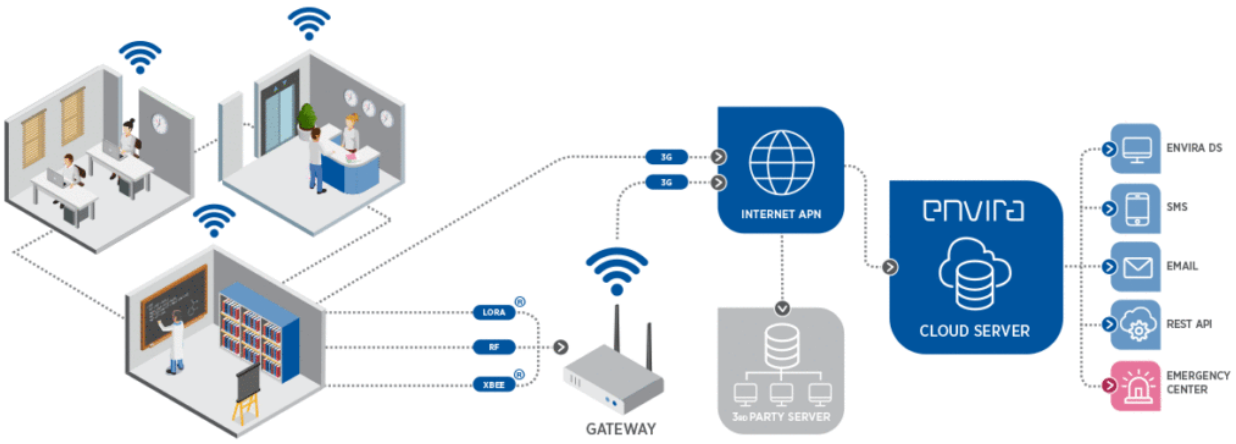
AIR QUALITY MONITORING

Developing a Python script for air quality monitoring involves several steps. Here's a high-level overview of the process:

Python program for developing Iot devices:

```
import time

# Function to read air quality data from the sensor (replace with your actual sensor code)

def read_air_quality():

  # Replace this with your sensor reading logic

  air_quality_data = {

    "pm2.5": 10.0,  # Replace with actual PM2.5 reading

    "pm10": 15.0,   # Replace with actual PM10 reading

    "co2": 400,     # Replace with actual CO2 reading

  }

  return air_quality_data


# Main monitoring loop

while True:

  try:

    air_quality = read_air_quality()

    print("Air Quality Data:")

    print(f"PM2.5: {air_quality['pm2.5']} µg/m³")

    print(f"PM10: {air_quality['pm10']} µg/m³")

    print(f"CO2: {air_quality['co2']} ppm")


    # Add data storage and analysis logic here
```

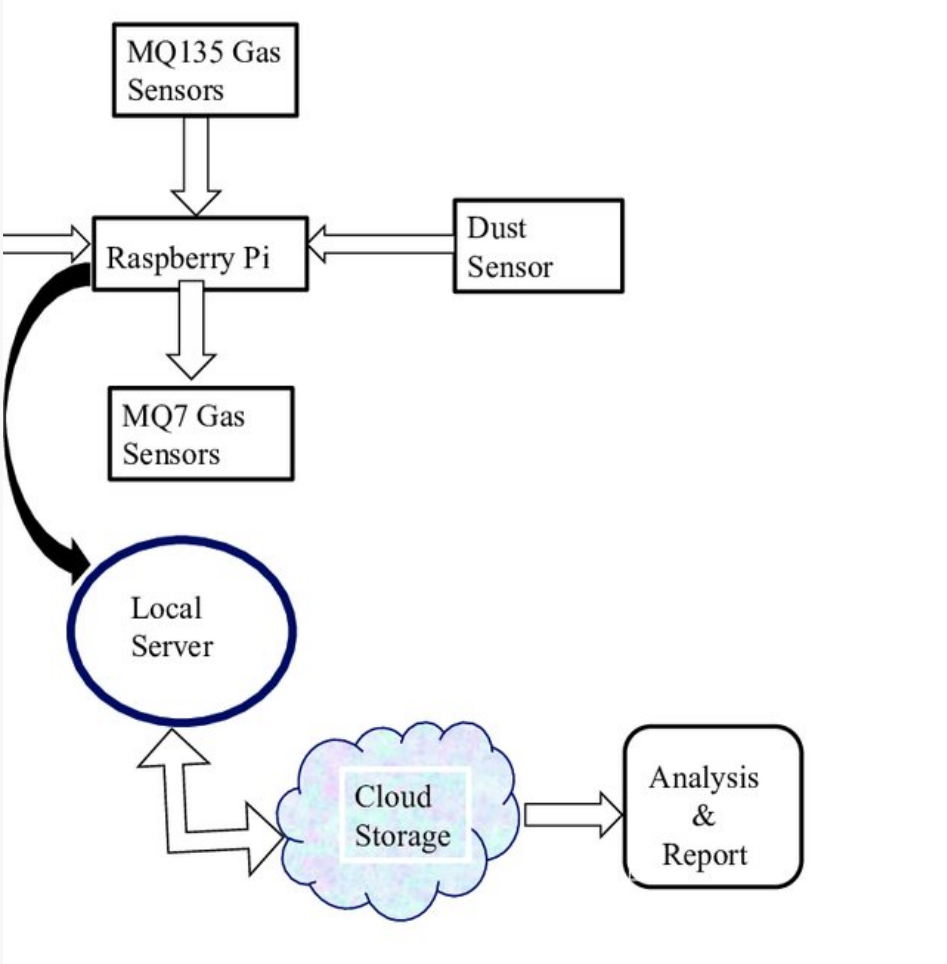
```
        time.sleep(300)  # Sleep for 5 minutes (adjust as needed)

    except KeyboardInterrupt:
        break

# Close any resources or connections when exiting the script
# Add your data storage and analysis logic here

print("Monitoring stopped.")
```



Developing a Python script for air quality monitoring involves several steps. Here's a high-level overview of the process:

1. **Set Up Your Environment:**
   - Ensure you have Python installed on your system.
   - Use virtual environments to manage dependencies.
2. **Choose an Air Quality Sensor:**
   - Select an appropriate air quality sensor (e.g., particulate matter, gas sensors) based on your requirements.
3. **Wiring and Hardware Setup:**
   - Connect the sensor to your Raspberry Pi, Arduino, or microcontroller, depending on your hardware.

4. **Install Required Libraries:**
    - Install libraries for sensor communication, such as Adafruit CircuitPython or smbus2 for I2C communication.



5. **Read Sensor Data:**
    - Write code to read data from the air quality sensor. This may involve reading analog or digital data and calibrating it.
6. **Data Processing:**
    - Process the sensor data if necessary (e.g., converting raw data into meaningful air quality metrics like PM2.5 or AQI).
7. **Data Storage:**
    - Choose a storage solution (e.g., SQLite, MySQL, or cloud-based databases) to save the air quality data.
8. **Data Visualization:**
    - Use libraries like Matplotlib, Plotly, or web frameworks like Flask or Django to create real-time or historical data visualizations.
9. **Data Analysis and Alerts:**
    - Implement logic to analyze air quality data and trigger alerts or notifications if air quality falls below a certain threshold.

10. **Logging and Reporting:**
    - Create a system for logging and reporting the air quality data over time.
11. **User Interface** (Optional):
    - Develop a user-friendly interface using tools like PyQt, Tkinter, or web technologies.
12. **Testing and Calibration:**
    - Test your setup extensively and calibrate the sensor for accuracy.
13. **Deployment:**
    - Deploy your script on your desired hardware platform and ensure it runs continuously.
14. **Data Sharing** (Optional):
    - If needed, implement APIs or protocols to share air quality data with other devices or services.
15. **Maintenance:**
    - Regularly monitor and maintain your air quality monitoring system to ensure it continues to function correctly.

Project Submitted By,

Name : A.chandra devi

NM Id : au713921106005

Topic : AIR QUALITY MONITORING

Mail Id : chandradeviaec@gmail.com

College code : 7139