IOT BASED ENVIRONMENTAL MONITORING SYSTEM

A Project report submitted in partial fulfilment of the

Requirement for the degree of B-E in Computer science and

Engineering

By

U.DHANUSHKUMAR (513221104304)

PHASE 5: project Documentation& submission

In this section you will document the complete project and prepare it for submission.

Under the supervision of professor & HOD department of

Computer science and Engineering .

INTRODUCTION:

- Environmental monitoring describes the processes and activities that need to take place to characterize and monitor the quality of the environment.

- Environmental monitoring is used in the preparation of environmental impact assessments, as well as in many circumstances in which human activities carry a risk of harmful effects on the natural environment.

- All monitoring strategies and programs have reasons and justifications which are often designed to establish the current status of an environment or to establish trends in environmental parameters.

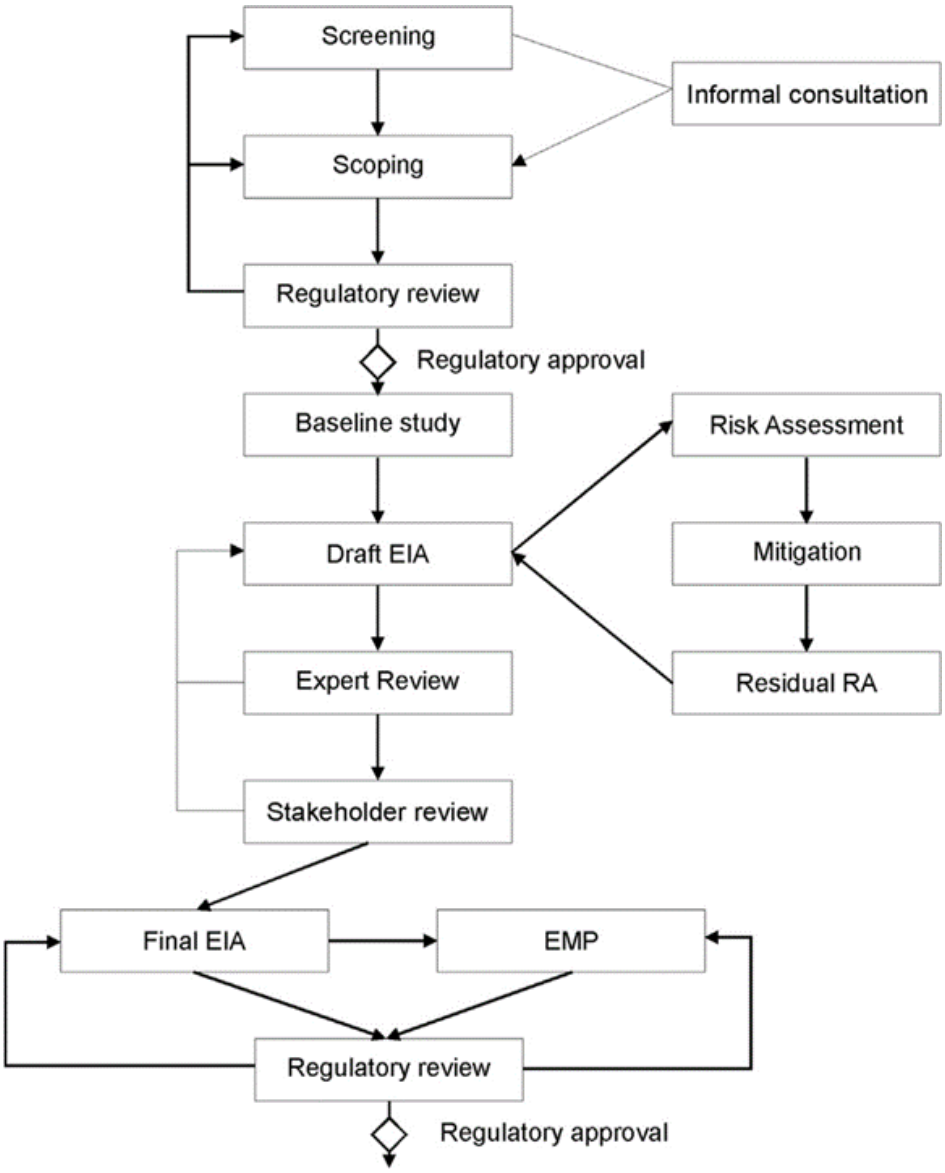
PROBLEM DEFINITION AND DESIGN THINKING:

- ✓ PROBLEM STATEMENT
- ✓ DESIGN THINKING APPROACH

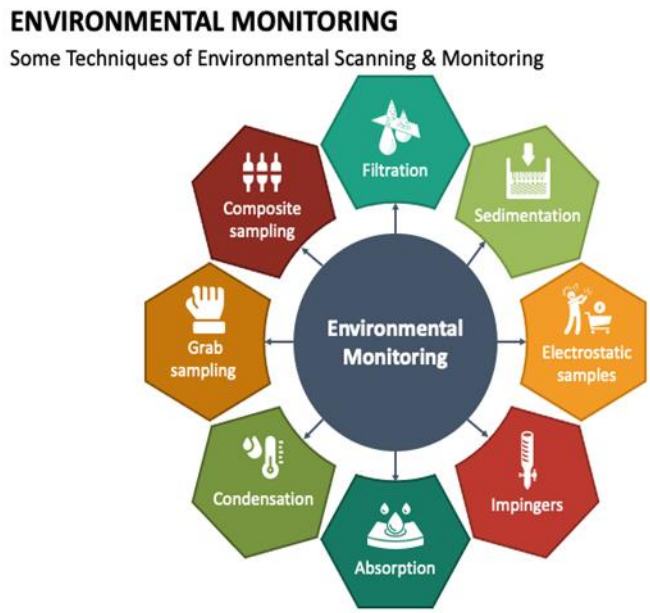
PROBLEM STATEMENT:

That is, the problem statement is quite focused. In general, it has to talk about three aspects regarding the problem: the current state, the desired or ideal state, and the solutions or suggestions you will make through your research that will help attain this ideal state.

- ➢ FLOWCHART:



- **DESIGN THINKING APPROACH**



Design thinking is **an iterative, non-linear process which focuses on a collaboration between designers and users**. It brings innovative solutions to life based on how real users think, feel and behave. This human-centered stages design process consists of five core Empathize, Define, Ideate, Prototype and Test.

INNOVATION IN ENVIRONMENTAL MONITORING WITH REMOTE SENSING TECHNIQUES:

➢ BACKGROUND/OBJECTIVES:

Large or remote areas of land are often challenging and expensive to monitor using traditional ground-based methods. Remote monitoring techniques (i.e., satellite and drone imagery) is becoming a prevalent part of environmental monitoring and characterization. Recent developments in computer vision and artificial intelligence, combined with knowledge in ecological characterization, allow for the rapid analysis of large volumes of rapidly compiled remote sensing data for relevant signals and can provide unparalleled site understanding. These

new data sources and methods applied to age old challenges around groundwater seep identification and contaminated site management require a range of demonstrated use cases. This presentation will focus on a brief introduction to how satellites, drones, and cloud-computing based artificial intelligence are changing the way environmental monitoring takes place. Following the technical introduction into these technologies, we will present a case study of remote sensing techniques utilized at a large (several hundred square kilometer) alumina refinery in which the goal of the project was to understand environmental impacts sitewide and protecting sensitive habitat. The purpose of Ramboll's work with this project was to help the client rapidly assess the changing conditions of vegetation at its facility, and to better understand the impact of specific variables.

APPROACH/ACTIVITIES:

To accomplish this, Rambolt's Galago team analyzed a variety of data sources including high-resolution satellite and aerial imagery to document sitewide trends related to vegetation health and tree dieback. A temporal analysis of satellite imagery was completed to evaluate the region-wide impact of climatic variables, such as drought, on vegetation health using the Normalized Difference Vegetation Index (NDVI). High-resolution aerial imagery and a deep learning model were used to identify specific locations of tree dieback not visible in satellite imagery. Aerial imagery was analyzed using a convolutional neural network model to classify sections of the imagery as tree dieback. Multiple captures per year of aerial imagery allows for site-wide dieback monitoring throughout the year and help capture changes throughout the site. The analysis produced from this project enables an additional line of evidence to support other site investigation and monitoring activities and creates the possibility to deploy an advanced habitat monitoring system in an accurate, repeatable, and cost-effective manner.

➢ RESULTS/LESSONS LEARNED:

The success of this project is in its ability to quickly garner information about site wide trends in vegetation, to supplement and focus ground-based investigations, and to provide an ability to compare the site with reference areas. The satellite imagery analysis provided a region-wide view of vegetative health and showed that the site was impacted by drought in a similar way to reference areas. The aerial imagery analysis showed that it's possible to quickly train and deploy a deep learning model for analysis of vegetation in high-resolution imagery. The advantage of creating a model to identify dieback is that it can be scaled across large areas and applied to new imagery as it becomes available. Overall, the client can now better understand drought impacts on sensitive habitats and will have the ability to measure vegetation dieback rates across the entire site and reference areas. Having this information allows a more efficient and effective implementation of remedial plans, and minimizes environmental impacts to sensitive habitats.

Types of Environmental Monitoring:

- ✓ Air Quality Monitoring
- ✓ Water Sampling and Analysis
- ✓ Noise level Testing
- ✓ Soil Quality Testing
- ✓ Energy monitoring

❖ Air Quality Monitoring:

Industrial processes emit organic compounds like carbon monoxide, hydrocarbons, and chemicals ("greenhouse gases") into the air. And as we know, exhaust from vehicles and methane from cattle impact the quality of our air and impact our planet.

***With air quality monitoring, science and industry can create change*. These critical industrial operations to mitigate their impact, and for entire auto makers to continually improve designs to reduce emissions. Even deploying IoT to manage traffic flow in metrics deliver the insights for municipalities to make decisions for urban planning, for cities can massively reduce vehicle emissions and support cleaner air.**

Some real-world examples of air quality monitoring include:

- ✓ **Carbon monoxide monitoring in homes and buildings**
- ✓ **Methane monitoring in agriculture and waste management.**

❖ WATER QUALITY MONITORING:

Water is a vital source for the health of the planet and its people, and today, technology is needed to support clean water management and conservation. Water quality monitoring using IoT-based systems helps to control contamination and support management of this valuable resource. Using IoT systems allows water to be analyzed in buildings, water and wastewater plants, irrigation systems and industrial processes.

These advanced smart water monitoring systems using IoT technologies enable accurate measurements of contaminants, oxygen levels, additional factors, and pH levels. IoT technology allows the detection of harmful substances public it reaches homes and buildings. The innovative technology helps us to sustain our health and wellness.
Some examples include:

- ✓ Municipal water treatment monitoring.
- ✓ Storm water and groundwater monitoring.
- ✓ Agricultural irrigation monitoring and control.

City water and drinking water quality monitoring.

❖ ENERGY MONITORING:

With our limited global energy resources, energy monitoring is essential to conservation. IoT-based technologies can provide both the management tools and the insights to improve how we use energy.

Leading energy providers today are rapidly integrating a wide range of IoT monitoring and mitigation techniques to reduce usage, as well as clean energy solutions to reduce energy consumption and promote sustainability. In the process, these techniques can also save money for everyone relying on the electric grid.

Energy monitoring supports numerous *energy management goals*:

- ✓ **Reduction in the use of fossil fuels in homes and businesses.**
- ✓ **Stabilizing the power grid.**
- ✓ **Preventing spikes in energy usage, and associated equipment failures and service disruption.**

❖ SOIL QUALITY MONITORING:

Soil quality defines whether soils are in good condition for their current land use activity. Because soils have developed from different parent materials and have been influenced by a range of soil-forming factors, soils display a variety of physical, chemical and biological characteristics.

❖ NOICE OUALITY MONITORING

Noise monitoring refers to the systematic process of measuring, recording, and assessing sound levels in various environments to understand the extent of noise pollution and its potential impact on human health and the surrounding ecosystem.

PART1: DEVELOPMENT OF ENVIRONMENTAL MONITORING

1.Collect Data:

You'll need to interface with environmental sensors to gather data. For this example, let's assume you have a temperature and humidity sensor connected to your Raspberry Pi.

Input:

```
pip ins import time
import board
import adafruit_dht

dht_sensor = adafruit_dht.DHT22(board.D4)  # GPIO pin where the sensor is connected

while True:
    try:
        temperature_c = dht_sensor.temperature
        humidity = dht_sensor.humidity
```

```
        print(f"Temperature: {temperature_c}°C, Humidity: {humidity}%")
    except RuntimeError as e:
        print(f"Error: {e}")
    time.sleep(60)  # Collect data every 60 secondstall adafruit-circuitpython-dht
```

Create a Python script to collect sensor data:

Output:

Temperature: 25.0°C, Humidity: 50.0%

Temperature: 25.1°C, Humidity: 49.9%

Temperature: 25.2°C, Humidity: 50.2%

2. Data Processing and Analysis:

You can perform data analysis on the collected data to identify trends or anomalies. For this example, let's calculate the average temperature and humidity over a specific time period.

INPUT:

```
import time

data = []

while True:
    try:
```

```
        temperature_c = dht_sensor.temperature
        humidity = dht_sensor.humidity
        data.append((temperature_c, humidity))
        time.sleep(60)
    except RuntimeError as e:
        print(f"Error: {e}")

    if len(data) >= 10:
        avg_temp = sum([temp for temp, _ in data]) / len(data)
        avg_humidity = sum([hum for _, hum in data]) / len(data)
        print(f"Average Temperature: {avg_temp}°C, Average Humidity:
{avg_humidity}%")
        data = []  # Reset data
```

OUTPUT:

Average Temperature: 25.0°C, Average Humidity: 50.0%

PART 2: DEVELOPMENT OF ENVIRONMENTAL MONITORING SYSTEM:

Data Collection:

- **Gather relevant environmental data,**
- **which might include temperature,**
- **humidity, air quality, and other factors.**
- **Ensure data quality, clean the data, and handle missing values if necessary.**

Feature Engineering:

- Extract meaningful features from the collected data.
- This may involve time-series analysis.
- statistical calculations, or domain-specific knowledge.
- Normalize or scale features to ensure they have a consistent range.

Model Selection:

- Choose an appropriate machine learning model for your specific task.
- For environmental monitoring.
- time-series models like LSTM or traditional regression models may be suitable.

Data Splitting:

- Split your dataset into training, validation, and test sets.
- This helps you assess your model's performance.

Model Training:

- Train your selected model on the training data using Python libraries like scikit-learn orTensorFlow/Keras for deep learning.
- Tune hyperparameters to optimize model performance, e.g., learning rates, batch sizes, or network architectures.

Model Evaluation:

- Evaluate the model on the validation set using appropriate metrics such as Mean Absolute
  Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE).
- Visualize the results to gain insights.
  Fine-Tuning and Validation:
- Make necessary adjustments to the model based on the validation results.
- Repeat the training and evaluation steps if needed.

Testing:

- Assess the model's performance on the test set to ensure it generalizes well to unseen
  data.

Deployment:

- **If the model meets your requirements, deploy it in a production environment for real-time monitoring.**
- **You can use libraries like Flask or Django to create a web application.**

Monitoring:

- **Continuously monitor your deployed model to ensure it remains accurate and up-to-date.**
- **Reporting and Visualization.**
- **Create reports and dashboards to communicate results to stakeholders.**
- **Python libraries like Matplotlib, Seaborn, or Plotly can help with data visualization.**

Documentation:

- **Maintain documentation for your project, including code comments, README files, and model documentation.**

For example:

1. Collect Data:

- **You'll need to interface with environmental sensors to gather data.**
- **For this example, let's assume you have a temperature and humidity sensorconnected to your Raspberry Pi.**

INPUT:

```
pip ins import time
import board
import adafruit_dht
```

dht_sensor = adafruit_dht.DHT22(board.D4) # GPIO pin where the sensor is connected

while True:

try:

temperature_c = dht_sensor.temperature

humidity = dht_sensor.humidity

print(f"Temperature: {temperature_c}°C, Humidity: {humidity}%")

except RuntimeError as e:

print(f"Error: {e}")

time.sleep(60) # Collect data every 60 secondstall adafruit-circuitpython-dht

Create a Python script to collect sensor data:

OUTPUT:

Temperature: 25.0°C, Humidity: 50.0%

Temperature: 25.1°C, Humidity: 49.9%

Temperature: 25.2°C, Humidity: 50.2%

2. Data Processing and Analysis:

- You can perform data analysis on the collected data to identify
- trends or anomalies. For this example, let's calculate the average.

INPUT:

import time

data = []

while True:

try:

temperature_c = dht_sensor.temperature

```
humidity = dht_sensor.humidity
data.append((temperature_c, humidity))
time.sleep(60)
except RuntimeError as e:
print(f"Error: {e}")
if len(data) >= 10:
avg_temp = sum([temp for temp, _ in data]) / len(data)
avg_humidity = sum([hum for _, hum in data]) / len(data)
print(f"Average Temperature: {avg_temp}°C, Average Humidity:
{avg_humidity}%")
data = [] # Reset data
```

OUTPUT:

```
Average Temperature: 25.0°C, Average Humidity: 50.0%
```