**Air Pollution**

**Monitoring System**

**Documentation**

**Introduction:** The Air Pollution Monitoring System is designed to monitor the quality of air in a specific environment using Arduino Uno, ESP01, a 16\*2 LCD display, a buzzer, and an MQ-135 air quality sensor. The system measures the concentration of various air pollutants, including carbon monoxide (CO), nitrogen dioxide (NO2), and volatile organic compounds (VOCs), and provides real-time readings on the LCD display. It also triggers an audible alarm through the buzzer if pollution levels exceed predefined thresholds.

**Components used:**

* Arduino Uno
* ESP01 Wi-Fi module
* 16\*2 LCD display
* Buzzer
* MQ-135 air quality sensor
* Breadboard and jumper wires
* USB cable for programming Arduino Uno
* Power supply for Arduino Uno

**System Overview:** The Air Pollution Monitoring System consists of the following modules:

1. Arduino Uno: The main controller of the system, responsible for gathering sensor data, processing it, and controlling the output devices.
2. ESP01 Wi-Fi module: Enables the system to connect to a Wi-Fi network and transmit air quality data to a remote server or cloud platform for monitoring purposes.
3. 16\*2 LCD display: Provides a visual interface to display real-time air quality readings.
4. Buzzer: Produces an audible alarm if pollution levels exceed predefined thresholds.
5. MQ-135 Air Quality Sensor: Measures the concentration of various air pollutants, including CO, NO2, and VOCs.

**Circuit Diagram and Explanation:**

First of all we will connect the **ESP8266 with the Arduino**. ESP8266 runs on 3.3V and if you will give it 5V from the Arduino then it won’t work properly and it may get damage. Connect the VCC and the CH\_PD to the 3.3V pin of Arduino. The RX pin of ESP8266 works on 3.3V and it will not communicate with the Arduino when we will connect it directly to the Arduino. So, we will have to make a voltage divider for it which will convert the 5V into 3.3V. This can be done by connecting three resistors in series like we did in the circuit. Connect the TX pin of the ESP8266 to the pin 10 of the Arduino and the RX pin of the esp8266 to the pin 9 of Arduino through the resistors.

ESP8266 Wi-Fi module gives your projects **access to Wi-Fi or internet**. It is a very cheap device and make your projects very powerful. It can communicate with any microcontroller and it is the most leading devices in the [IOT platform](http://circuitdigest.com/internet-of-things-iot-projects). Learn more about using ESP8266 with Arduino here.

Then we will connect the **MQ135 sensor with the Arduino**. Connect the VCC and the ground pin of the sensor to the 5V and ground of the Arduino and the Analog pin of sensor to the A0 of the Arduino.

Connect a buzzer to the pin 8 of the Arduino which will start to beep when the condition becomes true.

In last, we will [connect LCD with the Arduino](http://circuitdigest.com/microcontroller-projects/arduino-lcd-interfacing-tutorial). The connections of the LCD are as follows

* Connect pin 1 (VEE) to the ground.
* Connect pin 2 (VDD or VCC) to the 5V.
* Connect pin 3 (V0) to the middle pin of the 10K potentiometer and connect the other two ends of the potentiometer to the VCC and the GND. The potentiometer is used to control the screen contrast of the LCD. Potentiometer of values other than 10K will work too.
* Connect pin 4 (RS) to the pin 12 of the Arduino.
* Connect pin 5 (Read/Write) to the ground of Arduino. This pin is not often used so we will connect it to the ground.
* Connect pin 6 (E) to the pin 11 of the Arduino. The RS and E pin are the control pins which are used to send data and characters.
* The following four pins are data pins which are used to communicate with the Arduino.

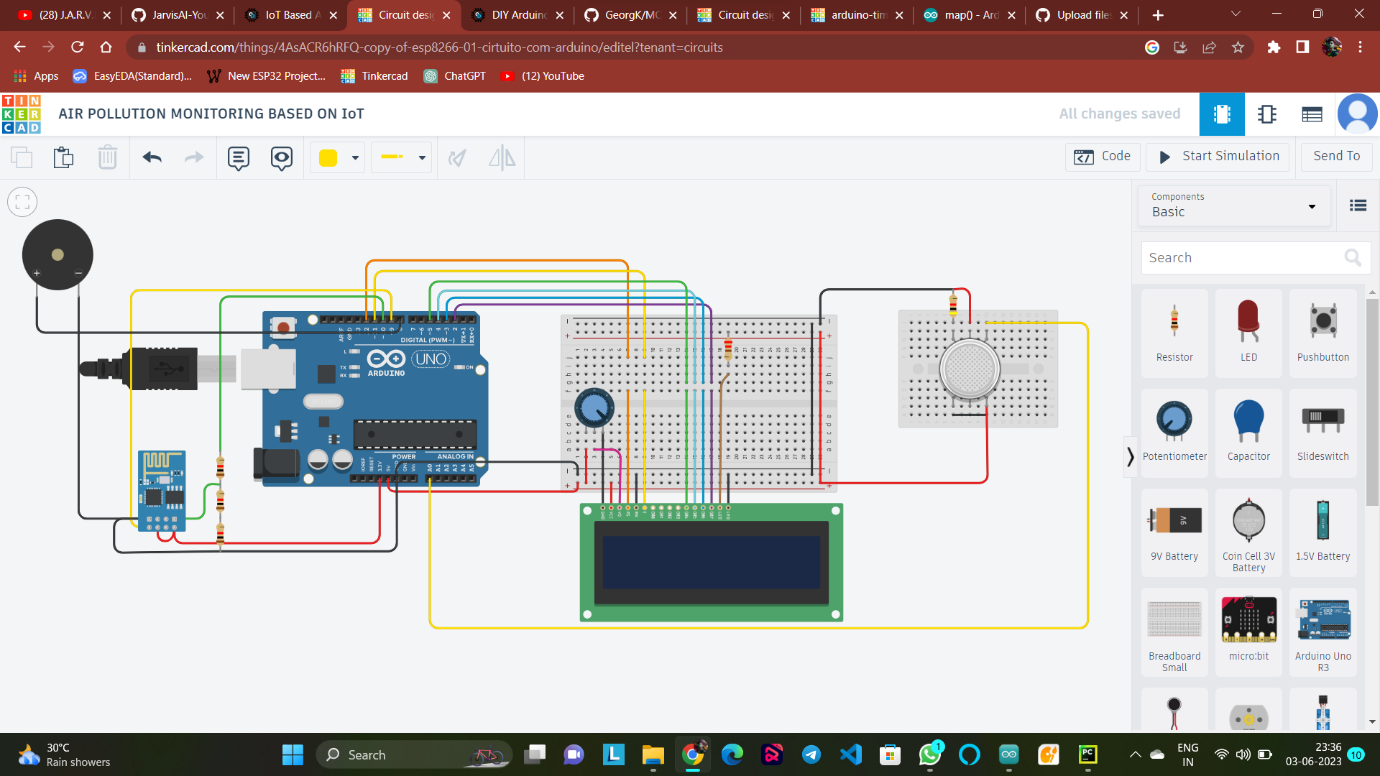
Connect pin 11 (D4) to pin 5 of Arduino.

Connect pin 12 (D5) to pin 4 of Arduino.

Connect pin 13 (D6) to pin 3 of Arduino.

Connect pin 14 (D7) to pin 2 of Arduino.

* Connect pin 15 to the VCC through the 220 ohm resistor. The resistor will be used to set the back light brightness. Larger values will make the back light much more darker.
* Connect pin 16 to the Ground.

[](https://circuitdigest.com/fullimage?i=circuitdiagram_mic/Iot-air-quality-monitoring-system-using-arduino-circuit.jpg)

**Working Explanation:**

The MQ135 sensor can sense NH3, NOx, alcohol, Benzene, smoke, CO2 and some other gases, so it is perfect gas sensor for our **Air Quality Monitoring Project**. When we will connect it to Arduino then it will sense the gases, and we will get the Pollution level in PPM (parts per million). MQ135 gas sensor gives the output in form of voltage levels and we need to convert it into PPM. So for converting the output in PPM, here we have used a library for MQ135 sensor, it is explained in detail in “Code Explanation” section below.

Sensor was giving us value of 90 when there was no gas near it and the safe level of air quality is 350 PPM and it should not exceed 1000 PPM. When it exceeds the limit of 1000 PPM, then it starts cause Headaches, sleepiness and stagnant, stale, stuffy air and if exceeds beyond 2000 PPM then it can cause increased heart rate and many other diseases.

When the value will be less than 1000 PPM, then the LCD and webpage will display “Fresh Air”.  Whenever the value will increase 1000 PPM, then the buzzer will start beeping and the LCD and webpage will display “Poor Air, Open Windows”. If it will increase 2000 then the buzzer will keep beeping and the LCD and webpage will display “Danger! Move to fresh Air”.

Code Explanation:

Before beginning the coding for this project, we need to first Calibrate the MQ135 Gas sensor. There are lots of calculations involved in converting the output of sensor into PPM value, we have done this calculation before in our previous [Smoke Detector project](http://circuitdigest.com/microcontroller-projects/arduino-smoke-detector-on-pcb-using-mq2-gas-sensor). But here we are using the Library for MQ135, you can download and install this MQ135 library from here: <https://github.com/GeorgK/MQ135>.

Using this library you can directly get the PPM values, by just using the below two lines:

MQ135 gasSensor = MQ135(A0);

float air\_quality = gasSensor.getPPM();

But before that we need to **calibrate the MQ135 sensor**, for calibrating the sensor upload the below given code and let it run for 12 to 24 hours and then get the *RZERO* value.

#include "MQ135.h"

void setup (){

Serial.begin (9600);

}

void loop() {

MQ135 gasSensor = MQ135(A0); // Attach sensor to pin A0

float rzero = gasSensor.getRZero();

Serial.println (rzero);

delay(1000);

}

After getting the *RZERO* value. **Put the RZERO value in the library file** you downloaded "MQ135.h": *#define RZERO 494.63*

Now we can begin the actual code for our Air quality monitoring project.

In the code, first of all we have defined the libraries and the variables for the Gas sensor and the LCD. By using the Software Serial Library, we can make any digital pin as TX and RX pin. In this code, we have made Pin 9 as the RX pin and the pin 10 as the TX pin for the ESP8266. Then we have included the library for the LCD and have defined the pins for the same. We have also defined two more variables: one for the sensor analog pin and other for storing *air\_quality* value.

#include <SoftwareSerial.h>

#define DEBUG true

SoftwareSerial esp8266(9,10);

#include <LiquidCrystal.h>

LiquidCrystal lcd(12,11, 5, 4, 3, 2);

const int sensorPin= 0;

int air\_quality;

Then we will declare the pin 8 as the output pin where we have connected the buzzer. l*cd.begin(16,2)* command will start the LCD to receive data and then we will set the cursor to first line and will print the *‘circuitdigest’*. Then we will set the cursor on the second line and will print *‘Sensor Warming’*.

pinMode(8, OUTPUT);

lcd.begin(16,2);

lcd.setCursor (0,0);

lcd.print ("Air pollution monitoring");

lcd.setCursor (0,1);

lcd.print ("by KISHAN & NIDHI");

delay(1000);

Then we will set the baud rate for the serial communication. Different ESP’s have different baud rates so write it according to your ESP’s baud rate. Then we will send the commands to set the ESP to communicate with the Arduino and show the IP address on the serial monitor.

Serial.begin(115200);

esp8266.begin(115200);

sendData("AT+RST\r\n",2000,DEBUG);

sendData("AT+CWMODE=2\r\n",1000,DEBUG);

sendData("AT+CIFSR\r\n",1000,DEBUG);

sendData("AT+CIPMUair\_quality=1\r\n",1000,DEBUG);

sendData("AT+CIPSERVER=1,80\r\n",1000,DEBUG);

pinMode(sensorPin, INPUT);

lcd.clear();

For [printing the output on the webpage](http://circuitdigest.com/microcontroller-projects/sending-arduino-data-to-webpage) in web browser, we will have to use **HTML programming**. So, we have created a string named *webpage* and stored the output in it. We are subtracting 48 from the output because the *read()*function returns the ASCII decimal value and the first decimal number which is 0 starts at 48.

if(esp8266.available())

{

if(esp8266.find("+IPD,"))

{

delay(1000);

int connectionId = esp8266.read()-48;

String webpage = "<h1>IOT Air Pollution Monitoring System</h1>";

webpage += "<p><h2>";

webpage+= " Air Quality is ";

webpage+= air\_quality;

webpage+=" PPM";

webpage += "<p>";

The following code will call a function named*sendData* and will send the data & message strings to the webpage to show.

sendData(cipSend,1000,DEBUG);

sendData(webpage,1000,DEBUG);

cipSend = "AT+CIPSEND=";

cipSend += connectionId;

cipSend += ",";

cipSend +=webpage.length();

cipSend +="\r\n";

The following code will print the data on the LCD. We have applied various conditions for checking air quality, and LCD will print the messages according to conditions and buzzer will also beep if the pollution goes beyond 1000 PPM.

lcd.setCursor (0, 0);

lcd.print ("Air Quality is ");

lcd.print (air\_quality);

lcd.print (" PPM ");

lcd.setCursor (0,1);

if (air\_quality<=1000)

{

lcd.print("Fresh Air");

digitalWrite(8, LOW);

Finally the below function will send and show the data on the webpage. The data we stored in string named *‘webpage’* will be saved in string named *‘command’*. The ESP will then read the character one by one from the *‘command’* and will print it on the webpage.

String sendData(String command, const int timeout, boolean debug)

{

String response = "";

esp8266.print(command); // send the read character to the esp8266

long int time = millis();

while( (time+timeout) > millis())

{

while(esp8266.available())

{

// The esp has data so display its output to the serial window

char c = esp8266.read(); // read the next character.

response+=c;

}

}

if(debug)

{

Serial.print(response);

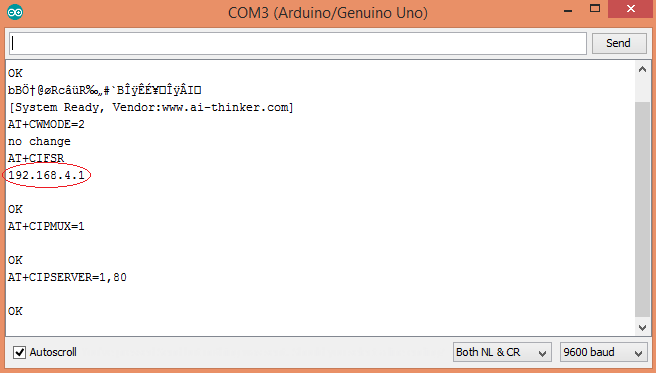
}

return response;

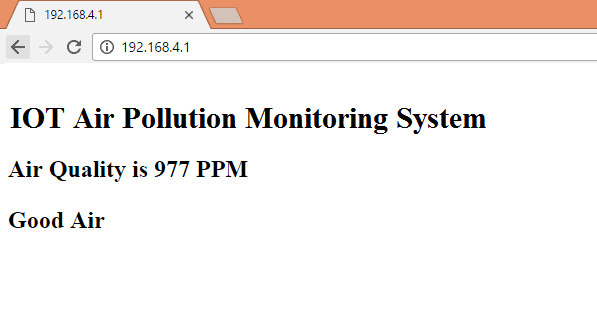
}

Testing and Output of the Project:

Before uploading the code, make sure that you are connected to the Wi-Fi of your ESP8266 device. After uploading, open the serial monitor and it will show the IP address like shown below.



Type this IP address in your browser, it will show you the output as shown below. You will have to refresh the page again if you want to see the current Air Quality Value in PPM.



We have setup a local server to demonstrate its working, you can check the **Video** below. But to monitor the air quality from anywhere in the world, you need to **forward the port 80 (used for HTTP or internet) to your local or private IP address** (192.168\*) of you device. After port forwarding all the incoming connections will be forwarded to this local address and you can open above shown webpage by just entering the public IP address of your internet from anywhere. You can forward the port by logging into your router (192.168.1.1) and find the option to setup the port forwarding.

System Operation: The Air Pollution Monitoring System follows the following operational steps:

1. Sensor Calibration: Calibrate the MQ-135 air quality sensor according to the manufacturer's instructions before deploying the system. This ensures accurate readings.
2. Hardware Setup: Connect the Arduino Uno, ESP01, LCD display, buzzer, and MQ-135 sensor using the breadboard and jumper wires. Follow the pin configuration guidelines provided by the sensor and display modules.
3. Arduino Programming: Use the Arduino IDE or any compatible programming environment to write the firmware code for the Arduino Uno. The code should include the necessary libraries for sensor data acquisition, LCD control, and Wi-Fi communication.
4. Sensor Data Acquisition: Continuously read sensor values using appropriate analog pins of the Arduino Uno. The MQ-135 sensor usually provides analog output, which can be connected to an analog pin of the Arduino Uno.
5. Data Processing and Display: Process the sensor data to obtain the concentration values for CO, NO2, and VOCs. Use appropriate formulas or calibration curves provided by the sensor manufacturer. Display the readings on the LCD display using the LCD library.
6. Threshold Monitoring: Set predefined thresholds for each pollutant. Compare the current readings with the thresholds and trigger the buzzer if any pollution level exceeds the threshold.
7. Wi-Fi Communication: Configure the ESP01 module to connect to a Wi-Fi network. Use appropriate libraries and protocols (e.g., HTTP or MQTT) to transmit the air quality data to a remote server or cloud platform for monitoring and analysis.

Conclusion: The Air Pollution Monitoring System provides a cost-effective solution for monitoring air quality in a specific environment. By combining Arduino Uno, ESP01, a 16\*2 LCD display, a buzzer, and an MQ-135 sensor, the system can measure air pollutants and provide real-time readings and alerts. This documentation serves as a guide for setting up and operating the system effectively.