IoT Based Air Quality Monitoring System

Noman Amin

Roll: 1903101

Department of Computer Science and Engineering , Rajshahi University of Engineering & Technology

Course No: CSE 2100

Supervisor: Dr. Mir Md. Jahangir Kabir, Professor, Dept. of CSE, RUET

October 11, 2022

Abstract

The density of contaminated air in the area rises as a result of an increase in the emissions of harmful gases from automobiles and industry. Depending on how bad the air pollution is, it can cause early death, bronchitis, lung illness, asthma, and other health issues in people. Additionally, air pollution can contribute to climate change, acid rain, unfavorable agricultural growth, and water supply source pollution. So, to assess the air quality, a smart monitoring system is necessary. This project suggests a system that would continuously check the air quality and notify the user via the Blynk mobile app. It is made out of an ESP32 microcontroller module, a 16x2 LCD display, a MQ135 gas sensor, a DHT11 temperature sensor, and some required connections. By connecting the ESP32 module to a Wi-Fi network, real-time data is transmitted, and the outcome is shown on an LCD display and in a mobile app. When the ESP32 module and the user are linked to a Wi-Fi network, the users can receive updates from anywhere.

Introduction

The release of toxic gases by industries, vehicle emissions, and an increase in the amount of harmful gases and particulate matter in the atmosphere are all contributing to air pollution. Due to variables that can harm human health, such as industries, urbanization, population growth, and automobile use, the level of pollution is rising quickly. The most major factor in the rise in air pollution is particulate matter. This necessitates the measurement and analysis of real-time air quality monitoring in order to enable prompt decision-making.

Therefore, the creation of an air quality monitoring system that can send notifications of updates on the levels of air quality is crucial to resolving the problem of hazardous gas poisoning in the environment. This project showcases low-cost, Internet of Things (IoT)-based air quality monitoring that integrates the Blynk App with an ESP32 chip. Microcontroller units (MCUs), electronic software frameworks, and Internet of Things (IoT) systems with embedded air quality sensors offer data sharing and the corresponding information.

Methodology

Project Development

The block diagram of the air quality monitoring system is shown in **Figure 1**. The MCU must be powered by a 5V supply in order to be turned on. The NodeMCU is attached to the DHT11 temperature sensor and the MQ135 gas sensor, and the communication network is connected to the NodeMCU output pin. The NodeMCU is programmed using the Arduino IDE in accordance with the specifications for the sensor unit. Using the Blynk app, the NodeMCU's Wi-Fi module transmits all of the sensor data to mobile apps and a web server. The user will receive a notification if the sensor determines that the air in the area is contaminated.

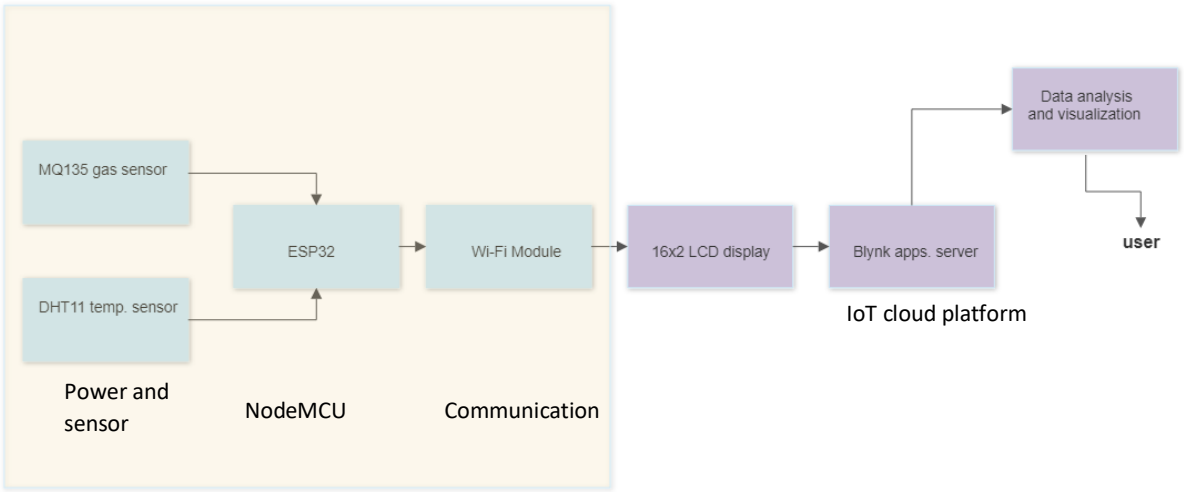

Figure 1: Block Diagram of the air quality monitoring system

The system's functioning is shown in **Figure 2**. First, the system must be initialized with a power supply at 5 V. The connected sensors are made sure to be in an active state so that the NodeMCU ESP32 microcontroller may receive data from the sensors. Over a Wi-Fi network, all sensor data is transmitted to an LCD display and a Blynk server. Blynk is a cutting-edge platform that allows users to create interfaces mostly from the Android and ios operating systems to track and monitor preferred applications. Once the communication network connection cannot be detected, the process is repeated beginning with the sensor installation inspection until the issue with the communication network connection is rectified.

The acquired data are then examined and used in data visualization to obtain information updates on the state of the air. Finally, the information results are transmitted through Wi-Fi for in-depth user monitoring.

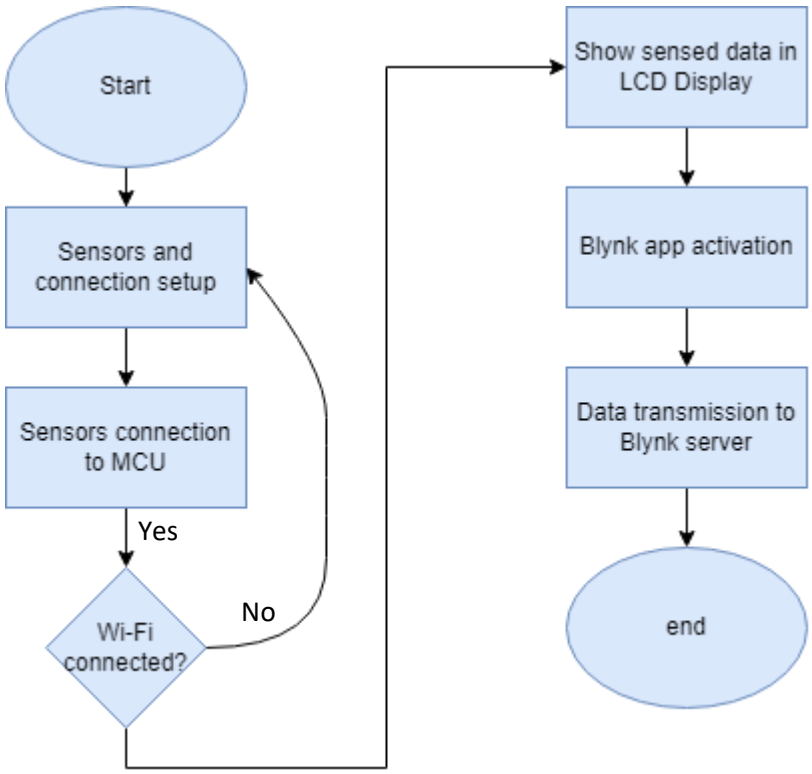

Figure 2: Flow chart of the system

Hardware Component

Figure 3 shows the circuit diagram of the air quality monitoring system. The system consists of NodeMCU ESP32 , a MQ135 sensor, a DHT11 sensor, I2C combination, an LCD 16x2 display.

ESP32 is a series of low-cost, low-power system on a chip microcontroller with integrated Wi-Fi and dual-mode Bluetooth [1].

The MQ-135 gas sensor senses gases like ammonia nitrogen, oxygen, alcohols, aromatic compounds, sulfide, and smoke. The boost converter of the chip MQ-3 gas sensor is PT1301. The operating voltage of this gas sensor is from 2.5V to 5.0V. The MQ-3 gas sensor has a lower conductivity to clean the air as a gas sensing material [2].

DHT11 is a low-cost digital sensor for sensing temperature and humidity. This sensor can be easily interfaced with any micro-controller such as Arduino, Raspberry Pi etc... to measure humidity and temperature instantaneously. The temperature range of DHT11 is from 0 to 50 degree Celsius with a 2-degree accuracy. Humidity range of this sensor is from 20 to 80% with 5% accuracy. The sampling rate of this sensor is 1Hz .i.e., it gives one reading for every second. DHT11 is small in size with operating voltage from 3 to 5 volts. The maximum current used while measuring is 2.5mA [3].

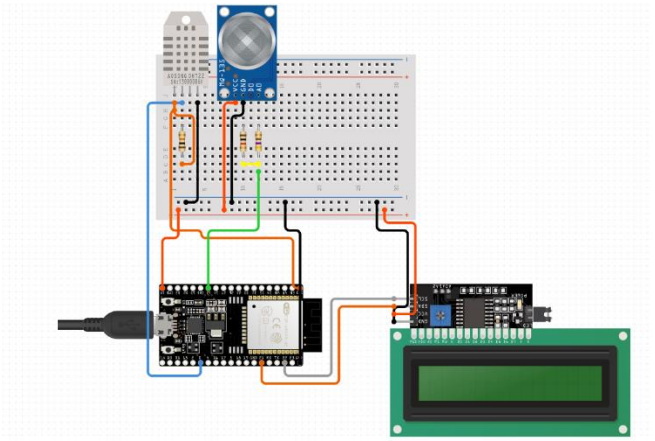

Figure 3: Circuit diagram for the air quality monitoring system

Numerical Evaluation

The most important phase is calibrating the sensor in clean air, after which an equation is created to transform the output voltage value of the sensor into our useful units PPM. The mathematical computations are shown below.

From Ohm's Law, we can derive I as follows:

(1) $$I = V / R$$

Which in our circuit is equal to:

(2) $$I = V_C / (R_S+R_L)$$

Going back to our original circuit from Figure 4, we can obtain the output voltage at the load resistor using the value obtained for I and Ohm's Law.

(3) $$V = I \times R$$

(4) $$V_{RL}= [V_C / (R_S + R_L)] \times R_L$$

(5) $$V_{RL} = (V_C \times R_L) / (R_S + R_L)$$

So now we solve for RS:

(6) $$V_{RL} \times (R_S + R_L) = V_C \times R_L$$

(7) $$(V_{RL} \times R_S) + (V_{RL} \times R_L) = V_C \times R_L$$

(8) $$(V_{RL} \times R_S) = (V_C \times R_L) - (V_{RL} \times R_L)$$

(9) $$R_S = [(V_C \times R_L) - (V_{RL} \times R_L)] / V_{RL}$$

(10) $$R_S = [(V_C \times R_L) / V_{RL}] – R_L$$

This formula will help us find the values of the sensor resistance for different gases [4].

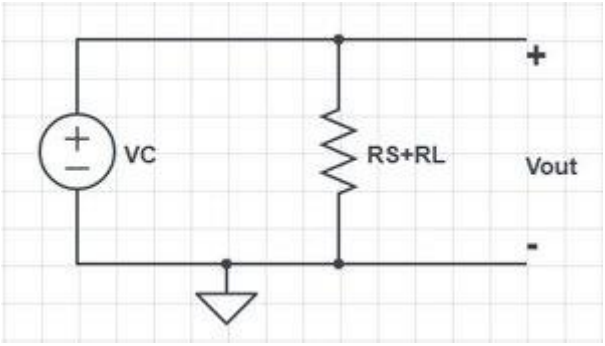

Figure 4: Internal diagram of MQ135 gas sensor

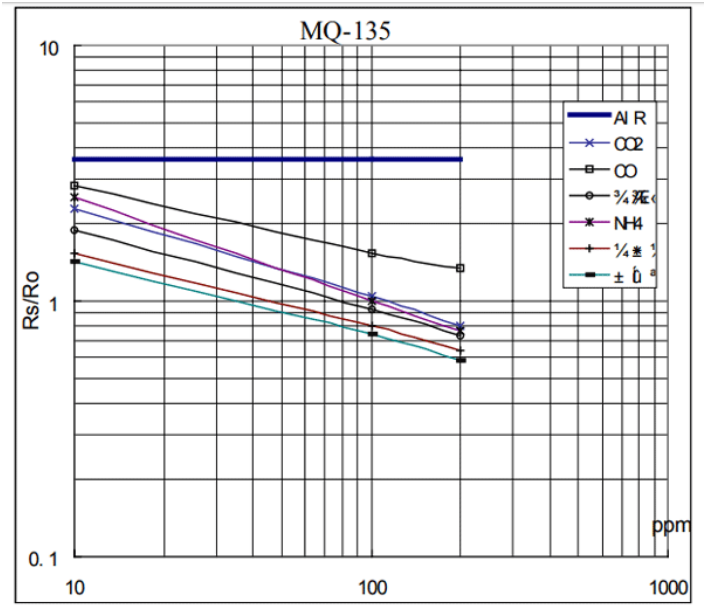

Figure 5: Datasheet of MQ135 sensor

The resistance ratio is:

(11) $$\frac{R_S}{R_0} = 3.6$$

The value 3.6 in equation 11 is derived from the datasheet mentioned in **figure 5**. To calculate R0, we will need to find the value of the Rs in fresh air. This will be done by taking the analog average readings from the sensor and converting it to voltage. Then we will use the R_S formula to find R_O [4].

The datasheet in figure 5 is used to convert the output of the sensor to the related ppm physical characteristics for the gas under test. The graphic above in figure 5 seems a power function

(12) $$y = a \times x^b$$

using power regression, we can obtain scaling factor (a), and exponent (b), for the gas we would like to measure. For CO_2 ppm = 116.6020682 (R_S/R_O)^-2.769034857 [5]&[6].

For power regression R programming language and WebPlotDigitizer [7] were used.

Software Development

The system includes integration of the peripheral devices with the system. The main component of the air quality monitoring system is a NodeMCU ESP32, along with a number of sensors, an LCD, and communication connections. The Blynk application must be synchronized with all hardware and virtual connections. An IoT platform called Blynk is used to remotely manage Arduino microcontrollers. The programming portion is then developed using an Arduino IDE application as an editor. Once the programming portion was finished, the code was uploaded using the MCU serial interface to the NodeMCU ESP32 board. The ESP32 board's integrated Wi-Fi module was then used to connect it to the internet. The sensors subsequently communicated the sensed data to the Blynk server and the LCD display after connecting to Wi-Fi. An interval of 6 seconds was used to update the data. The information was then examined and used by the server to provide end users with information updates about the state of the air quality based on notification alerts.

Programming

Definitions for the Blynk server

```
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPLhuml4X2a"
#define BLYNK_DEVICE_NAME "Air Quality Monitoring"
#define BLYNK_AUTH_TOKEN "a7Q-8_ZelDvFGI1v3oegdAhBCK5MfqJ9"
```

Including header file

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include<WebServer.h>
#include "DHT.h"
#include "MQ135.h"
#include <MQUnifiedsensor.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

Definitions for the DHT11 sensor

```
#define DHTTYPE DHT11
#define buz 23
```

Definitions for the MQ135 sensor

```
#define placa "ESP-32"
#define Voltage_Resolution 3.3
#define pin 34 //Analog input 0 of your arduino
#define type "MQ-135" //MQ135
#define ADC_Bit_Resolution 12 // For arduino UNO/MEGA/NANO
#define RatioMQ135CleanAir 3.6//RS / R0 = 3.6 ppm
MQUnifiedsensor MQ135(placa, Voltage_Resolution, ADC_Bit_Resolution,
pin, type);
```

Calibrating MQ135 sensor (with the details of the used function)

```
void setup() {
  Serial.begin(9600); //Baud rate
}

void loop() {
  float sensor_volt; //Define variable for sensor voltage
  float RS_air; //Define variable for sensor resistance
  float R0; //Define variable for R0
  float sensorValue = 0.0; //Define variable for analog readings
  for(int x = 0 ; x < 500 ; x++) //Start for loop
  {
    sensorValue = sensorValue + analogRead(34); //Add analog values
of sensor 500 times
  }
  sensorValue = sensorValue/500.0; //Take average of readings
  sensor_volt = sensorValue*(3.3/4095.0); //Convert average to
voltage
  RS_air = ((3.3*10.0)/sensor_volt)-10.0; //Calculate RS in fresh
air
  R0 = RS_air/3.6; //Calculate R0

  Serial.print("R0 = "); //Display "R0"
  Serial.println(R0); //Display value of R0
  delay(1000); //Wait 1 second
}
```

Calibrating MQ135 sensor (used in programming)

```
      // ******* CALIBRATING SENSOR
  MQ135.setRegressionMethod(1); // to set if the graph is linear or
exponential 1->exponential
  MQ135.init();
  Serial.println("Calribrating please wait : ");
  float calcR0 = 0;
  for(int i = 1; i <= 10; i++)
  {
    MQ135.update();
    calcR0 += MQ135.calibrate(RatioMQ135CleanAir);
    Serial.print(".");
  }
  MQ135.setR0(calcR0/10);
  Serial.println("   done!.");
  if(isinf(calcR0)) {Serial.println("Warning: Conection issue
founded, R0 is infite (Open circuit detected) please check your
wiring and supply"); while(1);}
  if(calcR0 == 0){Serial.println("Warning: Conection issue founded,
R0 is zero (Analog pin with short circuit to ground) please check
your wiring and supply"); while(1);}
```

Complete program

```
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPLhuml4X2a"
#define BLYNK_DEVICE_NAME "Air Quality Monitoring"
#define BLYNK_AUTH_TOKEN "a7Q-8_ZelDvFGI1v3oegdAhBCK5MfqJ9"

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include<WebServer.h>
#include "DHT.h"
#include "MQ135.h"
#include <MQUnifiedsensor.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

#define DHTTYPE DHT11
#define buz 23
// definitions for MQ135
#define placa "ESP-32"
#define Voltage_Resolution 3.3
#define pin 34 //Analog input 0 of your arduino
#define type "MQ-135" //MQ135
#define ADC_Bit_Resolution 12 // For arduino UNO/MEGA/NANO
#define RatioMQ135CleanAir 3.6//RS / R0 = 3.6 ppm
MQUnifiedsensor MQ135(placa, Voltage_Resolution, ADC_Bit_Resolution,
pin, type);
```

```
BlynkTimer timer;

// *****AUTHENTICATION FOR IOT SERVER
const char *auth = BLYNK_AUTH_TOKEN;
const char* ssid = "White_Walker";
const char* password = "noman6080";

WebServer server(80);

uint8_t DHTPin = 4;

DHT dht(DHTPin, DHTTYPE);

float temperature;
float humidity;
int threshold = 400;

void sendSensor()
{
  //************* TEMPERATUR AND HUMIDTY

  temperature = dht.readTemperature();
  humidity = dht.readHumidity();
  float heat_index = dht.computeHeatIndex(temperature, humidity,
false); //compute heat index in celcius (isFahrenheit = false)
  Serial.println("-----------------------------------------------");
  Serial.print("Temperature is : ");
  Serial.print(temperature);
  Serial.println(" °C");
  Serial. print("Humidity is :");
  Serial.print(humidity);
  Serial.println(" %");
  Serial.print("Heat index is ");
  Serial.print(heat_index);
  Serial.println(" °C");
  Serial.println("-----------------------------------------------");

   // MQ Reading and updating section
  MQ135.update(); // Update data, the arduino will be read the
voltage on the analog pin
  MQ135.setA(605.18); MQ135.setB(-3.937); // Configurate the
ecuation values to get CO concentration
  float CO = MQ135.readSensor(); // Sensor will read PPM
concentration using the model and a and b values setted before or in
the setup
  MQ135.setA(77.255); MQ135.setB(-3.18); // Configurate the ecuation
values to get Alcohol concentration
  float Alcohol = MQ135.readSensor(); // Sensor will read PPM
concentration using the model and a and b values setted before or in
the setup
  MQ135.setA(110.47); MQ135.setB(-2.862); // Configurate the
ecuation values to get CO2 concentration
```

```
  float CO2 = 410 + MQ135.readSensor(); // Sensor will read PPM
concentration using the model and a and b values setted before or in
the setup
  MQ135.setA(44.947); MQ135.setB(-3.445); // Configurate the
ecuation values to get Toluene concentration
  float Toluene = MQ135.readSensor(); // Sensor will read PPM
concentration using the model and a and b values setted before or in
the setup
  MQ135.setA(102.2 ); MQ135.setB(-2.473); // Configurate the
ecuation values to get NH4 concentration
  float NH4 = MQ135.readSensor(); // Sensor will read PPM
concentration using the model and a and b values setted before or in
the setup
  MQ135.setA(34.668); MQ135.setB(-3.369); // Configurate the
ecuation values to get Acetone concentration
  float Acetone = MQ135.readSensor(); // Sensor will read PPM
concentration using the model and a and b values setted before or in
the setup

  // ****** WRITING IN IOT SERVER
  Blynk.virtualWrite(V0,temperature);
  Blynk.virtualWrite(V1,humidity);
  Blynk.virtualWrite(V2, CO2);
  Blynk.virtualWrite(V8, heat_index);
  Blynk.virtualWrite(V3, NH4);
  Blynk.virtualWrite(V4, Alcohol);
  Blynk.virtualWrite(V5, Toluene);
  Blynk.virtualWrite(V6, CO);
  Blynk.virtualWrite(V7, Acetone);

  Serial.print("CO: "); Serial.println(CO);
  Serial.print("Alcohol: "); Serial.println(Alcohol);

  Serial.print("CO2: ");
  Serial.println(CO2);

  Serial.print("TOLUENE: ");
  Serial.println(Toluene);

  Serial.print("NH4: ");
  Serial.println(NH4);

  Serial.print("ACETONE: ");
  Serial.println(Acetone);

  if(CO2 > 450 || CO > 50 || Alcohol > 50 || Toluene > 50 || NH4 >
50 || Acetone > 50)
  {
    Serial.println("heavyyyy!!!");
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Air quality is");
    lcd.setCursor(0,1);
    lcd.print("   poor!!!");
```

```
  digitalWrite(buz,HIGH);
}
else
{
  Serial.println("Good");
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Air quality is");
  lcd.setCursor(0,1);
  lcd.print("   good.");
  digitalWrite(buz,LOW);
}

delay(2000);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Temp: ");
lcd.print(temperature);
lcd.setCursor(13,0);
lcd.print("C");
lcd.setCursor(0,1);
lcd.print("Hum: ");
lcd.print(humidity);
lcd.setCursor(13,1);
lcd.print("%");
delay(3000);

lcd.clear();
lcd.setCursor(0,0);
lcd.print("CO2: ");
lcd.print(CO2);
lcd.setCursor(13,0);
lcd.print("ppm");
lcd.setCursor(0,1);
lcd.print("CO: ");
lcd.print(CO);
lcd.setCursor(13,1);
lcd.print("ppm");
delay(3000);

lcd.clear();
lcd.setCursor(0,0);
lcd.print("Alc: ");
lcd.print(Alcohol);
lcd.setCursor(13,0);
lcd.print("ppm");
lcd.setCursor(0,1);
lcd.print("Toluene: ");
lcd.print(Toluene);
lcd.setCursor(13,1);
lcd.print("ppm");
delay(3000);

lcd.clear();
```

```
  lcd.setCursor(0,0);
  lcd.print("NH4: ");
  lcd.print(NH4);
  lcd.setCursor(13,0);
  lcd.print("ppm");
  lcd.setCursor(0,1);
  lcd.print("Acetone: ");
  lcd.print(Acetone);
  lcd.setCursor(13,1);
  lcd.print("ppm");
  delay(3000);
}

void setup() {

  Serial.begin(115200);
  delay(100);

  lcd.begin();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("                ");
  lcd.setCursor(0,1);
  lcd.print("                ");
  lcd.setCursor(0,0);
  lcd.print("Air Quality");
  lcd.setCursor(0,1);
  lcd.print("   Analyzer");

  pinMode(buz, OUTPUT);
  pinMode(DHTPin, INPUT);
  dht.begin();

  // **** CONNECTING TO WI-FI
  Serial.println("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while(WiFi.status()!= WL_CONNECTED){
    delay(1000);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi Connected...!");
  Serial.print("Got IP: "); Serial.println(WiFi.localIP());
  Serial.println("");

  Serial .println("");

  // ******* CALIBRATING SENSOR
  MQ135.setRegressionMethod(1); // to set if the graph is linear or
exponential 1->exponential
  MQ135.init();
```

```
  Serial.println("Calribrating please wait : ");
  float calcR0 = 0;
  for(int i = 1; i <= 10; i++)
  {
    MQ135.update();
    calcR0 += MQ135.calibrate(RatioMQ135CleanAir);
    Serial.print(".");
  }
  MQ135.setR0(calcR0/10);
  Serial.println("   done!.");
  if(isinf(calcR0)) {Serial.println("Warning: Conection issue
founded, R0 is infite (Open circuit detected) please check your
wiring and supply"); while(1);}
  if(calcR0 == 0){Serial.println("Warning: Conection issue founded,
R0 is zero (Analog pin with short circuit to ground) please check
your wiring and supply"); while(1);}

  //CONNECT TO BLYNK IOT SERVER
  Blynk.begin(auth, ssid, password);
  timer.setInterval(1000L, sendSensor);
}

void loop() {
  Blynk.run();
  timer.run();

}
```

Result and Discussion

Paper or tissue paper burning was used to evaluate the air quality monitoring device for data collection because it is a source of CO and CO2, which showed that the system was functioning properly. Every 12 seconds, data were collected.

The LCD displays the sensor data, which is subsequently transmitted directly to the user through a Wi-Fi connection after being detected by the sensor. Through the developed Blynk App, users are notified of changes in air quality. An LCD module that displays the information gathered from the MQ135 gas sensor is shown in Figure 6. An alert will be issued to the Blynk applications when the air quality is bad to remind users to take the appropriate safety measures.

Figure 6: LCD display showing the sensed data

When the MQ135 gas sensor identified gases in the environment, Figure 7 displays the graphical user interface (GUI) display result in the Blynk app. A notification that was delivered to the user when the air quality was bad is shown in Figure 8. The data was continuously updated by the server every 6 seconds.

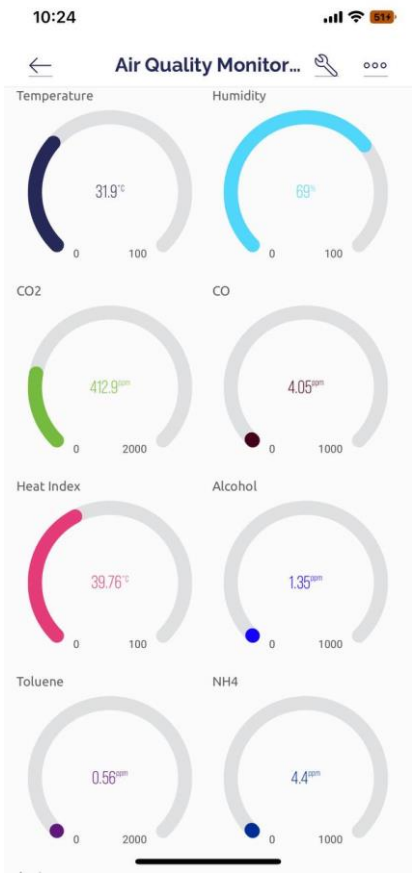

Figure 7: IoT application outcome

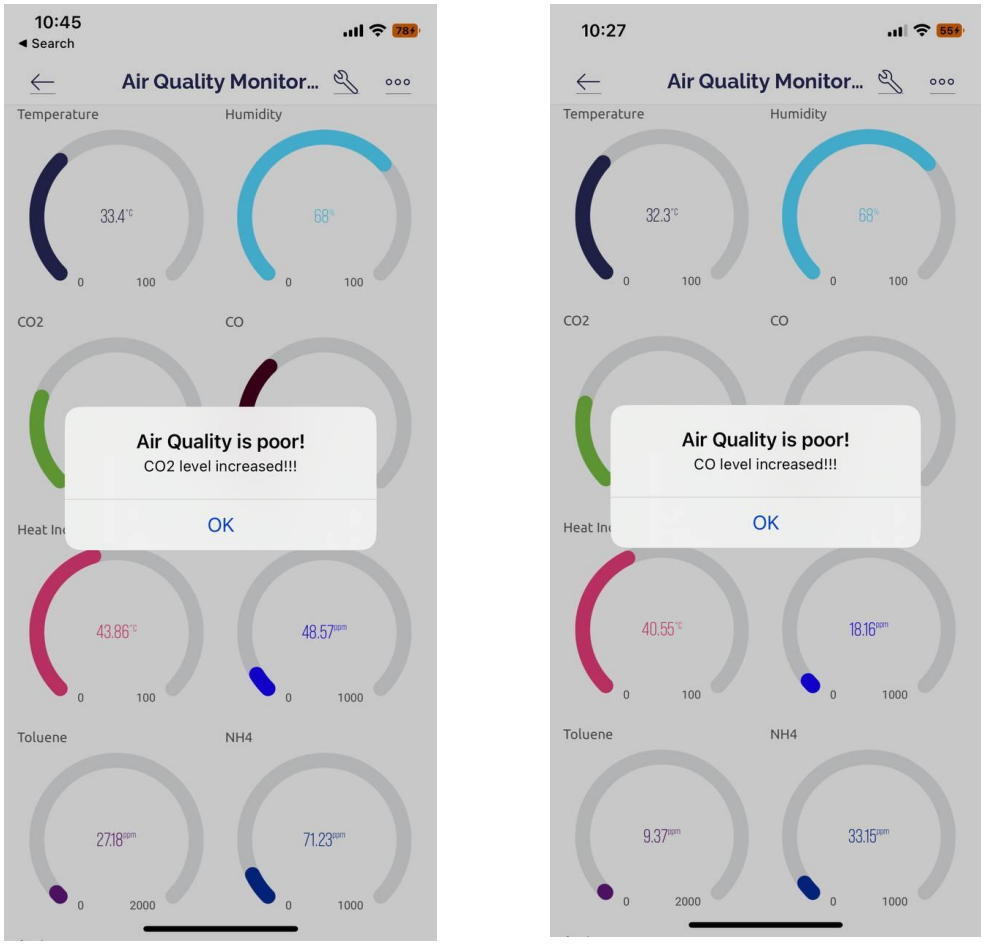

Figure 8: display of a notification when the gas density rises in accordance with the gas level

Conclusion

This paper describes the creation of the ESP32-based Blynk IoT platform for air quality monitoring system. To show the air quality monitoring system, implementation through system experimentation was carried out. A number of significant accomplishments from the air quality monitoring system were made, including: (1) the development of IoT-based systems that employ mobile applications to deliver warnings or messages depending on gas concentrations and temperatures in the atmosphere, and (2) the dependability and longevity of the sensors used in the expandable system allow easy installation of the platform by the user into a variety of acceptable monitoring situations.

The IoT and artificial intelligence can be combined in the future to enhance air quality monitoring systems and enable automatic implementation. Additionally, the device

system can be upgraded in the future by attaching an automated ventilation system, which would allow it to activate when it senses the presence of filthy air nearby. It can also be improved by using machine learning.

References

[1] ESP32. (2022, September 7). In Wikipedia. https://en.wikipedia.org/wiki/ESP32

[2] Agarwal, T. (2016, May 4). MQ135 Alcohol Sensor Circuit and Its Working. ElProCus - Electronic Projects for Engineering Students. Retrieved September 22, 2022, from https://www.elprocus.com/mq-135-alcohol-sensor-circuit-and-working/

[3] Agarwal, T. (2019, August 5). DHT11 Sensor Definition, Working and Applications. ElProCus - Electronic Projects for Engineering Students. Retrieved September 25, 2022, from https://www.elprocus.com/a-brief-on-dht11-sensor/

[4] How Do Gas Sensors Work? | Jaycon Systems. (2016, May 9). Jaycon Systems | Product Design, PCB & Injection Molding. Retrieved September 25, 2022, from https://jayconsystems.com/blog/understanding-a-gas-sensor

[5] Gironi, D., & complete profile, V. M. (2017, May 1). MQ gas sensor correlation function estimation by datasheet. Davide Gironi: MQ Gas Sensor Correlation Function Estimation by Datasheet. Retrieved September 27, 2022, from http://davidegironi.blogspot.com/2017/05/mq-gas-sensor-correlation-function.html#.YzgdgnZBy3A

[6] Gironi, D., & complete profile, V. M. (2014, January 25). Cheap CO2 meter using the MQ135 sensor with AVR ATmega. Davide Gironi: Cheap CO2 Meter Using the MQ135 Sensor With AVR ATmega. Retrieved September 27, 2022, from http://davidegironi.blogspot.com/2014/01/cheap-co2-meter-using-mq135-sensor-with.html#.YzgeyXZBy3A

[7] Rohatgi, A. (n.d.). WebPlotDigitizer - Copyright 2010-2022 Ankit Rohatgi. WebPlotDigitizer - Copyright 2010-2022 Ankit Rohatgi. Retrieved October 1, 2022, from https://apps.automeris.io/wpd/

[8] Setiawan, F. N., & Kustiawan, I. (2018, July 1). IoT based Air Quality Monitoring - IOPscience. IoT Based Air Quality Monitoring - IOPscience. Retrieved October 1, 2022, from https://iopscience.iop.org/article/10.1088/1757-899X/384/1/012008

[9] Kinnera, Bharath Kumar Sai & Subbareddy, Somula & Luhach, Ashish. (2019). IOT based Air Quality Monitoring System Using MQ135 and MQ7 with Machine Learning Analysis. Scalable Computing: Practice and Experience. 20. 599-606. 10.12694/scpe.v20i4.1561.