Real-time Environmental Monitoring System with ESP32 and DHT11 Sensor Integration via Web Server

A report

submitted by

Mekala Manikanta - 21BEC0992

Dintakurthi Swaroop -21BEC2052

Kasireddy Chaitanya – 21BEC2295

From F2 Slot

SENSE

submitted to

Prof. Manish Kumar

in partial fulfillment for the requirements of the course

Embedded C programming (BECE320E)

SCHOOL OF ELECTRONICS AND COMMUNICATION ENGNEERING

VELLORE INSTITUTE OF TECHNOLOGY

VELLORE-632014

MAY 2024

Real-time Environmental Monitoring System with ESP32 and DHT11 Sensor Integration via Web Server

Abstract:

This project aims to create a real-time environmental monitoring system using an ESP32 microcontroller and a DHT11 sensor, integrated with a web server. The ESP32 will operate in station mode, enabling it to connect to an existing Wi-Fi network for data communication.

The system will continuously read temperature and humidity values from the DHT11 sensor. These values will be processed by the ESP32 microcontroller and then displayed on a web server interface. Users can access this interface through a web browser on their devices connected to the same Wi-Fi network.

Key functionalities of the system include:

1.Sensor Data Acquisition: The DHT11 sensor will provide real-time temperature and humidity readings to the ESP32 microcontroller.

2.Wi-Fi Connectivity: The ESP32 will operate in station mode to connect to an existing Wi-Fi network, enabling seamless data transfer.

3.Web Server Integration: A web server hosted on the ESP32 will serve a user-friendly interface displaying the sensor data.

4.Real-time Monitoring: Users can monitor environmental conditions in real-time through a web browser interface, accessible from devices connected to the same Wi-Fi network.

This system facilitates convenient and remote monitoring of environmental parameters, making it suitable for various applications such as home automation, weather monitoring, and industrial control.

Keywords:

ESP32, DHT11 sensor, Web server, Station mode, Wi-Fi connectivity, Real-time monitoring, Environmental monitoring, Remote sensing, Internet of Things (IoT), Microcontroller-based system

Introduction:

This project aims to create a versatile system for real-time monitoring of temperature and humidity using an ESP32 microcontroller and a DHT11 sensor. The ESP32 will operate in station mode, connecting to an existing Wi-Fi network to facilitate data communication. A web server hosted on the ESP32 will display the sensor readings, providing users with easy access to environmental data. This system is ideal for applications requiring remote monitoring and control of environmental parameters.

Components Name	Quantity	Description
Breadboard	1	Small Size
ESP32 microcontroller	1	Controls the system and connects to the Wi-Fi network in station mode.
DHT11 sensor	1	Measures temperature and humidity data in real-time.
Web server	1	Hosted on ESP32, provides a user-friendly interface for displaying sensor readings.
Wi-Fi network	1	Existing network to which the ESP32 connects for data communication.

Table1: Component Requirements

DHT11 Sensor

The DHT11 sensor is a widely used digital temperature and humidity sensor known for its simplicity and reliability. It features a calibrated digital signal output, making it easy to interface with microcontrollers like the ESP32. The sensor's compact size and low power consumption make it suitable for various applications, from environmental monitoring systems to home automation projects.

The DHT11 sensor operates on a single-bus communication protocol, simplifying its integration into microcontroller-based projects. It uses a proprietary digital signal transmission format to provide accurate temperature readings ranging from 0°C to 50°C with a ±2°C accuracy and humidity readings ranging from 20% to 80% RH with a ±5% RH accuracy.

DHT11 sensor contains 3 pins. Following table shows the description of all the 3 pins in the DHT11 sensor:

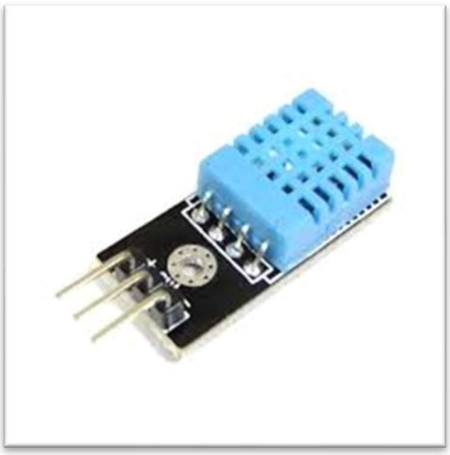


Fig -1: DHT11 Sensor

No	Name	Description
1	Vcc	The power supply pin of the sensor. It is connected to5V DC
2	DATA	Digital data output (connects to microcontroller)
3	GND	This pin is connected to ground

Table -2: DHT11 sensor PINs

Working of Ultra Sonic Sensor

The DHT11 sensor operates by using a humidity sensing component and a calibrated temperature measurement component to detect environmental conditions. It converts the analog signals from these components into digital signals, which are then transmitted through a single-wire communication protocol. When the microcontroller initiates a data request, the sensor responds by sending a data packet containing temperature and humidity information. The microcontroller processes these digital signals and interprets the data, allowing users to monitor and analyze real-time temperature and humidity levels in their applications.

How to Calculate Readings

To calculate the readings from a DHT11 sensor, you typically follow these steps:

Data Acquisition: Begin by initializing communication with the DHT11 sensor using a microcontroller such as the ESP32. Send a start signal to the sensor to request data.

Data Transmission: The DHT11 sensor responds by sending a data packet. This packet includes both temperature and humidity readings encoded in a specific format.

Data Decoding: Decode the data packet received from the sensor. The DHT11 uses a 40-bit data format where the first 16 bits represent the integer part of the humidity, the next 16 bits represent the integer part of the temperature, the next 8 bits represent the decimal part of the humidity, and the last 8 bits represent the decimal part of the temperature.

Calculation: Combine the integer and decimal parts to calculate the final temperature and humidity readings. For example, if the integer part of the temperature is 25°C and the decimal part is 5, the actual temperature is 25.5°C. Similarly, if the integer part of the humidity is 50% and the decimal part is 2, the actual humidity is 50.2%.

Display or Use: Once calculated, you can display the temperature and humidity readings on a display device or use them in your application for further processing or control actions.

Note: Different microcontrollers may have specific libraries or functions to simplify the decoding and calculation process for DHT11 sensor readings. Always refer to the documentation or resources provided for your microcontroller platform for accurate calculations and implementations.

HARDWARE ASSEMBLY

The setup utilizes an ESP32 microcontroller, a DHT11 sensor, and the Arduino IDE software to create a system for displaying temperature and humidity values on a web server. The ESP32, known for its powerful capabilities and built-in Wi-Fi functionality, is well-suited for IoT applications. On the other hand, the DHT11 sensor, despite being basic and affordable, effectively measures temperature and humidity levels. In the Arduino IDE, code is written to interact with the DHT11 sensor and establish a web server on the ESP32 using libraries like DHT.h for sensor communication and WiFi.h for network connectivity. Once the code is uploaded, the ESP32 connects to the specified Wi-Fi network and reads data from the DHT11 sensor, presenting it on a webpage. Users can simply enter the ESP32's IP address in a web browser to access and monitor the real-time temperature and humidity values remotely, offering a user-friendly solution for environmental monitoring.

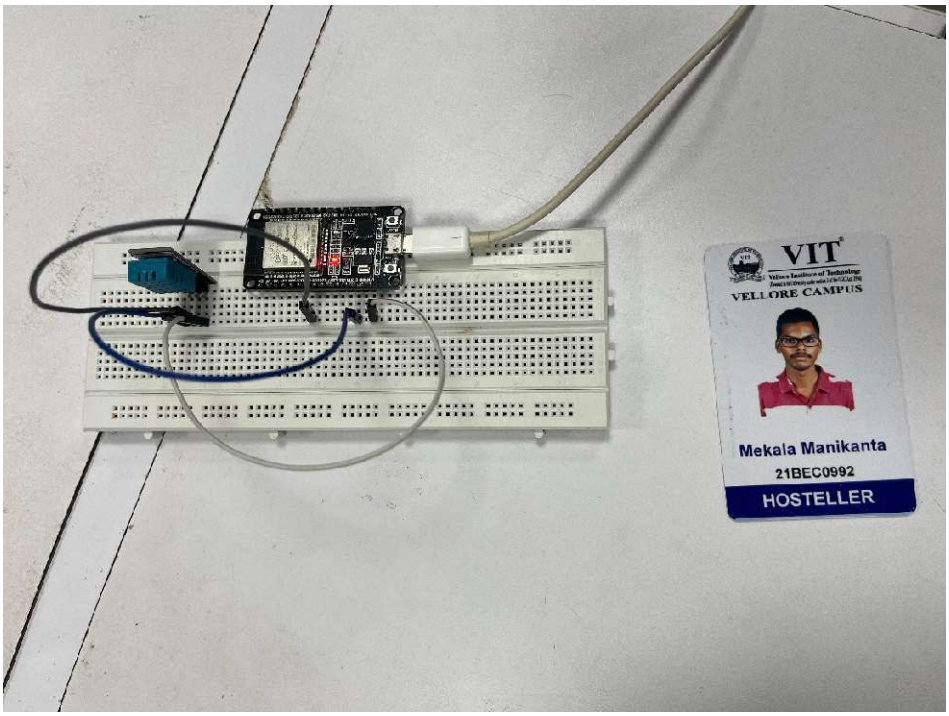


Fig -2: Sensor Circuit

SOFTWARE (Arduino IDE)

```
#include "WiFi.h"
#include "ESPAsyncWebServer.h"
#include <Adafruit_Sensor.h>
#include <DHT.h>

const char* ssid = "PRP108"; // Replace with your network credentials
const char* password = "108iotlab";

#define DHTPIN D5 // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11 // Uncomment for DHT 11 sensor

DHT dht(DHTPIN, DHTTYPE);

AsyncWebServer server(80);

String readDHTTemperature() {
  float t = dht.readTemperature();
  if (isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return "--";
  } else {
    Serial.println(t);
    return String(t);
  }
}

String readDHTHumidity() {
  float h = dht.readHumidity();
```

```
  if (isnan(h)) {
    Serial.println("Failed to read from DHT sensor!");
    return "--";
  } else {
    Serial.println(h);
    return String(h);
  }
}


const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css" integrity="sha384- fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr" crossorigin="anonymous">
<style>
html {
font-family: Arial;
display: inline-block;
margin: 0px auto;
text-align: center;
}
h2 { font-size: 3.0rem; }
p { font-size: 3.0rem; }
.units { font-size: 1.2rem; }
.dht-labels{
font-size: 1.5rem;
vertical-align:middle;
padding-bottom: 15px;
}
```

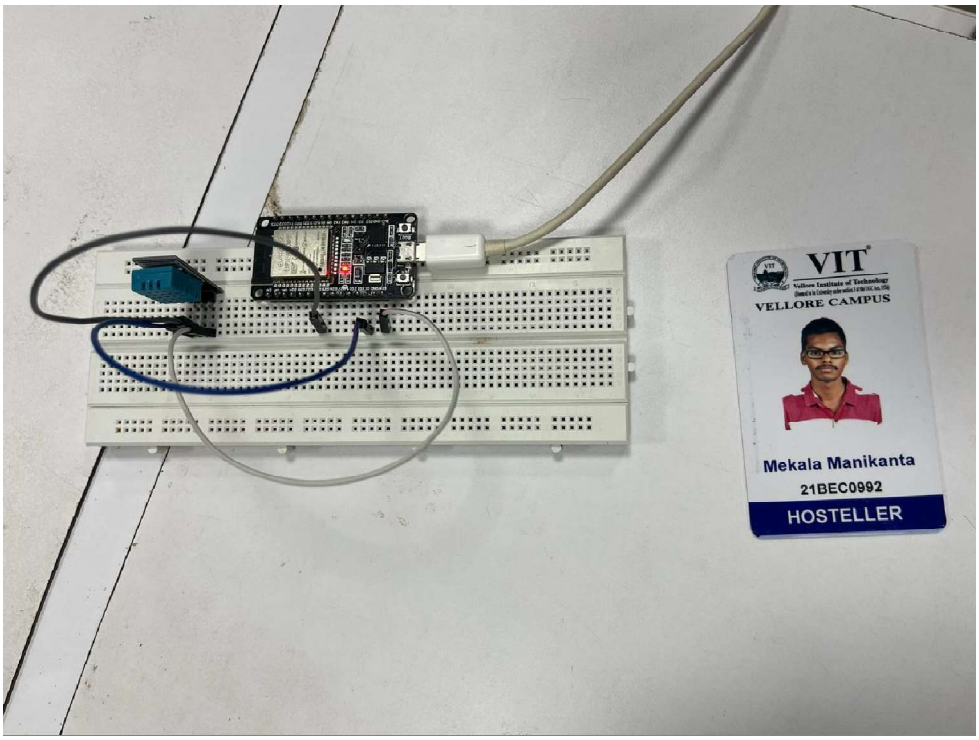
```
</style>
</head>
<body>
<h2>VIT Weather</h2>
<p>
<i class="fas fa-thermometer-half" style="color:#059e8a;"></i>
<span class="dht-labels">Temperature</span>
<span id="temperature">%TEMPERATURE%</span>
<sup class="units">&deg;C</sup>
</p>
<p>
<i class="fas fa-tint" style="color:#00add6;"></i>
<span class="dht-labels">Humidity</span>
<span id="humidity">%HUMIDITY%</span>
<sup class="units">&percnt;</sup>
</p>
</body>
<script>
setInterval(function() {
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
if (this.readyState == 4 && this.status == 200) {
document.getElementById("temperature").innerHTML = this.responseText;
}
};
xhttp.open("GET", "/temperature", true);
xhttp.send();
}, 10000);
setInterval(function() {
var xhttp = new XMLHttpRequest();
```

```
xhttp.onreadystatechange = function() {
if (this.readyState == 4 && this.status == 200) {
document.getElementById("humidity").innerHTML = this.responseText;
}
};
xhttp.open("GET", "/humidity", true);
xhttp.send();
}, 10000);
</script>
</html>)rawliteral";

String processor(const String& var){
if(var == "TEMPERATURE"){
return readDHTTemperature();
}
else if(var == "HUMIDITY"){
return readDHTHumidity();
}
return String();
}

void setup(){
Serial.begin(115200);
dht.begin();

WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(1000);
Serial.println("Connecting to WiFi..");
}
```

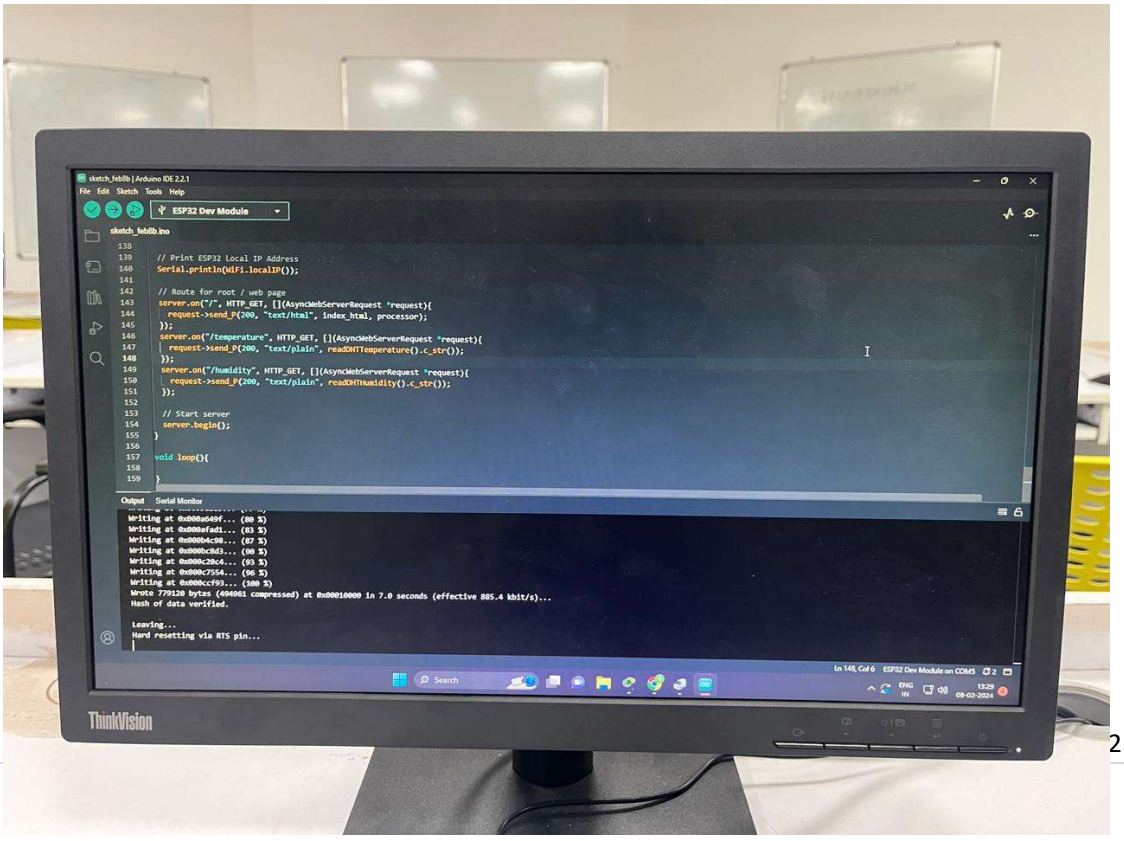
```
Serial.println(WiFi.localIP());

server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
request->send_P(200, "text/html", index_html, processor);
});

server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest *request){
request->send_P(200, "text/plain", readDHTTemperature().c_str());
});

server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest *request){
request->send_P(200, "text/plain", readDHTHumidity().c_str());
});

server.begin();
}

void loop(){
}
```

Output/Results:

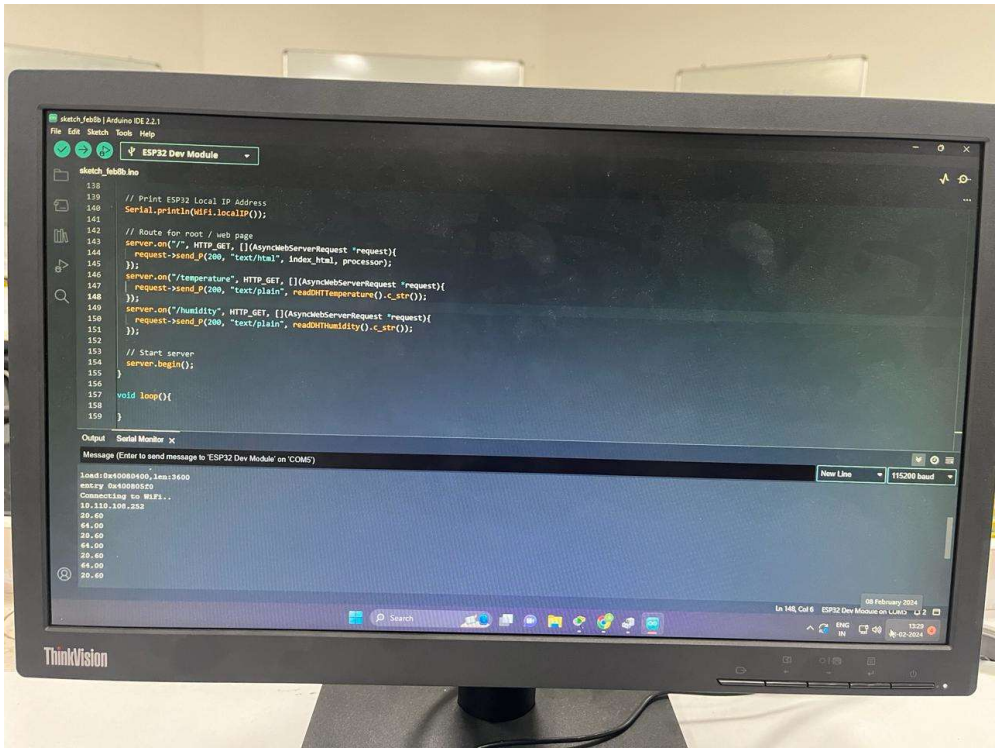
Setup the experiment showing below



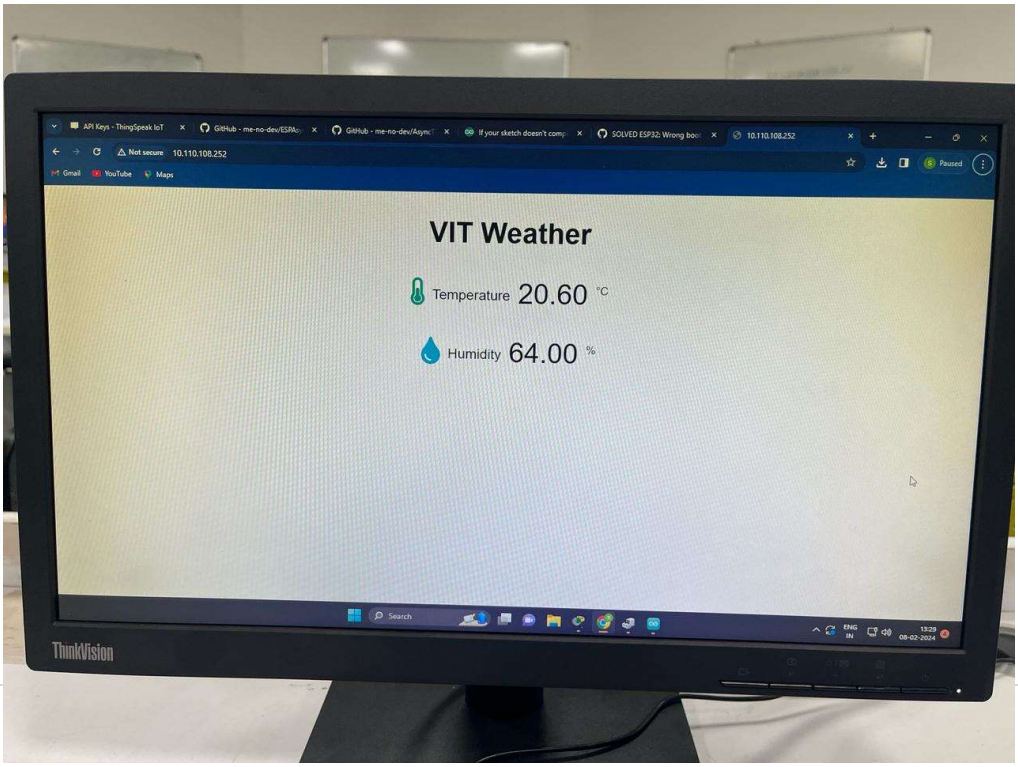
Write code and verify and upload



goto serial monitor and copy the IP address



After copying the IP address and paste in google chrome and get the output of temperature and humidity



Connect the ESP32 to the CPU through micro USB cable. Go to Arduino ide software and select the esp32 dev module port. Type the code for working of the sensor and interface with web. Connect DHT11 sensor to the ESP32 according to code Run the code and go to output. Then copy the Ip address in output and browse it. We can directly get the reading of the sensor there.

Procedure for the experiment:

Open Arduino IDE and Type the Code:

Open the Arduino IDE software and write the code to read data from the DHT11 sensor and create a web server on the ESP32.

Include the necessary libraries such as DHT.h and WiFi.h.

Select ESP32 Dev Module and Port:

Go to the Tools menu and select the ESP32 Dev Module as the board.

Select the appropriate COM port (e.g., COM6) for the ESP32 connected via USB.

Change SSID and Password:

Modify the sketch to include your Wi-Fi network's SSID and password.

Verify the sketch and upload it to the ESP32 board.

Run the Code and Access Serial Monitor:

Run the uploaded code on the ESP32.

Open the Serial Monitor in the Arduino IDE and change the baud rate to match the one set in the code (e.g., 115200).

Copy the IP Address:

In the Serial Monitor, note the IP address assigned to the ESP32 by the Wi-Fi network.

Access Web Server:

Open a web browser (e.g., Google Chrome) on a device connected to the same Wi-Fi network.

Enter the IP address of the ESP32 in the address bar and press Enter.

View Temperature and Humidity Values:

The web browser will display a webpage served by the ESP32, showing the current temperature and humidity values read from the DHT11 sensor.

This setup allows for real-time monitoring of temperature and humidity values from the DHT11 sensor using a web browser, thanks to the ESP32's web server capabilities and Wi-Fi connectivity.

Application:

Home Automation: Monitor and control temperature and humidity levels in different rooms of a smart home to ensure comfort and energy efficiency.

Greenhouse Monitoring: Keep track of environmental conditions such as temperature and humidity in a greenhouse to optimize plant growth and health.

Weather Station: Create a DIY weather station that displays real-time weather data including temperature and humidity for personal or educational purposes.

Industrial Monitoring: Monitor environmental conditions in industrial settings to ensure optimal working conditions for machinery and personnel.

Server Room Monitoring: Monitor temperature and humidity levels in server rooms to prevent equipment overheating and ensure data center reliability.

IoT Environmental Sensing: Integrate the system into IoT projects for environmental sensing and data logging applications.

Healthcare Monitoring: Monitor temperature and humidity levels in healthcare facilities to maintain a comfortable and hygienic environment for patients and staff.

Agricultural Monitoring: Monitor environmental conditions in agricultural settings to optimize crop growth and prevent diseases.

Testing and Resulting :

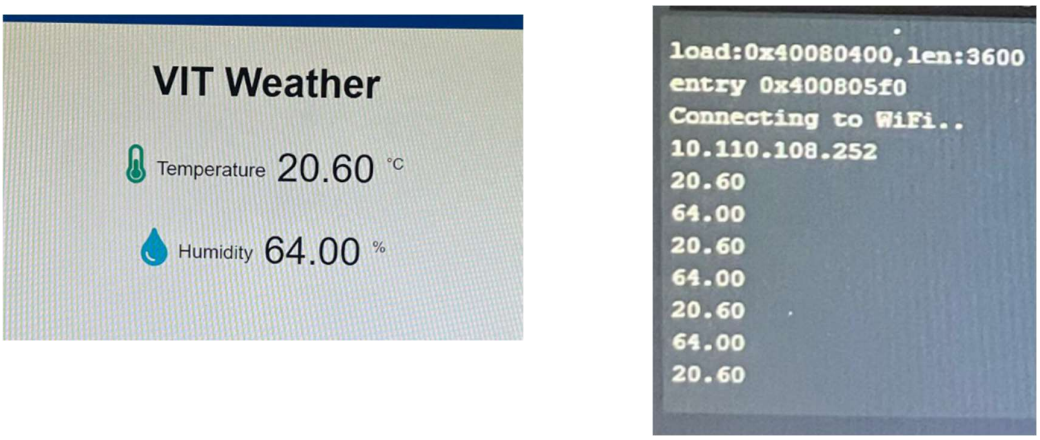


Fig -3: SENSOR CIRCUIT

Conclusion:

In conclusion, the project successfully achieved its aim of displaying temperature and humidity values from a DHT11 sensor on a web server using an ESP32 in station mode. The system was able to connect to an existing Wi-Fi network, and the values were accessible through a web browser. Finally, the experiment done well and the reading of the sensor are visible on web as we browse the Ip address came on output of Arduino software. Also readings on software and web browser are matching.

Reference:

[1] https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-web-server-arduino-ide/

[2] https://lastminuteengineers.com/esp32-dht11-dht22-web-server-tutorial/

[3] https://www.electronicwings.com/esp32/dht11-sensor-interfacing-with-esp32

[4] https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/