Air Quality Monitoring System

Objective: Air Quality Monitoring System is a real-time tracking system which analyzes the air quality using IoT devices and visualizes the percentage of air pollutants present in the atmosphere. By using this system, society can check the air quality of the city where they are living.

IoT Technology and Sensor: The IoT technology which we decided to use is NodeMCU(WiFi-ESP32) and the sensor we are going to use is BME680 which is a gas sensor which detects temperature, humidity, pressure and air quality. The data from the gas sensor will be displayed to the client as real-time data through esp-32.

Project Overview: Start by clearly defining the project's scope and objectives. What do you want to achieve with your air quality monitoring system? This might include monitoring pollutants like particulate matter (PM2.5, PM10), carbon dioxide (CO2), volatile organic compounds (VOCs), temperature, and humidity.

Hardware Selection: Choose the appropriate hardware for your project. ESP32 is a popular IoT microcontroller, and BM3680 seems to be a sensor module. Ensure that these components are compatible and can measure the required parameters. For instance, the BM3680 could contain sensors for air quality, and the ESP32 would serve as the controller.

Sensor Calibration: Calibrate the sensors to ensure accurate readings. This step is essential to correct any sensor drift or inaccuracies.

Wiring and Assembling: Connect the BM3680 sensor module to the ESP32. Pay close attention to the datasheets and pin configurations to ensure proper wiring. You may need additional components like resistors and capacitors.

Programming the ESP32: Write a Python script for the ESP32 using the MicroPython or CircuitPython environment. This script should configure the sensor, read data from it, and transmit the data to a central server or cloud platform. Make sure the script runs at regular intervals to continuously monitor air quality.

Connectivity: Implement connectivity options for the ESP32 to transmit data. You can use Wi-Fi, cellular, or other communication methods based on your project requirements. MQTT or HTTP can be used for data transfer.

Data Storage and Visualization: Set up a data storage solution to collect and store the air quality data. You can use a database system or cloud services like AWS, Google Cloud, or Azure. Create a web interface or use a dashboard to visualize the data in real-time.

Alerting and Notifications: Implement an alerting system to notify users or administrators when air quality reaches predefined thresholds. This could be done through email, SMS, or push notifications.

Power Management: Optimize power consumption to prolong the battery life of the IoT devices. Use sleep modes when the device is not actively monitoring air quality.

Deployment: Deploy the air quality monitoring system in the intended locations, ensuring that the devices are securely mounted and protected from environmental factors.

Data Analysis and Reporting: Over time, analyze the collected data to identify trends and patterns in air quality.

Python Code For Web Server:

```
def web_page():
  bme = BME680_I2C(i2c=i2c)
  html = """<html><head><title>ESP with BME680</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" href="data:,"><style>body { text-align: center; font-family: "Trebuchet MS", Arial;}
  table { border-collapse: collapse; margin-left:auto; margin-right:auto; }
  th { padding: 12px; background-color: #0043af; color: white; }
  tr { border: 1px solid #ddd; padding: 12px; }
  tr:hover { background-color: #bcbcbc; }
  td { border: none; padding: 12px; }
  .sensor { color:white; font-weight: bold; background-color: #bcbcbc; padding: 1px;
  </style></head><body><h1>ESP with BME680</h1>
  <table><tr><th>MEASUREMENT</th><th>VALUE</th></tr>
  <tr><td>Temp. Celsius</td><td><span class="sensor">""" + str(round(bme.temperature, 2)) + """ C</span></td></tr>
  <tr><td>Temp. Fahrenheit</td><td><span class="sensor">""" + str(round((bme.temperature) * (9/5) + 32, 2))  + """ F</span></td></tr>
  <tr><td>Pressure</td><td><span class="sensor">""" + str(round(bme.pressure, 2)) + """ hPa</span></td></tr>
```

```
  <tr><td>Humidity</td><td><span class="sensor">""" +
str(round(bme.humidity, 2)) + """ %</span></td></tr>
  <tr><td>Gas</td><td><span class="sensor">""" +
str(round(bme.gas/1000, 2)) + """
KOhms</span></td></tr></body></html>"""
  return html

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)

while True:
  try:
    if gc.mem_free() < 102000:
      gc.collect()
    conn, addr = s.accept()
    conn.settimeout(3.0)
    print('Got a connection from %s' % str(addr))
    request = conn.recv(1024)
    conn.settimeout(None)
    request = str(request)
    print('Content = %s' % request)
    response = web_page()
```

```
    conn.send('HTTP/1.1 200 OK\n')
    conn.send('Content-Type: text/html\n')
    conn.send('Connection: close\n\n')
    conn.sendall(response)
    conn.close()
  except OSError as e:
    conn.close()
    print('Connection closed')
```