School Of Computing,
UUM College Of Arts and Sciences

SKIH3113 - Sensor-Based Systems
Semester A232

Final Project:
Plant Monitoring System (PMS)

Prepared by:

_____

MUHAMMAD SYAFIQ BIN AZHARI (289869)

Submission Date: 10th July 2024

Lecturer's Name: MR. AHMAD HANIS BIN MOHD SHABLI

1.0 Introduction

In the current landscape of plant monitoring systems, traditional approaches often rely heavily on manual observation and intervention, lacking the automation and real-time data analysis required to optimise plant care. These conventional systems are typically limited in their functionality, providing basic monitoring without integrating advanced technologies to automate responses based on environmental conditions.

The Plant Monitoring System (PMS) aims to revolutionise this process by leveraging a combination of sensors, microcontrollers, and a web-based interface to provide comprehensive and automated plant care. The system is designed to continuously measure critical parameters such as temperature, humidity, light intensity, and water levels. When any of these conditions deviate from their optimal ranges, the system notifies the user and can even activate automated responses such as cooling fans or watering mechanisms.

This advanced level of monitoring and automation is crucial for maintaining the health and productivity of plants, particularly in environments where consistent and optimal conditions are vital. The PMS addresses the limitations of traditional systems by offering a solution that not only monitors but actively manages the plant's environment, ensuring better growth and reducing the need for constant human intervention.

Objectives

a) To develop a plant monitoring system that accurately measures temperature, humidity, and light intensity as well as tank water level.
b) To develop a plant monitoring system that shall notify the user of when the plant's environment is not in an optimal state.
c) To develop a web-based interface to display data and information on the plant’s environment.

2.0 System Design And Architecture

The Plant Monitoring System (PMS) is built upon a layered architecture, comprising various hardware and software components that work together seamlessly to ensure efficient and reliable plant care. At the core of the system are the sensors, which include temperature, humidity, light intensity, and soil moisture sensors. These sensors are strategically placed in the plant environment to gather real-time data, which is crucial for monitoring the health and growth conditions of the plants. The data collected by these sensors is then transmitted to a microcontroller, which serves as the primary processing unit in the system.

The microcontroller, equipped with necessary processing capabilities, performs initial data processing and validation. It checks whether the collected data falls within the predefined optimal ranges. If any parameter is found to be outside the optimal range, the microcontroller can initiate corrective actions, such as activating water pumps or cooling fans. The processed data is then sent to a central server for further analysis and long-term storage. The server acts as the main data repository and is responsible for handling data from multiple sensors and microcontrollers, ensuring scalability and robustness of the system.

The server is connected to a web-based interface, which serves as the user interaction layer of the system. This interface allows users to access real-time data, historical records, and receive notifications about the status of their plants. The web interface is designed to be user-friendly and accessible from various devices, including smartphones, tablets, and computers. Users can log in to the system, view detailed reports, and configure system settings according to their preferences. The integration of these components—sensors, microcontroller, server, database, and web interface—ensures a comprehensive, automated, and efficient plant monitoring system that significantly reduces the need for manual intervention while optimising plant care.

2.1 Block Diagram

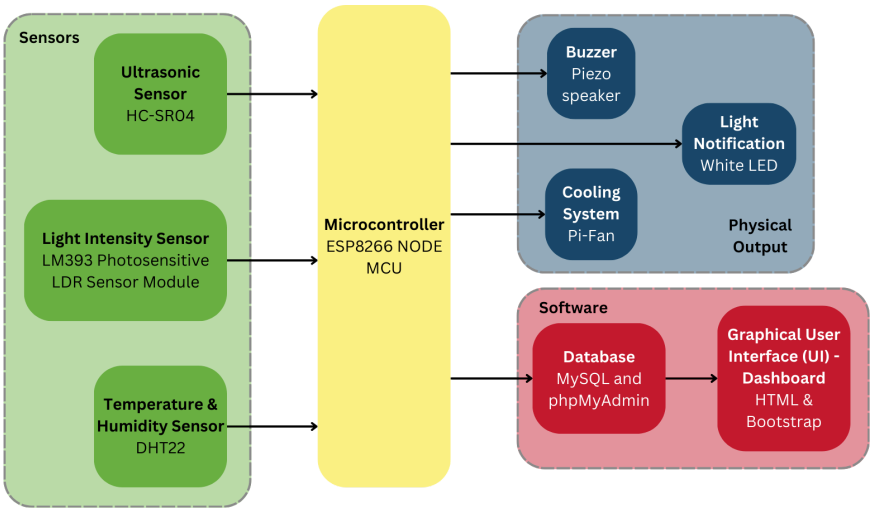


Figure 2.1: Block Diagram of PMS

2.2 Hardware Description

2.2.1 Sensors

a) HC-SR04 - Ultrasonic Sensor

Figure 2.2.1.1: HC-SR04 Sensor

- The HC-SR04 sensor measures the distance of objects by sending out a high-frequency sound wave that would be bounced off by the objects in front of it. The idea is that the sensor itself will transmit and then receive the reflected waves to then calculate the distance by measuring the time that has elapsed. In this project, the sensor will be used to measure the level of a water tank that is used to water the plant.

b) LM393 Photosensitive LDR Sensor Module - Light Intensity Sensor

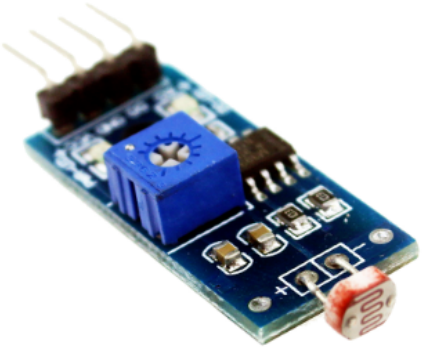


Figure 2.2.1.2: LM393 Photosensitive LDR Sensor Module

- The Photosensitive LDR sensor measures the light intensity of the surrounding environment. It does this by absorbing energy via the LDR's photoconductive material and changing the resistance of the component. In this project, the role of the LDR sensor is to determine the amount of light present in the environment to monitor the current light level to ensure the user is updated when the light level is not in an optimal state.

c) DHT22 - Temperature and Humidity Sensor

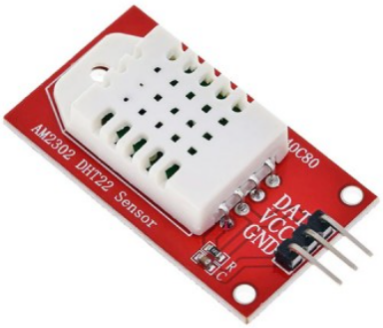


Figure 2.2.1.3: DHT22 Sensor

- The DHT22 sensor is really two sensors in one. It can measure both temperature, and humidity near-simultaneously with a single data pin. The humidity is measured by a capacitive humidity sensor that detects changes in humidity by measuring the dielectric constant of a humidity-sensitive material. As humidity increases, the capacitance changes, allowing the sensor to determine relative humidity (0% to 100%). As for the temperature, a thermistor measures the ambient temperature and produces data in celsius (°C). The role of this sensor is to supply data on both humidity and temperature to ensure that the ambient temperature and humidity is optimal for the plant.

2.2.2 Microcontroller - NodeMCU V2 ESP8266

Figure 2.2.2: NodeMCU V2 ESP8266

a) Sensor Data Acquisition

- The connected sensors will be collected by the microcontrollers via several pinouts on the board. The data that are sent are then taken as a whole (if they are analog inputs) or taken through a library such as data from the DHT22 sensor as it requires a library to read the data collected by it.

b) ESP8266 as a Client

- Once data has been collected, the ESP8266 will act as a client that will communicate with the server which holds a database. In this project, the

database will be a locally-hosted database. The communication will be done via a shared WiFi connection with the server and thus being able to connect through IP address.

c) HTTP Request

- Once a connection is made, the ESP8266 board will make a HTTP POST request to the server via a PHP script and then store the collected sensor data into the database.

2.2.3 Physical Output

a) Buzzer - Piezo Speaker

Figure 2.2.3.1: Piezo Speaker

- The piezo speaker will act as a buzzer by turning on when sub-optimal conditions are met such as temperature is too high or low, or the water tank is less than half full. The high pitched sound will give a sense of urgency to the user to try and fix the problem at hand.

b) Light Notification - White LED

Figure 2.2.3.2: White LED

- The white LED will notify the user when the current plant ambient conditions are sub-optimal. Although the speaker will already notify the user, the LED will act as a backup to be more inclusive to deaf or hard-hearing users.

c) Cooling System - Pi-Fan

Figure 2.2.3.3: Pi-Fan

- The Pi-Fan will turn on when the temperature gets too hot in order to cool down the room that the plant is situated in.

2.2.4 Software

a) Database - MySQL and phpMyAdmin

- As for the data storage, a relational database has been selected to be utilised in this project. The type of relational database was MySQL which was installed alongside phpMyAdmin via XAMPP. This combination allows the ease of data storage as the GUI is simple and is able to hold many records over time.

b) Graphical User Interface (GUI) - HTML and Bootstrap

- A GUI has been developed in the shape of an information dashboard that is equipped with a 5-second interval monitoring system. This dashboard is developed by using HTML, Javascript, PHP, and a CSS framework, Bootstrap. The dashboard allows the user to easily understand the data that has been collected as the dashboard comes equipped with prompts, alerts, and graphs.

2.3 Fritzing

2.3.1 Breadboard View

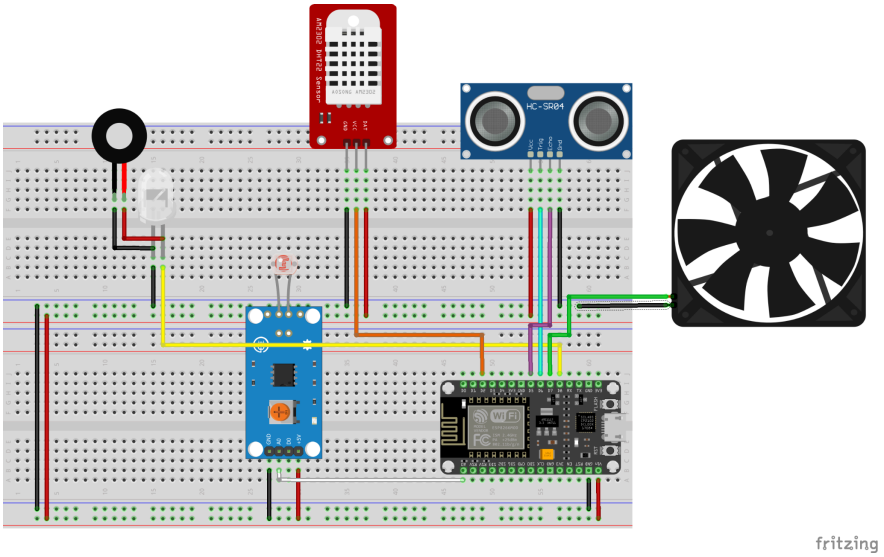


Figure 2.3.1: Fritzing Diagram of PMS Module

2.3.2 Data Flow Diagram

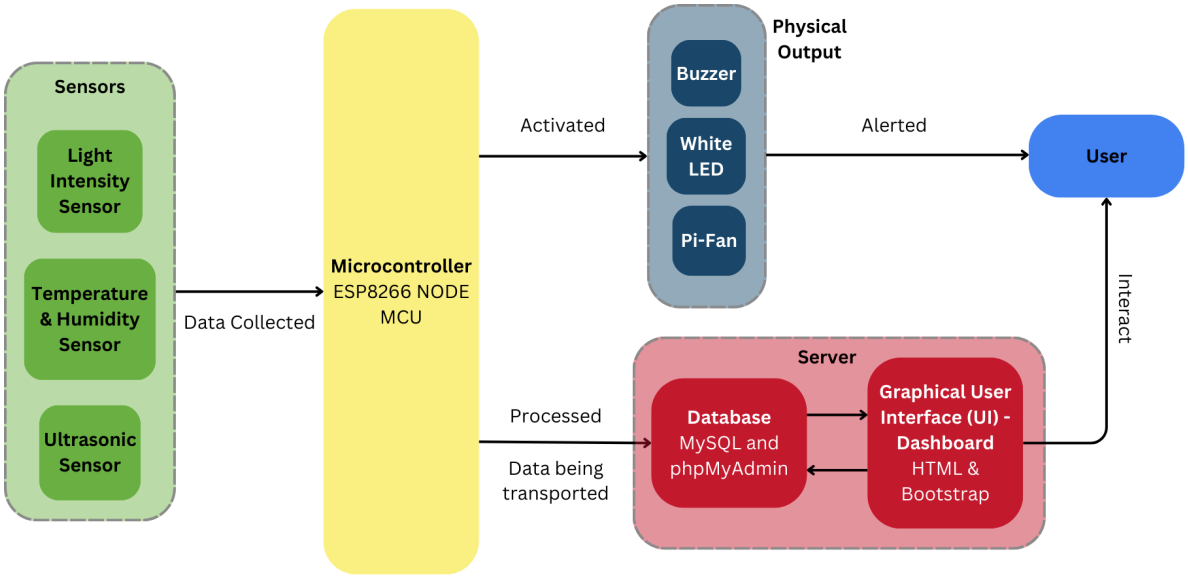


Figure 2.3.2: Data Flow Diagram of PMS

3.0 Project Implementation

3.1 Hardware Setup

As of the hardware itself, specific connections must be made correctly to ensure that the circuit and components work as intended.

a) Sensor

i) Connection of ESP8266 to DHT22

DHT22	ESP8266
VCC	VIN
GND	GND
DAT	D2

ii) Connection of ESP8266 to HC-SR04

HC-SR04	ESP8266
VCC	VIN
TRIG	D6
ECHO	D5
GND	GND

iii) Connection of ESP8266 to LDR Module

LDR Module	ESP8266
VCC	VIN
GND	GND
DO	-
AO	A0

b) Physical Outputs

i) Connection of ESP8266 to Buzzer

Buzzer	ESP8266
Positive terminal (+)	D8

Negative Terminal (-)	GND

ii) Connection of ESP8266 to White LED

White LED	ESP8266
Positive Terminal (+)	D8
Negative Terminal (-)	GND

iii) Connection of ESP8266 to Pi-Fan

Pi-Fan	ESP8266
Positive Terminal (+)	D7
Negative Terminal (-)	GND

c) Microcontroller - ESP8266
- Ensure the three sensors are correctly connected to the correct pinouts.
- Configure and code to allow the ESP8266 board to connect to a Wi-Fi network to connect to the localhost database.
- Connect the ESP8266 board to a power source via a micro-usb to usb type-A data cable.

3.2 Software Development

3.2.1 Data Storage

In terms of storing the data collected by the microcontroller, the project uses a relational database (MySQL) to store all of the data. Using phpMyAdmin, the handling of the database was much easier as a simple GUI is used to perform queries, edits and even other more complex operations on the database itself. The database itself has 6 different columns which are: *id, timestamp, sensor1, sensor2, sensor3,* and *sensor4.* The actual data that is stored in this database are temperature (sensor1), humidity (sensor2), water level (sensor3), and light intensity (sensor 4). To send data to the database, the ESP8266 will send a HTTP POST request via Wi-Fi containing the relevant sensor data to a PHP file (test_data.php) to be sent to the database in a local server.

```
CREATE DATABASE `midsem`
```

Note: Create database 'midsem' with SQL query

```
CREATE TABLE `sensor` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
    `timestampp` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
  `sensor1` float NOT NULL,
  `sensor2` float NOT NULL,
  `sensor3` float NOT NULL,
  `sensor4` float NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=51 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Note: Create table 'sensor' using SQL query

id	timestampp	sensor1	sensor2	sensor3	sensor4
1	2024-07-10 11:48:55	26	62	0.15	6.87
2	2024-07-10 11:49:00	26	63	0.07	7.21
3	2024-07-10 11:49:05	26	62	0.1	6.92
4	2024-07-10 11:49:10	26	63	0.13	6.95
5	2024-07-10 11:49:16	26	63	0.13	7.02
6	2024-07-10 11:49:21	26	63	0.06	6.97
7	2024-07-10 11:49:56	26	63	0.15	7.1
8	2024-07-10 11:50:01	26	61	0.05	7.05
9	2024-07-10 11:50:06	26	61	0.13	6.51
10	2024-07-10 11:50:11	26	61	0.05	6.47
11	2024-07-10 11:50:16	26	61	0.05	6.46
12	2024-07-10 11:50:21	26	61	0.08	6.88
13	2024-07-10 11:50:27	26	61	0.08	6.88

Figure 3.2.1: Database Table Structure

3.2.2 Data Retrieval

A singular PHP script is able to handle retrieving the data from the database into the web application. The file “fetch_data.php” allows the web application/dashboard to retrieve and access the records in the SQL database. The data are then processed and displayed in various formats that are simple and easy to read such as prompts and charts. Furthermore, averages are calculated to give the user a sense of what has happened in the past.

3.2.3 Data Insights

Scripts in both dashboard.php (dashboard) and finalcode.ino handle the data and process the data to generate insights once the data is retrieved from the database. In some cases, the data must first be calculated with the correct formula to generate meaningful insights. For example, the LDR sensor gives analog readings which are from 0 to 1024 which needs to be processed first. In both codes, there are thresholds of the optimal range of values for each sensor. The goal is to inform the user to maintain the optimal environment and notify if there are any deviations from the optimal scenarios.

a) Temperature Readings

Current Reading (℃)	Insight
Current < 21	● Temperature is too cold ● Environment needs to increase in temperature
21 < Current < 37	● Temperature is in the optimal range ● Do not change.
Current > 37	● Temperature is too hot ● Environment needs to be cooled down ● Turn on the fan.

b) Humidity Readings

Current Reading (%)	Insight
Current < 60	● Humidity is too low ● Spray some water into the environment
60 < Current < 80	● Humidity is in the optimal range ● Do not change.
Current > 80	● Humidity too high ● Allow for ventilation

c) Water Level Readings

As for water level readings, there will be an assumption given for this project. The assumption is that the water tank that we will be monitoring is only 15cm/0.15m in depth and about 10,000cm^2/1m^2 in base surface area. This means that the maximum volume would only be 150,000cm^3/0.15m^3/150 litres. Although a very odd shaped tank, this is only theoretical for the purposes of validation.

Current Reading (℃)	Insight
Current < 0.08	● Water level is less than half ● Add water
0.08 < Current < 0.15	● Water level is at an acceptable level ● Do not change.
Current > 0.15	● Water level is more than the capacity ● Overflowing imminent

To calculate for the water level, the formula given below could be used to obtain the depth in cm:

$$distance = (duration * speed\ of\ sound / 2) / 100$$

As the sound wave will be headed towards the water level and then be bounced back, the sound wave that will be picked up by the ultrasonic sensor would have travelled twice the distance of the water level from the sensor. Thus, the time taken multiplied by the speed of sound divided by two would give the distance in millimetre (mm). By dividing that value by 100, we then obtain the data in centimetre (cm).

d) Light Intensity Readings

As for the data obtained from the LDR sensor, the data is in an analog format. The analog format is dependant on the voltage and in this case, the range of values if from 0 - 1024. To process this raw data, a simple equation to convert the value into a 10-point scale of light intensity would be used using the equation below:

$$light\ value = (1024 - analog\ value) / 102.4$$

The value 1024 is subtracted by the analog value due to the analog value increasing when the LDR sensor receives little light. Thus, creating an inverse output that could be confusing for users. Thus, once the values are inverted into a positive relationship between LDR value and light intensity, the value is then divided by 102.4 to produce a range of possible values of 0 to 10.

Current Reading (℃)	Insight
Current < 0.08	• Water level is less than half • Add water
0.08 < Current < 0.15	• Water level is at an acceptable level • Do not change.
Current > 0.15	• Water level is more than the capacity • Overflowing imminent

3.2.4 Output

The web dashboard was designed to be simple and readable with the use of colours and figures while also providing near-real time visualisation of the plant's environment collected by the sensors. The website was created using HTML, Javascript, and Bootstrap (CSS framework) to give it a simple yet readable interface. On the other hand, PHP was used to fetch data from the SQL database in order to display the database information onto the dashboard.

The web-app itself has mainly two pages; a dashboard (Home) and a recent data table (Details) that can be accessed via the top bar.

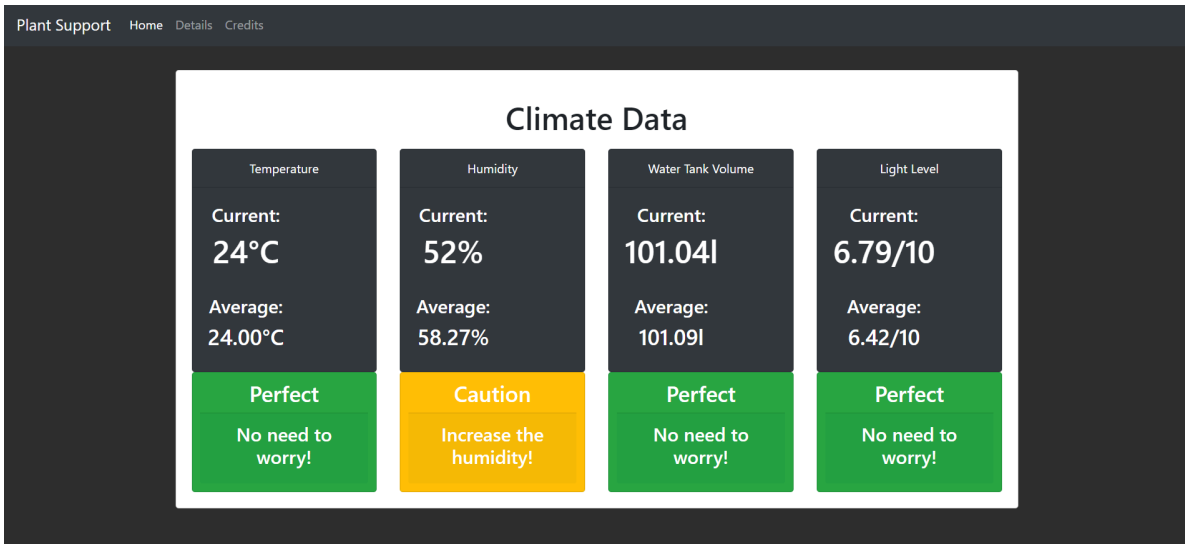


Figure 3.2.4.1: Climate Data on Home Dashboard

The main focus of the dashboard interface is the "Climate Data" section that will tell the user on the current reading, average readings of the past 60 readings, current status, and recommendation of all four of the sensors. As seen in Figure 3.2.4.1, the readings on temperature, water level, and light level are perfect. However, the humidity is too low and is not in the optimal range. Due to that, a simple notification via JavaScript alert() function will be shown as seen in Figure x.x.x.x as well as the buzzer and white LED will turn on as seen

in Figure 3.2.4.2. It is to be noted that if any of the sensors detect a current reading outside of their respective optimal range, the alert which includes the JavaScript alert, buzzer, and LED light will activate notifying the user.

Figure 3.2.4.2: Alert Notification

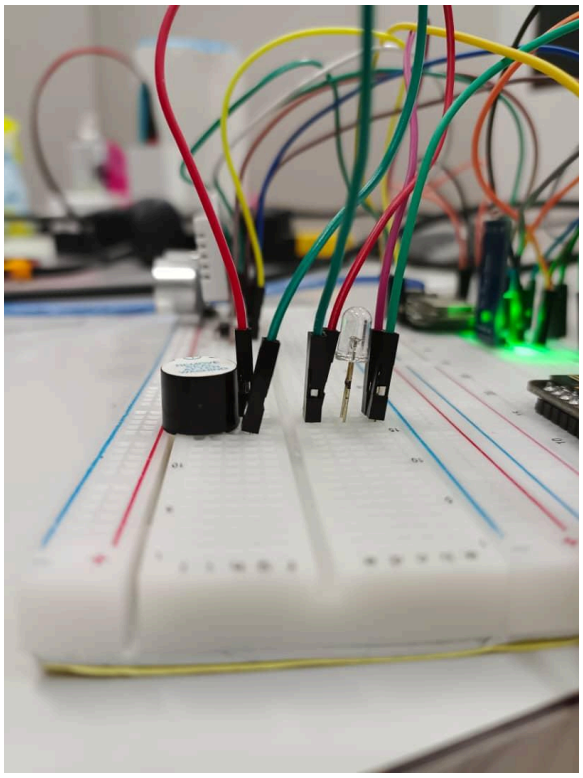


Figure 3.2.4.3: Alert System Off

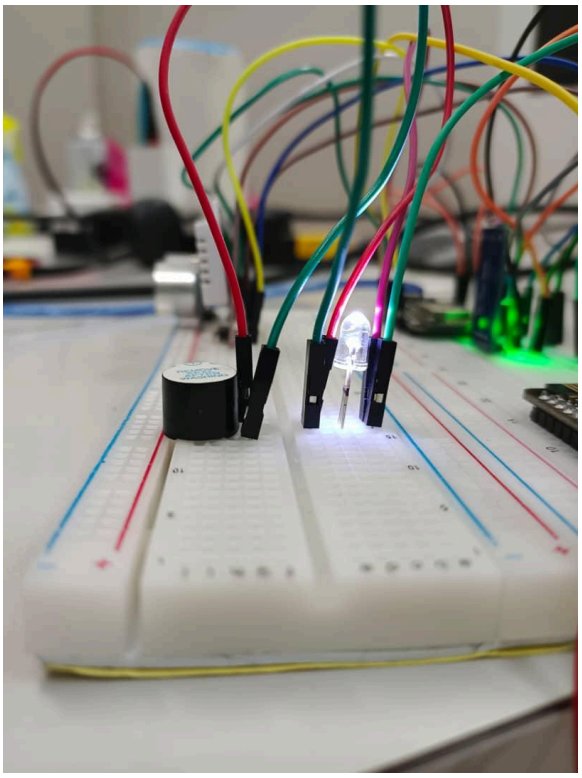


Figure 3.2.4.4: Alert System On

As for the bottom section of the dashboard, there is a Graphs & Charts section that shows the past 60 entries (past 5 minutes) of data onto their own respective graphs as seen in Figure Figure 3.2.4.5. These graphs show the trends and spikes in data which may help the user find problems or odd behaviours in the environment.

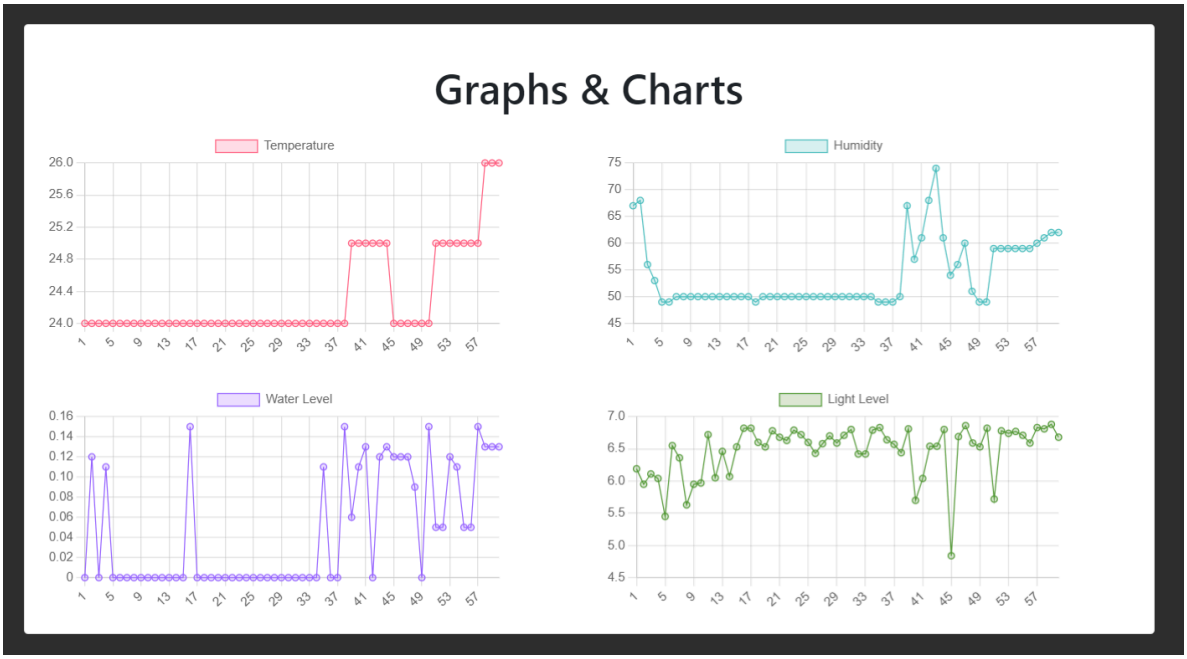
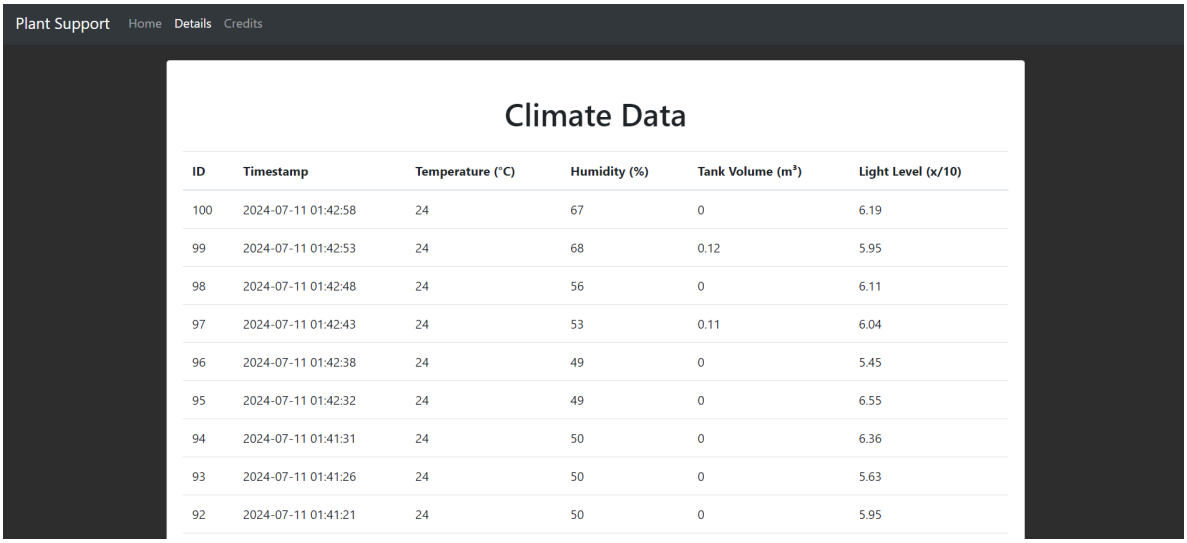


Figure 3.2.4.5: Graphs of last 60 records in Home Dashboard

The Details page shows a table of the last 100 records. This page allows the user to see not just the specific numerical data of each sensor but also the timestamp that it was recorded in. The page can be seen in Figure 3.2.4.6.



ID	Timestamp	Temperature (°C)	Humidity (%)	Tank Volume (m³)	Light Level (x/10)
100	2024-07-11 01:42:58	24	67	0	6.19
99	2024-07-11 01:42:53	24	68	0.12	5.95
98	2024-07-11 01:42:48	24	56	0	6.11
97	2024-07-11 01:42:43	24	53	0.11	6.04
96	2024-07-11 01:42:38	24	49	0	5.45
95	2024-07-11 01:42:32	24	49	0	6.55
94	2024-07-11 01:41:31	24	50	0	6.36
93	2024-07-11 01:41:26	24	50	0	5.63
92	2024-07-11 01:41:21	24	50	0	5.95

Figure 3.2.4.6: Past 100 records of climate data in Details page

Lastly, if the temperature reading is higher than the maximum of the optimal range, the system will turn on the Pi-Fan to attempt to cool down the plant's environment. In combination with the other functionalities, this system allows the user to not only monitor but pinpoint problems over a long period of time.

4.0 Testing And Validation

In using the sensors, prior testing and validation needs to be carried out to ensure that the data collected is correct and reliable. However, due to the limitations in a controlled environment, the DHT22 sensor cannot be properly validated as there is no proper controlled room where both temperature and humidity can be controlled. However with that said, the DHT22 sensor has been teste in other less professional ways to see if it can detect differences in humidity and temperature.

4.1 Ultrasonic Sensor Validation

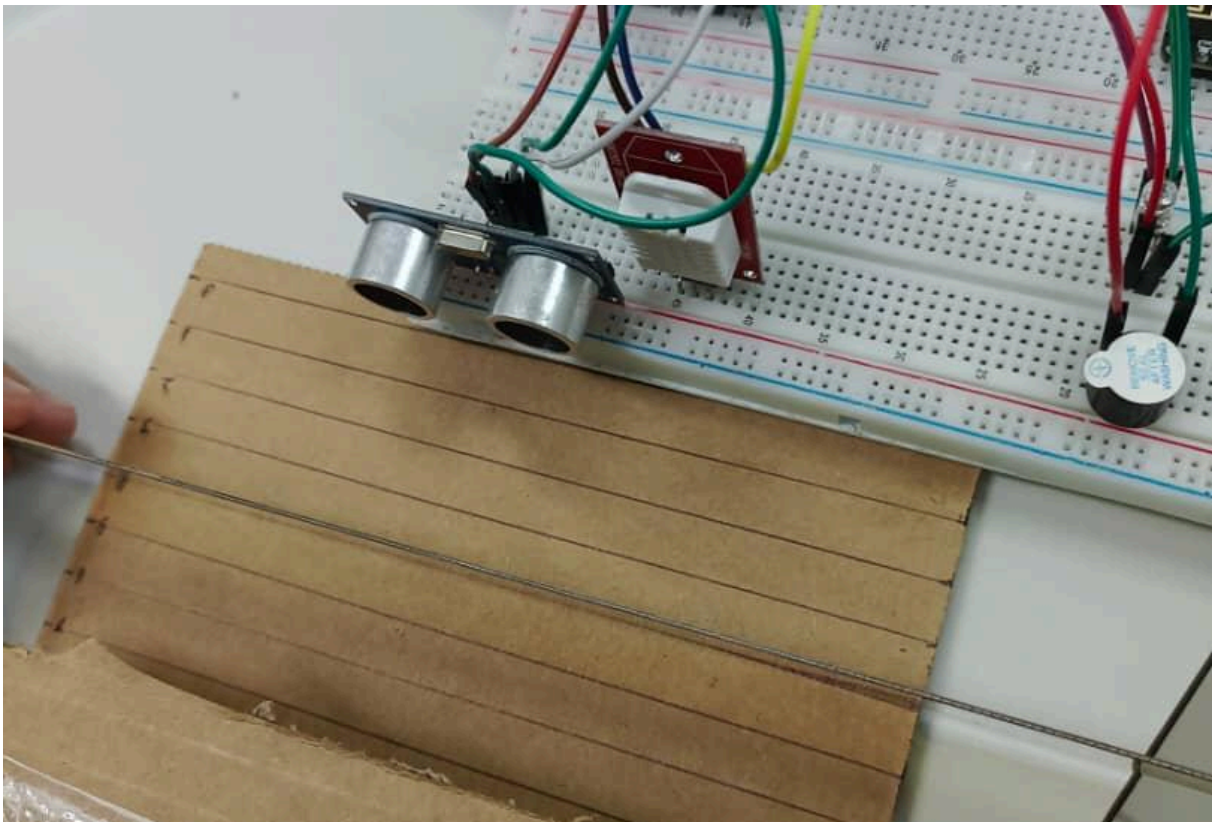


Figure 4.1: Using a ruler and cardboard to validate the HC-SR04 sensor

To validate the ultrasonic sensor, a simple cardboard measuring tool was created to see whether the distance measured would be accurate and stable. The small validation test will be conducted at three different distances; 2cm, 4cm, and 6cm where a total of 5 separate readings will be taken. The results have been documented in Table 4,1.

Iteration	Actual Distance (cm)	Reading(cm)
1	2cm	2
2		2
3		2

4		2
5		2
1	4cm	4
2		4
3		4
4		4
5		3
1	6cm	5
2		6
3		5
4		6
5		5

Table 4.1: Ultrasonic Validation Test Results

From Figure 4,1, the averages of the 2cm, 4cm, and 6cm tests are 2.0cm, 3.8cm, and 5.4cm respectively. This shows some level of reliability at very close distances however becomes more unstable as the l;ength increases.

4.2 DHT22 (Temperature and Humidity) Validation

As mentioned previously, there is no formal way to validate the DHT22 sensor due to the lack of a controllable environment. However it is found that breathing out hot hair towards the DHT22 sensor does increase temperature a little and increases humidity by a lot. This does coincide with human breath which is high in moisture and slightly higher in temperature compared to the surrounding air.

4.3 LDR Sensor Validation

As for the LDR sensor, a simple test would be done where there will be three scenarios. The first is a control test where the LDR will be left out in a lit room, the second is where an object (a thumb) will cover the LDR supposedly resulting in a dark environment, and the last scenario is a bright test where a phone flashlight will be shined upon the LDR resulting in a very bright environment. The test will be conducted over 5 iterations per scenario. The results are seen in Table 4.3.

Iteration	Situation	Reading(cm)
1	Control	6.64

2		6.54
3		5.47
4		6.36
5		6.72
1	Dark Room	0.00
2		0.14
3		0.00
4		0.24
5		0.02
1	Bright Room	9.14
2		9.21
3		9.09
4		9.13
5		9.12

Table 4.3: LDR Validation Test Results

From Figure 4.3, the averages for situation control, dark room, and bright room are 6.346, 0.08, and 9.138 respectively. This shows that the LDR sensor is working as intended and is reading data that is predictable and correct.

4.4 Sensor Calibration Phase

In order to ensure that the sensors do not give false readings upon activating for the first time, the microcontroller will not send data for the first 5 attempts (25 seconds) and send it on the 6th attempt. This is to ensure that the readings taken by the sensor are not caused by a sudden change in voltage or current in the circuit. The code that allows this in the finalcode.ino file is:

```
Int validCount;

void setup() {
    validCount = 0;
}

void loop() {

   //Other code
```

```
    if(validCount < 5) {
        //Code that send HTTP request
    }else{
        //Calibration
        Serial.println("...Calibrating…");
        validCount++;
    }
}
```

5.0 Conclusion

The development of the Plant Monitoring System marks a significant advancement in the field of automated plant care. By integrating multiple sensors and a user-friendly web interface, the system provides real-time insights and automated responses to maintain optimal growing conditions. This approach not only enhances the efficiency and effectiveness of plant monitoring but also reduces the burden on users, allowing for better plant health and productivity.

Through rigorous testing and validation, the system has demonstrated its reliability and accuracy in measuring environmental parameters and responding appropriately. The ability to notify users of sub-optimal conditions and automatically take corrective actions ensures that plants receive the best possible care, minimising the risk of damage or poor growth due to environmental stress.

Overall, the Plant Monitoring System exemplifies the potential of sensor-based systems in transforming traditional practices, offering a scalable and efficient solution for modern agriculture and horticulture.

5.1 Future Work

a) **Advanced Automation for Plant Care:** Future improvements could focus on further automating plant care tasks, such as integrating an automatic watering system. This system would adjust water delivery based on real-time soil moisture levels and weather conditions, ensuring optimal hydration without user intervention, reducing water waste, and preventing over or under-watering.

b) **Enhanced User Interface:** Improving the graphical user interface (GUI) for a more intuitive and engaging experience is another future work area. This could include visual indicators of plant health, such as animated icons or images of plants appearing happy or sad based on their status. Such visual feedback would help users quickly assess plant health and understand necessary actions, making the interface more user-friendly.

c) **Integration with Smart Home Systems:** Integrating the Plant Monitoring System with smart home ecosystems like Google Home, Amazon Alexa, or Apple HomeKit is another promising development. This integration would allow users to control and monitor their plant care system through voice commands and automated routines,

creating a cohesive smart home environment that optimises living conditions for both plants and inhabitants.

Code for finalcode.ino

```
/*****************************************************************************************
***
 *  Originaly Created By: Tauseef Ahmad
 *  Originialy Created On: 3 April, 2023
 *
 *  Adapted By: Muhammad Syafiq
 *  Adapted On: 11 July 2024
 *
 *  Code was adapted from:
 *
 *  Ahmed Logs - For database connection
 *  - YouTube Video: https://youtu.be/VEN5kgjEuh8
 *  - Youtube Channel:
https://www.youtube.com/channel/UCOXYfOHgu-C-UfGyDcu5sYw/
 *
 *  INOVATRIX - For Soil Moisture Sensor
 *  - Github Repo:
https://github.com/INOVATRIX/ESP8266-SOIL-MOISTURE-SM/tree/main
 *
 *  Co-Pilot
 *  - To Generate the graph code and reactive website
 *

*****************************************************************************************
*/

// Include the DHT sensor library
#include "DHT.h"

// Define the digital pin connected to the DHT sensor and the sensor type
#define DHTPIN 4       // D2 pin on the ESP8266
#define DHTTYPE DHT22  // DHT22 (AM2302) type
DHT dht(DHTPIN, DHTTYPE);

// Include the ESP8266 WiFi and HTTP client libraries
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

//define sound velocity in cm/uS
const int trigPin = 12;  // Trigger pin connected to the Ultrasonic sensor at D6
const int echoPin = 14;  // Echo pin connected to the Ultrasonic sensor at D5
#define SOUND_VELOCITY 0.034
#define CM_TO_INCH 0.393701

//define Ultrasonic related variables
```

```
long duration;
float distanceCm;

//Define fan and buzzer
const int fanPin = 13;
const int buzzPin = 15;

// URL for sending sensor data to the server
String URL = "http://192.168.145.163/finalproj2/test_data.php";

// WiFi credentials
const char* ssid = "V2027";
const char* password = "password";
const int ledPin = 5;

// Variables to store sensor readings
int s1, s2, sensorValue, validCount;
float s3, s4, dist;

//Variables for sensor logic
bool flagS1, flagS2, flagS3, flagS4;


// Setup function runs once at startup
void setup() {
  Serial.begin(115200);      // Start serial communication at 115200 baud rate
  connectWiFi();             // Connect to WiFi network
  dht.begin();               // Initialize the DHT sensor
  pinMode(trigPin, OUTPUT);  // Sets the trigPin as an Output
  pinMode(echoPin, INPUT);   // Sets the echoPin as an Input
  pinMode(fanPin, OUTPUT);   // Sets the fanPin as Output
  pinMode(buzzPin, OUTPUT);  // Sets the buzzPin as Output
  digitalWrite(fanPin, LOW);
  digitalWrite(buzzPin, LOW);
  validCount = 0;
}

// Loop function runs repeatedly
void loop() {
  // Reconnect to WiFi if connection is lost
  if (WiFi.status() != WL_CONNECTED) {
    connectWiFi();
  }

  // Read temperature and humidity from DHT sensor
  s1 = dht.readTemperature();
  s2 = dht.readHumidity();

  // Read distance
```

```
  //Initialise trigger pin to low at the start
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);  //Shoot out signal
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);  //Stops signal

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
  dist = (duration * SOUND_VELOCITY / 2) / 100;

  if (dist > 0.15) {
    dist = 0.15;
  }

  Serial.println("Distance in m: ");
  Serial.println(dist);

  // Calculate volume of resevoir (assume resevoir at full is 0.15m tall, and lets say
1m2 in base area)
  s3 = (0.15 - dist) * 1000;  //Max should be 0.15m3 or 0.15l or 150l

  Serial.println("Volume in Litre:");
  Serial.println(s3);

  // Prints the distance on the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);

  sensorValue = analogRead(A0);       // read the input on analog pin 0
  s4 = (1024 - sensorValue) / 102.4;  // Convert the analog reading (which goes
from 0 - 1023) to a voltage (0 - 5V)

  if (validCount > 5) {
    //Alert
    alertFunc();

    // Prepare data for HTTP POST request
    String postData = "sensor1=" + String(s1) + "&sensor2=" + String(s2) +
"&sensor3=" + String(s3) + "&sensor4=" + String(s4);

    // Create HTTP and WiFi client objects
    HTTPClient http;
    WiFiClient wclient;

    // Begin HTTP connection to the server
```

```
    http.begin(wclient, URL);
    // Set the content type for the POST request
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");

    // Send the POST request and get the response code
    int httpCode = http.POST(postData);
    // Get the response payload from the server
    String payload = http.getString();

    // Check if the HTTP request was successful
    if (httpCode > 0) {
      if (httpCode == HTTP_CODE_OK) {
        // Print the server's response payload
        Serial.println(payload);
      } else {
        // Print the HTTP response code
        Serial.printf("[HTTP] GET... code: %d\n", httpCode);
      }
    } else {
      // Print the HTTP error
      Serial.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
    }

    // Close the HTTP connection
    http.end();

    // Print debugging information
    Serial.print("URL : ");
    Serial.println(URL);
    Serial.print("Data: ");
    Serial.println(postData);
    Serial.print("httpCode: ");
    Serial.println(httpCode);
    Serial.print("payload : ");
    Serial.println(payload);
    Serial.println("-------------------------------------------------");
    // Wait for 5 seconds before repeating the loop
    delay(5000);
  } else {
    Serial.println("...Calibrating...");
    validCount++;
    delay(5000);
  }
}

// Function to connect to the WiFi network
void connectWiFi() {
  // Turn off WiFi to reset any previous configurations
```

```
  WiFi.mode(WIFI_OFF);
  delay(1000);
  // Set WiFi to station mode and connect to the network
  WiFi.mode(WIFI_STA);

  // Begin connection process
  WiFi.begin(ssid, password);
  Serial.println("Connecting to WiFi");

  // Wait until connection is established
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  // Print connection details
  Serial.print("connected to : ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

bool checkFlag(float min, float max, float inp) {

  if (inp < min) {
    Serial.println("Lower Bounds");
    return true;
  } else if (inp > max) {
    Serial.println("Upper Bounds");
    return true;
  } else {
    Serial.println("OK!");
    return false;
  }
}

void alertFunc() {

  float tempMin = 21;
  float tempMax = 37;
  float humMin = 60;
  float humMax = 80;
  float volMin = 80;
  float volMax = 150;
  float lightMin = 5;
  float lightMax = 10;

  flagS1 = checkFlag(tempMin, tempMax, s1);
  flagS2 = checkFlag(humMin, humMax, s2);
```

```
  flagS3 = checkFlag(volMin, volMax, s3);
  flagS4 = checkFlag(lightMin, lightMax, s4);

  if (flagS1 || flagS2 || flagS3 || flagS4) {
    digitalWrite(buzzPin, HIGH);
  } else {
    Serial.println("ALL OK!");
    digitalWrite(buzzPin, LOW);
  }

  if (flagS1) {
    Serial.println("Too Hot");
    if (s1 > tempMax) {
      digitalWrite(fanPin, HIGH);
      Serial.println("FAN ON");
    } else {
      digitalWrite(fanPin, LOW);
      Serial.println("FAN OFF");
    }
  }
}
```

Code for test_data.php

```
<?php
/*********************************************************************************************
***
 *  Originally Created By: Tauseef Ahmad
 *  Originally Created On: 3 April, 2023
 *
 *  YouTube Video: https://youtu.be/VEN5kgjEuh8
 *  My Channel:
https://www.youtube.com/channel/UCOXYfOHgu-C-UfGyDcu5sYw/
 *
 *  Adapted by: Muhammad Syafiq
 *  Adapted on: 11 July 2024
 *

*********************************************************************************************
*/

// Database connection parameters
$hostname = "localhost";
$username = "root";
$password = "";
$database = "midsem";
```

```
// Establish a new database connection
$conn = mysqli_connect($hostname, $username, $password, $database);

// Check if the connection was successful, if not, end the script
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

echo "Database connection is OK<br>";

echo "Attempting to send value";

// Check if all sensor values are set in the POST request
if (isset($_POST["sensor1"]) && isset($_POST["sensor2"]) &&
isset($_POST["sensor3"]) && isset($_POST["sensor4"])) {
    // Retrieve sensor values from the POST request
    $s1 = $_POST["sensor1"];
    $s2 = $_POST["sensor2"];
    $s3 = $_POST["sensor3"];
    $s4 = $_POST["sensor4"];


    // Prepare an SQL query to insert sensor values into the 'sensor' table
    $sql = "INSERT INTO sensor (sensor1, sensor2, sensor3, sensor4) VALUES (" .
$s1 . "," . $s2 . "," . $s3 . "," . $s4 . ")";

    // Execute the SQL query and check if it was successful
    if ($conn->query($sql) === TRUE) {
        echo "Values inserted in MySQL database table.";
        echo "Received sensor values: s1=$s1, s2=$s2, s3=$s3, s4=$s4";
    } else {
        // If the query execution failed, print the error
        echo "Error: " . $sql . "<br>" . $conn->error;
    }
}
?>
```

Code for dashboard.php

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Climate Data</title>
    <link rel="stylesheet"
```

```
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>

<body style="background-color: #303030;">
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="#">Plant Support</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNavAltMarkup"
            aria-controls="navbarNavAltMarkup" aria-expanded="false"
aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
            <div class="navbar-nav">
                <a class="nav-item nav-link active" href="#">Home <span
class="sr-only">(current)</span></a>
                <a class="nav-item nav-link" href="/finalproj2/table.php">Details</a>
            </div>
        </div>
    </nav>
    <div class="container text-center" style="padding: 30px;">
        <div class="card" style="padding: 20px;">
            <h1 class="text-center my-3">Climate Data</h1>
            <div class="row">
                <div class="col-lg-3">
                    <div class="card bg-dark text-white h-70">
                        <div class="card-header">Temperature</div>
                        <div class="card-body">
                            <div class="d-flex justify-content-between align-items-center">
                                <div class="me-3">
                                    <h4>Current:</h4>
                                    <h1 class="text-lg fw-bold" id="currtemperature"></h1>
                                    <br>
                                    <h4>Average:</h4>
                                    <h3 class="text-lg fw-bold" id="avgtemperature"></h3>
                                </div>
                            </div>
                        </div>
                    </div>
                    <div id="tempMsg" class="card text-white text-center h-20"
style="padding: 10px;">
                        <h3></h3>
                        <div class="card-footer">
                            <h4></h4>
                        </div>
                    </div>
                </div>
                <div class="col-lg-3">
```

```
                <div class="card bg-dark text-white h-70">
                    <div class="card-header">Humidity</div>
                    <div class="card-body">
                        <div class="d-flex justify-content-between align-items-center">
                            <div class="me-3">
                                <h4>Current:</h4>
                                <h1 class="text-lg fw-bold" id="currhumidity"></h1>
                                <br>
                                <h4>Average:</h4>
                                <h3 class="text-lg fw-bold" id="avghumidity"></h3>
                            </div>
                        </div>
                    </div>
                </div>
                <div id="humMsg" class="card text-white text-center h-20"
style="padding: 10px;">
                    <h3></h3>
                    <div class="card-footer">
                        <h4></h4>
                    </div>
                </div>
            </div>
            <div class="col-lg-3">
                <div class="card bg-dark text-white h-70">
                    <div class="card-header">Water Tank Volume</div>
                    <div class="card-body">
                        <div class="d-flex justify-content-between align-items-center">
                            <div class="me-3">
                                <h4>Current:</h4>
                                <h1 class="text-lg fw-bold" id="currsoilMoisture"></h1>
                                <br>
                                <h4>Average:</h4>
                                <h3 class="text-lg fw-bold" id="avgsoilMoisture"></h3>
                            </div>
                        </div>
                    </div>
                </div>
                <div id="waterMsg" class="card text-white text-center h-20"
style="padding: 10px;">
                    <h3></h3>
                    <div class="card-footer">
                        <h4></h4>
                    </div>
                </div>
            </div>
            <div class="col-lg-3">
                <div class="card bg-dark text-white h-70">
                    <div class="card-header">Light Level</div>
                    <div class="card-body">
```

```
                    <div class="d-flex justify-content-between align-items-center">
                        <div class="me-3">
                            <h4>Current:</h4>
                            <h1 class="text-lg fw-bold" id="currlight"></h1>
                            <br>
                            <h4>Average:</h4>
                            <h3 class="text-lg fw-bold" id="avglight"></h3>
                        </div>
                    </div>
                </div>
            </div>
            <div id="lightMsg" class="card text-white text-center h-20"
style="padding: 10px;">
                <h3></h3>
                <div class="card-footer">
                    <h4></h4>
                </div>
            </div>
        </div>
    </div>
    <br><br>
    <br>
    <div class="card" style="padding: 20px;">
        <h1 class="text-center my-3">Graphs & Charts</h1>
        <div class="row">
            <div class="col-lg-6">
                <canvas id="temperatureChart"></canvas>
            </div>
            <div class="col-lg-6">
                <canvas id="humidityChart"></canvas>
            </div>
        </div>
        <br>
        <div class="row">
            <div class="col-lg-6">
                <canvas id="soilMoistureChart"></canvas>
            </div>
            <div class="col-lg-6">
                <canvas id="lightChart"></canvas>
            </div>
        </div>
    </div>
</div>

<!-- Do i need this php? -->
<?php
$conn = mysqli_connect("localhost", "root", "", "midsem");
$s1result = mysqli_query($conn, "SELECT sensor1 FROM `sensor` ORDER BY
```

```
`sensor`.`id` DESC LIMIT 60");
   $s1currresult = mysqli_query($conn, "SELECT sensor1 FROM `sensor` ORDER
BY `id` DESC LIMIT 1");

   $temperatureData = [];
   while ($row = mysqli_fetch_array($s1result)) {
      $temperatureData[] = $row['sensor1'];
   }

   $s2result = mysqli_query($conn, "SELECT sensor2 FROM `sensor` ORDER BY
`sensor`.`id` DESC LIMIT 60");
   $s2currresult = mysqli_query($conn, "SELECT sensor2 FROM `sensor` ORDER
BY `id` DESC LIMIT 1");
   $humidityData = [];
   while ($row = mysqli_fetch_array($s2result)) {
      $humidityData[] = $row['sensor2'];
   }

   $s3result = mysqli_query($conn, "SELECT sensor3 FROM `sensor` ORDER BY
`sensor`.`id` DESC LIMIT 60");
   $s3currresult = mysqli_query($conn, "SELECT sensor3 FROM `sensor` ORDER
BY `id` DESC LIMIT 1");
   $soilMoistureData = [];
   while ($row = mysqli_fetch_array($s3result)) {
      $soilMoistureData[] = $row['sensor3'];
   }

   $s4result = mysqli_query($conn, "SELECT sensor4 FROM `sensor` ORDER BY
`sensor`.`id` DESC LIMIT 60");
   $s4currresult = mysqli_query($conn, "SELECT sensor4 FROM `sensor` ORDER
BY `id` DESC LIMIT 1");
   $lightData = [];
   while ($row = mysqli_fetch_array($s4result)) {
      $lightData[] = $row['sensor4'];
   }

   $temperatureDataJson = json_encode($temperatureData);
   $humidityDataJson = json_encode($humidityData);
   $soilMoistureDataJson = json_encode($soilMoistureData);
   $lightDataJson = json_encode($lightData);

   $currtempDataJson =
json_encode(mysqli_fetch_assoc($s1currresult)['sensor1']);
   $currhumDataJson =
json_encode(mysqli_fetch_assoc($s2currresult)['sensor2']);
   $currmoistDataJson =
json_encode(mysqli_fetch_assoc($s3currresult)['sensor3']);
   $currlightDataJson = json_encode(mysqli_fetch_assoc($s4currresult)['sensor4']);
   ?>
```

```
<script type="text/javascript">
    var temperatureData = <?php echo $temperatureDataJson; ?>;
    var humidityData = <?php echo $humidityDataJson; ?>;
    var soilMoistureData = <?php echo $soilMoistureDataJson; ?>;
    var lightData = <?php echo $lightDataJson; ?>;

    var currtemperatureData = <?php echo $currtempDataJson; ?>;
    var currhumidityData = <?php echo $currhumDataJson; ?>;
    var currsoilMoistureData = <?php echo $currmoistDataJson; ?>;
    var currlightData = <?php echo $currlightDataJson; ?>;

    var labels = Array.from({ length: 60 }, (_, i) => i + 1); // labels for 60 seconds

    var ctx1 = document.getElementById('temperatureChart').getContext('2d');
    var temperatureChart = new Chart(ctx1, {
        type: 'line',
        data: {
            labels: labels,
            datasets: [{
                label: 'Temperature',
                data: temperatureData,
                backgroundColor: 'rgba(255, 99, 132, 0.2)',
                borderColor: 'rgba(255, 99, 132, 1)',
                borderWidth: 1
            }]
        }
    });

    var ctx2 = document.getElementById('humidityChart').getContext('2d');
    var humidityChart = new Chart(ctx2, {
        type: 'line',
        data: {
            labels: labels,
            datasets: [{
                label: 'Humidity',
                data: humidityData,
                backgroundColor: 'rgba(75, 192, 192, 0.2)',
                borderColor: 'rgba(75, 192, 192, 1)',
                borderWidth: 1
            }]
        }
    });

    var ctx3 = document.getElementById('soilMoistureChart').getContext('2d');
    var soilMoistureChart = new Chart(ctx3, {
        type: 'line',
        data: {
```

```
        labels: labels,
        datasets: [{
            label: 'Water Level',
            data: soilMoistureData,
            backgroundColor: 'rgba(153, 102, 255, 0.2)',
            borderColor: 'rgba(153, 102, 255, 1)',
            borderWidth: 1
        }]
    }
});

var ctx4 = document.getElementById('lightChart').getContext('2d');
var lightChart = new Chart(ctx4, {
    type: 'line',
    data: {
        labels: labels,
        datasets: [{
            label: 'Light Level',
            data: lightData,
            backgroundColor: 'rgba(80, 152, 50, 0.2)',
            borderColor: 'rgba(80, 152, 50, 1)',
            borderWidth: 1
        }]
    }
});

function calculateAverage(data) {
    const total = data.reduce((sum, value) => sum + parseFloat(value), 0);
    return (total / data.length).toFixed(2);
}

function updateStatus(cardId, value, min, max, type) {
    var card = document.getElementById(cardId);
    var flag = false;
    if (value >= min && value <= max) {
        card.className = "card bg-success text-white text-center h-20";
        card.querySelector("h3").innerText = "Perfect";
        card.querySelector("h4").innerText = "No need to worry! ";

    } else {
        card.className = "card bg-warning text-white text-center h-20";
        card.querySelector("h3").innerText = "Caution";
        var prompt = "";
        if (value < min) {
            switch (type) {
                case 'temp':
                    prompt = "Increase the temperature!"; break;
                case 'hum':
                    prompt = "Increase the humidity!"; break;
```

```
                case 'water':
                    prompt = "Fill up the tank!"; break;
                case 'light':
                    prompt = "Too Dark! Let in more sun!"; break;
                default:
                    prompt = "Out of range! Danger! less than"; break;

            }

        } else if (value > max) {
            switch (type) {
                case 'temp':
                    prompt = "Decrease the temperature!"; break;
                case 'hum':
                    prompt = "Decrease the humidity"; break;
                case 'water':
                    prompt = "Tank overflowing!"; break;
                case 'light':
                    prompt = "Too Bright! Need more shade!"; break;
                default:
                    prompt = "Out of range! Danger! more than"; break;

            }

        }
        flag = true;
        card.querySelector("h4").innerText = prompt;
      }
      return flag;
    }

    document.getElementById('currtemperature').innerText = currtemperatureData
+ '°C';
    document.getElementById('currhumidity').innerText = currhumidityData + '%';
    document.getElementById('currsoilMoisture').innerText = currsoilMoistureData
+ 'l';
    document.getElementById('currlight').innerText = currlightData + '/10';

    document.getElementById('avgtemperature').innerText =
calculateAverage(temperatureData) + '°C';
    document.getElementById('avghumidity').innerText =
calculateAverage(humidityData) + '%';
    document.getElementById('avgsoilMoisture').innerText =
calculateAverage(soilMoistureData) + 'l';
    document.getElementById('avglight').innerText = calculateAverage(lightData)
+ '/10';

    updateStatus("tempMsg", currtemperatureData, 21, 37, 'temp');
    updateStatus("humMsg", currhumidityData, 60, 80, 'hum');
```

```
        updateStatus("waterMsg", currsoilMoistureData, 80, 150, 'water');
        updateStatus("lightMsg", currlightData, 5, 10, 'light');

        // Set an interval to fetch new data from the server every 5 seconds
        setInterval(function () {
            fetch('fetch_data.php') // Fetch data from the server
                .then(response => response.json()) // Parse the response as JSON
                .then(data => {
                    // Update the chart data with the new data from the server
                    temperatureChart.data.datasets[0].data = data.temperatureData;
                    humidityChart.data.datasets[0].data = data.humidityData;
                    soilMoistureChart.data.datasets[0].data = data.soilMoistureData;
                    lightChart.data.datasets[0].data = data.lightData;

                    //Update the current values
                    document.getElementById('currtemperature').innerText =
data.currtemperatureData + '°C';
                    document.getElementById('currhumidity').innerText =
data.currhumidityData + '%';
                    document.getElementById('currsoilMoisture').innerText =
data.currsoilMoistureData + 'l';
                    document.getElementById('currlight').innerText = data.currlightData +
'/10';

                    // Update the averages
                    document.getElementById('avgtemperature').innerText =
calculateAverage(data.temperatureData) + '°C';
                    document.getElementById('avghumidity').innerText =
calculateAverage(data.humidityData) + '%';
                    document.getElementById('avgsoilMoisture').innerText =
calculateAverage(data.soilMoistureData) + 'l';
                    document.getElementById('avglight').innerText =
calculateAverage(data.lightData) + '/10';

                    // Update the status cards
                    flagTemp = updateStatus("tempMsg", data.currtemperatureData, 21,
37, 'temp');
                    flagHum = updateStatus("humMsg", data.currhumidityData, 60, 80,
'hum');
                    flagWater = updateStatus("waterMsg", data.currsoilMoistureData, 80,
150, 'water');
                    flagLight = updateStatus("lightMsg", data.currlightData, 5, 10, 'light');

                    if (flagTemp || flagHum || flagWater || flagLight) {
                        alert("Caution! Check on your plants!");
                    }

                    // Update the charts to reflect the new data
                    temperatureChart.update();
```

```
                humidityChart.update();
                soilMoistureChart.update();
                lightChart.update();
            });
        }, 5000); // Fetch new data every 5 seconds

    </script>
</body>

</html>
```

Code for table.php

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Climate Data</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>

<?php
$conn = mysqli_connect("localhost", "root", "", "midsem");
$results = mysqli_query($conn, "SELECT * FROM `sensor` ORDER BY
`sensor`.`id` DESC LIMIT 100");

$id = [];
$timestamp = [];
$temperatureData = [];
$humidityData = [];
$soilMoistureData = [];
$lightData = [];

while ($row = mysqli_fetch_array($results)) {
    $id[] = $row['id'];
    $timestamp[] = $row['timestampp'];
    $temperatureData[] = $row['sensor1'];
    $humidityData[] = $row['sensor2'];
    $soilMoistureData[] = $row['sensor3'];
    $lightData[] = $row['sensor4'];
}
```

```
$idJson = json_encode($id);
$timestampJson = json_encode($timestamp);
$temperatureDataJson = json_encode($temperatureData);
$humidityDataJson = json_encode($humidityData);
$soilMoistureDataJson = json_encode($soilMoistureData);
$lightDataJson = json_encode($lightData);
?>

<body style="background-color: #303030;">
   <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <a class="navbar-brand" href="#">Plant Support</a>
      <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNavAltMarkup"
         aria-controls="navbarNavAltMarkup" aria-expanded="false"
aria-label="Toggle navigation">
         <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
         <div class="navbar-nav">
            <a class="nav-item nav-link" href="/finalproj2/dashboard.php">Home</a>
            <a class="nav-item nav-link active"
href="/finalproj2/table.php">Details<span
                  class="sr-only">(current)</span></a>
         </div>
      </div>
   </nav>
   <div class="container" style="padding: 20px;">
      <div class="card" style="padding: 20px;">
         <h1 class="text-center my-4">Climate Data</h1>

         <div class="datatable-container">
            <table id="datatablesSimple" class="table table-stiped">
               <thead>
                  <tr>
                     <th>ID</th>
                     <th>Timestamp</th>
                     <th>Temperature (°C)</th>
                     <th>Humidity (%)</th>
                     <th>Tank Volume (l)</th>
                     <th>Light Level (x/10)</th>
                  </tr>
               </thead>
               <tbody>
                  <script type="text/javascript">
                     var id = <?php echo $idJson; ?>;
                     var timestamp = <?php echo $timestampJson; ?>;
                     var temperatureData = <?php echo $temperatureDataJson; ?>;
                     var humidityData = <?php echo $humidityDataJson; ?>;
                     var soilMoistureData = <?php echo $soilMoistureDataJson; ?>;
```

```
                    var lightData = <?php echo $lightDataJson; ?>

                    var tableBody = document.querySelector('tbody');

                    for (var i = 0; i < id.length; i++) {
                        var row = document.createElement('tr');

                        var cellId = document.createElement('td');
                        cellId.textContent = id[i];
                        row.appendChild(cellId);

                        var cellTimestamp = document.createElement('td');
                        cellTimestamp.textContent = timestamp[i];
                        row.appendChild(cellTimestamp);

                        var cellTemperature = document.createElement('td');
                        cellTemperature.textContent = temperatureData[i];
                        row.appendChild(cellTemperature);

                        var cellHumidity = document.createElement('td');
                        cellHumidity.textContent = humidityData[i];
                        row.appendChild(cellHumidity);

                        var cellSoilMoisture = document.createElement('td');
                        cellSoilMoisture.textContent = soilMoistureData[i];
                        row.appendChild(cellSoilMoisture);

                        var cellLight = document.createElement('td');
                        cellLight.textContent = lightData[i];
                        row.appendChild(cellLight);

                        tableBody.appendChild(row);
                    }
                </script>
            </tbody>

        </table>
    </div>
</div>
</div>
</script>
</body>

</html>
```

Code for fetch_data.php

```php
<?php
// Establish a new database connection
$conn = mysqli_connect("localhost", "root", "", "midsem");

// Check the connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Fetch the latest 60 data points from the database for each sensor
$s1result = mysqli_query($conn, "SELECT sensor1 FROM `sensor` ORDER BY
`sensor`.`id` DESC LIMIT 60");
$s2result = mysqli_query($conn, "SELECT sensor2 FROM `sensor` ORDER BY
`sensor`.`id` DESC LIMIT 60");
$s3result = mysqli_query($conn, "SELECT sensor3 FROM `sensor` ORDER BY
`sensor`.`id` DESC LIMIT 60");
$s4result = mysqli_query($conn, "SELECT sensor4 FROM `sensor` ORDER BY
`sensor`.`id` DESC LIMIT 60");

// Fetch the current values for each sensor
$curr1result = mysqli_query($conn, "SELECT sensor1 FROM `sensor` ORDER BY
`sensor`.`id` DESC LIMIT 1");
$curr2result = mysqli_query($conn, "SELECT sensor2 FROM `sensor` ORDER BY
`sensor`.`id` DESC LIMIT 1");
$curr3result = mysqli_query($conn, "SELECT sensor3 FROM `sensor` ORDER BY
`sensor`.`id` DESC LIMIT 1");
$curr4result = mysqli_query($conn, "SELECT sensor4 FROM `sensor` ORDER BY
`sensor`.`id` DESC LIMIT 1");

// Initialize arrays to store the sensor data
$temperatureData = [];
$humidityData = [];
$soilMoistureData = [];
$lightData = [];

// Fetch the temperature data from the database and store it in the
temperatureData array
while ($row = mysqli_fetch_assoc($s1result)) {
    $temperatureData[] = $row['sensor1'];
}

// Fetch the humidity data from the database and store it in the humidityData array
while ($row = mysqli_fetch_assoc($s2result)) {
    $humidityData[] = $row['sensor2'];
}

// Fetch the soil moisture data from the database and store it in the
soilMoistureData array
while ($row = mysqli_fetch_assoc($s3result)) {
```

```
    $soilMoistureData[] = $row['sensor3'];
}

// Fetch the light data from the database and store it in the lightData array
while ($row = mysqli_fetch_assoc($s4result)) {
    $lightData[] = $row['sensor4'];
}

// Calculate the averages for the first 10 data points
$totalTemp = array_sum(array_slice($temperatureData, 0, 10)) / 10;
$totalHum = array_sum(array_slice($humidityData, 0, 10)) / 10;
$totalSoil = array_sum(array_slice($soilMoistureData, 0, 10)) / 10;
$totalLight = array_sum(array_slice($lightData, 0, 10)) / 10;

// Fetch the current values for each sensor
$currTemp = mysqli_fetch_assoc($curr1result)['sensor1'];
$currHum = mysqli_fetch_assoc($curr2result)['sensor2'];
$currSoil = mysqli_fetch_assoc($curr3result)['sensor3'];
$currLight = mysqli_fetch_assoc($curr4result)['sensor4'];

// Set the content type of the response to application/json
header('Content-Type: application/json');

// Encode the sensor data arrays as a JSON object and echo it as the response
echo json_encode([
    'temperatureData' => $temperatureData,
    'humidityData' => $humidityData,
    'soilMoistureData' => $soilMoistureData,
    'lightData' => $lightData,
    'currtemperatureData' => $currTemp,
    'currhumidityData' => $currHum,
    'currsoilMoistureData' => $currSoil,
    'currlightData' => $currLight,
    'avgTemp' => $totalTemp,
    'avgHum' => $totalHum,
    'avgSoil' => $totalSoil,
    'avgLight' => $totalLight
]);
?>
```