PHASE – 3 DEVELOPMENT

PART-1

OBJECTIVE:

The objective of setting up an IoT network to measure air quality parameters and publicly sharing the data is to raise awareness about air quality issues and their direct impact on public health. By providing real-time, accessible, and accurate air quality information, our goal is to empower communities, individuals, and policymakers to make informed decisions and take proactive measures to improve air quality and safeguard public health.

IMPLEMENTATION:

Establishing an IoT network for monitoring air quality parameters and developing a Python script on IoT devices to transmit the collected data entails the following components and steps:

COMPONENTS:

❖ ***IoT Device:***

Devices like Raspberry Pi, Arduino, ESP8266, and other microcontrollers with internet connectivity capabilities can certainly be used as IoT devices for air pollution monitoring. These devices can serve as the hardware platform to connect various air quality sensors and collect data on parameters such as particulate matter (PM2.5 and PM10), carbon monoxide (CO), carbon dioxide (CO2), ozone (O3), nitrogen dioxide (NO2), and volatile organic compounds (VOCs).

❖ ***Sensors:***

Various sensors can be utilized for measuring air quality parameters like temperature, humidity, pH, and turbidity. Popular choices include the DHT22 for temperature and humidity, various pH sensors, the MQ-135 gas sensor for air quality, the BME-280 sensor for temperature, humidity, and pressure, and the PM2.5 Dust and Particulate Matter Sensor for air pollution monitoring.

❖ ***Internet Connectivity:***

The IoT device should be able to connect to the internet, either through Wi-Fi or Ethernet.

❖ ***IoT Platform:***

The IoT platforms for measuring air quality parameters:

- **i.** Arduino IoT Cloud
- **ii.** Adafruit IO
- **iii.** ThingSpeak
- **iv.** Cayenne by myDevices
- **v.** Losant
- **vi.** Blynk
- **vii.** Particle

❖ ***Python Script:***

Develop a Python script to read data from the sensors and send it to your IoT platform.

The Python script collects air quality and environmental data from Arduino and BME280 sensors and sends it to ThingSpeak for storage and visualization.

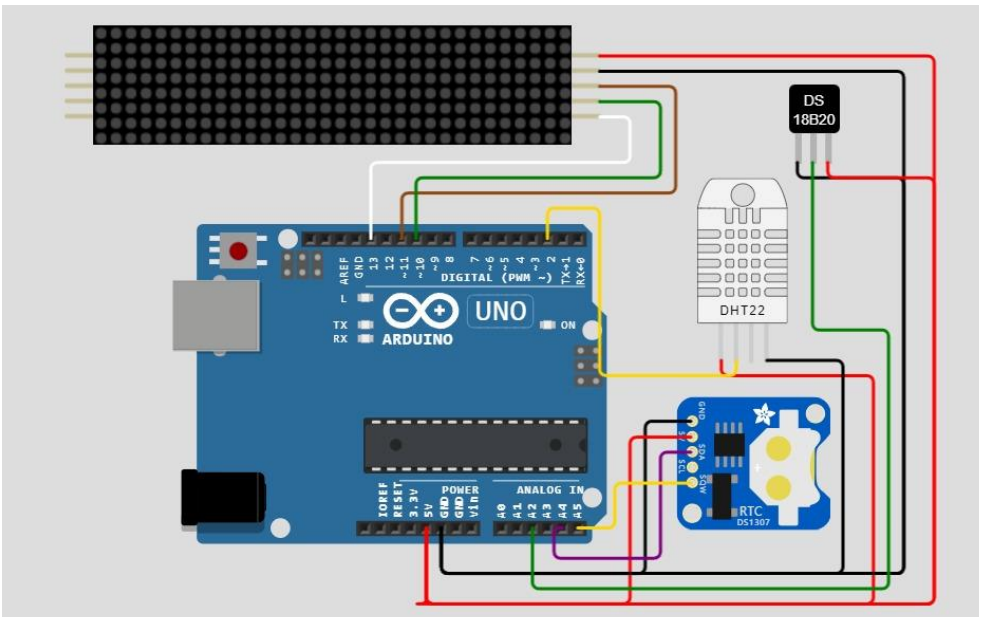
STEPS:

<u>Import Required Libraries:</u>

```
# Libraries for working with sensors in Python
from smbus import SMBus
import time
import board
import digitalio
import busio
from mq135 import MQ135
import adafruit_bme280
```

<u>ESP8266:</u>

Air Pollution/Quality Monitoring with ESP8266 integrates specialized air quality sensors with the ESP8266 microcontroller. These sensors are designed to detect a range of air pollutants, employing various detection methods, such as light scattering for PM sensors and chemical reactions for gas sensors. The ESP8266 processes sensor data, converting analogue readings into digital values and ensuring accuracy through calibration. This data is then transmitted to a central server or cloud platform via Wi-Fi or other communication means. Users access real-time air quality information through a user-friendly interface, promoting public health, environmental research, and policymaking awareness.



<u>Sensor Setup:</u>

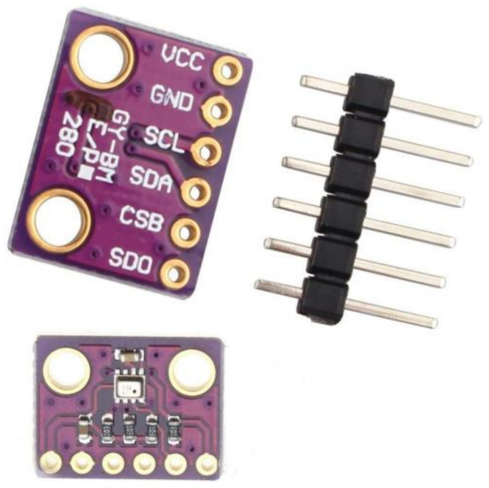
1. MQ-135 gas sensor:

The MQ-135 gas sensor is a vital tool in air quality control and pollution monitoring. Designed to detect a wide range of gases within a concentration range of 10 to 1000 ppm, it plays a crucial role in safeguarding human health and the environment. In this introduction, we'll explore its operational principles and versatile applications, highlighting its importance in modern environmental monitoring.

The working principle of the MQ-135 sensor is based on its ability to measure changes in electrical conductivity when exposed to different gases. In clean air, it exhibits low conductivity. However, when the sensor encounters gases such as carbon dioxide (CO2), carbon monoxide (CO), ammonia (NH3), nitrogen dioxide (NO2), or volatile organic compounds (VOCs), its conductivity increases proportionally to the gas concentration. This change in conductivity is converted into an output signal that corresponds to the detected gas concentration. This real-time data allows for continuous monitoring of air quality, making it an effective early warning system for hazardous gas concentrations, such as carbon monoxide. Additionally, its integration with IoT systems enables remote data collection and analysis for proactive air quality management.

2. **BME-280 Temperature, humidity and pressure sensor:**

The BME-280 sensor operates on the principle of using various integrated sensors to measure temperature, humidity, and atmospheric pressure. Each of these sensors has a distinct working principle:



Temperature Sensing: The temperature sensor within the BME-280 relies on the change in electrical resistance of a temperature-sensitive element. This change in resistance is directly proportional to temperature fluctuations, enabling precise temperature measurements.

Humidity Sensing: The humidity sensor in the BME-280 functions by measuring changes in the capacitance of a thin-film polymer capacitor. As humidity levels change, the capacitance varies, allowing the sensor to accurately determine humidity levels.

Pressure Sensing: The pressure sensor operates on the principle of detecting the deformation of a silicon diaphragm caused by atmospheric pressure. This deformation is converted into an electrical signal, providing accurate measurements of atmospheric pressure.

These sensors work together to provide real-time data on temperature, humidity, and pressure, making the BME-280 sensor a valuable tool for various applications, including weather forecasting, indoor climate control, and environmental research.

3. **PM2.5 Dust and Particulate Matter Sensor:**

The working principle of a PM2.5 Dust and Particulate Matter Sensor is based on light scattering. It typically includes a light source, a scattering chamber, and a detector. Here's how it works:

Light Source: The sensor emits a beam of light, usually in the form of laser or LED light, into the scattering chamber.

Scattering Chamber: Inside the chamber, the emitted light encounters suspended particles in the air. These particles include fine dust, smoke, and other particulate matter.

Light Scattering: When the emitted light interacts with these particles, it gets scattered in different directions. The extent and pattern of scattering are influenced by the size and concentration of the particles in the air.

Detector: A detector within the sensor measures the intensity and angles of the scattered light. The scattered light is typically collected at various angles to capture a range of particle sizes.

Data Analysis: The sensor analyzes the scattered light to determine the concentration of particulate matter in the air, with a focus on PM2.5 particles due to their ability to deeply penetrate the respiratory system and pose health risks.

Data Output: The sensor provides real-time data on the concentration of PM2.5 particulate matter, which can be used for air quality monitoring, pollution assessment, and health risk evaluation.

In summary, a PM2.5 Dust and Particulate Matter Sensor measures the concentration of fine particulate matter in the air by analyzing the scattering of light caused by these particles. This information is crucial for assessing air quality and its impact on health and the environment.

ThingSpeak:

ThingSpeak, as an IoT analytics platform, boasts compatibility with a plethora of platforms and technologies. It seamlessly connects with various IoT devices like **Arduino, Raspberry Pi, and ESP8266**. Additionally, it can collect data from a wide array of sources, including environmental sensors and industrial equipment. With robust APIs for web integration, ThingSpeak easily integrates into web and mobile applications. Its close ties with MATLAB enable advanced data analysis, while support for IoT protocols like MQTT and HTTP ensures broad compatibility. Furthermore, ThingSpeak's capacity to collaborate with third-party services, cloud platforms, and mobile apps makes it a versatile choice for IoT projects, offering users both accessibility and adaptability.

With ThingSpeak, you can leverage real-time air quality data in a variety of ways:

1. **Create Real-Time Dashboards:** ThingSpeak enables you to craft dynamic, real-time dashboards that display current air quality data. These dashboards are invaluable for monitoring air quality in diverse settings, whether it's your home, school, or workplace. The ability to visualize air quality data in real-time empowers you to make informed decisions and take action as needed.

2. **Generate Alerts:** ThingSpeak also provides a mechanism to generate alerts when air quality falls below a predefined threshold. This proactive feature serves as an early warning system, allowing you to promptly respond to deteriorating air quality conditions. Such alerts are instrumental in safeguarding both your well-being and that of others in your environment.

3**. Analyze Trends:** Beyond real-time monitoring, ThingSpeak's capabilities extend to trend analysis of air quality data. By examining these trends, you can pinpoint potential sources of air pollution and devise effective strategies for enhancing air quality. This analytical insight is invaluable for long-term planning and environmental improvements.

ThingSpeak offers a comprehensive solution for managing air quality data, from immediate awareness through real-time dashboards and alerts to long-term improvements via trend analysis.

Thingspeak Setup:

Create a ThingSpeak Account:
Visit the ThingSpeak website (https://thingspeak.com/).
Sign up for an account if you don't already have one.
Create a New Channel:
Once you're logged in, go to the "Channels" tab.
Click on "My Channels" and then "New Channel."
Fill in the necessary details for your channel, such as the name and field names for the data you plan to log.
API Key:
After creating the channel, go to the "API Keys" tab within the channel settings.
Note down your "Write API Key" – you'll need this key to send data to your channel.

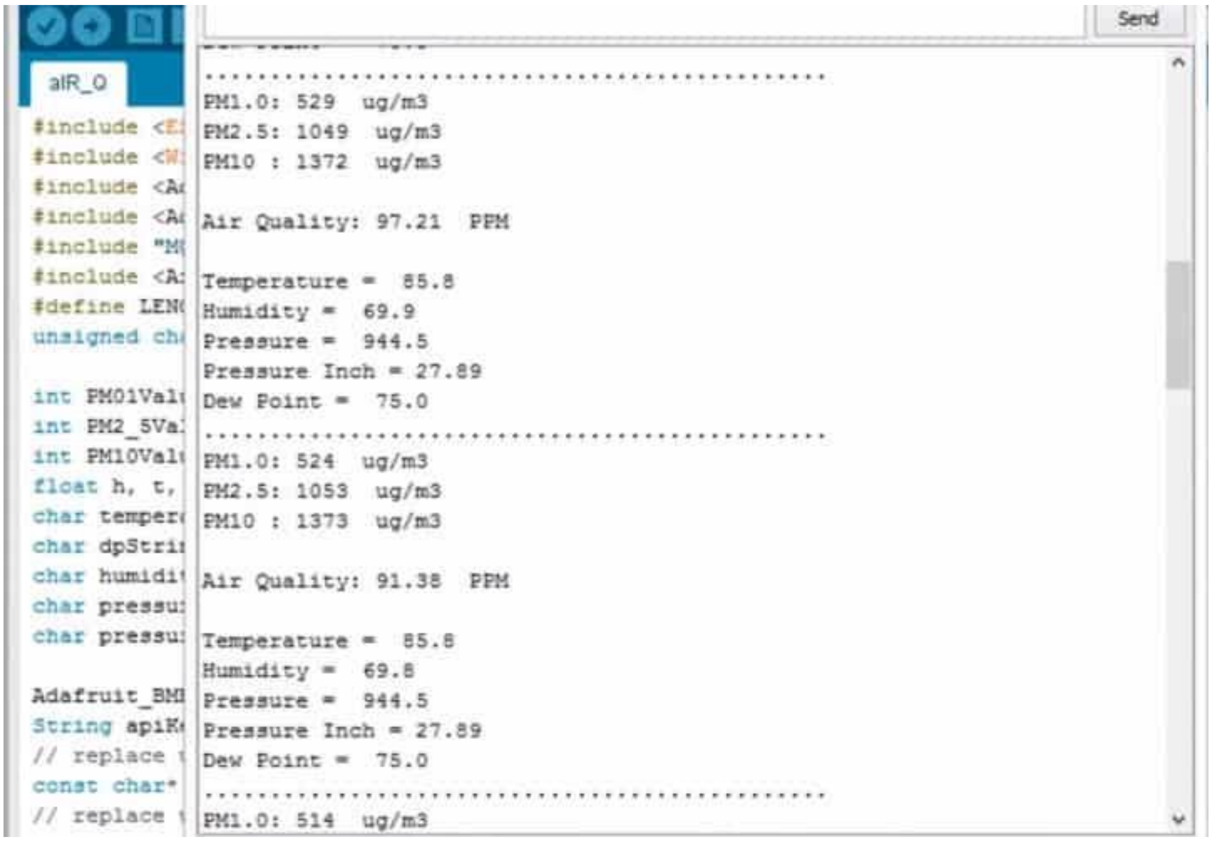
Sending Data to ThingSpeak:
In your IoT device or application (e.g., your Python script), use the Write API Key to send data to your ThingSpeak channel. You can use the ThingSpeak API to make HTTP POST requests or use MQTT for data transmission.
View and Analyze Data:
Go back to your ThingSpeak channel and click on the "Charts" tab to view and analyze your data.
You can set up visualizations, analyze trends, and create alerts based on your data.

CODE:

```
import serial
import time
import requests

# Define constants
LENG = 31
API_KEY = "85JJUZZ7WX809P3M"
THINGSPEAK_SERVER = "api.thingspeak.com"
THINGSPEAK_PORT = 80
```

```
# Create a serial object to communicate with the Arduino
ser = serial.Serial("/dev/ttyUSB0", 9600)

# Create a BME280 sensor object
bme280 = Adafruit_BME280.BME280()

# Connect to ThingSpeak
def connect_to_thingspeak():
  client = requests.Session()
  client. headers.update({
    "Host": THINGSPEAK_SERVER,
    "Connection": "close",
    "X-THINGSPEAKAPIKEY": API_KEY,
    "Content-Type": "application/x-www-form-urlencoded"
  })
  return client

# Send data to ThingSpeak
def send_data_to_thingspeak(client, pm01, pm2_5, pm10, air_quality, temperature, humidity, pressure):
  payload = {
    "field1": pm01,
    "field2": pm2_5,
    "field3": pm10,
    "field4": air_quality,
    "field5": temperature,
    "field6": humidity,
    "field7": pressure
  }
  response = client.post(f"https://{THINGSPEAK_SERVER}/update", data=payload)
  if response.status_code != 200:
    raise Exception(f"Failed to send data to ThingSpeak: {response.status_code}")

def read_data_from_arduino():
  # Wait for the start byte
  while ser.read(1) != b"\x42":
    pass

  # Read the rest of the data
  data = ser.read(LENG - 1)

  # Check the checksum
  checksum = sum(data[:-2]) + 0x42
  if checksum != (data[-2] << 8) + data[-1]:
    raise Exception("Checksum failed")

  # Extract the PM values
  pm01 = (data[3] << 8) + data[4]
  pm2_5 = (data[5] << 8) + data[6]
  pm10 = (data[7] << 8) + data[8]

  return pm01, pm2_5, pm10

def main():
  # Connect to ThingSpeak
  client = connect_to_thingspeak()

  while True:
    # Read data from the Arduino
    pm01, pm2_5, pm10 = read_data_from_arduino()

    # Read temperature, humidity, and pressure from the BME280 sensor
    temperature = bme280.read_temperature()
    humidity = bme280.read_humidity()
    pressure = bme280.read_pressure()

    # Calculate the air quality index
    air_quality = (pm2_5 / 10) + (pm10 / 100)

    # Send the data to ThingSpeak
    send_data_to_thingspeak(client, pm01, pm2_5, pm10, air_quality, temperature, humidity, pressure)

    # Wait for 1 second
    time.sleep(1)

if _name_ == "_main_":
```

OUTPUT:



```
#include <E
#include <W
#include <A
#include <A
#include "M
#include <A
#define LEN
unsigned ch

int PM01Val
int PM2_5Va
int PM10Val
float h, t,
char temper
char dpStri
char humidi
char pressu
char pressu

Adafruit_BM
String apiK
// replace
const char*
// replace
```

```
................................................
PM1.0: 529  ug/m3
PM2.5: 1049  ug/m3
PM10 : 1372  ug/m3

Air Quality: 97.21  PPM

Temperature =  85.8
Humidity =  69.9
Pressure =  944.5
Pressure Inch = 27.89
Dew Point =  75.0
................................................
PM1.0: 524  ug/m3
PM2.5: 1053  ug/m3
PM10 : 1373  ug/m3

Air Quality: 91.38  PPM

Temperature =  85.8
Humidity =  69.8
Pressure =  944.5
Pressure Inch = 27.89
Dew Point =  75.0
................................................
PM1.0: 514  ug/m3
```