Experiment -3 Develop simple application - testing temperature, light sensor - IOT Application - using open platform / Raspberry Pi.

Aim:

To design and test an application for measuring temperature and light intensity using sensors with Raspberry Pi, and to simulate the experiment online.

Apparatus Required

1. **Online Simulator**

Components:

Raspberry Pi
DHT11 or DHT22 (Temperature and Humidity Sensor)
LDR (Light Dependent Resistor)
Resistors (10kΩ for the LDR)
LED (optional, for visual feedback)
Jumper Wires
Breadboard
Power Supply for Raspberry Pi
Python Programming Environment

Background Theory

Temperature Sensor (DHT11 or DHT22): The DHT series of sensors can measure both temperature and humidity. The DHT11 is suitable for basic projects, while the DHT22 provides a wider range of temperature and humidity measurement. The sensor outputs a digital signal which is processed by Raspberry Pi to get the temperature reading.

Light Sensor (LDR or Photoresistor): An LDR is a resistor whose resistance changes with light intensity. It's typically used in light-sensitive applications like street lighting or brightness adjustment in electronic devices. The Raspberry Pi can read the varying voltage using its analog-to-digital converter (ADC) or by using a simple voltage divider circuit.

Algorithm

1. **Initialize the Sensors and Raspberry Pi GPIO**:
   Set up the temperature sensor (DHT11/DHT22) and light sensor (LDR) connections.
   Set the Raspberry Pi GPIO pins for reading the sensor signals.

2. **Setup the Hardware Connections**:
   Temperature Sensor (DHT11/DHT22):
   VCC → 5V (Raspberry Pi)
   GND → GND (Raspberry Pi)

Signal Pin → GPIO Pin (e.g., GPIO17)

Light Sensor (LDR):

One leg of LDR → GPIO Pin (e.g., GPIO18)

Other leg of LDR → GND (through a 10kΩ resistor to 3.3V to form a voltage divider)

Optional LED for output feedback:

Anode → GPIO Pin (e.g., GPIO23, through a 220Ω resistor)

Cathode → GND

3. **Read Sensor Data**:
   Read the temperature from the DHT sensor using a library.
   Read the light intensity from the LDR using the Raspberry Pi GPIO pins.

4. **Process Data**:
   Convert the raw sensor data into readable temperature (°C) and light intensity (lux) values.
   If the light level is above a threshold, turn on the LED (or perform another action like sending data to a cloud server).

5. **Display the Results**:
   Print the temperature and light values to the terminal or virtual serial monitor.
   Optionally, transmit the data to a cloud platform (using MQTT or HTTP) for further analysis or remote monitoring.

6. **Terminate the Program**:
   Stop any processes and clean up the GPIO settings.

Procedure

1. Hardware Setup in Raspberry Pi:

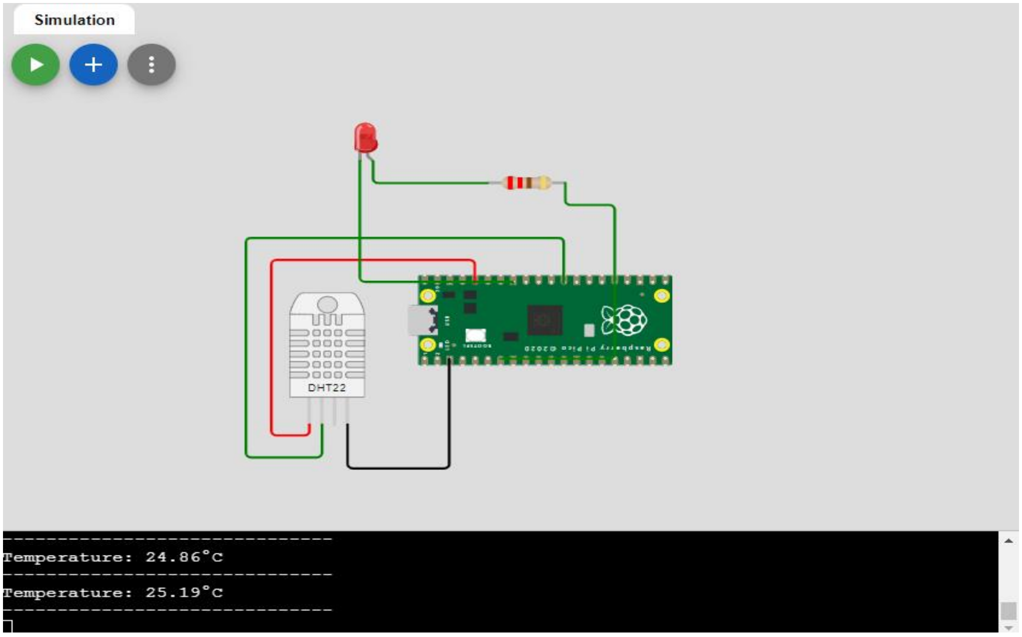
1. **Set up Raspberry Pi**:
   Connect Raspberry Pi to a monitor, keyboard, and mouse.
   Ensure Raspberry Pi OS is installed and booted.
2. **Connect the Sensors**:
   DHT11 or DHT22:
   Connect the VCC and GND of the DHT sensor to 5V and GND pins on the Raspberry Pi.
   Connect the signal pin of the DHT sensor to a GPIO pin (e.g., GPIO17).
   - **LDR**:
     Connect one leg of the LDR to a GPIO pin (e.g., GPIO18).
     Connect the other leg to GND through a 10kΩ resistor to create a voltage divider.
3. **LED**:
   Connect an LED to a GPIO pin (e.g., GPIO23) through a 220Ω resistor for visual feedback.

2. Software Setup:

1. Install necessary Python libraries for sensor reading.
2. Write the code to read data from both the DHT11/DHT22 and LDR sensors.

3. Simulation:

1. Run the Python script on your Raspberry Pi.
2. Monitor the console output for temperature, humidity, and light intensity readings.
3. Optionally, visualize the data on an IoT platform like ThingSpeak or Blynk.
4. Use real-time data to trigger actions (e.g., send alerts or automate devices based on environmental conditions).



Program

```
import dht
from machine import Pin
import time
import random  # To simulate slight changes

# Initialize DHT22 Sensor (Temperature Only)
sensor = dht.DHT22(Pin(22))

# Initialize LED with Resistor
led = Pin(5, Pin.OUT)  # LED connected to GPIO 5

# Temperature Threshold
temp_threshold = 25  # LED will blink if temperature > 25°C

# Previous temperature reading for comparison
prev_temp = None
```

```
while True:
    try:
        sensor.measure()
        temperature = sensor.temperature() + random.uniform(-3, 3)  # Simulate variation

        # Check for changes before printing
        if temperature != prev_temp:
            print(f"Temperature: {temperature:.2f}°C")
            print("-" * 30)

            prev_temp = temperature

        # Blink LED only if temperature exceeds the threshold
        if temperature > temp_threshold:
            led.on()
            time.sleep(0.5)
            led.off()
            time.sleep(0.5)
        else:
            led.off()  # Ensure LED is OFF if temperature is below threshold

    except OSError as e:
        print("Failed to read from DHT22 sensor:", e)

    time.sleep(2)  # Delay before next reading
```

Result

1. Successfully read temperature and humidity data using the DHT11 sensor connected to the Raspberry Pi.
2. The Serial Monitor displayed the temperature (in Celsius) and humidity (in percentage) values in real-time.
3. The LDR (Light Dependent Resistor) successfully detected the ambient light intensity, with the status displayed as "High" or "Low" based on the surrounding light conditions.
4. An LED was used for visual feedback, and it blinked when the light intensity was low, indicating that the LDR detected low light levels.

Pre-Lab Questions

1. What is the purpose of interfacing temperature, humidity, and light sensors with Raspberry Pi?
2. What communication protocols can be used to connect sensors with Raspberry Pi? Explain one.
3. What is the role of the Wokwi platform in simulating hardware experiments?
4. Describe the difference between analog and digital sensors. Which type are the sensors

used in this experiment?
5. How does the Raspberry Pi GPIO pin work, and why is it critical for interfacing sensors?

Post-Lab Questions

1. What challenges did you face while interfacing the sensors with Raspberry Pi, and how did you resolve them?
2. Explain how data from the temperature, humidity, and light sensors were captured and visualized in the experiment.
3. What modifications could be made to your program to include additional features, such as alert systems or data logging?
4. What are the advantages of using the Wokwi platform for prototyping sensor-based applications?
5. How can the data collected from these sensors be used in real-world applications, such as smart home automation or environmental monitoring?