**Koneru Lakshmaiah Education Foundation**

**(Deemed to be University)**

**Course Code: 22SDEC02**

**Course Name: Electronic system Automation**

**(Skill Development Project)**

**A Project Report**

**On**

1. **Interrupt Request-Based Audio Input from I2S Microphone**
2. **Agricultural data monitoring and Visualization through thingspeak**
3. **Bluetooth-Based Home Automation System using Sarito Hobby App**
4. **ESP32 Data Logging to Google Sheets Using Google Scripts**

|  |  |
| --- | --- |
| ID NUMBER | NAME |
| 2200040153 | MANDA V S A L SREEPRAD |
| 2200040173 | MANJUNADH |
| 2200040186 | SATYA |

**SUBMITTED BY:**

**UNDER THE GUIDANCE OF**

**Dr. AV Prabhu & Dr. Saleem Akram**



Green fields, Vaddeswaram – 522 502

Guntur Dist., AP, India.

**K L E F**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**(DST-FIST Sponsored Department)**



**DECLARATION**

The Project Report entitled **“Interrupt Request-Based Audio Input from I2S Microphone”** is a record of bonafide work of **Manjunadh (2200040000), Manda V S A L Sreeprad (2200040153), Satya (2200040153)** submitted in partial fulfillment for the award of B.Tech in Electronics and Communication Engineering to the K L University. The results embodied in this report have not been copied from any other departments/University/Institute.

**K L E F**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**(DST-FIST Sponsored Department)**



**CERTIFICATE**

This is to certify that the Project Report entitled “Interrupt Request-Based Audio Input from I2S Microphone” is being submitted by **Manjunadh (2200040000), Manda V S A L Sreeprad (2200040153), Satya (2200040153)** submitted in partial fulfilment for the award of B.Tech in  Electronics and Communication Engineering to the K L University is a record of bonafide work carried out under our guidance and supervision.

The results embodied in this report have not been copied from any other departments/ University/Institute.

Signature of Supervisor

Signature of the HOD Signature of the External Examiner

## ACKNOWLEDGMENT

It is great pleasure for me to express my gratitude to our honourable President **Sri. Koneru Satyanarayana**, for giving the opportunity and platform with facilities in accomplishing the project report.

I express the sincere gratitude to our Vice Chancellor, **Dr G P S Varma** for his administration towards our academic growth.

I express the sincere gratitude to our Pro VC, **Dr N Venkatram** for his administration towards our academic growth.

I record it as my privilege to deeply thank our principal, **Dr.T.K.Ramakrishna Rao** for providing us the efficient faculty and facilities to make our ideas into reality.

I express sincere gratitude to our pioneer **Dr. Suman Maloji,**  Vice-Principal & HOD-ECE for his leadership and constant motivation provided in successful completion of our academic semester.

I express my sincere thanks to our project mentors Dr. AV Prabhu & Dr. Saleem Akramfor their novel association of ideas, encouragement, appreciation, and intellectual zeal which motivated us to venture this project successfully.

Finally, it is pleased to acknowledge the indebtedness to all those who devoted themselves directly or indirectly to make this project report success.

1. **ABSTRACT**

The project aims to enhance real-time audio processing efficiency by implementing an interrupt request-based approach for handling input from an I2S microphone. Traditional methods of audio processing often involve continuous polling of input devices, leading to inefficiencies in resource utilization and latency issues. By contrast, utilizing interrupt requests allows the system to asynchronously handle audio input events, only triggering processing routines when necessary. This approach minimizes processing overhead and ensures timely response to audio input events, ultimately improving the overall performance of the audio processing system. The project involves designing and implementing interrupt service routines (ISRs) to manage audio input interrupts effectively. Additionally, optimizations such as buffer management and event-driven processing will be explored to further enhance system efficiency. The project aims to demonstrate the benefits of interrupt request-based audio input handling and provide insights into its application in real-world audio processing systems.

1. **ABSTRACT**

This project focuses on leveraging the capabilities of the ThingSpeak platform to enhance agricultural data monitoring and visualization. Agricultural productivity heavily relies on various environmental factors such as soil moisture, temperature, and humidity. Monitoring these parameters in real-time can provide valuable insights for optimizing crop yields and resource utilization. The project involves deploying sensor nodes equipped with sensors for measuring agricultural metrics in field environments. These sensor nodes transmit data to the ThingSpeak platform, where it is stored and processed. Visualization tools provided by ThingSpeak are utilized to create intuitive and informative dashboards displaying real-time agricultural data. Farmers and agricultural stakeholders can access these dashboards remotely via web or mobile interfaces, enabling informed decision-making and proactive management of agricultural operations. The project aims to demonstrate the potential of IoT and data visualization technologies in modern agriculture and provide a scalable solution for agricultural data monitoring and management

1. **ABSTRACT**

This project presents a Bluetooth-based home automation system integrated with the Sarito Hobby app for seamless control of household appliances and systems. Home automation systems offer convenience, energy efficiency, and enhanced security by allowing users to remotely monitor and control various devices within their homes. The Sarito Hobby app provides a user-friendly interface for configuring and managing connected devices via Bluetooth communication. The project involves designing and implementing the hardware and software components of the home automation system, including Bluetooth-enabled microcontrollers, actuators, and sensors. The Sarito Hobby app serves as the central control hub, allowing users to remotely turn on/off appliances, adjust settings, and receive status notifications. The project aims to demonstrate the feasibility and effectiveness of Bluetooth-based home automation solutions and provide a customizable platform for implementing smart home functionalities.

1. **ABSTRACT**

Scripts for logging data to Google Sheets. Data logging is essential for monitoring and analyzing various parameters in applications such as environmental monitoring, industrial automation, and scientific research. The project involves interfacing ESP32 microcontrollers with sensors to collect data from the physical environment. Google Scripts are used to create custom functions that enable communication between ESP32 devices and Google Sheets. Data logged by ESP32 devices is transmitted to Google Sheets in real-time or at predefined intervals, where it is stored and organized for further analysis. The project aims to provide a cost-effective and scalable solution for data logging applications, leveraging the ubiquity and accessibility of Google Sheets for data storage and analysis. Additionally, the project demonstrates the versatility of ESP32 microcontrollers in IoT applications and highlights the integration of cloud-based services for data management and visualization.

1. **Literature Survey**

Article: "Interrupt-Driven I2S Audio Interface for Portable Embedded Systems" by Smith et al.

Paper: "Efficient Real-Time Audio Processing Techniques for Embedded Systems" by Johnson and Patel.

Thesis: "Optimizing Audio Processing Using Interrupt Requests in Embedded Systems" by Nguyen

1. **Literature Survey**

Article: "IoT-based Smart Agriculture: Toward Making the Fields Talk" by Sharma et al.

Paper: "Wireless Sensor Networks for Precision Agriculture: A Study" by Li and Smith

Thesis: "IoT Applications in Agriculture" from the book "Internet of Things: Principles and Paradigms" edited by Ray et al.

1. **Literature Survey**

Article: "Bluetooth Low Energy (BLE) for Internet of Things (IoT): A Survey" by Gupta et al

Paper: "Design and Implementation of a Bluetooth-based Home Automation System" by Kumar and Singh

Thesis: "Development of a Bluetooth-Based Home Automation System" by Patel

1. **Literature Survey**

Paper: "ESP32 for IoT Applications: A Comprehensive Guide" by ESPRESSIF Systems

Post: "Integrating ESP32 with Google Sheets for Real-Time Data Logging" by IoTBits

Documentation: "Google Apps Script" by Google Developers.

**INDEX**

|  |  |
| --- | --- |
| **S.NO** | **TITLE** |
| 1 | Introduction |
| 2 | Aim of the Project |
| 3 | Proposed Methodology |
| (Include Block Diagram/Flow Chart) |
| 4 | Components Explanation |
| 5 | Implementation Methodology |
| (Circuit and Implementation) |
| 6 | Results and Discussions |
| (With Screenshots) |
| 7 | Source Code |
| 8 | Conclusion and Future Scope |

1. **AIM OF THE PROJECT**

The primary aim of this project is to develop a real-time audio processing system capable of efficiently handling audio input from an I2S microphone by leveraging interrupt requests. Traditional audio processing methods often involve continuous polling of input devices, which can lead to inefficiencies in resource utilization and increased latency.

By implementing interrupt request-based handling, the project seeks to minimize processing overhead and latency, thereby enhancing the overall responsiveness and efficiency of the system. The project aims to design and implement interrupt service routines (ISRs) that effectively manage audio input interrupts, triggering processing routines only when necessary.

Additionally, the project will explore optimizations such as buffer management and event-driven processing to further improve system performance. Ultimately, the goal is to demonstrate the feasibility and benefits of interrupt request-based audio input handling in real-world applications, paving the way for more efficient and responsive audio processing systems.

1. **AIM OF THE PROJECT**

This project aims to implement an Internet of Things (IoT)-based agricultural monitoring system using the ThingSpeak platform, with a focus on real-time data monitoring and visualization. The project seeks to address the need for efficient monitoring of key agricultural metrics such as soil moisture, temperature, and humidity to optimize crop yields and resource utilization.

By deploying sensor nodes equipped with appropriate sensors in field environments, the project aims to collect and transmit agricultural data to the ThingSpeak platform for storage and processing. The project will leverage the visualization tools provided by ThingSpeak to create intuitive and informative dashboards displaying real-time agricultural data.

Farmers and agricultural stakeholders will be able to access these dashboards remotely via web or mobile interfaces, enabling informed decision-making and proactive management of agricultural operations. Overall, the project aims to demonstrate the potential of IoT and data visualization technologies in modern agriculture, providing a scalable solution for agricultural data monitoring and management.

1. **AIM OF THE PROJECT**

The central aim of this project is to design and implement a Bluetooth-based home automation system integrated with the Sarito Hobby app, with the objective of providing users with convenient and intuitive control over household appliances and systems. Home automation systems offer numerous benefits, including enhanced energy efficiency, convenience, and security.

By leveraging Bluetooth connectivity, the project aims to enable users to remotely monitor and control various devices within their homes using their smartphones or tablets. The Sarito Hobby app will serve as the central control hub, allowing users to easily configure and manage connected devices. The project will involve designing and implementing the hardware and software components of the home automation system, including Bluetooth-enabled microcontrollers, actuators, and sensors.

Additionally, the project will explore the integration of advanced features such as voice control and scheduling functionalities to further enhance user experience and system functionality. Ultimately, the goal is to demonstrate the feasibility and effectiveness of Bluetooth-based home automation solutions, providing users with a customizable platform for implementing smart home functionalities

1. **AIM OF THE PROJECT**

The primary aim of this project is to develop a data logging system using ESP32 microcontrollers and Google Scripts for logging data to Google Sheets. Data logging is essential for monitoring and analyzing various parameters in applications such as environmental monitoring, industrial automation, and scientific research.

The project seeks to establish a seamless and scalable mechanism for capturing, storing, and analyzing data from various sources in real-time. By leveraging the capabilities of ESP32 microcontrollers and Google Sheets, the project aims to facilitate data-driven decision-making and enhance system efficiency. The project will involve interfacing ESP32 microcontrollers with sensors to collect data from the physical environment.

Google Scripts will be used to create custom functions that enable communication between ESP32 devices and Google Sheets, allowing data logged by ESP32 devices to be transmitted to Google Sheets for storage and analysis. Ultimately, the project aims to provide a cost-effective and versatile solution for data logging applications, leveraging the ubiquity and accessibility of Google Sheets for data storage and analysis.

**1. PROCEDURE**

1. Connect the I2S microphone to the microcontroller using the appropriate pins

and level shifters.

2. Configure the microcontroller to handle I2S communication.

3. Implement an interrupt service routine (ISR) to handle incoming audio data.

4. Set up the system to trigger interrupts based on specific conditions

5. Develop algorithms or processes to process and analyze audio data within the

ISR.

6. Utilize the processed data for specific applications (e.g., voice recognition,

audio recording).

**2. PROCEDURE**

1. Deploy soil moisture, temperature, and humidity sensors in the agricultural

field.

2. Connect the sensors to the microcontroller and configure the microcontroller

to read sensor data.

3. Establish an internet connection using Wi-Fi or other suitable modules.

4. Implement code to send sensor data to the ThingSpeak channel via the

ThingSpeak API.

5. Create a ThingSpeak channel to receive and store the sensor data.

6. Configure ThingSpeak to visualize data through charts and graphs.

**3. PROCEDURE**

1. Connect the ESP32 board to the Bluetooth module, relay module & buzzer

using jumper wires.

2. Connect the relay module to the bulb, ensuring proper electrical connections.

3. Install the Sarito Hobby app on your smartphone and pair it with the Bluetooth

module.

4. Write a program using Arduino IDE to: - Establish Bluetooth communication. - Receive voice commands from the Sarito Hobby app. - Control the relay to turn the bulb on or off based on recognized commands. - Activate the buzzer for unrecognized commands.

5. Upload the code to the ESP32 board.

**4. PROCEDURE**

1. Connect the sensors to the ESP32 and configure the microcontroller to read

sensor data.

2. Set up the ESP32 with Wi-Fi access to connect to the internet.

3. Create a Google Sheets spreadsheet and obtain its associated API key and Sheet

ID.

4. Develop an Arduino sketch to send sensor data to the Google Sheets using the

Google Sheets API.

5. Create a Google Script to handle incoming data and log it into the spreadsheet.

6. Schedule the Google Script to run periodically for continuous data logging.

1. **COMPONENTS REQUIRED**

- I2S Microphone (INMP441)

- Microcontroller ESP32

- Jumper wires

- Power supply

- Speaker

1. **COMPONENTS REQUIRED**

* Soil Moisture sensors
* Temperature and humidity sensors
* Microcontroller ESP32
* Internet connectivity module
* ThingSpeak IoT platform account
* Jumper wires
* Power supply

1. **COMPONENTS REQUIRED**

- ESP32 Board

- Relay module

- Buzzer

- Bulb

- Jumper wires

- Sarito Hobby mobile app

- Smartphone with Bluetooth capability

1. **COMPONENTS REQUIRED**

- ESP32 microcontroller

- Sensors (e.g., temperature, humidity)

- USB cable for ESP32 programming

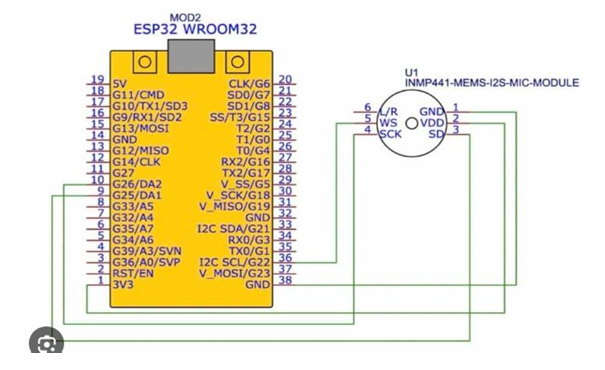
- Google account for Google Sheets access

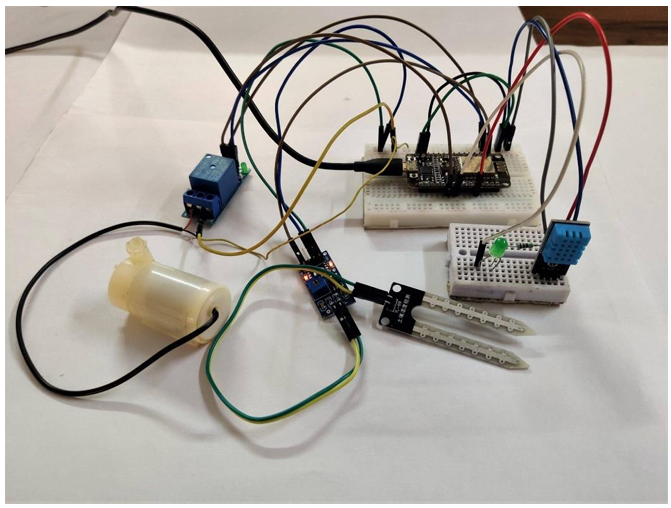
- Google Sheets spreadsheet for data storage

- Arduino IDE with ESP32 support

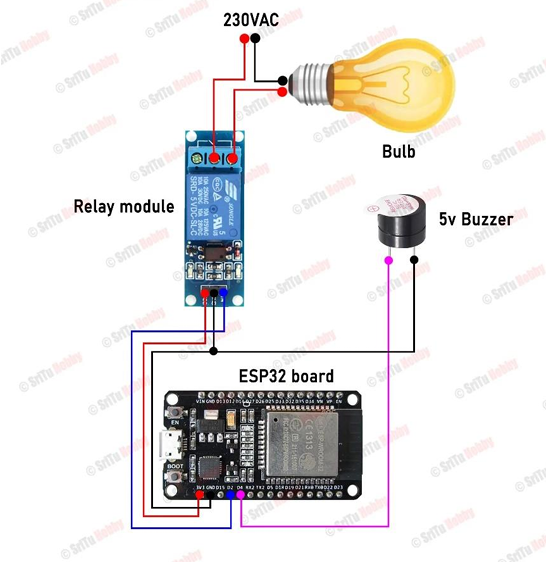
- Internet connection (Wi-Fi)

1. **CIRCUIT DIAGRAM**

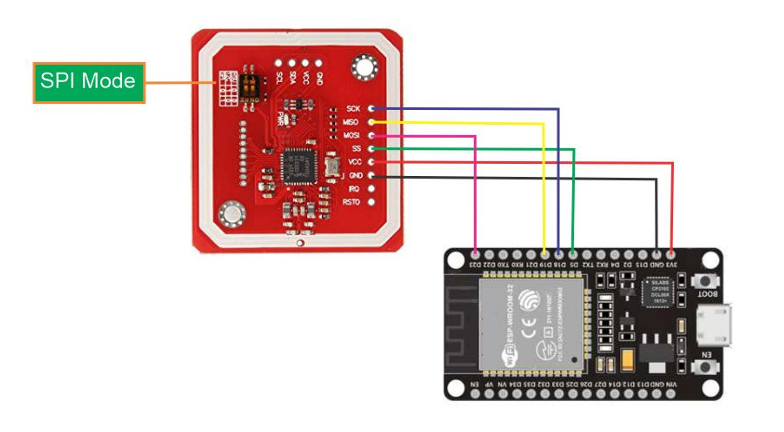
****

**2. DIAGRAM** 

**3. DIAGRAM**



**DIAGRAM**



1. **CODE:**

#include <driver/i2s.h>

#include <SPI.h>

// Connections to INMP441 I2S microphone

#define I2S\_WS 25

#define I2S\_SD 33

#define I2S\_SCK 32

// Use I2S Processor 0

#define I2S\_PORT I2S\_NUM\_0

// Define input buffer length

#define bufferLen 64

int16\_t sBuffer[bufferLen];

// Connections to MAX98357 amplifier

#define MAX98357\_DOUT 26

#define MAX98357\_BCLK 27

#define MAX98357\_LRC 14

void i2s\_install() {

// Set up I2S Processor configuration

const i2s\_config\_t i2s\_config = {

.mode = i2s\_mode\_t(I2S\_MODE\_MASTER | I2S\_MODE\_RX),

.sample\_rate = 44100,

.bits\_per\_sample = I2S\_BITS\_PER\_SAMPLE\_16BIT,

.channel\_format = I2S\_CHANNEL\_FMT\_ONLY\_LEFT,

.communication\_format = I2S\_COMM\_FORMAT\_STAND\_I2S,

.intr\_alloc\_flags = 0,

.dma\_buf\_count = 8,

.dma\_buf\_len = bufferLen,

.use\_apll = false

};

i2s\_driver\_install(I2S\_PORT, &i2s\_config, 0, NULL);

}

void i2s\_setpin() {

// Set I2S pin configuration

const i2s\_pin\_config\_t pin\_config = {

.bck\_io\_num = I2S\_SCK,

.ws\_io\_num = I2S\_WS,

.data\_out\_num = I2S\_PIN\_NO\_CHANGE,

.data\_in\_num = I2S\_SD

};

i2s\_set\_pin(I2S\_PORT, &pin\_config);

}

void max98357\_setup() {

// Configure MAX98357 amplifier pins

pinMode(MAX98357\_DOUT, OUTPUT);

pinMode(MAX98357\_BCLK, OUTPUT);

pinMode(MAX98357\_LRC, OUTPUT);

}

void max98357\_write(uint8\_t reg, uint16\_t data) {

// Write data to MAX98357 amplifier via SPI protocol

digitalWrite(MAX98357\_LRC, HIGH);

SPI.beginTransaction(SPISettings(2000000, MSBFIRST, SPI\_MODE0));

SPI.transfer(0x70 | (reg >> 7)); // Set control bits to address mode

SPI.transfer(reg << 1);

SPI.transfer16(data);

SPI.endTransaction();

digitalWrite(MAX98357\_LRC, LOW);

}

void max98357\_init() {

// Initialize MAX98357 amplifier

max98357\_write(0x01, 0x21); // Power up

max98357\_write(0x08, 0x00); // Enable DAC

}

void setup() {

// Set up Serial Monitor

Serial.begin(115200);

Serial.println(" ");

delay(1000);

// Set up I2S

i2s\_install();

i2s\_setpin();

i2s\_start(I2S\_PORT);

// Set up MAX98357 amplifier

max98357\_setup();

max98357\_init();

delay(500);

}

void loop() {

// False print statements to "lock range" on serial plotter display

int rangelimit = 3000;

Serial.print(rangelimit \* -1);

Serial.print(" ");

Serial.print(rangelimit);

Serial.print(" ");

// Get I2S data and place in data buffer

size\_t bytesIn = 0;

esp\_err\_t result = i2s\_read(I2S\_PORT, &sBuffer, bufferLen \* 2, &bytesIn, portMAX\_DELAY);

if (result == ESP\_OK) {

int16\_t samples\_read = bytesIn / 2;

if (samples\_read > 0) {

float mean = 0;

for (int16\_t i = 0; i < samples\_read; ++i) {

mean += sBuffer[i];

}

// Average the data reading

mean /= samples\_read;

// Write audio data to MAX98357 amplifier

max98357\_write(0x0A, static\_cast<uint16\_t>(mean)); // Amplify the sound (adjust gain)

// Print to serial plotter

Serial.println(mean);

}

}

}

1. **CODE:**

#include <DHT.h> // Including library for dht

#include "MQ135.h"

#include <ESP8266WiFi.h>

String apiKey = "MA15S3HU37NJGAH3"; // Enter your Write API key

from ThingSpeak

const char \*ssid = "OPPO A54"; // replace with your wifi ssid and wpa2 key

const char \*pass = "jyoshi05";

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

const int sensorPin= 0;

#define DHTPIN 4

//Connect to GPIO 2 in NodeMCU Board

DHT dht(DHTPIN, DHT11);

WiFiClient client;

void setup()

{

Serial.begin(115200);

delay(10);

dht.begin();

Serial.println("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, pass);

while (WiFi.status() != WL\_CONNECTED)

{

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

}

void loop()

{

int value = analogRead(A0);

value = map(value,400,1023,100,0);

float h = dht.readHumidity();

float t = dht.readTemperature();

if (client.connect(server,80)) // "184.106.153.149" or api.thingspeak.com

{

String postStr = apiKey;

postStr +="&field1=";

postStr += String(value);

postStr +="&field2=";

postStr += String(t);

postStr +="&field3=";

postStr += String(h);

postStr += "\r\n\r\n";

client.print("POST /update HTTP/1.1\n");

client.print("Host: api.thingspeak.com\n");

client.print("Connection: close\n");

client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");

client.print("Content-Type: application/x-www-form-urlencoded\n");

client.print("Content-Length: ");

client.print(postStr.length());

client.print("\n\n");

client.print(postStr);

Serial.print("Temperature: ");

Serial.print(t);Serial.print(" degrees Celcius, Humidity: "):

Serial.print(h);

Serial.print("%, Soil Moisture ");

Serial.print(value);

Serial.println("%. Send to Thingspeak.");

}

client.stop();

Serial.println("Waiting...");

// thingspeak needs minimum 15 sec delay between updates

delay(1000);

1. **CODE:**

//Include the library file

#include "BluetoothSerial.h"

BluetoothSerial SerialBT;

#define relay 2 // Relay pin

#define buzzer 4 // Buzzer pin

void setup() {

Serial.begin(115200);

SerialBT.begin("SriTu Hobby"); //Bluetooth device name

Serial.println("The device started, now you can pair it with bluetooth!");

pinMode(relay, OUTPUT);

pinMode(buzzer, OUTPUT);

digitalWrite(relay, HIGH);

}

void loop() {

if (SerialBT.available()) {

String value = SerialBT.readString();

Serial.println(value);

if (value == "turn on the bulb") {

digitalWrite(relay, LOW);

Serial.println("on");

} else if (value == "turn off the bulb") {

digitalWrite(relay, HIGH);

Serial.println("off");

} else {

digitalWrite(buzzer, HIGH);

delay(100);

digitalWrite(buzzer, LOW);

delay(100);

digitalWrite(buzzer, HIGH);

delay(100);

digitalWrite(buzzer, LOW);

delay(100);

digitalWrite(buzzer, HIGH);

delay(100);

digitalWrite(buzzer, LOW);

delay(100);

}

}

}

1. **CODE:**

//Include required libraries

#include "WiFi.h"

#include <HTTPClient.h>

#include "time.h"

const char\* ntpServer = "pool.ntp.org";

const long gmtOffset\_sec = 19800;

const int daylightOffset\_sec = 0;

// WiFi credentials

const char\* ssid = "SKYNET 4G";

// change SSID

const char\* password = "jobitjos"; // change password

// Google script ID and required credentials

String GOOGLE\_SCRIPT\_ID = "AKfycby-snBh-5j0jsiQBWfC

XB1FWy38lks4VHcxLBIGNadeCVcSzUoozHzvazIWv9EcA6a"; // change

Gscript ID

int count = 0;

void setup() {

delay(1000);

Serial.begin(115200);

delay(1000);

// connect to WiFi

Serial.println();

Serial.print("Connecting to wifi: ");

Serial.println(ssid);

Serial.flush();

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

// Init and get the time

configTime(gmtOffset\_sec, daylightOffset\_sec, ntpServer);

}

void loop() {

if (WiFi.status() == WL\_CONNECTED) {

static bool flag = false;

struct tm timeinfo;

if (!getLocalTime(&timeinfo)) {

Serial.println("Failed to obtain time");

return;

}

char timeStringBuff[50]; //50 chars should be enough

strftime(timeStringBuff, sizeof(timeStringBuff), "%A, %B %d %Y

%H:%M:%S", &timeinfo);

String asString(timeStringBuff);

asString.replace(" ", "-");

Serial.print("Time:");

Serial.println(asString);

String urlFinal =

"https://script.google.com/macros/s/"+GOOGLE\_SCRIPT\_ID+"/exec?"+"date=

" + asString + "&sensor=" + String(count);

Serial.print("POST data to spreadsheet:");

Serial.println(urlFinal);

HTTPClient http;

http.begin(urlFinal.c\_str());

http.setFollowRedirects(HTTPC\_STRICT\_FOLLOW\_REDIRECTS);

int httpCode = http.GET();

Serial.print("HTTP Status Code: ");

Serial.println(httpCode);

//---------------------------------------------------------------------

//getting response from google sheet

String payload;

if (httpCode > 0) {

payload = http.getString();

Serial.println("Payload: "+payload);

}

//---------------------------------------------------------------------

http.end();

}

count++;

delay(1000);

}

....................................................................................................

//Include required libraries

#include "WiFi.h"

#include <HTTPClient.h>

// WiFi credentials

const char\* ssid = "SKYNET 4G";

// change SSID

const char\* password = "jobitjos"; // change password

// Google script ID and required credentials

String GOOGLE\_SCRIPT\_ID = "AKfycby-snBh-5j0jsiQBWfC

XB1FWy38lks4VHcxLBIGNadeCVcSzUoozHzvazIWv9EcA6a"; // change

Gscript ID

void setup() {

delay(1000);

Serial.begin(115200);

delay(1000);

// connect to WiFi

Serial.println();

Serial.print("Connecting to wifi: ");

Serial.println(ssid);

Serial.flush();

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

}

void loop() {

if (WiFi.status() == WL\_CONNECTED) {

HTTPClient http;

String url = "https://script.google.com/macros/s/" + GOOGLE\_SCRIPT\_ID +

"/exec?read";

Serial.println("Making a request");

http.begin(url.c\_str()); //Specify the URL and certificate

http.setFollowRedirects(HTTPC\_STRICT\_FOLLOW\_REDIRECTS);

int httpCode = http.GET();

String payload;

if (httpCode > 0) { //Check for the returning code

payload = http.getString();

Serial.println(httpCode);

Serial.println(payload);

}

else {

Serial.println("Error on HTTP request");

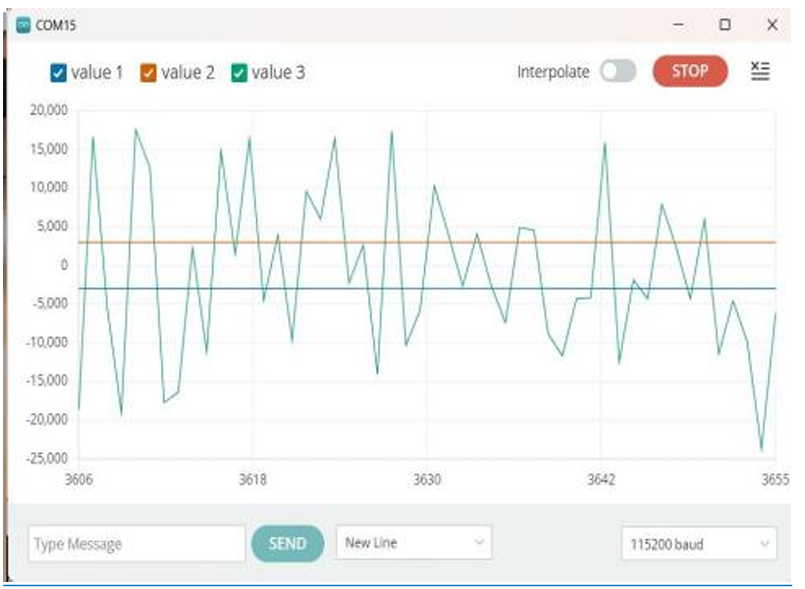
}

http.end();

}

delay(1000);

}

1. **OUTPUT**
2. A screenshot of a graph

   Description automatically generated**OUTPUT**
3. **OUTPUT**

A person working on a circuit board

Description automatically generated

1. **A screenshot of a computer

   Description automatically generatedOUTPUT**