AIR QUALITY MONITORING – PHASE 3

INTRODUCTION:

In an era where environmental awareness and public health are paramount, the convergence of IoT technology and air quality monitoring has become a vital tool. This project employs a network of sensors to continuously measure and transmit data on air pollutants, temperature, humidity, and other environmental factors. By harnessing the power of the Internet of Things, we gain the ability to access real-time insights, make informed decisions, and work towards a cleaner, healthier, and more sustainable world. This project holds immense promise for enhancing our quality of life and safeguarding the environment for future generations.

OBJECTIVES:

The objectives for an air quality monitoring IoT project are multi-faceted, and they typically revolve around enhancing environmental awareness, public health, and data-driven decision-making. Here are some key objectives for such a project:

Real-time Data Collection: Collect and transmit real-time data on air quality, including levels of particulate matter (PM2.5 and PM10), gases (e.g., CO, NO2, O3), temperature, humidity, and other relevant environmental parameters.

Public Health Protection: Provide valuable air quality information to the public, local authorities, and healthcare providers to help protect individuals from the adverse health effects of air pollution.

Environmental Monitoring: Monitor and record long-term trends in air quality to understand the impact of pollution on the environment and ecosystems.

Alerting System: Develop an alerting system that informs the public and relevant authorities when air quality deteriorates beyond safe thresholds, enabling timely action.

Data Visualization: Create user-friendly data visualization tools such as dashboards and mobile apps to make air quality data accessible and understandable.

Historical Data Analysis: Store historical air quality data for trend analysis, research, and policy development.

Environmental Policy Support: Provide data that can support the formulation of evidence-based environmental policies and regulations.

Community Engagement: Foster community involvement and awareness about air quality and its impact on health and wellbeing.

Sensor Calibration: Ensure sensor accuracy through regular calibration and maintenance procedures.

Scalability: Design the system to be easily scalable, allowing for the addition of more monitoring stations or sensors in the future.

Data Integration: Collaborate with meteorological and other relevant data sources to offer a comprehensive view of how air quality is affected by weather and environmental conditions.

Energy Efficiency: Implement power-efficient and sustainable solutions to minimize the environmental impact of the monitoring infrastructure.

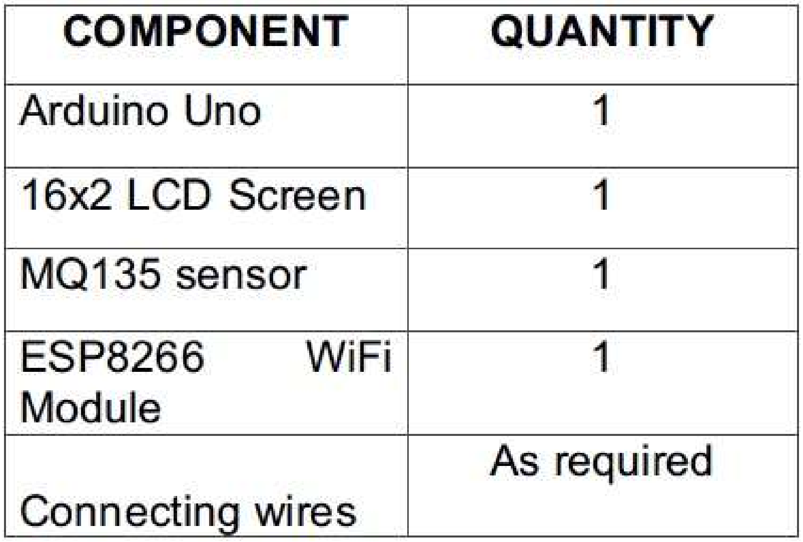
Data Security and Privacy: Prioritize the security and privacy of the collected data to ensure its integrity and protect individuals' personal information.

Education and Outreach: Conduct educational programs and outreach efforts to inform the public about the importance of air quality and the project's goals.

Partnerships: Collaborate with governmental and non-governmental organizations, research institutions, and technology companies to leverage resources, expertise, and funding.

These objectives are essential for ensuring that an air quality monitoring IoT project is effective, impactful, and capable of contributing to cleaner and healthier communities.

COMPONENTS REQUIRED:



COMPONENT	QUANTITY
Arduino Uno	1
16x2 LCD Screen	1
MQ135 sensor	1
ESP8266 WiFi Module	1
Connecting wires	As required

HARDWARE COMPONENTS:

- Raspberry Pi (or similar)
- SDS011 Air Quality Sensor
- DHT22 or DHT11 Temperature and Humidity Sensor
- Jumper wires

PYTHON LIBRARIES:

- RPi.GPIO
- Adafruit-circuitpython-dht

PYTHON SCRIPT FOR AIR QUALITY MONITORING:

```
import time
import RPi.GPIO as GPIO
import Adafruit_DHT
import serial

# GPIO Pin for DHT22 sensor
DHT_SENSOR_PIN = 4

# Initialize DHT sensor
dht_sensor = Adafruit_DHT.DHT22

# Initialize SDS011 serial connection
ser = serial.Serial('/dev/ttyUSB0')

def read_dht_sensor():
    humidity, temperature =
Adafruit_DHT.read_retry(dht_sensor, DHT_SENSOR_PIN)
    return temperature, humidity

def read_sds011_sensor():
    data = ser.read(10)
    if data[0] == 66 and data[1] == 77:
        pm25 = (data[3] * 256 + data[2]) / 10.0
        pm10 = (data[5] * 256 + data[4]) / 10.0
        return pm25, pm10
    return None, None

try:
    while True:
        temperature, humidity = read_dht_sensor()
        pm25, pm10 = read_sds011_sensor()

        if temperature is not None and humidity is not
None:
            print(f'Temperature: {temperature:.1f}°C,
Humidity: {humidity:.1f}%')

        if pm25 is not None and pm10 is not None:
            print(f'PM2.5: {pm25} µg/m³, PM10: {pm10}
µg/m³')

        time.sleep(60)  # Read data every minute

except KeyboardInterrupt:
    pass

ser.close()
GPIO.cleanup()
```

EXPLAINATION:

Importing Libraries: The code begins by importing the necessary Python libraries:

- time: Used for creating time delays.
- RPi.GPIO as GPIO: This library is used for GPIO (General Purpose Input/Output) operations on the Raspberry Pi.

- Adafruit_DHT: A library for working with DHT temperature and humidity sensors.
- serial: Used for serial communication with the SDS011 sensor.

<u>GPIO Pin and Sensor Initialization:</u>

- DHT_SENSOR_PIN is set to 4, which indicates the GPIO pin to which the DHT22 sensor is connected.
- dht_sensor is initialized as a DHT22 sensor for temperature and humidity readings.
- A serial connection (ser) is established to communicate with the SDS011 sensor, assuming it's connected to the /dev/ttyUSB0 port.

<u>Sensor Reading Functions:</u>

- read_dht_sensor(): This function reads data from the DHT22 sensor and returns temperature and humidity values.
- read_sds011_sensor(): This function reads data from the SDS011 sensor. It reads 10 bytes from the serial connection, extracts PM2.5 and PM10 data, and returns these values in micrograms per cubic meter (µg/m³).

<u>Main Loop:</u>

- The script enters a while loop that runs indefinitely, continuously monitoring and printing sensor data.

- Inside the loop, it calls the read_dht_sensor() and read_sds011_sensor() functions to read temperature, humidity, and air quality data.
- If valid data is received from the sensors, it prints this data to the console with appropriate formatting.

Sleep and KeyboardInterrupt Handling:

- The script adds a time.sleep(60) statement, causing the script to sleep for 60 seconds (1 minute) before reading data again. This is to avoid excessive data polling.
- The try...except KeyboardInterrupt block is used to gracefully exit the script when the user presses Ctrl+C.

Cleanup:

- Upon exiting the loop, the script closes the serial connection to the SDS011 sensor and performs GPIO cleanup.

CONCLUSION:

In conclusion, the Air Quality Monitoring IoT project represents a significant stride toward a cleaner, safer, and more sustainable world. By merging cutting-edge technology with a commitment to environmental awareness and public health, this project offers a comprehensive solution to monitor and manage air quality.