Experiment No._05

Title: "Real-Time Smart Sensing Dashboard with ESP32 & IoT Using Node-RED"

Batch: B–1 **Roll No.: 16010422234** **Experiment No.: 05**

Aim: The aim is to create a real-time smart sensing system using ESP32, IoT, and Node-RED, simulating and monitoring sensor data via an interactive dashboard.

Resources needed: Internet Connection, ESP32, Temperature Sensor, Motion Sensor, Light Sensor, Relays, Motors, Switches, Wokwi Simulator, Power Supply, Wires & Connectors

Theory:

Pre Lab/ Prior Concepts:

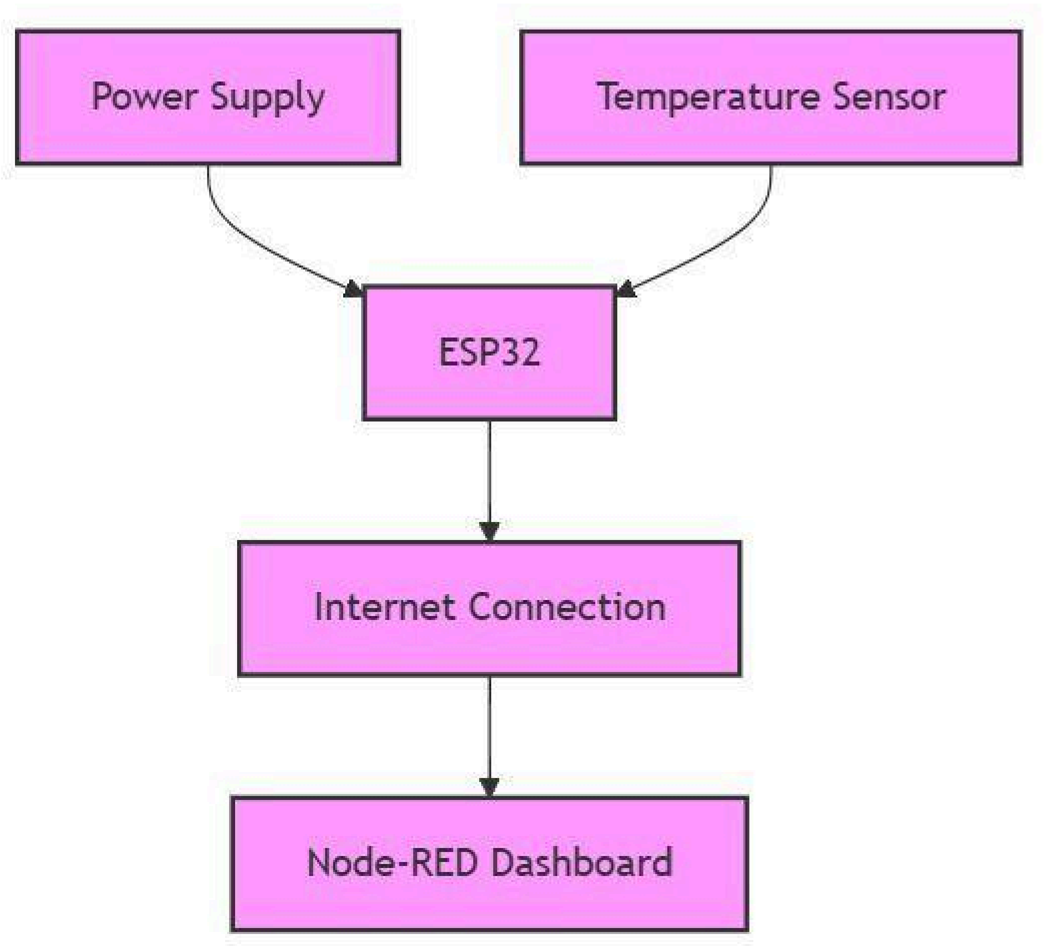
Before starting the experiment, it is important to have a basic understanding of the **ESP32** microcontroller, its features, and how it connects to the internet for IoT applications. Familiarity with **IoT** concepts, including data collection, processing, and remote monitoring, is essential. Knowledge of **Node-RED** will help in building flow-based applications and creating a dashboard to visualize sensor data in real time. Understanding how to interface common **sensors** such as temperature, motion, and light sensors with microcontrollers is crucial, as well as the operation of **actuators** like relays and motors for controlling external devices. Prior experience with the **Wokwi Simulator** will be helpful for simulating circuits before physical implementation. Additionally, knowledge of **Wi-Fi** connectivity and **real-time data communication** between the microcontroller and the dashboard is key for successful project execution. Understanding **power management** for the ESP32 and other components is also important for ensuring stable operation throughout the experiment.

Components of the Platform:

1. ESP32 Microcontroller: The central unit for processing and controlling the system, with built-in Wi-Fi and Bluetooth capabilities for IoT communication.
2. Sensors: Devices like temperature, motion, and light sensors that collect data from the environment and send it to the ESP32 for processing.
3. Actuators: Relays, motors, and switches that can be controlled by the ESP32 based on sensor data or commands from the dashboard.
4. Node-RED: A flow-based development tool for wiring together hardware devices, APIs, and online services to create a dashboard for real-time monitoring and control of the system.
5. Wokwi Simulator: An online platform for simulating the hardware components and their interaction, allowing testing and troubleshooting without physical components.
6. Internet Connection: Provides the necessary network access for communication between the ESP32 and the dashboard, enabling real-time data transmission.

7. Power Supply: Supplies the necessary power to the ESP32 and connected components for their operation.
8. Wires & Connectors: Physical components used to wire and connect the sensors, actuators, and the ESP32 together, ensuring proper communication and power distribution.

Diagram Representation



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

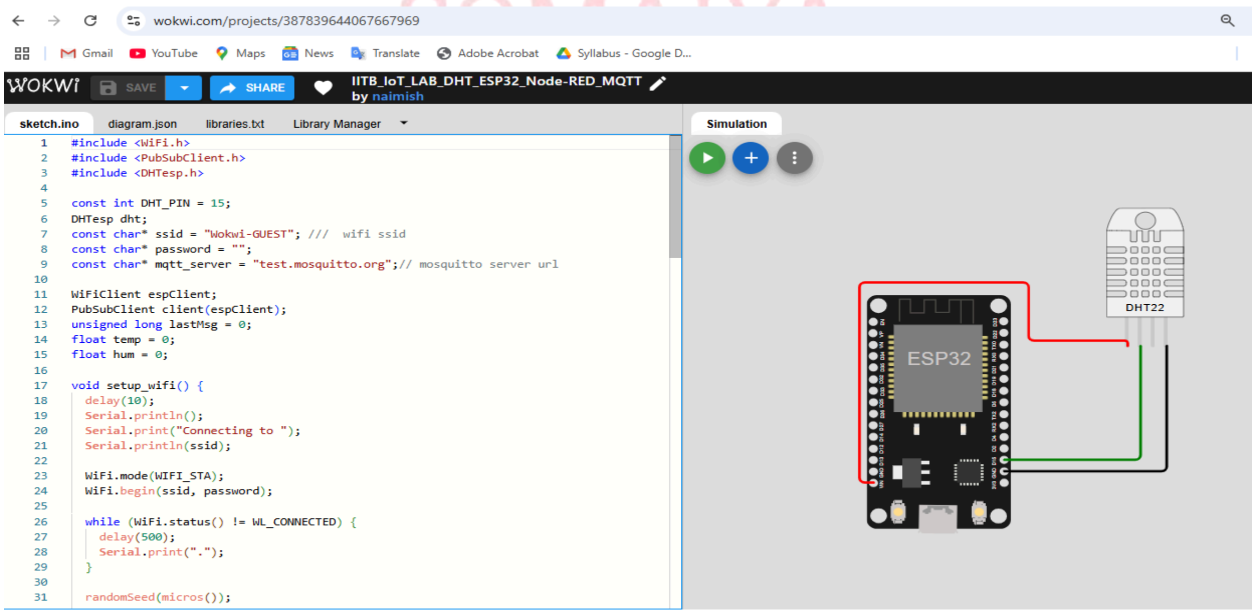
- Open the **"Serial Monitor"** to debug and view real-time data.

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.



```
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHTesp.h>

const int DHT_PIN = 15;
DHTesp dht;
const char* ssid = "Wokwi-GUEST"; ///  wifi ssid
const char* password = "";
const char* mqtt_server = "test.mosquitto.org";// mosquitto server url

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
float temp = 0;
float hum = 0;

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());
```

1) Install Node.js (Windows)

- **Visit the Node.js website**: Node.js Downloads
- Download the **LTS version** (Long-Term Support).
- Follow the installation instructions. Once installed, it should add Node.js and npm (Node package manager) to your system PATH automatically.

2) Install Node-RED on Windows

1. **Open Command Prompt** (as Administrator if possible):
   - Press Win + X and select **Command Prompt (Admin)** or **Windows PowerShell (Admin)**.
2. **Install Node-RED Globally**:
   - In the Command Prompt, run the following command:
   - npm install -g --unsafe-perm node-red
3. This will install Node-RED globally on your system.

3) Run Node-RED

1. **Start Node-RED**:
   a. After installation, simply type the following command to run Node-RED: node-red
2. **Access the Node-RED Editor**:
   a. Open your web browser and visit: http://localhost:1880
3. This will open the Node-RED flow editor.

4) Installing MQTT Nodes for Node-RED

1. Install MQTT Nodes:
   a. In the Node-RED editor, click the **menu** in the top-right corner (three horizontal lines).
   b. Go to **Manage palette** > **Install** tab.
   c. Type node-red-node-mqtt in the search bar and click **Install** next to the result.

This installs the MQTT input and output nodes for communication with MQTT brokers.

5) Start Building Flows in Node-RED

Now that you have Node-RED installed, you can create flows, connect MQTT nodes, and start working with your devices.

6) Create a New Flow

1. Open the Node-RED editor at http://localhost:1880.
2. Drag and drop the following nodes:
   a. **MQTT Input Node** (for subscribing to topics like /IITB_IoT_LAB/temp and /IITB_IoT_LAB/hum).
   b. **Debug Node** (to monitor incoming data).
   c. **Inject Node** (optional, for testing or triggering data).

7) Configure the MQTT Input Nodes

1. Double-click the **MQTT Input Node** for temperature.
    a. Configure the **Broker URL** (e.g., test.mosquitto.org).
    b. Set the **Topic** to /IITB_IoT_LAB/temp (or any topic you want to subscribe to).
2. Repeat the above for the **humidity topic**, setting the **Topic** to /IITB_IoT_LAB/hum.

8) Add Debug Nodes

1. Drag a **Debug Node** to the workspace.
2. Connect the **MQTT Input Node** for temperature to the Debug Node.
3. Connect the **MQTT Input Node** for humidity to a separate Debug Node.
4. Configure the **Debug Nodes** to show the complete message (msg.payload).

9) Deploy the Flow

1. Click the **Deploy** button in the top-right corner to activate your flow.

10) Monitor the Data in the Debug Window

1. Open the **Debug** tab in the right sidebar.
2. You should now see incoming temperature and humidity data from the ESP32 in real-time.

11) Optional: Add UI Dashboard for Visualization

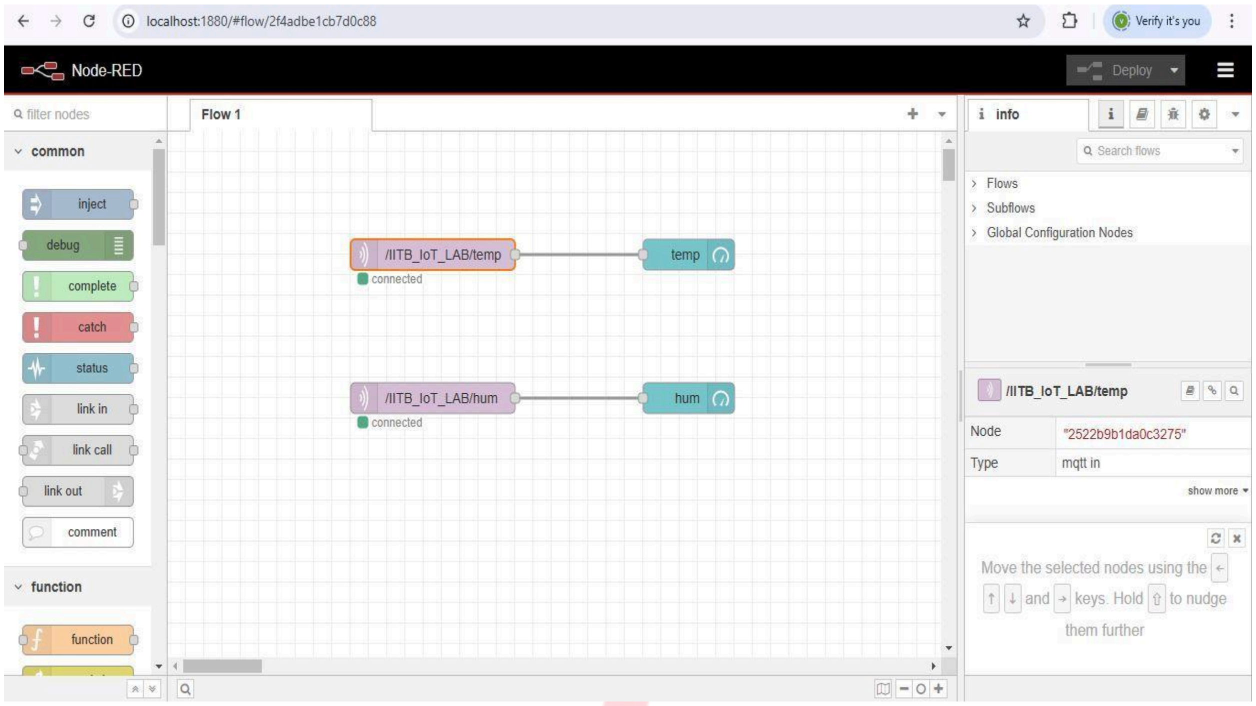
1. Drag **UI Text** or **UI Gauge** nodes from the sidebar to display temperature and humidity values.
2. Connect the **MQTT Input Nodes** to the **UI Text/Gauge Nodes**.
3. Click **Deploy** again.
4. Visit http://localhost:1880/ui to view the dashboard.

12) Optional: Store Data or Trigger Actions

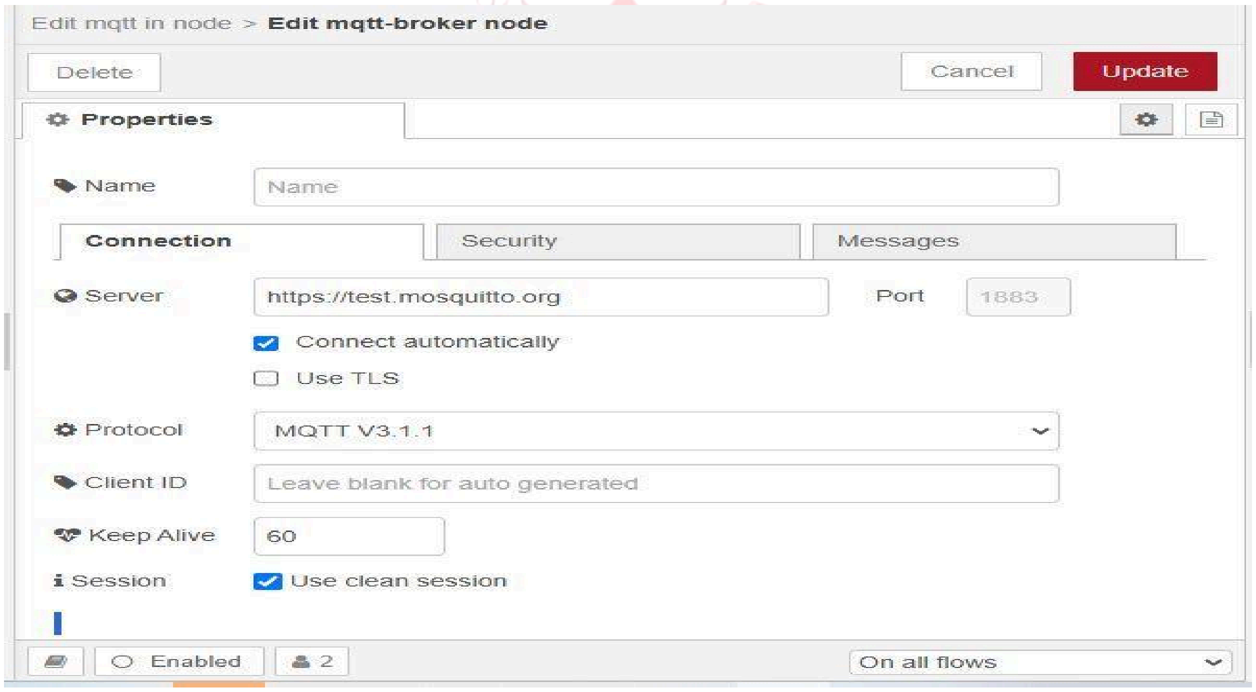
1. Add **Database nodes** (e.g., **SQLite**, **InfluxDB**) to store data.
2. Use **Function Nodes** to trigger actions (e.g., turn on a fan if the temperature exceeds a threshold).

13) Test and Debug

1. Use the **Inject Node** to simulate sending MQTT messages to your topics.
2. Verify that messages are received correctly in the **Debug Panel**.



Node	"2522b9b1da0c3275"
Type	mqtt in



Name: Name

Server: https://test.mosquitto.org Port: 1883

☑ Connect automatically

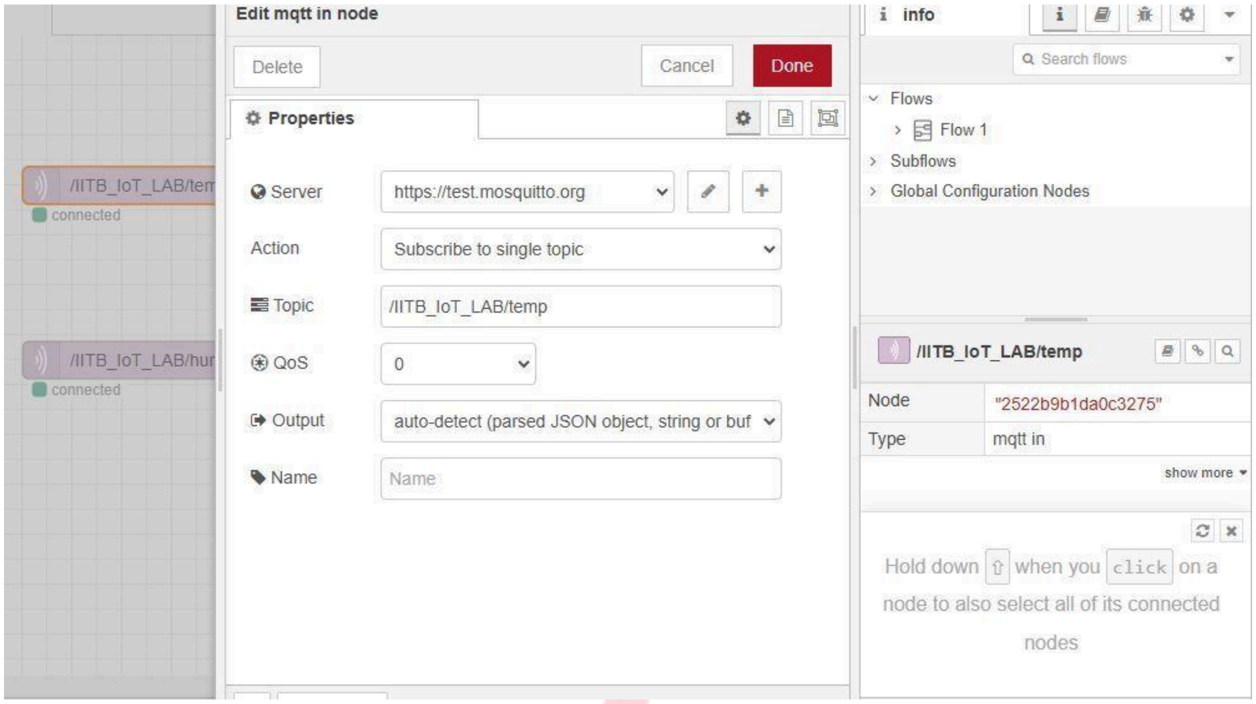
☐ Use TLS

Protocol: MQTT V3.1.1

Client ID: Leave blank for auto generated

Keep Alive: 60

Session: ☑ Use clean session

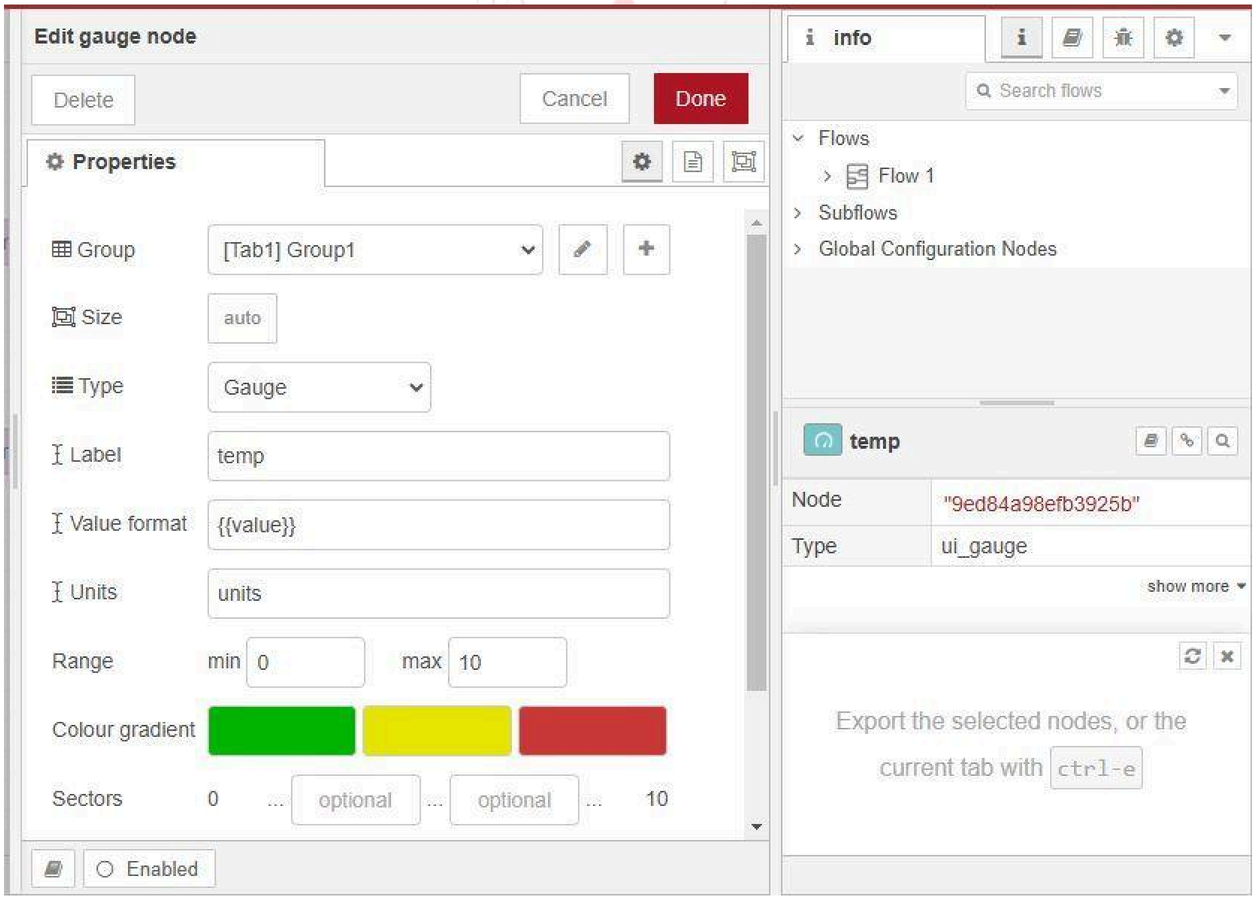

Edit mqtt in node

Server: https://test.mosquitto.org
Action: Subscribe to single topic
Topic: /IITB_IoT_LAB/temp
QoS: 0
Output: auto-detect (parsed JSON object, string or buf
Name: Name

/IITB_IoT_LAB/temp

Node	"2522b9b1da0c3275"
Type	mqtt in

Hold down ⇧ when you click on a node to also select all of its connected nodes

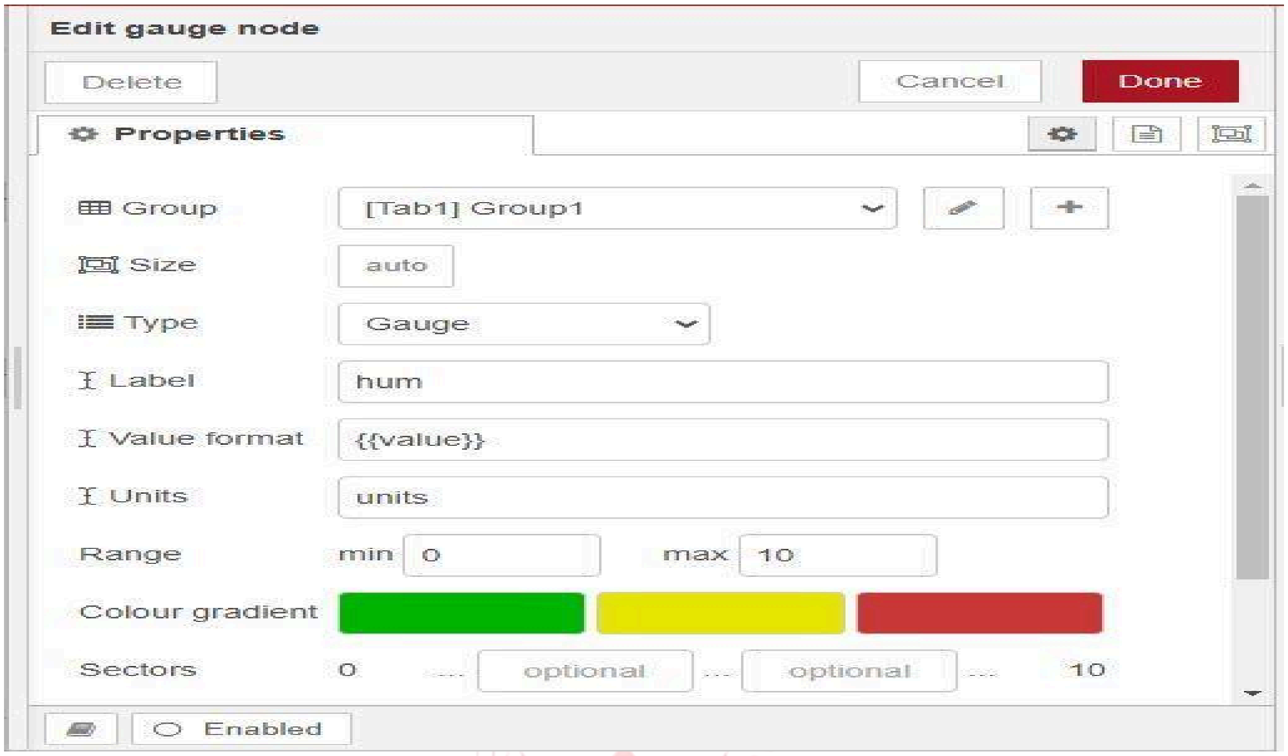

Edit gauge node

Group: [Tab1] Group1
Size: auto
Type: Gauge
Label: temp
Value format: {{value}}
Units: units
Range: min 0 max 10
Colour gradient: 🟩 🟨 🟥
Sectors: 0 ... optional ... optional ... 10

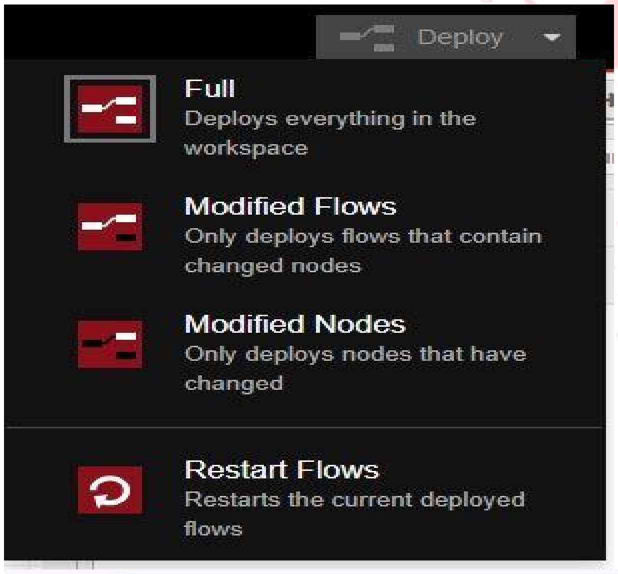
temp

Node	"9ed84a98efb3925b"
Type	ui_gauge

Export the selected nodes, or the current tab with ctrl-e



Group	[Tab1] Group1
Size	auto
Type	Gauge
Label	hum
Value format	{{value}}
Units	units
Range	min 0 max 10
Colour gradient	🟩 🟨 🟥
Sectors	0 ... optional ... optional ... 10

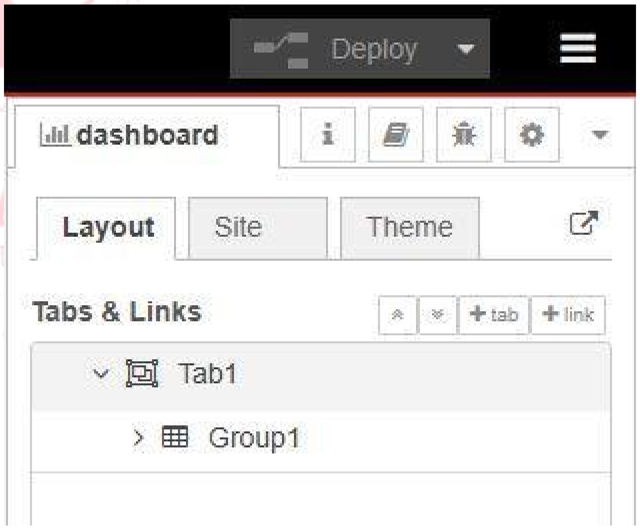


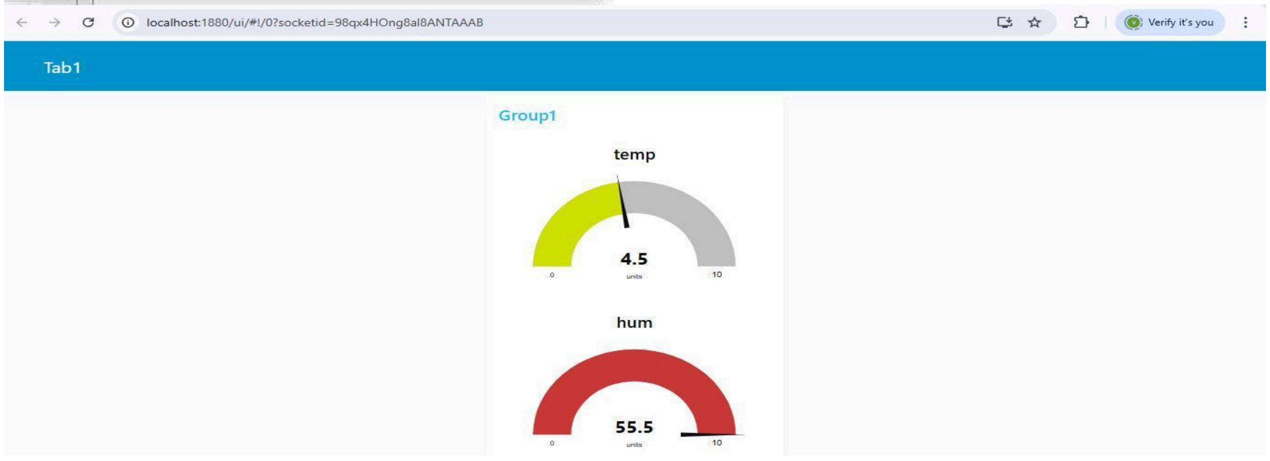
Full
Deploys everything in the workspace

Modified Flows
Only deploys flows that contain changed nodes

Modified Nodes
Only deploys nodes that have changed

Restart Flows
Restarts the current deployed flows

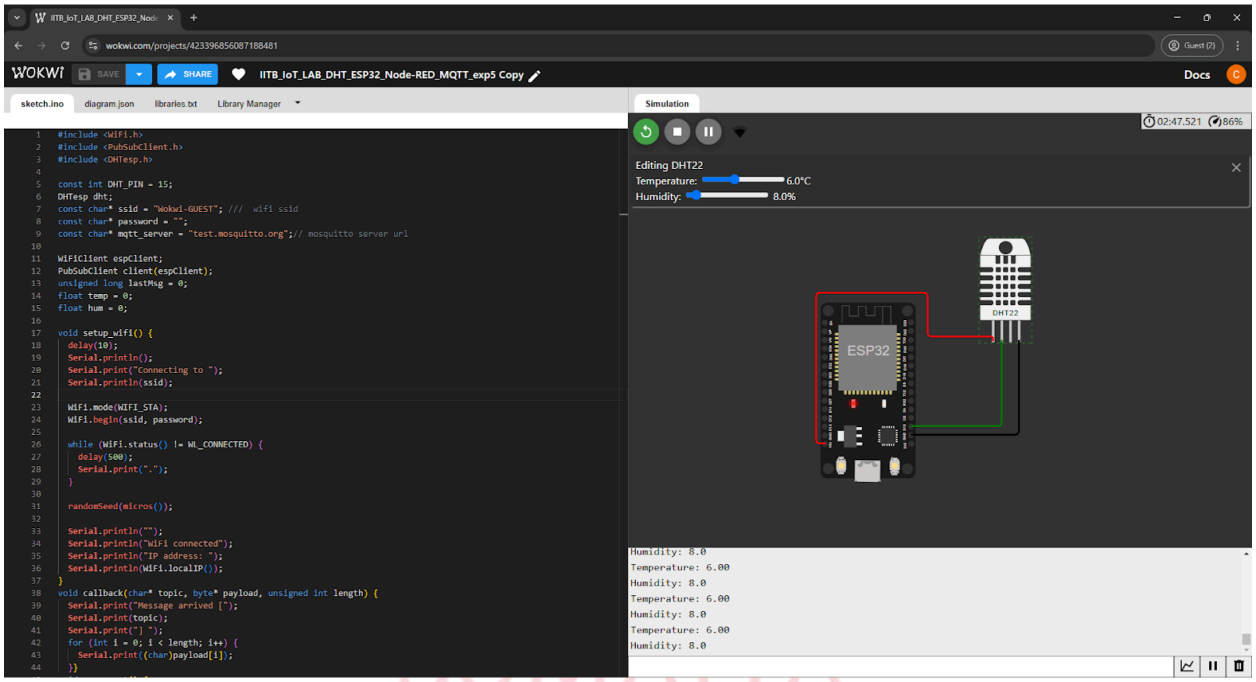




Activity:

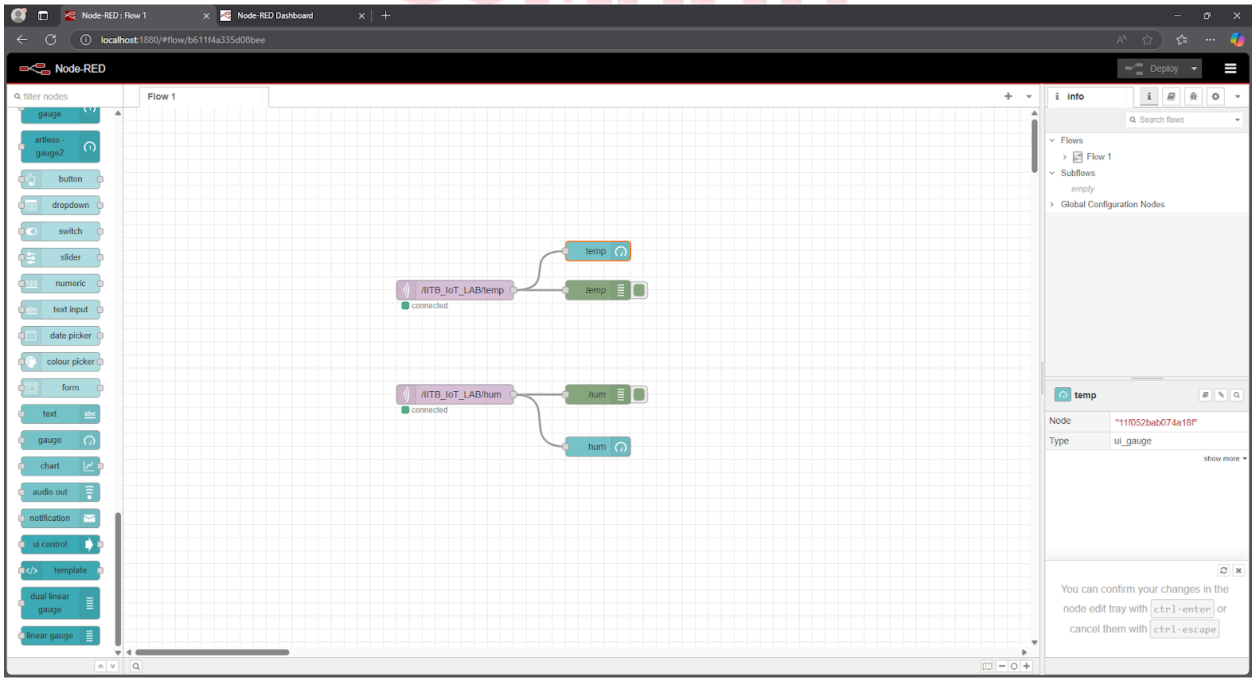
Control LED via MQTT Using Wokwi and Node-RED
Control a Relay to Operate a Bulb or Fan Using MQTT and Wokwi
Monitor Soil Moisture Using a Soil Moisture Sensor and Display Data in Node-RED
Measure Distance Using Ultrasonic Sensor (HC-SR04) and Send Data to Node-RED via MQTT
Control RGB LED Color Using MQTT and Node-RED in Wokwi

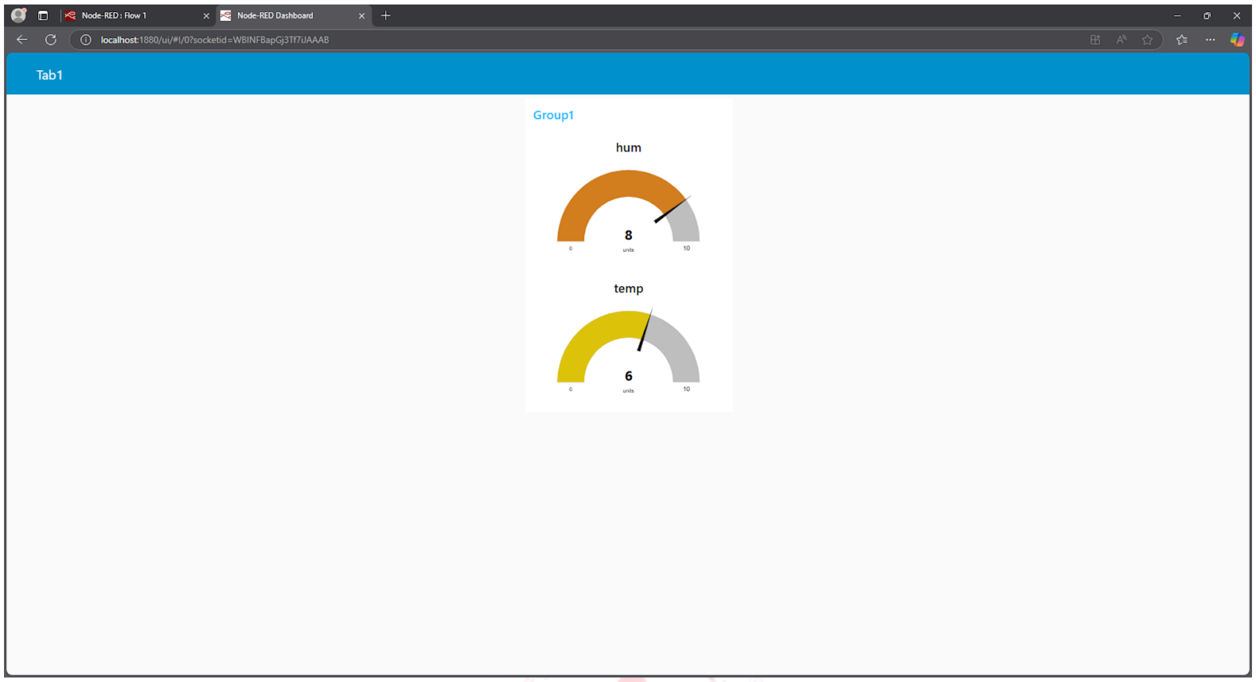
Reference : https://wokwi.com/projects/387839644067667969



```
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHTesp.h>

const int DHT_PIN = 15;
DHTesp dht;
const char* ssid = "Wokwi-GUEST"; ///  wifi ssid
const char* password = "";
const char* mqtt_server = "test.mosquitto.org";// mosquitto server url

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
float temp = 0;
float hum = 0;

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }}
```

```
Humidity: 8.0
Temperature: 6.00
Humidity: 8.0
Temperature: 6.00
Humidity: 8.0
Temperature: 6.00
Humidity: 8.0
```





Questions:

What are the key steps to visualize and control IoT sensor data (e.g., temperature, humidity, distance) using MQTT and Node-RED, and how can you integrate real-time device control (like LED and relay operation) through an IoT dashboard?

Ans: To visualize and control IoT sensor data (e.g., temperature, humidity, distance) using MQTT and Node-RED, the key steps are:

1. Set up MQTT Communication: Configure the ESP32 to collect data from sensors and send it via MQTT to an MQTT broker (e.g., Mosquitto). Topics like /IITB_IoT_LAB/temp for temperature and /IITB_IoT_LAB/hum for humidity are commonly used.

2. Create Flows in Node-RED: In the Node-RED editor, add MQTT Input Nodes to subscribe to the relevant topics for sensor data. These nodes receive real-time data and can trigger further actions.

3. Visualize Data on a Dashboard: Use UI Nodes like UI Text, UI Gauge, or UI Chart to display incoming sensor data on a dashboard accessible at http://localhost:1880/ui.

4. Control Devices via MQTT: Use MQTT Output Nodes in Node-RED to send control signals (e.g., turning on/off a relay or LED) based on sensor data or user actions from the dashboard.

5. Real-Time Device Control: By integrating devices like relays and LEDs into Node-RED flows, real-time actions can be triggered. For example, turning on a fan or activating a motor based on the temperature or distance measured by the sensors.

By combining MQTT, Node-RED, and IoT sensors, you can create an interactive, real-time monitoring and control system with minimal hardware and flexible software integration.

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

---

Conclusion:
In this experiment, we developed a real-time smart sensing system using the ESP32, Node-RED, and MQTT to collect and visualize sensor data (temperature, humidity, distance) on an interactive dashboard. The system also enabled device control, such as turning on/off LEDs and relays, based on sensor data or user input. This experiment reinforced key IoT concepts, including sensor integration, real-time communication, and device control using Node-RED.

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

Signature of faculty in-charge with date

References:
For your IoT experiment, which involves controlling devices like LEDs, relays, sensors (such as DHT22 and HC-SR04), and integrating MQTT communication with Node-RED, here is a list of relevant references tailored to the components, protocols, and platforms you used:
References for This IoT Experiment:

1. Wokwi Official Website
    a. Wokwi
    b. Description: Use Wokwi to simulate IoT projects, including ESP32 and Arduino with sensors (e.g., DHT22, HC-SR04), LEDs, relays, and more.
2. Arduino Official Documentation
    a. Arduino Documentation
    b. Description: The official reference for Arduino programming and hardware interfacing, covering libraries, functions, and tutorials. Ideal for understanding how to control sensors and devices in your experiment.
3. ESP32 Documentation
    a. ESP32 Documentation
    b. Description: Provides detailed documentation on using the ESP32 microcontroller, including setting up the environment and utilizing MQTT communication and sensor interfacing.
4. ThingSpeak IoT Cloud
    a. ThingSpeak
    b. Description: A cloud platform for collecting and analyzing IoT sensor data. Useful for storing data from your DHT22 sensor or controlling devices remotely via the cloud.

5. Blynk IoT Platform
    a. Blynk
    b. Description: Use Blynk to create mobile apps for controlling devices like LEDs and relays. Blynk supports MQTT for communication between the app and IoT devices.
6. HC-SR04 Ultrasonic Sensor Guide
    a. HC-SR04 Tutorial
    b. Description: A tutorial on how to interface the HC-SR04 ultrasonic sensor with Arduino or ESP32 to measure distance. Relevant if your experiment involves measuring distance with ultrasonic sensors.
7. PWM and Servo Control Basics
    a. PWM and Servo Basics
    b. Description: Learn how to use PWM to control the brightness of LEDs or the position of servo motors, which could be part of your experiment if you're working with motorized devices.
8. Relay Module Interfacing with Arduino
    a. Relay Module Tutorial
    b. Description: Guide on using relay modules with Arduino or ESP32 to control high-voltage devices like bulbs or fans. Essential for controlling relays in your experiment.
9. Button and Switch Basics in Arduino
    a. Button Tutorial
    b. Description: Learn to interface buttons and switches with your microcontroller for controlling actions like turning LEDs on/off or toggling relays. Could be used in your experiment for manual control.
10. Serial Communication Basics
    a. Serial Communication Reference
    b. Description: Learn about serial communication to interface with your microcontroller via the Serial Monitor. Useful for debugging, logging, and monitoring data from your sensors (e.g., DHT22 readings).

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