Experiment No. 04

Title: "DIY Smart Sensing Using ESP 32 & IoT | Wokwi Simulation".

SOMAIYA
VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering

Batch: B–1 **Roll No.: 16010422234** **Experiment No.: 04**

Aim: To develop an ESP32-based smart home automation system using Wokwi, enabling remote control of appliances for energy efficiency and convenience.

Resources needed: Internet Connection, ESP32, Sensors (Temperature, Motion, Light, etc.), Actuators (Relays, Motors, Switches), Wokwi Simulator, Power Supply, Wires & Connectors

---

Theory:

Pre Lab/ Prior Concepts:

Familiarity with Arduino and ESP32 microcontrollers, basic programming skills in C/C++, understanding of sensors and actuators, knowledge of circuit connections, and basic networking concepts. Awareness of IoT (Internet of Things) principles, including data transmission and remote control, will enhance the effectiveness of implementing this smart home automation project.

Components of the Wokwi IoT Platform

1. ESP32 / Arduino Simulator – Virtual microcontroller to run and test code.

2. Sensors & Actuators – Virtual temperature, motion, and light sensors, along with relays and motors.

3. Code Editor – Built-in editor for writing and uploading code.

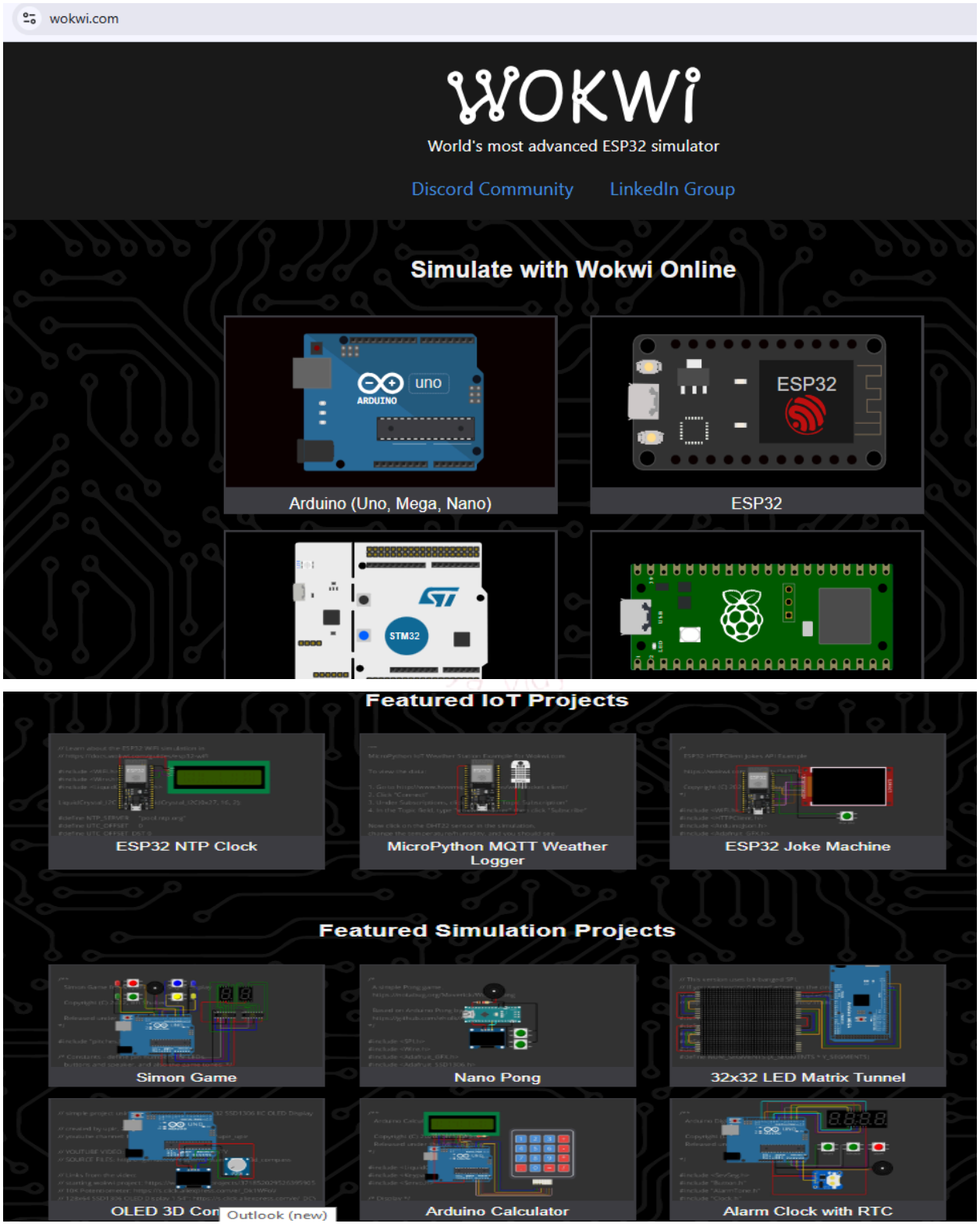
4. Virtual Serial Monitor – Debugging tool for monitoring output and serial communication.

5. Cloud Connectivity – Simulated MQTT and HTTP protocols for IoT communication.

6. Dashboard & UI Elements – Virtual buttons, switches, and displays for user interaction.

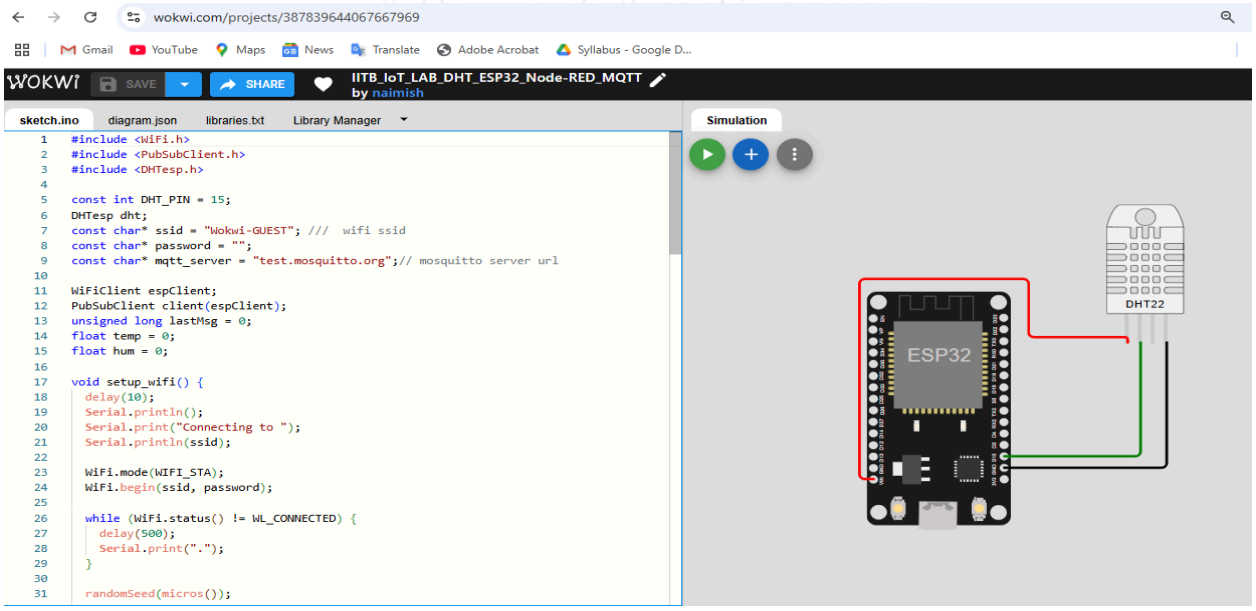
7. Power Supply & Circuit Builder – Interactive wiring tool for circuit design and connection.

Diagram Representation


wokwi.com
WOKWi
World's most advanced ESP32 simulator
Discord Community
LinkedIn Group
Simulate with Wokwi Online
Arduino (Uno, Mega, Nano)
ESP32
STM32
Featured IoT Projects
ESP32 NTP Clock
MicroPython MQTT Weather Logger
ESP32 Joke Machine
Featured Simulation Projects
Simon Game
Nano Pong
32x32 LED Matrix Tunnel
OLED 3D Cor
Outlook (new)
Arduino Calculator
Alarm Clock with RTC

Steps to Perform Experiments Using Wokwi

1. **Open Wokwi**
   - Visit Wokwi in your web browser.
2. **Create a New Project**
   - Click on **"Start a New Project"** or choose a template for Arduino, ESP32, or Raspberry Pi.
3. **Add Components**
   - Use the **"Parts"** panel to add microcontrollers, sensors, actuators, and other electronic components.
4. **Connect the Circuit**
   - Drag and connect the components using virtual wires, ensuring proper pin connections.
5. **Write the Code**
   - Use the **built-in code editor** to write your program in C/C++ (for Arduino/ESP32) or Python (for Raspberry Pi).
6. **Simulate the Experiment**
   - Click the **"Start Simulation"** button to test the circuit and observe output.
7. **Monitor Serial Output**
   - Open the **"Serial Monitor"** to debug and view real-time data.
8. **Modify and Optimize**
   - Make changes to the circuit or code based on test results and rerun the simulation.
9. **Save and Share**
   - Click **"Save Project"**, then copy the project link to share or access it later.
10. **Download or Export (Optional)**
    - Download the code or use it for hardware implementation.



How to Create a ThingSpeak Channel

ThingSpeak is an IoT analytics platform that allows you to collect, analyze, and visualize sensor data in real-time. Follow these steps to create a **ThingSpeak Channel**:

Step 1: Sign Up / Log In

1. Go to **ThingSpeak**.
2. Click on **Sign Up** and create a MathWorks account (or log in if you already have one).

Step 2: Create a New Channel

1. After logging in, click on **"Channels"** from the top menu.
2. Click **"New Channel"**.

Step 3: Configure the Channel

1. **Enter a Channel Name** (e.g., "Smart Home Data").
2. **Fill in the Fields**:
    a. ThingSpeak allows **up to 8 data fields** (e.g., Temperature, Humidity, Motion, etc.).
    b. Enter appropriate **Field Names** for each sensor data you want to log.
3. **Enable Public View (Optional)**: If you want others to see your data, check **"Make Public"**.

Step 4: Save the Channel

1. Click **"Save Channel"** to create it.
2. Your new **ThingSpeak Channel** is now ready!

Step 5: Get Your API Keys

1. Go to the **API Keys** tab.
2. Note down the **Write API Key** (for sending data) and **Read API Key** (for retrieving data).

Step 6: Send Data to ThingSpeak

To send data from **Arduino/ESP32**, use an HTTP request or MQTT with the API key. Example:
https://api.thingspeak.com/update?api_key=YOUR_WRITE_API_KEY&field1=25.5
Replace YOUR_WRITE_API_KEY with your actual **Write API Key** and 25.5 with your sensor data.

Step 7: View and Analyze Data

1. Click on **"Private View"** or **"Public View"** to visualize real-time graphs.
2. Use the **"Export Data"** option to download data for further analysis.

Procedure to Connect ESP32 with ThingSpeak using Wokwi
This guide will help you **send sensor data from ESP32 to ThingSpeak** using MQTT.

Step 1: Set Up ThingSpeak Channel

1. **Log in** to ThingSpeak.
2. Click **"Channels" > "New Channel"**.
3. **Enable Field 1** (for temperature) and Field 2 (for humidity).
4. Click **"Save Channel"**.
5. Go to **"API Keys"** and note down the **Write API Key**.

Step 2: Set Up Wokwi ESP32 Simulation

1. Open Wokwi and start a new ESP32 project.
2. Add **ESP32** and **DHT22 Sensor** from the **Library Manager**.
3. Connect **DHT22** to ESP32 as shown in your image:
    - **VCC → 3.3V**
    - **GND → GND**
    - **Data → GPIO 15 (or any available pin)**

Step 3: Install Required Libraries
In your **Arduino IDE (or Wokwi's Code Editor)**, install the following libraries:

- WiFi.h (For ESP32 WiFi Connection)
- DHTesp.h (For DHT22 Sensor)
- PubSubClient.h (For MQTT Communication)

Step 4: Write Code for ESP32
Modify the following code in **Wokwi's code editor**:

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHTesp.h>
const int DHT_PIN = 15;
DHTesp dht;
// WiFi credentials
const char* ssid = "Wokwi-GUEST";
const char* password = "";
// ThingSpeak MQTT details
const char* mqtt_server = "mqtt.thingspeak.com";
const char* mqtt_user = "your_thingspeak_username";
const char* mqtt_password = "your_thingspeak_mqtt_password";
const char* topic = "channels/YOUR_CHANNEL_ID/publish/YOUR_WRITE_API_KEY";
WiFiClient espClient;
```

```
PubSubClient client(espClient);
void setup() {
  Serial.begin(115200);
  dht.setup(DHT_PIN, DHTesp::DHT22);
  // Connect to WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConnected to WiFi");
  // Connect to MQTT Broker (ThingSpeak)
  client.setServer(mqtt_server, 1883);
  while (!client.connected()) {
    Serial.print("Connecting to MQTT...");
    if (client.connect("ESP32Client", mqtt_user, mqtt_password)) {
      Serial.println("Connected!");
    } else {
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      delay(2000);
    }
  }
}
void loop() {
  float temp = dht.getTemperature();
  float hum = dht.getHumidity();
  if (!client.connected()) {
    setup(); // Reconnect if disconnected
  }
  char payload[50];
  sprintf(payload, "field1=%.2f&field2=%.2f", temp, hum);

  if (client.publish(topic, payload)) {
    Serial.println("Data sent to ThingSpeak!");
  } else {
    Serial.println("Failed to send data.");
  }
  delay(15000); // Send data every 15 seconds
}
```

SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering

Step 5: Run the Simulation

1. Click the **Run (Play) Button** on Wokwi.
2. Check the **Serial Monitor** for WiFi and MQTT connection status.
3. If successful, the data will be **sent to ThingSpeak** every 15 seconds.

Step 6: View Data on ThingSpeak

1. Go to your **ThingSpeak Channel**.
2. Click on **"Private View" / "Public View"**.
3. Your **temperature and humidity** data will appear as a live graph.

Troubleshooting

- Ensure you have entered the **correct WiFi SSID and Password**.
- Check your **ThingSpeak API Key and Channel ID**.
- Verify the **MQTT Connection** in Serial Monitor.
- Use **DHT22 sensor** correctly wired to ESP32.
- Link for this experiment : https://wokwi.com/projects/387822122274688001

Activity:

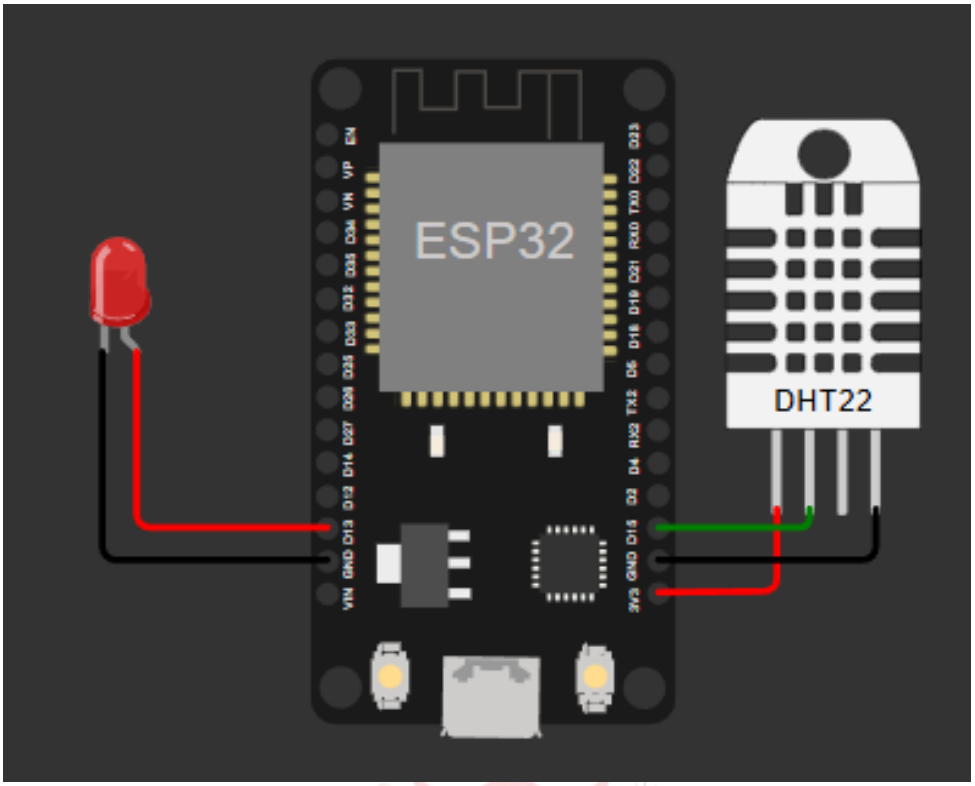
Blink an LED Using ESP32/Arduino
Read Digital Input from a Push Button
Measure Distance Using Ultrasonic Sensor (HC-SR04)
Generate PWM to Control LED Brightness
Control a Relay to Operate a Bulb/Fan

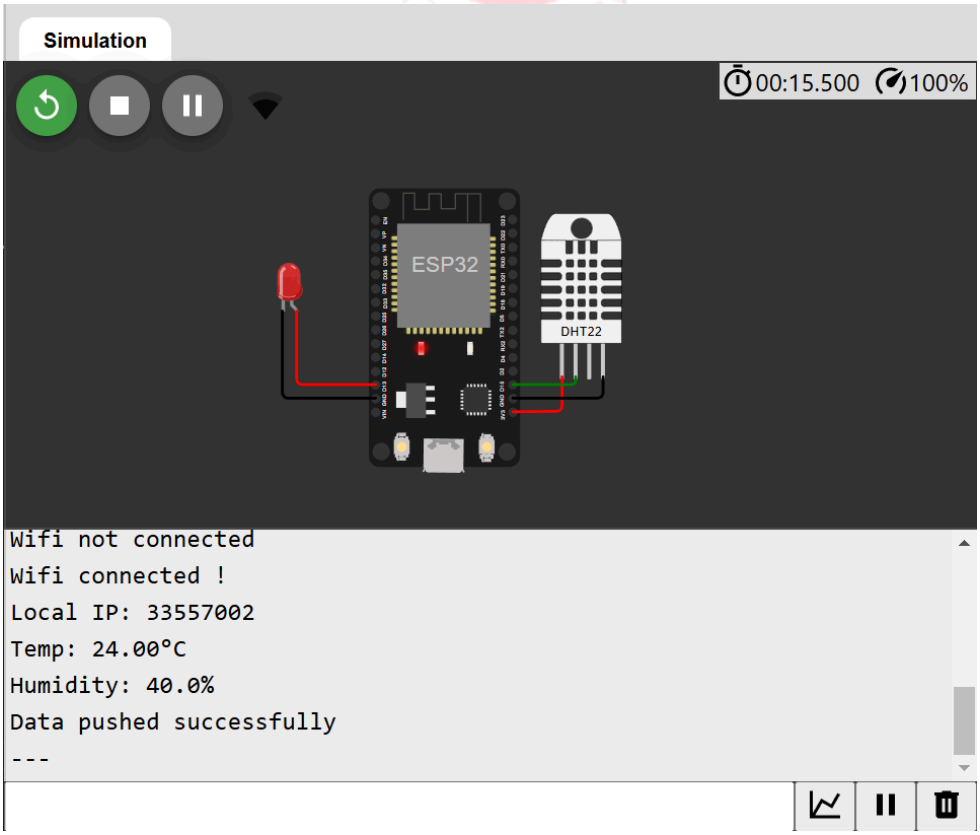
Results: (Program printout with output / Document printout as per the format)

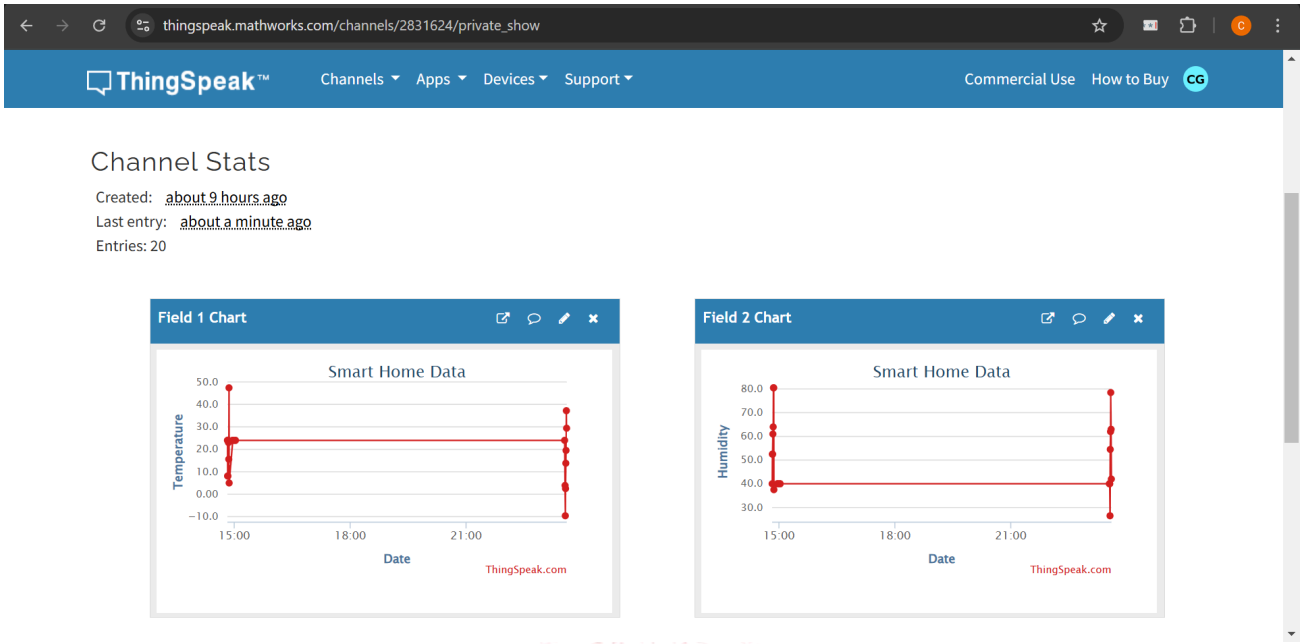
```
#include <WiFi.h>
#include "DHTesp.h"
#include "ThingSpeak.h"

// Pin configuration
const int DHT_PIN = 15;   // Pin connected to the DHT sensor
const int LED_PIN = 13;   // Pin connected to the LED

// WiFi credentials
const char* WIFI_NAME = "Wokwi-GUEST";
const char* WIFI_PASSWORD = "";
```

```
// ThingSpeak channel details
const int myChannelNumber = 2831624;
const char* myApiKey = "48BJAUX76BL6U4CA";
const char* server = "api.thingspeak.com";

// DHT sensor and WiFi client
DHTesp dhtSensor;
WiFiClient client;

void setup() {
  Serial.begin(115200);

  // Initialize DHT sensor
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);

  // Set up LED pin
  pinMode(LED_PIN, OUTPUT);

  // Connect to WiFi
  WiFi.begin(WIFI_NAME, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Wifi not connected");
  }
  Serial.println("Wifi connected !");
  Serial.println("Local IP: " + String(WiFi.localIP()));

  // Set WiFi mode to STA (station) mode
  WiFi.mode(WIFI_STA);

  // Initialize ThingSpeak client
  ThingSpeak.begin(client);
}

void loop() {
  // Read temperature and humidity from DHT sensor
  TempAndHumidity data = dhtSensor.getTempAndHumidity();

  // Update ThingSpeak fields
  ThingSpeak.setField(1, data.temperature);
```

```
  ThingSpeak.setField(2, data.humidity);

  // Check temperature and humidity conditions
  if (data.temperature > 35 || data.temperature < 12 || data.humidity > 70
|| data.humidity < 40) {
    digitalWrite(LED_PIN, HIGH);  // Turn on LED if conditions are met
  } else {
    digitalWrite(LED_PIN, LOW);   // Turn off LED if conditions are not
met
  }

  // Send data to ThingSpeak
  int x = ThingSpeak.writeFields(myChannelNumber, myApiKey);

  // Print temperature and humidity to serial monitor
  Serial.println("Temp: " + String(data.temperature, 2) + "°C");
  Serial.println("Humidity: " + String(data.humidity, 1) + "%");

  // Check if data was successfully pushed to ThingSpeak
  if (x == 200) {
    Serial.println("Data pushed successfully");
  } else {
    Serial.println("Push error: " + String(x));
  }
  Serial.println("---");

  // Delay for 10 seconds before the next iteration
  delay(10000);
}
```


ESP32
DHT22


Simulation
00:15.500
100%
ESP32
DHT22
Wifi not connected
Wifi connected !
Local IP: 33557002
Temp: 24.00°C
Humidity: 40.0%
Data pushed successfully



Questions:

1. List out and explain other platforms which can be used as a dashboard for IoT applications?

Ans: Here are some other platforms that can be used as dashboards for IoT applications:

1. Blynk:

Description: Blynk is a mobile-based IoT platform that provides a drag-and-drop interface to design dashboards for controlling hardware over the internet. It supports a wide range of IoT devices, including Arduino, ESP32, and Raspberry Pi.

Features: Real-time data monitoring, remote control of IoT devices, and customizable user interfaces.

Website: https://blynk.io/

2. Adafruit IO:

Description: Adafruit IO is a cloud service for the Internet of Things, which helps users connect sensors and devices to the cloud. It provides real-time data visualization tools and enables easy integration with a variety of IoT sensors.

Features: Graphs, dashboards, real-time monitoring, and remote control capabilities.

Website: https://io.adafruit.com/

3. ThingsBoard:

Description: ThingsBoard is an open-source IoT platform that allows users to collect and analyze data from IoT devices. It provides advanced dashboard design capabilities with drag-and-drop widgets and supports real-time monitoring.
Features: Device management, real-time data visualization, advanced data analytics, and MQTT support.
Website: https://thingsboard.io/

4. Kaa IoT Platform:
Description: Kaa is an open-source IoT platform that provides a full suite of tools for device management, real-time analytics, and application development. It allows users to create custom dashboards for their IoT projects.
Features: Device management, data visualization, event-driven architecture, and cloud integration.
Website: https://www.kaaproject.org/

5. Node-RED:
Description: Node-RED is a flow-based development tool for visual programming, used to wire together devices, APIs, and online services. It is popular for creating dashboards and connecting IoT devices to various services.
Features: Flow-based programming, real-time data processing, and integration with multiple IoT devices.
Website: https://nodered.org/

6. Ubidots:
Description: Ubidots is a cloud-based IoT platform designed to help you build dashboards to monitor and control your IoT devices. It's a versatile platform with easy integration for sensors and controllers.
Features: Real-time data visualization, alerting, API integrations, and device management.
Website: https://ubidots.com/

7. Home Assistant:
Description: Home Assistant is an open-source platform that focuses on privacy and local control of IoT devices. It can be used to create dashboards for home automation systems.
Features: Customizable dashboards, automation, device control, and local processing.
Website: https://www.home-assistant.io/

These platforms provide the tools and features necessary for monitoring, controlling, and visualizing IoT data in real-time, allowing developers to create customized dashboards based on their project requirements.

Outcomes: CO2 – Comprehend IoT architecture, enabling technologies and protocols

Conclusion:

The ESP32-based smart home automation system, utilizing platforms like Wokwi and ThingSpeak, offers a powerful solution for integrating sensors, actuators, and IoT communication protocols. This experiment demonstrates how an ESP32 can be leveraged for real-time data transmission and monitoring, making homes smarter and more efficient. The integration of ThingSpeak allows for remote data visualization, while the Wokwi simulator simplifies the hardware and software testing process. Additionally, exploring various IoT dashboard platforms, such as Blynk, Adafruit IO, and ThingsBoard, opens up numerous possibilities for creating intuitive user interfaces to interact with IoT devices. Overall, this experiment enhances the understanding of smart home technology and IoT principles, providing practical insights for developing and deploying IoT-based solutions.

Grade: AA / AB / BB / BC / CC / CD / DD

Signature of faculty in-charge with date

Somaiya Vidyavihar
SOMAIYA
VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering

References:

References for Wokwi Experiments

1. **Wokwi Official Website** – https://wokwi.com/
2. **Arduino Official Documentation** – https://www.arduino.cc/reference/en/
3. **ESP32 Documentation** – https://docs.espressif.com/projects/esp-idf/en/latest/esp32/
4. **ThingSpeak IoT Cloud** – https://thingspeak.com/
5. **Blynk IoT Platform** – https://blynk.io/
6. **HC-SR04 Ultrasonic Sensor Guide** – https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/
7. **PWM and Servo Control Basics** – https://www.arduino.cc/en/Tutorial/PWM
8. **Relay Module Interfacing with Arduino** – https://circuitdigest.com/microcontroller-projects/arduino-relay-module-interfacing
9. **Button and Switch Basics in Arduino** – https://www.arduino.cc/en/tutorial/button
10. **Serial Communication Basics** –
11. https://www.arduino.cc/reference/en/language/functions/communication/serial/

Books:

1. Jan Holler, Vlasios Tsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle, “From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence”, 1st Edition, Academic Press, 2014.
2. Vijay Madisetti and Arshdeep Bahga, “Internet of Things (A Hands-on-Approach)”, 1stEdition, VPT, 2014.
3. Dr. Ovidiu Vermesan, Dr. Peter Friess, “Internet of Things - From Research and Innovation to Market Deployment”, River Publisher, 2014

SOMAIYA
VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering