ENVIRONMENT MONITORIN

USING INTERNET OF THINGS(IOT)

Submitted by

DHANUS MANI S 611821106011

THENNARASU G 611821106060

ABISHEK J 611821106003

LOKESHWARAN S 611821106028

NIRANJAN S 611821106038

BACHELOR OF ENGINEERING

IN

DEPARTMENT OF ELECTRONICS AND

COMMUNICATION ENGINEERING

P.S.V. COLLEGE OF ENGINEERING AND TECHNOLOGY

(Accredited by the NAAC with 'A' Grade)

(An ISO 9001:2015 Certified Institution)

KRISHNAGIRI

ANNA UNIVERSITY-CHENNAI 600 025

NOV/DEC-2023

INTRODUCTION

An Environmental Monitoring System using the Internet of Things (IoT) is a cuttingedge solution that leverages interconnected devices, sensors, and data analytics togather, manage, and analyze environmental data in real-time. This technology plays acrucial role in addressing environmental challenges, tracking the state of naturalresources, and ensuring a sustainable future. It can be applied in various contexts, fromurban areas to remote wilderness, helping to assess and manage environmentalconditions more effectively.

Overview:

The use of IoT in an EMS allows for the monitoring and control of variousenvironmental parameters, such as air quality, water quality, energy consumption, waste

management, and more. Sensors and devices can be deployed to collect data on these

parameters, which is then transmitted to a central system for analysis and action.

The real-time nature of IoT data allows organizations to respond quickly to

environmental incidents or deviations from set targets. For instance, if a sensor detects a

sudden increase in air pollution levels, an alert can be generated, enabling immediate

action to be taken to mitigate the issue.

In summary, incorporating IoT into an EMS provides organizations with the ability to

collect and analyze real-time environmental data, leading to improved environmental

performance, resource efficiency, and sustainability. It allows for proactive environmental

management, reduces costs, and enhances overall operational effectiveness.

VARIOUS TOOLS USED FOR THIS PROJECT :

Sensors: IoT environmental monitoring systems rely on a variety of sensors designed to

measure parameters like temperature, humidity, air quality, water quality, soil moisture,

radiation levels, and more. These sensors are strategically deployed to collect real-time data.

IoT Devices: These are the hardware components that house sensors, process data, and facilitate communication. Devices like Raspberry Pi, Arduino, or specialized IoT modules are commonly used to collect data from sensors and transmit it to central systems.

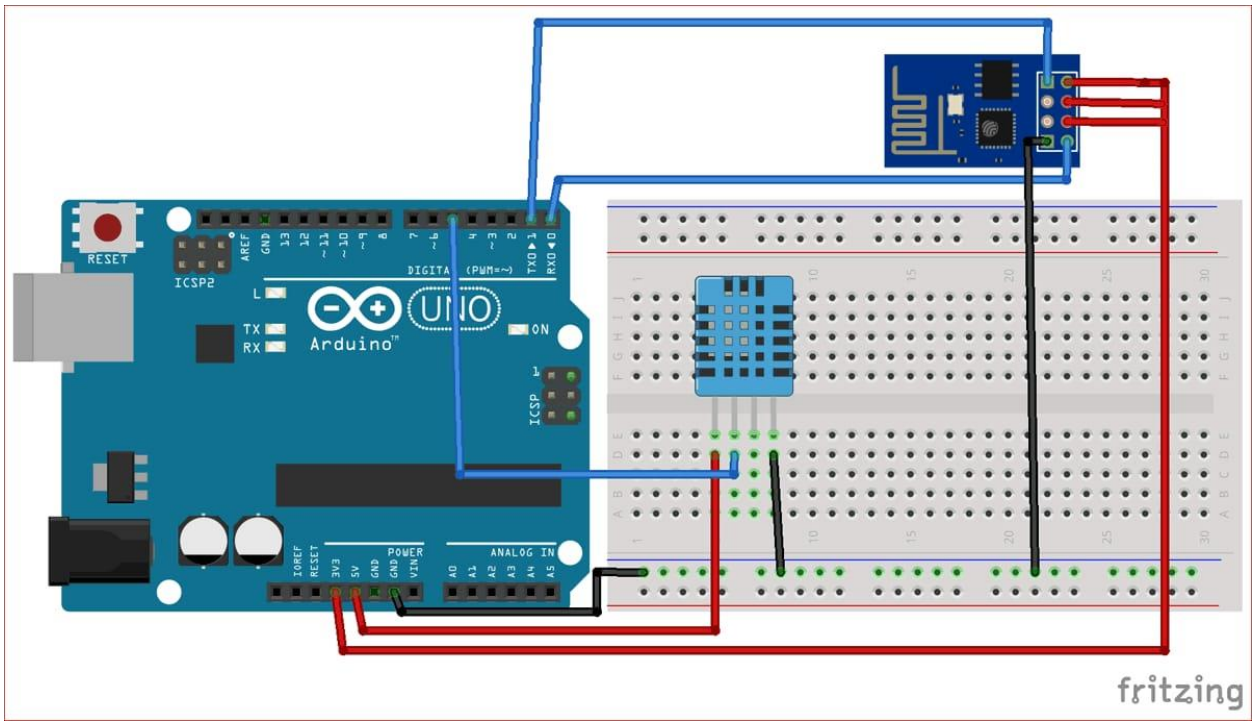
Communication Networks: Data from sensors is transmitted using various communication protocols, such as Wi-Fi, Bluetooth, LoRaWAN, Zigbee, or cellular networks. The choice of network depends on the specific application's range and data transfer requirements.

IoT Platform: Data collected by sensors is sent to an IoT platform or cloud service, such as AWS IoT, Google Cloud IoT, or Microsoft Azure IoT. These platforms provide storage, data processing, real-time monitoring, and visualization tools.

Data Processing and Analytics: The collected data is processed and analyzed to derivevaluable insights. Advanced analytics techniques may be used to detect trends, anomalies,and patterns within the data

Creating a real-time Environment Management platform involves a combination of front endand back end technologies. Here's a simplified outline using C and C++ and phythonprograming with wifi connection for the front end and Node.js for the back end.

DIAGRAM:



IOT environment monitoring is a process that uses inter net of things technology to collect data About the environment such as air qulity, temperature,and humidity levels.

Design thinking in environmental monitoring

Design thinking is a human-centered approach to problem-solving and innovation that can be applied to environmental monitoring in parks. It involves empathizing with the needs of stakeholders, defining the problem, ideating creative solutions, prototyping, and testing. Here's how design thinking can be applied to improve environmental monitoring in parks:

1.Empathize:

Understand the diverse stakeholders involved, including park managers, conservationists, scientists, visitors, and local communities. - Conduct interviews, surveys, and field observations to gain insights into their needs, concerns, and expectations related to environmental monitoring. - Identify the specific environmental challenges and goals of the park.

2. Define:

Clearly define the problem or challenge you want to address through environmental monitoring. For example, it could be improving water quality, mitigating the impact of invasive species, or reducing visitor disturbance to wildlife. - Create user personas to represent the different stakeholders and their goals. - Develop a problem statement that encapsulates the challenge and its impact on the park's ecosystem.

3. Ideate:

Organize brainstorming sessions with a cross-functional team that includes park rangers, scientists, designers, and community members. - Generate a wide range of ideas for innovative monitoring solutions. Encourage creativity and explore both high-tech and low-tech options. - Use techniques like mind mapping, storyboarding, or idea clustering to structure and refine ideas.

4. Prototype:

Create low-fidelity prototypes of the proposed monitoring solutions. These could be simple models, mock-ups, or diagrams. - Experiment with different technologies and approaches to monitoring, considering factors like costeffectiveness, scalability, and ease of implementation. - Seek feedback from stakeholders on the prototypes to refine and improve the concepts.

5. Test:

Pilot test the prototypes in a real park environment. Start with a small-scale implementation to gather data and assess the effectiveness of the monitoring solution. - Collect feedback from park staff, scientists, and visitors about the usability, accuracy, and practicality of the solution. - Adjust and iterate the prototype based on the feedback received.

6. Implement:

Once a monitoring solution has been refined and proven effective through testing, develop a plan for full-scale implementation in the park. - Secure necessary resources, including funding, equipment, and trained personnel. - Train park staff and volunteers on the new monitoring protocols and technologies.

7 Evaluate and Iterate:

Continuously monitor and evaluate the effectiveness of the environmental monitoring system. - Collect and analyze data to track changes in the park's ecosystem and assess the impact of monitoring efforts. - Be open to making adjustments and improvements based on ongoing feedback and evolving park conditions.

8. Communicate and Educate:

Share the results of environmental monitoring with the public, stakeholders, and local communities through reports, presentations, and interpretive programs.Use storytelling and educati monitoring and conservation. onal materials to engage park visitors in the importance of environmental Design thinking can help ensure that environmental monitoring in parks is not only effective but also responsive to the needs and values of the people who use and care for these natural spaces. It encourages a holistic and usercentered approach to solving complex environmental challenges.

ENVIRONMENTAL MONITORING INNOVATION

Design steps for environmental monitoring

1. Define the Monitoring System:

Clearly define what environmental parameters you want to monitor (e.g., temperature, humidity, air quality).

2. Select Sensors:

Choose appropriate sensors for each parameter. Ensure they are compatible with Tinkercad's available components

. 3. Design the Circuit:

Use Tinkercad's Circuit Editor to design the electronic circuit. Connect sensors to microcontrollers (e.g., Arduino) and add any additional components needed (resistors, capacitors, etc.

4. Programming Microcontrollers:

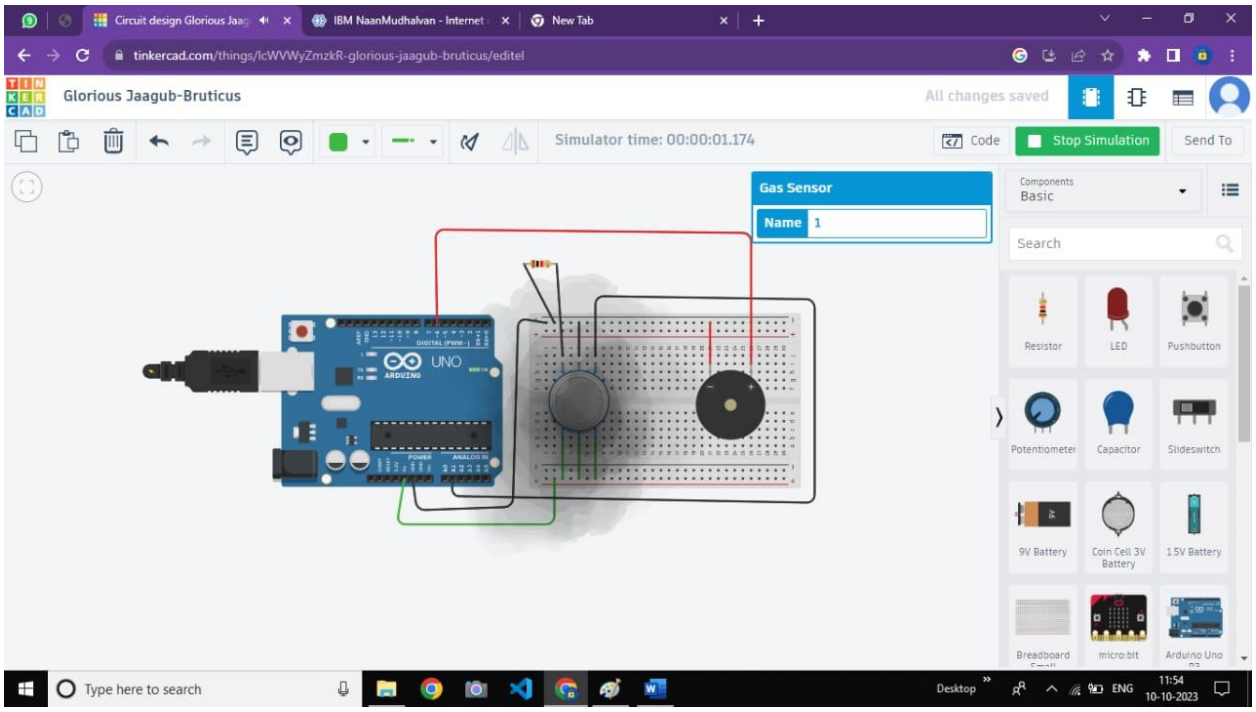
Write the code to read data from sensors and transmit it to a data storage system. Use Arduino programming language or other compatible languages

. 5. Simulate the Circuit:

Use Tinkercad's simulation tools to test the circuit virtually. This allows you to identify and fix any potential issues before building the physical prototype.

6. 3D Design and Printing (Optional):

If needed, design system to protect the components. and 3D print an enclouse for your monitoring system to protect the components



//C++code

//

/*

Envirnoment monitor

*/

int valueofgassensor = 0;

void setup()

pinMode(A1, INPUT);

Serial.begin(9600);

```
pinMode(6, OUTPUT);
 }
 void loop()
 {
 // Gas sensor with buzzer
valueofgassensor = analogRead(A1);
Serial.println(valueofgassensor);
if (valueofgassensor > 200) {
tone(6, 523, 1000); // play tone 60 (C5 = 523 Hz)
 }
 delay(10); // Delay a little bit to improve simulation performance
}
```

7. Assemble the Prototype:

Connect the actual sensors, microcontroller, and any other components according to the design.

8. Upload Code to Microcontroller:

Upload the code you wrote to the microcontroller using Tinkercad's programming interface.

9. Test t he Prototype:

Ensure that the sensors are providing accurate data and that the microcontroller is transmitting it correctly.

10. Data Visualization (Optional):

MmIntegrate a display or connect the prototype to a computer for realtime visualization data

Web Platform Development:

Describe the web platform you will develop. Include details about its functionalities, features, and the technologies you'll use for web development.ENVIRONMENT MONITORING

ENVIRONMENT MONITORING

Mobile App Development:

To connect your environmental monitoring IoT project with a mobile app, you'll need to establish a communication link between your IoT devices or sensors and the mobile app. Here are the steps to connect your IoT project to a mobile app

Choose the Right Communication Protocol:

You need to select a communication protocol that your IoT devices and the mobile app can both support. Some common protocols for IoT communication include MQTT, HTTP/HTTPS, CoAP, and WebSockets. Choose the one that fits your project's requirements.

Set Up a Cloud Platform: Thoroughly test the mobile app to ensure that it can successfully communicate with the IoT devices and cloud platform. Debug any issues that arise during the testing phase.

Deployment:

Publish the mobile app on app stores (e.g., Apple App Store and Google Play Store) for users to download and install.

User Training and Documentation:

Provide users with training and documentation on how to use the mobile app and understand the environmental data it presents.

Maintenance and Updates:

Regularly maintain and update both the mobile app and the IoT devices' firmware to address bugs, add new features, and improve security.

Monitoring and Analytics:

Implement analytics and monitoring tools to track app usage and IoT device performance. This can help you identify issues and optimize the system.

Program:

Creating a Python program for an environmental monitoring IoT project that connects to a mobile app involves several steps. I'll provide an example Python code snippet for a simplified scenario to get you started. This example assumes you have a cloud-based IoT platform for data storage and retrieval. You can adapt this code to your specific project requirements.

Code:

```
import paho.mqtt.client as mqtt

import json


# MQTT configuration

mqtt_broker = "your-mqtt-broker.com"
```

```
mqtt_port = 1883
mqtt_topic = "environmental_data"

# Replace with your authentication credentials
username = "your_username"
password = "your_password"

# Create an MQTT client
client = mqtt.Client()
client.username_pw_set(username, password)

# Callback function when connected to MQTT broker
def on_connect(client, userdata, flags, rc):
    print("Connected to MQTT Broker with result code " + str(rc))
    # Subscribe to the topic
    client.subscribe(mqtt_topic)

# Callback function when a message is received
def on_message(client, userdata, msg):
    payload = json.loads(msg.payload.decode())
    # Process the received data
    temperature = payload["temperature"]
    humidity = payload["humidity"]
    air_quality = payload["air_quality"]
    # You can process the data further or send it to the mobile app
```

```
# Set callback functions
client.on_connect = on_connect
client.on_message = on_message

# Connect to MQTT broker
client.connect(mqtt_broker, mqtt_port, 60)

# Start the MQTT client loop
client.loop_start()

# Your code to send data to the mobile app
# You can use a library like Flask to create a REST API or use a WebSocket library to send data in real-time.

# In a real-world scenario, you would have additional logic to process and send data to the mobile app.

# Keep the script running
try:
    while True:
        pass
except KeyboardInterrupt:
    client.disconnect()
    print("Disconnected from MQTT Broker")
```

Thoroughly test the mobile app to ensure that it can successfully communicate with the IoT devices and cloud platform. Debug any issues that arise during the testing phase.

Deployment:

Publish the mobile app on app stores (e.g., Apple App Store and Google Play Store) for users to download and install.

User Training and Documentation:

Provide users with training and documentation on how to use the mobile app and understand the environmental data it presents.

Maintenance and Updates:

Regularly maintain and update both the mobile app and the IoT devices' firmware to address bugs, add new features, and improve security.

Monitoring and Analytics:

Implement analytics and monitoring tools to track app usage and IoT device performance. This can help you identify issues and optimize the system.

Program:

Creating a Python program for an environmental monitoring IoT project that connects to a mobile app involves several steps. I'll provide an example Python code snippet for a simplified scenario to get you started. This example assumes you have a cloud-based IoT platform for data storage and retrieval. You can adapt this code to your specific project requirements.

Code:

```
import paho.mqtt.client as mqtt

import json


# MQTT configuration
```

```
mqtt_broker = "your-mqtt-broker.com"
mqtt_port = 1883
mqtt_topic = "environmental_data"

# Replace with your authentication credentials
username = "your_username"
password = "your_password"

# Create an MQTT client
client = mqtt.Client()
client.username_pw_set(username, password)

# Callback function when connected to MQTT broker
def on_connect(client, userdata, flags, rc):
    print("Connected to MQTT Broker with result code " + str(rc))
    # Subscribe to the topic
    client.subscribe(mqtt_topic)

# Callback function when a message is received
def on_message(client, userdata, msg):
    payload = json.loads(msg.payload.decode())
    # Process the received data
    temperature = payload["temperature"]
    humidity = payload["humidity"]
    air_quality = payload["air_quality"]
```

```
    # You can process the data further or send it to the mobile app


# Set callback functions
client.on_connect = on_connect
client.on_message = on_message

# Connect to MQTT broker
client.connect(mqtt_broker, mqtt_port, 60)

# Start the MQTT client loop
client.loop_start()

# Your code to send data to the mobile app
# You can use a library like Flask to create a REST API or use a WebSocket library to send data in real-time.

# In a real-world scenario, you would have additional logic to process and send data to the mobile app.

# Keep the script running
try:
    while True:
        pass
except KeyboardInterrupt:
    client.disconnect()
    print("Disconnected from MQTT Broker")
```

ENVIROENMTEL MONITORING-DEVELOPMENT PART 1

Design a environmental monitoring system based on the esp32 using wokwi application can be a complex, but I can provide you with a high-level overview steps involved

1. Define Objectives and Requirements:

Clearly state the purpose of your environmental monitoring system. In this case, you want to monitor temperature and humidity levels.

2.Select Sensors:

Choose sensors capable of measuring temperature and humidity. Common choices include DHT11, DHT22, or similar sensors

3.Select Microcontroller:

- Choose a microcontroller compatible with

3.Set Up the Wokwi Environment:

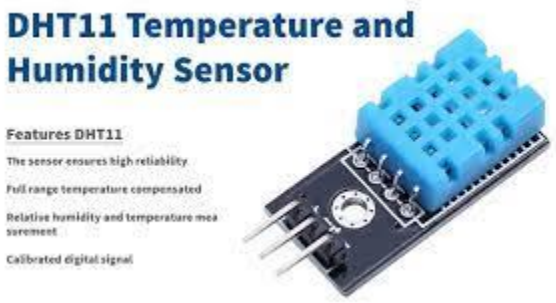
Launch the Wokwi platform and select the microcontroller you've chosen. Open a new project.

4.Connect Sensors to Microcontroller:

- Wire the temperature and humidity sensor to the microcontroller. Refer to the sensor's datasheet and ensure the correct pins are used.

5.Power Supply:

Ensure a stable power supply for your microcontroller and sensors. This could be achieved through batteries, USB power, or other sources depending on your specific application.



6.Write the Code:

- Use the Arduino IDE or the Wokwi simulator's built-in code editor to write the code. Utilize a library compatible with your chosen sensor (e.g., DHT library for DHT sensors).

7.Code Functionality:

Write code to read data from the sensor and display it. You can choose to display the information on a virtual display in the simulator or even transmit it to a remote server if needed.

8.Test and Debug:

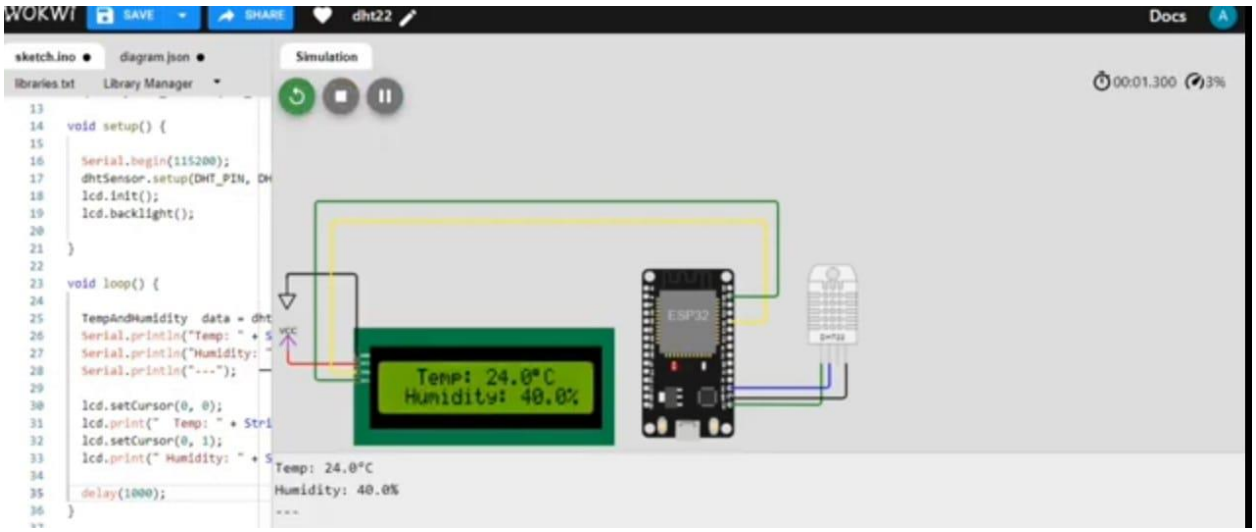
• Simulate the project in the Wokwi environment to ensure the sensors are providing accurate readings and the microcontroller is processing the data correctly.

. 9 Monitor and Fine-tune: •

Observe the system's performance in the actual environment. Make any necessary adjustments to improve accuracy or reliability

10. Optional: Data Storage and Visualization:

• If desired, you can add features to store and visualize the collected data. This might involve using additional components or connecting to external platforms



CODING

```
#include "DHTesp.h"

 #include<liquidcrystel.12c.h>

#define 12c _ADDER 0*27

 #define LCD _COLUMNS 20

 #define LCD_LIMES 4

 Const int DHT_PIN=15;
```

```
DHTesp dhtsensor;
liquidcrystel_12c lcd(12c_ADDER, LCD_COLUMNS,LCD_LINES);
VOID setup()
{
Serial.BEGIN(115200);
dhtsensor.setup(DHT_PIN,DHTesp::DHT22);
lcd.init(); lcd.backlight();
}
void loop()
{
tempAndHumidity data = dhtsensor.getTempHumidity();
Serial.println("temp: " + string(data.temperature,1)+"c");
Serial.println("Humidity: " + string(data.Humidity,1)+"%");
Serial.println("----"); lcd.setcursor(0,0);
lcd.print("temp: " + string(data.temperature,1)+"\xDF"+"c"):
lcd.println(0,1);
Serial.println("Humidity: " + string(data.Humidity,1)+"%");
lcd.print("wokwi online IOT"):
s delay(10000);
}
```

ENVIROEMENTAL MONITORINGDEVELOPMENT PART 2

Design a Enviromental Monitoring system based on Inventor , it following steps

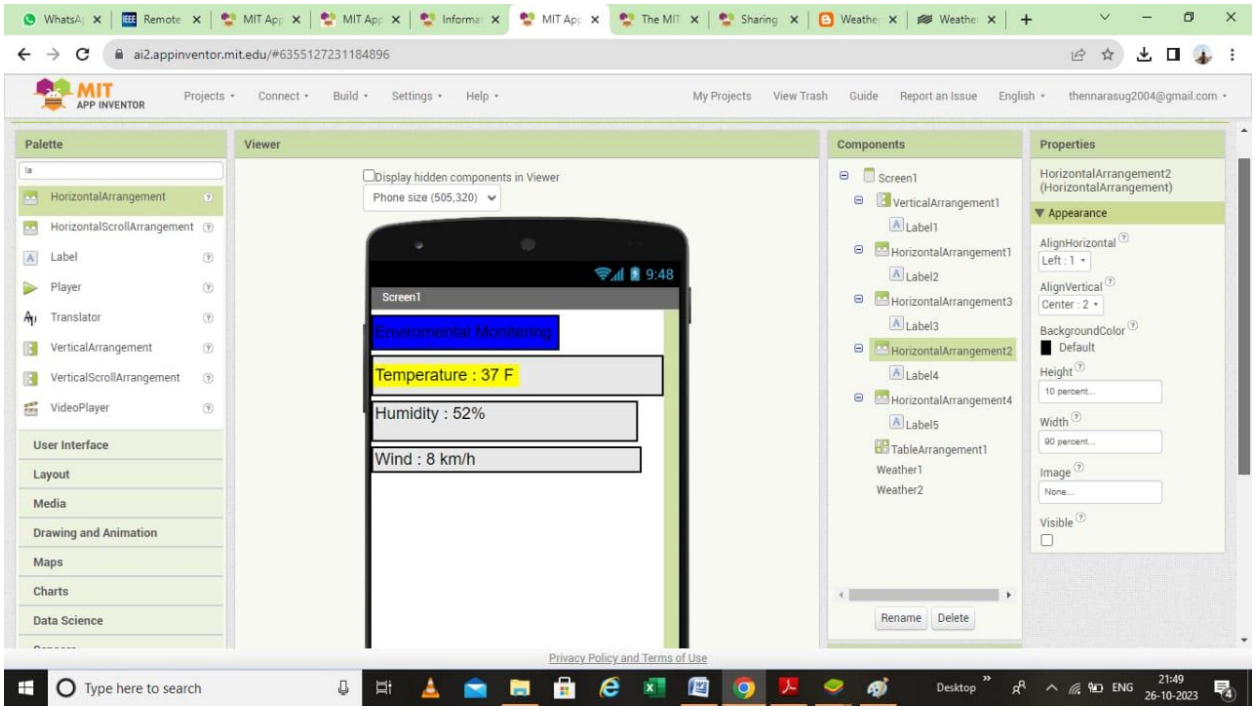
1. Designing the App Interface:
   • Use MIT App Inventor's dragandmobile application using MIT drop interface to create screens temperature, humidity, and wind monitoring.
   • Add labels, text boxes, buttons, an for d any other necessary component.

2. Sensor Integration:

Connect appropriate sensors to your microcontroller or development board (e.g., Arduino, Raspberry Pi). • Write cod e to read data from the sensors and transmit it to MIT App Inventor.

2. Bluetooth/WiFi Communication:

   • Establish a communication protocol (Bluetooth or Wi microcontroller and your Android device. • Ensure that the microcontroller can sendFi) between the data in…



In M IT App Inventor, create blocks to parse the incoming data and display it on the respective screens for temperature, humidity, and wind.

5. Web Development for Data Visualization:

• You'll need a server to host a webpage for data visualization. You can use platforms like AWS, Google Cloud, or even a local server. Create a webpage with HTML, CSS, and JavaScript to display the environmental data. Use libraries

6. Data Transmission to Web Server: •

Modify your microcontroller code to also send data to the web suitable format (e.g., using HTTP POST requests). server in a

7. Web Page Interaction with Data:

Use JavaScript to fetch data from your server and update the visualizations in real

8. Security and Authentication (Optional):

• Implement security meth measures like HTTPS, API keys, or other authentication ods to protect data transmission.

9. User Interface Refinement:

• Enhance the app's user interface for a polished look and feel.

10. Testing and Debugging:

• Thoroughly test the entire system to ensure that data is transmitted accurately and visualized correctly both in the app and on t he web page.

11 . User Documentation (Optional):

Create a simple guide on how to use the app.

12. Deployment:

• Once you're satisfied with export options. the app, you can package it for Android devices using MIT App Inventor's

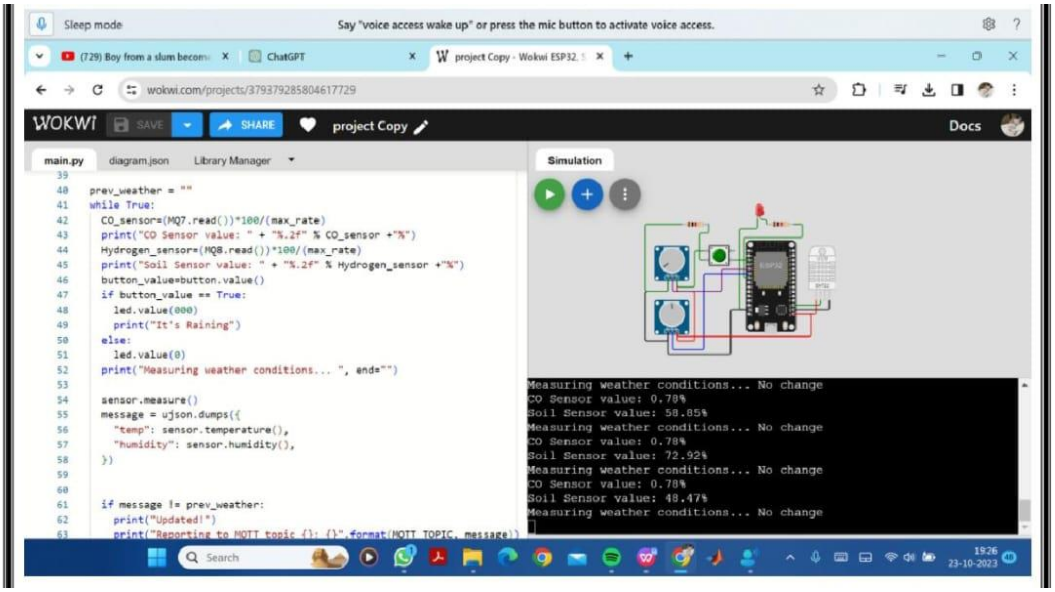
C++ CODE

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <Adafruit_MQTT.h>
#include <Adafruit_MQTT_Client.h>

// Replace these with your Wi-Fi credentials.
const char* WIFI_SSID = "YourWiFiSSID";
const char* WIFI_PASS = "YourWiFiPassword";
```

```
// Replace with your Adafruit IO credentials.
#define ADAFRUIT_IO_USERNAME "YourAdafruitUsername"
#define ADAFRUIT_IO_KEY "YourAdafruitAIOKey"

// Define the DHT sensor.
#define DHT_PIN 2 // The pin where your DHT sensor is connected.
#define DHT_TYPE DHT22 // DHT sensor type (DHT11, DHT22, AM2302, etc.)

DHT dht(DHT_PIN, DHT_TYPE);

WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, "io.adafruit.com", 1883, ADAFRUIT_IO_USERNAME,
ADAFRUIT_IO_KEY);

// Define MQTT feeds.
Adafruit_MQTT_Publish temperature = Adafruit_MQTT_Publish(&mqtt, ADAFRUIT_IO_USERNAME
"/feeds/temperature");
Adafruit_MQTT_Publish humidity = Adafruit_MQTT_Publish(&mqtt, ADAFRUIT_IO_USERNAME
"/feeds/humidity");

void setup() {
Serial.begin(115200);

// Connect to Wi-Fi.
WiFi.begin(WIFI_SSID, WIFI_PASS);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.println("Connecting to WiFi...");
}

Serial.println("Connected to WiFi");
// Connect to Adafruit IO.
mqtt.connect();
Serial.println("Connected to Adafruit IO");
}

void loop() {
// Read temperature and humidity data from the DHT sensor.
float temperatureValue = dht.readTemperature();
float humidityValue = dht.readHumidity();

// Publish data to Adafruit IO.
```

```
if (!isnan(temperatureValue)) {
temperature.publish(temperatureValue);
Serial.print("Temperature: ");
Serial.println(temperatureValue);
} else {
Serial.println("Failed to read temperature");
}
if (!isnan(humidityValue)) {
humidity.publish(humidityValue);
Serial.print("Humidity: ");
Serial.println(humidityValue);
}
else {
Serial.println("Failed to read humidity");
}

delay(60000); // Delay for 60 seconds (adjust as needed).
}
```

RESULT



Conclusion:

Environmental Monitoring System using the Internet of Things(IoT) represents a transformative and highly

valuable technology for addressing awide range of environmental challenges. This system harnesses the power ofinterconnected sensors, devices, and data analytics to collect, manage, and analyzeenvironmental data in real-time.