IoT & Automation Lab Record

**Lab-1**

**Lab Exercise: Blinking the In-Built LED**

void setup() {

pinMode(LED\_BUILTIN, OUTPUT);

}

void loop() {

digitalWrite(LED\_BUILTIN, HIGH);

delay(1000);

digitalWrite(LED\_BUILTIN, LOW);

delay(1000);

}

****

**[Wokwi Lab-1 @ project](https://wokwi.com/projects/410077373723539457)**

**Lab-2**

**Lab Exercise: Blinking an External Red LED**

#define vk 12

void setup() {

  pinMode(vk, OUTPUT);

}

void loop() {

  digitalWrite(vk, HIGH);

  delay(500);

  digitalWrite(light, LOW);

  delay(500);

  }

****

**[Wokwi Lab-2 @ Sketch](https://wokwi.com/projects/410077546199647233)**

**Lab-3**

**Lab Exercise: Reading and Displaying Humidity and Temperature Data**

#include <DHT.h>

#define vk 7

#define DHTTYPE DHT22

DHT dht(vk, DHTTYPE);

float humid, temp;

void setup() {

  Serial.begin(9600);

  dht.begin();

}

void loop() {

  delay(200);

  humid = dht.readHumidity();

  temp = dht.readTemperature();

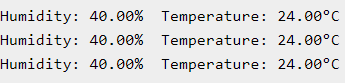
  Serial.print("Humidity: ");

  Serial.print(humid);

  Serial.print(" %  Temperature: ");

  Serial.print(temp);

  Serial.println("°C");

  delay(1000);

}

**[wokwi Lab-3 @ Sketch](https://wokwi.com/projects/410078009353666561)**

**Lab#4**

**Configuring MQTT Service in my Machine**

* **In SystemOS [ Windows11 ]:**

- Installed Mosquitto as a Service from Official Eclipse Page [ <https://mosquitto.org/download/> ].

- *This allows the MQTT Broker to run automatically in the background*.

- **Added** mosquittio.exe to the **System Environment Variables PATH** [ ' *C:\Program Files\mosquitto* ' ], which **allows** us to use **MQTT commands** directly in the *Command Prompt* or, *Terminal*.

* Starting @ boot byDefault:

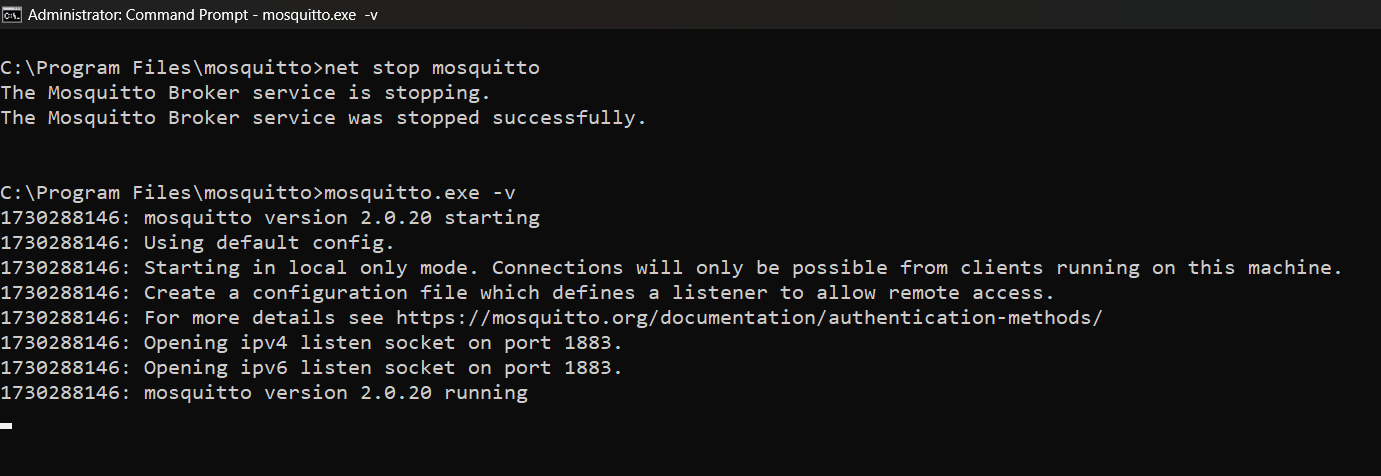
**net start mosquitto**

* Stopping:

In **Elevated** CMD > **net stop mosquitto**

* For Transmission: Navigate to [ **cd C:/Program Files/mosquitto** ]

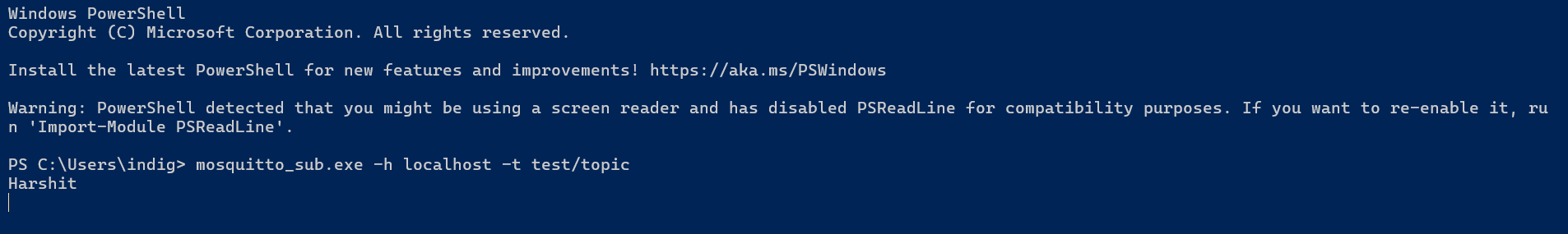
**mosquitto.exe -v**

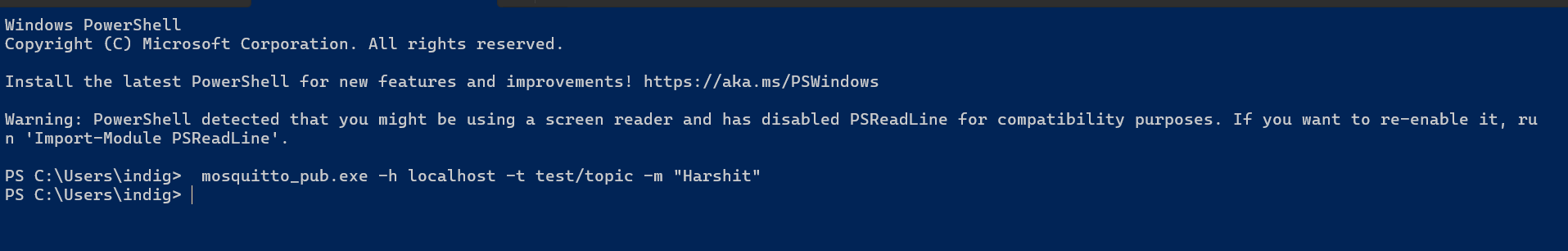


* **Testing MQTT Services [ Message Transmission]**

- Open 2 Terminals:

# 1st: **mosquitto\_sub.exe -h localhost -t test/topic**

****

# 2nd: **mosquitto\_pub.exe -h localhost -t test/topic -m "Harshit"**

**Lab#5**

**Realtime DHT Sensor Data on NodeRED 🀄**

* **Install Node.js :**
* Installed **NodeJS** from Official Eclipse Page [ [*https://nodejs.org/en/download/package-manager*](https://nodejs.org/en/download/package-manager)].
* Added node.js to the System Environment Variables PATH [ *C:/Users/Lokesh Patra/AppData/Roaming/npm* '], which **allows** us to use **npm** commands directly in the Command Prompt or, Terminal.
* **Installing & Initialising NodeRED:**
* Open Node.js > **npm install node-red-dashboard**
* [PostInstallation] > Elevated CMD: **node-red**
* In Client Application, browsed **http://127.0.0.1:1880/** [ *Accessing NodeRED* ]
* Inside the NodeRED window, a flow was created w/ the nodes as:

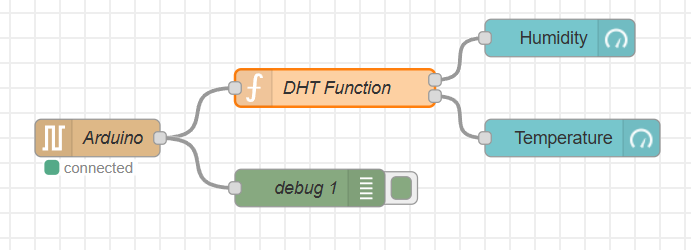
            > SERIAL-IN ( *Arduino Uno R3 Board* )

            > DEBUGGER

            > DHT FUNCTION

            > 2 GAUGES ( *Humidity & Temperature* )

Figure 9 Node-RED Flow Diagram



* Serial In Node: Configured it to read from the correct serial port where my Arduino is connected (e.g., COM7) > Set the baud rate to 9600.
* Configure the DHT Function as:

**var m = msg.payload.split(',');**

**if (m.length === 2) {**

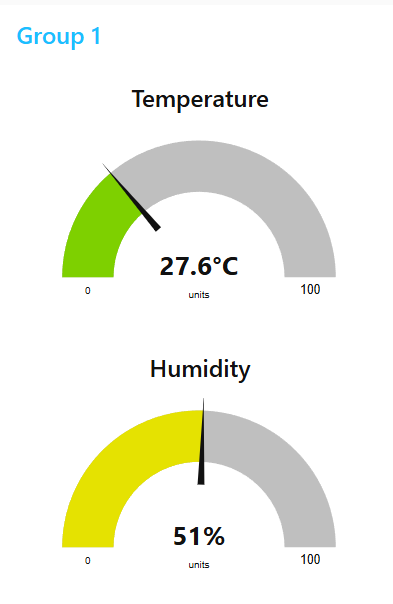
**var H = { payload: parseFloat(m[0]) };**

**var T = { payload: parseFloat(m[1]) };**

**return [H, T];**

**} else {**

**return null; }**

* Adjusting Gauge Nodes:

Humidity:

        - Title as “Humidity”.

        - Value format as ‘ **{{value}}%** ’.

        - Range Value: 0 ~ 100 %.

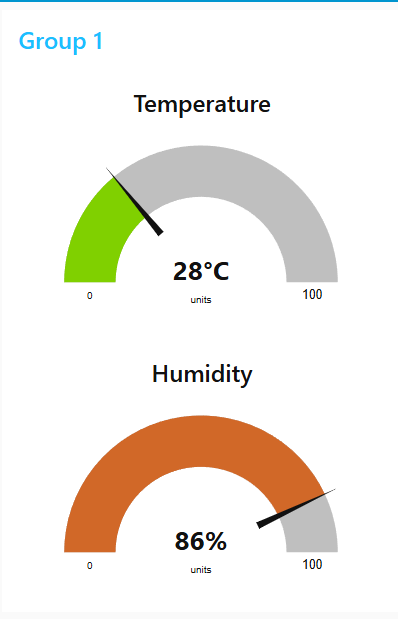
Temperatue:

        - Title as ' **Temperature** '.

        - Value format as ‘ **{{value}}°C** ’.

*\*\*Ensure that Humidity & Temperature are in the* ***same*** *group.*

* **Deployment:**
* Uploaded DHT11 /22 Sketch to the Arduino Board through its IDE:

**#include <DHT.h>**

**#define DHTPIN 3**

**#define DHTTYPE DHT11**

**DHT dht(DHTPIN, DHTTYPE);**

**void setup() {**

**Serial.begin(9600);**

**dht.begin();**

**}**

**void loop()  {**

**float H = dht.readHumidity();**

**float T = dht.readTemperature();**

**if (isnan(H) || isnan(T)) {**

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

**} else {**

**Serial.println(String(H) + "," + String(T));**

**}**

**delay(2000);**

**}**

* After uploading this sketch, close the IDE.
* Deploy the flow in NodeRED.
* Check the Dashboard in the upper-right corner, for the Humidity and Temperature Gauge.

**Lab#6**

**Arduino Project: Button, LED & Ultrasonic Sensor (HC-SR04)**

**Components Needed**

* Arduino UNO R3
* Push Button
* LED
* Ultrasonic Sensor (HC-SR04)
* Jumper Wires
* Breadboard (optional)

**Wiring Instructions**

**Button Connection**

1. Connect two jumper wires diagonally to the button.
   * **Wire 1**: Connect one wire to Digital Pin 2.
   * **Wire 2**: Connect the other wire to GND.

**LED Connection**

1. Connect the LED to the board:
   * **Anode (Longer leg, +ve)**: Connect to Digital Pin 13.
   * **Cathode (Shorter leg, -ve)**: Connect to GND.

**Ultrasonic Sensor Connection**

1. Connect the HC-SR04 sensor:
   * **VCC**: Connect to 5V.
   * **Trigger**: Connect to Digital Pin 9.
   * **Echo**: Connect to Digital Pin 8.
   * **GND**: Connect to GND.

**Sketches**

**Step 1: Button Functionality**

Upload the following sketch to the Arduino IDE to test the button functionality.

const int buttonPin = 2; // Pin where the button is connected

void setup() {

Serial.begin(115200); // Initialize serial communication at 115200 baud

pinMode(buttonPin, INPUT\_PULLUP); // Set the button pin as input with internal pull-up resistor

}

void loop() {

int buttonState = digitalRead(buttonPin); // Read the state of the button

if (buttonState == LOW) { // Check if the button is pressed

Serial.println("Button Held!"); // Print message to Serial Monitor

} else {

Serial.println("Button Released!"); // Print message to Serial Monitor

}

delay(500); // Add a small delay to debounce the button

}

**Output**: Displays "Button Held!" when pressed and "Button Released!" when released.

**Step 2: Button + LED**

To make the LED blink when the button is pressed, upload the following sketch.

const int buttonPin = 2; // Pin where the button is connected

const int ledPin = 13; // Pin where the LED is connected (built-in LED for most Arduino boards)

void setup() {

pinMode(buttonPin, INPUT\_PULLUP); // Set the button pin as input with internal pull-up resistor

pinMode(ledPin, OUTPUT); // Set the LED pin as output

}

void loop() {

int buttonState = digitalRead(buttonPin); // Read the state of the button

if (buttonState == LOW) { // Check if the button is pressed

digitalWrite(ledPin, HIGH); // Turn the LED on

delay(500); // Wait for 500 milliseconds

digitalWrite(ledPin, LOW); // Turn the LED off

delay(500); // Wait for 500 milliseconds

} else {

digitalWrite(ledPin, LOW); // Ensure the LED stays off when the button is not pressed

}

}

**Output**: The external LED blinks every 500 ms while the button is held.

**Step 3: Ultrasonic Sensor**

To measure distance with the HC-SR04 sensor, upload the following sketch.

#define PIN\_TRIG 9 // Define the pin for the trigger

#define PIN\_ECHO 8 // Define the pin for the echo

void setup() {

Serial.begin(9600); // Initialize serial communication at 9600 baud

pinMode(PIN\_TRIG, OUTPUT); // Set the trigger pin as output

pinMode(PIN\_ECHO, INPUT); // Set the echo pin as input

}

void loop() {

// Start a new measurement:

digitalWrite(PIN\_TRIG, HIGH); // Set the trigger pin high

delayMicroseconds(10); // Wait for 10 microseconds

digitalWrite(PIN\_TRIG, LOW); // Set the trigger pin low

// Read the result:

int duration = pulseIn(PIN\_ECHO, HIGH); // Read the duration of the pulse from the echo pin

// Calculate distance in centimeters:

Serial.print("Distance in CM: ");

Serial.println(duration / 58); // Print the distance in centimeters

// Calculate distance in inches:

Serial.print("Distance in inches: ");

Serial.println(duration / 148); // Print the distance in inches

delay(1000); // Wait for 1 second before taking the next measurement

}

**Output**: Displays distance in centimeters and inches.

**Step 4: HC-SR04 + Button Integration**

Finally, to integrate the ultrasonic sensor and button functionality, upload the following sketch.

#define PIN\_TRIG 9 // Define the pin for the trigger

#define PIN\_ECHO 8 // Define the pin for the echo

#define buttonPin 2 // Define the pin for the button

void setup() {

Serial.begin(9600); // Initialize serial communication at 9600 baud

pinMode(PIN\_TRIG, OUTPUT); // Set the trigger pin as output

pinMode(PIN\_ECHO, INPUT); // Set the echo pin as input

pinMode(buttonPin, INPUT\_PULLUP); // Set the button pin as input with internal pull-up resistor

}

void loop() {

int buttonState = digitalRead(buttonPin); // Read the state of the button

if (buttonState == LOW) { // Check if the button is pressed

// Start a new measurement:

digitalWrite(PIN\_TRIG, HIGH); // Set the trigger pin high

delayMicroseconds(10); // Wait for 10 microseconds

digitalWrite(PIN\_TRIG, LOW); // Set the trigger pin low

// Read the result:

int duration = pulseIn(PIN\_ECHO, HIGH); // Read the duration of the pulse from the echo pin

// Calculate distance in centimeters:

Serial.print("Distance in CM: ");

Serial.println(duration / 58); // Print the distance in centimeters

// Calculate distance in inches:

Serial.print("Distance in Inches: ");

Serial.println(duration / 148); // Print the distance in inches

} else {

Serial.println("Button Released!"); // Print message to Serial Monitor

}

delay(500); // Add a small delay to debounce the button

}

**Output**: Displays the distance in centimeters and inches when the button is pressed, and shows "Button Released!" when the button is not pressed.

[wokwi Lab-6 @ Sketch](https://wokwi.com/projects/413164898303761409)

**Lab#7**

**Use of Breadboard**

**Objective**

To understand the functionality and application of a breadboard for building and testing electronic circuits without soldering.

**Materials Needed**

* Breadboard
* Jumper wires
* LEDs
* Resistors (220Ω and 1kΩ)
* Power supply (battery or DC power supply)
* Arduino (optional, for more complex circuits)

**Introduction**

A breadboard is a reusable platform for prototyping electronic circuits. It allows components to be easily inserted and connected without soldering. The breadboard consists of a grid of holes, organized into rows and columns, where components can be placed and connected using jumper wires.

**Step-by-Step Procedure**

**Step #1: Understand the Breadboard Layout**

* **Power Rails:** The outer rows (usually marked with red and blue) are used for power distribution. The red rail is typically for positive voltage, and the blue rail is for ground (GND).
* **Terminal Strips:** The central area consists of vertical and horizontal strips where components are connected. Each vertical strip is connected internally, allowing components to be linked without jumper wires.

**Step #2: Connect Power and Ground**

1. Connect the positive terminal of the power supply to the red power rail.
2. Connect the negative terminal of the power supply to the blue power rail.

**Step #3: Build a Simple LED Circuit**

1. Insert the LED into the breadboard:
   * Place the long leg (anode) into a hole in the terminal strip.
   * Place the short leg (cathode) into another hole.
2. Connect a resistor (220Ω) to the cathode of the LED and then connect the other leg of the resistor to the ground rail (blue).
3. Connect a jumper wire from the anode of the LED to the positive rail (red).

**Step #4: Power the Circuit**

* Turn on the power supply. The LED should light up, indicating that the circuit is complete and functioning.

**Step #5: Experiment with Circuit Modifications**

* Replace the 220Ω resistor with a 1kΩ resistor and observe the changes in brightness of the LED.
* Try adding multiple LEDs in series or parallel and adjust the resistor values accordingly.

**Conclusion**

Using a breadboard allows for rapid prototyping and testing of electronic circuits. Understanding how to properly use a breadboard is essential for building and experimenting with various electronic components and systems.

**Safety Precautions**

* Ensure that the power supply voltage matches the requirements of the components being used.
* Do not exceed the current rating of the breadboard to avoid damage.
* Disconnect power before making changes to the circuit.

**Lab#8**

**Setting Up ESP32**

**Step #1: Install ESP32 Board Package**

* Open Arduino IDE.
* Navigate to File > Preferences.
* In the Additional Boards Manager URLs field, paste:

https://dl.espressif.com/dl/package\_esp32\_index.json

* Click OK.

**Step #2: Install the ESP32 Board**

* Go to Tools > Board > Boards Manager.
* Search for esp32 by Espressif Systems and click Install.

**Step #3: Install CP210x USB-to-UART Driver**

* Download the **CP210x USB-to-UART Bridge VCP Driver** from the Silicon Labs website.
* Follow installation instructions for your operating system.

**Step #4: Configure Arduino IDE for ESP32**

* Reboot your computer after driver installation.
* Connect your **ESP32** to your computer via USB.
* In **Arduino IDE**, select the correct **COM port** (e.g., **COM12**).
* Choose **ESP32 Dev Module** under **Tools > Board**.

**Note:**

* While uploading a sketch, press and hold the **BOOT button** during the upload process for 3-4 seconds after the "Connecting..." message.
* Alternatively, hold the **BOOT button**, press the **EN button** for 1 second, then release the **EN button** followed by the **BOOT button** to enter bootloader mode.

**Step #5: Blink the Internal LED**

* Use the following code in the IDE:

#define LED\_PIN 2 // Onboard LED connected to GPIO 2

void setup() {

pinMode(LED\_PIN, OUTPUT); // Initialize LED pin as output

}

void loop() {

digitalWrite(LED\_PIN, HIGH); // Turn LED on

delay(1000); // Wait 1 second

digitalWrite(LED\_PIN, LOW); // Turn LED off

delay(1000); // Wait 1 second

}

* Click **Upload** to upload the code. The internal LED will blink on and off every second.

[**wokwi Lab-8 @ Sketch**](https://wokwi.com/projects/413177461697813505)