AIR QUALITY MONITORING

PHASE 3

DEVELOPMENT PART 1

1. INTRODUCTION:

Atmospheric conditions continue to deteriorate each year due to the growth of civilization and increasing unclean emissions from industries and automobiles. The health effects of pollution are very severe that causes stroke, lung cancer, and heart disease. Furthermore, air pollutants have a negative impact on humans and the earth's ecosystem, as observed in recent global air pollution problems like ozone depletion .

To reduce exposure to air contamination (especially aerosols), new measures have been pursued, including development of air quality measuring devices and air purifiers.The Ministry of the Environment in Korea assessed the efficacy of 17 widely used air quality measuring devices by analyzing their accuracy and reliability.

Therefore, an IoT-based indoor air quality monitoring platform based on integration of cloud computing and IoT is presented in this research.

2. COMPONENTS FEATURES:

HARDWARE COMPONENTS:

SENSORS:

Gas Sensors:

Gas sensors in the context of the Internet of Things (IoT) refer to sensors that detect and measure various gases present in the environment. These sensors play a critical role in monitoring air quality, ensuring safety in industrial environments, and enabling environmental monitoring systems.

1. Types of Gas Sensors: Gas sensors can detect a wide range of gases, including but not limited to:

Carbon Monoxide (CO) Sensors

Carbon Dioxide (CO2) Sensors

Methane (CH4) Sensors

Oxygen (O2) Sensors

Volatile Organic Compounds (VOC) Sensors

Nitrogen Dioxide (NO2) Sensors

Sulfur Dioxide (SO2) Sensors

Ammonia (NH3) Sensors

Hydrogen (H2) Sensors

Ozone (O3) Sensors

2. Integration with IoT:

Gas sensors are integrated into IoT devices or systems to monitor gas levels in real-time and transmit this data to a central server or a cloud platform for analysis and further action.

IoT-enabled gas sensors are equipped with communication protocols such as Wi-Fi, Bluetooth, Zigbee, LoRaWAN, or MQTT, allowing seamless data transmission to centralized IoT platforms.

3. Applications:

Air Quality Monitoring: Gas sensors are used to monitor air pollutants in urban areas,

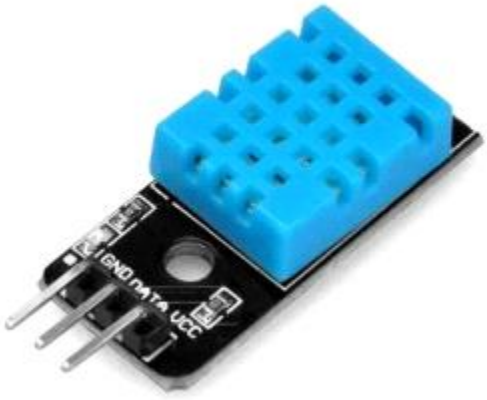
industrial zones, and indoor environments to assess air quality and provide alerts for potential health hazards.

Industrial Safety: Gas sensors are deployed in industrial settings to monitor for potentially dangerous gases and ensure the safety of workers.

Environmental Monitoring: Gas sensors help in monitoring and analyzing the presence of harmful gases in natural environments such as forests, bodies of water, and remote locations.

Automotive Applications: Gas sensors are utilized in vehicles to monitor exhaust emissions and maintain compliance with environmental regulations.

Sensors for Humidity and Temperature:



Integrated Sensors: Many modern sensors are capable of measuring both humidity and temperature simultaneously. Common examples include digital humidity and temperature sensors such as the DHT series (e.g., DHT11, DHT22), BME280, and SHT series sensors.

Humidity Sensors: These sensors measure the moisture content in the air and provide readings in percentage (% relative humidity).

Temperature Sensors: These sensors measure the ambient temperature and provide readings in degrees Celsius (°C) or Fahrenheit (°F).

Arduino Uno:

Arduino Uno is a microcontroller board based on the ATmega328P microcontroller. It has

digital and analog input/output pins that can be used to connect sensors, actuators, and other components.

Arduino Uno can be programmed using the Arduino Integrated Development Environment (IDE), which supports a simplified C/C++ programming language.

Connectivity:

To enable IoT capabilities, you need to connect the Arduino Uno to the internet or other devices. Common connectivity options include:

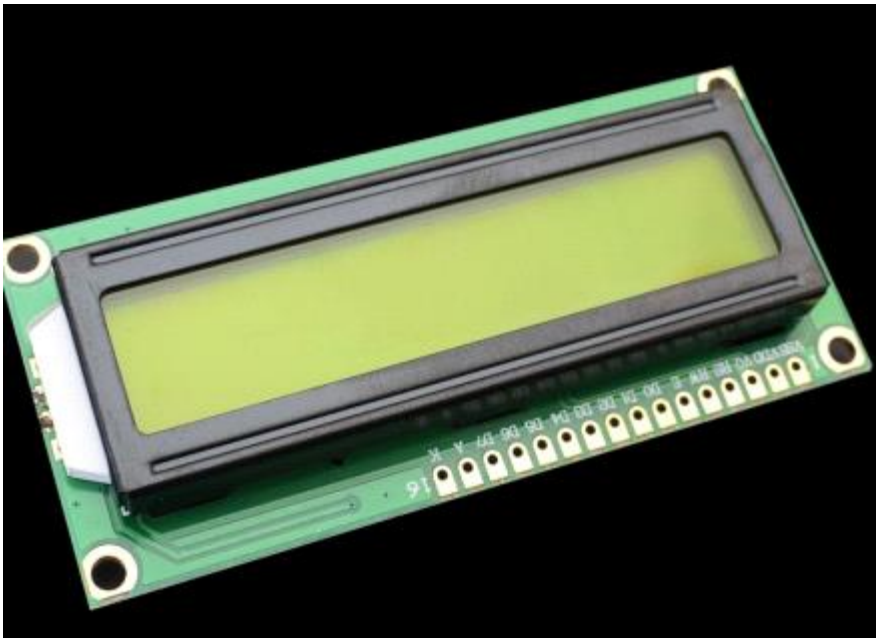
Wi-Fi: Use Wi-Fi shields or modules (e.g., ESP8266, ESP32) to connect Arduino Uno to a Wi-Fi network for IoT communication.

Ethernet: Utilize Ethernet shields to enable wired internet connectivity for IoT applications.

Bluetooth and Zigbee: Integrate Bluetooth or Zigbee modules for short-range communication between Arduino Uno and other devices.

LoRaWAN: Use LoRaWAN modules for long-range, low-power communication suitable for IoT applications.

LCD DISPLAY:



1. Selecting an LCD Display:

Choose an appropriate LCD based on your requirements, such as size, types(character, graphical), and interface (e.g., I2C, SPI, parallel).

2. Connecting the LCD to Arduino Uno:

Wire the LCD display to the Arduino Uno using the required connections (e.g., data pins, power, ground). For example, if using an I2C-enabled LCD, connect it to the I2C pins (SDA and SCL) on the Arduino Uno.

3. Installing the LCD Library:

In the Arduino IDE, go to "Sketch" > "Include Library" > "ManageLibraries.&quot. Search for the appropriate LCD library for your display type (e.g.,LiquidCrystal, LiquidCrystal_I2C) and install it.

SOFTWARE COMPONENTS:

Air quality monitoring using IoT typically involves a combination of software components for data collection, processing, visualization, and communication. Here are the key software components used in an air quality monitoring system:

1. Sensor Drivers and Libraries: These software components are responsible for interfacing with the air quality sensors. They provide the necessary drivers and libraries to communicate with the sensors and retrieve data.

2. Data Acquisition and Processing Software:

Data Collection: Software that collects data from the air quality sensors, often via sensors' drivers or libraries.

Data Filtering and Preprocessing: This software component can clean and preprocess the raw sensor data, removing noise and anomalies.

Data Storage: Data is typically stored in databases or cloud storage for historical analysis. Popular databases for IoT data include PostgreSQL, MongoDB, and cloud-based solutions like AWS IoT Core or Google Cloud IoT Core.

3. IoT Communication Protocols:

MQTT (Message Queuing Telemetry Transport): MQTT is commonly used for lightweight, low-bandwidth data transmission between IoT devices and the cloud. It is suitable for real-time data streaming.

HTTP/HTTPS: Used for web-based communication, APIs, and data transmission to web services.

CoAP (Constrained Application Protocol): A lightweight protocol for resource-constrained IoT devices.

4. IoT Middleware or Cloud Platform:

IoT Platforms: Platforms like AWS IoT, Google Cloud IoT, Microsoft Azure IoT, and IBM Watson IoT provide scalable, cloud-based solutions for data management, analytics, and device management.

IoT Edge Computing: Some data processing and analysis can be done at the edge (on the device) to reduce latency and minimize data transmission.

5. Data Analysis and Visualization:

Data Analytics: Tools for analyzing air quality data, including statistical analysis, anomaly

detection, and machine learning for predictive analysis.

Visualization Tools: Software for creating dashboards and data visualizations, allowing users to monitor air quality in real-time. Popular tools include Grafana, Kibana, and custom web-based interfaces.

6. Alerting and Notification Systems:

Software that can trigger alerts and notifications based on predefined thresholds or anomaly detection. Notifications can be sent via email, SMS, or push notifications to mobile apps.

7. Database Management Systems (DBMS):

Used for storing historical air quality data. Popular options include SQL databases (e.g., PostgreSQL, MySQL) or NoSQL databases (e.g., MongoDB).

8.Firmware and Over-the-Air (OTA) Updates: For remote device management and updates, ensuring the IoT devices' firmware remains up-to-date and secure.

Creating an IoT-based air quality monitoring system using Arduino and Python involves several components and steps:

Components:

1. Arduino board (e.g., Arduino Uno or Arduino Nano)
2. Air quality sensor (e.g., MQ series sensor)
3. Wi-Fi module (e.g., ESP8266 or ESP32) for IoT connectivity
4. A computer or Raspberry Pi for running the Python program
5. Blynk or a similar IoT platform for data visualization (optional)

Steps:

1. Hardware Setup:

- Connect the air quality sensor to the Arduino following the sensor's datasheet and your chosen Arduino model.
- Connect the Wi-Fi module to the Arduino, ensuring it can connect to your Wi-Fi network.

2. Arduino Code:

Write an Arduino sketch to read data from the air quality sensor and send it to the IoT platform (e.g., Blynk). Make sure to include the necessary libraries for your sensor and Wi-Fi module.

Here's a simplified example using the Adafruit CCS811 air quality sensor and an ESP8266 Wi-Fi module:

```
#include <Wire.h>
#include <Adafruit_CCS811.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char auth[] = "Your_Blynk_Auth_Token";
char ssid[] = "Your_WiFi_SSID";
char pass[] = "Your_WiFi_Password";

Adafruit_CCS811 ccs;

void setup() {
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);
  ccs.begin();
}

void loop() {
  if (!ccs.available()) {
    return;
  }
  if (!ccs.readData()) {
    return;
  }
  float temperature = ccs.calculateTemperature();
  float humidity = ccs.calculateHumidity();
  int eCO2 = ccs.geteCO2();
  int TVOC = ccs.getTVOC();

  Serial.print("eCO2: ");
  Serial.println(eCO2);
  Serial.print("TVOC: ");
  Serial.println(TVOC);

  Blynk.virtualWrite(V1, eCO2); // Send eCO2 data to virtual pin V1
  Blynk.virtualWrite(V2, TVOC); // Send TVOC data to virtual pin V2
  Blynk.run();

  delay(1000); // Delay between readings
}
```

3. Python Code:

Create a Python program to receive the data sent by the Arduino via Wi-Fi. You can use the `pySerial` library to communicate with the Arduino over the serial port. Additionally, you may use an IoT platform's API for data retrieval if you're not using Blynk for visualization.

Here's a simple Python script to read data sent by the Arduino over the serial port:

```
import serial

ser = serial.Serial('COMX', 115200)  # Replace 'COMX' with the appropriate serial port

while True:
    data = ser.readline().decode('utf-8').strip()
    print(data)
```

4. Visualization and Analysis:

Use a library or IoT platform of your choice (e.g., Blynk, ThingSpeak, or custom web application) to visualize and analyze the air quality data received from the Arduino.

Remember to replace placeholders like "Your_Blynk_Auth_Token," "Your_WiFi_SSID," and "COMX" with your specific values. Additionally, adapt the code to the specific air quality sensor and Wi-Fi module you are using.