AIR QUALITY MONITORING

Phase 3: Development Part 1

INTRODUCTION

As we transition from the planning phase to the execution phase, we are poised to transform our air quality monitoring project into a tangible reality. The innovative vision laid out in Phase 2 is taking shape, driven by our commitment to delivering an effective IoT air quality monitoring system.

DEVELOPMENT TASKS:

Configuring IoT Devices:

The heart of our project lies in the configuration of essential IoT devices, ensuring that they operate in harmony to monitor air quality. Our project now features a range of components:

- MQ135 Gas Sensor: This critical component is the cornerstone of our air quality measurement. It detects various harmful gases that impact air quality and plays a pivotal role in our system.

- Arduino Uno: The Arduino Uno serves as the brain of our operation, responsible for orchestrating data collection and transmission.

- Wi-Fi Module ESP8266: The integration of the ESP8266 Wi-Fi module empowers our system to communicate with external data-sharing platforms, enabling real-time air quality data transmission.

- 16X2 LCD: The 16X2 LCD display is our visual interface, providing human-readable outputs for the measurements our system collects.

- Breadboard: The breadboard acts as a platform for securely connecting and prototyping our components.

- 10K Potentiometer: This component is a valuable addition, allowing us to make precise adjustments to certain sensors, ensuring the reliability of our data.

- 1K Ohm Resistors: These resistors play a role in stabilizing the connections and ensuring the sensors operate within specified parameters.

- 220 Ohm Resistor: Like its counterparts, the 220-ohm resistor contributes to ensuring the stability of our electrical connections.

- Buzzer: The buzzer component can be employed to provide audible alerts or notifications based on air quality thresholds.

Data Transmission to Data-Sharing Platform:

Transmitting our air quality data to the chosen data-sharing platform is the lifeblood of our project. It's essential that the transmission process accommodates the new components and their data.

Integrating New Data: The Python script's data transmission mechanism now includes data from the MQ135 Gas Sensor, enriching the dataset we send to the data-sharing platform.

QUALITY ASSURANCE:

Testing and Validation:

Quality assurance is paramount as we integrate new components and functionalities into our system.

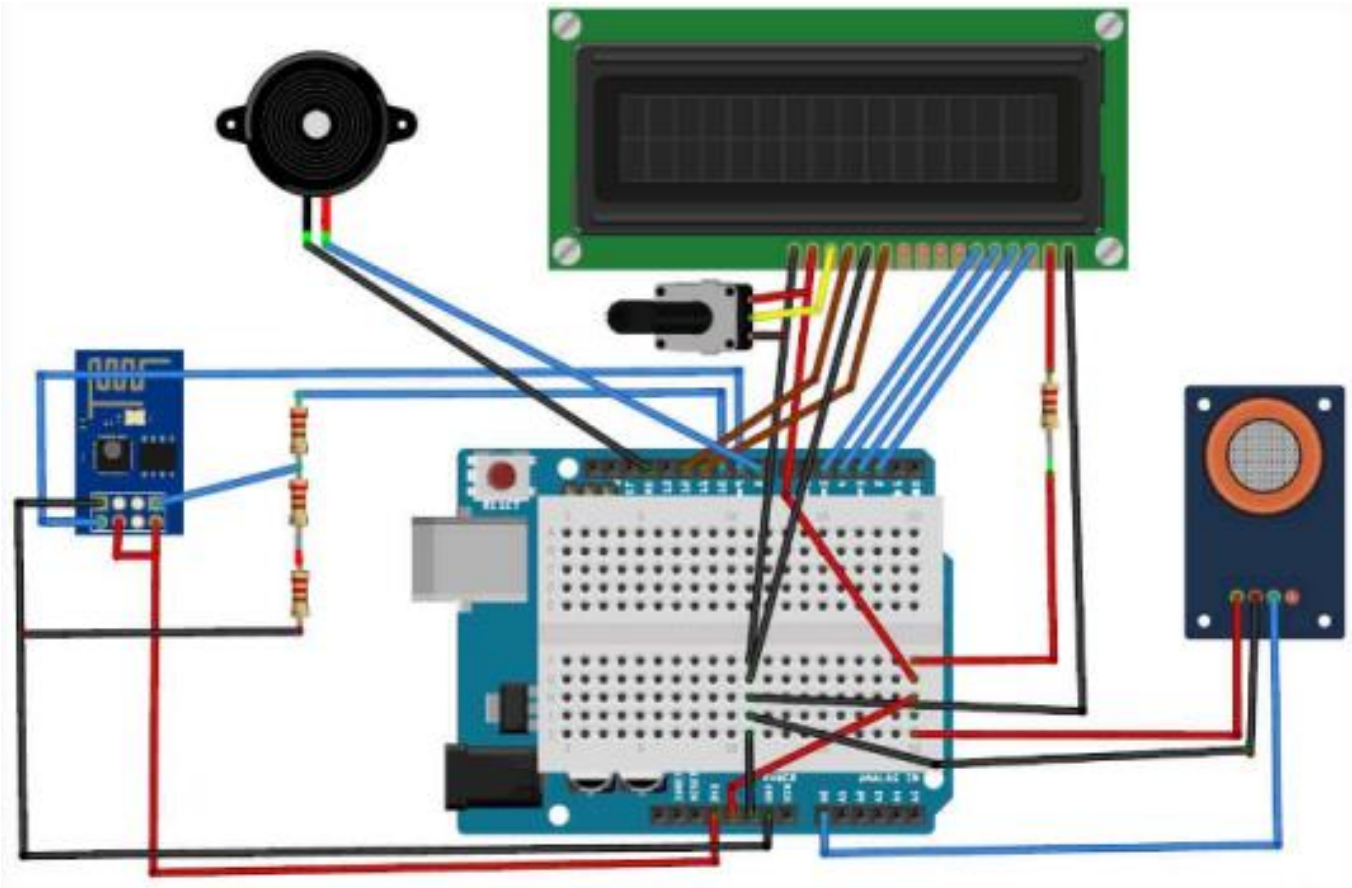
Comprehensive Testing: Our testing phase now includes rigorous validation of the MQ135 Gas Sensor's data. We aim to confirm its accuracy and consistency, just as we do with the existing sensors.

COMPONENTS REQUIRED:

- MQ135 Gas sensor
- Arduino Uno
- Wi-Fi module ESP8266
- 16X2 LCD
- Breadboard
- 10K potentiometer
- 1K ohm resistors
- 220 ohm resistor
- Buzzer

CIRCUIT DIAGRAM:

The circuit diagram for this IoT based air quality monitoring is given below:



PYTHON SCRIPT:

```
import time

import serial

import requests

import network

# Define the serial port for the SDS011 sensor

sdsSerial = serial.Serial('COM3', 9600)

# Wi-Fi module (ESP8266)
```

```
ssid = "Your_WiFi_SSID"
password = "Your_WiFi_Password"
server = "api.thingspeak.com"
apiKey = "Your_ThingSpeak_API_Key"

# 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 read_sensor_data():
    # Read PM2.5 and PM10 data from SDS011 sensor
    data = sdsSerial.read(10)
    if data and data[0] == 170 and data[1] == 192:
        pm25 = data[2] + data[3] * 256
        pm10 = data[4] + data[5] * 256
        return pm25, pm10
    else:
        return None
```

```
def send_data_to_thingspeak(pm25, pm10):
    if pm25 is not None and pm10 is not None:
        url = "https://api.thingspeak.com/update?api_key=" + apiKey + "&field1=" + str(pm25) + "&field2=" + str(pm10)
        response = requests.get(url)
        if response.status_code == 200:
            print("Data sent to ThingSpeak successfully")
        else:
            print("Failed to send data to ThingSpeak. Status code:", response.status_code)



if __name__ == "__main__":
    while True:
        pm25, pm10 = read_sensor_data()
        if pm25 is not None and pm10 is not None:
            print("PM2.5: {} µg/m³, PM10: {} µg/m³".format(pm25, pm10))
            send_data_to_thingspeak(pm25, pm10)
        time.sleep(300)  # Adjust the interval as needed (300 seconds = 5 minutes)
```

EXPLANATION:

1. We import necessary libraries, including time for delays, serial for communication with the SDS011 sensor, and requests for sending data to ThingSpeak via HTTP requests.
2. We define the serial port for the SDS011 sensor (e.g., 'COM3' for Windows or '/dev/ttyUSB0' for Linux). This line establishes the communication channel with the sensor.
3. For the Wi-Fi module (ESP8266), we specify your Wi-Fi SSID and password, the ThingSpeak server URL, and your ThingSpeak API key. Make sure to replace these placeholders with your actual network and ThingSpeak credentials.
4. We connect to Wi-Fi using the network library, waiting until the connection is established before proceeding. This ensures a reliable network connection.
5. The read_sensor_data function reads data from the SDS011 sensor. It checks for the correct start bytes (0xAA and 0xC0) to ensure valid data and calculates the PM2.5 and PM10 values.
6. The send_data_to_thingspeak function constructs a URL with the PM2.5 and PM10 data and your ThingSpeak API key. It then sends a GET request to ThingSpeak to update the data fields.
7. In the main part of the script (if __name__ == "__main__":), we continuously read sensor data, print it to the console, and send it to ThingSpeak every 5 minutes (adjustable by changing the time.sleep interval).

This script connects to the SDS011 sensor, reads air quality data, and sends it to ThingSpeak for monitoring.

CONCLUSION:

As we advance into Phase 3, our IoT air quality monitoring system takes on a more robust form with the integration of new components. The expansion of our sensor network, the addition of a potentiometer for fine-tuning, and the inclusion of a buzzer all contribute to the growing potential of our system. We are now better equipped to deliver accurate and comprehensive air quality data, setting the stage for the continued development of our project in Phase 4.