

IoT-Based Temperature and Humidity Monitoring System Report

Project Title: IoT-Based Temperature and Humidity Monitoring System

Presented By: **Group 31**

Shaganti Samhitha (21CSB0A52) CSE 'A'

Yashvini Katare (21CSB0A67) CSE 'A'

Savvy Jain (21CSB0F15) CSE 'A'

GITHUB LINK:

<https://github.com/savvy8j/lot-TemperatureAndHumidityMonitoringSystem/blob/main/README.md>

1. Project Overview

Problem Statement:

If humidity and temperature levels in classrooms and laboratories are not kept in check, students' comfort and health may be compromised, which decreases their productivity and overall wellness.

Objectives:

- Develop an IoT-based system to monitor temperature and humidity.
- Trigger alerts when values exceed set thresholds.
- Store and visualize data using cloud platforms.

Solution Explanation:

Why This IoT-based technology is chosen?

- *Wokwi(Simulation Platform)*:- No need for real hardware, making testing and debugging faster and cost-free.
- *ESP32 (Microcontroller)* :- Built-in WiFi, low power consumption, and better processing power than Arduino.
- *DHT22 (Sensor)* :-More accurate and stable than DHT11, measuring temperature and humidity.
- *ThingSpeak (IoT Cloud)* :- Free for basic use, real-time data monitoring, and easy ESP32 integration without extra libraries

A DHT22 sensor is used to obtain temperature and humidity readings as well as activate things when set parameters are met. Processed by the microcontroller (ESP32) the data is uploaded to ThingSpeak(cloud-based platform) for real time data retrieval. If temperature or humidity exceeds predefined limits, the system activates alerts via LEDs and a buzzer. An alert is sent via email to indicate limits exceeded.

2. Design & Implementation

Components Used:

- Temperature and humidity sensor (DHT22)
- Microcontroller (ESP32)
- Cloud platform: ThingSpeak for real-time data logging and alert generation
- Open-source simulators: Wokwi
- Additional components: LEDs, buzzer
- Resistor: 10k Ω
- Jumper Wires: For circuit connections.

Circuit Design:

- The circuit is implemented using Wokwii
- ThingSpeak API is used for real-time data logging.

A circuit is designed to integrate the DHT sensor with the microcontroller. The sensor reads temperature and humidity values, which are then sent to the cloud. If the readings exceed the threshold, LEDs change color, and a buzzer sounds an alarm.

The circuit consists of the following connections:

DHT22 Sensor

VCC → 3.3V (ESP32)

GND → GND (ESP32)

DATA → GPIO4 (ESP32)

Pull-up Resistor (10k Ω) connected between VCC and DATA

LEDs

Green LED (Normal Condition) → GPIO5

Red LED (Alert Condition) → GPIO18

Both LEDs' cathodes are connected to GND

Buzzer

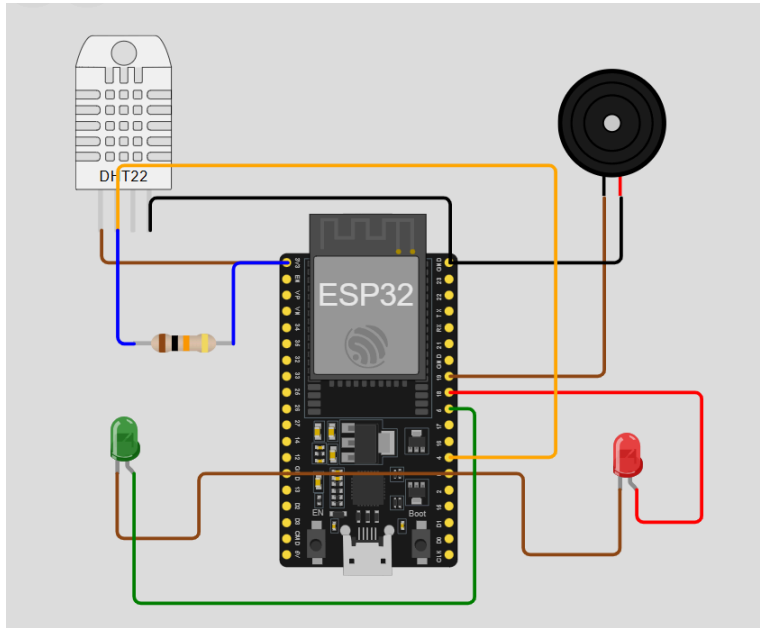
Positive Terminal → GPIO19

Negative Terminal → GND

WiFi Connection

ESP32 connects to the **Wokwi-GUEST** WiFi network

Data is sent to **ThingSpeak** cloud




Implementation Steps:

1. Connect the DHT22 sensor to the microcontroller.
 2. Establish a Wi-Fi connection for data transmission.
 3. Read temperature and humidity values from the sensor.
 4. Send the collected data to ThingSpeak.
 5. Define threshold values for temperature and humidity.
 6. Trigger alerts (LED and buzzer) when limits are exceeded.
 7. Visualize data through ThingSpeak dashboards.
 8. Email sent on exceeding limit as an Alert!
-

Configurations on ThingSpeak :

[←](#) [→](#) [🔄](#) [🔍](#) thingspeak.mathworks.com/channels/2898709/edit

 Channels ▾ Apps ▾ Devices ▾ Support ▾

Temperature and Humidity Monitoring

Channel ID: **2898709**
Author: **mwa0000037290213**
Access: Private

This channel collects real-time temperature and humidity data from the ESP32 microcontroller using a DHT sensor. The data is used for monitoring classroom/lab conditions.

[Private View](#) [Public View](#) [Channel Settings](#) [Sharing](#) [API Keys](#) [Data Import / Export](#)

Channel Settings

Percentage Complete 50%

Channel ID 2898709

Name

Description

Field 1 ☒

Field 2 ☒

Field 3 ☐

Field 4 ☐

Help

Channels store data and can hold any type of data in a channel, you can...

Channel Settings

- **Percentage Complete**: name, description, and...
- **Channel Name**: name, description, and...
- **Description**: name, description, and...
- **Field#**: Check up to 8 fields...
- **Metadata**: name, description, and...
- **Tags**: Enter tags...
- **Link to External**

[Apps](#) / [ThingHTTP](#)
[New ThingHTTP](#)

| Name | Created |
|---|------------|
| Gmail Alert | 2025-03-30 |
| View Edit | |

[Apps](#) / [ThingHTTP](#) / [Gmail Alert](#)
[Edit ThingHTTP](#)

Name: Gmail Alert

API Key: 66WA5SZX4CDZ2Y68

[Regenerate API Key](#)

URL: https://script.google.com/macros/s/AKfycbzxCWL9KY9sinazsTiGual0upkkjZHCoTImx2d7e_Higt4oKX0N-r9vNc9_qME24LM6/exec

HTTP Auth Username:

HTTP Auth Password:

Method: GET

Content Type:

HTTP Version: 1.1

[Apps](#) / [React](#)
[New React](#)

| Name | Created | Last Ran |
|--|------------|--------------------|
| <input checked="" type="checkbox"/> React 1 <div>View Edit</div> | 2025-03-30 | 2025-04-02 5:48 am |
| <input checked="" type="checkbox"/> React 2 <div>View Edit</div> | 2025-03-30 | 2025-04-02 5:48 am |

[Apps](#) / [React](#) / [React 1](#)
[Edit React](#)


| | |
|-----------------|--|
| Name: | React 1 |
| Condition Type: | Numeric |
| Test Frequency: | On data insertion |
| Last Ran: | 2025-04-02 05:48 |
| Channel: | Temp_Humidity_Monitoring |
| Condition: | Field 1 (Temperature(°C)) is greater than 25 |
| ThingHTTP: | Gmail Alert |
| Run: | Only the first time the condition is met |
| Created: | 2025-03-30 7:20 pm |

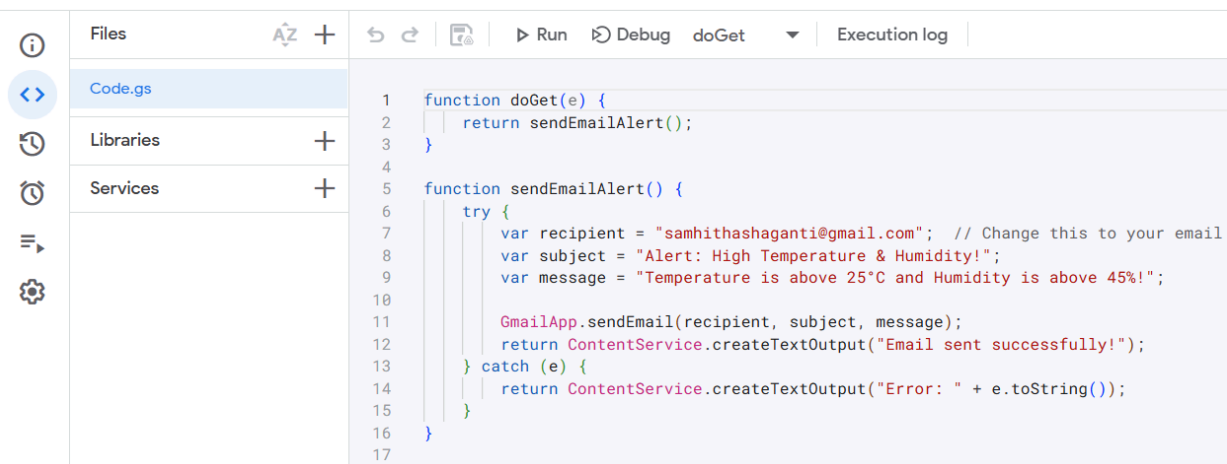
Apps / React / React 2

Edit React

| | |
|-----------------|--|
| Name: | React 2 |
| Condition Type: | Numeric |
| Test Frequency: | On data insertion |
| Last Ran: | 2025-04-02 05:48 |
| Channel: | Temp_Humidity_Monitoring |
| Condition: | Field 2 (Humidity(%)) is greater than 45 |
| ThingHTTP: | Gmail Alert |
| Run: | Only the first time the condition is met |
| Created: | 2025-03-30 7:45 pm |

Project code in Google Apps Script :

 Apps Script Email Alerts for Temp and hum... 



```
1  function doGet(e) {
2      return sendEmailAlert();
3  }
4
5  function sendEmailAlert() {
6      try {
7          var recipient = "samhithashaganti@gmail.com"; // Change this to your email
8          var subject = "Alert: High Temperature & Humidity!";
9          var message = "Temperature is above 25°C and Humidity is above 45%!";
10
11          GmailApp.sendEmail(recipient, subject, message);
12          return ContentService.createTextOutput("Email sent successfully!");
13      } catch (e) {
14          return ContentService.createTextOutput("Error: " + e.toString());
15      }
16  }
17
```

3. Code & Simulation Files(Uploaded on GClassroom)

Source Code:

The following C++ code collects temperature and humidity data, sends it to ThingSpeak, and triggers alerts when thresholds are exceeded

```
#include <WiFi.h>

#include "DHT.h"

#include "ThingSpeak.h"

#define DHTPIN 4

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

#define GREEN_LED 5

#define RED_LED 18

#define BUZZER 19

#define TEMP_THRESHOLD 25.0

#define HUM_THRESHOLD 45.0

const char* WIFI_NAME = "Wokwi-GUEST";

const char* WIFI_PASSWORD = "";

const int channelNumber = 2898709;

const char* myApiKey = "3J5Z4E94VE6YQO2U";

const char* server = "api.thingspeak.com";

WiFiClient client;

void connectWiFi() {

    Serial.print("Connecting to WiFi: ");
```



```
Serial.println(WIFI_NAME);

WiFi.mode(WIFI_STA);

WiFi.begin(WIFI_NAME, WIFI_PASSWORD);

int attempt = 0;

while (WiFi.status() != WL_CONNECTED && attempt < 20) {

    delay(1000);

    Serial.print(".");

    attempt++;

} if (WiFi.status() == WL_CONNECTED) {

    Serial.println("\nWiFi Connected!");

    Serial.print("Local IP: ");

    Serial.println(WiFi.localIP());

} else {

    Serial.println("\nWiFi Connection Failed! Restarting...");

    ESP.restart();

}

}

void setup() {

    Serial.begin(115200);

    dht.begin();

    pinMode(GREEN_LED, OUTPUT);

    pinMode(RED_LED, OUTPUT);

    pinMode(BUZZER, OUTPUT);

    connectWiFi();
```

```
ThingSpeak.begin(client);

digitalWrite(GREEN_LED, HIGH);

digitalWrite(RED_LED, LOW);

digitalWrite(BUZZER, LOW);

}

void loop() {

    float temp = dht.readTemperature();

    float hum = dht.readHumidity();

    if (isnan(temp) || isnan(hum)) {

        Serial.println("Failed to read from DHT sensor!");

        return;
    }   Serial.print("Temperature: ");

    Serial.print(temp);

    Serial.print(" °C | Humidity: ");

    Serial.print(hum);

    Serial.println(" %");

    if (temp > TEMP_THRESHOLD || hum > HUM_THRESHOLD) {

        digitalWrite(GREEN_LED, LOW);

        digitalWrite(RED_LED, HIGH);

        Serial.println("ALERT! High Temperature or Humidity!");

        for (int i = 0; i < 5; i++) {

            digitalWrite(BUZZER, HIGH);

            delay(200);
```

```

        digitalWrite(BUZZER, LOW);

        delay(200);

    }

} else {

    digitalWrite(GREEN_LED, HIGH);

    digitalWrite(RED_LED, LOW);

    digitalWrite(BUZZER, LOW);

} ThingSpeak.setField(1, temp);

ThingSpeak.setField(2, hum);

int check = ThingSpeak.writeFields(channelNumber, myApiKey);

    if (check == 200) {

        Serial.println("Data pushed successfully");

    } else {

        Serial.println("Push error: " + String(check));

    }

    delay(2000);

}

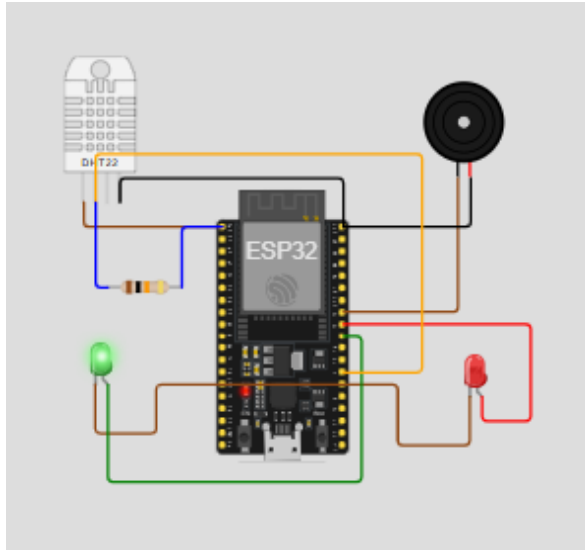
*****

```

4. Results & Testing

Simulator Results:

Case 1:When the Temperature and Humidity levels<= Threshold



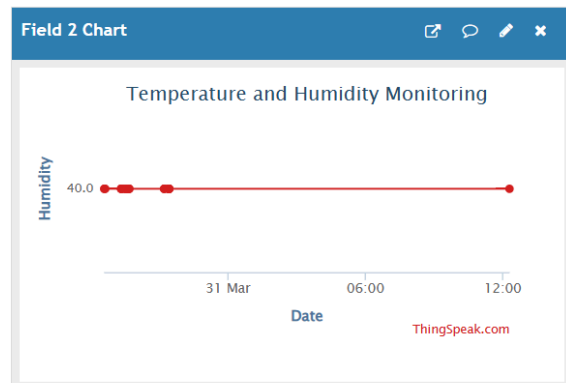
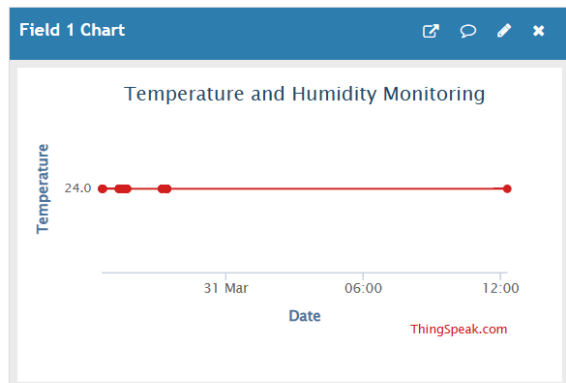
Green LED 'ON', Red LED 'OFF' ,Buzzer 'OFF'

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
Connecting to WiFi: Wokwi-GUEST
..
WiFi Connected!
Local IP: 10.10.0.2
DHT22 Sensor Simulation on Wokwi!
Temperature: 24.00 °C | Humidity: 40.00 %
Data pushed successfully
****
```

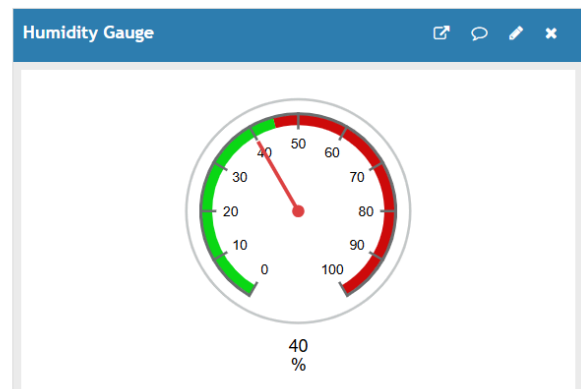
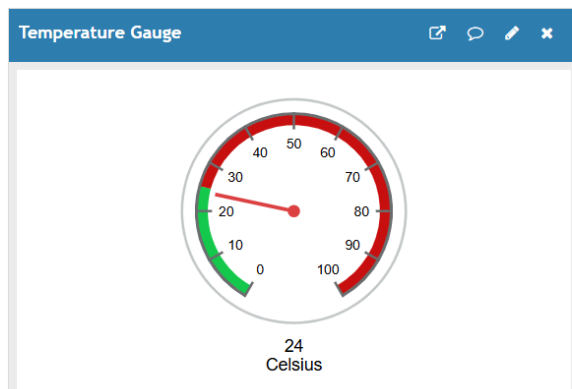
ThingSpeak Results:

Case 1:When the Temperature and Humidity levels \leq Threshold

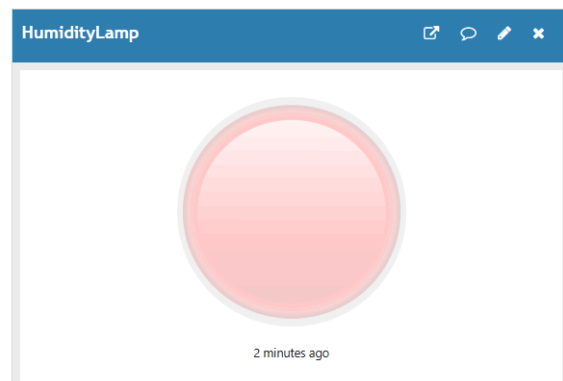
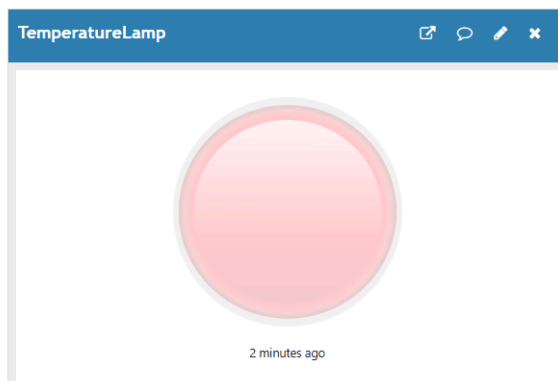
Line Charts



Gauge Readings

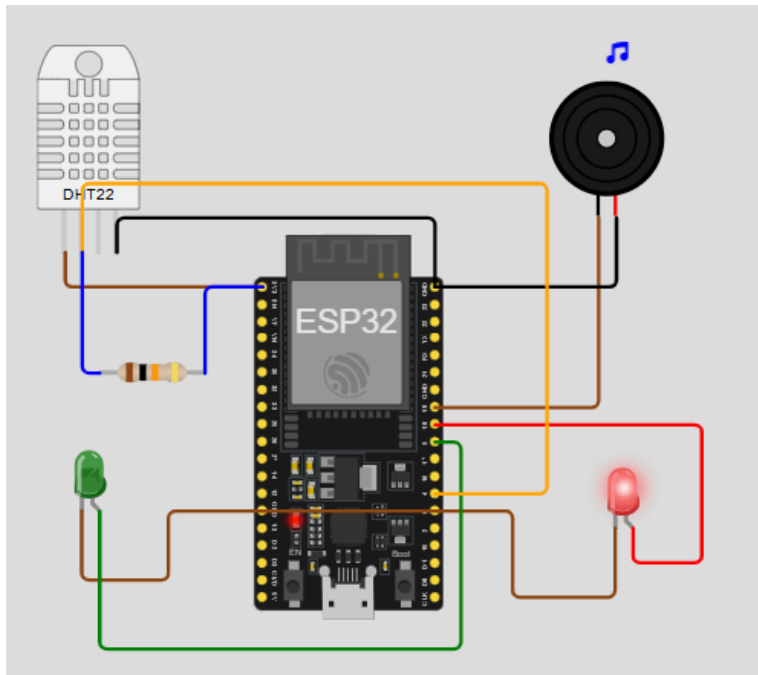


Lamp Readings(OFF here as threshold NOT exceeded)



Case 2:When the Temperature and Humidity levels> Threshold

Simulator Results:

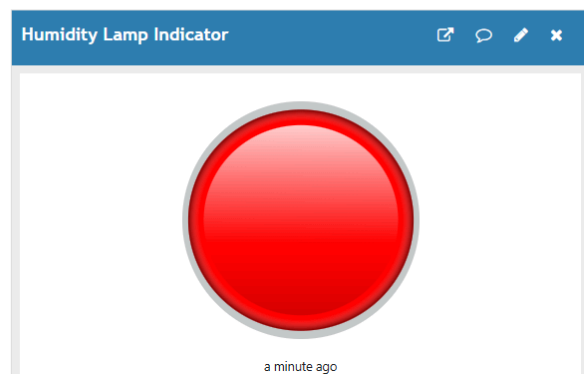
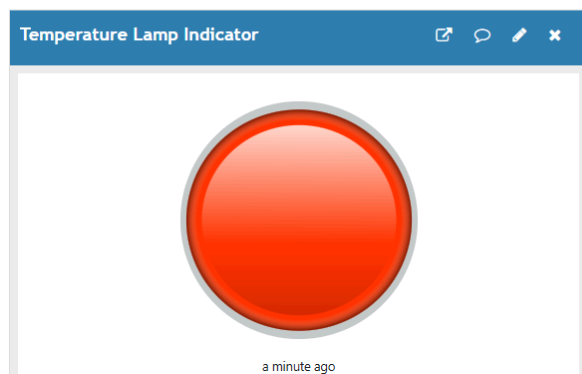
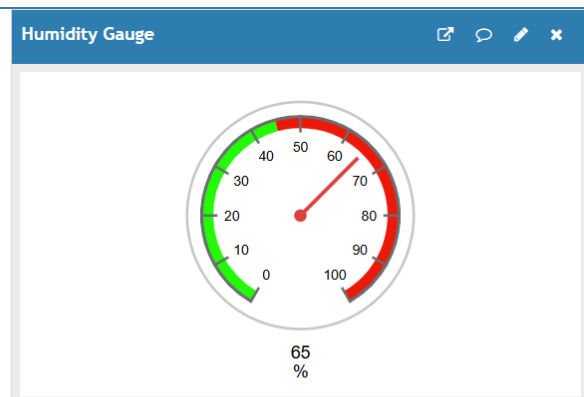
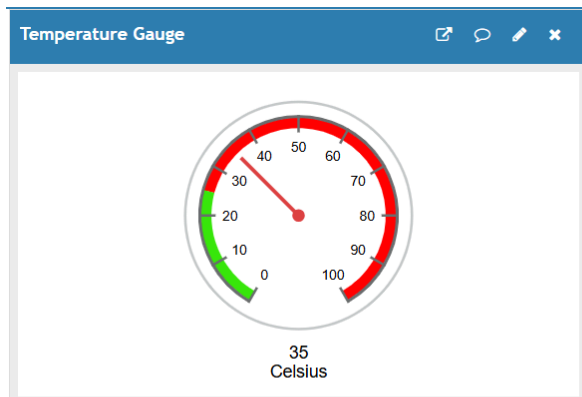
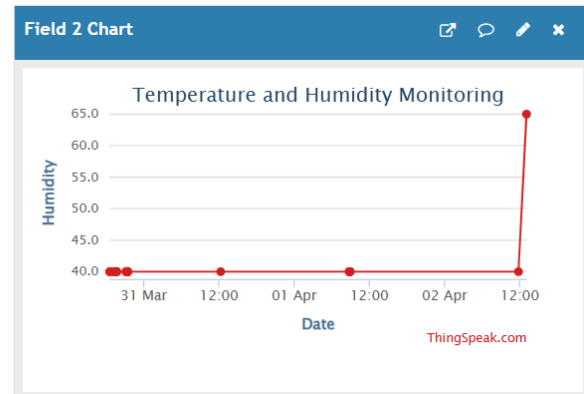
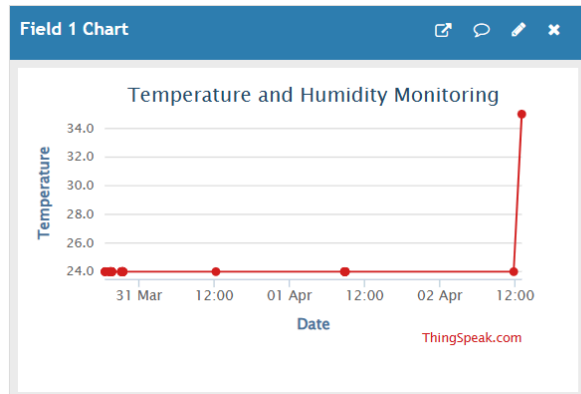


Green LED '**OFF**', Red LED '**ON**' ,Buzzer '**ON**'

->As Wokwi is a Visual Simulator, we cannot hear the buzzer beeping, but the music notes emerging out of it showcase that it is on.

```
Temperature: 35.00 °C | Humidity: 65.00 %  
⚠ ALERT! High Temperature or Humidity!  
Data pushed successfully  
****
```

ThingSpeak Results:

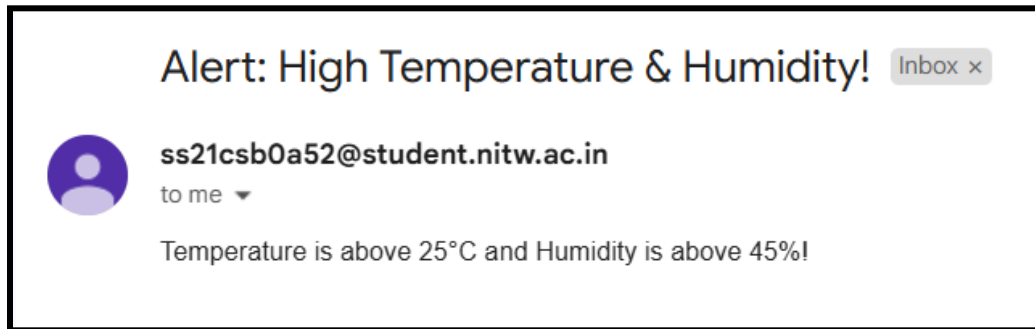


Lamp Readings(ON here as threshold exceeded)

Email Alert Screenshot:

☐ ☆ ss21csb0a52

Alert: High Temperature & Humidity! - Temperature is above 25°C and Humidity is above 45%!



5. Conclusion

This IoT-based system provides a simple yet effective solution for monitoring classroom temperature and humidity. It ensures comfort and safety by triggering real-time alerts when thresholds are exceeded. Future enhancements may include:

- Integration with mobile notifications.
- Predictive maintenance using data analytics.
- AI-based temperature control mechanisms.

6. Group Contributions & Peer Review

Samhitha Shaganti(21CSB0A52)

- Worked on Arduino Code(.ino) for exceeding Thresholds .
- Setting up thingHTTP and React apps in ThingSpeak .
- Connected thingHTTP to Google Apps Script for sending email alerts.
- Observing the alerts and respective readings due to exceeding thresholds on ThingSpeak.
- Worked on PPT(Approach and Results)
- Worked on Report(Design and Implementation,Results part)

Savvy Jain (21CSB0F15)

- Connected the components in Wokwi platform.
- Worked on arduino code(.ino)
- Connecting wokwi with thingspeak
- Creating channels on ThingSpeak and observing the graphs and gauge readings and lamp indicators.
- Worked on PPT(Design and Implementation)
- Worked on Report(Design and Implementation,Results part)

Yashsvini Katare (21CSB0A67)

- Part of report and ppt
- Tried code also

7.Resources:

- <https://iotdesignpro.com/projects/temperature-humidity-monitoring-over-thingspeak-using-arduino-esp8266>
- <https://www.youtube.com/watch?v=8JdOLlxw9Yc>