**PHASE 5:PROJECT DOCUMENTATION AND SUBMISSION**

**SMART ENVIRONMENT MONITORING USING IOT**

**INTRODUCTION:**

Our World is undergoing an unprecedented transformation. Cities are rapidly expanding, and technology is advancing at a breathtaking pace. Consequently,the delicate equilibrium of our environment is constantly under threat. Factors like air quality,temperature,humidity and water quality plays a critical role in shaping the quality of life for individuals and entire communities.However,the traditional methods of environmental monitoring often fall short in delivering timely,comprehensive,and precise data necessary for well-informed decision-making.

**PROBLEM STATEMENT:**

As we run to the speed of technology,we often fail to notice the small things around us that can have notable impact on our health.In this project,we aim to monitor various aspects of environment and give a summarised report on current status.

Various problem is observed in environment monitoring system today including cost,heavy power consumption,data security etc.We hope to solve these problems in this project by deploying low cost yet more reliable sensors and using alternative source of energy.

**DESIGN THINKING:**

As the domain of smart environmental monitoring is vast,it is more efficient to focus on a single area.Every healthy man is a man with healthy home.According to the United States Environmental Protection Agency (EPA),indoor air is 100 times more contaminated than outside air.Most modern populations spend 80 to 90 percent of their time indoors;therefore,indoor air has a great direct impact on human health than outside air.Moreover,in contrast to atmospheric pollution ,indoor pollutants are about 1000 times more likely to be transmitted to the lungs ,causing diseases such as sick building syndrome,multiple chemical sensitives and dizziness.

In this project,we are focusing on improving the air quality especially in household.We plan on doing the project using:

1.Microcontroller(raspberry pi or Arduino with a Wi-Fi)

2.Temperature and Humidity Sensor:DHT22/DHT11

3.Carbon Dioxide (CO2)Sensor:Non-Dispersive Infrared (NDIR)Sensors

4.Volatile Organic Compounds(VOC)Sensor:SGP30

5.Air Quality Sensors:MQ Series Sensors,Bosch BME680

6.Carbon Monoxide(CO)Sensor:NDIR CO Sensor.

**INNOVATION**

Creating an indoor air quality monitoring project using IoT involves a combination of hardware components and sensors. Here’s a list of commonly used elements for our project.

**PROJECT STEPS:**

1. **Sensors:**
   * **Gas Sensors:** Use sensors capable of measuring specific gases like carbon dioxide (CO2), carbon monoxide (CO), volatile organic compounds (VOCs), methane, etc.
   * **Particulate Matter (PM) Sensors:** Measure the concentration of particulate matter in the air, especially PM2.5 and PM10.
   * **Temperature and Humidity Sensors:** Monitor the temperature and humidity levels in the indoor environment.
2. **Microcontroller/Single Board Computer:**
   * Choose a microcontroller or a single-board computer (SBC) to process and transmit the sensor data. Common choices include Raspberry Pi, Arduino, or ESP8266/ESP32.
3. **Connectivity Module:**
   * Integrate a communication module such as Wi-Fi, Bluetooth, or GSM to enable the IoT device to transmit data to a central server or cloud platform.
4. **Power Supply:**
   * Depending on your deployment, you might use a power adapter, batteries, or a combination of both. Ensure the chosen power supply method is reliable and suits the deployment environment.
5. **Enclosure:**
   * Protect your components from environmental factors with a suitable enclosure. This is especially important for outdoor or harsh indoor environments.
6. **Data Storage/Cloud Platform:**
   * Set up a cloud platform (e.g., AWS, Azure, Google Cloud) or a local server to store and analyze the collected data. This allows for remote monitoring and analysis.
7. **User Interface:**
   * Create a user interface for monitoring and interacting with the data. This could be a web dashboard, mobile app, or both, depending on your application.
8. **Power Management:**
   * Implement power management features to optimize energy consumption. This may include sleep modes for sensors or the entire system when not actively collecting data.
9. **Calibration Tools:**
   * Periodically calibrate the sensors to ensure accurate and reliable measurements. Calibration tools and procedures should be part of the system.
10. **Security Measures:**
    * Implement security features to protect the device and data, especially when transmitting data over the internet. This includes encryption, secure communication protocols, and access controls.
11. **Real-time Clock (RTC):**
    * Use an RTC to maintain accurate timestamps, even when the device is powered off. This is crucial for correlating data over time.

**PROGRAM:**

import com.pi4j.io.gpio.\*;

import com.pi4j.temperature.TemperatureScale;

import com.pi4j.temperature.TemperatureSensor;

import com.pi4j.temperature.TemperatureSensorFactory;

import com.pi4j.util.Console;

public class IndoorAirQualityMonitor {

public static void main(String[] args) {

final Console console = new Console();

// Initialize GPIO

GpioController gpio = GpioFactory.getInstance();

GpioPinDigitalInput dhtPin = gpio.provisionDigitalInputPin(RaspiPin.GPIO\_04, PinPullResistance.PULL\_DOWN);

// Initialize DHT22 sensor

TemperatureSensor dhtSensor = TemperatureSensorFactory.getTemperatureSensor(TemperatureScale.CELSIUS, dhtPin);

try {

while (true) {

// Read temperature and humidity from DHT22

double temperature = dhtSensor.getTemperature();

double humidity = dhtSensor.getHumidity();

// Print the data

console.println("Temperature: " + temperature + " °C, Humidity: " + humidity + "%");

// You can further process and store the data as needed

Thread.sleep(5000); // Delay between readings (adjust as needed)

}

} catch (InterruptedException e) {

console.println("Monitoring stopped");

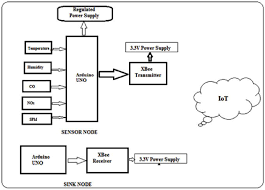
}

}

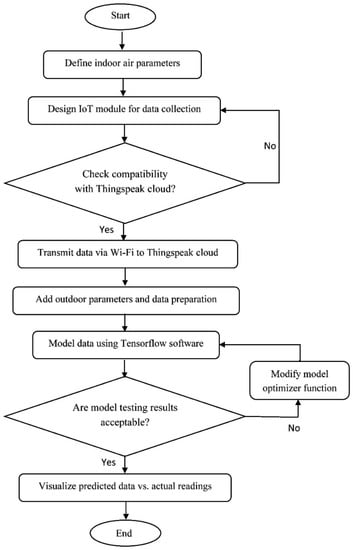
}

**DEVELOPMENT PART 1**

**Block diagram:**



**Flow Chart:**



**CODE:**

**# Import necessary libraries**

**import time**

**import board**

**import busio**

**import adafruit\_ccs811 # For the CCS811 CO2 sensor**

**import requests**

**# Set up the CCS811 sensor**

**i2c = busio.I2C(board.SCL, board.SDA)**

**ccs = adafruit\_ccs811.CCS811(i2c)**

**# ThingSpeak API key and endpoint**

**THINGSPEAK\_API\_KEY = "YOUR\_THINGSPEAK\_API\_KEY"**

**THINGSPEAK\_ENDPOINT = f"https://api.thingspeak.com/update?api\_key={THINGSPEAK\_API\_KEY}"**

**# Function to read sensor data and send it to ThingSpeak**

**def send\_to\_thingspeak():**

**try:**

**# Read sensor data**

**temperature = ccs.temperature**

**humidity = ccs.humidity**

**co2 = ccs.eco2**

**# Display data locally**

**print(f"Temperature: {temperature} C, Humidity: {humidity}%, eCO2: {co2} ppm")**

**# Send data to ThingSpeak**

**payload = {"field1": temperature, "field2": humidity, "field3": co2}**

**response = requests.post(THINGSPEAK\_ENDPOINT, params=payload)**

**# Print the response from ThingSpeak**

**print(f"ThingSpeak Response: {response.text}")**

**except Exception as e:**

**print(f"Error: {e}")**

**# Main loop**

**while True:**

**# Wait for the sensor to be ready**

**while not ccs.data\_ready:**

**pass**

**# Call the function to read sensor data and send to ThingSpeak**

**send\_to\_thingspeak()**

**# Wait for some time before the next reading**

**time.sleep(60) # Adjust the sleep duration based on your requirement**

**DEVELOPMENT PART 2**

**MODEL PERFORMANCE TESTING:**

The existing air quality monitoring systems generally monitor PM 2.5 along with temperature and humidity but the proposed system focuses on the multiple air pollutants including Carbon Dioxide (CO2), Particulate Matter (PM) 2.5, Nitrogen Dioxide (NO2), Carbon Monoxide (CO), Methane (CH4), temperature and humidity. The predictive analysis of air quality based on these pollutants make the proposed system more attractive and reliable as compared to the existing systems. • The web portal is developed which provide real time data of air quality which further helps to compute Air Quality Index (AQI), since time series data of 24 hour is required to compute AQI.

**SIMULATION:**

import random

import time

class AirQualitySimulator:

def \_\_init\_\_(self):

self.temperature = 25.0 # in Celsius

self.humidity = 50.0 # in percentage

self.co2\_level = 400 # in parts per million (ppm)

self.pm25\_level = 10 # in micrograms per cubic meter (µg/m³)

self.simulation\_duration = 60 # in seconds

def simulate(self):

start\_time = time.time()

while time.time() - start\_time < self.simulation\_duration:

# Simulate changes in air quality parameters

self.temperature += random.uniform(-1, 1)

self.humidity += random.uniform(-1, 1)

self.co2\_level += random.uniform(-1, 1)

self.pm25\_level += random.uniform(-1, 1)

# Ensure values stay within reasonable ranges

self.temperature = max(0, min(self.temperature, 40))

self.humidity = max(0, min(self.humidity, 100))

self.co2\_level = max(300, min(self.co2\_level, 1000))

self.pm25\_level = max(0, min(self.pm25\_level, 50))

# Display simulated data

self.display\_data()

time.sleep(1)

def display\_data(self):

print(f"Temperature: {self.temperature:.2f}°C | Humidity: {self.humidity:.2f}% | CO2 Level: {self.co2\_level} ppm | PM2.5 Level: {self.pm25\_level} µg/m³")

if \_\_name\_\_ == "\_\_main\_\_":

air\_quality\_simulator = AirQualitySimulator()

air\_quality\_simulator.simulate()

**SIMULATOR CODE :**

+-------------------+ +-------------------------+ +------------------+

| Simulation Script | ---> | Air Quality Simulator | ---> | Display/Logging |

| | | | | |

+-------------------+ +-------------------------+ +------------------+

|

v

+------------------+

| Data Store |

| (e.g., Database) |

+------------------+

**WEB TECHNOLOGIES :**

For building an indoor air quality monitoring system using IoT (Internet of Things), various web technologies can be employed to enable data communication, visualization, and user interaction. Here are some key web technologies commonly used in IoT projects:

1. **Web Development Frameworks:**
   * **Django or Flask (Python):** These frameworks can be used for building the backend of your web application. They provide tools for handling HTTP requests, managing databases, and integrating with IoT devices.
   * **Node.js (JavaScript):** Particularly suitable for projects with a large number of simultaneous connections, Node.js can be used to build a lightweight and efficient backend.
2. **Frontend Frameworks and Libraries:**
   * **React, Angular, or Vue.js:** These frameworks can be used for building dynamic and responsive user interfaces. They are especially useful for creating real-time dashboards and interactive visualizations of air quality data.
   * **Charting Libraries (e.g., Chart.js, D3.js):** Incorporate these libraries to create graphical representations of air quality metrics.
3. **Communication Protocols:**
   * **MQTT (Message Queuing Telemetry Transport):** A lightweight and efficient publish-subscribe messaging protocol that is commonly used in IoT applications for communication between devices and servers.
   * **HTTP/HTTPS:** Traditional web protocols can be used for communication between IoT devices and the server.
4. **Database Systems:**
   * **MySQL, PostgreSQL, MongoDB:** Choose a database system based on the requirements of your project. SQL databases (e.g., MySQL, PostgreSQL) are suitable for structured data, while NoSQL databases (e.g., MongoDB) can handle more unstructured data.
5. **IoT Platforms:**
   * **ThingSpeak, Blynk, Ubidots:** These IoT platforms provide tools and services to facilitate the integration of IoT devices, data storage, and visualization. They often offer APIs for connecting your devices and web application.
6. **Security Measures:**
   * **HTTPS (SSL/TLS):** Ensure secure communication between the IoT devices and the server.
   * **Token-based Authentication:** Implement secure authentication mechanisms to control access to the monitoring system.
7. **Cloud Services:**
   * **AWS IoT, Google Cloud IoT, Microsoft Azure IoT:** These cloud platforms provide scalable and reliable infrastructure for managing IoT devices, storing data, and hosting web applications.
8. **Containerization and Orchestration:**
   * **Docker, Kubernetes:** Use containerization for easy deployment and scaling of your web application.
9. **WebSockets:**
   * **Socket.io (JavaScript), Django Channels (Python):** Implement real-time communication between the server and the web browser for instant updates in air quality data.
10. **Mobile App Development (Optional):**
    * **React Native, Flutter:** If you plan to develop a mobile app for monitoring air quality, these frameworks can be used to create cross-platform applications.

Top of Form