AN PRESENTATION ON AIR QUALITY MONITORING

Based on forecast air quality and historical data and System in ESP32

Submitted by

k.Badri prasath

P.Ganapathi

k.Anjali

p.Indhumathi

k.karthika

INTRO

Since 1970, implementation of the Clean Air Act and technological advances from American innovators have dramatically improved air quality in the U.S. Since that time, the combined emissions of criteria and precursor pollutants have dropped by 78%. Cleaner air provides important public health benefits, and we commend our state, local, community and industry partners for helping further long-term improvement in our air quality.

Air Quality Trends Show Clean Air Progress

Nationally, concentrations of air pollutants have dropped significantly since 1990:

Carbon Monoxide (CO) 8-Hour, 79%
Lead (Pb) 3-Month Average, 85% (from 2010)
Nitrogen Dioxide (NO2) Annual, 61%
Nitrogen Dioxide (NO2) 1-Hour, 54%
Ozone (O3) 8-Hour, 21%

Particulate Matter 10 microns (PM10) 24-Hour, 32%

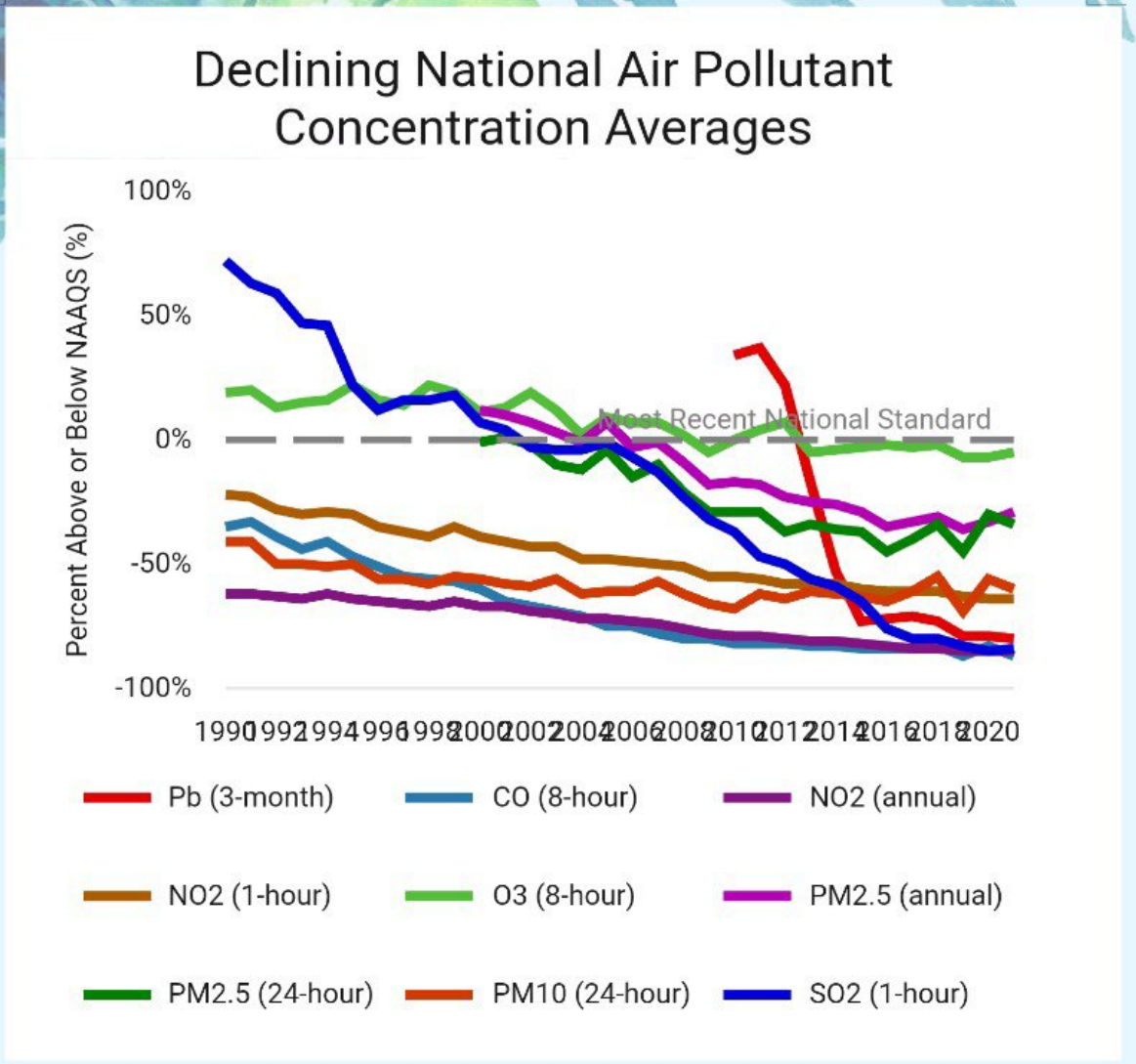
Particulate Matter 2.5 microns (PM2.5) Annual, 37% (from 2000)

Particulate Matter 2.5 microns (PM2.5) 24-Hour, 33% (from 2000)

Sulfur Dioxide (SO2) 1-Hour, 91%

Numerous air toxics have declined with percentages varying by pollutant

Despite increases in air concentrations of pollutants associated with fires, carbon monoxide and particle pollution, national average air quality concentrations remain below the current, national standards.



Air Pollution Includes Gases and Particles

Air pollution consists of gas and particle contaminants that are present in the atmosphere. Gaseous pollutants include sulfur dioxide (SO2), oxides of nitrogen (NOx), ozone (O3), carbon monoxide (CO), volatile organic compounds (VOCs), and certain toxic

air pollutants. Particle pollution (PM2.5 and PM10) includes a mixture of compounds that can be grouped into five major categories: sulfate, nitrate, elemental (black) carbon, organic carbon and crustal material.

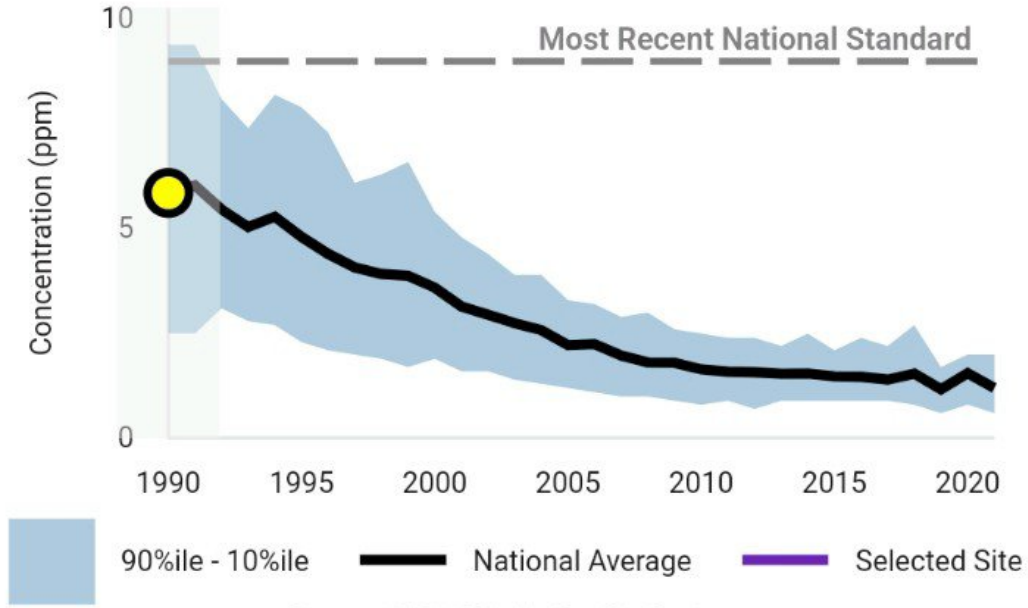
Some pollutants are released directly into the atmosphere while other pollutants are formed in the air from chemical reactions. Ground-level ozone forms when emissions of NOx and VOCs react in the presence of sunlight. Air pollution impacts human health and the environment through a variety of pathways.

National Ambient Air Quality Standards (NAAQS)

For more than 50 years, the Clean Air Act has brought Americans cleaner air and lower risks of adverse health effects.

Criteria Pollutant Trends Show Clean Air Progress

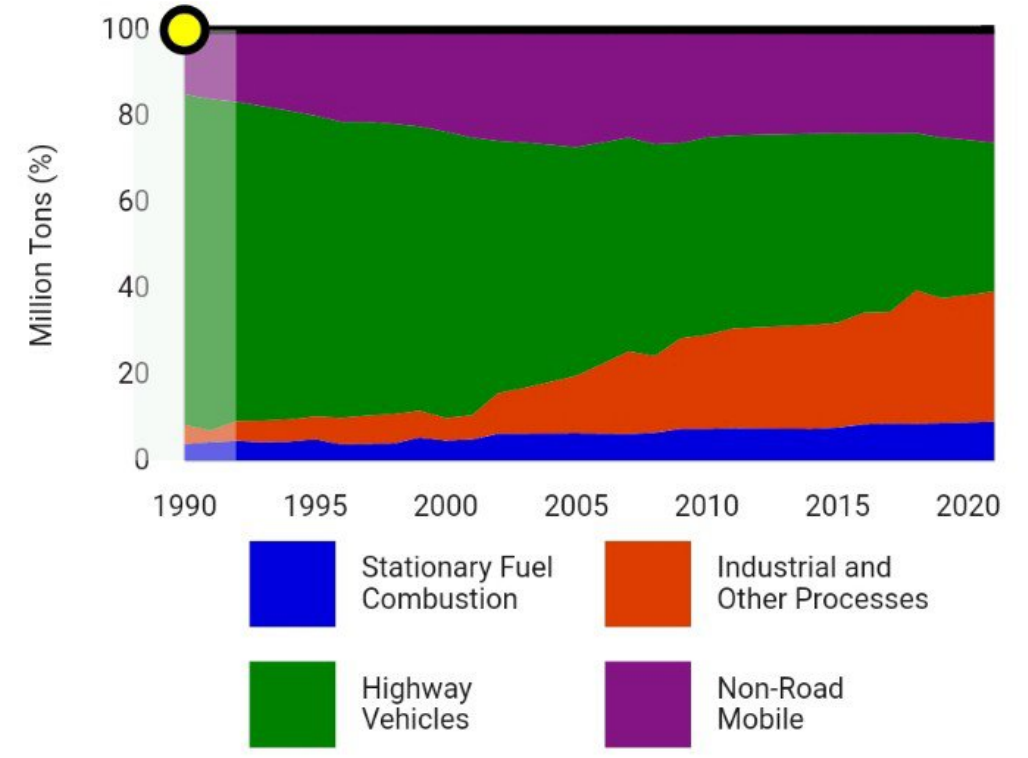
CO 8-hour Concentration



Source: U.S. EPA Air Quality System

CO Emissions

CO Emissions



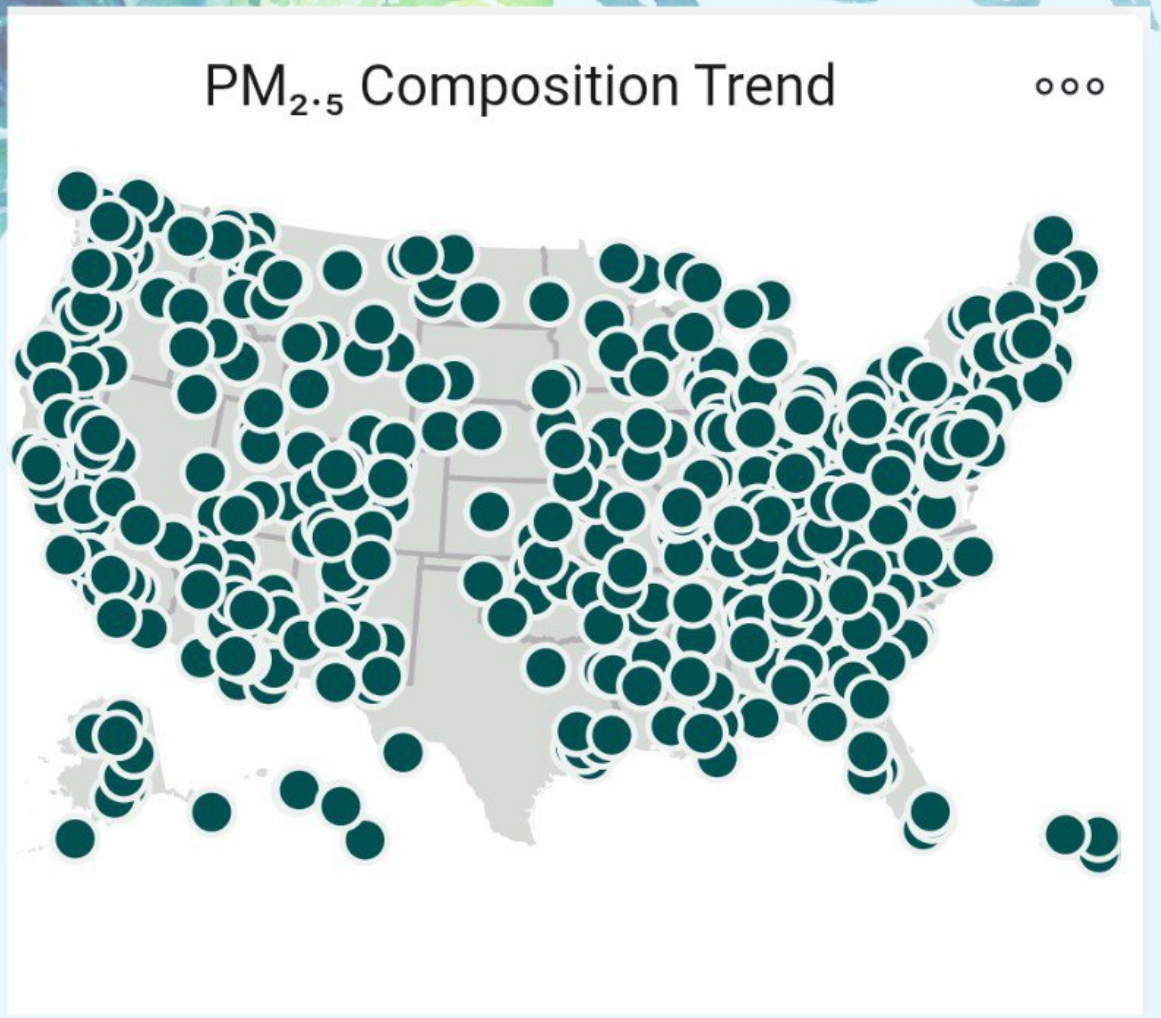
Source: U.S. EPA National Emissions Inventory 2017

Understanding PM2.5 Composition Helps Reduce Fine Particle Pollution

The different components that make up particle pollution come from specific sources and are often formed in the atmosphere. The major components, or species, are elemental carbon (EC), organic carbon (OC), sulfate and nitrate compounds, and crustal materials such as soil and ash.

Assessing particle pollution concentrations along with composition data aids in understanding the effectiveness of pollution controls and in quantifying the impacts to public health, regional visibility, ecology and climate.

Tip Click any point to display 2000-2020 annual and quarterly PM2.5 speciation trends, and select maximize to enlarge the chart. Double click the map to zoom in and click the home button to reset.



Air Toxics

Following the 1990 Clean Air Act Amendments, significant improvements in public health protection occurred as a result of reductions in air toxics emissions from large industrial facilities and transportation.

Air Toxics Levels Trending Down

Ambient monitoring data show that some of the toxic air pollutants, such as benzene, 1,3-butadiene and several metals, are declining at most sites.

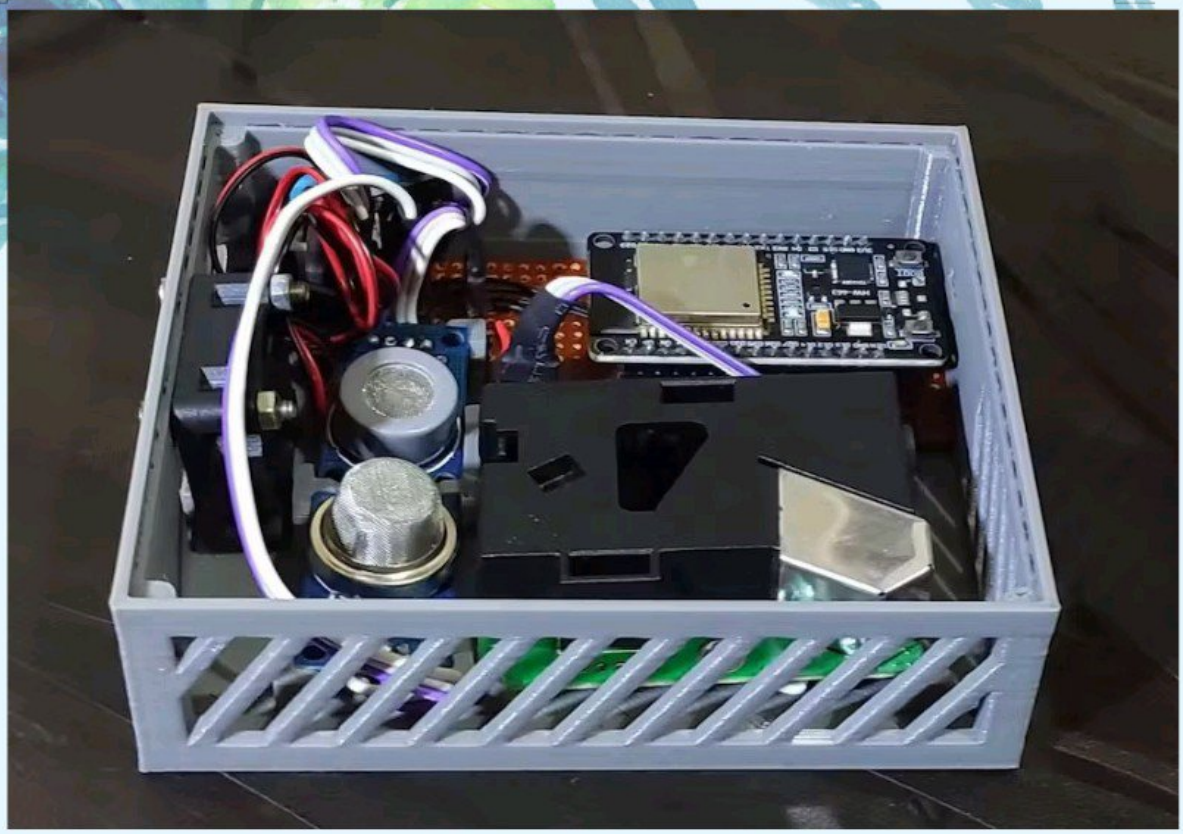
Points on the map indicate the long-term statistical trend direction: decreasing, increasing and no trend. There is insufficient data to determine a trend at sites depicted in gray.

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.

Air Quality Monitoring System (AQMS)

A compact and portable device for monitoring air quality in your surrounding, even for indoor air quality check.



AIR QUALITY MONITORING USING ESP32

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

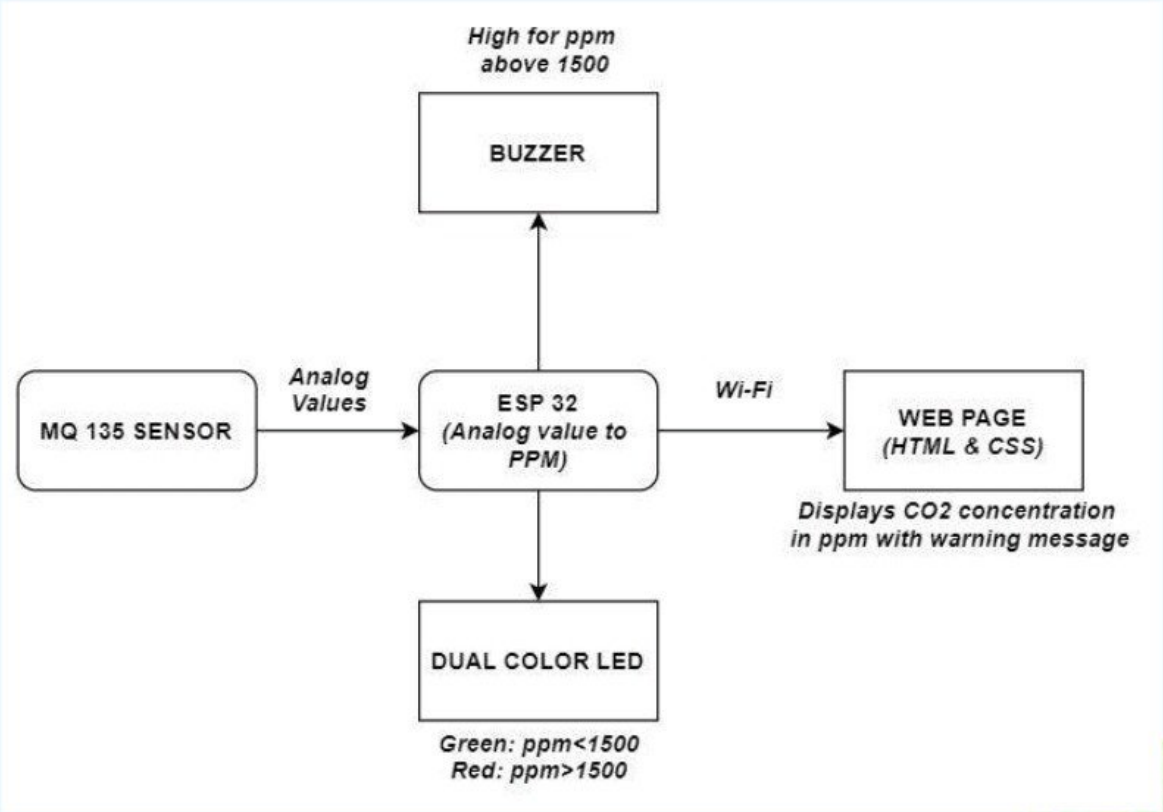
LEDs

Resistors

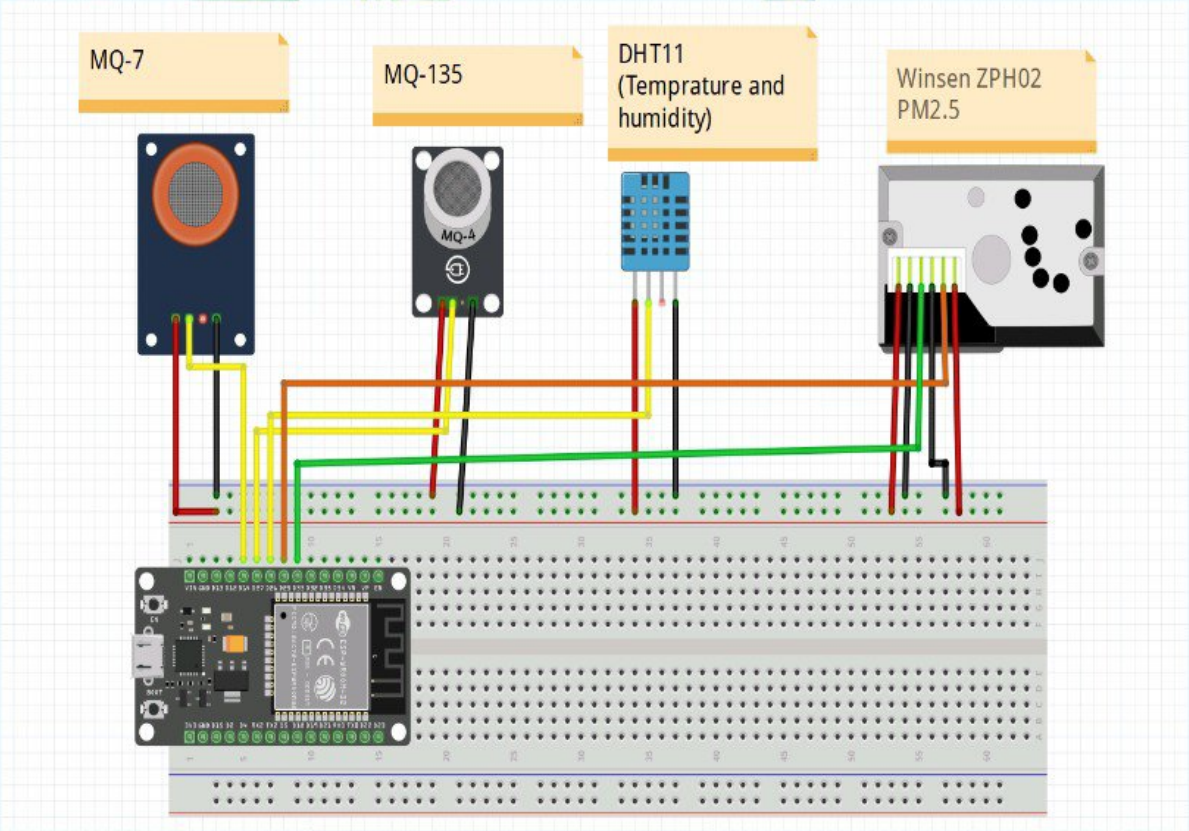
Buzzer

Some wires and a breadboard

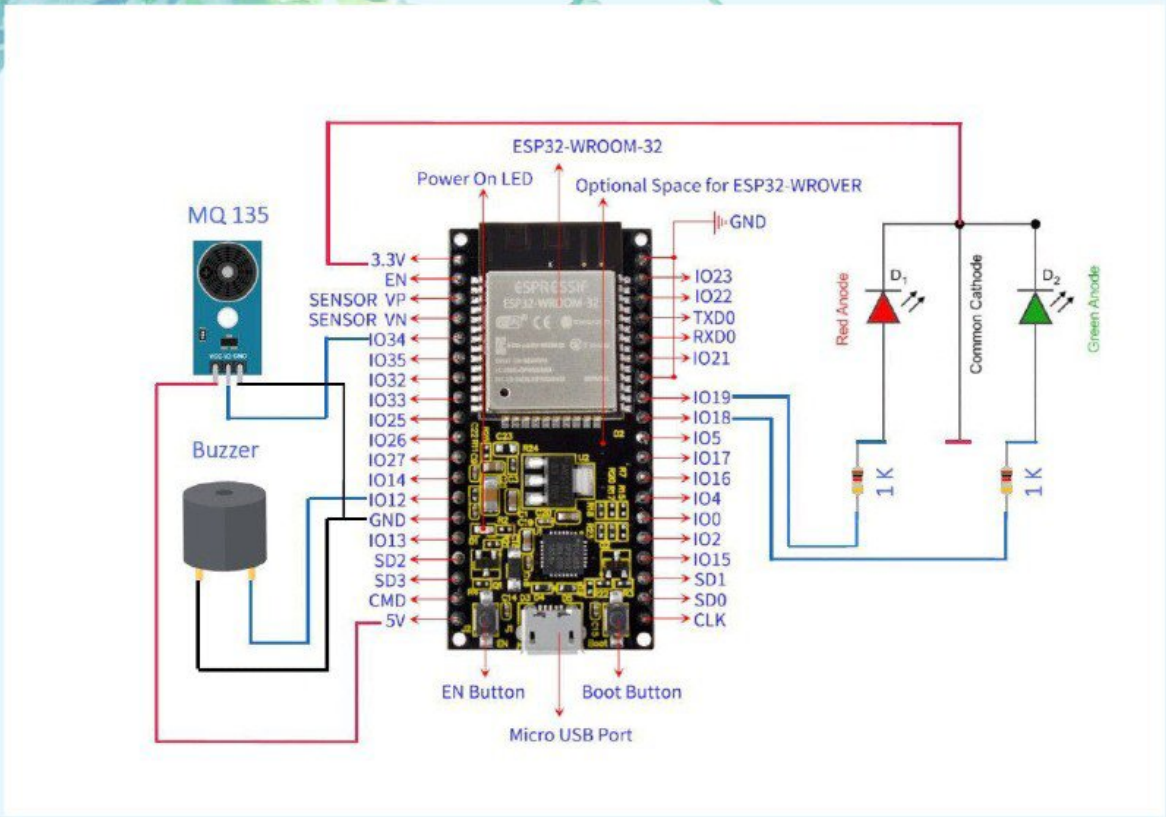
Block Diagram



CIRCUIT DIAGRAM



Circuitry



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.

Air Quality Index Calculation

The AQI in India is calculated based on the average concentration of a particular pollutant measured over a standard time interval (24 hours for most pollutants, 8 hours for carbon monoxide and ozone). For example, the AQI for PM2.5 and PM10 is based on 24-hour average concentration and AQI for Carbon Monoxide is based on 8-hour average concentration). The AQI calculations include the eight pollutants that are PM10, PM2.5, Nitrogen Dioxide (NO2), Sulphur Dioxide (SO2), Carbon Monoxide (CO), ground-level ozone (O3), Ammonia (NH3), and

Lead (Pb). However, all of the pollutants are not measured at every location.

Based on the measured 24-hour ambient concentrations of a pollutant, a sub-index is calculated, which is a linear function of concentration (e.g. the sub-index for PM2.5 will be 51 at concentration 31 µg/m3, 100 at concentration 60 µg/m3, and 75 at a concentration of 45 µg/m3). The worst sub-index (or maximum of all parameters) determines the overall AQI.

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(c);
```

```
// 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>FR
ESH AIR!</h2>");
                }
                else if (air_quality >800 &&
air_quality <2000)
                {

client.println("</style></head><body><h2>PO
OR AIR :(</h2>");
                }
```

```
                    else if (air_quality >= 2000)
                                                {

client.println("</style></head><body><h2>AL
ERT!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("");
    }

}
```

HARDWARE DESCRIPTION

Microcontroller:

The microcontroller we have used in this project is ESP32. ESP32 microcontroller has a powerful chip with a dual processing core. This makes it faster than the other controllers discussed above. It is a low-cost, low-power microcontroller and it has an in-built Wi-Fi module, on-chip Bluetooth Low Energy (BLE), good deep sleep modes. It supports 18 channels for 12-bit ADC and 2 channels for 8-bit DAC. All the incoming data from the sensors is processed by ESP32.

Sensing unit:

The sensing unit includes four sensors that are used for monitoring air quality.

MQ135:

MQ135 is an air quality sensor that detects a wide variety of gases like Ammonia (NH3),

Nitrogen Oxide (NOx), benzene, alcohol, smoke, Carbon dioxide, etc. For our project, we are measuring Carbon dioxide (CO2) with the help of this sensor. Its measuring range is from 10-1000 PPM. This sensor gives output in the form of voltage levels and hence it needs to be converted to PPM.
This can be done by writing appropriate code. Since it gives analog output, it is connected to the analog pin of ESP32.

MQ7:

MQ7 is a Carbon dioxide sensor. It detects the concentration of Carbon monoxide in the air and gives analog output. The sensor can measure concentrations from 10 to 10,000 PPM.

GP2Y1010AU0F:

GP2Y1010AU0F is a dust sensor that we have used for detecting fine particles like dust, cigarette smoke and it is very effective in

detecting it. Sensing is based on optical phenomenon.

The reflected light of dust in the air is detected by the sensor. It is easily able to differentiate smoke from house dust and this is done by the pulse pattern of output voltage.It is a low-power sensor. It is responsive to really low values. It responds in less than one second. It gives an analog output.

DHT11:

DHT11 is an extremely low power, low cost, ultra-small size digital temperature, and humidity sensor. It gives digital output. It can measure temperature from 0-50° C and humidity from 20-90% with an accuracy of ±1°C and ±1% respectively.

SOFTWARE DESCRIPTION

Arduino Integrated Development

Environment:

Arduino IDE is a publicly accessible software where one can easily write codes and upload them to the board. It makes code compilation very easy. Being a cross-platform application, codes can be written in C and C++ language for Windows, macOS, and Linux.

ThingSpeak:

ThingSpeak is an open-source IoT platform where one can combine, picture, and examine live streams of data in the cloud. Graphs, charts, numeric values of sensor data can be plotted on ThingSpeak.
The sensor data is stored and resolved over HTTP which works based on a request and response system.

METHODOLOGY

Our IoT-based air quality monitoring system is highly accurate, easy to use, and quite affordable.

GP2Y1010AU0F is a PM sensor connected to pin 36 of ESP32. MQ135 and MQ7 are connected to analog pin 32 and 34 of ESP32 respectively. DHT11 is connected to digital pin 2 of ESP32.

The buzzer is connected to digital pin 18 of ESP32. GP2Y1010AU0F has a hollow part at its center. So, whenever the fine particles are in the hollow detecting area, only then they can be sensed, and accordingly, appropriate values can be obtained.

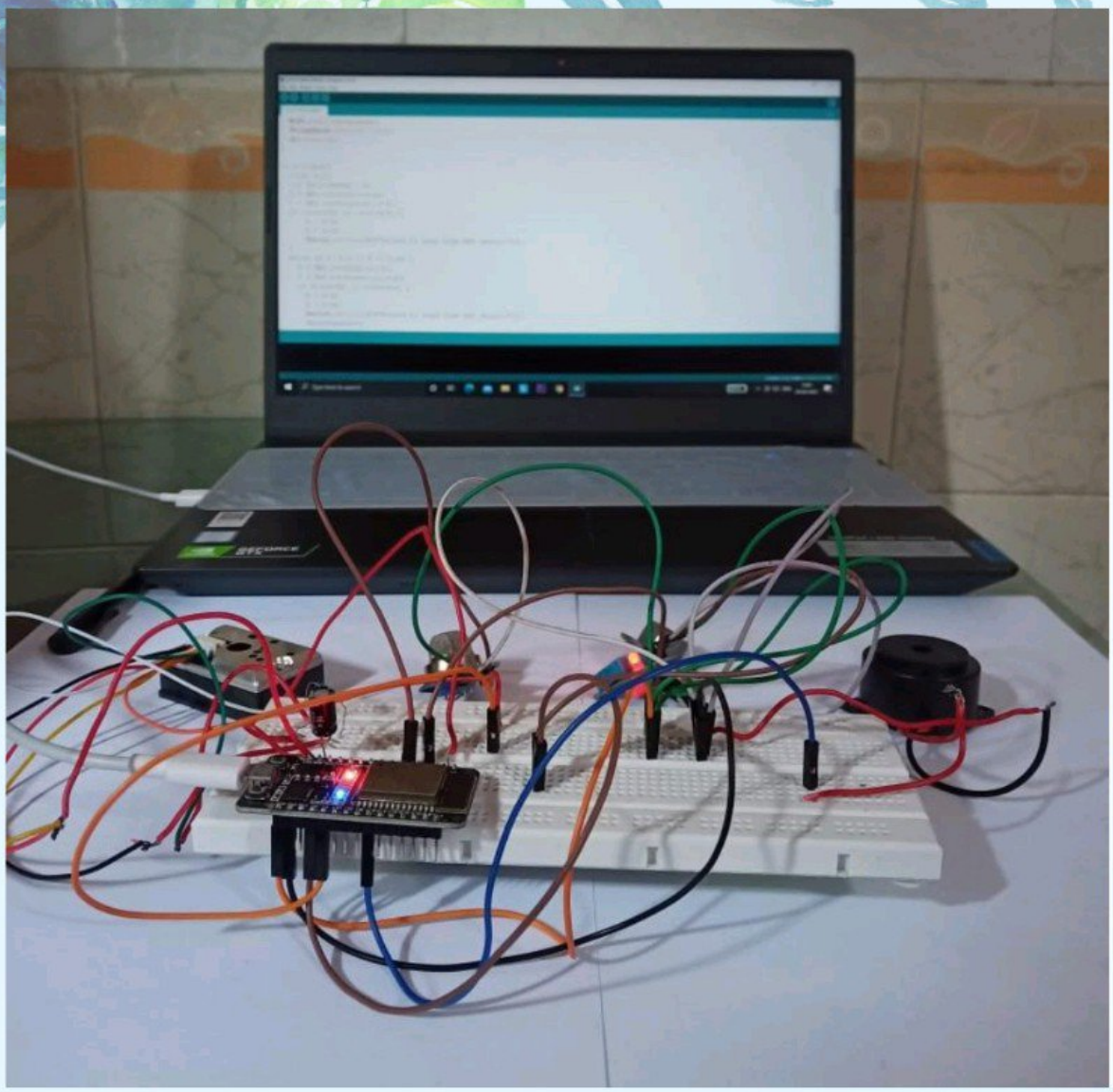
The output was calculated using the Dust density characteristics curve and is measured in mg/m3. Before starting to work with MQ135 and MQ7, they need to be preheated and then calibrated. Preheating means they need to be given a 5V power supply for 24 hours at least since they work on the heating principle.

Since they give output in voltage levels, with the help of their respective sensitivity characteristics curve they need to be converted to PPM. We have used MQ135 for measuring CO2 concentration. CO2 levels in the air are 250- 400 PPM: Normal. From 400 to 1000 PPM: Typical with good air exchange. More than 1000 PPM: Poor air.

As we also know, the normal range of CO2 in our environment should be around 390-450 PPM. As soon as CO2 concentration goes above 1000 PPM, the buzzer would start ringing. We have used MQ7 formeasuring CO concentration.

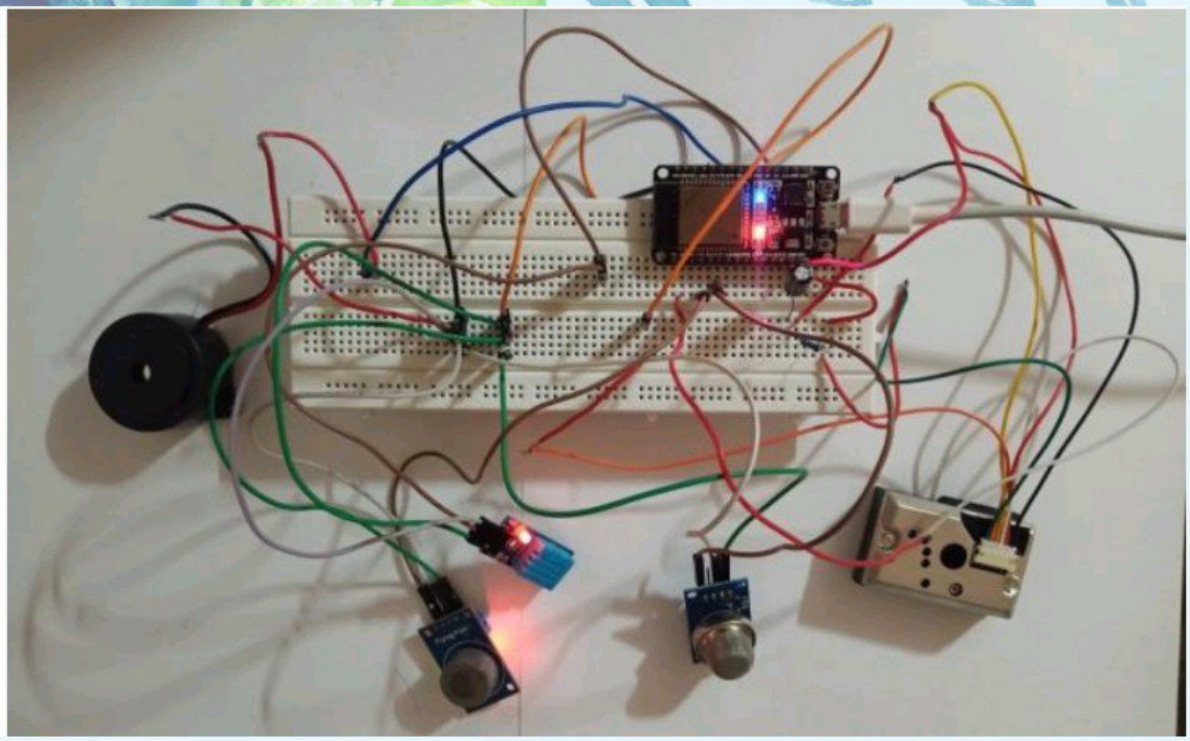
EXPERIMENTAL SETUP

In fig-2, the complete setup for the system consisting of sensors, ESP32, buzzer has been shown



Experimental setup

connection of sensors and buzzer toESP32 has been shown.



Connection

RESULTS

The obtained values of air quality were measured in a room of 300m² and displayed on ThingSpeak.

The X-axis of all the graphs represents date and time, whereas the Y-axis represents different parameters.

Chart -1: Temperature (°C) in atmosphere w.r.t time

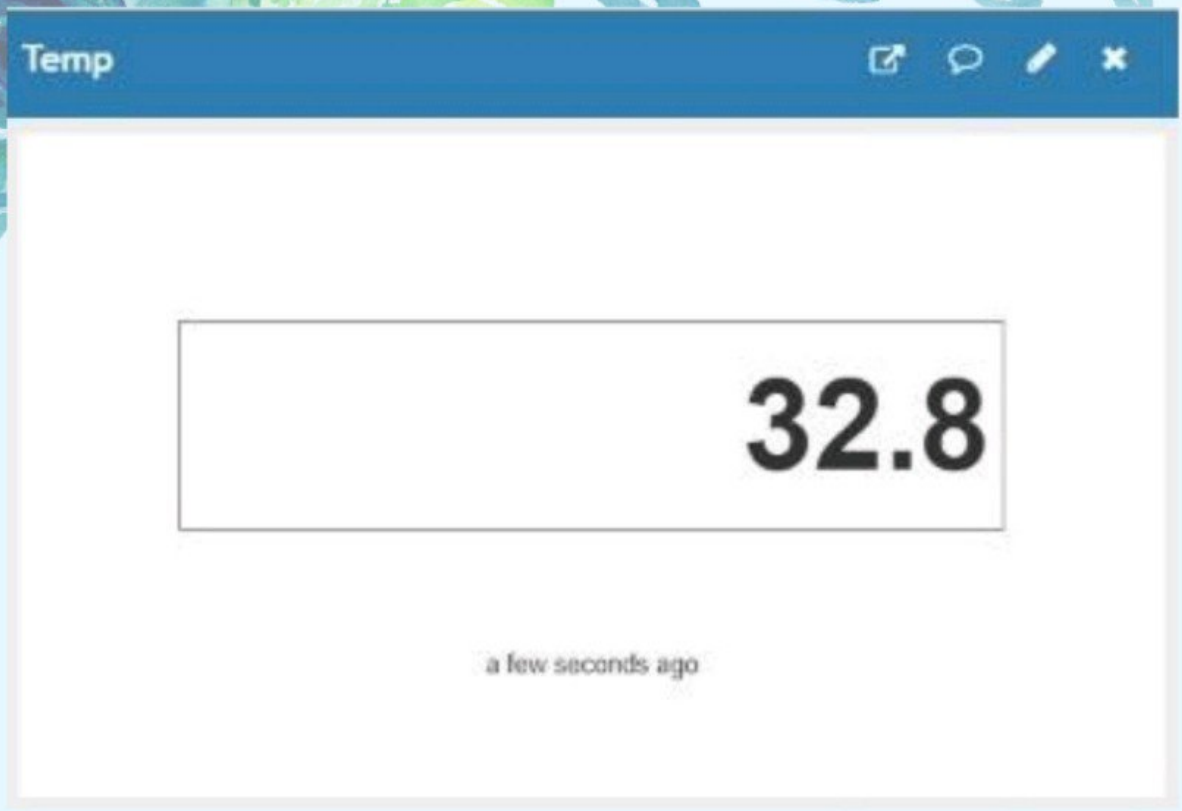


Chart -2: Temperature in °C (numeric display)

Chart -3: Humidity(%)inatmosphere w.r.t time

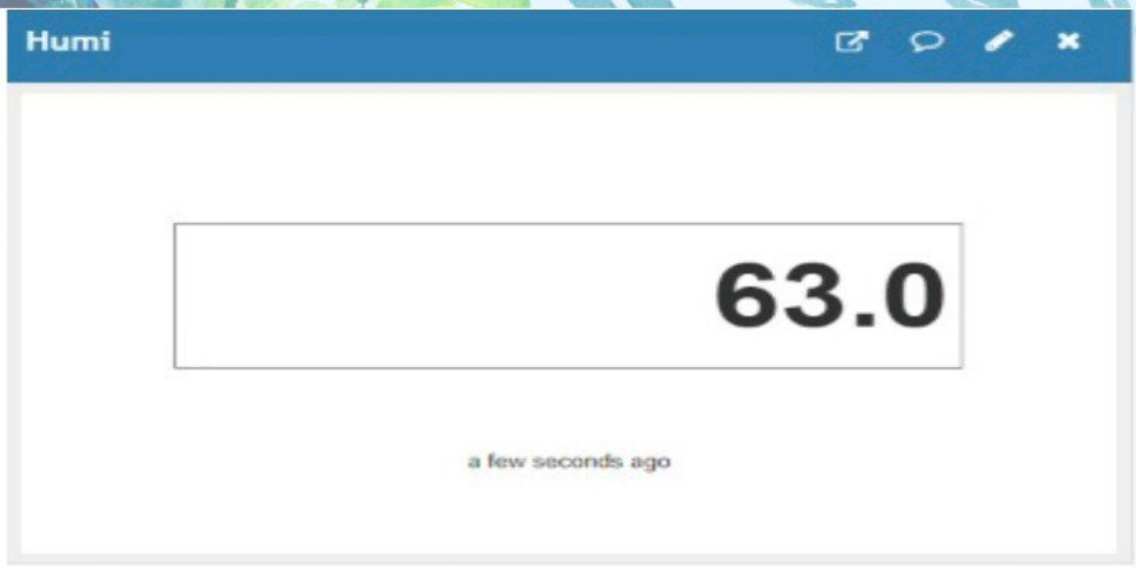


Chart -4: Humidity in % (numeric display)

Chart -5: CO2 content (PPM) in atmosphere w.r. time

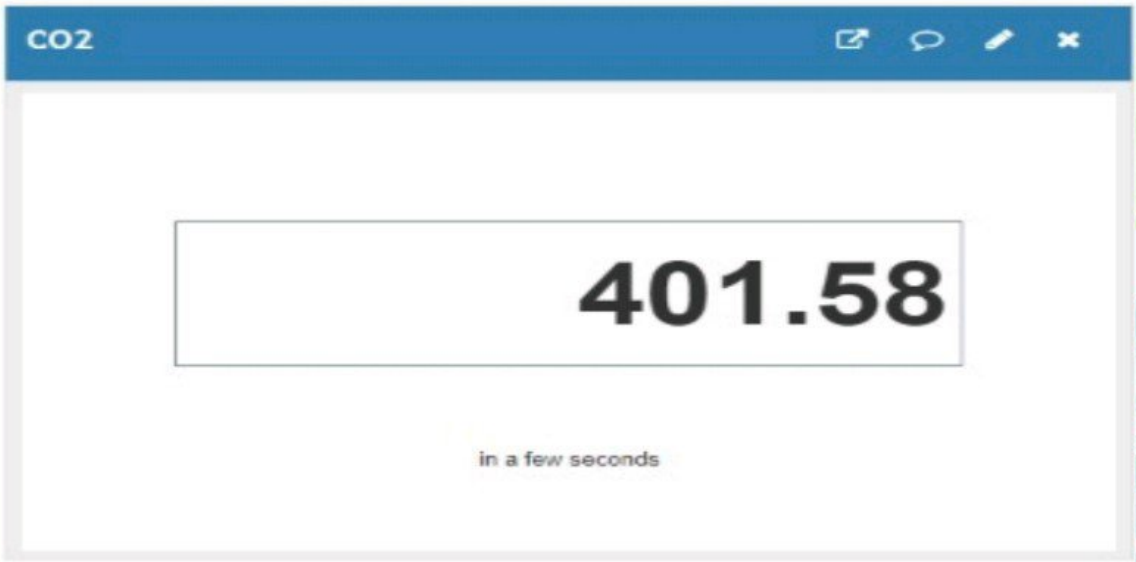


Chart -6: CO2 content in PPM (numeric display)

Chart -7: CO content (PPM) in atmosphere w.r.t time

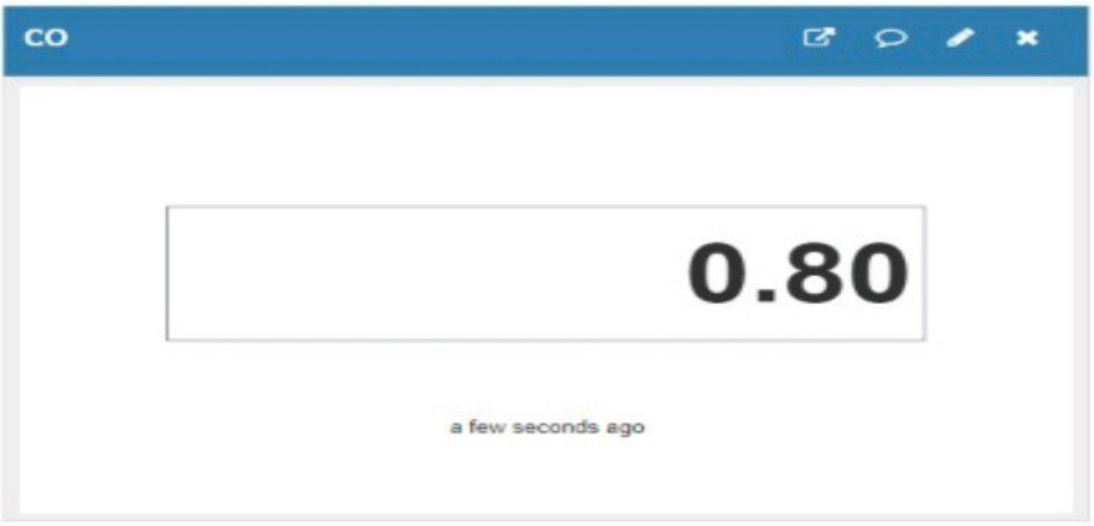


Chart -:8 CO content in PPM (numeric display)

Chart -9: Dust density (mg/m³) in atmosphere w.r.t time

Chart -10: Dust density in mg/m³ (gauge)

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.

Leveraging the concept of IoT, the air around the installed system can be monitored by anyone and from anywhere using a phone or a computer.

The continuous updating of data enables the users to take timely actions immediately whenever needed.

This helps in curbing air pollution in the environment around us which is a big concern. Apart from being low in cost and power consumption, it covers less space and can be installed anywhere.This provides great efficiency and flexibility.


THANK
YOU