**Week 9:**

Select at least 1 input sensor (not PIR) and 1 output device and make the RPi control the chosen output device in response to activity by the input device **(e.g., a temperature sensor as input and two or more LEDs indicating the current temperature in binary code) .**

Note: DHT11 temperature sensor as input and LEDs as output. The LEDs will indicate the temperature in binary.

**Program:**

import RPi.GPIO as GPIO

import Adafruit\_DHT

import time

**# Define sensor and GPIO pins**

DHT\_SENSOR = Adafruit\_DHT.DHT11

DHT\_PIN = 4 # GPIO pin for DHT11

LED\_PINS = [17, 27, 22] # GPIO pins for LEDs

**# Setup GPIO mode**

GPIO.setmode(GPIO.BCM)

for pin in LED\_PINS:

GPIO.setup(pin, GPIO.OUT)

GPIO.output(pin, GPIO.LOW)

def display\_temperature(temp):

"""Display temperature in binary using LEDs."""

binary\_rep = bin(temp)[2:].zfill(len(LED\_PINS)) # Convert to binary

for i in range(len(LED\_PINS)):

GPIO.output(LED\_PINS[i], int(binary\_rep[i]))

try:

while True:

humidity, temperature = Adafruit\_DHT.read(DHT\_SENSOR, DHT\_PIN)

if temperature is not None and humidity is not None:

print(f"Temperature: {temperature}°C, Humidity: {humidity}%")

display\_temperature(temperature)

else:

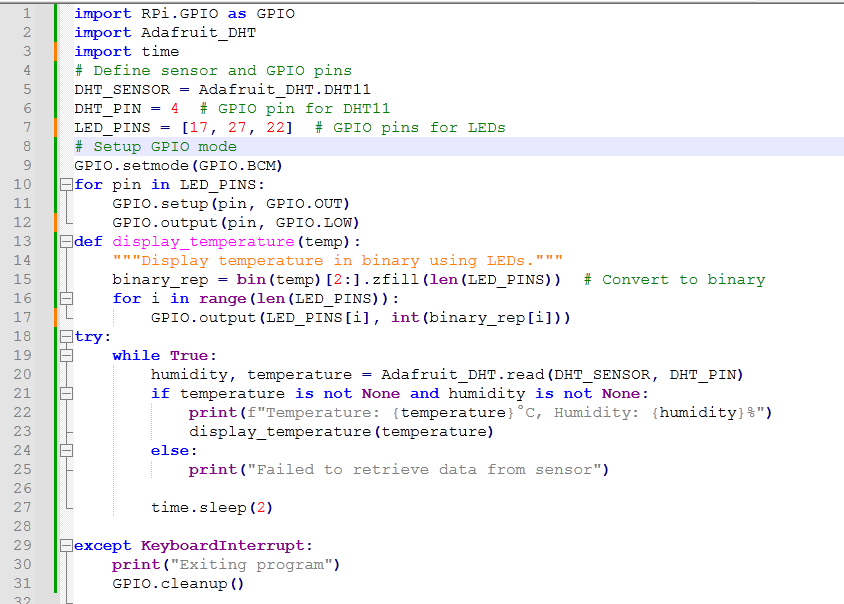
print("Failed to retrieve data from sensor")

time.sleep(2)

except KeyboardInterrupt:

print("Exiting program")

GPIO.cleanup()



**Explanation:**

 **DHT11 Sensor**: Reads temperature.

 **LEDs**: Represent temperature in binary.

 **GPIO Setup**: Configures LED pins.

 **Binary Conversion**: The temperature is converted to binary and displayed via LEDs.

 **Loop**: Reads temperature every 2 seconds.

 **Graceful Exit**: Cleans up GPIO on exit.

**LED Indication (Binary Representation of Temperature):**

* The LEDs will light up based on the binary form of the temperature.
* Example:
  + If **Temperature = 5°C**, its binary is **101**. The LEDs will turn on/off as follows:

**LED1 (MSB) - ON**

**LED2 - OFF**

**LED3 (LSB) - ON**

* + **If Temperature = 3°C (Binary 011):**

**LED1 - OFF**

**LED2 - ON**

**LED3 - ON**

**# binary\_rep = bin(temp)[2:].zfill(len(LED\_PINS))**

This line of code converts the temperature value into a **binary string** and ensures it has the correct length for the LEDs.

**binary\_rep = bin(temp)[2:].zfill(len(LED\_PINS))**

1. **bin(temp)**
   * **Converts the temperature (integer) into a binary string.**
   * **Example: bin(5) → '0b101'.**
2. **[2:]**
   * **Removes the '0b' prefix, leaving only the binary digits.**
   * **Example: '0b101' → '101'**.
3. **zfill(len(LED\_PINS))**
   * Ensures the binary string has exactly the same number of digits as the number of LEDs.
   * zfill(N) pads a string with **leading zeros** to make it exactly **N characters long**.
   * It adds leading zeros if necessary.
   * Example:
     + **If len(LED\_PINS) = 3 and**
     + **temp = 2 (Binary '10'),**
     + **zfill(3) makes it '010'.**

**Example Cases:**

| **Temperature (temp)** | **bin(temp)** | **[2:]** | **zfill(3)** | **Final binary\_rep** |
| --- | --- | --- | --- | --- |
| 5 | '0b101' | '101' | '101' | '101' |
| 2 | '0b10' | '10' | '010' | '010' |
| 7 | '0b111' | '111' | '111' | '111' |

Thus, this ensures that **all LEDs receive a proper binary value to represent the temperature**.

**# If temp = 27°C, let's break down how the binary representation works with the given code:**

**Step-by-step conversion**

1. **Convert 27 to binary:**

**bin(27) # Output: '0b11011'**

* + Removing the '0b' prefix: : **'11011'**

1. **Applying zfill(len(LED\_PINS))**
   * Your LED\_PINS = [17, 27, 22] has **3 LEDs**, **so len(LED\_PINS) = 3.**

The binary '11011' is **5 bits long**, but we only need the last **3 bits** to fit the LEDs.

**Taking the last 3 digits:** **'011'**

**LED Output:**

| **Bit Position (MSB → LSB)** | **LED State** |
| --- | --- |
| 1st LED (GPIO 17) | OFF |
| 2nd LED (GPIO 27) | ON |
| 3rd LED (GPIO 22) | ON |

So, for **27°C**, **LED2 and LED3 will be ON, and LED1 will be OFF**.

**#humidity, temperature = Adafruit\_DHT.read(DHT\_SENSOR, DHT\_PIN)**

This line of code is responsible for reading the humidity and temperature values from the DHT11 sensor.

 **Adafruit\_DHT.read(DHT\_SENSOR, DHT\_PIN)**

* Calls the read() function from the Adafruit\_DHT library to get sensor data.
* DHT\_SENSOR specifies the sensor type (e.g., Adafruit\_DHT.DHT11).
* DHT\_PIN is the **GPIO pin number** where the sensor is connected.
* The function returns **two values**:
  + humidity → The relative humidity percentage (%).
  + temperature → The temperature in **Celsius**.

 **Unpacking the Tuple:**

* The function returns a tuple: (humidity\_value, temperature\_value).
* Example return value:

(55.2, 27.0)

* This means humidity = 55.2 and temperature = 27.0.
* Humidity: 55.2%
* Temperature: 27.0°C

**LED\_PINS[i]**

* LED\_PINS is a list of GPIO pins assigned to LEDs.
* LED\_PINS[i] selects the **i-th LED pin** in the list.

 **binary\_rep[i]**

* binary\_rep is a **binary string** representing the temperature.
* Example: If temperature = 5°C, then:

binary\_rep = '101' # Binary for 5 in 3-bit format

* binary\_rep[i] selects the **i-th binary digit** (either '0' or '1').

 **int(binary\_rep[i])**

* binary\_rep[i] is a **character** ('0' or '1'), so int() converts it into an integer (0 or 1).
* This allows the GPIO output function to properly control the LED (ON/OFF).

**GPIO.output(LED\_PINS[i], int(binary\_rep[i]))**

* This sets the **i-th LED** to either **ON (1) or OFF (0)**, depending on the corresponding binary bit.

### ****Example Scenario****

#### ****Assume:****

* LED\_PINS = [17, 27, 22] (3 LEDs connected to GPIO 17, 27, and 22).
* Temperature = 5°C, which in binary (3 bits) is '101'.
* binary\_rep = '101'.

#### ****Loop Execution:****

| **i** | **LED\_PINS[i]** | **binary\_rep[i]** | **int(binary\_rep[i])** | **LED State** |
| --- | --- | --- | --- | --- |
| **0** | **17** | **'1'** | **1** | **ON** |
| **1** | **27** | **'0'** | **0** | **OFF** |
| **2** | **22** | **'1'** | **1** | **ON** |

### ****Final LED Status:****

* **LED 1 (GPIO 17) → ON**
* **LED 2 (GPIO 27) → OFF**
* **LED 3 (GPIO 22) → ON**

**#GPIO.setmode(GPIO.BCM)**

Sets the GPIO numbering mode to BCM (Broadcom SOC channel numbers).

This means we refer to pins by their GPIO numbers rather than physical pin positions.