CS314 WSN Lab Experiment 1-4 Combined

Yatharth Shivhare
202251163

Experiment 1: Monitoring Temperature and Humidity using DHT11 Sensor with LED Indication on ESP8266

Objective:

Design a system to monitor environmental conditions using a DHT11 temperature and humidity sensor connected to an ESP8266 microcontroller. The system will display readings and indicate temperature ranges using LEDs.

Hardware Requirements:

- ESP8266 (NodeMCU)
- DHT11 Sensor
- LEDs (Red, Green, Yellow)
- Breadboard
- Jumper Wires
- USB Cable (for uploading code and monitoring via Serial Monitor)

Software Requirements:

- Arduino IDE
- DHT sensor library
- ESP8266 board package installed in Arduino IDE

Theoretical Background:

The DHT11 sensor is a popular choice for monitoring temperature and humidity in Wireless Sensor Network (WSN) projects. It communicates using a single-wire digital protocol, making it simple to interface with microcontrollers like the ESP8266.

In this project, the ESP8266 reads temperature and humidity data from the DHT11 sensor and sends the information to the Serial Monitor. LEDs serve as visual indicators for temperature ranges:

- **Green LED:** Ideal conditions (22°C–25°C)
- **Red LED:** Warning range (25°C–28°C)
- **Yellow LED:** Critical range (28°C–31°C)

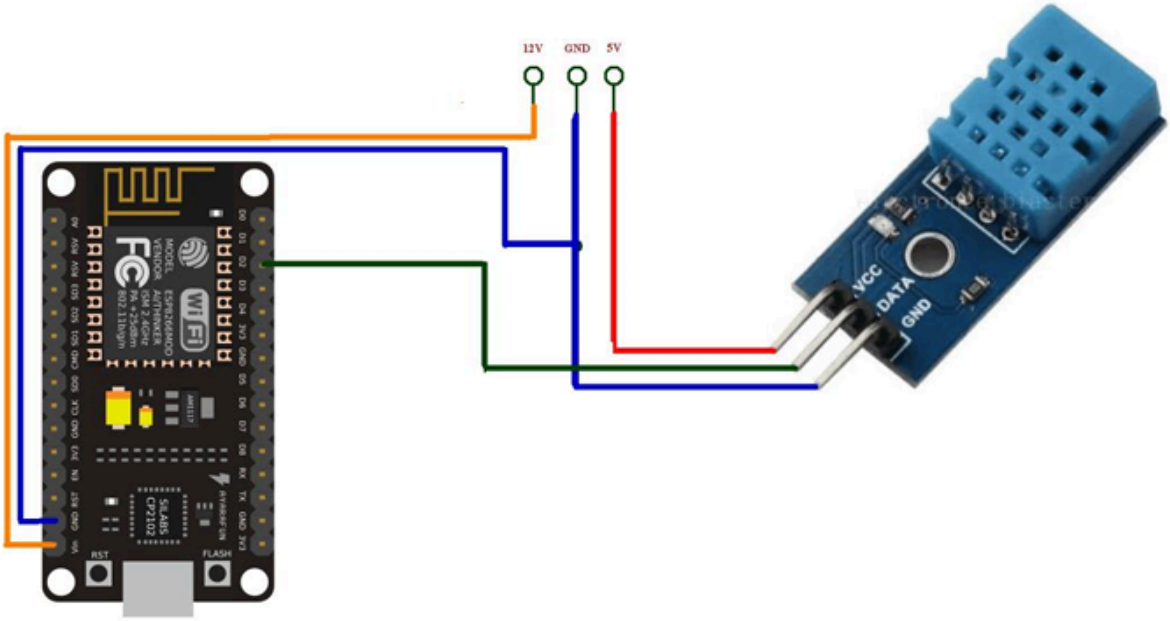
System Overview:

The **DHT11 sensor's** temperature readings determine which LED is activated:

- Green LED: Indicates optimal conditions.
- Red LED: Signals a temperature warning.
- Yellow LED: Highlights critical temperature levels.

This project combines hardware and software to create an efficient environmental monitoring system.

Circuit Diagram:



Procedure:

1. Connect the DHT11 Sensor to the ESP8266:
    - VCC → 3.3V
    - GND → GND
    - Data → GPIO4 (D2 on NodeMCU)
2. Connect the LEDs to the ESP8266:
    - Green LED → GPIO13 (D6)
    - Red LED → GPIO14 (D5)
    - Yellow LED → GPIO12 (D4)

3. Upload the Code: Use Arduino IDE to upload the provided sketch to the ESP8266.
4. Monitor Data: Open the Serial Monitor in Arduino IDE with a baud rate of 9600 to view live temperature and humidity readings.
5. Observe LED Behavior: LEDs will illuminate according to the temperature range detected.

Result:
The ESP8266 successfully retrieves and displays real-time temperature and humidity data from the DHT11 sensor. It activates the appropriate LED based on the specified temperature thresholds.

Conclusion:
This project demonstrates a simple yet effective implementation of a Wireless Sensor Network (WSN) node using an ESP8266 and a DHT11 sensor. The system is versatile and can be expanded for applications in environmental monitoring, IoT-enabled home automation, and smart agriculture solutions.

Experiment 2: Distance Measurement Using Ultrasonic Sensor with LED Alerts

Objective:

Interface an HC-SR04 ultrasonic sensor with a microcontroller to measure distances in real-time. Use LEDs to indicate whether an object is within a predefined safe range.

Hardware Requirements:

- ESP8266 (NodeMCU)
- HC-SR04 Ultrasonic Sensor
- LEDs (Red and Green)
- Breadboard
- Jumper Wires
- USB Cable (for uploading code and monitoring via Serial Monitor)

Software Requirements:

- Arduino IDE
- ESP8266 Board Package (installed in Arduino IDE)

Theoretical Background:

The **HC-SR04 ultrasonic sensor** calculates the distance to an object by emitting an ultrasonic pulse and measuring the time taken for the pulse to return after reflecting off the object.

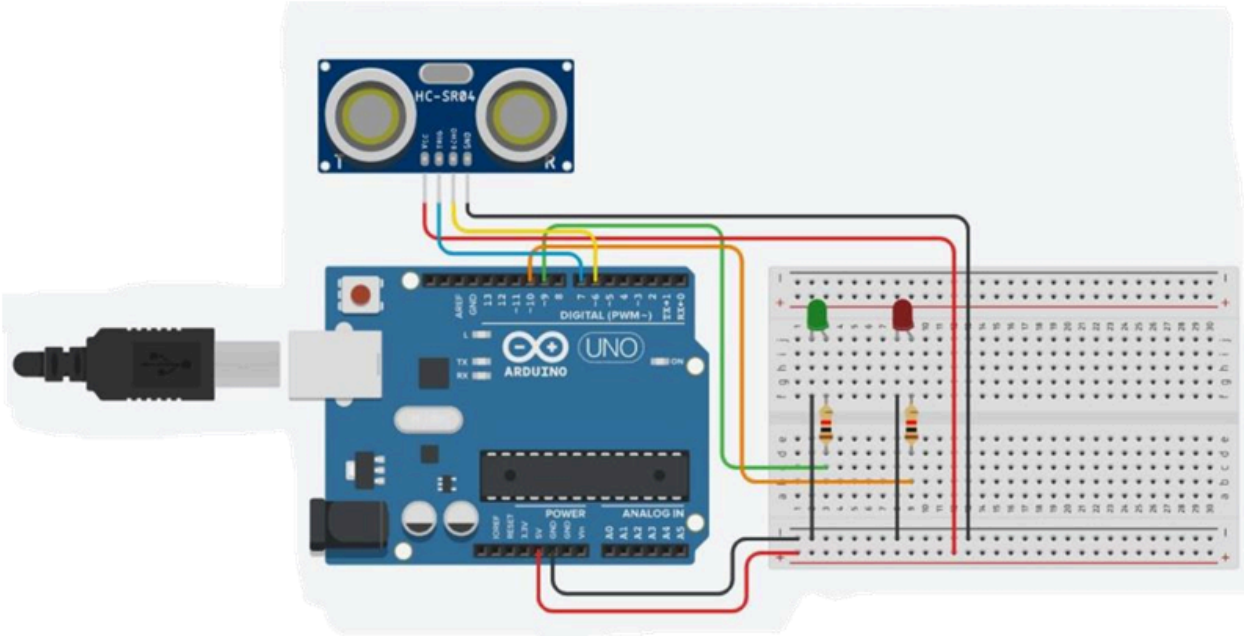
Distance Calculation Formula:

$$Distance = (Time * Speed) / 2$$

The ESP8266 microcontroller processes the time readings from the HC-SR04, calculates the distance, and triggers LED indicators:

- Green LED ON: Object is within the safe range (10 cm to 15 cm).
- Red LED ON: Object is either too close or too far (outside the safe range).

Circuit Diagram:



The circuit diagram represents the connections, with variations made to align with the GPIO pin mappings in the code. The connections are as follows:

- **HC-SR04 Pins:**
  - **VCC** → 5V
  - **GND** → GND
  - **TRIG** → GPIO5 (D1)
  - **ECHO** → GPIO4 (D2)
- **LED Connections:**
  - **Green LED** → GPIO12 (D6)
  - **Red LED** → GPIO13 (D7

Software Implementation :

● **SOUND_VELOCITY:** Defined as 0.034 cm/µs, the average speed of sound in air.

● **CM_TO_INCH:** Conversion constant from centimeters to inches.

Procedure:

1. **Connect the HC-SR04 Ultrasonic Sensor:**
    - **VCC** → 5V
    - **GND** → GND
    - **Trig** → GPIO5
    - **Echo** → GPIO18
2. **Connect the LEDs:**
    - **Red LED** → GPIO19
    - **Green LED** → GPIO21
    - Use appropriate current-limiting resistors to protect the LEDs.
3. **Upload the Code:**
    - Write or upload the provided sketch to the ESP8266 using Arduino IDE.
4. **Monitor Data:**
    - Open the Serial Monitor in Arduino IDE, setting the baud rate to **115200**.
    - Observe real-time distance readings.
5. **Test the Setup:**

    Place an object at different distances and monitor LED behavior:

    - **Green LED ON:** Object is within the safe range (10–15 cm).
    - **Red LED ON:** Object is outside the safe range.

Result:

The ESP8266 accurately measures the distance using the HC-SR04 ultrasonic sensor and triggers LEDs based on proximity. The **Green LED** lights up when the object is in the desired range, and the **Red LED** activates when the object is too close or too far.

Conclusion:

This experiment effectively showcases a distance monitoring system using an ultrasonic sensor and LEDs for visual feedback.

The setup is a fundamental example of obstacle detection and can be adapted for practical applications such as:

- Automated gates
- Parking assistance
- Proximity-based alerts in Wireless Sensor Networks (WSN)

Experiment 3 : Controlling LEDs via Web Interface using ESP8266 Web Server

Objective:

Design a web-based control system using the ESP8266 microcontroller, enabling users to toggle Red, Green, and Yellow LEDs through a browser interface connected via a local Wi-Fi network.

Hardware Requirements:

- ESP8266 NodeMCU board
- Red, Green, and Yellow LEDs
- 220Ω resistors (for current limiting)
- Breadboard
- Jumper wires
- Micro USB cable

Software Requirements:

- Arduino IDE
- ESP8266 Board Support Package
- **ESP8266WiFi** and **ESP8266WebServer** libraries

Theoretical Background:

Wireless Sensor Networks (WSNs) often incorporate web interfaces for remote monitoring and control. By using the ESP8266 as a Wi-Fi-enabled microcontroller, we can host a local web server to control connected devices.

- **Server Functionality:** Hosts an HTML page for user interaction.
- **Communication Protocol:** Handles HTTP requests to toggle LEDs.
- **Control Logic:** Switches LEDs based on the endpoint accessed.

Circuit Description:

- **LED Connections:**
  - **Red LED:** GPIO14 (D5)
  - **Green LED:** GPIO2 (D4)
  - **Yellow LED:** GPIO16 (D0)
- Each LED is connected in series with a 220Ω resistor for current limiting.
- The GPIO pins of the ESP8266 are configured as outputs to drive the LEDs.

Software Implementation:

Key Features:

1. **Libraries Used:**
   - **ESP8266WiFi**: Handles Wi-Fi connectivity.
   - **ESP8266WebServer**: Creates an HTTP server on port 80.
2. **Endpoints:**
   - `/`: Displays the homepage with buttons to control LEDs.
   - `/red`: Turns on the Red LED.
   - `/green`: Turns on the Green LED.
   - `/yellow`: Turns on the Yellow LED.
3. **LED Control Logic:**
   - When an LED-specific endpoint is accessed:
     - All LEDs are first turned OFF.
     - The requested LED is turned ON.
   - The server sends a **303 See Other** response, redirecting the user back to the homepage.

Implementation Note : Ensure the HTML variable in `handleRoot()` contains the proper string for the interface. The HTML should have buttons or links to the respective LED control routes.

Observations:

1. The ESP8266 successfully connects to the specified Wi-Fi network, displaying the local IP address in the Serial Monitor.
2. Users can access the control interface via this IP address in a browser.
3. Clicking on any button toggles the corresponding LED and refreshes the interface.

Results:

The system behaves as expected, with the following outcomes:

Web Route	LED State
/red	Red ON, others OFF
/green	Green ON, others OFF
/yellow	Yellow ON, others OFF
/	Displays LED control page

Conclusion:

This experiment demonstrates the ability of the ESP8266 microcontroller to act as a simple web server, allowing remote control of GPIO outputs through a web browser. This setup provides a foundational framework for developing **IoT applications**, such as:

- Smart home automation systems
- Remote-controlled electronics
- IoT-based monitoring and control nodes

Experiment 4: Controlling Servo Motor using Arduino (ESP8266 Conceptual) – TinkerCAD Simulation

Objective: Using Arduino Uno Controlling servo motor in Tinkercad**.**

a. Only controlling the servo motor

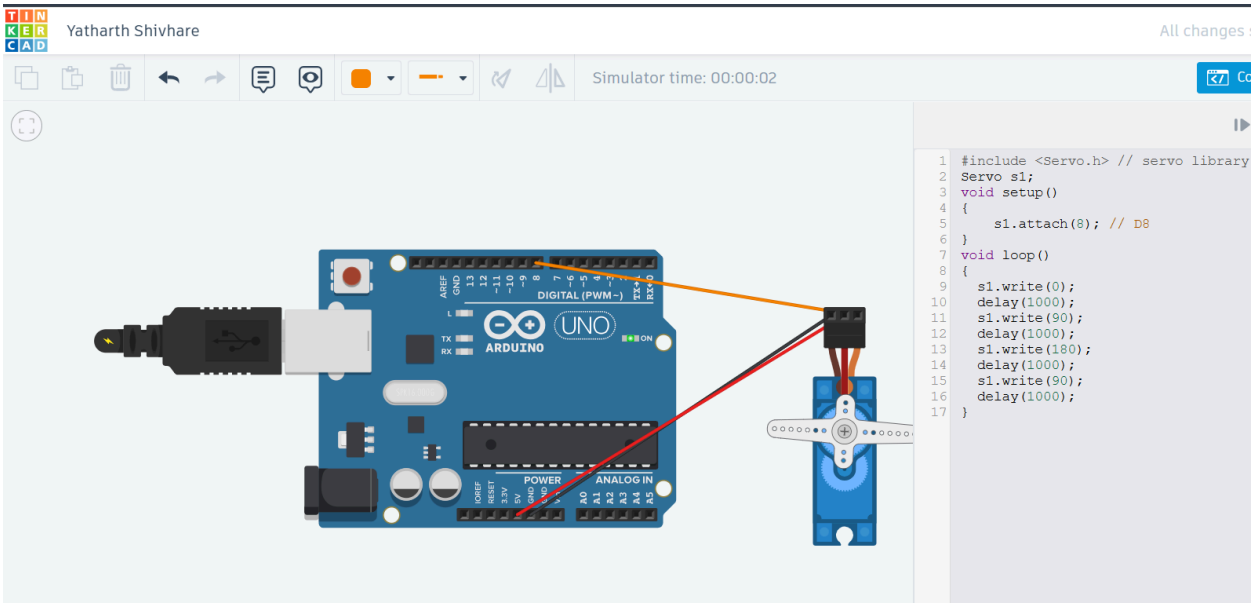
b. Specific angles 0, 90, 180, 270 degrees

c. Application as a speedometer)

Components Required in Tinkercad:

1. Arduino Uno R3
2. Micro Servo

Circuit Connection :



Arduino Code :

```
#include <Servo.h> // servo library

Servo s1;

void setup()
{
    s1.attach(8); // D8
}

void loop()
{
    s1.write(0);
    delay(1000);
    s1.write(90);
    delay(1000);
    s1.write(180);
    delay(1000);
    s1.write(90);
    delay(1000);
}
```

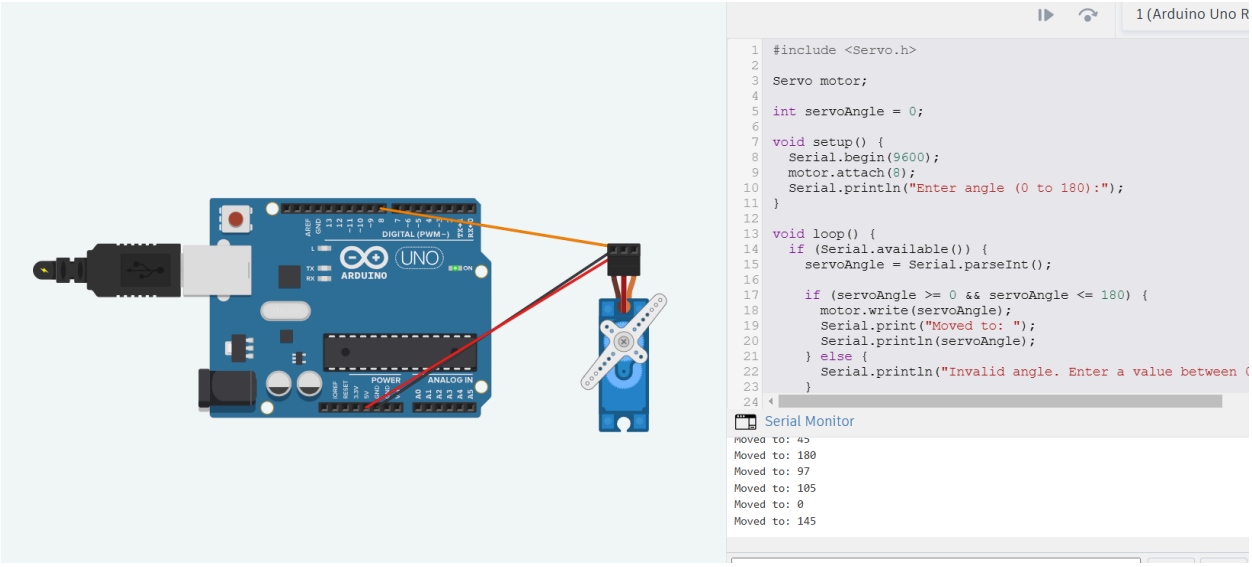
Working :

The servo motor alternates between positions: moves to 0°, pauses for 1 second, moves to 90°, pauses for 1 second, moves to 180°, pauses for 1 second, and returns to 90° before repeating the cycle continuously.

B) In Tinkercad, direct WiFi functionality is unavailable. Therefore, we simulate WiFi control by using the Serial Monitor to input user commands for controlling the servo motor.

Simulation Setup and Component Connections:

- **Servo Motor Connections:**
  - Signal Pin -> D8 of Arduino Uno
  - VCC -> 5V
  - GND -> GND



Arduino Code :

```
#include <Servo.h>

Servo motor;

int servoAngle = 0;

void setup() {

  Serial.begin(9600);

  motor.attach(8);

  Serial.println("Enter angle (0 to 180):");

}

void loop() {

  if (Serial.available()) {

    servoAngle = Serial.parseInt();

    if (servoAngle >= 0 && servoAngle <= 180) {

      motor.write(servoAngle);

      Serial.print("Moved to: ");

      Serial.println(servoAngle);

    } else {

      Serial.println("Invalid angle. Enter a value between 0 and 180.");

    }

    while (Serial.available()) {

      Serial.read(); // Clear the serial buffer

    }

  }

}
```

Working :

Input from Serial Monitor: "145" moves the servo to 145°, "105" moves the servo to 105°, "87" moves the servo to 87°.

Invalid input: Displays an error message asking for a valid angle between 0 and 180.

Link to TinkerCad circuit :

https://www.tinkercad.com/things/cO11ep951AR/editel?returnTo=%2Fdashboard&sharecode=lsnZwABcEzIpWQyWSk3SMlsWOa1klPTEohk1FNDgRu0