IoT & Automation Lab. Record

**Lab#1**

**Blinking the InBuilt LED**

void setup() {

pinMode(LED\_BUILTIN, OUTPUT);

}

void loop() {

digitalWrite(LED\_BUILTIN, HIGH);

delay(500);

digitalWrite(LED\_BUILTIN, LOW);

delay(500);

****}

Figure Uno R3: InBuilt LED Blink

[**Wokwi Link #1.0**](https://wokwi.com/projects/406490864427470849)

**Lab#2**

**Blinking an External LED ( Red ) w/ Resistor**

#define light 12

void setup() {

  pinMode(light, OUTPUT);

}

void loop() {

  digitalWrite(light, HIGH);

  delay(200);

  digitalWrite(light, LOW);

  delay(500);

  }

****

Figure Uno R3: External LED Blink

[**Wokwi Link #1**](https://wokwi.com/projects/406483824958897153)

**Lab#3**

**Using a Digital Humidity & Temperature Sensor**

#include <DHT.h>

#define light 7

#define DHTTYPE DHT22

DHT dht(light, DHTTYPE);

float humid, temp;

void setup() {

  Serial.begin(9600);

  dht.begin();

}

void loop() {

Figure Uno R3: DHT22

  delay(200);

  humid = dht.readHumidity();

  temp = dht.readTemperature();

  Serial.print("Humidity: ");

  Serial.print(humid);

  Serial.print(" %  Temperature: ");

  Serial.print(temp);

  Serial.println("°C");

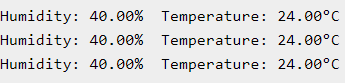
  delay(1000);

Figure Wokwi Output [ DHT Sensor ]

}

[**Wokwi Link #2**](https://wokwi.com/projects/406487282785269761)

**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**

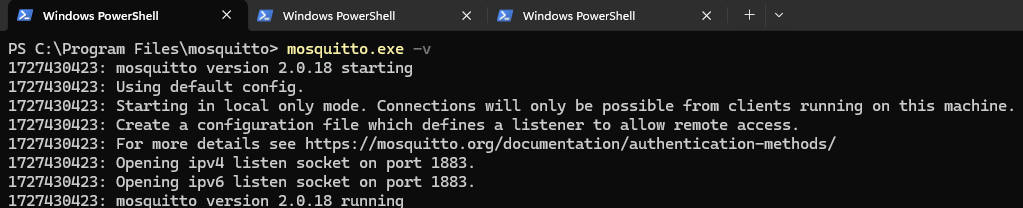
 *// -v is a Verbose Output flag, that enables us to see the backend processes, log messages, that'd help us to debug whenever necessary.*

Figure Mosquitto Initialization in WinOS [ Windows 11 ]

* **In Linux [ WSL\*: Ubuntu 22.04 LTS ]:**

- In Terminal > **wsl --install -d Ubuntu-22.04** > \ E / N \ T / E \ R /

- Restart the machine, and Launch Ubuntu 22.04

- **$sudo apt update**

- **$sudo apt install mosquitto mosquitto-clients**

* Starting mosquitto services:

- **$sudo systemctl ( enable /start ) mosquitto**

* Mosquitto Broker Service Status can be checked here:

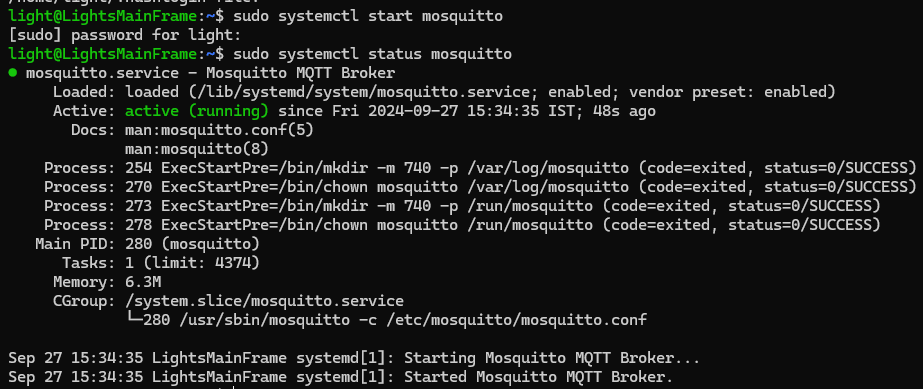
 - **$sudo systemctl status mosquitto**

Figure Mosquitto Initialization in Ubuntu 22.04

- Once verified service status, transmission can be carried on.

* Stopping mosquitto services:

- **$sudo systemctl stop mosquitto**

* **Testing MQTT Services [ Message Transmission: WinOS11 + Ubuntu 22.04 ]:**

- Open 2 Terminals:

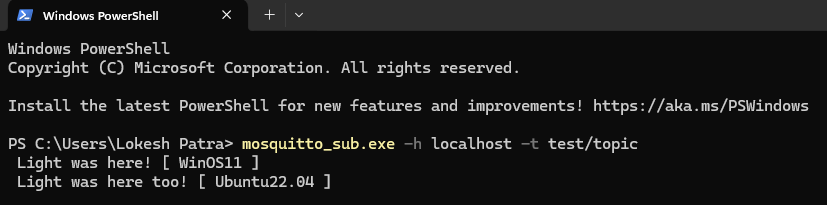
 # 1st: **mosquitto\_sub.exe -h localhost -t test/topic**

Figure MQTT Subscriber Testing [ Message Transmission ]

# 2nd: **mosquitto\_pub.exe -h localhost -t test/topic -m " Light was here! "**

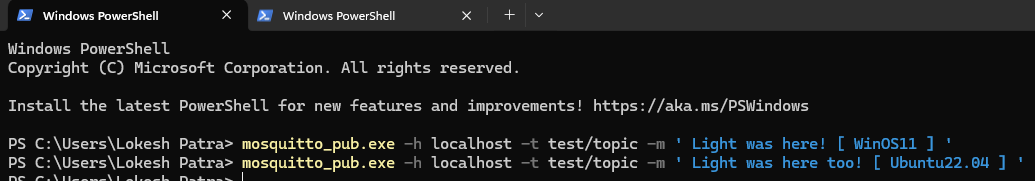


Figure MQTT Publisher Testing

\*Windows Subsystem for Linux

**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 **localhost:1880** [ *Accessing NodeRED* ]
* Inside the NodeRED window, a flow was created w/ the nodes as:

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

            > DEBUGGER

            > DHT FUNCTION

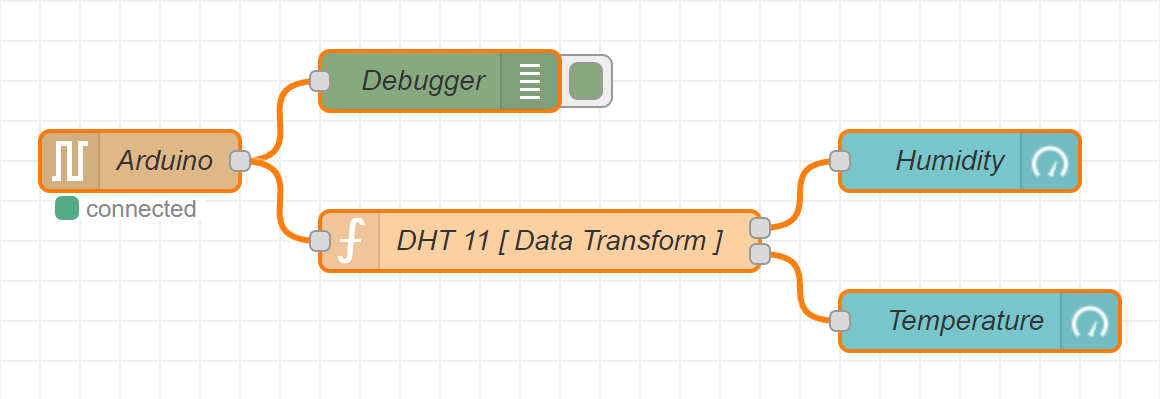
            > 2 GAUGES ( *Humidity & Temperature* )

Figure 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; }**

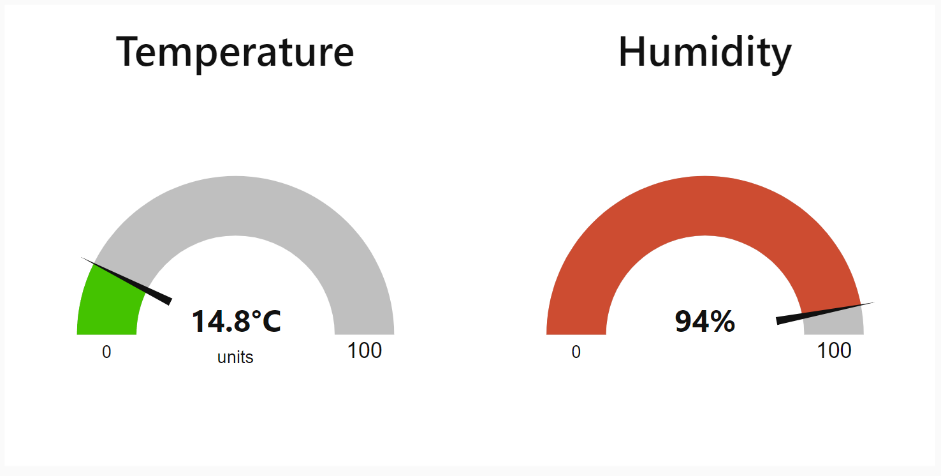
* Adjusting Gauge Nodes:

Figure NodeRED Dashboard [ Temperature + Humidity ]

Humidity:

        - Title as “ **Humidity** ”.

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

        - Range Value: 0 ~ 100 %.

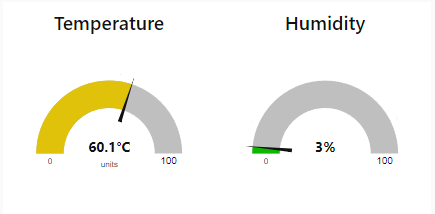
Temperatue:

        - 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();**

Figure Temperature maxxed @ 60.1° & Humidity @ min. (3% )

**}**

**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.