**Air Quality Monitoring**

**1. Objective**

The objective of this project is to develop an IoT-based air quality monitoring system that collects data on various air pollutants, environmental parameters, and sends this data to a central server for analysis and visualization.

**2. Hardware Components**

• IoT Devices (e.g., Raspberry Pi, Arduino, or specialized IoT boards)

• Air Quality Sensors

• Environmental Sensors (e.g., temperature, humidity)

• Connectivity (Wi-Fi, Bluetooth, LoRa, or cellular)

• Power source (e.g., batteries or a power supply)

**3. Sensors for Air Quality Monitoring**

The choice of sensors may vary based on the specific requirements and budget. Common sensors for air quality monitoring include:

1. Particulate Matter (PM) Sensors: Detects fine particulate matter in the air, typically categorized as PM2.5 and PM10.

2. Gas Sensors: These sensors can detect various gases, including:

◦ Carbon Dioxide (CO2) Sensors: Measure indoor air quality.

◦ Carbon Monoxide (CO) Sensors: Detects dangerous levels of CO.

◦ Ozone (O3) Sensors: Measures ozone levels, especially in outdoor environments.

◦ Nitrogen Dioxide (NO2) Sensors: Measures pollution from combustion processes.

◦ Sulfur Dioxide (SO2) Sensors: Detects the presence of sulfur dioxide in the air.

3. Volatile Organic Compounds (VOC) Sensors: Measure indoor air quality by detecting organic compounds.

4. Temperature and Humidity Sensors: Provide environmental data that can affect air quality measurements.

5. Pressure Sensors: Monitor atmospheric pressure, which can influence air quality.

4**. Python Script for IoT Devices**

Develop a Python script to read data from the sensors, process the data, and send it to a central server. The script should also perform tasks like data logging and alerting. Here are some tasks the script should perform:

• Read data from air quality and environmental sensors.

• Process the sensor data, such as calculating air quality index (AQI).

• Send the data to a central server using protocols like MQTT, HTTP, or MQTT.

• Store the data in a database for historical analysis.

• Generate real-time visualizations and alerts when air quality reaches critical levels.

**5. Documentation**

Create a comprehensive document that includes the following sections:

• Project Overview: Briefly explain the purpose of the project and its importance.

• Hardware Setup: Provide a list of hardware components used and how they are connected.

• Software Setup: Explain how to set up the Python script on IoT devices.

• Sensor Calibration: Detail the calibration process for sensors if required.

• Data Transmission: Describe how data is sent to the central server.

• Data Analysis and Visualization: Explain how to analyze and visualize air quality data.

• Troubleshooting: Include common issues and their solutions.

• Conclusion: Summarize the project's achievements and potential future improvements.

**Python script:**

import machine

import network

import urequests as requests # This is a MicroPython library for HTTP requests

ssid = "Your\_SSID"

password = "Your\_PASSWORD"

api\_key = "Your\_ThingSpeak\_API\_Key"

server = "api.thingspeak.com"

http\_port = 80

air\_quality\_pin = machine.ADC(0)

# Connect to Wi-Fi

sta\_if = network.WLAN(network.STA\_IF)

sta\_if.active(True)

sta\_if.connect(ssid, password)

while not sta\_if.isconnected():

pass

print("Connected to WiFi")

def send\_data\_to\_thingspeak(data):

url = f"/update?field1={data}&api\_key={api\_key}"

response = requests.get(f"http://{server}:{http\_port}{url}")

if response.status\_code == 200:

print("Data sent to ThingSpeak")

else:

print("Failed to send data to ThingSpeak")

while True:

t = air\_quality\_pin.read()

print("Air Quality =", t)

if t <= 500:

print("Fresh Air")

elif 500 < t <= 1000:

print("Poor Air")

else:

print("Very Poor")

# Send data to ThingSpeak

send\_data\_to\_thingspeak(t)

machine.delay(10000) # Delay between readings