ENVIRONMENTAL MONITORING

Project objective:

The problem is the need for effective environmental monitoring to address various environmental challenges such as climate change, pollution, biodiversity loss, and resource depletion. Environmental monitoring involves collecting, analyzing, and interpreting data related to the natural environment to make informed decisions and take appropriate actions. The challenge lies in developing a comprehensive and efficient monitoring system that can provide accurate, real-time, and actionable data to support environmental conservation and sustainable practices.

1. Empathize:

- Understand the stakeholders: Identify the key stakeholders, including government agencies, environmental organizations, scientists, and the general public, and understand their needs and concerns related to environmental monitoring.

- User research: Conduct surveys, interviews, and workshops to gather insights into the specific environmental issues and data requirements of different user groups.

2. Define:

- Problem statement: Clearly define the problem by synthesizing the insights gained during the empathize phase. For example, "How might we create a scalable and user-friendly environmental monitoring system to address climate change and pollution effectively?"

- Identify constraints: Consider budget limitations, technological constraints, and regulatory requirements that may impact the design.

3. Ideate:

- Brainstorm solutions: Encourage creative thinking to generate a wide range of ideas for monitoring systems and tools.

- Prioritize ideas: Evaluate and prioritize the ideas based on their feasibility, potential impact, and alignment with user needs.

4. Prototype:

- Create a prototype of the environmental monitoring system: Develop a simplified version of the system to test and iterate upon.

- Test with users: Gather feedback from stakeholders and users to refine the prototype and make necessary improvements.

5. Test:

- Pilot testing: Implement a small-scale pilot project to assess the effectiveness of the monitoring system in a real-world setting.

- Collect feedback: Continuously gather feedback from users and stakeholders during the pilot phase to identify any issues or improvements needed.

6. Implement:

- Scale up: If the pilot is successful, plan for the full-scale implementation of the environmental monitoring system.

- Collaborate: Partner with relevant organizations and agencies to ensure data sharing and cooperation.

7. Evaluate:

- Monitor impact: Continuously assess the impact of the monitoring system on environmental awareness, policy decisions, and positive changes in behavior.

- Iterate: Use the feedback and data collected to make ongoing improvements to the system.

8. Communicate:

- Share results: Communicate the findings and results of the environmental monitoring system with the public, policymakers, and other stakeholders to raise awareness and drive positive environmental action.

Design thinking is an iterative process, and it's essential to revisit and refine the environmental monitoring system regularly to adapt to changing environmental challenges and technological advancements.

SETTING IOT SENSOR SETUP:

Setting up an IoT (Internet of Things) sensor network for environmental monitoring typically involves deploying various sensors to collect data on parameters such as air quality, temperature, humidity, water quality, and more. These sensors are equipped with connectivity features, allowing them to transmit real-time data to a central monitoring system.

Here's a simplified diagram of the IoT sensor setup:

Sensor

              |

              |

           Wi-Fi

              |

              |

          Central

          Server

SENSORS FOR VIRTUAL ENVIRONMENT

1. \*\*Temperature Sensor:\*\* This sensor is designed to gauge fluctuations in temperature within the environment, a fundamental aspect of climate change monitoring.

2. \*\*Humidity Sensor:\*\* Humidity sensors specialize in measuring moisture levels in the air, rendering them invaluable for tracking humidity's influence on agriculture and meteorological phenomena.

3. \*\*Air Quality Sensor:\*\* These sensors are engineered to detect key air pollution parameters, encompassing particulate matter (PM2.5 and PM10), carbon monoxide (CO), and volatile organic compounds (VOCs). They assume a pivotal role in monitoring air quality.

4. \*\*Light Sensor:\*\* Light sensors are adept at quantifying ambient light levels, proffering insights into daylight patterns, plant growth cycles, and the repercussions of light pollution.

5. \*\*Sound Sensor:\*\* These sensors adeptly capture ambient noise levels, facilitating the surveillance of noise pollution and its ramifications for both wildlife and human communities.

6. \*\*Gas Sensors:\*\* Tailored to your specific needs, gas sensors possess the capability to identify and quantify distinct gases such as methane, ozone, or nitrogen dioxide.

7. \*\*GPS Module:\*\* Should geographic precision be imperative, a GPS module can furnish precise geographical coordinates.

8. \*\*Water Quality Sensors:\*\* For the meticulous examination of bodies of water, sensors are equipped to evaluate critical parameters such as pH levels, dissolved oxygen concentrations, turbidity, and electrical conductivity.

1. \*\*Soil Moisture Sensor:\*\* These sensors prove instrumental in agricultural applications and soil health assessments by measuring soil moisture content.
2. \*\*Motion Sensors:\*\* In specialized instances, motion sensors like Passive Infrared (PIR) sensors can discern the presence of animals or humans, serving purposes ranging from wildlife monitoring to security applications.

ENVIRONMENTAL MONITORING USING RASPBERRY PI:

Environmental monitoring using a Raspberry Pi involves setting up sensors to track various parameters such as temperature, humidity, air quality, and more. By connecting these sensors to the Raspberry Pi, you can collect, process, and visualize the data. This setup allows you to monitor and analyze the environmental conditions in real time, enabling proactive measures for maintaining optimal conditions and ensuring sustainability.

1. Hardware setup: Connect sensors (e.g., temperature, humidity, and air quality) to your Raspberry Pi.

2. Software installation: Install necessary libraries for sensor data collection and processing.

3. Data collection: Configure the Raspberry Pi to collect real-time data from the connected sensors.

4. Data processing: Write scripts to process the collected data for analysis and visualization.

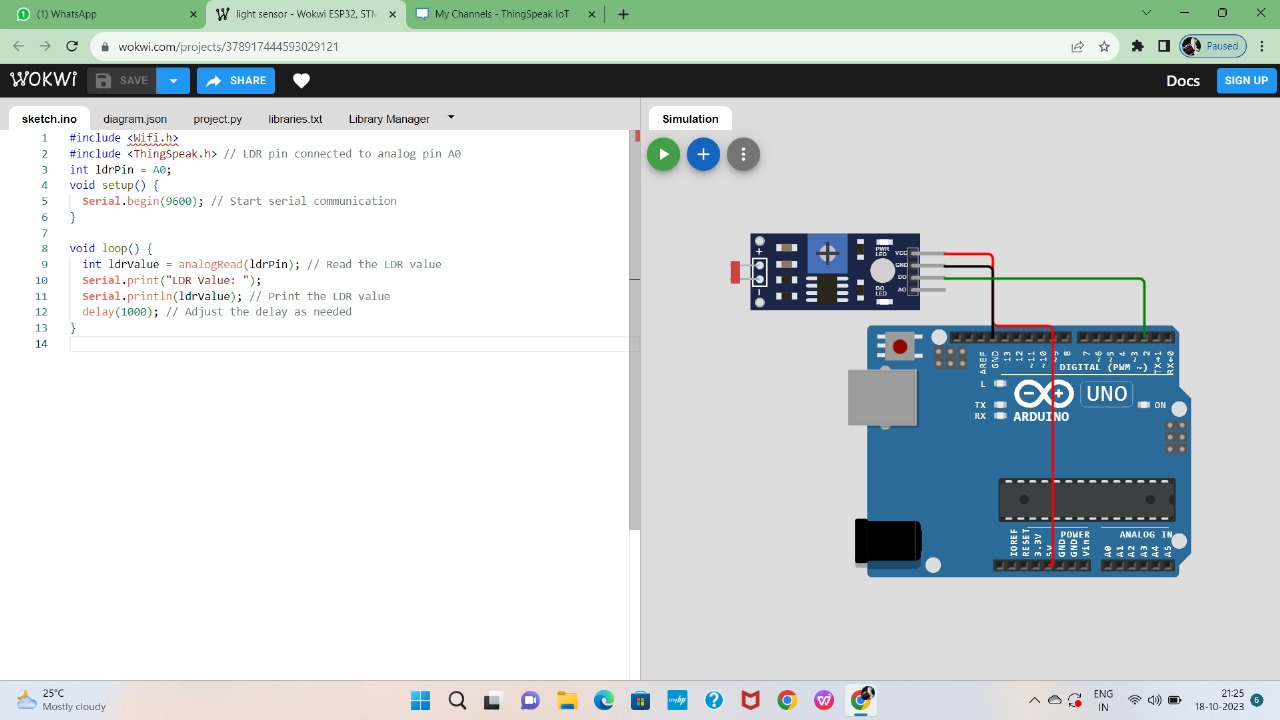
5. Data visualization: Utilize tools like graphs or charts to represent the data for easy understanding.

6. Alert system: Set up alerts or notifications for critical environmental changes or thresholds.

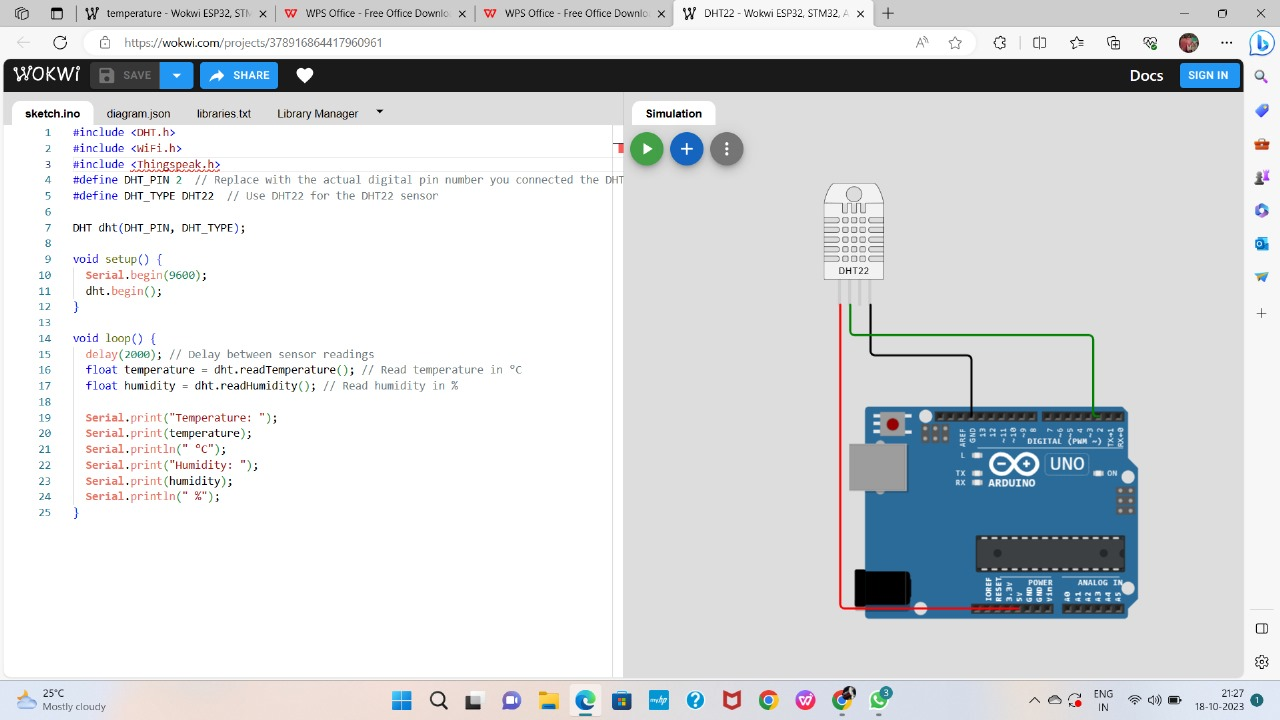
7. Remote access: Enable remote access to monitor the data from anywhere and ensure the system’s accessibility.

SCREENSHOTS

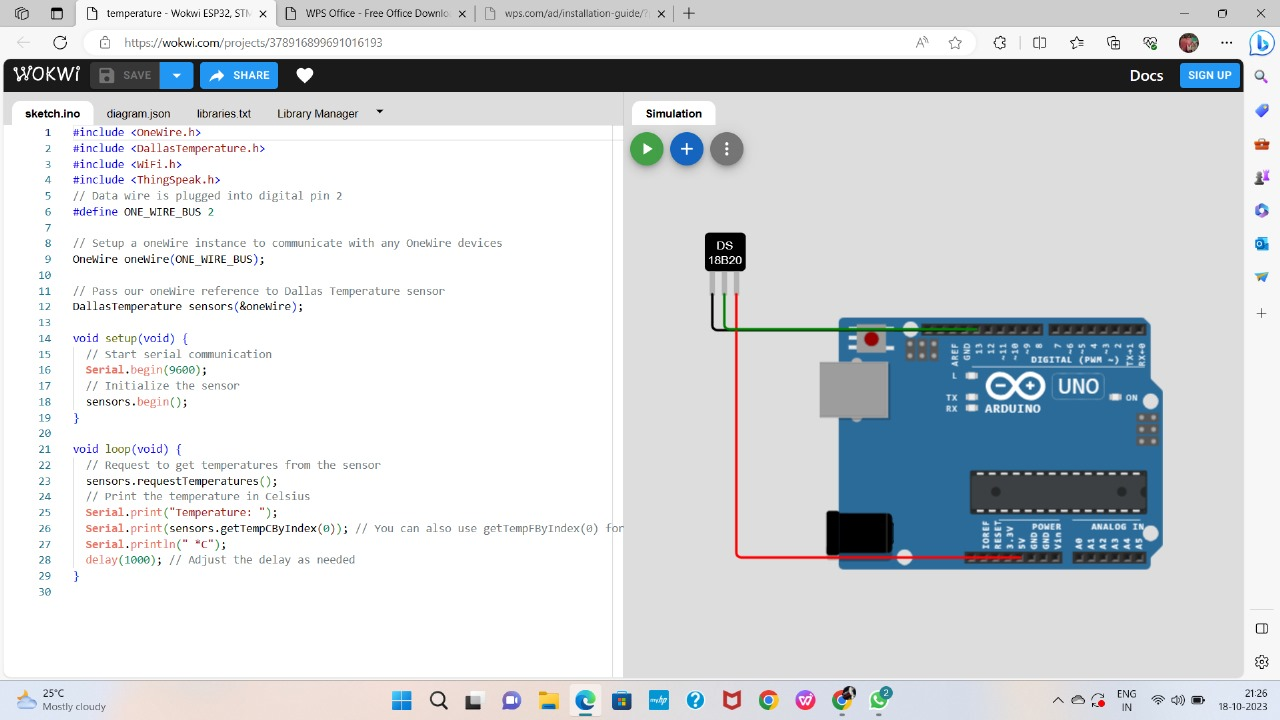
LIGHT SENSOR



HUMIDITY SENSOR



TEMPERATURE SENSOR



PYTHON SCRIPT CODE FOR SENSORS:

Python script for sharing data from temperature and humidity sensor

import requests

import time

import json

thingspeak\_url = "https://api.thingspeak.com/update"

api\_key = "UZUC7BGXH0VMGKJX"

ssid = "Wokwi-GUEST"

password = ""

DHT\_PIN = 15

TRIG\_PIN = 13

ECHO\_PIN = 12

def get\_distance():

  from machine import Pin

  import dht

  dht\_sensor = dht.DHT22(Pin(DHT\_PIN))

  while True:

    try:

        dht\_sensor.measure()

        temperature = dht\_sensor.temperature()

        humidity = dht\_sensor.humidity()

        distance = get\_distance()

        print("Temperature: {:.2f}°C, Humidity: {:.2f}%, Distance: {:.2f} cm".format(temperature, humidity, distance))

        data = {

            "api\_key": api\_key,

            "field1": temperature,

            "field2": humidity,

            "field3": distance

        }



Data sharing for light sensor

import requests

import time

import json

thingspeak\_url = "https://api.thingspeak.com/update"

api\_key = "UZUC7BGXH0VMGKJX"

Channel\_id=2310798

ssid = "Wokwi-GUEST"

password = ""

DHT\_PIN = 15

TRIG\_PIN = 13

ECHO\_PIN = 12

def get\_distance():

  from machine import Pin

  import dht

  dht\_sensor = dht.DHT22(Pin(DHT\_PIN))

  while True:

    try:

        dht\_sensor.measure()

        temperature = dht\_sensor.temperature()

        humidity = dht\_sensor.humidity()

        distance = get\_distance()

        print("Temperature: {:.2f}°C, Humidity: {:.2f}%, Distance: {:.2f} cm".format(temperature, humidity, distance))

        data = {

            "api\_key": api\_key,

            "field1": temperature,

            "field2": humidity,

            "field3": distance

        }

        response = requests.post(thingspeak\_url, data=data)

        print("Data sent to ThingSpeak. Status code:", response.status\_code)

    except Exception as e:

        print("Error:", str(e))

    time.sleep(15)

APP DEVELOPMENT:

**To create a mobile app that displays real-time temperature and humidity data received from a Raspberry Pi using Python, you would typically use a mobile app development framework like Flutter, as mentioned. However, it's essential to clarify that Flutter primarily uses Dart as its programming language, not Python. If you want to use Python for mobile app development, you might consider frameworks like Kivy, BeeWare, or Pyqtdeploy.**

1.Define App Requirements:

Determine the specific environmental parameters you want to monitor, such as temperature, humidity, air quality, pollution levels, or any other relevant data.

Select Data Sources:

Identify the source of environmental data. This could involve IoT sensors, external APIs, weather stations, or other data providers.

2.Create a Flutter Project:

Start a new Flutter project using the Flutter framework, using the Flutter CLI or your preferred IDE.

3.Design the User Interface (UI):Create the UI for your app. This may include widgets for displaying environmental data, charts for historical data, and settings for user preferences.

4.Connect to Data Sources:

Integrate with your selected data sources. This may involve making HTTP requests, connecting to IoT devices, or using third-party APIs.

5.Real-Time Data Updates:

Implement real-time data updates. You can use libraries like WebSocket or StreamBuilder to handle real-time updates and refresh the UI as new data becomes available.

6.Display Data:

Display environmental data on the app's user interface. You can use Flutter widgets like Text, Charts, or custom widgets as needed to present the data in a user-friendly manner.

7.User Settings:

Implement settings where users can configure their preferences, such as units (Celsius/Fahrenheit), alerts, and notification settings.

8.Historical Data Storage:

If you want to display historical data, consider using local storage (e.g., SQLite) or cloud-based databases to store and retrieve this data.

9.Notifications and Alerts:

Set up notifications and alerts based on user preferences and predefined thresholds for environmental parameters.

10.Mapping and Visualization:Implement maps and visualizations to display the geographical distribution of environmental data, especially if you are monitoring data from multiple locations.

11.Testing:

Thoroughly test your app to ensure that it functions correctly and is responsive to real-time data updates.

12.Optimization and Performance:

Optimize your app for performance, especially when dealing with real-time data. Minimize unnecessary updates and network requests.

13.Deployment:

Prepare your app for deployment to app stores (Google Play Store, Apple App Store). Follow platform-specific guidelines for app submission.

14.Documentation:

Document your app's features, functions, and how to use it for both users and developers.

15.User Support:

Be ready to provide user support and address issues that may arise after deployment.

RECEIVING DATA FROM RASPBERRY PI:

import requests

class EnvironmentalMonitor:

def \_init\_(self, base\_url):

self.base\_url = base\_url

def get\_temperature(self):

try:

response = requests.get(f"{self.base\_url}/temperature")

if response.status\_code == 200:

return response.json()

else:

return None

except requests.exceptions.RequestException as e:

print(f"Error: {e}")

return None

def get\_humidity(self):

try:

response = requests.get(f"{self.base\_url}/humidity")

if response.status\_code == 200:

return response.json()

else:

return None

except requests.exceptions.RequestException as e:

print(f"Error: {e}")

return None

CREATING A MODEL TO REPRESENT REAL TIME TEMPERATURE AND HUMIDITY:

class EnvironmentalData:

def \_init\_(self, temperature, humidity):

self.temperature = temperature

self.humidity = humidity

def \_str\_(self):

return f"Temperature: {self.temperature}°C, Humidity: {self.humidity}%"

# Usage:

# Create an instance of the EnvironmentalData class with real-time data

real\_time\_data = EnvironmentalData(25.5, 60.0)

# Access the temperature and humidity properties

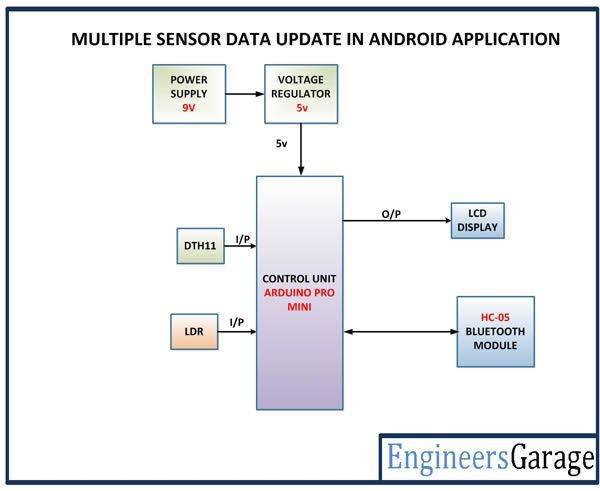
print(f"Current Temperature: {real\_time\_data.temperature}°C")

print(f"Current Humidity: {real\_time\_data.humidity}%")

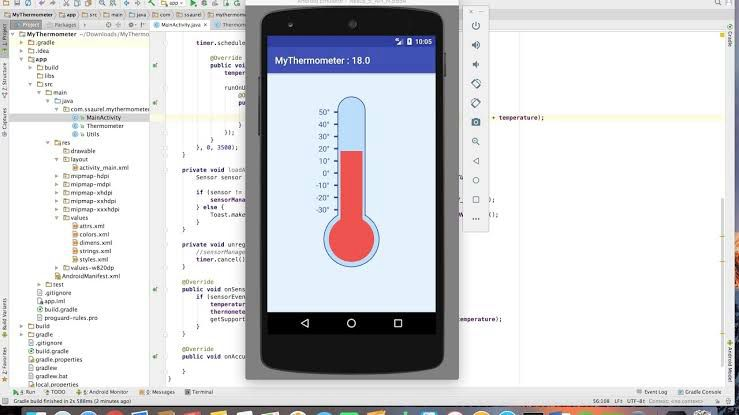
# Display the data using the \_str\_ method

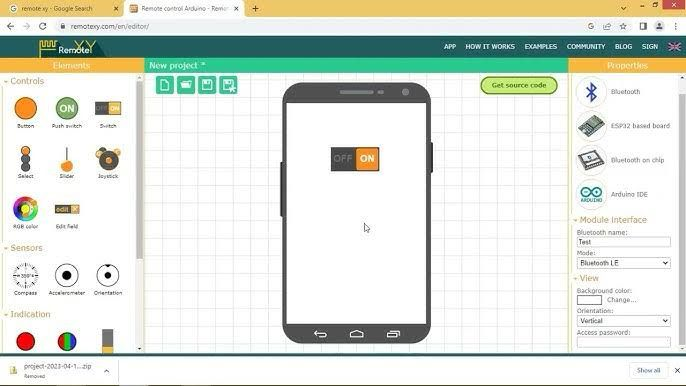
print(real\_time\_data)

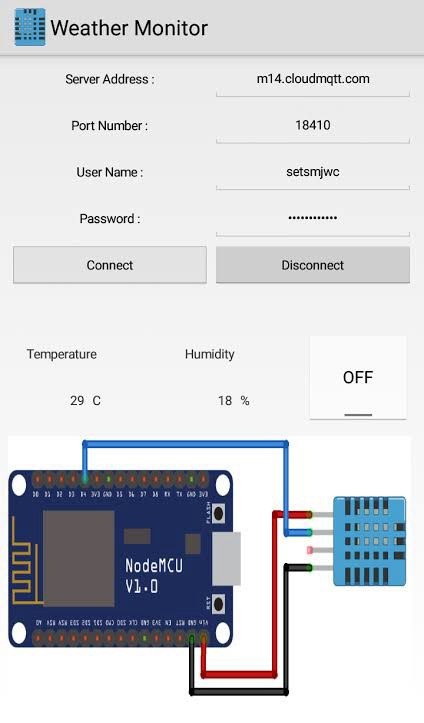
MODEL FOR ENVIRONMENTAL MONITORING:



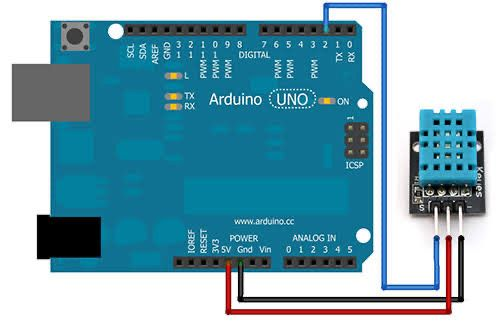
DEVELOPING THE APP:

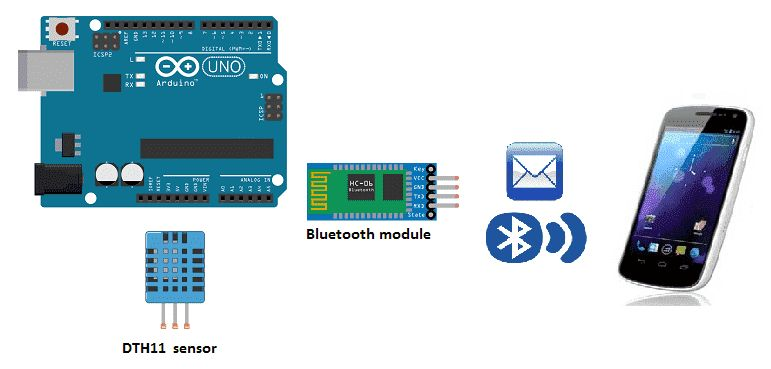




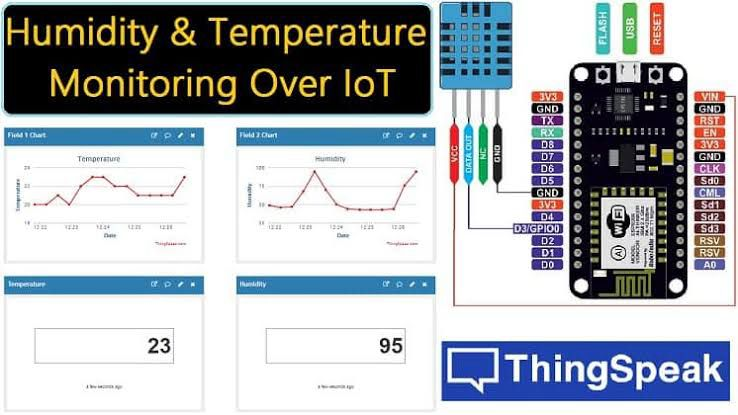


CONNECTIONS:





HUMIDITY AND TEMPERATURE MONITORING OVER THINGSPEAK:



BENEFITS OF REAL TIME ENVIRONMENT FOR PARK VISITORS AND OTHER OUTDOOR ACTIVITIES:

The Environmental Monitoring in Parks project is a comprehensive initiative aimed at assessing and preserving the ecological balance within public parks. It involves the systematic collection and analysis of data related to various environmental factors such as air and water quality, biodiversity, and climate conditions. The project employs advanced monitoring technologies and techniques to track changes in the park environment over time, enabling authorities to make informed decisions for the conservation and sustainable management of these natural spaces. Ongoing research and data-driven strategies contribute to the overall goal of ensuring the long-term health and vitality of the park ecosystems.