VELAMMAL INSTITUTE OF TECHNOLOGY
Approved by AICTE New Delhi, & Affiliated to Anna University Chennai
CSE,ECE,EEE,IT,MECH Accredited by NBA & NAAC
"Velammal Knowledge Park", Chennai – Kolkatta Highway, Ponneri – 601 204

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROJECT NAME: ENVIRONMENT MONITORING

TEAM NAME: Proj_224786_Team_2

TEAM MEMBERS:

SANNAREDDY ANUPAMA (113321104084)

SHAIK RESHMA (113321104089)

SHALINI M (113321104090)

SHANTHI G (113321104092)

Phase2 : Innovation

Consider integrating historical environmental data and machine learning algorithms to predict congestion patterns.

PROJECT OBJECTIVES:-

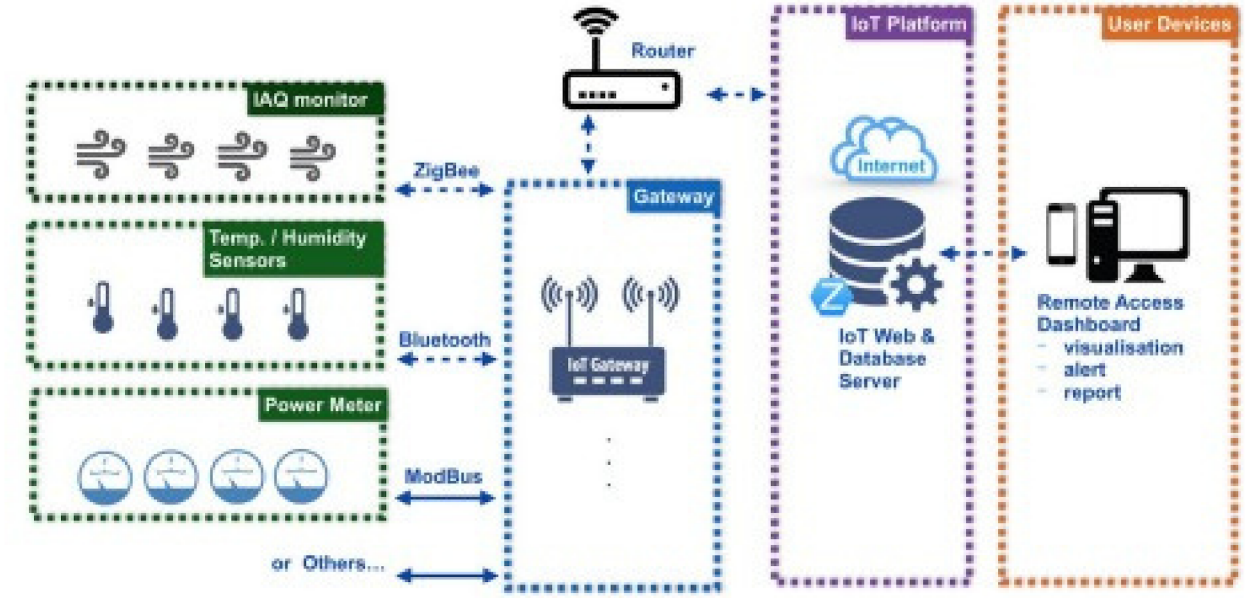
Effective environmental monitoring is crucial for preserving our planet and addressing pressing issues like climate change, pollution, and habitat degradation. The following objectives guide a comprehensive environmental monitoring project:

- **Data Accuracy and Precision** : Ensure the collection of highly accurate and precise data by employing cutting-edge sensors and measurement techniques, reducing errors in environmental data that inform policy and decision-making.
- **Real-time Monitoring** : Develop a system capable of continuous, real-time data acquisition and transmission, enabling prompt responses to environmental changes and potential hazards.
- **Long-term Trend Analysis:** Establish a robust database for long-term trend analysis, allowing us to identify patterns and anticipate potential environmental issues, making the project a valuable resource for future generations.

· **Community Engagement:** Promote public awareness and participation in environmental monitoring by creating user-friendly interfaces and interactive platforms, fostering a sense of collective responsibility.

· **Policy Influence:** Utilize the collected data and insights to influence and inform local and global environmental policies, driving positive changes towards a sustainable and resilient future.

IoT SENSOR SETUP:-



Step 1: Sensor Selection

Weather Sensors:

Weather stations employ sensors like anemometers, barometers, and thermometers to measure parameters such as wind speed, air pressure, and temperature. These sensors provide data for weather forecasting and climate studies.

Air Quality Sensors:

Gas sensors like particulate matter (PM) sensors and gas detectors are used to monitor air quality, detecting pollutants such as carbon monoxide, ozone, and fine particulate matter, aiding in pollution control and public health management.

Water Quality Sensors:

Water quality sensors measure parameters like pH levels, dissolved oxygen, turbidity, and nutrient concentrations. These sensors are crucial for assessing the health of water bodies, ensuring safe drinking water, and managing aquatic ecosystems.

Soil Sensors:

Soil moisture sensors and soil pH sensors are used to monitor soil conditions. This data is valuable for precision agriculture, irrigation management, and assessing soil health for sustainable land use.

Remote Sensing Devices:

Satellites equipped with remote sensing instruments capture data on a global scale, monitoring changes in land cover, vegetation health, and climate patterns. These sensors are essential for studying large-scale environmental changes and natural disasters.

Step 2: Data Collection

Data collection for environmental monitoring is critical for understanding and addressing various environmental challenges. Here are three key methods of data collection:

Sensor Networks:

Deploying sensor networks, including ground-based sensors and remote sensing technologies like satellites and drones, allows for continuous and real-time data collection. These sensors measure various environmental parameters, such as temperature, air quality, water quality, and more, providing comprehensive insights into the state of the environment.

Sampling and Laboratory Analysis:

Environmental scientists and researchers collect samples of air, water, soil, and biological specimens from specific locations. These samples are then analyzed in laboratories to measure pollutant concentrations, nutrient levels, and other critical parameters. This method is essential for in-depth, highly accurate analysis and can validate data from sensor networks.

Step 3: Data Processing

Data processing is a crucial step in environmental monitoring to make sense of the collected information and draw meaningful insights. Here are three key data processing methods used in environmental monitoring:

- Environmental monitoring often involves multiple sources of data from various sensors and platforms. Data fusion and integration techniques combine data from different sources and formats to create a unified dataset.
- Data analysis involves statistical techniques and algorithms to identify trends, patterns, and anomalies in environmental data. Environmental scientists use data modeling to create predictive models that help in forecasting future environmental conditions
- GIS technology is widely used in environmental monitoring to visualize and analyze spatial data. It allows for mapping environmental variables, assessing their spatial distribution, and identifying areas of concern.

Step 4: Visualization and Control

To make sense of the environmental data, user-friendly interfaces are essential. This can encompass environmental monitoring dashboards, mobile applications, and even smart signage. These tools assist environmental managers in informed decision-making and communication with stakeholders and the public.

Raspberry Pi integration

Step 1: Understanding Raspberry Pi

Raspberry Pi is a small, affordable computer that we can use to enhance our traffic management system. It's like a tiny brain that can help us collect and process data from various sensors. Here's how we can integrate Raspberry Pi into traffic management:

Step 2: Sensor Integration

Initially, we need to link our Raspberry Pi to the environmental sensors. These sensors might include the ones previously discussed, such as cameras, soil moisture sensors, or air quality monitors. This connection is established by interfacing the sensors with the Raspberry Pi, either through GPIO pins or USB connections. For instance, if we're using air quality monitors, we connect them to
the Raspberry Pi via USB ports, while soil moisture sensors can be linked to the GPIO pins.

Step 3: Data Collection

With the Raspberry Pi successfully interfaced with the environmental sensors, data collection can commence. For environmental cameras, it captures images or videos of the surroundings. In the case of soil moisture sensors, it records soil moisture levels and trends. The acquired data is then stored in the Raspberry Pi's memory for further analysis and processing

Step 4: Data Processing

After acquiring the data, the Raspberry Pi can initiate data processing. Programs written in languages like Python can be utilized to analyze the data. For instance, we can employ image recognition algorithms to identify environmental changes in captured images or calculate relevant metrics like pollutant concentrations from sensor data.

Step 5: Data Storage

To safeguard crucial environmental data, we can establish connections between the Raspberry Pi and external storage solutions, such as external hard drives or cloud storage services. This approach ensures that we maintain a secure backup of all the collected environmental data.

Mobile app development

Step 1: User Interface Design:

Now, let's focus on the user interface (UI) design, a vital aspect as our application should prioritize user-friendliness. We'll craft screens for functionalities like real-time environmental data visualization, incident reporting, and navigation features, ensuring an intuitive and accessible experience for users.

Step 2: Data Integration:

To deliver real-time environmental information, our app will integrate with data sources like environmental sensors, satellite data, and official environmental agencies. This integration guarantees our users receive precise and current data on environmental conditions.

Step 3: Core Features Development:

The essence of our app centers around its core functionalities:

- Real-Time Environmental Data: We'll create the capability to gather and present real-time environmental information on a map, covering factors like air quality, weather conditions, and pollutant levels.

- Environmental Navigation: We'll integrate GPS navigation with up-to-the-minute environmental updates, offering guidance on healthier or safer routes based on environmental conditions.

- Incident Reporting: Users will have the ability to report environmental incidents, such as pollution sources or ecological concerns, by submitting photos and attaching location tags.

Code implementation

Step 1: Data Collection

Read data from the connected sensors at specified intervals. The data collection process can be triggered periodically, at set times, or in response to external events, depending on your monitoring requirements.

```
import time

sensor_pin = 18  # Replace with the GPIO pin connected to your sensor

try:
    while True:
        with open("/sys/class/gpio/gpio{}/value".format(sensor_pin), "r") as
            sensor_data = sensor_file.read()
            print("Sensor Data: " + sensor_data.strip())
        time.sleep(10)  # Collect data every 10 seconds (adjust as needed)
except KeyboardInterrupt:
    print("Data collection stopped.")
```

Step 2: Data Processing

- Analyze and preprocess the collected data. This step may involve filtering, calibration, and data validation to ensure the quality and accuracy of the information before further processing.

```python
import pandas as pd

# Load data from a CSV file
data = pd.read_csv('environmental_data.csv')

# Data processing
# Example: Calculate the average temperature
average_temperature = data['Temperature'].mean()

# Example: Filter data for pollution levels above a certain threshold
high_pollution_data = data[data['Pollution'] > 50]

# Example: Normalize data
normalized_data = (data['SensorReading'] - data['SensorReading'].mean()) / 

# Example: Calculate the daily maximum temperature
daily_max_temperature = data.resample('D')['Temperature'].max()

# Save processed data to a new CSV file
processed_data.to_csv('processed_environmental_data.csv', index=False)
```

Step 3: Data Storage

Decide where and how to store the collected data. You can save it in memory, on an external device (e.g., an SD card), or transmit it to a remote server or cloud-based storage for long-term retentio

```python
import time

# Sample function to read sensor data (replace this with your actual sensor
def read_sensor_data():
    # Replace this with your sensor reading logic
    return "Sample Sensor Data"

try:
    while True:
        data = read_sensor_data()

        # Append the data to a text file
        with open("environmental_data.txt", "a") as file:
            timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
            file.write(f"{timestamp}: {data}\n")

        time.sleep(60)  # Store data every minute

except KeyboardInterrupt:
    print("Data storage stopped.")
```

n and analysis.

Step 4: Data Visualization

If needed, create user-friendly interfaces or visualization tools to display real-time or historical environmental data. This step is crucial for making the data accessible to users or stakeholders.

```python
import matplotlib.pyplot as plt
import random

# Simulated environmental data
time = list(range(1, 11))
temperature = [random.uniform(20, 30) for _ in time]

# Create a line plot
plt.plot(time, temperature, marker='o')
plt.xlabel('Time (hours)')
plt.ylabel('Temperature (°C)')
plt.title('Environmental Temperature Monitoring')
plt.grid(True)
plt.show()
```

Step 5: Building a Machine Learning Model

Now, let's create and train a machine-learning model. In this example, we'll use a simple decision tree regressor.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load your environmental dataset (X) and corresponding labels (y)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

# Create a Random Forest Classifier
model = RandomForestClassifier()

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)

print("Model Accuracy:", accuracy)
```

Step 6: Data Transmission and Reporting

Implement mechanisms for transmitting data to remote servers or generating reports. This step enables sharing environmental data with relevant parties, such as researchers, government agencies, or the public, for informed decision-making.

```
import requests

# Environmental data to be transmitted
data = {
    'temperature': 25.5,
    'humidity': 60.2,
    'pollution_level': 35.8
}

# Remote server URL to post the data
url = "https://example.com/api/environmental_data"

# Send data to the server
response = requests.post(url, json=data)

if response.status_code == 200:
    print("Data transmitted successfully")
else:
    print("Failed to transmit data")
```