## ­­­­­­­ Project 1: AQM

## Project Definition:

## The project involves setting up IoT devices to measure air quality parameters and make the data publicly available for raising awareness about air quality and its impact on public health. The objective is to create a platform that provides real-time air quality information to the public. This project includes defining objectives, designing the IoT monitoring system, developing the data-sharing platform, and integrating them using IoT technology and Python.

**1.Project Objectives:**

Industrial processes like burning fossil fuels, chemical processing, mixing and drilling into tunnels and mines, metal casting, coatings, oil paintings emit a large amount of toxic gases like carbon monoxide, carbon dioxide, methane, sulphur dioxide, hydrogen, etc that lead to adverse effects on work environment. In case these gases are inhaled or not controlled in limited proportions, there are chances that it can lead to serious health hazards.Industries like mining, petrochemicals, metal refining, oil & gas, recyclable, fertilizers and refractories are considered dangerous and they ought to manage safety compliances. The accidents due to gas leakages are common there. With an average person spending around 90% of their time indoors, the effects of poor indoor air quality are more injurious compared to outdoor air pollution. That’s why it is important to monitor and control gas emissions from these industrial units.Our IoT-based Smart Air Quality Monitoring Solution with automation capabilities allows you to assess toxic and flammable gas proportions along with the concentration of air pollutants round the clock.

**2.IOT Devices Desings:**

Previously we have built the LPG detector using MQ6 sensor, Smoke dector sensor using MQ2 sensor, and Air quality analyzer 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.

**3.Data sharing platforms:**

The implementation of the project focuses on the deployment of the sensor in home, the effort to prevent weather conditions from affecting the sensor measurements, and how the actual program of the sensor was designed to work. The sensor had limited mounting options in the home where it could be installed and it had to function reliably in a long period. Implementation challenges were to protect the sensor from the sun and the rain to prolong its effective lifetime and to avoid any kind of technical malfunctions. At the time of the deployment, Uppsala had a late winter snowstorm, which made it necessary to cover up the sensor probes to prevent them from sinking into the snow and getting wet while still enabling them to take air quality measurements. By adding extra protection, the fluent air flow was disturbed, which became another challenge.

**4.Integration Approach:**

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.

**MQ6 SENSOR**

**CIRCUIT CONNECTIONS**

There are two circuits in the project – transmitter circuit and the receiver circuit. The transmitter circuit is built on Arduino UNO and it has MQ6 gas sensor interfaced to it. The circuit is provided human interface using two tactile switches and an LCD display. The circuit is interfaced with the RF transmitter to send alert signals to the receiver circuit.

The receiver circuit is basically RF receiver circuit with buzzer circuit interfaced with the decoder IC. More than one receiver circuits can also be made and used with the project to add multiple alarms. All the alarm circuit added in the project will alert simultaneously when there will be a gas leakage. It should be taken care that if multiple alarm circuits are added in the project, all the RF receivers in those circuits should be configured to the same address byte as that of the RF transmitter in the transmitter circuit.

**Power Supply –** Both the circuits will be powered by 5V DC supply. The 7805 voltage regulator is used to supply the desired voltage. The power can be drawn from a regular battery which can be connected to the 7805 IC. The IC has three pins – pin 1 should be connected to anode of the battery, pin 2 and 3 with the cathode (common ground). The 5V DC should be drawn from the pin 3 of the IC.

**MQ6 Gas Sensor –** The MQ6 gas sensor is a gas sensor module. The module has 4 pins for interfacing of which two pins are VCC and ground, one pin is analog output and one pin is digital pin via a comparator (LM358). The analog output pin of the module is used for detecting concentration level of gas leakage and interfaced with the A0 analog input pin of the Arduino board.

## MQ2 Gas Sensor

The MQ2 sensor is one of the most widely used in the MQ sensor series. It is a MOS (Metal Oxide Semiconductor) sensor. Metal oxide sensors are also known as **Chemiresistors**because sensing is based on the change in resistance of the sensing material when exposed to gasses.

The MQ2 gas sensor operates on 5V DC and consumes approximately 800mW. It can detect **LPG**, **Smoke**, **Alcohol**, **Propane**, **Hydrogen**, **Methane** and **Carbon Monoxide** concentrations ranging from 200 to 10000 ppm.

**Working process:**

When a SnO2 semiconductor layer is heated to a high temperature, oxygen is adsorbed on the surface. When the air is clean, electrons from the conduction band of tin dioxide are attracted to oxygen molecules. This creates an electron depletion layer just beneath the surface of the SnO2 particles, forming a potential barrier. As a result, the SnO2 film becomes highly resistive and prevents electric current flow.

Air Quality Monitoring

**Project Proposal: IoT Air Quality Monitoring System**

# **Project Overview:**

AIR QUALITY MONITORING SYSTEM. Protecting the atmospheric environment involves control of atmospheric emissions as well as an understanding of pollutant dispersion, monitoring emission levels, i.e. concentration in ambient air. To monitor these levels there are Air Quality Monitoring Networks.

## Objectives

The main objectives stated for the development of an air quality measurement and surveillance programme might be to: − facilitate the background concentration(s) measurements, − monitor current levels as a baseline for assessment, − check the air quality relative to standards or limit values, − detect the importance of individual sources, − enable comparison of the air quality data from different areas and countries, − collect data for the air quality management, traffic and land-use planning purposes, − observe trends (related to emissions), − develop abatement strategies, − determine the exposure and assess the effects of air pollution on health, vegetation or building materials, − inform the public about the air quality and raise the awareness, − develop warning systems for the prevention of undesired air pollution episodes, − facilitate the source apportionment and identification, − supply data for research investigations, − develop/validate management tools (such as models), − develop and test analytical instruments and − to support legislation in relation to the air quality limit values and guidelines.

## Scope

* Deployment of IoT sensors in selected public areas.
* Development of a Python script for data collection and transmission.
* Integration with a data-sharing platform.
* User-friendly data visualization for water consumption monitoring.
* Testing and validation of the system's accuracy and reliability

# **Hardware**

**Microcontrollers:**

They are used to interface with the IoT sensors, process data, and facilitate communication with the data-sharing platform.

**Used:** Arduino,Esp32

**Connectivity:**

**Wi-Fi Module**: Use Wi-Fi for data transmission.

# **Sensors**

We have used the following sensors for our project:

Pid sensor

CO2 sensor

## Their uses

PID sensor :

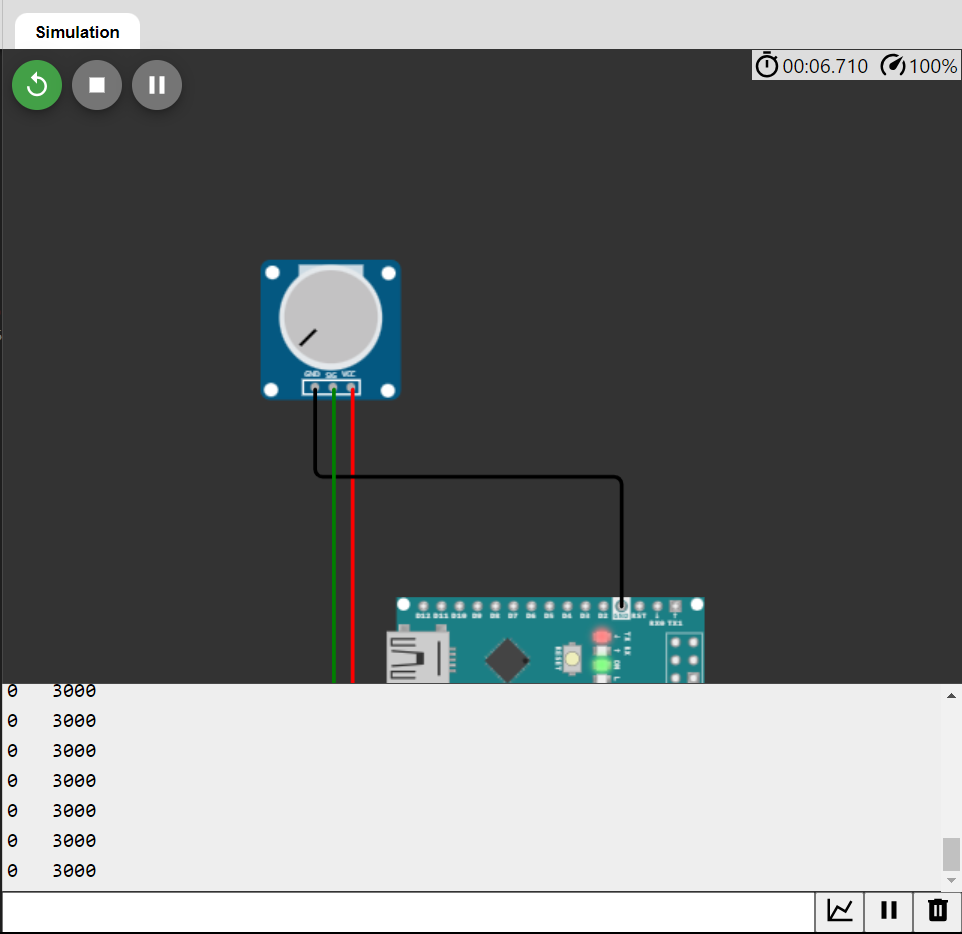
A Photoionization Detector (PID) is a gas detector used to measures volatile organic compounds (VOCs), such as benzene, and other gases.

CO2 sensor:

CO2 sensors are used to monitor fermentation, respiration, photosynthesis, and other carbon dioxide consuming or producing processes. They can be used for HVAC applications to monitor air quality. There are wider applications for CO2 sensors in the agricultural, food, pharmaceutical, refrigeration and brewing sectors.

Screenshots:

PID sensor



# Script for data sharing

Python script for sharing data from PID sensor

|  |
| --- |
| import time |
| import os.path  channel\_id = " 2311196"  write\_api\_key = "BUDE9LL0TGY28R1D "  thing\_speak\_url = "https://api.thingspeak.com/channels.json?api\_key=BUDE9LL0TGY28R1D" |
|  |
| from OmegaExpansion import AdcExp |
| from OmegaExpansion import pwmExp |
|  |
| pwmExp.setVerbosity(-1) |
| pwmExp.driverInit() |
| adc = AdcExp.AdcExp() |
|  |
| targetT = 35 |
| P = 10 |
| I = 1 |
| D = 1 |
|  |
| pid = PID.PID(P, I, D) |
| pid.SetPoint = targetT |
| pid.setSampleTime(1) |
|  |
| def readConfig (): |
| global targetT |
| with open ('/tmp/pid.conf', 'r') as f: |
| config = f.readline().split(',') |
| pid.SetPoint = float(config[0]) |
| targetT = pid.SetPoint |
| pid.setKp (float(config[1])) |
| pid.setKi (float(config[2])) |
| pid.setKd (float(config[3])) |
|  |
| def createConfig (): |
| if not os.path.isfile('/tmp/pid.conf'): |
| with open ('/tmp/pid.conf', 'w') as f: |
| f.write('%s,%s,%s,%s'%(targetT,P,I,D)) |
|  |
| createConfig() |
|  |
| while 1: |
| readConfig() |
| #read temperature data |
| a0 = adc.read\_voltage(0) |
| temperature = (a0 - 0.5) \* 100 |
|  |
| pid.update(temperature) |
| targetPwm = pid.output |
| targetPwm = max(min( int(targetPwm), 100 ),0) |
|  |
| print "Target: %.1f C | Current: %.1f C | PWM: %s %%"%(targetT, temperature, targetPwm) |
|  |
| # Set PWM expansion channel 0 to the target setting |
| pwmExp.setupDriver(0, targetPwm, 0)  TIMT.SLEEP(0.5)  while True:      try:            pid\_sensor\_rate = pid\_sensor\_data()            data = {              "field1": pid\_sensor\_rate          }          response = requests.post(thing\_speak\_url, data=data)          print("Data sent to ThingSpeak. Status code:", response.status\_code)      except Exception as e:          print("Error:", str(e))      time.sleep(15) |

Python script for sharing data fromCO2 Sensor

if \_\_name\_\_ == "\_\_main\_\_":

import co2meter

channel\_id = " 2311196"

write\_api\_key = "BUDE9LL0TGY28R1D "

thing\_speak\_url = "https://api.thingspeak.com/channels.json?api\_key=BUDE9LL0TGY28R1D"

while True:

tb = time.time()

mon = co2meter.CO2monitor()

now = pandas.Timestamp.now()

output\_filename = "{}-{}-{}.csv".format(now.year, now.month, now.day)

if not os.path.exists(output\_filename):

with open(output\_filename, "w") as f:

f.write("Time,Concentration,Temperature\n")

data = mon.read\_data()

t = time.mktime(data.index[0].timetuple())

row = t, np.float64(data["co2"]), np.float64(data["temp"])

print("{}, {} PPM, {} °C".format(\*row))

with open(output\_filename, "a") as f:

writer = csv.writer(f)

writer.writerow(row)

make\_plot()

tsleep = 60 - (time.time() - tb)

if tsleep > 0:

time.sleep(tsleep)

while True:

    try:

        CO2\_sensor\_rate = CO2\_sensor\_data()

data = {

            "field2": CO2\_sensor\_rate

        }

        response = requests.post(thing\_speak\_url, data=data)

        print("Data sent to ThingSpeak. Status code:", response.status\_code)

    except Exception as e:

        print("Error:", str(e))

    time.sleep(15)

# Air Quality Monitoring

To create a platform that displays real-time air quality data, you can use various web development technologies. Here's a high-level overview of the technologies and components you might consider:

1. Front-End Development:

- HTML/CSS: Create the structure and style of your web application.

- JavaScript: Use JavaScript to fetch and display real-time air quality data, as well as create interactive features for your platform.

2. Data Source:

- APIs: You will need access to a reliable air quality data source. Government agencies, environmental organizations, and third-party services often provide APIs for this purpose. Some popular sources include the World Air Quality Index (AQI) API, OpenWeatherMap, or government-specific air quality monitoring APIs.

3. Backend Development:

- Server: Set up a server to handle API requests and perform any necessary data processing.

- Node.js, Python, Ruby, or any other backend language\*\*: Choose a backend language that you're comfortable with.

4. Database(optional):

- Storing historical data or user preferences might require a database. You can use technologies like MySQL, PostgreSQL, MongoDB, or Firebase.

5. Real-Time Data Updates:

- For real-time updates without user interaction, you might consider using \*\*WebSockets\*\* for pushing data to the client. Alternatively, you can use technologies like \*\*Server-Sent Events (SSE)\*\*.

6. Front-End Frameworks and Libraries:

- React, Vue.js, or Angular: These frameworks help in building dynamic and interactive user interfaces.

- Charting Libraries: Use libraries like Chart.js, D3.js, or Highcharts to create interactive and informative data visualizations.

7. Mapping Integration (if displaying data on a map):

- Leaflet, Google Maps, Mapbox: Integrate one of these mapping libraries to display air quality data on a map.

8. User Authentication and Authorization(if necessary):

- Implement user management for features like saving preferences or sharing data.

9. Responsive Design:

- Ensure that your platform works well on both desktop and mobile devices using responsive design principles.

10. Hosting and Deployment:

- Host your application on a cloud platform like AWS, Azure, Google Cloud or use Heroku, Netlify, or Vercel for a simplified deployment process.

11. Security:

- Implement security best practices, including input validation, rate limiting, and ensuring data privacy.

12. Testing and Quality Assurance:

- Perform thorough testing to ensure that your platform works as expected and is bug-free.

13. Monitoring and Maintenance:

- Set up monitoring and alerting to ensure your platform runs smoothly. Regularly update and maintain your application to keep it up-to-date.

14. Documentation:

- Create clear and comprehensive documentation for users and developers who may interact with your platform.

15. Accessibility:

- Ensure that your platform is accessible to people with disabilities, following WCAG (Web Content Accessibility Guidelines).

16. Scalability:

- Plan for scalability, as the amount of data and users may increase over time.

Remember to always consider legal and ethical considerations when displaying and using air quality data. Ensure you have the necessary permissions and follow any legal requirements in your region.

Creating a web platform to display real-time air quality data is a substantial project, and providing all the code in a single response is not feasible due to its complexity. However, I can provide a simplified example using JavaScript and the World Air Quality Index (AQI) API to get you started. This example will show you how to fetch and display air quality data on a web page.

```html

<!DOCTYPE html>

<html>

<head>

<title>Real-time Air Quality</title>

</head>

<body>

<h1>Real-time Air Quality</h1>

<div id="aqiData">

Loading...

</div>

<script>

// Replace with your API endpoint

const API\_URL = 'https://api.waqi.info/feed/here/?token=YOUR\_API\_KEY';

async function fetchAQIData() {

try {

const response = await fetch(API\_URL);

const data = await response.json();

const aqi = data.data.aqi;

const city = data.data.city.name;

const time = data.data.time.s;

const aqiInfo = `Air Quality Index (AQI) in ${city}: ${aqi} (Last updated: ${time})`;

document.getElementById('aqiData').textContent = aqiInfo;

} catch (error) {

console.error('Error fetching data: ', error);

document.getElementById('aqiData').textContent = 'Error fetching data';

}

}

// Fetch data when the page loads and then every N seconds

fetchAQIData();

setInterval(fetchAQIData, 300000); // Update every 5 minutes

</script>

</body>

</html>

```

In this example:

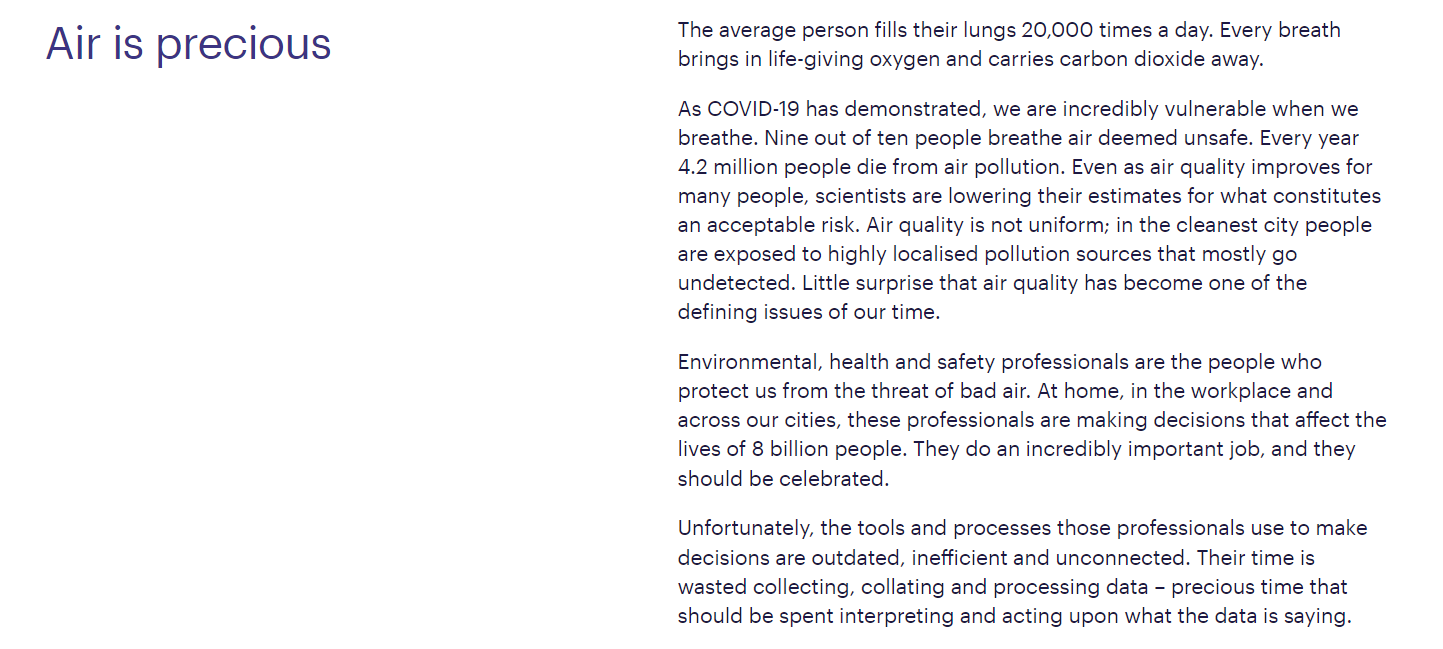
1. Replace `'YOUR\_API\_KEY'` in the `API\_URL` with your actual API key from the World Air Quality Index API.

2. The script fetches data from the API, extracts the AQI, city, and last update time, and displays it on the web page.

3. The data is fetched when the page loads and then updated every 5 minutes using `setInterval`. You can adjust the update interval as needed.

Please note that this is a basic example. A complete platform would require much more functionality, including data visualization, user management, and a well-structured codebase. Additionally, ensure that you respect any terms of use or licensing agreements for the data source you choose.

**Air Quality Designing Model**

****

**Software page:**



Home page:



Next page: