Building the IOT Air Quality Monitoring

INTRODUCTION.

In this project we are going to make an IoT Based Air Pollution Monitoring System in which we will monitor the Air Quality over a webserver using internet and will trigger a alarm when the air quality goes down beyond a certain level, means when there are sufficient amount of harmful gases are present in the air like CO2, smoke, alcohol, benzene and NH3. It will show the air quality in PPM on the LCD and as well as on webpage so that we can monitor it very easily.

Previously we have built the LPG detector using MQ6 sensor, Smoke detector using MQ2 sensor, and Air Quality Analyser but this time we have used MQ135 sensor as the air quality sensor which is the best choice for monitoring Air Quality as it can detects most harmful gases and can measure their amount accurately. In this IOT project, you can monitor the pollution level from anywhere using your computer or mobile. We can install this system anywhere and can also trigger some device when pollution goes beyond some level, like we can switch on the Exhaust fan or can send alert SMS/mail to the user.

REQUIRED COMPONENTS:

MQ135 Gas sensor
Arduino Uno
Wi-Fi module ESP8266
16X2 LCD
Breadboard
10K potentiometer

1K ohm resistors
220 ohm resistor
Buzzer

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. 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.

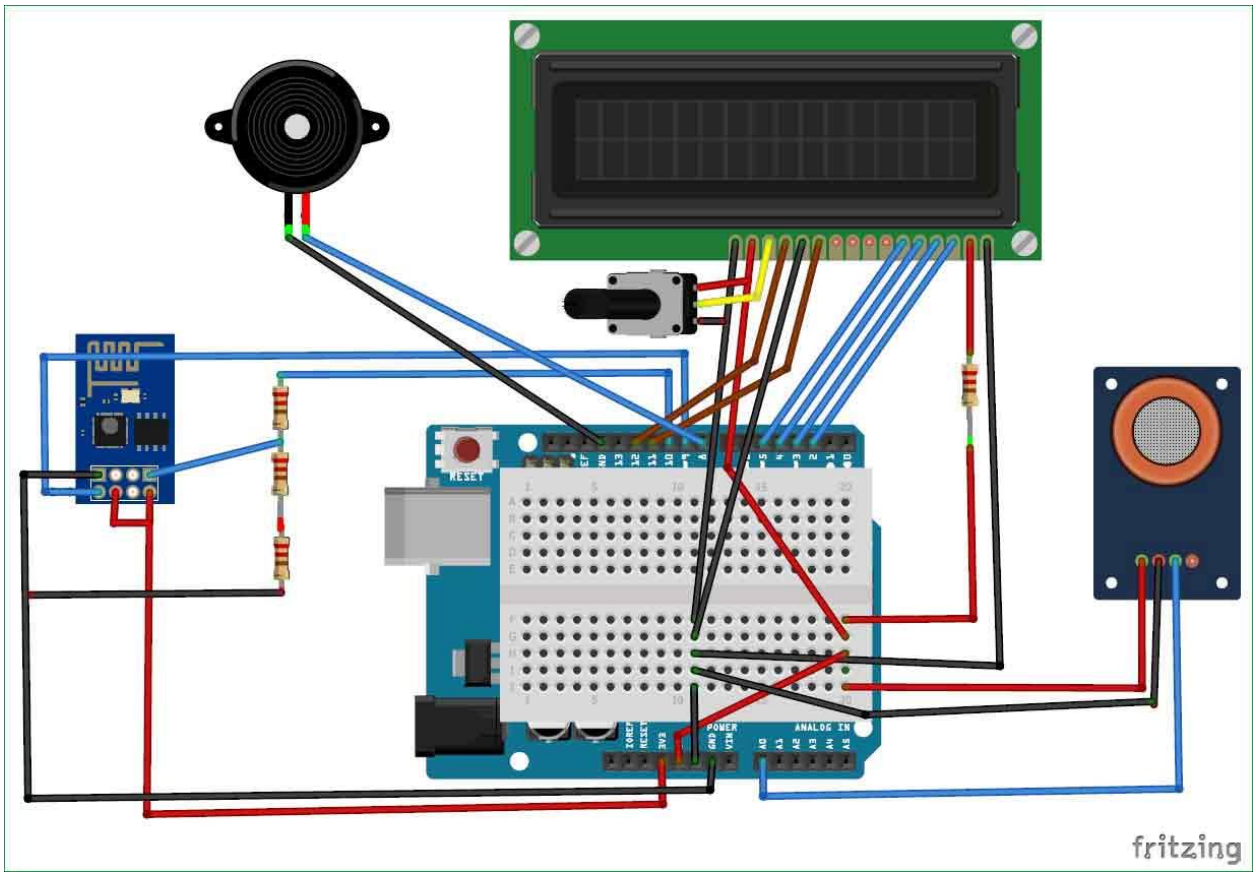
- Connect a buzzer to the pin 8 of the Arduino which will start to beep when the condition becomes true.

In last, we will connect LCD with the Arduino. 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.



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.

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.

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

```
Read air quality data from the sensor
  Air_quality_data = read_air_quality_data()

  # Prepare the data for transmission (replace with your sensor data)
  Data_to_send = {
    “timestamp”: time.time(),
    “co2_level”: air_quality_data[“co2”],
    “pm25_level”: air_quality_data[“pm25”],
    # Add more sensor data as needed
  }

  Try:
    # Send the data to the server
    Response = requests.post(server_url, json=data_to_send)
    If response.status_code == 200:
      Print(“Data sent successfully”)
    Else:
      Print(“Failed to send data”)
(MEC): Except Exception as e:
    Print(“Error:”, e)

  # Data transmission interval (adjust as needed)
```

Time.sleep(60) # Send data every minute

importing pandas module for data frame

Import pandas as pd

loading dataset and storing in train variable

Train=pd.read_csv('AQI.csv')

# display top 5 data
Train.head()

	PM2.5-AVG	PM10-AVG	NO2-AVG	NH3-AVG	SO2-AG	CO	OZONE-AVG	air_quality_index
0	190	131	107	4	42	0	63	190
1	188	131	110	4	40	0	62	188
2	280	174	155	2	37	0	52	280
3	302	181	144	2	39	0	78	302
4	285	160	121	3	19	0	71	285

A.AKKILA