Peri Institute of technology

Project title: Environmental monitoring system.

Batch member: S. Abinash, Abdul majith, R. Gokul, D. Gokul.

Department: B.E computer science engineering.

Describe the project's objectives, IoT sensor deployment, platform and mobile app development, and code implementation:

Project objectives:

1.Assessing Environmental Quality: Monitoring to determine the state of environmental parameters such as air and water quality, soil conditions, and biodiversity.

2.Detecting Pollution: Identifying and quantifying pollutants to ensure compliance with environmental regulations and standards.

3.Early Warning: Providing early warning of environmental threats such as natural disasters or industrial accidents.

Iot sensor deployment:

1.Select Sensors: Choose appropriate IoT sensors based on your objectives. Ensure they can collect and transmit data reliably in the target environment.

2.Sensor Placement: Position sensors strategically in the target locations to capture relevant data accurately. Consider factors like altitude, proximity to pollution sources, or areas prone to environmental changes.

3.Power Supply: Ensure sensors have a stable power source, which could be batteries, solar panels, or energy harvesting methods, depending on the location.

4.Data Analysis: Implement data analytics and visualization tools to process the collected data. This allows for real-time monitoring and historical analysis.

Platform development:

1.Architecture Design: Plan the software architecture. Decide if it will be a cloud-based platform or an on-premises solution. Consider the scalability, data storage, and processing components.

2.Sensor Integration: Develop APIs or protocols to connect with IoT sensors and collect data. Ensure compatibility with various sensor types and communication protocols.

3.Data Ingestion: Create a data ingestion pipeline to collect data from sensors. This might involve real-time streaming or batch data collection, depending on the use case.

4.Data Storage: Design a database schema or storage solution to store collected data securely and efficiently. Consider factors like data retention policies and disaster recovery.

Mobile app development:

1.Feature Set: Define the core features of the app, which may include real-time data visualization, historical data access, customizable alerts, map integration, and user notifications.

2.Alerts and Notifications: Implement a system for customizable alerts and notifications, allowing users to set thresholds for specific environmental parameters.

3.Privacy and Data Security: Implement robust data security measures to protect sensitive environmental data and user information.

Code implementation:

Sensor Data Collection:

Interface with IoT sensors to collect data.

Implement protocols (e.g., MQTT, HTTP, LoRa) for sensor communication.

Ensure real-time or periodic data collection as needed.

Data Processing:

Develop algorithms for data cleaning and preprocessing.

Apply statistical or machine learning techniques for data analysis.

Calculate averages, outliers, and trends in environmental data.

Data Storage:

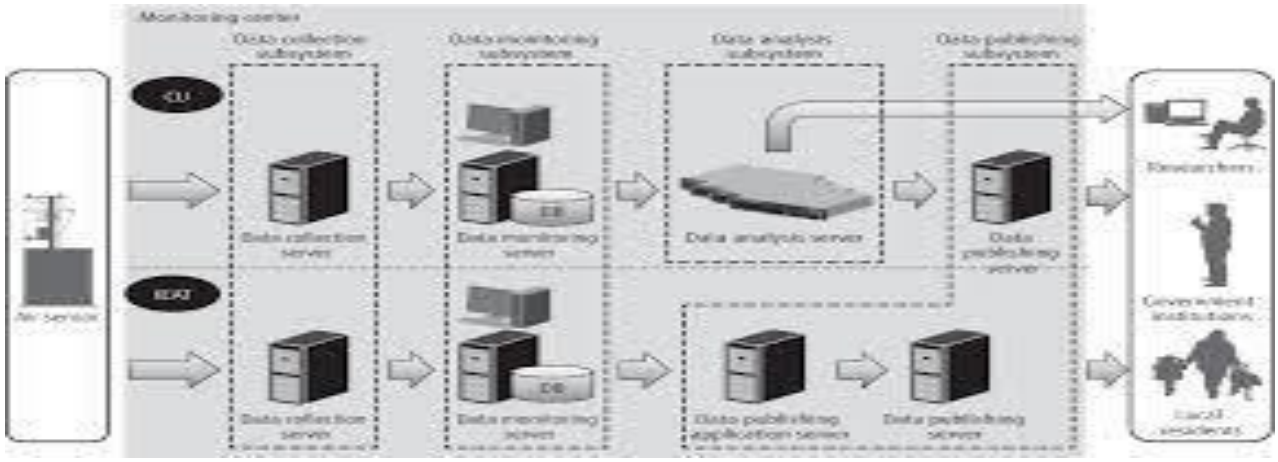
Design a database schema or choose appropriate data storage solutions (e.g., SQL databases, NoSQL databases, or time-series databases).

Store collected data securely and efficiently.

Implement data retention policies for historical data.

Include diagrams, schematics, and screenshots of the IoT sensors, environmental monitoring information Platform, and mobile app interfaces:

1.Mapping for environmental monitoring:



2.enviornmental monitoring system:

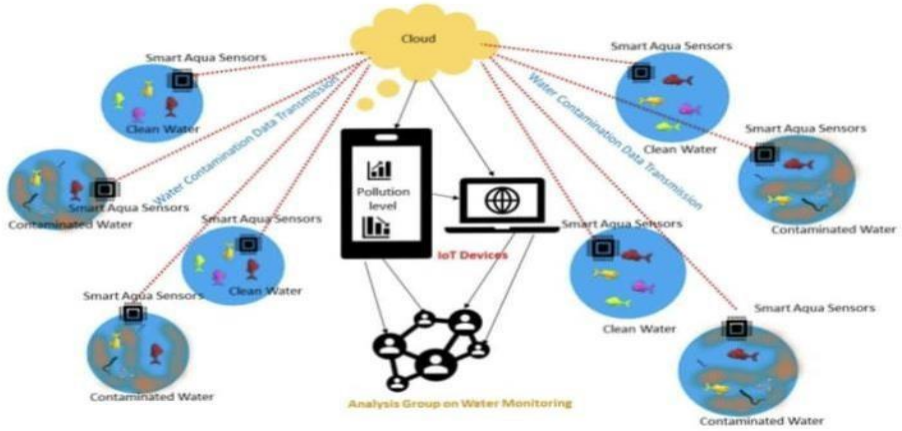
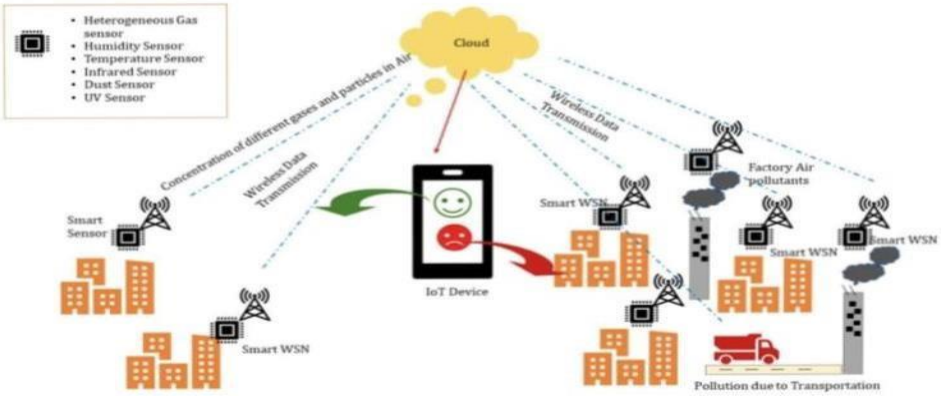
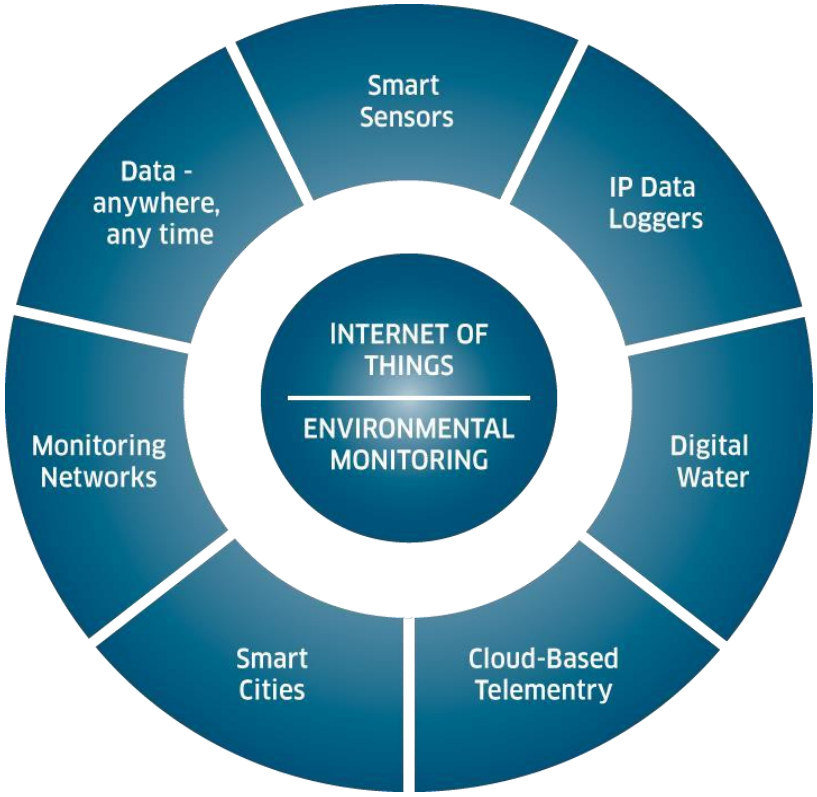


Figure 1. Smart environment monitoring (SEM) system highlighting water contamination and its monitoring using the cloud connecting internet of things (IoTs) and sensors.



3.Overcomes for environmental monitoring systems



IoT Sensors Deployment Schematic:

IoT Sensors: These are the physical devices that collect environmental data. Specify the types of sensors you plan to use (e.g., temperature, humidity, air quality, water quality, etc.).

Sensor Nodes: These are often equipped with microcontrollers and wireless communication modules. Show how sensor nodes are distributed across the monitored area.

Cloud Platform: Data received from the gateway is stored and processed in the cloud. Visualize the cloud platform and data storage.

User Interface: This can be a web application or a mobile app. Show how users will interact with the system to access real-time and historical data.

Environmental monitoring system Information Platform Diagram:

Data Sources: Start with icons or symbols representing the various environmental sensors collecting data.

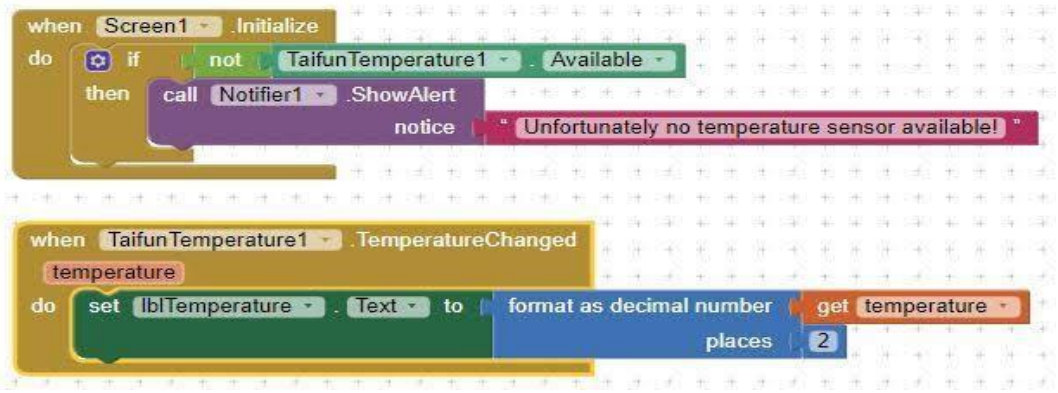
Data Processing: Represent data processing components like servers, algorithms, and analytics tools within the cloud platform.

Mobile App User Interface Screenshots:

1.Capture screenshots of the mobile app's user interface to showcase its functionality and design.

2.Include screens displaying real-time data, historical data, location filtering, and notifications.

3.Highlight the app's user-friendly features and visual elements.



Explain how the real-time environmental monitoring system promotes public awareness and contributes to temperature and humidity level mitigation:

Data Accessibility: By providing real-time data on temperature and humidity levels, the system allows the public to access up-to-date information. This information empowers individuals and communities to make informed decisions about their activities and adapt to current environmental conditions.

Public Alerts: The monitoring system can issue alerts when extreme temperature or humidity conditions are detected. These alerts can be sent to mobile devices or displayed on websites, helping people take precautionary measures to stay safe and comfortable.

Education: The system can serve as an educational tool by providing information on the impacts of temperature and humidity levels on health, agriculture, and the environment. This can increase public understanding and awareness of these issues.

Behavioral Changes: Real-time data can influence public behavior. For example, when people are aware of high temperatures, they may reduce energy consumption by using air conditioning more efficiently, conserving water, or limiting outdoor activities during peak heat.

Energy Efficiency: Awareness of temperature and humidity levels can encourage the public to use energy-efficient appliances and practices, which can reduce energy consumption and, in turn, lower greenhouse gas emissions contributing to temperature rise.

Urban Planning: City officials and urban planners can use the data to make informed decisions regarding urban development and green infrastructure. This can include designing green spaces, planting trees, and improving city layouts to mitigate temperature and humidity effects.

Mitigation Strategies: Public awareness of environmental data can drive the adoption of mitigation strategies, such as implementing cool roof programs, urban forestry initiatives, or water conservation measures to combat high temperatures and humidity.

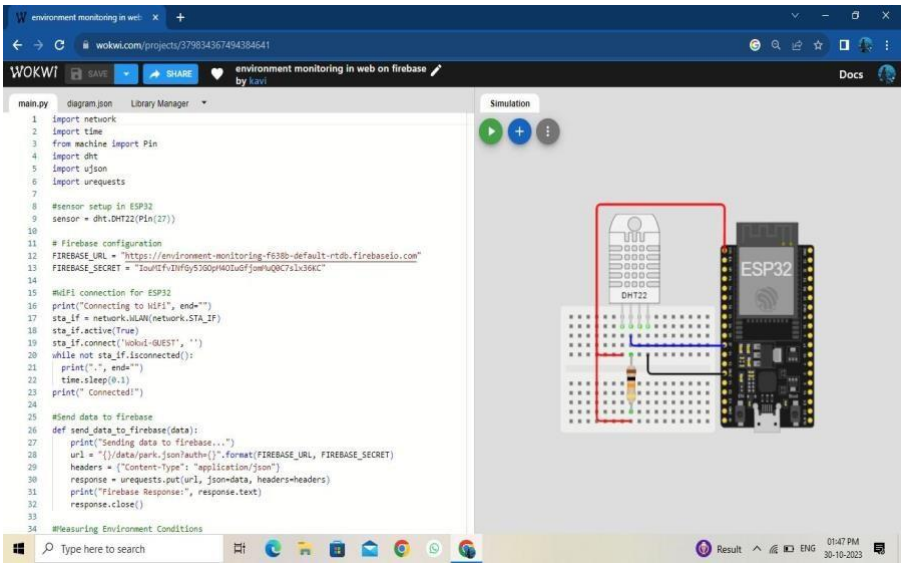
Community Engagement: Real-time environmental data can foster community engagement and collaborative efforts to address temperature and humidity-related challenges. Local organizations and residents can work together to implement solutions and share best practices.

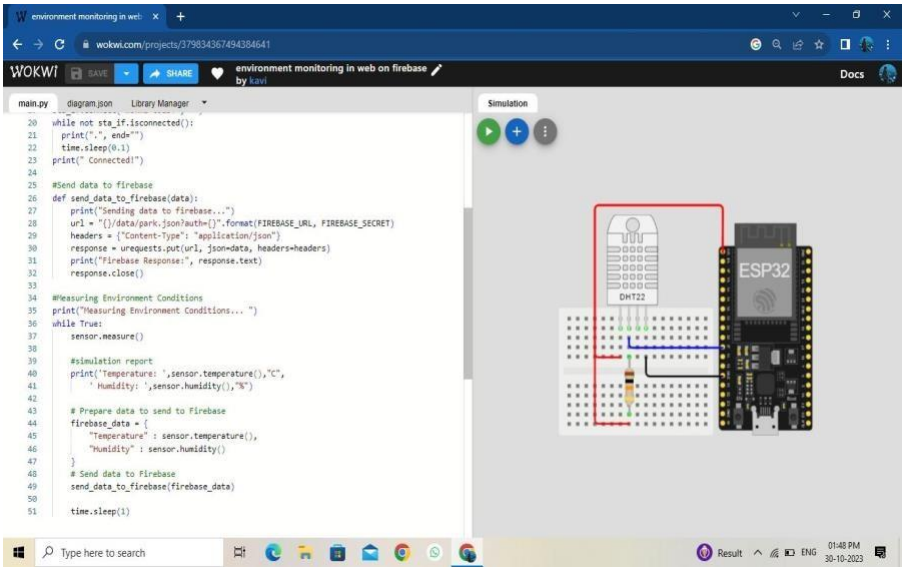
Positive Impact on Quality of Life:

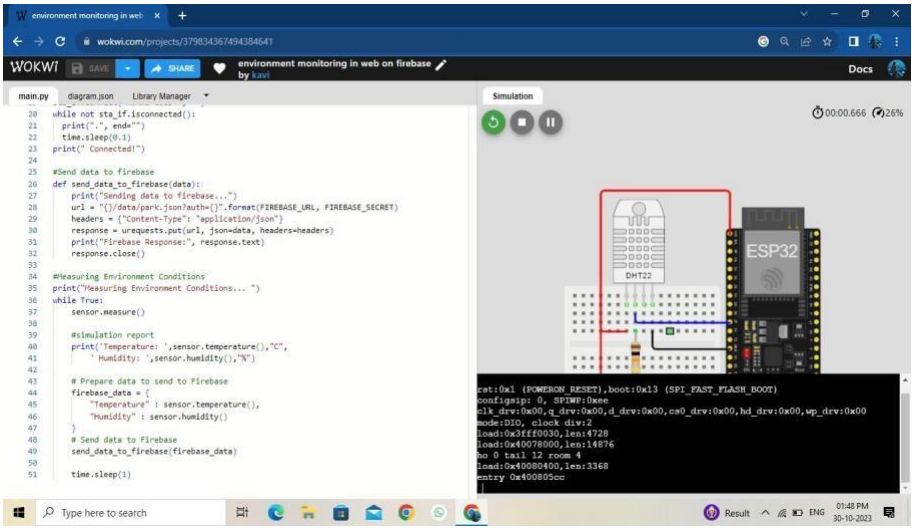
Environmental monitoring systems have far-reaching positive impacts, from protecting human health and ecosystems to mitigating climate change and promoting sustainable development. They are vital tools for a healthier, more sustainable planet.

Wowki stimilation:

<Wowki online stimulations for custom type temperature sensor in environmental monitoring systems>







Program used to create this:

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")
```

<u>Thank you.</u>