**IOT\_Phase4**

**(Environmental Monitoring)**

**Introduction:**

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:**

A computer screen shot of a computer

Description automatically generated

**Web Code:**

<!DOCTYPE html>

<html>

<head>

    <title>Environment Temperature and Humidity Values Using Arduino</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>Environment Temperature and Humidity Using Arduino</h1>

    <div class="container">

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

            <h2>Temperature:</h2>

            <p id="temp-value">Loading...</p>

        </div>

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

            <h2>Humidity:</h2>

            <p id="humidity-value">Loading...</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:**

**A screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated

**Conclusion:**

The IoT Environment Monitoring System provides real-time temperature and humidity data, which is sent to ThingSpeak for storage and analysis. This system is a practical and versatile solution for monitoring weather conditions and can be extended to include additional sensors and features.

By leveraging the power of IoT and cloud platforms, the system provides valuable data that can be used for various applications, such as home automation, agriculture, or weather tracking. This project demonstrates the potential of using affordable and readily available components to create a connected and useful IoT device