuable. Troubleshooting the battery task improved my problem-solving skills and perseverance, while successfully completing the DHT sensor experiments gave me confidence in my coding and hardware integration abilities. A memorable moment was finally seeing the LEDs respond correctly after debugging the circuit multiple times it was a satisfying payoff for the effort I had invested. Overall, this week deepened my understanding of embedded systems and reinforced the idea that persistence and careful observation are key in electronics and programming.

Lab5 Video Link

Record a video of approximately 3 minutes to explain your research process. The video should include the following:

- 1.A description of the tools or methods used during your research process.
- 2.A presentation of the results and outcomes from LAB1 to LAB7.

Link : https://youtu.be/eHwLMVL02ws

Thank You