Ex. No. 5 **CONNECT ARDUINO BOARD AND GLOW LED, READ ANALOG AND DIGITAL SENSORS SUCH AS RELAY, TEMPERATURE, HUMIDITY**

AIM:

To connect an Arduino board with a DHT sensor and read the temperature and humidity.

REQUIRED COMPONENTS:

- Arduino board (e.g., Arduino Uno)
- LED
- Resistor
- Relay module
- DHT11 or DHT22 Temperature and Humidity Sensor
- Jumper wires
- Breadboard

EXPERIMENTAL SETUP:

Connecting the LED:

- Connect the longer leg (anode) of the LED to digital pin 13 on the Arduino.
- Connect the shorter leg (cathode) of the LED to a current-limiting resistor.
- Connect the other end of the resistor to the ground (GND) pin on the Arduino.

Connecting the Relay Module:

- Connect the signal pin of the relay module to digital pin 7 on the Arduino.
- Connect the VCC and GND of the relay module to the 5V and GND pins on the Arduino.

Connecting the DHT Sensor:

- Connect the sensor's VCC to the 5V pin on the Arduino.
- Connect the sensor's GND to the GND pin on the Arduino.
- Connect the sensor's data pin to digital pin 2 on the Arduino.

Programming the Arduino:

- Use the provided Arduino code to control the LED and read temperature and humidity data from the sensor.
- Upload the code to the Arduino board using the Arduino IDE.

ARDUINO CODE:

```
#include <DHT.h>

#define DHT_TYPE DHT11
#define DHT_PIN 2  // Connect the DHT sensor data pin to digital pin 2

int ledPin = 13;  // LED connected to digital pin 13
int relayPin = 7;  // Relay connected to digital pin 7

DHT dht(DHT_PIN, DHT_TYPE);

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(relayPin, OUTPUT);

  Serial.begin(9600);
  dht.begin();
}

void loop() {
  // Read temperature and humidity from DHT sensor
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  // Control LED based on temperature
  if (temperature > 25.0) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }

  // Control relay based on humidity
  if (humidity > 60.0) {
    digitalWrite(relayPin, HIGH);  // Turn on the relay
  } else {
    digitalWrite(relayPin, LOW);  // Turn off the relay
  }

  // Print temperature and humidity to Serial Monitor
  Serial.print("Temperature: ");
  Serial.print(temperature);
```

```
  Serial.print(" °C, Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");

  delay(2000);  // Delay for 2 seconds
}
```

INSTRUCTIONS:

- Make sure you the necessary libraries are installed.
- Connect the Arduino board to the computer and upload the code using the Arduino IDE.
- Open the Serial Monitor in the Arduino IDE to see the temperature and humidity readings.
- Observe the LED and relay behavior based on the temperature and humidity conditions.

Note: Adjust the pin numbers in the code if you have connected the components to different pins on the Arduino.

PRE-LAB QUESTIONS:

1. How can a temperature and humidity sensor be used in an industrial climate control system?
2. What are the advantages of using a relay in home automation applications?
3. How does reading sensor data through an Arduino help in smart agriculture?
4. What modifications would be needed to connect these sensors to an IoT-based monitoring system?
5. How would you design an automated emergency system using relays and temperature sensors?

POST-LAB QUESTIONS:

1. How can you improve this system to send real-time sensor data to a cloud server for remote monitoring?
2. What challenges did you face while interfacing sensors, and how can they be resolved in large-scale applications?
3. How would you modify this setup to create an energy-efficient smart lighting system?
4. What steps would be required to integrate an alert system (like a buzzer or SMS notification) when abnormal sensor readings are detected?
5. If the relay failed to switch ON/OFF a device, what could be the possible reasons, and how would you debug the issue?

CONCLUSION:

The LED illuminated as expected when powered through the Arduino, confirming proper digital output functionality. The relay module responded accurately to digital signals from the Arduino, toggling connected devices on and off as programmed. The DHT11 sensor provided real-time temperature and humidity data, which were read and displayed correctly by the Arduino.