IOT PHASE 4

Problem statement

The project involves setting up IoT devices in parks for people to make come to parks conveniently and encourage a healthy lifestyle. The primary objective is to encourage outdoor activities for long time by monitoring climatic conditions like temperature and humidity and providing enough facilities in the parks like automatic street lights.

Development

We continue to build the project using web development technologies such as html, css, JavaScript to create a platform that displays real time environmental monitoring.

Wokwi simulator is used to simulate thee arduino code and the data retrieved is stored in thingspeak cloud which is an Iot cloud device platform.

This data is sent to the user when the user presses the button in our web based application.

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Temperature and Humidity Monitor</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <div class="data-section">
      <h2>Temperature</h2>
      <div id="temp-data"></div>
      <button id="temp-button">Get Temperature</button><br>
    </div>
    <div class="data-section">
      <h2>Humidity</h2>
      <div id="humid-data"></div>
      <button id="humid-button">Get Humidity</button><br>
    </div>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

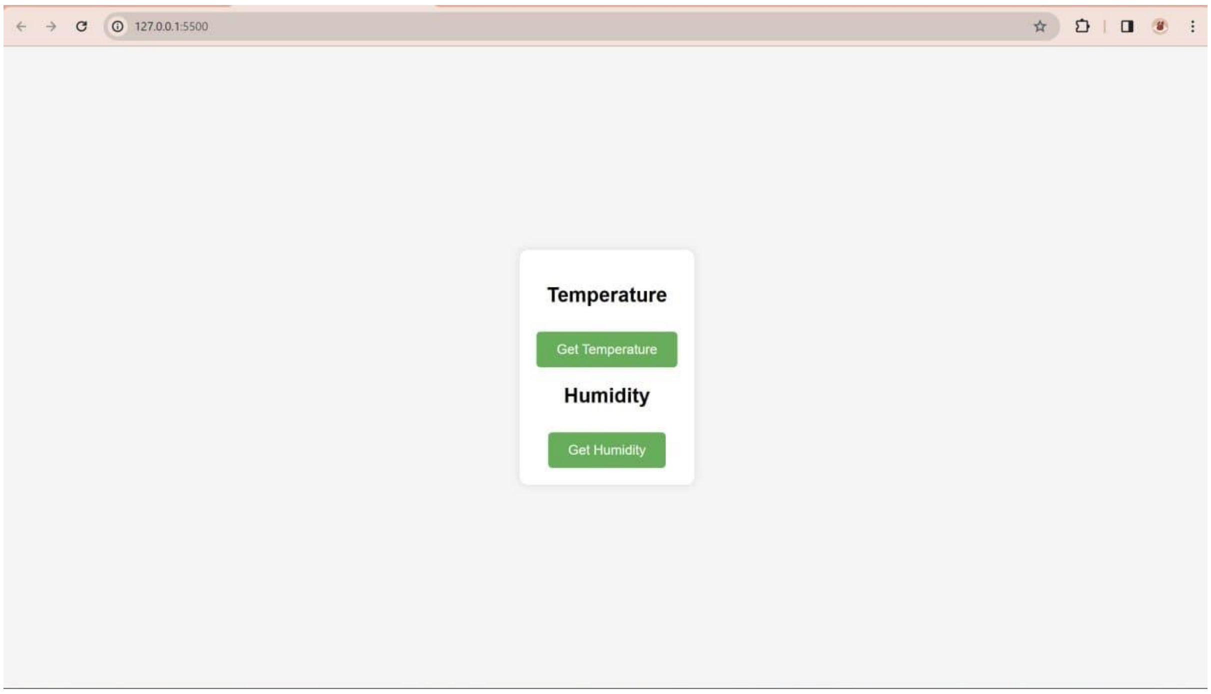
Styles.css

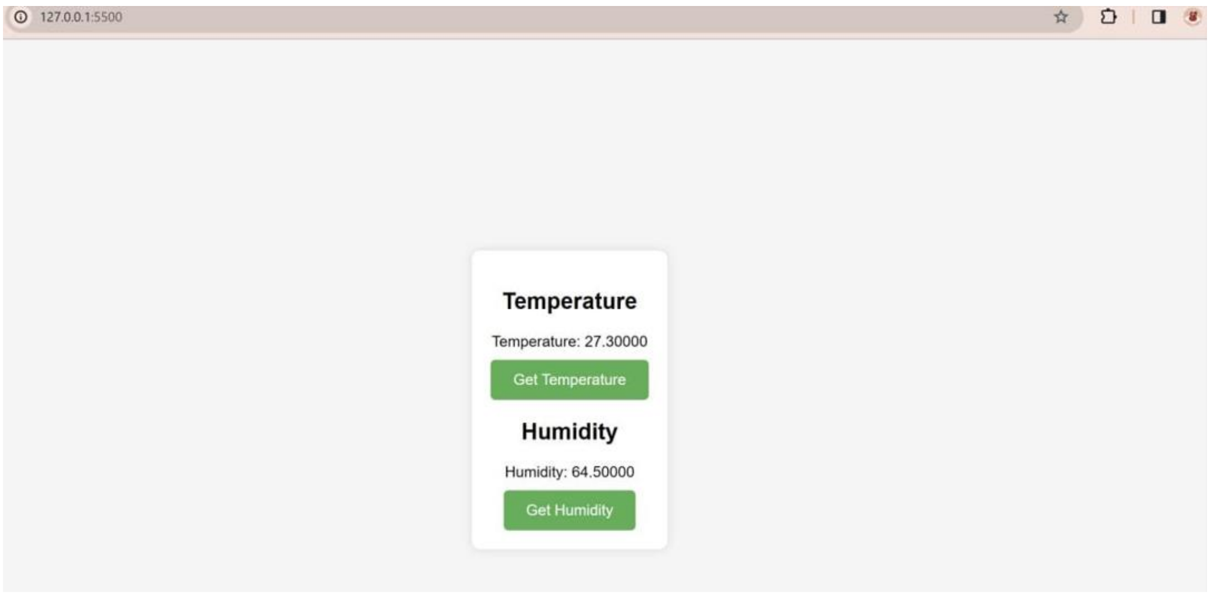
```
body {
  font-family: 'Arial', sans-serif;
  background-color: #f5f5f5;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  height: 100vh;
  margin: 0;
}
.container {
  text-align: center;
  background-color: #ffffff;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0px 0px 10px 0px rgba(0,0,0,0.1);
}
#data-container {
  font-size: 28px;
  margin-bottom: 20px;
  color: #333333;
}
#temp-button,
#humid-button {
  background-color: #4CAF50;
  color: white;
  border: none;
  padding: 12px 24px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  margin-top: 10px;
  cursor: pointer;
  border-radius: 5px;
  transition: background-color 0.3s ease;
}
#temp-button:hover,
#humid-button:hover {
  background-color: #45a049;
}
```

DESCRIPTION

The HTML file defines the structure with two sections for temperature and humidity, each having a heading, data display area, and a button. The CSS file styles the elements, providing a clean and centered layout with colored buttons and hover effects. For dynamic functionality, you can link this interface with JavaScript (**script.js**) to handle button clicks and update data dynamically.

OUTPUT





Temperature: 27.30000

Humidity: 64.50000

Script.js

```
document.getElementById('temp-button').addEventListener('click', getTemperature);

document.getElementById('humid-button').addEventListener('click', getHumidity);


function getTemperature() {
  fetchThingSpeakData('field1', 'temp-data', 'Temperature');
}

function getHumidity() {
  fetchThingSpeakData('field2', 'humid-data', 'Humidity');
}

function fetchThingSpeakData(field, elementId, dataType) {
  const channelID = '2326456';
  const apiKey = 'VB88Y4IRP15ZQ1MO';
  const baseURL = `https://api.thingspeak.com/channels/${channelID}/feeds.json`;

  fetch("${baseURL}?api_key=${apiKey}&results=1&${field}")
    .then(response => response.json())
    .then(data => {
      const value = data.feeds[0][field];
      document.getElementById(elementId).textContent = "${dataType}: ${value}";
    })
    .catch(error => {
      console.error("Error fetching ${field} data:", error);
      document.getElementById(elementId).textContent ="Error fetching ${dataType} data";
    });
}
```

DESCRIPTION

1. The first two event listeners are attached to the HTML elements with IDs **get-temperature** and **get-humidity**. When these elements are clicked, they send messages **'getTemperature'** and **'getHumidity'** respectively to the parent window.
2. The **window.addEventListener('message', event => { ... });** block listens for messages sent from the parent window. When a message is received, it checks the origin of the message (ensuring it's from the specified URL **'https://wokwi.com/projects/new/esp32'**) and processes the message data.

- If the message type is **'temperature'**, it updates the element with the ID **temperature-data** to display the received temperature value.
- If the message type is **'humidity'**, it updates the element with the ID **humidity-data** to display the received humidity value.

Sketch.ino

```
#include<WiFi.h>

#include<ThingSpeak.h>

#include <LiquidCrystal_I2C.h>

#include <DHT.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

char ssid[]="Wokwi-GUEST";

char pass[]="";

WiFiClient client;

unsigned long  myChannelNumber=2326366;

const char* myWriteAPIKey="W99IL4JFKFZKTP7L";

int statusCode;

#define DHTPIN 2

#define DHTTYPE DHT22

#define light 8

DHT dht(DHTPIN, DHTTYPE);

float H; //Humidity value

float T; //Temperature value

void setup() {

  lcd.init();

  lcd.backlight();

  dht.begin();

  pinMode(light,OUTPUT);

  Serial.begin(9600);

  Serial.println("DHT22 sensor with Arduino Uno R3!");

    ThingSpeak.begin(client);

}
```

```
void loop() {
  delay(2000);
  H = dht.readHumidity();
  T = dht.readTemperature();
  Serial.print("Humidity: ");
  Serial.print(H);
  Serial.println(" %; ");
  Serial.print("Temperature: ");
  Serial.print(T);
  Serial.println(" Celsius.\n");

  if (H >= 70.00 && T >= 30.00) {
    digitalWrite(light,HIGH);
    lcd.println("  Too warm!    ");
    lcd.setCursor(0, 1);
    lcd.println("  Cool down!   ");
    lcd.setCursor(0, 0);
    delay(2000);
    digitalWrite(light,LOW);
  }
else {
    lcd.println("Temp & humi is");
    lcd.setCursor(0, 1);
    lcd.println("in normal limits");
    lcd.setCursor(0, 0);
  }
  if (H < 70.00  && T >= 30.00) {
    lcd.println("Be ware!       ");
    lcd.setCursor(0, 1);
    lcd.println("Temp. too high! ");
    lcd.setCursor(0, 0);
```

```
 }
 if (H >= 70.00 && T < 30.00) {
  lcd.println("Be ware!" );
 }
 if(WiFi.status()!=WL_CONNECTED)
 {
  Serial.println("Attempting to connect");
  while(WiFi.status()!=WL_CONNECTED)
  {
    WiFi.begin(ssid,pass);
    Serial.print(".");
    delay(5000);
  }
 }
  Serial.println("\nConnected");
  ThingSpeak.setField(1,H);
  ThingSpeak.setField(2,T);
  statusCode=ThingSpeak.writeFields(myChannelNumber,myWriteAPIKey);
  if(statusCode==200)
   {
     Serial.println("Channel update successful");
   }
  else
   {
     Serial.println("Problem waiting data:HTTp error code:"+String(statusCode));
    }
   delay((15000));
}
```

DESCRIPTION

Initialization:

- The DHT22 sensor and LCD are initialized.
- Wi-Fi credentials are set up.

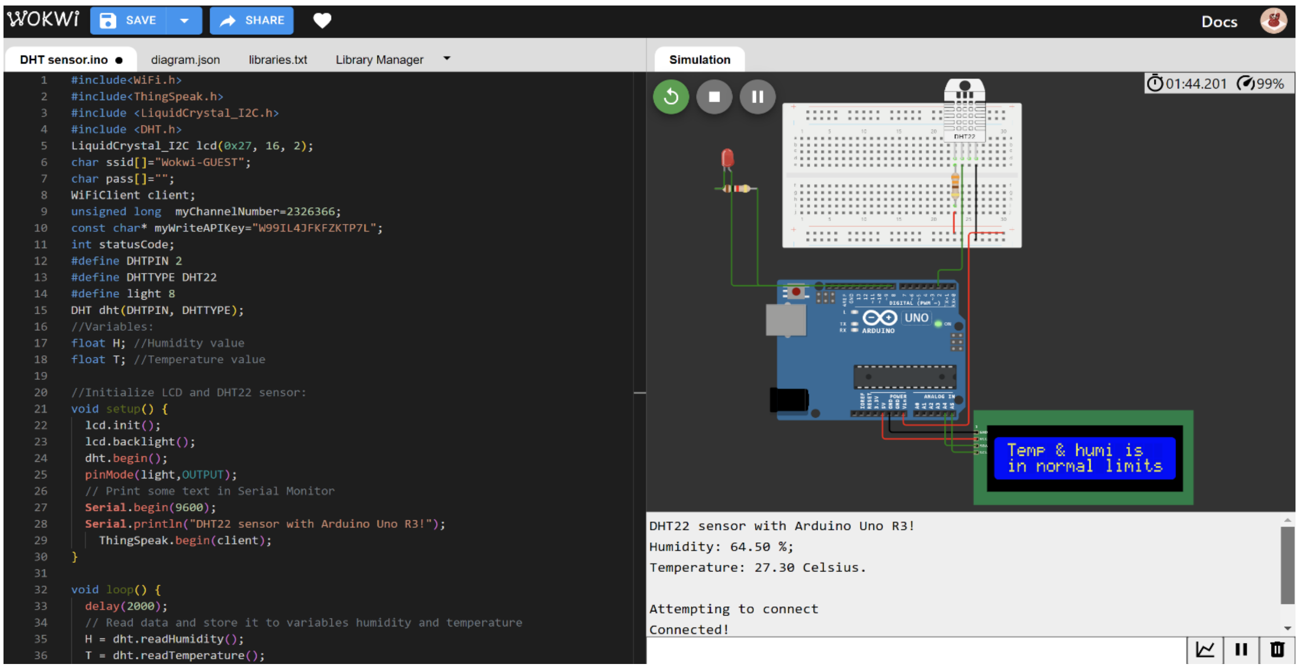
Setup:

- Serial communication is started for debugging purposes.
- ThingSpeak communication is initialized.

Loop:

- In the **loop()** function, the program checks the temperature and humidity values from the DHT22 sensor.
- If the temperature and humidity are within certain limits, it displays a message on the LCD indicating normal conditions.

OUTPUT



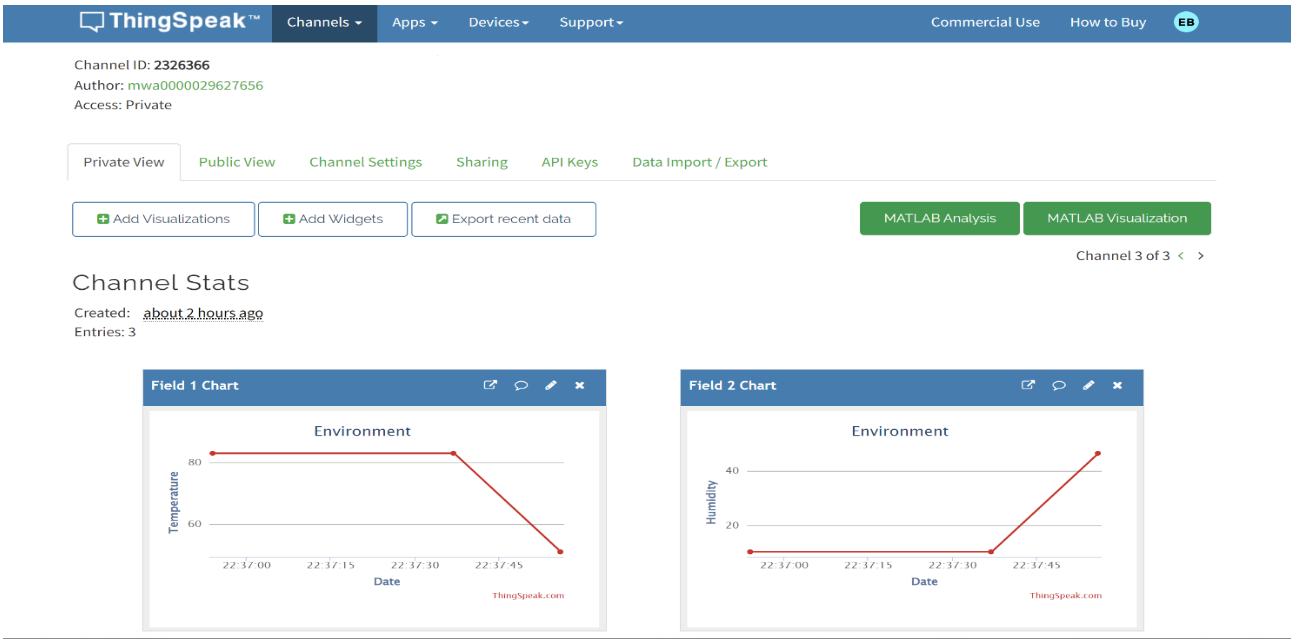
```
#include<WiFi.h>
#include<ThingSpeak.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
char ssid[]="Wokwi-GUEST";
char pass[]="";
WiFiClient client;
unsigned long  myChannelNumber=2326366;
const char* myWriteAPIKey="W99IL4JFKFZKTP7L";
int statusCode;
#define DHTPIN 2
#define DHTTYPE DHT22
#define light 8
DHT dht(DHTPIN, DHTTYPE);
//Variables:
float H; //Humidity value
float T; //Temperature value

//Initialize LCD and DHT22 sensor:
void setup() {
  lcd.init();
  lcd.backlight();
  dht.begin();
  pinMode(light,OUTPUT);
  // Print some text in Serial Monitor
  Serial.begin(9600);
  Serial.println("DHT22 sensor with Arduino Uno R3!");
    ThingSpeak.begin(client);
}

void loop() {
  delay(2000);
  // Read data and store it to variables humidity and temperature
  H = dht.readHumidity();
  T = dht.readTemperature();
```

```
DHT22 sensor with Arduino Uno R3!
Humidity: 64.50 %;
Temperature: 27.30 Celsius.

Attempting to connect
Connected!
```



Channel ID: 2326366
Author: mwa0000029627656
Access: Private

Channel Stats

Created: about 2 hours ago
Entries: 3