10/27/2023

IOT:PHASE 4

Name: Abhishek A Stephen

Reg.No:950321104002

Naan Mudalvan id:
DA23CBF861C8240A2594EDE2CCF68DBA

Internet of Things (IOT)-PHASE 4

[Document subtitle]

1. AIR QUALITY MONITORING

Nowadays, most of us out there face the problem of impure air quality due to the increase in pollutants and impurities mixed with the air. Pollutants like CO2, CO, CH4 emitted from vehicles and SO2, SO4 and other gases emitted from industries add to the amount of impurity in the air. Also, fireworks and crackers during festive seasons, occasions and events further increase the pollution level in the atmosphere.

As a result of this, various health-related and respiratory issues are also increasing among the common masses.

So to keep a track of the air quality around you, this is a mini IoT based air quality monitoring system that will monitor the air quality over a web server using the internet and alarm or trigger if the air quality goes poorer than a certain level, i.e. if there are sufficient amount of harmful gases present in the air. It shows the air quality in PPM on the LCD as well as on the webpage so that you can monitor it easily.

Objective

To measure and monitor the air pollution levels

To detect highly polluted conditions

To alert user as and when the levels cross a specified limit

Components you'll require

ESP32 system on a chip microcontroller

MQ135 gas sensor

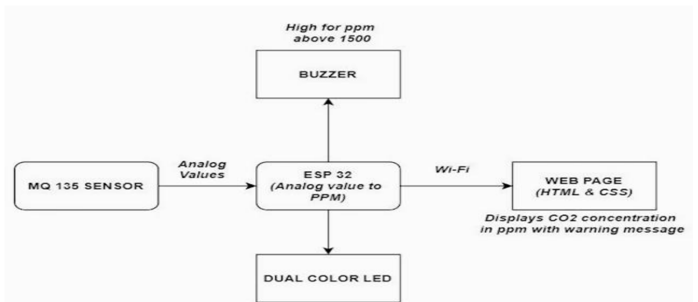
LEDs

Resistors

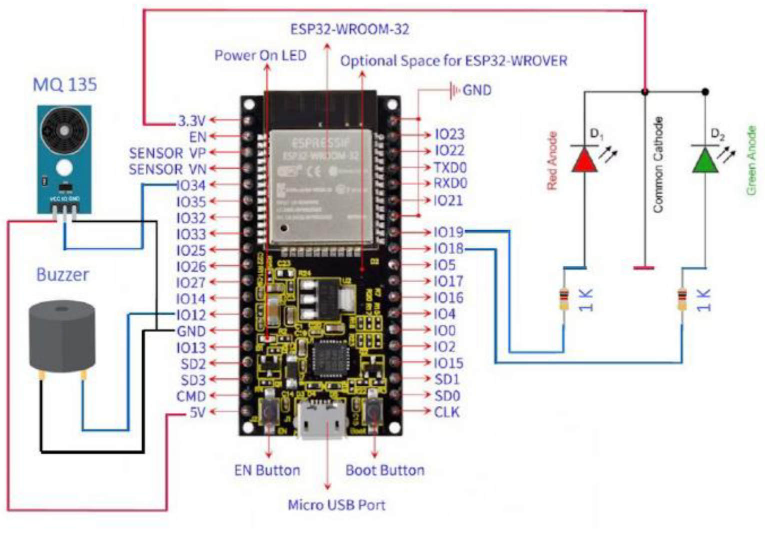
Buzzer

Some wires and a breadboard

Block Diagram



Circuit



Software Used

Arduino IDE

Working

Make the connections as per the circuit diagram above.

Connect the ESP32 with your computer/laptop. Select a suitable port and Esp32 board and hit 'Upload' to upload the code into the ESP32.

(I've used the MQ135 gas sensor here which is basically a Air Quality sensor capable of sensing gases like NH3, NOx, alcohol, Benzene, smoke, CO2 and other harmful gases. MQ135 will sense the gases and we will get the Pollution/Air quality level in PPM (parts per million). The MQ135 sensor will give us output in form of voltage levels and we convert it into PPM (our code and the module libraries will take care of that).

For this project, I've set good air quality levels upto 800 PPM, poor levels in between 800 to 2000 and alert us if it exceeds 2000 PPM.)

Provide the sensor with different types of smoke inputs and observe the output PPM levels in the web browser. All the web browser implementation is taken care by the code.

Real time PPM values can also be viewed/observed in the Arduino IDE Serial Monitor. Just paste the IP address in the web browser and observe the PPM levels.

Green LED will be ON until the PPM level is 800, Red LED will be ON post 800 mark and Red LED with Buzzer will be ON post 2000 mark.

Code

```
#include “MQ135.h”

#include <WiFi.h>

#include <Wire.h>

Int air_quality;

Const char* ssid     = “Enter your wifi”;          //Replace with your network name

Const char* password = “Enter your wifi password”;  //Replave with your network password

// Set web server port number to 80

WiFiServer server(80);

// Variable to store the HTTP request

String header;

// Current time

Unsigned long currentTime = millis();

// Previous time

Unsigned long previousTime = 0;

// Define timeout time in milliseconds (example: 2000ms = 2s)

Const long timeoutTime = 2000;

Void setup() {

  Delay(1000);

  Serial.begin(115200);

  pinMode(12, OUTPUT);      //Buzzer will be output to ESP32

  pinMode(18, OUTPUT);      //Green Led will be Output to ESP32

  pinMode(19, OUTPUT);      //Red Led will be output to ESP32

  pinMode(34, INPUT);      //Gas sensor will be an input to the ESP32

  digitalWrite(18, HIGH);
  digitalWrite(19, HIGH);
  Serial.print(“Connecting to “);
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  While (WiFi.status() != WL_CONNECTED) {
    Delay(500);
    Serial.print(“.”);
  }
  // Print local IP address and start web server
  Serial.println(“”);
  Serial.println(“WiFi connected.”);
  Serial.println(“IP address: “);
  Serial.println(WiFi.localIP());
  Server.begin();
}
Void loop() {

  MQ135 gasSensor = MQ135(34);
  Float air_quality = gasSensor.getPPM();

  If (air_quality >= 2000)
  {
    digitalWrite(12, HIGH);   //Buzzer is ON
    digitalWrite(19, LOW);    //RED LED is ON
    digitalWrite(18, HIGH);   //GREEN LED is OFF
    delay(1000);
    digitalWrite(12, LOW);
  }
  If (air_quality >800 && air_quality <2000)
```

```
{
  digitalWrite(12, LOW);     //Buzzer is OFF
  digitalWrite(19, LOW);     //RED LED is ON
  digitalWrite(18, HIGH);    //GREEN LED is OFF
  delay(1000);
  digitalWrite(12, LOW);
}
If (air_quality < 800)
{
  digitalWrite(12, LOW);     //Buzzer is OFF
  digitalWrite(18, LOW);     //GREEN LED is ON
  digitalWrite(19, HIGH);    //RED LED id OFF
}
Delay(100);
Serial.print(air_quality);
Serial.println(“ PPM”);
Delay(200);
WiFiClient client = server.available();   // Listen for incoming clients


If (client) {                              // If a new client connects,
  currentTime = millis();
  previousTime = currentTime;
  Serial.println(“New Client.”);          // print a message out in the serial port
  String currentLine = “”;                // make a String to hold incoming data from the client
  While (client.connected() && currentTime – previousTime <= timeoutTime) {  // loop while the client’s
connected
    currentTime = millis();
    if (client.available()) {            // if there’s bytes to read from the client,
      char c = client.read();            // read a byte, then
```

```
        Serial.write©;                    // print it out the serial monitor
        Header += c;
        If (c == '\n') {                  // if the byte is a newline character
          // if the current line is blank, you got two newline characters in a row.
          // that's the end of the client HTTP request, so send a response:
          If (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
            // and a content-type so the client knows what's coming, then a blank line:
            Client.println("HTTP/1.1 200 OK");
            Client.println("Content-type:text/html");
            Client.println("Connection: close");
            Client.println();

            // Display the HTML web page
            Client.println("<!DOCTYPE html><html>");
            Client.println("<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1\">");
            Client.println("<link rel=\"icon\" href=\"data:,\">");
            // CSS to style the table
            Client.println("<style>body { text-align: center; font-family: \"Trebuchet MS\", Arial;}");
            Client.println("table { border-collapse: collapse; width:35%; margin-left:auto; margin-right:auto;
}");
            Client.println("th { padding: 12px; background-color: #0043af; color: white; }");
            Client.println("tr { border: 1px solid #ddd; padding: 12px; }");
            Client.println("tr:hover { background-color: #bcbcbc; }");
            Client.println("td { border: none; padding: 12px; }");
            Client.println(".sensor { color:black; font-weight: bold; padding: 1px; }");

            // Web Page Heading
```

```
 Client.println("</style></head><body><h2>AIR POLLUTION MONITOR</h2>");
If (air_quality < 800)
{
  Client.println("</style></head><body><h2>FRESH AIR!</h2>");
}
Else if (air_quality >800 && air_quality <2000)
{
  Client.println("</style></head><body><h2>POOR AIR ☹</h2>");
}
      Else if (air_quality >= 2000)
      {
        Client.println("</style></head><body><h2>ALERT!TOXIC AIR!ALERT</h2>");
      }

Client.println("<table><tr><th>MEASUREMENT</th><th>VALUE</th></tr>");
Client.println("<tr><td>Pollution PPM</td><td><span class=\"sensor\">");
Client.println(air_quality);
Client.println("</span></td></tr></table>");

// The HTTP response ends with another blank line
Client.println();
// Break out of the while loop
Break;
} else { // if you got a newline, then clear currentLine
currentLine = "";
}
} else if (c != '\r') {  // if you got anything else but a carriage return character,
currentLine += c;      // add it to the end of the currentLine
```

```
      }
     }
   }
   // Clear the header variable
   Header = “”;
   // Close the connection
   Client.stop();
   Serial.println(“Client disconnected.”);
   Serial.println(“”);
  }
}
```

Conclusion

This is how you can build a small scale air quality monitoring system of your own and in the process, learn about Arduino, ESP32, environmental impacts of pollutants etc. on the go.