**ESP32 SENSOR READINGS IN VALUES – WEBSERVER**

**MINIPROJECT REPORT**

***A report submitted in the partial fulfillment of the requirements for the Award of the Degree of***

**BACHELOR OF TECHNOLOGY**

**in**

**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

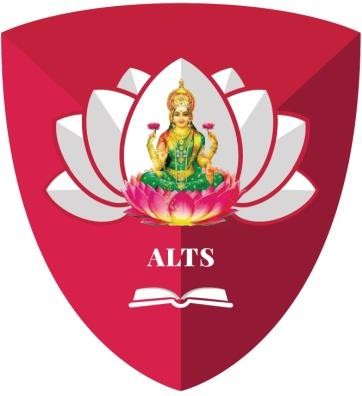
**By**

|  |  |
| --- | --- |
| **GANGULA BHAVITHA REDDY** | **212G1A3913** |
| **CHAKRANTHAM SAI LIKHITHA** | **212G1A3910** |
| **GOLLAPALLI NEEHARIKA** | **212G1A3917** |
| **GORANTLA JASHNAVI** | **212G1A3918** |
| **GHEE SHAIK TAMAN AFSHA** | **212G1A3915** |
| **FABIHA ANJUM** | **212G1A3911** |

**Under Supervision of**

**Mrs. K. SWATHI M. Tech (Ph. D),**

**Assistant Professor in Computer Science and Engineering,**

**ALTS, Ananthapur.**

### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING ANANTHA LAKSHMI INSTITUTE OF TECHNOLOGY AND SCIENCES

**Approved by AICTE, New Delhi & Affiliated to J.N.T.U. Ananthapuram, Accredited by NAAC & NBA**

**Near S.K. University, Itikalapalli(V), Anantapur (Dt) – 515 721.A.P.**

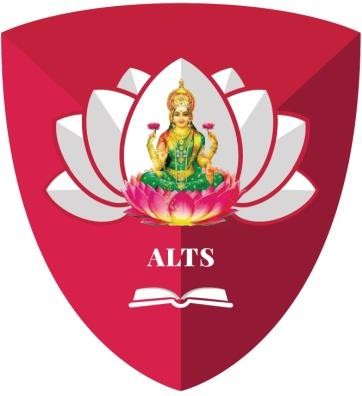
**(2023-2024)**

**ANANTHA LAKSHMI INSTITUTE OF TECHNOLOGY AND SCIENCES**

**Approved by AICTE, New Delhi & Affiliated to J.N.T.U. Ananthapuram, Accredited by NAAC & NBA**

**Near S.K. University, Itikalapalli(V), Anantapur (Dt) – 515 721.A.P.**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

This is to certify that the **“ESP32 SENSOR READINGS IN VALUES – WEB SERVER”** submitted by **GANGULA BHAVITHA REDDY(212G1A3913)**, **CHAKRANTHAM SAI LIKHITHA(212G1A3910), GOLLAPALLI NEEHARIKA(212G1A3917), GORANTLA JASHANVI(212G1A3918) , FABIHA ANJUM(212G1A3911), GHEE SHAIK TAMANAFSHA (212G1A3915)** is the work done by her/him and submitted during 2023-2024 academic year, in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in **ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**.

|  |  |
| --- | --- |
| **Supervisor Signature**  **K. Swathi M.Tech (Ph. D)**  Assistant Professor,  Department of Computer science and Engineering,  Anantha Lakshmi Institute of Technology  and Sciences, Anantapur. | **Head of the Department**  **Dr. K MuralidharM.Tech, Ph.D**  Associate Professor & HOD,  Department of Computer science and Engineering,  Anantha Lakshmi Institute of Technology  and Sciences, Anantapur. |

**DECLARATION OF THE CANDIDATE**

We hereby declare that the work which is being presented in this dissertation entitled “**ESP32 SENSOR READINGS IN VALUES – WEB SERVER**” submitted towards the partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE & ENGINEERING**, Anantha Lakshmi Institute Of Technology And Sciences is an authentic record of our own work carried out During 2023-2024 under the supervision K.Swathi M.Tech(Ph.D), **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING, Anantha Lakshmi Institute Of Technology And Sciences, Anantapuramu.**

The matter embodied in this dissertation report has not been submitted by us for the award of any other degree or diploma. Further, the technical details furnished in the various chapters in this documentation are purely relevant to the above project.

**Submitted By:**

GANGULA BHAVITHA REDDY 212G1A3913

CHAKRANTHAM SAI LIKHITHA 212G1A3910

GOLLAPALLI NEEHARIKA 212G1A3917

GORANTLA JASHNAVI 212G1A3918

GHEE SHAIK TAMAN AFSHA 212G1A3915

FABIHA ANJUM 212G1A3911

7

**ACKNOWLEDGEMENTS**

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that we have now the opportunity to express our gratitude for all of them.

It is with immense pleasure that we would like to express my indebted gratitude to K.SWATHI M.Tech(Ph.D), **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**, who has guided us a lot and encouraged us in every step of mini project. We thank her for the stimulating guidance, constant encouragement and constructive criticism which have made possible to accomplish this work.

It is with immense pleasure that we would like to express my indebted gratitude to **Dr. K Muralidhar M.Tech., Ph.D., Associate Professor & HOD**, **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**, who has guided us a lot and encouraged us in every step of mini project. We thank him for the stimulating guidance, constant encouragement and constructive criticism which have made possible to accomplish this work.

I wish to convey my special thanks to **Dr. N Ramamurthy MTech., Ph.D., Principal** of **Anantha Lakshmi Institute of Technology and Sciences** for giving the required information in doing my project. Not to forget, we thank all other faculty and non-teaching staff, and my friends who had directly or indirectly helped and supported us in completing my project in time.

We also express our sincere thanks to **Sri M. ANANTHA RAMUDU, Chairman** and **Sri M. RAMESH NAIDU, Vice Chairman** of **Anantha Lakshmi Institute of Technology and Sciences** for providing excellent facilities**.**

**Gangula Bhavitha Reddy (212G1A3913),**

**Chrakantham Sai Likhitha (212G1A3910),**

**Gollapalli Neeharika (212G1A3917),**

**Gorantla Jashnavi (212G1A3918),**

**Fabiha Anjum (212G1A3911),**

**Ghee Shaik Taman Afsha (212G1A3915).**

**ABSTRACT**

To display sensor readings (temperature, humidity) on a web server using the ESP32 with Arduino IDE. The ESP will host a web page with two real time values that have new readings added every 30 seconds. We are using the HTML to build the web page and the asynchronous “ESPA sync Web Server library”. The web page will be stored on the ESP32 or ESP8266 Filesystem (SPIFFS). We will be displaying temperature and humidity readings from a BME280 Sensor on a webpage. To display the values we are using different libraries. We are creating two values: Temperature and Humidity over time. The reading is added every 30 seconds.

**LIST OF CONTENTS**

**Page No**

1. **OVERVIEW OF THE PROJECT**
2. **OBJECTIVE OF THE PROJECTTop of Form**
3. **OUTCOME OF THE PROJECT**
4. **INTRODUCTION TO THE PROJECT**
5. **WHY YOU CHOSE THIS PROJECT**
6. **WHAT YOU HOPE TO LEARN**
7. **TOOLS AND TECHNOLOGIES USED**
8. **STEP-BY-STEP APPROACH**
9. **DESCRIPTION OF WHAT YOU BUILT (PROGRAM CODE)**
10. **SCREENSHOTS OR VISUALS OF OUTPUT/RESULTS**
11. **NEW SKILLS ACQUIRED**
12. **INSIGHTS GAINED FROM THE PROJECT**
13. **CONCLUSION**

* **SUMMARY OF THE PROJECT**
* **KEY TAKEAWAYS**
* **NEXT STEPS**

**CHAPTER - 1**

**OVERVIEW OF PROJECT**

The "ESP32 Sensor Readings in Values - Web Server" project entails a comprehensive approach to enable remote monitoring of environmental conditions through a user-friendly web interface. It capitalizes on the robust features of the ESP32 microcontroller, renowned for its versatility in IoT applications. Central to the project is the BME280 sensor, renowned for its precision in measuring temperature and humidity. This sensor, combined with jumper wires, forms the hardware backbone for data acquisition.

The software aspect of the project is equally crucial. The Arduino IDE serves as the primary development environment, offering a familiar platform for coding the ESP32. The incorporation of libraries such as the BME280 Library and ESPAsyncWebServer Library streamlines communication between the microcontroller and sensor while facilitating the creation of a responsive web server.

The project's workflow spans various stages, each essential for its successful implementation. Hardware setup involves meticulous wiring to ensure reliable communication between the ESP32 and the BME280 sensor. Software setup encompasses library installation and configuration within the Arduino IDE, laying the groundwork for subsequent programming tasks.

Programming the microcontroller entails initializing the sensor, configuring the web server, and defining routes to handle HTTP requests effectively. This involves implementing logic to retrieve sensor readings and updating them dynamically on the web page.

Designing the web page is an iterative process focused on creating an intuitive user interface. HTML, CSS, and JavaScript are employed to structure the page layout and incorporate dynamic elements for real-time data visualization.

**CHAPTER – 2**

**OBJECTIVE OF THE PROJECT**

The primary objective of the “ESP32 Sensor Readings in Values – Web Server” project is to develop a comprehensive solution for remotely monitoring temperature and humidity sensor readings. By leveraging the capabilities of the ESP32 microcontroller, the project aims to establish a web server that hosts a user-friendly interface accessible from any device with internet connectivity. This interface will dynamically display real-time sensor data, providing users with immediate insights into environmental conditions without the need for physical presence near the sensors. Through seamless integration of hardware components, such as the BME280 sensor, and software development tools like Arduino IDE, the project endeavors to create a versatile system capable of catering to a wide range of IoT applications requiring remote monitoring of environmental parameters.

Furthermore, the project seeks to address the increasing demand for IoT solutions that offer remote accessibility and real-time data visualization. By harnessing the power of asynchronous web server libraries and utilizing HTML for web page design, the project aims to deliver a responsive and interactive user experience. This includes implementing features such as automatic data updates at regular intervals, ensuring that users have access to the most up-to-date sensor readings. Ultimately, the project strives to contribute to the advancement of IoT technology by providing a practical and efficient means of monitoring temperature and humidity sensor readings remotely, thereby facilitating smarter decision-making and enhancing convenience in various domains.

**CHAPTER - 3**

**OUTCOME OF THE PROJECT**

The outcome of the "ESP32 Sensor Readings in Values - Web Server" project is a functional and user-friendly system for remotely monitoring temperature and humidity sensor readings in real-time. Through meticulous hardware setup and software development, the project successfully integrates the ESP32 microcontroller with the BME280 sensor, facilitating accurate data acquisition. The implementation of a web server using asynchronous web server libraries allows users to access sensor readings conveniently from any web-enabled device, enabling remote monitoring without physical proximity to the sensors. The web interface, designed using HTML and enhanced with dynamic updates via JavaScript or AJAX, offers an intuitive and responsive platform for users to visualize and interpret sensor data effectively.

Moreover, the project demonstrates the seamless integration of hardware and software components in IoT applications, highlighting the versatility of the ESP32 microcontroller in hosting web servers and interfacing with sensors. By providing a practical solution for remote monitoring of environmental parameters, the project contributes to advancements in IoT technology, enabling applications in areas such as smart homes, environmental monitoring systems, and industrial automation. The project's outcome not only enhances accessibility to sensor data but also empowers users with valuable insights into environmental conditions, fostering informed decision-making and improved efficiency in various domains.

**CHAPTER - 4**

**INTRODUCTION TO PROJECT**

The "ESP32 Sensor Readings in Values - Web Server" project introduces an innovative approach to remotely monitoring environmental conditions using IoT technology. With a focus on real-time temperature and humidity readings, the project harnesses the capabilities of the ESP32 microcontroller to host a web server accessible from any internet-enabled device. This introduction signifies a pivotal step towards enhancing accessibility and convenience in monitoring environmental parameters, offering users the flexibility to stay informed about their surroundings without physical proximity to the sensors.

By integrating hardware components such as the BME280 sensor and employing software tools like Arduino IDE and web server libraries, the project aims to create a seamless interface between the physical world and digital platforms. The introduction of this project underscores the growing importance of IoT solutions in modern-day applications, from home automation to industrial monitoring systems. Through its innovative approach and practical implementation, the project sets the stage for leveraging IoT technology to address real-world challenges and pave the way for smarter, more connected environments.

**CHAPTER - 5**

**WHY YOU CHOOSE THIS PROJECT**

**1. Practical Application:** This project offers a practical application of IoT technology by enabling remote monitoring of environmental parameters like temperature and humidity. Such monitoring can be beneficial in various scenarios, including home automation, greenhouse management, and industrial settings.

**2. Learning Opportunity:** Developing this project provides an excellent learning opportunity, especially for those interested in IoT, embedded systems, and web development. It involves hardware interfacing, software programming, and web server configuration, allowing enthusiasts to gain valuable skills in multiple domains.

**3. Remote Accessibility:** By creating a web server to display sensor readings, this project enables users to access data remotely from anywhere with an internet connection. This feature adds convenience and flexibility, allowing users to monitor environmental conditions without physically being present at the sensor location.

**4. Integration of Hardware and Software:** This project involves integrating hardware components like sensors with software development tools like Arduino IDE and web server libraries. It offers a holistic learning experience by demonstrating the seamless integration of hardware and software in IoT applications.

Overall, choosing the "ESP32 Sensor Readings in Values - Web Server" project provides a combination of practical application, learning opportunities, and exploration of versatile technologies, making it an appealing choice for enthusiasts and professionals alike.

**CHAPTER - 6**

**WHAT YOU HOPE TO LEARN**

**1. IoT Fundamentals:** By working on this project, individuals can deepen their understanding of Internet of Things (IoT) concepts, including sensor integration, data communication, and remote monitoring.

**2. Hardware Interfacing:** This project involves connecting sensors like the BME280 to the ESP32 microcontroller. Through this process, individuals can learn about hardware interfacing techniques, such as pin connections, sensor communication protocols (e.g., I2C), and sensor data acquisition.

**3. Software Development:** Developing the code for the ESP32 to read sensor data and host a web server offers opportunities to improve programming skills, particularly in languages like C/C++ (for Arduino sketches) and JavaScript (for web development).

**4. Troubleshooting and Debugging:** Like any project, individuals may encounter challenges and bugs along the way. By troubleshooting and debugging issues that arise during the project, they can enhance their problem-solving skills and gain experience in resolving technical issues.

**5. Project Management:** Planning and executing a project from start to finish involves various stages, including research, design, implementation, testing, and documentation. Through this process, individuals can improve their project management skills and learn to effectively organize and execute complex tasks.

Overall, working on the "ESP32 Sensor Readings in Values - Web Server" project offers a rich learning experience encompassing hardware, software, and project management skills, which can be valuable for both personal growth and professional development.

**CHAPTER – 7**

**TOOLS AND TECHNOLOGIES USED**

The "ESP32 Sensor Readings in Values - Web Server" project amalgamates an array of tools and technologies to deliver a comprehensive solution for remote environmental monitoring. At its core lies the ESP32 or ESP8266 microcontroller, serving as the backbone for hosting the web server and facilitating communication with the BME280 sensor. This hardware setup is complemented by the versatile Arduino IDE, which provides a user-friendly platform for programming the microcontroller and integrating sensor data retrieval functionalities. Leveraging the BME280 library ensures seamless interfacing between the ESP32/ESP8266 and the sensor, enabling precise acquisition of temperature and humidity readings.

On the software front, the project harnesses the power of the ESPAsyncWebServer library, specifically tailored for asynchronous handling of HTTP requests on the ESP32/ESP8266 platform. This library forms the cornerstone of the web server functionality, allowing for simultaneous client connections and dynamic content delivery. Additionally, the ESPAsyncTCP library plays a pivotal role in optimizing TCP connections, enhancing the overall efficiency of data transfer between the microcontroller and connected clients. Moreover, the project integrates web technologies such as HTML, CSS, and JavaScript to design an intuitive user interface for the web server, enabling real-time visualization of sensor readings and ensuring a seamless user experience.

To complement the software stack, the project leverages SPIFFS (ESP8266) or LittleFS (ESP32) file systems, enabling storage of HTML web pages and static assets directly on the microcontroller's flash memory. This ensures efficient serving of web content without the need for external hosting, enhancing the project's portability and accessibility. By amalgamating these tools and technologies, the "ESP32 Sensor Readings in Values - Web Server" project empowers users with a robust and user-friendly platform for remotely monitoring temperature and humidity sensor readings, facilitating applications across diverse domains such as smart homes, industrial automation, and environmental monitoring.

**CHAPTER - 8**

**STEP-BY-STEP APPROACH**

**Gather Components:** Collect all necessary components including the ESP32 or ESP8266 microcontroller, BME280 sensor, and jumper wires.

**Set Up Development Environment:**

* Download and install the Arduino IDE.
* Install required libraries: BME280, ESPAsyncWebServer, and ESPAsyncTCP.

**Hardware Setup:**

* Connect the BME280 sensor to the ESP32/ESP8266 using jumper wires according to the pinout specifications.

**Code Initialization:**

* Open the Arduino IDE and start a new sketch.
* Include necessary libraries at the beginning of the sketch.

**Initialize WiFi Connection:**

* Set up WiFi credentials to enable the ESP32/ESP8266 to connect to a local network.

**Initialize BME280 Sensor:**

* Configure the BME280 sensor and initialize communication with the ESP32/ESP8266.

**Set Up Web Server:**

* Initialize the ESPAsyncWebServer instance.
* Define routes for handling HTTP requests, including serving the main web page and handling AJAX requests for sensor data.

**Create HTML Web Page:**

* Design an HTML web page to display temperature and humidity readings.
* Utilize JavaScript or AJAX to dynamically update sensor readings without page refresh.

**Store HTML File on ESP:**

* Upload the HTML file to the ESP32/ESP8266's filesystem using SPIFFS (ESP8266) or LittleFS (ESP32).

**Read Sensor Data:**

* Implement functions to read temperature and humidity values from the BME280 sensor.

**Update Web Page:**

* Set up a timer or delay to periodically update sensor readings and send them to the web page.

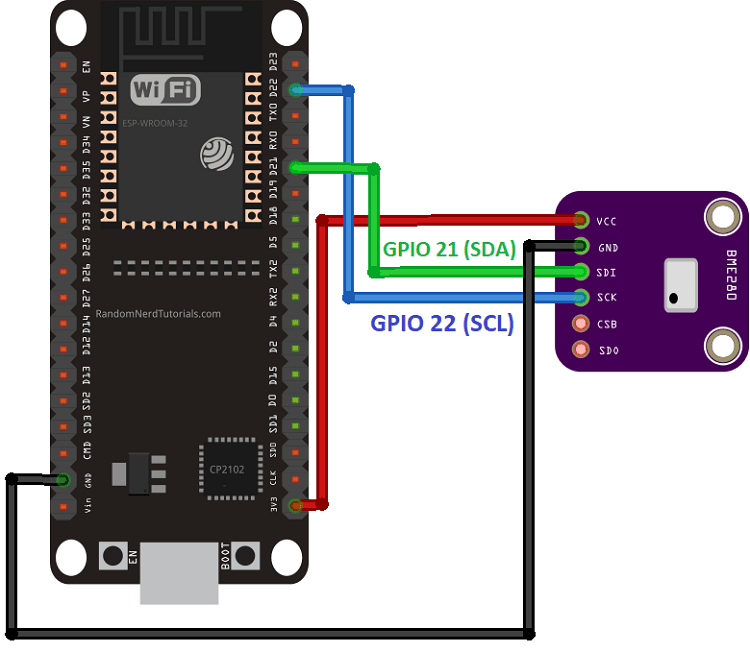
**Test and Debug:**

* Upload the sketch to the ESP32/ESP8266.
* Power up the device and connect to its WiFi network.
* Access the web server from a web browser to verify real-time sensor readings.
* Debug any issues encountered during testing.

**Optimize and Refine:**

* Fine-tune the code and web interface for improved performance and usability.
* Implement additional features or functionalities as needed.

**CIRCUIT DIAGRAM**



**Fig- 8.1 – Circuit Diagram**

**CHAPTER - 9**

**DESCRIPTION OF WHAT YOU BUILT**

The "ESP32 Sensor Readings in Values - Web Server" project is a compact yet robust system enabling remote monitoring of temperature and humidity sensor readings in real-time. Leveraging the ESP32 microcontroller and the BME280 sensor, the project hosts a web server accessible from any internet-enabled device. The user-friendly web interface, developed using HTML, CSS, and JavaScript, dynamically displays sensor data, offering immediate insights into environmental conditions. With the ESPAsyncWebServer library facilitating multiple client connections and SPIFFS or LittleFS file systems enabling storage of HTML files, the project represents an efficient fusion of hardware and software technologies. Through meticulous setup and configuration, the project provides a practical solution for remote environmental monitoring, showcasing the versatility and potential of IoT applications in creating smarter, interconnected environments.

**SOURCE CODE:**

#ifdef ESP32

#include <WiFi.h>

#include <ESPAsyncWebServer.h>

#include <SPIFFS.h>

#include "DHT.h"

#else

#include <Arduino.h>

#include <ESP8266WiFi.h>

#include <Hash.h>

#include <ESPAsyncTCP.h>

#include <ESPAsyncWebServer.h>

#include "FS.h"

#endif

// Replace with your network credentials

const char\* ssid = "AIML";

const char\* password = "987654321";

#define DHTPIN 27

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

String readDHTTemperature() {

// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)

// Read temperature as Celsius (the default)

float t = dht.readTemperature();

// Read temperature as Fahrenheit (isFahrenheit = true)

//float t = dht.readTemperature(true);

// Check if any reads failed and exit early (to try again).

if (isnan(t)) {

Serial.println("Failed to read from DHT sensor!");

return "--";

}

else {

Serial.println(t);

return String(t);

}

}

String readDHTHumidity() {

// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)

float h = dht.readHumidity();

if (isnan(h)) {

Serial.println("Failed to read from DHT sensor!");

return "--";

}

else {

Serial.println(h);

return String(h);

}

}

// Create AsyncWebServer object on port 80

AsyncWebServer server(80);

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>ALTS Weather Station</h2>

<img src="cloud">

<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";

// Replaces placeholder with DHT values

String processor(const String& var){

//Serial.println(var);

if(var == "TEMPERATURE"){

return readDHTTemperature();

}

else if(var == "HUMIDITY"){

return readDHTHumidity();

}

return String();

}

void setup(){

// Serial port for debugging purposes

Serial.begin(115200);

// Connect to Wi-Fi

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.println("Connecting to WiFi..");

}

dht.begin();

if(!SPIFFS.begin()){

Serial.println("An Error has occurred while mounting SPIFFS");

return;

}

// Print ESP32 Local IP Address

Serial.println(WiFi.localIP());

// Route for root / web page

server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/html", index\_html);

});

/\*server.on("/sun", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/sun.png", "image/png");

});

server.on("/sun-cloud", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/sun-cloud.png", "image/png");

});

server.on("/cloud", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/cloud.png", "image/png");

});

server.on("/rain", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/rain.png", "image/png");

});

server.on("/storm", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/storm.png", "image/png");

});

server.on("/snow", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/snow.png", "image/png");

});\*/

server.on("/cloud", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(SPIFFS, "/cloud.png", "image/png");

});

server.on("/temperature", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(200, "text/plain", readDHTTemperature().c\_str());

});

server.on("/humidity", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send(200, "text/plain", readDHTHumidity().c\_str());

});

// Start server

server.begin();

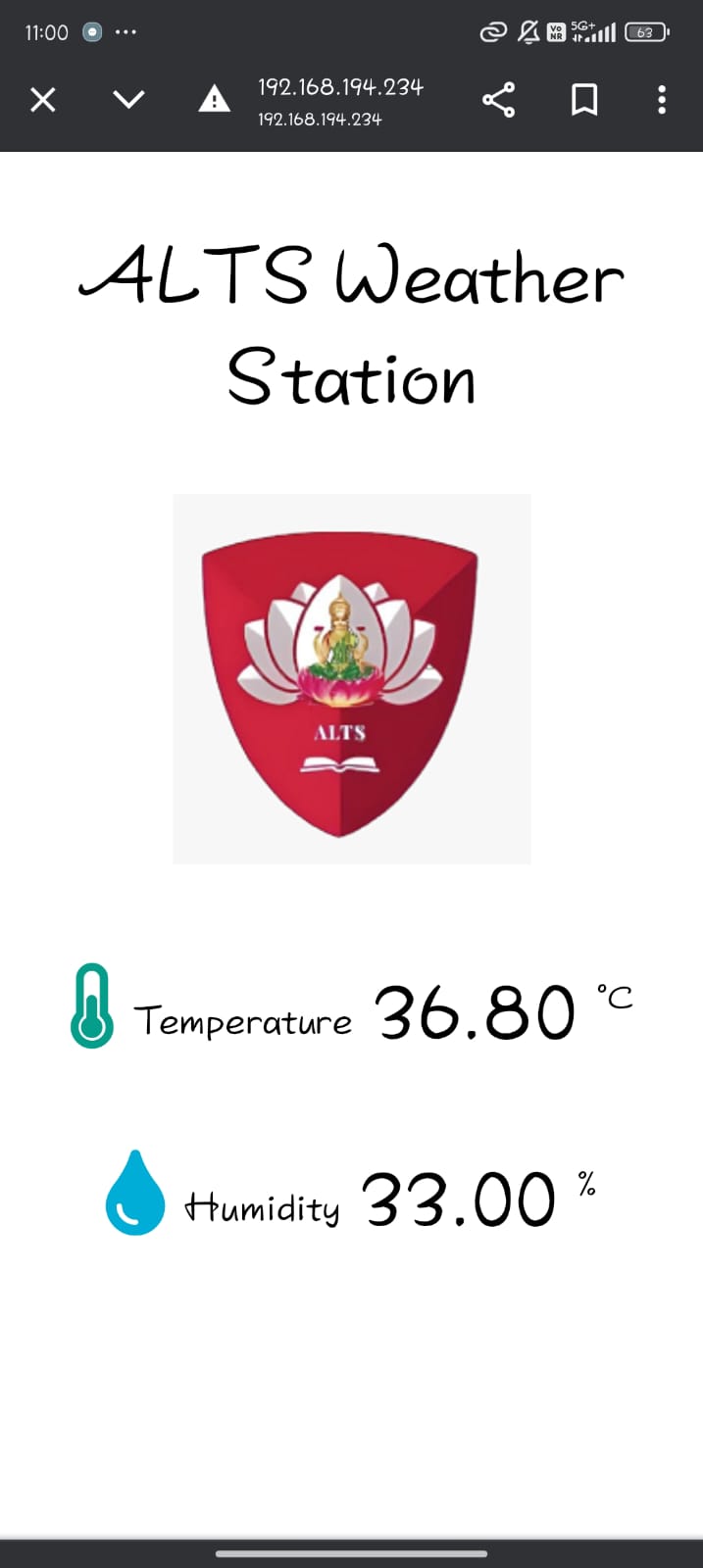
}

void loop(){

}

**CHAPTER – 10**

**SCREENSHOTS OR VISUALS OF OUTPUT/RESULT**



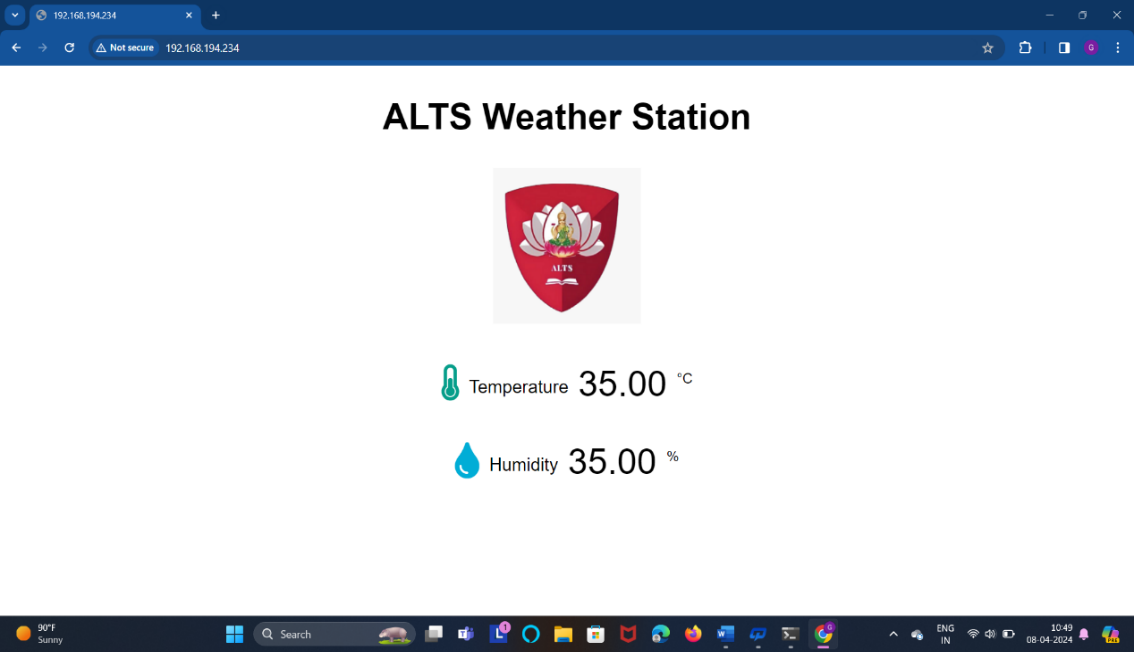


Fig – 10.2 – Screenshot of output in laptop view

Fig – 10.1 – Screenshot of output in mobile view

**CHAPTER - 11**

**NEW SKILLS ACQUIRED**

**IoT Development:**

* Integrating sensors with microcontrollers.
* Creating web server interfaces for remote monitoring.

**Embedded Systems Programming:**

* Sensor interfacing and data processing.
* Web server configuration on platforms like ESP32/ESP8266.

**Web Server Development:**

* Handling HTTP requests and serving web pages.
* Implementing dynamic content updates using asynchronous programming.

**Sensor Interfacing:**

* Understanding sensor datasheets and configuring communication protocols.
* Retrieving sensor data for further processing.

**File Systems Management:**

* Utilizing SPIFFS (ESP8266) or LittleFS (ESP32) to store and serve web content.
* Managing file systems on microcontroller flash memory.

**Client-Server Communication:**

* Understanding principles of HTTP communication.
* Sending and receiving data between web clients and the microcontroller.

**Web Development:**

* Designing user interfaces using HTML, CSS, and JavaScript.
* Implementing real-time data visualization on web pages.

**Troubleshooting and Debugging:**

* Identifying and resolving hardware and software issues.
* Debugging network configuration problems.

**CHAPTER - 12**

**INSIGHTS GAINED FROM THE PROJECT**

Implementing the "ESP32 Sensor Readings in Values - Web Server" project yields a plethora of insights across various domains:

* **IoT Architecture**: Understanding the architecture and components required for IoT applications offers insights into the structure and functionality of IoT systems.
* **Hardware-Software Integration**: Integrating hardware components with software development tools like the Arduino IDE provides insights into how hardware and software interact, applicable to embedded systems and IoT development.
* **Web Server Configuration:** Configuring a web server on a microcontroller entails understanding web protocols, server-side programming, and client-server communication, providing insights into web server technologies.
* **Real-Time Data Visualization:** Implementing real-time updates of sensor data on a web page deepens understanding of JavaScript and asynchronous programming techniques for dynamic content rendering.
* **Remote Monitoring Applications:** Exploring remote monitoring applications in environmental sensing informs decision-making processes in various domains such as agriculture, weather forecasting, and smart home automation.
* **Troubleshooting and Debugging:** Addressing challenges and debugging issues encountered during project implementation enhances problem-solving skills applicable to hardware connections, software errors, and network issues.

**CHAPTER - 13**

**CONCLUSION**

* 1. **- SUMMARY OF THE PROJECT**

The "ESP32 Sensor Readings in Values - Web Server" project is a comprehensive endeavor aimed at creating a robust system for remotely monitoring temperature and humidity sensor readings in real-time. By harnessing the capabilities of the ESP32 microcontroller and integrating it with the versatile BME280 sensor, the project facilitates seamless data acquisition and communication. The implementation of an asynchronous web server, coupled with HTML, CSS, and JavaScript, enables the creation of a user-friendly web interface for accessing sensor data from any internet-enabled device. Through meticulous hardware setup, software development, and web server configuration, the project offers insights into IoT architecture, hardware-software integration, web server technologies, and real-time data visualization techniques. Furthermore, troubleshooting and debugging challenges encountered during project implementation provide valuable learning opportunities, enhancing problem-solving skills and fostering a deeper understanding of embedded systems development. Overall, the "ESP32 Sensor Readings in Values - Web Server" project exemplifies the intersection of hardware and software technologies in creating practical solutions for remote environmental monitoring, with implications for various domains including home automation, industrial monitoring, and environmental sensing.

* 1. **- KEY TAKEAWAYS**

The "ESP32 Sensor Readings in Values - Web Server" project encapsulates several key takeaways essential for IoT and embedded systems development. Through this endeavor, understanding the integration of hardware components like sensors with microcontrollers and web servers becomes paramount. The project demonstrates the practical implementation of real-time monitoring, showcasing the creation of a web server capable of providing up-to-date sensor data remotely accessible from any device with internet connectivity. In parallel, the development of the web interface underscores the importance of web development skills, as participants navigate the intricacies of HTML, CSS, and JavaScript to craft an intuitive user interface with dynamic data updates. Challenges encountered during implementation provide invaluable opportunities for troubleshooting and debugging, refining problem-solving abilities in areas such as hardware connections, software errors, and network configuration. Moreover, effective documentation and communication practices foster collaboration and knowledge sharing within the community, enriching both personal and collective learning experiences. Leveraging the versatility of the ESP32 microcontroller, this project not only showcases its capabilities in hosting web servers and interfacing with sensors but also lays the groundwork for exploring a myriad of remote monitoring applications across diverse domains. Ultimately, the project serves as a springboard for continued learning and innovation in the burgeoning field of IoT, empowering individuals to develop practical solutions for a connected world.

* 1. **- Next Steps**

After completing the "ESP32 Sensor Readings in Values - Web Server" project, several potential next steps can be considered to further enhance its functionality or explore new avenues:

**1. Expand Sensor Support:** Incorporate additional sensors such as light sensors, gas sensors, or motion sensors to monitor more environmental parameters

**2. Implement Data Logging:** Integrate data logging capabilities to store sensor readings over time, enabling historical analysis and trend identification.

**3. Enhance User Interface:** Improve the web interface by adding features such as data visualization graphs, customizable settings, or user authentication for secure access.

**4. Implement Alerts and Notifications:** Introduce alerts and notifications functionality to notify users via email or SMS when sensor readings exceed predefined thresholds.

**5. Integrate IoT Platforms:** Explore integration with IoT platforms like AWS IoT, Google Cloud IoT, or Azure IoT Hub for advanced data analytics, remote management, and scalability.

**6. Explore Machine Learning Integration:** Investigate the integration of machine learning algorithms for predictive analysis or anomaly detection based on historical sensor data.

**7. Develop Mobile Applications:** Create companion mobile applications to complement the web interface, providing users with additional flexibility and convenience in accessing sensor data.