**AIR QUALITY MONITORING**

**PHASE-3 DOCUMENT SUBMISSION**

**TEAM MEMBERS :**

**DHINESH M**

**LAL PRASATH L**

**SABARESHWARAN G**

**BALA ARUN R**

**PROJECT:** AIR QUALITY MONITORING

**INTRODUCTION**

Air quality monitoring means checking the air to see if it's clean or polluted. We use special machines to measure things in the air, like dust, gases, and tiny particles. This helps us know if the air is safe to breathe. If it's not safe, we can take steps to make it cleaner and protect our health. Air quality monitoring is important because clean air is better for us and the environment

**Iot-Enabled Air Quality Monitoring**

IoT-enabled air quality monitoring is like having smart devices that can continuously check and report the quality of the air around us. These devices use the Internet to send data to a central place where it can be monitored and analyzed.

Imagine having a small, smart box in your neighborhood or inside your home that can measure things like dust, smoke, gases, and even pollen in the air. This box collects this information and shares it online. With your smartphone or computer, you can see how clean or polluted the air is in real-time.

This kind of monitoring helps us know if the air we're breathing is healthy or if there are problems. It's like having a personal air quality watchdog to make sure we're safe and our environment is clean. This data can also help governments and organizations make decisions to improve air quality and protect our health.

**OBJECTIVES**

Creating a simple air quality monitoring project involves setting clear objectives to ensure you achieve your goals effectively. Here are some straightforward objectives for a basic air quality monitoring project:

**Real-Time Monitoring:** Develop a system that provides real-time data on air quality. Users should be able to access current air quality information easily.

**Key Air Quality Parameters:** Monitor essential air quality parameter such as particulate matter (PM2.5 and PM10), carbon dioxide (CO2), carbon monoxide (CO), ozone (O3), nitrogen dioxide (NO2), and sulfur dioxide (SO2).

**Data Collection and Storage:** Collect and store air quality data in a structured database for analysis and historical tracking.

**User-Friendly Interface:** Create a user-friendly interface, which can be a web dashboard or a mobile app, to display air quality data graphically for easy interpretation.

**Alerting and Notifications:** Implement a system that sends alerts or notifications when air quality parameters exceed predefined thresholds. These alerts could be via email, SMS, or push notifications.

**Data Visualization:** Provide visualization tools such as graphs and charts to help users understand air quality trends and changes over time.

**Geospatial Information:** If applicable, include geospatial information to show air quality variations in different locations, such as maps with color-coded regions.

**Historical Data Analysis**: Enable users to access historical air quality data for trend analysis, comparisons, and research purposes.

**Calibration and Maintenance**: Ensure the system is calibrated regularly to maintain data accuracy, and have a maintenance plan in place to keep the sensors and devices functioning optimally

**Public Awareness:** If the project is for a community or educational purposes, consider integrating an educational component to raise awareness about air quality issues and their impact on health

**Cost-Effective Solution:** Strive to keep the project cost-effective, especially if it's for a non-commercial or educational purpose. Look for affordable sensors and platforms that still provide reliable data.

**Open Data Sharing:** If possible, allow for data sharing with the public or other research institutions to contribute to larger air quality studies.

**Sustainability:** Ensure the project's sustainability by establishing a plan for funding, sensor maintenance, and system updates.

**Compliance with Standards:** Make sure the project complies with relevant air quality monitoring standards and guidelines, if applicable.

**User Feedback:** Gather user feedback and make improvements based on their input to enhance the project's functionality and usability.

**Scalability:** Design the system in a way that it can be scaled up or expanded in the future, should the need arise.

Remember that the objectives can vary depending on the scale and scope of your project. Start with a basic setup, and then consider expanding and adding more features as you gain experience and resources.

窗体底端

### 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](http://circuitdigest.com/internet-of-things-iot-projects).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.

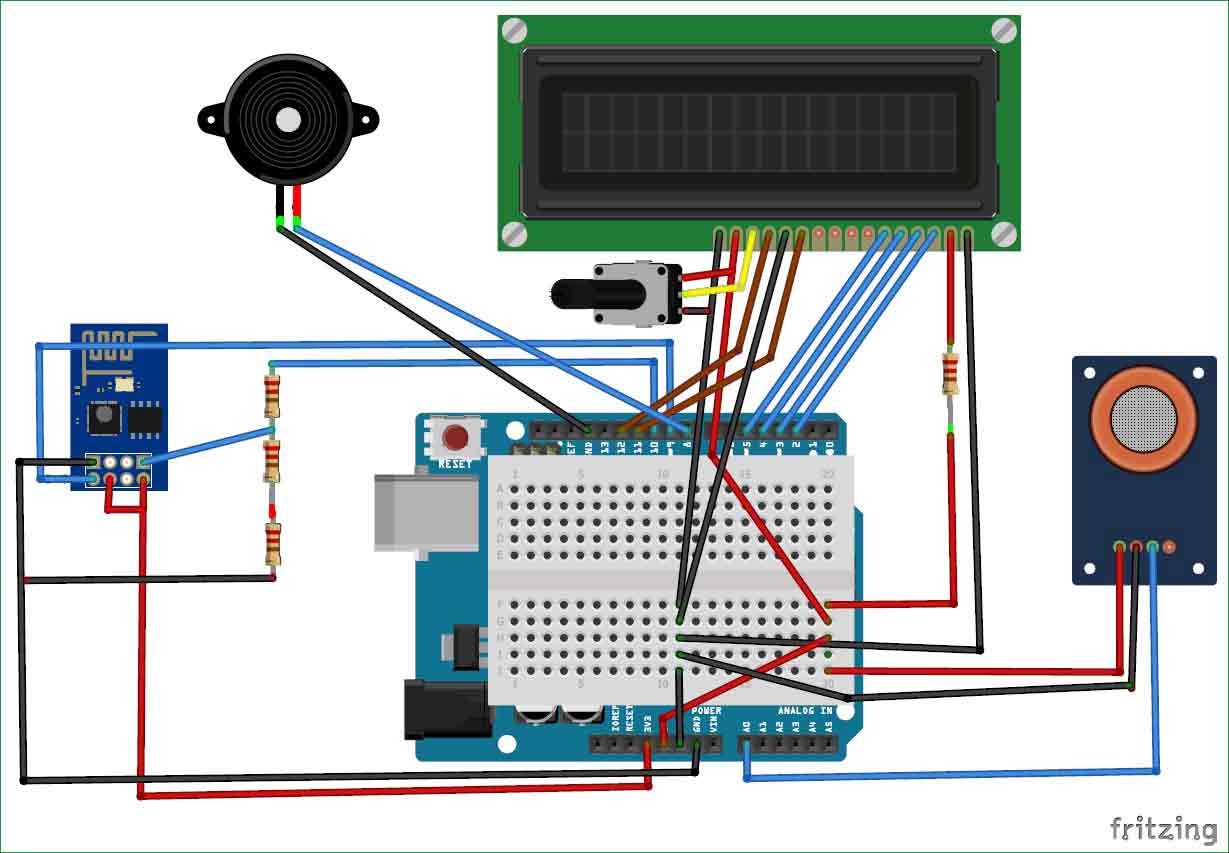
Connect pin 11 (D4) to pin 5 of Arduino.

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.



### 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”.

**Arduino code for data collection.**

#include <MQ135.h>

// Define the pin where the MQ135 sensor is connected

const int MQ135Pin = A0;

// Create an instance of the MQ135 sensor

MQ135 gasSensor = MQ135(MQ135Pin);

void setup() {

Serial.begin(9600);

}

void loop() {

// Read the sensor value

float rZero = gasSensor.getRZero();

float ppm = gasSensor.getPPM();

// Print the sensor values

Serial.print("RZero: ");

Serial.println(rZero);

Serial.print("PPM: ");

Serial.println(ppm);

delay(2000); // Delay between readings, adjust as needed

}

**DATA COLLECTION:**

The code you provided is intended for reading data from an MQ135 air quality sensor, and it calculates and prints two values: "RZero" and "PPM" (Parts Per Million). However, the code snippet you shared doesn't include a dataset because it's designed to collect real-time data from the MQ135 sensor when you run it on an Arduino.

If you want to collect a dataset of air quality data using this code, you would need to run the code on your Arduino with the MQ135 sensor connected, and the Arduino would continuously collect data and output it to the serial monitor. You can then log the data manually or automate the data logging process using software on your computer.

To create a dataset, you can follow these steps:

1. Set up your Arduino and MQ135 sensor as per the code you provided.

2. Upload the code to your Arduino using the Arduino IDE.

3. Open the Arduino IDE's Serial Monitor (Tools > Serial Monitor) to view the "RZero" and "PPM" values as they are printed.

4. Collect data over a specified period by allowing the Arduino to run the code.

5. Manually record the values, or use software (e.g., Arduino's Serial Plotter or external data logging software) to capture and save the data.

You can adjust the `delay(2000)` in the code to control the time interval between data readings. This allows you to control how frequently data points are collected.

Please note that if you want to collect a large dataset automatically, you may need to modify the code to log the data to an SD card or transmit it to a computer via a communication protocol like UART or Wi-Fi.

Calibration and testing procedures

Calibrating and testing an air quality monitoring system is crucial to ensure accurate and reliable measurements. Calibration adjusts the sensor's output to match reference measurements, and testing verifies the system's performance. Here's a step-by-step guide on how to calibrate and test an air quality monitoring system:

**1. Equipment and Environment Setup:** Ensure you have the necessary equipment, including the air quality sensors, reference measurement devices (if available), an Arduino or microcontroller, and a computer for data analysis.

- Set up the monitoring system in an environment representative of the conditions where it will be used.

**2. Gather Reference Data (If Available):** If you have access to reliable reference air quality measurements, use them for calibration. Reference measurements can come from air quality monitoring stations or certified air quality meters.

**3. Sensor Warm-Up:**Allow the sensors to warm up for a period recommended by the sensor manufacturer, typically 24 hours or as specified.

**4. Calibration Procedure**: Perform a two-point calibration if possible. This typically involves exposing the sensor to two different concentrations of the target gas, e.g., clean air and a known concentration of a pollutant.

- Note that for some sensors, you may need to create a calibration curve, as opposed to a simple two-point calibration.

**5. Data Collection:** Capture readings from both the air quality sensor and the reference measurement devices simultaneously. Record data over a sufficient period to ensure a representative dataset.

**6. Calculate Calibration Parameters:**Use the recorded data to determine calibration parameters. These parameters may include slope, intercept, or sensitivity. They help you convert the sensor's output into accurate air quality measurements.

**7. Apply Calibration:**Adjust your sensor code to apply the calibration parameters. This ensures that the sensor's readings are corrected to match the reference measurements.

**8. Testing:** Test the calibrated system by exposing it to different concentrations of the target gas or varying air quality conditions.

- Compare the system's measurements to reference measurements. Ensure that the calibrated system provides accurate results across a range of conditions.

**9. Repeat Calibration and Testing (If Necessary):**Calibrate the system whenever conditions change significantly, or if you suspect drift in the sensor's accuracy.

Periodically test the system to verify its ongoing accuracy.

**10. Data Analysis:**Analyze the data obtained during calibration and testing to assess the system's performance and accuracy. Pay attention to factors like drift, linearity, and sensor response time.

**11. Documentation:**Document all calibration and testing procedures, including calibration parameters, sensor specifications, test results, and environmental conditions.

**12. Quality Assurance:** Implement a quality assurance plan to regularly verify and maintain the system's accuracy and reliability.

Remember that different sensors and systems may require specific calibration procedures, so it's essential to consult the manufacturer's documentation and guidelines for your specific air quality monitoring equipment. Calibration should be an ongoing process to ensure long-term accuracy in air quality monitoring.