Name:_____ Class: D16 Roll No: __

EXPERIMENT NO: 06

Aim: To Create ESP8266 NodeMCU Web Server using WiFi Station (STA) mode and to interface temperature and humidity sensor with Node MCU (ESP8266) and obtain output on node MCU server dashboard.

Apparatus: NODE MCU ESP8266, DHT 11 temperature and Humidity sensor, Micro USB Cable, connectors/connecting wires.

Theory:

In this experiment, we are going to configure our ESP8266 NodeMCU into Station (STA) mode, and create a web server to serve up web pages to any connected client under an existing network For this first we need to modify the following two variables with your network credentials, so that ESP8266 NodeMCU can establish a connection with the existing network const char ssid = "YourNetworkName"; // Enter SSID here

Hardware Connections And Circuit Diagram:

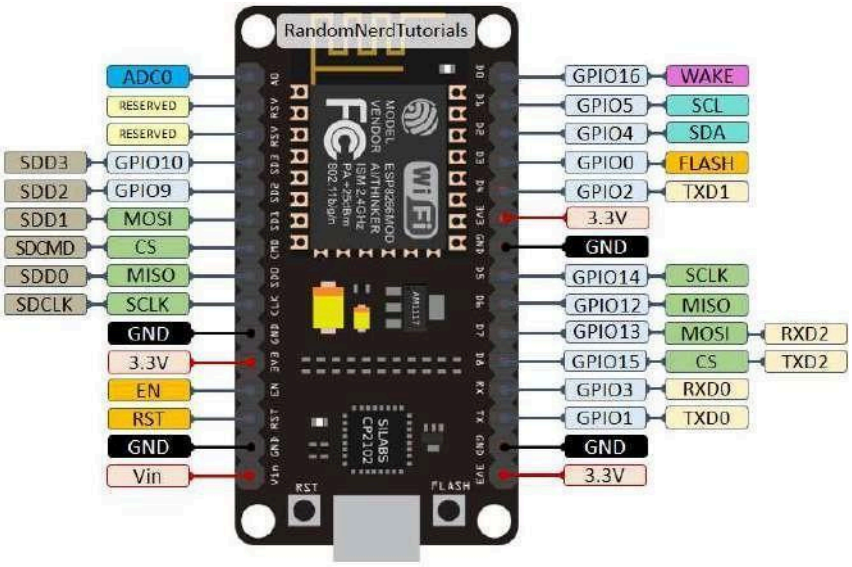


Fig 1. Node MCU GPIO Pin Connections

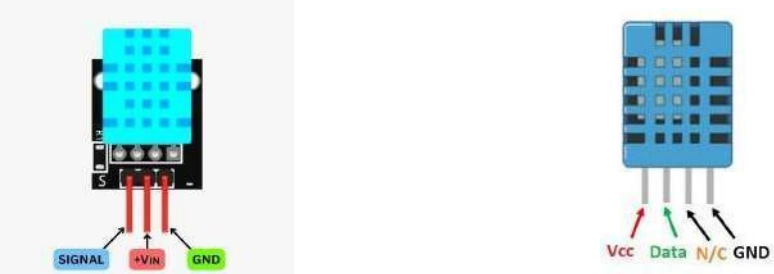


Fig 2. DHT 11 Sensor

For DHT 11 Sensor Module		
1	VCC	Power Supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	Ground	Connected to the ground of the circuit

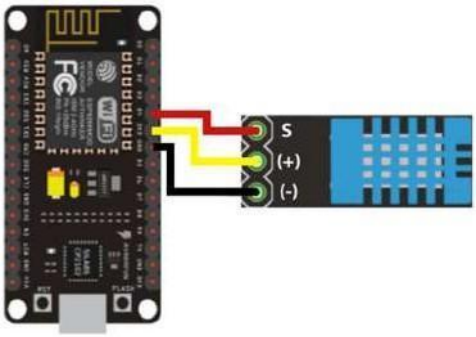


Fig 3. Circuit Diagram

Wiring the DHT11 to the Node MCU is really easy, but the connections are different depending on which type you have either 3-pins or 4-pins The wiring connections are made as follows:

- Pin 1 of the DHT11 goes into +3V of the Node MCU.
- Pin 2 of the DHT11 goes into Digital Pin D0 of the Node MCU.
- Pin 3 of the DHT 11 goes into Ground Pin (GND) of the Node MCU.

PROCEDURE:

1. Make the connections as specified in the circuit diagram.
2. Provide the power supply to NODE MCU by connecting it with the help of micro-USB cable to the CPU of the computer.
3. Open the Arduino IDE and then choose New where a new sketch opens up.
4. Write the code on to a new sketch and then click on save.
5. Before using the DHT11 with Node MCU, install the DHTLib library. It has all the functions needed to get the humidity and temperature readings from the sensor.
6. Then go to Sketch > Include Library Manage Libraries > Search DHTLib.
7. Include ESP8266WiFi.h library. This library provides ESP8266 NodeMCU specific Wi-Fi methods we are calling to connect to the network
8. Following that we also include the ESP8266WebServer.h library, which has some methods available that will help us setting up a server and handle incoming HTTP requests.
9. Upload the code to the NODE MCU by selecting the right board and port.
10. If there are no errors, the code will be compiled and uploaded to the Node MCU successfully else check for the errors mentioned.

CODE:

For Arduino IDE(DHT11 data sent to Firebase):

```
#include "DHT.h"
#define DHTPIN D7
#define DHTTYPE DHT11

#include <Arduino.h>
#if defined(ESP32)
  #include <WiFi.h>
#elif defined(ESP8266)
  #include <ESP8266WiFi.h>
#endif
#include <Firebase_ESP_Client.h>
```

```
DHT dht(DHTPIN, DHTTYPE);

// Token and helper includes
#include "addons/TokenHelper.h"
#include "addons/RTDBHelper.h"

// Wi-Fi credentials
#define WIFI_SSID "shreyash"
#define WIFI_PASSWORD "shreyash"

// Firebase API Key and DB URL
#define API_KEY "AIzaSyBfXAOSed-XLFoPW-Hg1K22hvZ-eJaAsSM"
#define DATABASE_URL "https://weather-monitering-28fd5-default-rtdb.firebaseio.com/"

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

bool signupOK = false;

void setup(){
  pinMode(DHTPIN, INPUT);
  dht.begin();
  Serial.begin(115200);

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED){
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());

  config.api_key = API_KEY;
```

```
  config.database_url = DATABASE_URL;

  if (Firebase.signUp(&config, &auth, "", "")){
    Serial.println("SignUp OK");
    signupOK = true;
  } else {
    Serial.printf("SignUp FAILED: %s\n", config.signer.signupError.message.c_str());
  }

  config.token_status_callback = tokenStatusCallback;
  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);
}

void loop(){
  delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)){
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  if (Firebase.ready() && signupOK ) {
    if (Firebase.RTDB.setFloat(&fbdo, "DHT/humidity", h)){
      Serial.print("Humidity: ");
      Serial.println(h);
    } else {
      Serial.println("FAILED: " + fbdo.errorReason());
    }

    if (Firebase.RTDB.setFloat(&fbdo, "DHT/temperature", t)){
      Serial.print("Temperature: ");
      Serial.println(t);
    } else {
```

```
    Serial.println("FAILED: " + fbdo.errorReason());
   }
  }
  Serial.println("_____");
}
```

For Dashboard(Firebase to Webdashboard):

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Firebase Weather Dashboard</title>
   <script src="https://www.gstatic.com/firebasejs/8.10.1/firebase-app.js"></script>
   <script src="https://www.gstatic.com/firebasejs/8.10.1/firebase-database.js"></script>
   <style>
      @import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap');

      :root {
         --bg-color: #121212;
         --card-color: #1e1e1e;
         --text-color: #e0e0e0;
         --primary-color: #4a90e2;
         --secondary-color: #50e3c2;
         --shadow-dark: rgba(0, 0, 0, 0.4);
      }

      body {
         font-family: 'Poppins', sans-serif;
         background-color: var(--bg-color);
         color: var(--text-color);
         display: flex;
         justify-content: center;
         align-items: center;
         min-height: 100vh;
         margin: 0;
```

```
    transition: background-color 0.3s ease;
}

.dashboard-container {
    text-align: center;
    padding: 3rem;
    border-radius: 15px;
    box-shadow: 0 10px 30px var(--shadow-dark);
    background: linear-gradient(145deg, var(--card-color), #2a2a2a);
    width: 90%;
    max-width: 700px;
}

h1 {
    font-weight: 600;
    margin-bottom: 0.5rem;
    font-size: 2.5em;
    color: var(--primary-color);
    text-shadow: 0 0 10px var(--primary-color);
}

.card-container {
    display: flex;
    justify-content: space-around;
    gap: 2rem;
    margin-top: 2rem;
}

.card {
    flex: 1;
    background-color: var(--card-color);
    padding: 2rem;
    border-radius: 12px;
    box-shadow: 0 4px 15px var(--shadow-dark);
    transition: transform 0.3s ease, box-shadow 0.3s ease;
}
```

```
.card:hover {
    transform: translateY(-10px);
    box-shadow: 0 15px 35px var(--shadow-dark);
}

.icon {
    font-size: 3rem;
    margin-bottom: 0.5rem;
}

h2 {
    font-size: 1.2em;
    font-weight: 400;
    margin: 0;
    color: var(--text-color);
}

p {
    font-size: 3.5em;
    font-weight: 600;
    margin: 0.2em 0 0;
    color: var(--secondary-color);
}

#temperature:after {
    content: "°C";
    font-size: 0.4em;
    vertical-align: super;
    font-weight: 300;
}

#humidity:after {
    content: "%";
    font-size: 0.4em;
    vertical-align: super;
```

```
            font-weight: 300;
        }
    </style>
</head>
<body>
    <div class="dashboard-container">
        <h1>Firebase Weather Dashboard</h1>
        <div class="card-container">
            <div class="card">
                <span class="icon"> 🌡️ </span>
                <h2>Temperature</h2>
                <p id="temperature">--</p>
            </div>
            <div class="card">
                <span class="icon"> 💧 </span>
                <h2>Humidity</h2>
                <p id="humidity">--</p>
            </div>
        </div>
    </div>

    <script>
        // Replace with your Firebase project's config
        const firebaseConfig = {
            apiKey: "AIzaSyBfXAOSed-XLFoPW-Hg1K22hvZ-eJaAsSM",
            databaseURL: "https://weather-monitering-28fd5-default-rtdb.firebaseio.com/",
        };

        // Initialize Firebase
        firebase.initializeApp(firebaseConfig);

        // Get a reference to the database service
        const database = firebase.database();

        // Get a reference to the 'DHT' node in the database
        const dhtRef = database.ref('DHT');
```

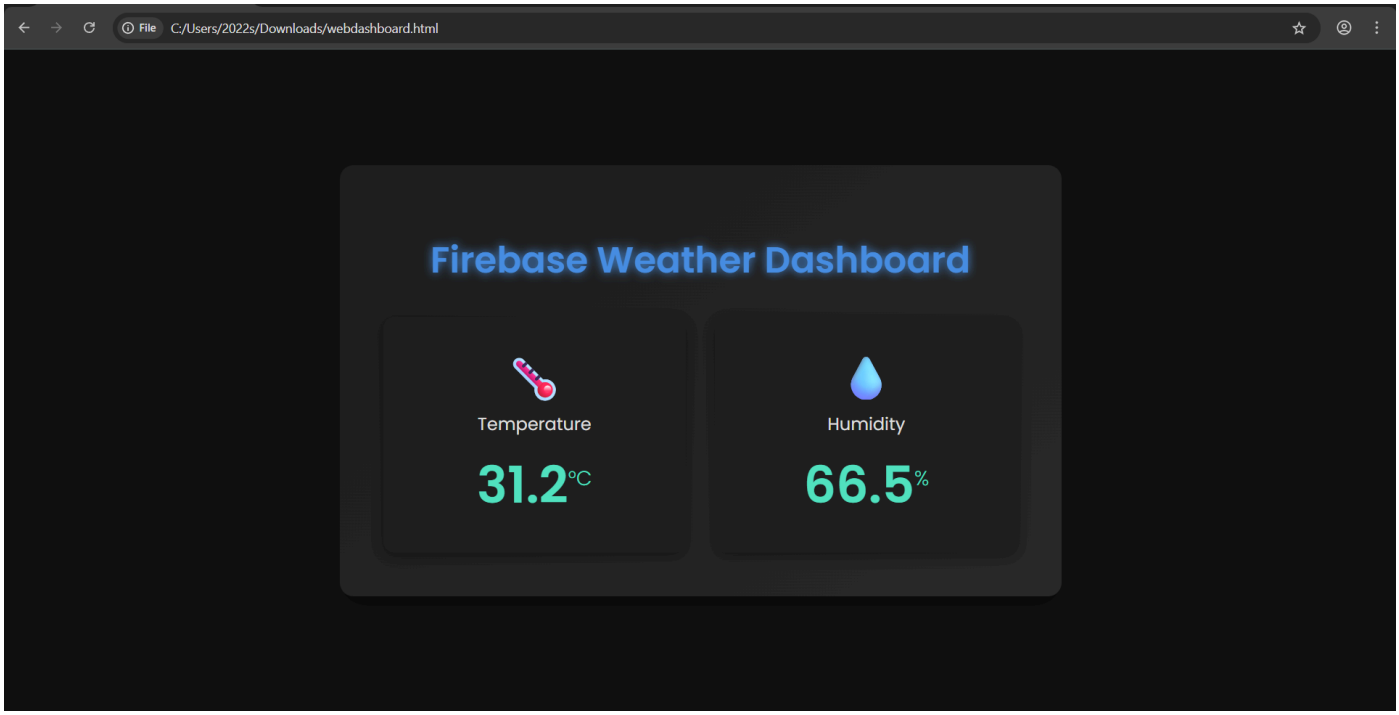
```
        // Listen for changes in the 'DHT' node
        dhtRef.on('value', (snapshot) => {
            const data = snapshot.val();
            const tempElement = document.getElementById('temperature');
            const humElement = document.getElementById('humidity');

            if (data) {
                const temperature = data.temperature;
                const humidity = data.humidity;

                tempElement.textContent = temperature.toFixed(1);
                humElement.textContent = humidity.toFixed(1);
            } else {
                console.log("No data available from Firebase.");
                tempElement.textContent = '--';
                humElement.textContent = '--';
            }
        }, (error) => {
            console.error("Error fetching data:", error);
            document.getElementById('temperature').textContent = 'Error';
            document.getElementById('humidity').textContent = 'Error';
        });
    </script>
</body>
</html>
```

Accessing the Web Server:

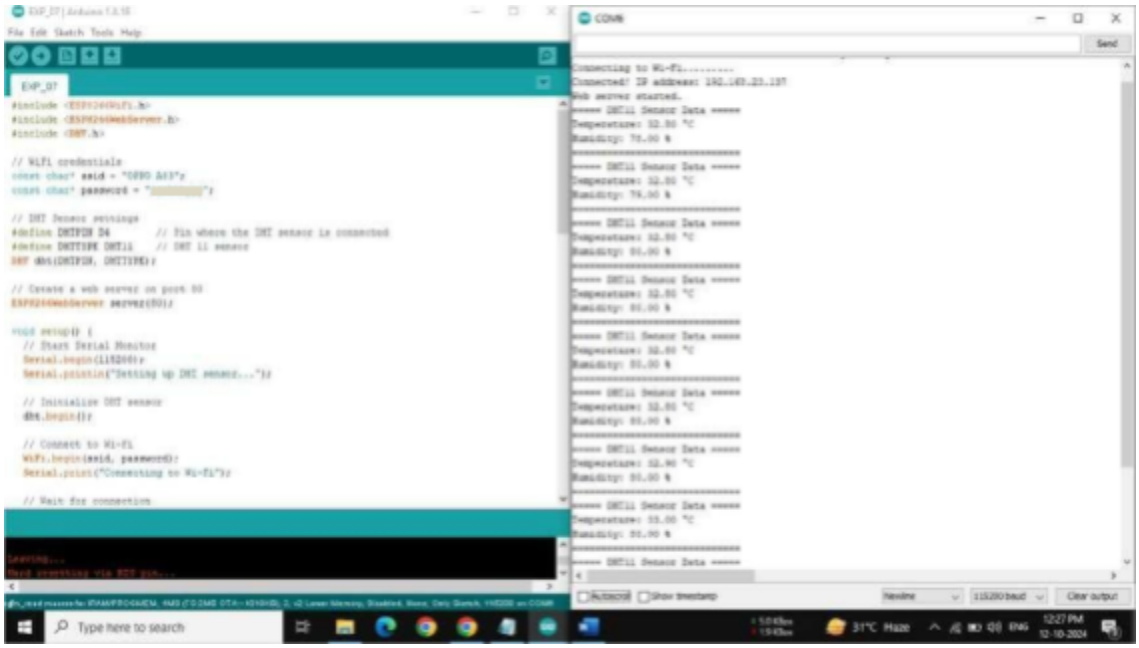
After uploading the sketch, open the Serial Monitor at a baud rate of 115200. Press the RESET button on the NodeMCU. If everything is OK, it will output the dynamic IP address obtained from your router and show the HTTP server started message.

OUTPUT:

1. Screenshot of the dashboard



2. On running the code in Arduino IDE, Screenshot of Serial Monitor



CONCLUSION:

1. Learnt about Web Server
2. Learnt to create webserver using Arduino IDE
3. Learnt to post data using ESP8266 on a Web Server
4. Learn the embed<html> code with ‘C’ on Arduino.