GOVERNMENT COLLEGE OF ENGINEERING, ERODE

INTERNET OF THINGS

ENVIRONMENTAL MONITORING

PHASE-3

Team Members:

Brintha Shree S S - [22CSE56L]

Kalpana Chawla M– [21CSE17]

Kavipriya P – [21CSE19]

Yogalakshmi S – [21CSE54]

To Deploy IoT devices (e.g., temperature and humidity sensors) in various locations within public parks to measure environmental conditions.

TO FIND TEMPERATURE AND HUMIDITY USING AN ESP32

Hardware Setup:

Choose suitable IoT hardware, like Raspberry Pi, Arduino, or specific IoT boards. Select the right hardware: Choose a humidity and temperature sensor that suits your needs. Examples of such sensors include DHT11, DHT22, and BME280.

Software Dependencies:

Ensure your IoT device is running an operating system (e.g., Raspbian for Raspberry Pi). Install necessary Python libraries for IoT, such as RPi.GPIO or Adafruit CircuitPython (for Raspberry Pi) or libraries specific to your hardware.

Set up the microcontroller:

Select a microcontroller board that is compatible with the sensor and can be programmed to collect data. Examples of such boards include Arduino Uno, Arduino Nano, and ESP8266.

Connect the sensor to the microcontroller:

Use jumper wires to connect the sensor's VCC, GND, data, and clock pins to the appropriate pins on the microcontroller board.

Program the microcontroller:

Write a program in your preferred programming language (e.g., Arduino IDE, Python, or JavaScript) that collects data from the sensor, processes it, and sends it to the central monitoring system.

Connect the microcontroller to the internet:

Set up the microcontroller to connect to the internet via a Wi-Fi module or a cellular modem. This will allow the sensor data to be transmitted to a remote server.

Create a central monitoring system:

Design a web-based platform or mobile application that can receive the sensor data from the microcontroller and display it in a user-friendly format. This system can be hosted on a remote server.

Establish secure communication channels:

Implement secure communication protocols (e.g., SSL/TLS, HTTPS, or WebSockets) to ensure that the data transmitted between the microcontroller and the central monitoring system is encrypted and secure.

Set up data storage and retrieval:

Configure the central monitoring system to store the received sensor data in a database for later analysis and retrieval.

Design user interfaces for easy data access:

Create intuitive user interfaces that allow users to access the sensor data, compare trends over time, and receive notifications if any environmental conditions are out of the ordinary.

Conduct testing and quality assurance:

Thoroughly test the system in the field and fix any bugs or issues that arise. Ensure that the hardware and software components are robust and can withstand harsh environmental conditions.

INTERFACE DHT11 WITH ARDUINO UNO

- Plug the DHT11 sensor into the breadboard. The DHT11 sensor consists of four pins: VCC, GND, DATA, and NC.
- Connect the VCC pin of the DHT11 sensor to the 5V pin on the Arduino Uno board using a jumper wire.
- Connect the GND pin of the DHT11 sensor to the GND pin on the Arduino Uno board using a jumper wire.
- Connect the DATA pin of the DHT11 sensor to the digital pin 2 on the Arduino Uno board using a jumper wire.
- Plug the LED into the breadboard. Connect the longer leg (positive) of the LED to a 220-ohm resistor, which is then connected to the 5V pin on the Arduino Uno board.
- Connect the shorter leg (negative) of the LED to a digital pin on the Arduino Uno board (e.g., digital pin 3).

In the Arduino IDE, create a new sketch and upload the following code:

```
#include <DHT.h>

#define DHTPIN 2

#define DHTTYPE DHT11

#define LED_PIN 3

DHT dht(DHTPIN, DHTTYPE);

void setup() {

 Serial.begin(9600);

 dht.begin();

 pinMode(LED_PIN, OUTPUT);

}

void loop() {
```

```
 float humidity = dht.readHumidity();
 float temperature = dht.readTemperature();
 if (isnan(humidity) || isnan(temperature)) {
   Serial.println("Failed to read from DHT sensor!");
   return;
 }
 Serial.print("Humidity: ");
```

Connect the DHT11 sensor to the Arduino Uno board using jumper wires. Here is how to connect the pins:

- DHT11 VCC to Arduino 5V
- DHT11 GND to Arduino GND
- DHT11 DATA to Arduino Digital Pin 2

Library Installation

Install the DHT sensor library by Michael C. Adams from the Arduino Library Manager.

Upload the following code to the Arduino Uno board:

```
#include <DHT.h>
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
void setup() {
 Serial.begin(9600);
 Serial.println("DHT11 test!");
```

```
 dht.begin();
}
void loop() {
 delay(2000);
 float humidity = dht.readHumidity();
 float temperature = dht.readTemperature();
 if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
 }
 Serial.print("Humidity: ");
 Serial.print(humidity);
 Serial.print(" %\t");
 Serial.print("Temperature: ");
 Serial.print(temperature);
 Serial.println(" *C");
}
```

Code for posting on HTTP to public platform

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <DHT.h>
// Dummy WiFi credentials (not used in the simulation)
```

```
const char* ssid = "your_network_name";
const char* password = "your_network_password";
// Local web server address and port (update with your server details)
const char* serverAddress = "127.0.0.1";  // Replace with your server's IP address
const int serverPort = 80;  // Replace with your server's port
// DHT sensor configuration
#define DHTPIN 4        // Define the GPIO pin to which the DHT22 is connected
#define DHTTYPE DHT22    // Define the sensor type (DHT11 or DHT22)
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(115200);
  // Dummy WiFi connection (not used in the simulation)
  Serial.println("Connecting to WiFi (simulated)...");
  delay(1000);
  Serial.println("Connected (simulated) to WiFi");
  // Initialize the DHT sensor
  dht.begin();
}
void loop() {
  // Read temperature and humidity
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  if (isnan(temperature) || isnan(humidity)) {
```

```
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // Simulate an HTTP POST request to the local web server
  HTTPClient http;
  // Construct the URL
  String url = "http://" + String(serverAddress) + ":" + String(serverPort) + "/endpoint";
// Update with your server's endpoint
  http.begin(url);
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");
  String postData = "temperature=" + String(temperature) + "&humidity=" +
String(humidity);
  int httpResponseCode = http.POST(postData);
  if (httpResponseCode > 0) {
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
    // Print the response from the server (simulated)
    String response = http.getString();
    Serial.print("Server Response: ");
    Serial.println(response);
  } else {
    Serial.print("Error in HTTP request. HTTP Response code: ");
    Serial.println(httpResponseCode);
```

```
  }
  http.end();
  delay(60000); // Simulate sending data every 1 minute (adjust as needed)
}
```

Code for Beeceptor End Point

```
#include <WiFi.h>
#include <HTTPClient.h> // Include the HTTPClient library for HTTPS support
#include <DHT.h>
// Dummy WiFi credentials (not used in the simulation)
const char* ssid = "your_network_name";
const char* password = "your_network_password";
// Beeceptor endpoint
const char* serverUrl = "https://esp32dht.free.beeceptor.com/"; // Use HTTPS for secure communication
// DHT sensor configuration
#define DHTPIN 4       // Define the GPIO pin to which the DHT22 is connected
#define DHTTYPE DHT22    // Define the sensor type (DHT11 or DHT22)
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(115200);
  // Dummy WiFi connection (not used in the simulation)
  Serial.println("Connecting to WiFi (simulated)...");
```

```
  delay(1000);
  Serial.println("Connected (simulated) to WiFi");
  // Initialize the DHT sensor
  dht.begin();
}
void loop() {
  // Read temperature and humidity
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // Create an HTTPS client using HTTPClient
  HTTPClient https;
  // Send temperature and humidity data to Beeceptor as form parameters
  https.begin(serverUrl);
  https.addHeader("Content-Type", "application/x-www-form-urlencoded");
  String postData = "temperature=" + String(temperature) + "&humidity=" +
String(humidity);
  int httpResponseCode = https.POST(postData);
  if (httpResponseCode > 0) {
    Serial.print("HTTP Response code: ");
```

```
    Serial.println(httpResponseCode);
    Serial.println("Data sent to Beeceptor.");
  } else {
    Serial.print("Error in HTTP request. HTTP Response code: ");
    Serial.println(httpResponseCode);
  }
  https.end();
  delay(60000); // Send data every 1 minute (adjust as needed)
}
```

==========