**Environmental monitoring**

## 

## Project Team Name: Proj\_228486\_Team\_1

**Problem Definition and Design Thinking**

**Problem Definition:**

The problem at hand revolves around environmental monitoring, specifically the need for real- time data collection and analysis to address environmental concerns. The scope includes

parameters such as air quality, soil health, and water quality. The lack of an efficient monitoring system poses challenges in timely detection and response to environmental changes, which can impact ecosystems and human health.

# Objective:

The objective of an environmental monitoring project is to systematically collect and analyze data about the natural environment to assess its health, detect pollution, support conservation, ensure compliance with regulations, provide early warnings, inform research and policy-making, raise public awareness, and enable data-driven decisions for environmental sustainability.

# Working principle:

The working principle of an environment monitoring project involves collecting data from various sensors placed in the environment to measure parameters such as temperature, humidity, air quality, and more. This data is then transmitted to a central system, often through wireless communication, where it is analyzed and displayed in real-time. Users can access this

information through a web interface or mobile app to monitor environmental conditions and make informed decisions based on the data collected.

**Design Thinking:**

# Empathize with End Users:

* + Conduct surveys, interviews, and engage with potential users to understand their concerns and expectations.
  + Identify pain points in the existing monitoring systems and gather user feedback.

# Ideation:

* + Brainstorm technological solutions, considering the use of IoT devices for real-time data collection.
  + Explore sensor technologies for measuring air quality, soil conditions, and water parameters.

# Prototyping:

* + Develop a prototype IoT device integrating various sensors.
  + Establish a communication protocol for seamless data transmission to a central server.

# Testing:

* + Conduct field tests to evaluate the accuracy and reliability of the prototype.
  + Gather feedback from test users and make necessary adjustments.

# Iterative Design:

* + Incorporate feedback from testing to refine the design and functionality of the IoT devices.
  + Ensure scalability and compatibility with diverse environmental conditions.

# Build:

* + Develop the actual hardware and software components of the monitoring system.
  + Integrate sensors, data transmission methods, and a user interface as per the refined design.

# User Interface:

* + Design a user-friendly interface for accessing real-time and historical environmental data.
  + Prioritize simplicity and clarity for users with varying technical expertise. .

# Refine:

* + Based on feedback, refine the design and functionality of the monitoring system.
  + Ensure that the system is user-friendly and provides actionable insights.

# Sustainability:

* + Evaluate the environmental impact of the monitoring devices.
  + Opt for eco-friendly materials and energy-efficient components.

# Monitor and Maintain:

* + Continuously collect and analyze environmental data.
  + Regularly update the system to address software and hardware maintenance needs.

# 11Evaluate:

* + - Assess the impact of the environment monitoring project by measuring how well it achieves its defined objectives.
    - Make adjustments and improvements based on long-term data and user feedback.

## 

## Innovation

**Description:**

Innovation in environmental monitoring involves the development and application of advanced technologies and techniques to better understand, measure, and manage the environment

## Key Features:

Innovations in environmental monitoring have become increasingly important in addressing environmental challenges. Here are some key features of innovation in this field:

**Real-time Data:** Advanced sensors and technology provide real-time data on environmental parameters, enabling rapid response to changes and emergencies.

**Remote Sensing:** Satellite technology and drones are used for remote monitoring of large areas, facilitating a broader and more comprehensive view of the environment.

**Data Integration**: Environmental monitoring systems often integrate data from various sources, including ground sensors, satellites, and citizen science initiatives.

**Big Data and Analytics:** Advanced data analytics and machine learning are used to process vast amounts of data, extract meaningful insights, and predict trends.

**IoT and Sensor Networks**: The Internet of Things (IoT) is utilized for deploying sensor networks that continuously collect and transmit data for analysis.

**Blockchain Technology:** Blockchain can be used for secure and transparent data management, ensuring the integrity and authenticity of environmental data.

**Cloud Computing**: Environmental data is often stored and processed in the cloud, making it accessible to a wide range of stakeholders and researchers.

**Mobile Apps and Citizen Engagement**: Mobile apps engage citizens in data collection and reporting, fostering a sense of environmental responsibility

**Environmental Models:** Advanced modeling and simulation tools help in predicting the impact of environmental changes and policies.

**Scalability**: Monitoring systems are designed to be scalable, allowing for expansion and adaptation to changing environmental conditions.

**AI and Machine Learning**: These technologies are used to automate data analysis, identify patterns, and make predictions, enhancing the efficiency of monitoring systems.

**Environmental Dashboards:** User-friendly dashboards provide accessible and visual representations of data for decision-makers and the public.

**Cross-sector Collaboration**: Innovations often involve collaboration between government, academia, private sector, and NGOs, fostering a holistic approach to environmental monitoring.

**Sustainability**: Many monitoring innovations focus on sustainable practices, both in terms of data collection and the use of clean energy sources for monitoring equipment.

**Regulatory Compliance:** Monitoring technologies help organizations and governments meet environmental regulations and report data accurately.

These features collectively contribute to more effective and responsive environmental monitoring, which is crucial for understanding and mitigating environmental issues.

## HARDWARE COMPONENTS USED IN IOT:

IoT (Internet of Things) systems for environmental monitoring typically consist of various hardware components, including:

**Sensors**: These are the primary components for data collection. Environmental sensors can include temperature sensors, humidity sensors, air quality sensors, water quality sensors, and more. These sensors gather data about the environment.

**Communication Modules**: IoT devices need a way to transmit data to a central system or the cloud. Common communication modules include Wi-Fi, cellular, LoRa (Long Range), and Zigbee.

**Microcontrollers**: These are the brains of the IoT device. They process data from the sensors, handle communication with the network, and control other components. Common microcontrollers used in IoT applications are Arduino, Raspberry Pi, and various microcontroller units (MCUs).

**Power Sources**: Depending on the application, power can be supplied through batteries, solar panels, or even energy harvesting methods. Long battery life is crucial for many environmental monitoring applications.

**Enclosures:** IoT devices deployed in the environment need protection from the elements. Weatherproof enclosures safeguard the internal components.

**Data Storage**: Data collected from sensors is often sent to the cloud for storage. IoT devices may include memory storage or utilize cloud-based storage services.

**Actuators (optional):** In some cases, IoT systems can take action based on the collected data. For example, they might activate fans to improve air quality or adjust irrigation systems for precision agriculture.

**Gateway/Router (for large-scale deployments):** In scenarios with numerous IoT devices, a gateway or router can aggregate data and relay it to the central server, reducing the load on individual devices.

**Security Features:** IoT devices should include security mechanisms to protect data and prevent unauthorized access. This may involve encryption, authentication, and secure boot processes.

**Display Interfaces (optional):** Some environmental monitoring systems may have displays or user interfaces for on-site data visualization or configuration.

## CODE:

Creating a program for environmental monitoring typically involves collecting and analyzing data from various sensors. Here’s a simplified example in Python using the Raspberry Pi and the DHT22 temperature and humidity sensor. This is a basic starting point, and you can expand upon it depending on your specific needs and available hardware

Import Adafruit\_DHT Import time

# Set up the sensor

Sensor = Adafruit\_DHT.DHT22

Pin = 4 # GPIO pin where the sensor is connected

While True: Try:

Humidity, temperature = Adafruit\_DHT.read\_retry(sensor, pin) If humidity is not None and temperature is not None:

Print(f’Temperature: {temperature:.2f}°C, Humidity: {humidity:.2f}%’) Else:

Print(‘Failed to retrieve data from the sensor.’) Except Exception as e:

Print(f’Error: {str€}’)

Time.sleep(60) # Read data every 60 seconds

This code reads temperature and humidity data from the sensor and prints it every 60 seconds. Depending on your specific requirements, you can store the data in a database, send it to a cloud service, or integrate other sensors and features.

In this example, we’re using the Adafruit\_DHT library to read data from the DHT22 sensor. You’ll need to install the library if you haven’t already:

Pip install Adafruit\_DHT

**DEVELOPMENT PART 1**

## Python script:

Python scripts play a pivotal role in the realm of Environmental Monitoring in the Internet of Things (IoT). In this context, Python serves as a powerful and flexible tool for collecting, processing, and analyzing data from various sensors and devices.Python scripts are instrumental in interfacing with sensors, aggregating data from them, and subsequently transmitting it for analysis and visualization.

With its ease of use, a wide range of libraries, and extensive community support, Python empowers developers to create sophisticated IoT solutions that aid in tracking, understanding, and responding to

environmental changes, contributing to the betterment of our planet. This introduction sets the stage for exploring how Python scripts can be employed to build efficient and sustainable IoT systems for

environmental monitoring.

## CODE:

Creating a complete Python script for IoT devices to send real-time environmental data to a monitoring platform is a complex task, and it typically depends on the specific IoT hardware and platform you’re using. However, I can provide you with a high-level example script that you can adapt to your specific hardware and platform.

```python

Import time

Import random Import requests

# Replace these with your IoT device’s credentials and endpoint Device\_id = “your\_device\_id”

Api\_key = “your\_api\_key”

Monitoring\_endpoint = <https://your-monitoring-platform.com/api/data>

Def read\_environmental\_data():

# Replace this with your code to read environmental data from sensors Temperature = random.uniform(20.0, 30.0)

Humidity = random.uniform(40.0, 60.0)

Return {“temperature”: temperature, “humidity”: humidity}

Def send\_data\_to\_monitoring\_platform(data): Headers = {

“Authorization”: f”Bearer {api\_key}”

}

Try:

Response = requests.post(monitoring\_endpoint, json=data, headers=headers) If response.status\_code == 200:

Print(“Data sent successfully”) Else:

Print(f”Failed to send data. Status Code: {response.status\_code}”) Except requests.exceptions.RequestException as e:

Print(f”Failed to send data. Error: {e}”)

If name == “ main ”:

While True:

Environmental\_data = read\_environmental\_data()

Send\_data\_to\_monitoring\_platform({“device\_id”: device\_id, “data”: environmental\_data})

Time.sleep(60) # Adjust the interval as needed

```

Please note that you’ll need to replace the placeholder values for `device\_id`, `api\_key`, and

`monitoring\_endpoint` with you’re actual IoT device’s credentials and the URL of your monitoring platform’s API.

This script generates random environmental data (temperature and humidity) for demonstration purposes. You should modify the `read\_environmental\_data` function to read data from your actual sensors.

.

## DEVELOPMENT PART 2

**Components:**

The IoT Environmental Monitoring System is a project designed to monitor temperature and humidity using a DHT22 sensor .This report outlines the components used, the setup, and the functionality of the system.

## Hardware Components:

* + NodeMCU ESP8266: The NodeMCU ESP8266 is used as the microcontroller and Wi-Fi module for connecting to the internet and interfacing with the DHT22 sensor.
  + DHT22 Sensor: The DHT22 sensor is utilized to measure temperature and humidity data. It is connected to the NodeMCU ESP8266 through the DHT\_PIN (GPIO 15).
  + LED Indicator: An LED (Light Emitting Diode) is connected to the NodeMCU ESP8266 via the LED\_PIN (GPIO 13). It acts as an indicator that turns on when the temperature or humidity falls outside a predefined range.

## Software Components:

* + Arduino IDE: The Arduino Integrated Development Environment is used for writing and uploading code to the NodeMCU ESP8266.
  + DHTesp Library: This library is used to interface with the DHT22 sensor and retrieve temperature and humidity data.

## System Design:

**Sensor Configuration**

The DHT22 sensor is connected to the NodeMCU ESP8266 on pin 15 (DHT\_PIN). It is configured to monitor temperature and humidity data.

## Wi-Fi Connectivity

The system connects to a Wi-Fi network with the provided credentials (WIFI\_NAME and WIFI\_PASSWORD). It waits until a connection is established before proceeding.

## Data Collection

The DHT22 sensor periodically collects temperature and humidity data, which is then stored in the data object

## LED Indicator

The LED is turned on when the temperature is above 35°C or below 12°C, or when the humidity is above 70% or below 40%. It serves as a visual indicator for extreme weather conditions.

## ThingSpeak Integration

The collected data is sent to ThingSpeak using the ThingSpeak library. The temperature is sent as Field 1, and humidity as Field 2. The data is uploaded to ThingSpeak's server with the provided API key and channel number.

## Operation:

* + Upon power-up or reset, the system initializes and connects to the Wi-Fi network.
  + Once connected, it begins monitoring temperature and humidity data.
  + If the temperature or humidity falls outside the specified ranges, the LED is turned on.
  + The collected data is sent to ThingSpeak every 10 seconds.
  + The system prints the temperature, humidity, and the status of the data upload to the serial monitor.

## Simulation Code:

//LCD I2C library:

#include <LiquidCrystal\_I2C.h>

//DHT22 sensor library:

#include <DHT.h>;

//LCD I2C address 0x27, 16 column and 2 rows! LiquidCrystal\_I2C lcd(0x27, 16, 2);

//Constants:

#define DHTPIN 2 //what pin we're connected to #define DHTTYPE DHT22 //DHT 22 (AM2302)

DHT dht(DHTPIN, DHTTYPE); //Initialize DHT sensor for normal 16mhz Arduino

//Variables:

float H; //Humidity value float T; //Temperature value int buzzer = 12;

//Initialize LCD, DHT22 sensor and buzzer:

void setup(){

lcd.init(); lcdback.light(); dht.begin(); pinMode(buzzer, OUTPUT);

//Print some text in Serial Monitor

**Serial**.begin(9600); **Serial**.println("DHT22 sensor with Arduino Uno R3!"); pinMode(9, OUTPUT); pinMode(10, OUTPUT); pinMode(11, OUTPUT);

}

void loop(){ delay(2000);

//Read data and store it to variables hum and temp

H = dht.readHumidity(); T = dht.readTemperature();

//Print temp and humidity values to serial monitor

**Serial**.print("Humidity: "); **Serial**.print(H); **Serial**.println(" %; "); **Serial**.print("Temperature: "); **Serial**.print(T); **Serial**.println(" Celsius.\n");

/\*If humidity is higher than 70% & temperature is higher than 30 degrees Celsius

then it will show on LCD „Too warm! Cool down!”\*/

if(H >= 70.00 && T >= 30.00){

digitalWrite(9, HIGH); digitalWrite(10, LOW); digitalWrite(11, LOW);

lcd.println(" Too warm! "); lcd.setCursor(0, 1); lcd.println(" Cool down! "); lcd.setCursor(0, 0);

digitalWrite(buzzer, 1); tone(buzzer, 900, 100); delay(400);

digitalWrite(buzzer, 0); tone(buzzer, 900, 100); delay(400);

digitalWrite(buzzer, 1); tone(buzzer, 900, 100); delay(400);

digitalWrite(buzzer, 0); tone(buzzer, 900, 100); delay(400);

}else{

/\*If humidity is lower than 70% & temperature is lower than 30 degrees Celsius

then it will show on LCD „Temp. & hum. are in normal limits”\*/ digitalWrite(9, LOW); digitalWrite(10, LOW); digitalWrite(11, HIGH); lcd.println("Temp. & hum. are"); lcd.setCursor(0, 1); lcd.println("in normal limits"); lcd.setCursor(0, 0); digitalWrite(buzzer, 0);

}

/\*If either humidity is lower than 70%, but temperature is higher than 30 degrees Celsius,

then it will show on LCD „Be ware! Temp. too high” or

humidity is higher than 70%, but

temperature is lower than 30 degrees Celsius, then it will show on LCD „Be ware! Hum. too high”\*/ if(H < 70.00 && T >= 30.00){

digitalWrite(9, LOW); digitalWrite(10, HIGH); digitalWrite(11, LOW); lcd.println("Be ware! "); lcd.setCursor(0, 1);

lcd.println("Temp. too high! "); lcd.setCursor(0, 0); digitalWrite(buzzer, 1); tone(buzzer, 400, 400); delay(400);

digitalWrite(buzzer, 0); tone(buzzer, 400, 400); delay(400);

}

if(H >= 70.00 && T < 30.00){

digitalWrite(9, LOW); digitalWrite(10, HIGH); digitalWrite(11, LOW); lcd.println("Be ware! "); lcd.setCursor(0, 1); lcd.println("Hum. too high! "); lcd.setCursor(0, 0); digitalWrite(buzzer, 1); tone(buzzer, 400, 400); delay(400);

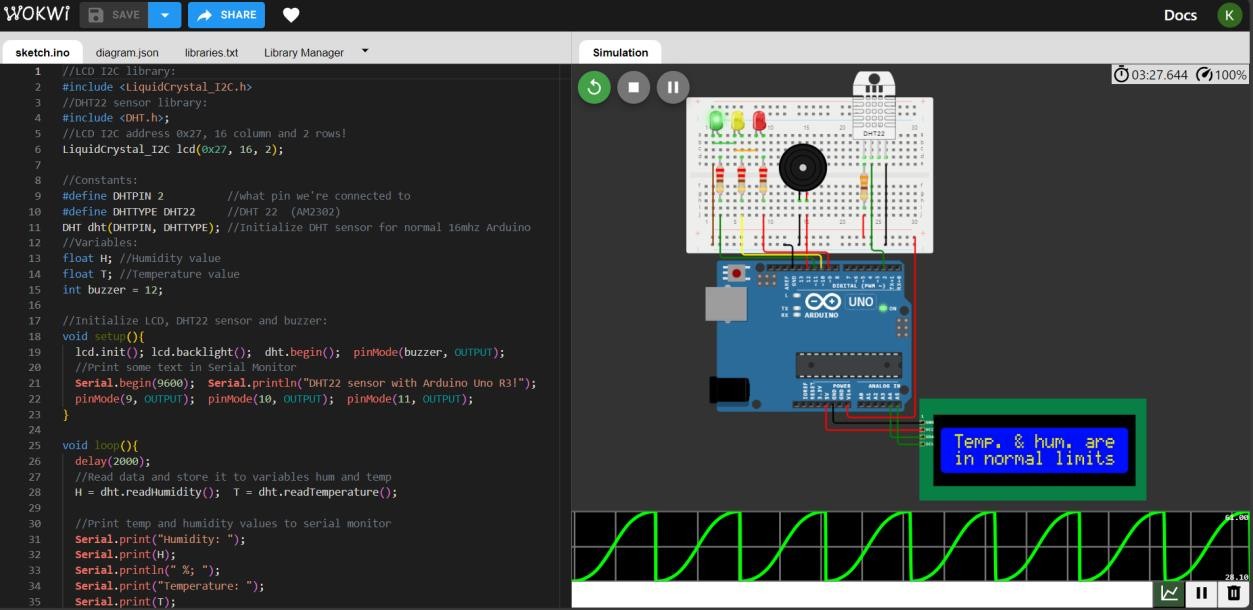
digitalWrite(buzzer, 0); tone(buzzer, 400, 400); delay(400);

}

}

## Simulation link:

[**https://wokwi.com/projects/379572007533323265**](https://wokwi.com/projects/379572007533323265) **Simulation Output:**



## Web Code:

<!DOCTYPE html>

<html>

<head>

<title> </title>

<style>

body

font-family Arial sans-serif background-color #f2f2f2 text-align center

h1

color #333

.container

display flex

justify-content space-around margin-top 20px

.data-box

background-color #fff border 1px solid #ddd border-radius 5px padding 10px

box-shadow 0px 2px 5px rgba 0 0 0 0.2

h2

color #555

p

font-size 20px color #333

margin 5px 0

</style>

</head>

<body>

<h1> </h1>

<div class "container">

<div class "data-box" id "temperature">

<h2> </h2>

<p id "temp-value"> </p>

</div>

<div class "data-box" id "humidity">

<h2> </h2>

<p id "humidity-value"> </p>

</div>

</div>

<script>

// Function to generate random temperature and humidity values function generateRandomData

var randomTemperature =Math random \* 30 + 15 // Random temperature between 15°C and 45°C

var randomHumidity = Math random \* 40 + 30 // Random humidity between 30% and 70%

document getElementById "temp-value" textContent = randomTemperature toFixed 2 + " °C"

document getElementById "humidity-value" textContent = randomHumidity toFixed 2 + " %"

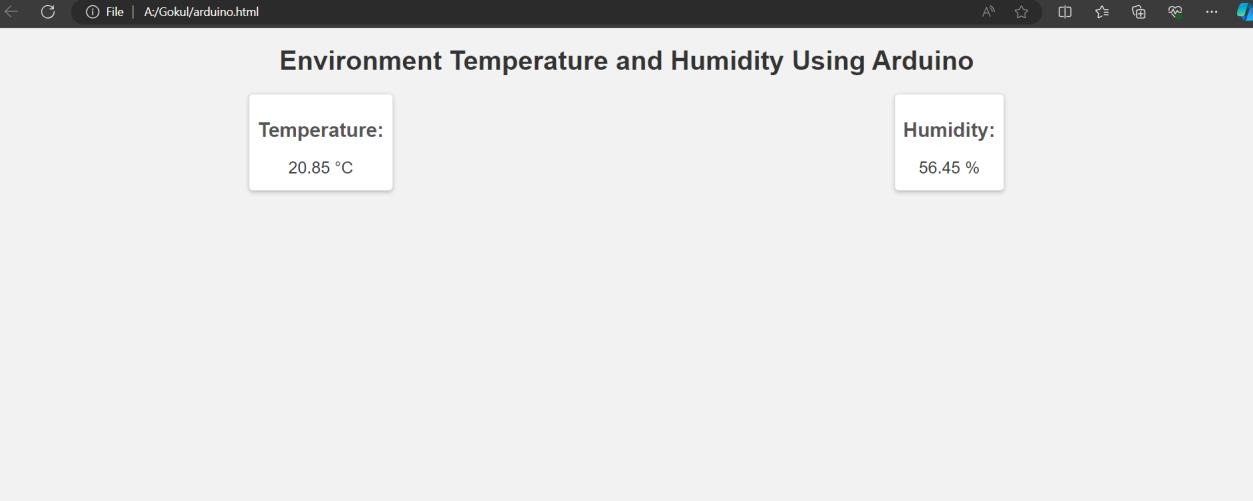
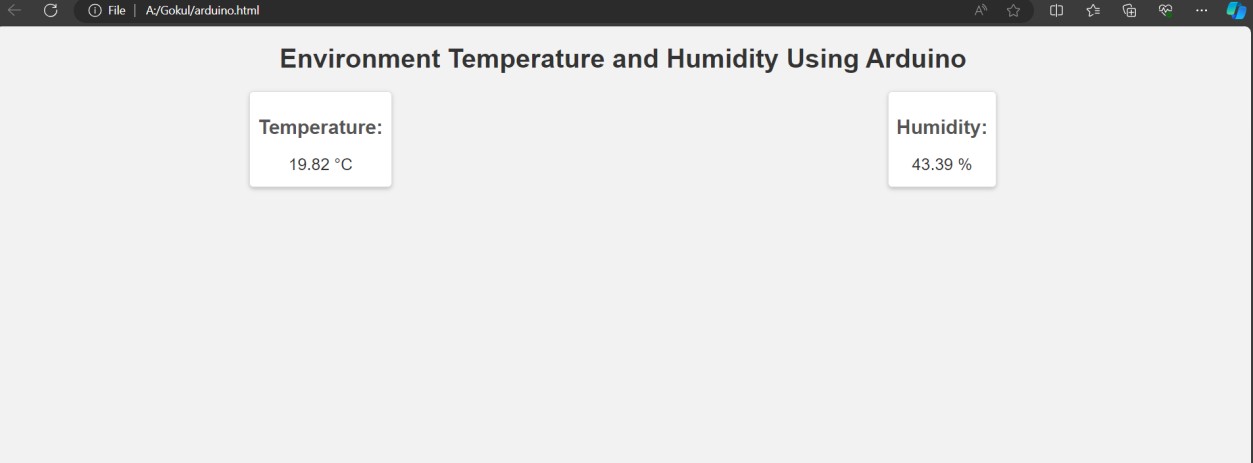
// Generate random data every 2 seconds setInterval generateRandomData 2000

</script>

</body>

</html>

## Output:



**Thinkspeak Visualization:**

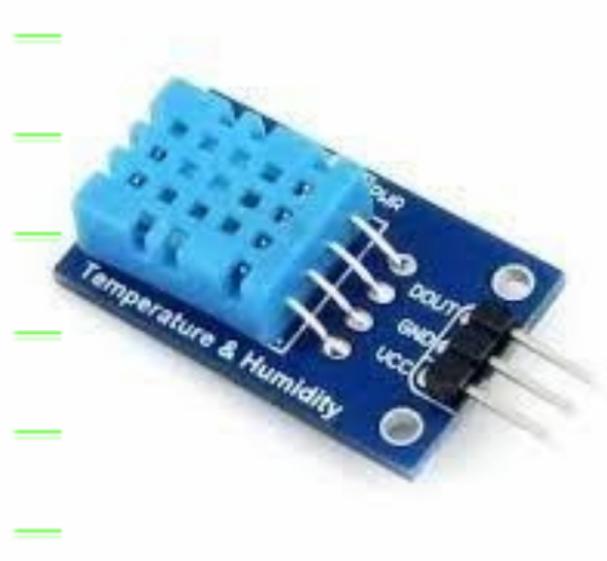
**Link:** [**https://thingspeak.com/channels/2319329**](https://thingspeak.com/channels/2319329)

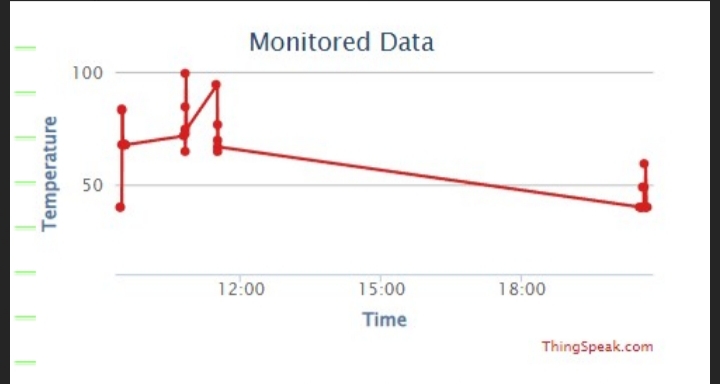
**IOT DEVICES:**

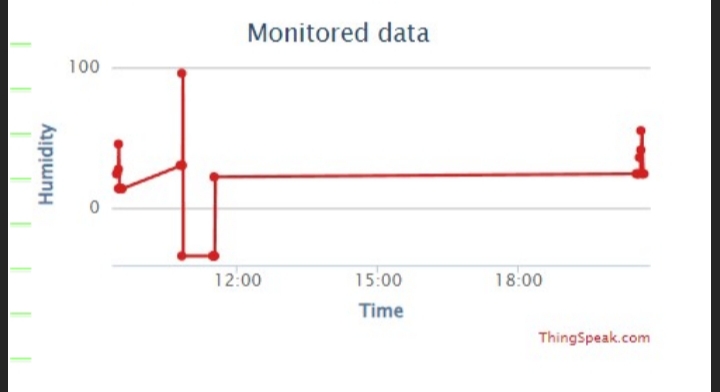
**ARDINO UNO**

****

**DHT SENSOR :**

****

****



**CONCLUSION:**

In conclusion implementing environmental monitoring through IoT in parks provides numerous benefits for park visitors and the natural ecosystem. It enhances visitor experiences by offering real-time information on air quality, weather, and other environmental factors, ensuring safety and enjoymen xxt. Moreover, it aids in the conservation and protection of the park’s natural resources, contributing to the long-term sustainability of these beautiful natural spaces. Overall, IoT-based environmental monitoring is a win-win for both park enthusiasts and the environment they cherish.