PHASE 5 : PROJECT DOCUMENTATION & SUBMISSION

COLLEGE : MADRAS INSTITUTE TECHNOLOGY OF,ANNA UNIVERSITY

S.NO	NAME	REG NO
1.	JAGANNATHAN S	2021504711
2.	NITHESH K	2021504713
3.	GOKUL RAJ S	2021504714
4.	SATHISH KUMAR A	2021504715

The project involves developing an IoT-based air quality monitoring system with the primary objective of providing real-time air quality data to the public through an accessible online platform. The goal is to create awareness about the impact of air quality on public health and the environment.

KEY OBJECTIVES:

1. **Data Collection:** Deploy IoT devices equipped with air quality sensors to measure critical parameters, including CO, NO2, SO2, O3, and VOCs.

2. **Real-Time :** Monitoring: Install these devices strategically across the target area to continuously monitor air quality.

3. **Data Transmission:** Establish efficient data transmission protocols for sending sensor data to a central server.

4. **User-Friendly Platform:** Develop an intuitive web or mobile platform where the public can access real-time and historical air quality data.

5. **Data Visualization**: Implement data visualization tools to present air quality information clearly.

6. **Alerting System**: Create an alerting mechanism to notify users when air quality levels exceed predefined thresholds.

7. **Public Engagement**: Encourage public engagement by providing educational content on the platform, explaining the significance of air quality, and promoting community involvement.

8. **Scalability**: Design the system to accommodate future expansion with additional sensors and monitoring locations.

9. **Python Integration**: Utilize Python for data analysis, processing, and platform development.

DESIGN THINKING:

1. **Understand User Needs:** Begin by understanding the perspectives and needs of potential users, including the public, environmentalists, and authorities. Explore the air quality challenges in the target area.

2. **Define the Problem:** Clearly define the core problem and project objectives based on user insights and identified pain points related to accessing air quality information.

3. **Generate Ideas**: Brainstorm creative solutions for the IoT monitoring system and the awareness platform, involving diverse team members in the ideation process.

4. **Prototype:** Build a simple prototype of the IoT device and user interface, focusing on core functionalities to gather feedback.

5. **Test and Iterate:** Collect user feedback during usability tests with the prototype and make necessary improvements based on insights.

6. **Develop the Solution**: Create the complete IoT-based monitoring system and user platform, aligning with defined objectives and user needs.

7. **Deploy and Launch:** Implement the system, deploying IoT devices and launching the user platform to make real-time air quality data accessible to the public.

8. **Evaluate and Improve:** Continuously monitor system performance, gather user feedback, and make ongoing improvements based on data analytics and stakeholder input.

9. **Iterate and Adapt:** Keep the user at the center of decision-making, be open to change, and consider expansion or additional features as needed to effectively raise public awareness about air quality.

IMPORTANT COMPONENTS:

Here are the important components for an IoT-based air quality monitoring system:

1. Sensor Devices
2. Data Transmission and Communication
3. Central Data Management System
4. Data Analysis and Processing
5. Alerting System
6. User Interface and Visualization
7. Geospatial Mapping

- After thorough research and analysis, we arrived at an innovative solution to solve the above problem as detailed in phase 1 of our project.
- We will be using the ESP32 micro controller as well as Arduino UNO microcontroller as both these suit the best for our project.
- We made this choice because we only require data on the concentrations of CO 2 ,NO 2 and smoke in the desired atmosphere to be posted on a public platform.

SENSOR

- We use MQ135 sensor in our air quality monitoring system because it can effectively detect CO2,NO2 gases and smoke and provides valuable data for comprehensive air quality assessment ensuring environmental safety.

CONNECTIVITY

- Wi-Fi as it enables real time data transmission, allowing us to monitor air quality remotely

CLOUD

- We use Beeceptor it ensures scalability, data storage and analytics.

PROTOCOL

- HTTP: These are widely used protocols for transmitting data over the internet which are easy to implement and are supported by almost all web server.

PUBLIC PLATFORM

- We are going to design a website for Air quality monitoring system.

PROBLEM:

To develop the python script on IoT devices as per the project requirement.

SOLUTION:

An Arduino-based air quality monitoring system offers real-time data on pollutants like CO2 and NO2, temperature, humidity, and particulate matter. This information is transmitted to a central server and displayed through a user-friendly platform, enabling informed decisions, alerts, and fostering environmental awareness and community involvement.

SOURCE CODE:

```
#define BLYNK_TEMPLATE_ID “TMPLwToQUqRw”
 #define BLYNK_TEMPLATE_NAME “Air Quality Monitoring”
#define BLYNK_AUTH_TOKEN “7kuX0IEEPHLVRSK2Jhgf81qpCgL3D0Nr”
#define BLYNK_PRINT Serial
#include
#include
 #include
 #include LiquidCrystal_I2C lcd(0x27, 16, 2); byte degree_symbol[8] =
{
0b00111,
0b00101,
0b00111,
0b00000,
 0b00000,
0b00000,
0b00000,
0b00000
};
Char auth[] = BLYNK_AUTH_TOKEN;
 Char ssid[] = “Wokwi-GUEST”; // type your wifi name
 Char pass[] = “”; // type your wifi password
 BlynkTimer timer;
 Int gas = 32; I
nt sensorThreshold = 100;
#define DHTPIN 2 //Connect Out pin to D2 in NODE MCU
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
Void sendSensor()
{
Float h = dht.readHumidity();
```

```
Float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit
If (isnan(h) || isnan(t)) {
Serial.println("Failed to read from DHT sensor!");
Return;
}
Int analogSensor = analogRead(gas);
Blynk.virtualWrite(V2, analogSensor);
Serial.print("Gas Value: ");
Serial.println(analogSensor);
// You can send any value at any time.
// Please don't send more that 10 values per second.
Blynk.virtualWrite(V0, t);
Blynk.virtualWrite(V1, h);
Serial.print("Temperature : ");
 Serial.print(t);
 Serial.print(" Humidity : ");
 Serial.println(h);
}
Void setup()
{
Serial.begin(115200);
//pinMode(gas, INPUT);
Blynk.begin(auth, ssid, pass);
Dht.begin();
Timer.setInterval(30000L, sendSensor);
//Wire.begin(); Lcd.begin(16,2);
 // lcd.backlight();
// lcd.clear();
Lcd.setCursor(3,0);
Lcd.print("Air Quality");
Lcd.setCursor(3,1);
Lcd.print("Monitoring");
 Delay(2000); Lcd.clear();
}
Void loop()
{
 Blynk.run();
 Timer.run();
Float h = dht.readHumidity();
Float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit
Int gasValue = analogRead(gas);
Lcd.setCursor(0,0);
 Lcd.print("Temperature ");
```

```
 Lcd.setCursor(0,1);
Lcd.print(t);
Lcd.setCursor(6,1);
Lcd.write(1);
Lcd.createChar(1, degree_symbol);
Lcd.setCursor(7,1);
Lcd.print("C");
Delay(4000);
Lcd.clear();
Lcd.setCursor(0, 0);
Lcd.print("Humidity ");
Lcd.print(h); Lcd.print("%");
 Delay(4000);
Lcd.clear();
//lcd.setCursor(0,0);
 // lcd.print(gasValue);
 // lcd.clear();
Serial.println("Gas Value");
Serial.println(gasValue);
 If(gasValue
{
Lcd.setCursor(0,0);
Lcd.print("Gas Value: ");
Lcd.print(gasValue);
Lcd.setCursor(0, 1);
Lcd.print("Fresh Air");
Serial.println("Fresh Air");
Delay(4000); Lcd.clear();
}
Else
 if(gasValue>1200)
{
 Lcd.setCursor(0,0);
Lcd.print(gasValue);
Lcd.setCursor(0, 1);
Lcd.print("Bad Air");
Serial.println("Bad Air");
Delay(4000);
Lcd.clear();
}
If(gasValue>1200)
{
//Blynk.email(karthiaenit@gmail.com, "Alert", "Bad Air!");
```

```
 Blynk.logEvent("pollution_alert","Bad Air");
}
}
```

LIBRARY:

The code relies on standard Arduino libraries like Wire, Adafruit_Sensor, Adafruit_BME280, and ESP8266WiFi, which can be installed via the Arduino IDE. The "WokwiHTTPClient" is specific to the Wokwi simulation environment.

Setup() Function:

- Initializes the Arduino and serial communication for debugging.
- Attempts to connect to a Wi-Fi network with the provided SSID and password.
- Continues to print "Connecting to WiFi..." until a successful connection is established.
- Once connected to Wi-Fi, it prints "Connected to WiFi."

Loop() Function:

- Continuously runs in a loop to collect sensor data and send it to the server.
- Reads CO2, NO2, and smoke sensor values using the readCO2() method from the MQ135 sensor library.
- Calls the sendDataToServer() function with the sensor readings.

SendDataToServer() Function:

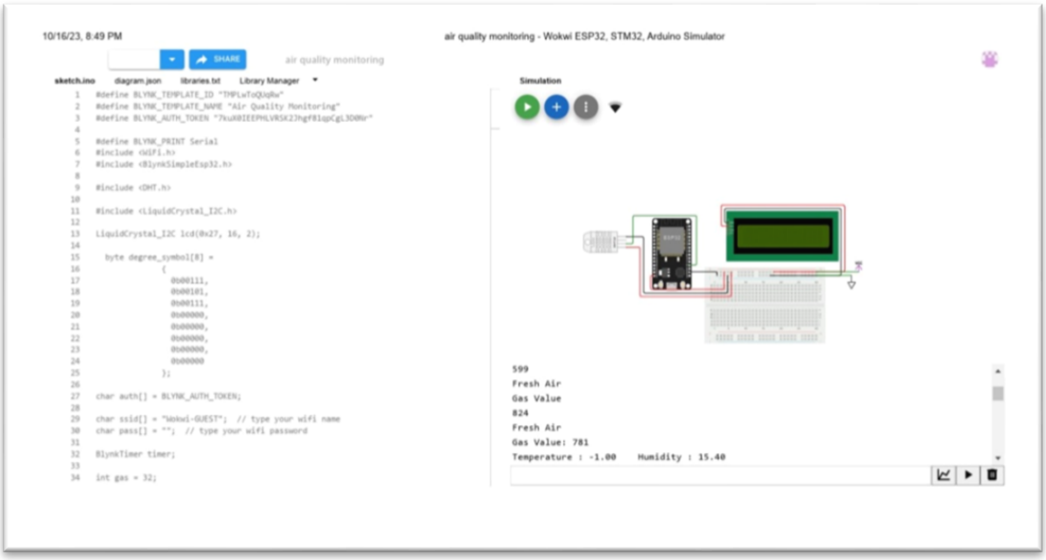
- Prepares data in the form of a string containing CO2, NO2, and smoke values.
- Uses the WokwiHTTPClient library (for simulation purposes) to make an HTTP POST request to the specified serverURL.
- Adds the "Content-Type" header to specify the data format.
- Checks the HTTP response code. If the code is greater than zero, it means a successful response.
- If successful, it prints the HTTP response to the serial monitor. Otherwise, it prints an HTTP error. Closes the HTTP connection with http.end().

Overall, this code initializes the Wi-Fi connection, reads air quality sensor data, sends this data to a server via HTTP POST, and provide feedback via the serial monitor. It's designed for simulation purposes, but with the appropriate server setup, it could be adapted for real-world air quality monitoring.

SIMULATION LINK:

https://wokwi.com/projects/378756614608888833

SIMULATION IMAGES



- In this technology project we will continue building our project by developing the platform as per project requirement.
- The platform we are going to develop is through Beeceptor cloud with HTTP protocol.

CODE :

```
#define BLYNK_TEMPLATE_ID "TMPLwToQUqRw"
#define BLYNK_TEMPLATE_NAME "Air Quality Monitoring"
#define BLYNK_AUTH_TOKEN "7kuX0IEEPHLVRSK2Jhgf81qpCgL3D0Nr"
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <DHT.h>
#include <LiquidCrystal_I2C.h>
#include <HTTPClient.h>
#include <WiFiClient.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
byte degree_symbol[8] = {
0b00111,
0b00101,
0b00111,
0b00000,
0b00000,
0b00000,
0b00000,
0b00000

};
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Wokwi-GUEST"; // type your WiFi name
char pass[] = ""; // type your WiFi password
BlynkTimer timer;
```

```
int gas = 32;
int sensorThreshold = 100;
#define DHTPIN 2 // Connect Out pin to D2 in NODE MCU
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
// Beeceptor endpoint URL
const char* beeceptorURL = "https://akmns.free.beeceptor.com";
void sendDataToBeeceptor(float temperature, float humidity, int gasValue) {
HTTPClient http;
// Build the JSON payload
String payload = "{\"temperature\":" + String(temperature, 2) +
",\"humidity\":" + String(humidity, 2) +
",\"gasValue\":" + String(gasValue) + "}";
// Send the POST request to Beeceptor
http.begin(beeceptorURL);
http.addHeader("Content-Type", "application/json");
int httpResponseCode = http.POST(payload);
if (httpResponseCode > 0) {
Serial.print("HTTP Response Code: ");
Serial.println(httpResponseCode);
String response = http.getString();
Serial.println(response);
} else {
Serial.print("HTTP Error: ");
Serial.println(httpResponseCode);
}
http.end();
}
void sendSensor() {
float h = dht.readHumidity();
float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit
if (isnan(h) || isnan(t)) {
Serial.println("Failed to read from DHT sensor!");
return;
}
int analogSensor = analogRead(gas);
Blynk.virtualWrite(V2, analogSensor);
Serial.print("Gas Value: ");
Serial.println(analogSensor);
Blynk.virtualWrite(V0, t);
Blynk.virtualWrite(V1, h);
Serial.print("Temperature: ");
Serial.print(t);
Serial.print(" Humidity: ");
Serial.println(h);
// Send data to Beeceptor
SendDataToBeeceptor(t, h, analogSensor);
}
void setup() {
```

```
Serial.begin(115200);
Blynk.begin(auth, ssid, pass);
dht.begin();
timer.setInterval(30000L, sendSensor);
lcd.begin(16, 2);
lcd.setCursor(3, 0);
lcd.print("Air Quality");
lcd.setCursor(3, 1);
lcd.print("Monitoring");
delay(2000);
lcd.clear();
}
void loop() {
Blynk.run();
timer.run();
float h = dht.readHumidity();
float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit
int gasValue = analogRead(gas);
lcd.setCursor(0, 0);
lcd.print("Temperature ");
lcd.setCursor(0, 1);
lcd.print(t);
lcd.setCursor(6, 1);
lcd.write(1);
lcd.createChar(1, degree_symbol);
lcd.setCursor(7, 1);
lcd.print("C");
delay(4000);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Humidity ");
lcd.print(h);
lcd.print("%");
delay(4000);
lcd.clear();
Serial.println("Gas Value");
Serial.println(gasValue);
if (gasValue < 1200) {
lcd.setCursor(0, 0);
lcd.print("Gas Value: ");
lcd.print(gasValue);
lcd.setCursor(0, 1);
lcd.print("Fresh Air");
Serial.println("Fresh Air");
delay(4000);
lcd.clear();
} else if (gasValue > 1200) {
lcd.setCursor(0, 0);
lcd.print(gasValue);
```

```
lcd.setCursor(0, 1);
lcd.print("Bad Air");
Serial.println("Bad Air");
delay(4000);
lcd.clear();
}
if (gasValue > 1200) {
// Blynk.email(karthiaenit@gmail.com, "Alert", "Bad Air!");
Blynk.logEvent("pollution_alert", "Bad Air");
}
}
```

WOKWI SIMULATION LINK WITH BEECEPTOR CLOUD:

https://wokwi.com/projects/379988279376105473

****