EEL 7170 : Introduction to IoT

Lab Report

Name:	**Anushkaa Ambuj**
Roll Number:	**B21ES006**
Program:	**B.Tech in ES**

Lab 5: Node-RED and MQTT

14 October, 2024

1 InLab

1.1 Objective

The objective of this lab is to gain hands-on experience with setting up and using Node-RED on a Raspberry Pi and configuring and utilizing the MQTT protocol for message brokering and data communication in IoT applications.

1.2 Components Used

- Raspberry Pi
- NodeMCU (ESP8266)
- DHT11 Sensor
- LED Bulb
- Jumper Wires & Breadboard
- NODE-RED (a flow-based development tool for visual programming)
- Mosquitto (MQTT Broker)

1.3 Procedure

Part 1: Setting Up Node-RED on Raspberry Pi

- **Step 1: Connecting Raspberry Pi to Hotspot**
- **Step 2: Installing Node-RED on Raspberry Pi**
  - Open a terminal on the Raspberry Pi and run the following command to install Node-RED:
    ```
    bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/
    master/deb/update-nodejs-and-nodered)
    ```
  - When prompted, press 'y' and hit Enter to continue.
- **Step 3: Configuring Node-RED**
  - After installation, set the username & password, (here, `admin` and `admin123`).
  - Choose `"Full access"` for user permissions, and skip adding another user by selecting 'No'.
  - Skip the projects features and set the default flows.json when prompted.

- Use the following command to start Node-RED:

  ```
  node-red-start
  ```

- **Step 4: Accessing Node-RED**
  - Open a browser on the Raspberry Pi and navigate to the following URL to access Node-RED:

    ```
    http://127.0.0.1:1880/
    ```

Part 2: Installing MQTT Broker on Raspberry Pi

- **Step 1: Installing Mosquitto**
  - Open a new terminal and install the Mosquitto MQTT broker using the following command:

    ```
    sudo apt install -y mosquitto mosquitto-clients
    ```

- **Step 2: Configuring Mosquitto**
  - Open the Mosquitto configuration file:

    ```
    sudo nano /etc/mosquitto/mosquitto.conf
    ```

  - Add the following lines to allow anonymous connections:

    ```
    listener 1883
    allow_anonymous true
    ```

  - Save and exit the file by pressing `Ctrl + X`, then 'Y', and Enter.
  - Restart the Mosquitto service to apply the changes:

    ```
    sudo systemctl restart mosquitto
    ```

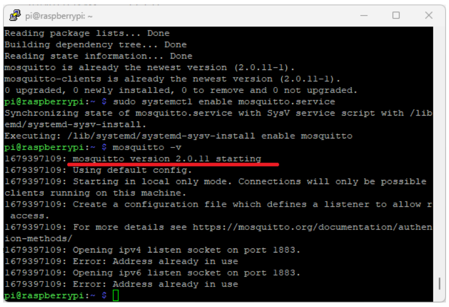
Part 3: Establishing an MQTT communication with Node-RED

Reference URL: https://randomnerdtutorials.com/esp8266-and-node-red-with-mqtt/

- **Step 1: Installing Node-RED Dashboard**
  - Open the Node-RED by the URL http://127.0.0.1:1880/, login, click on the menu icon on the top-right and go to "Manage palette.".
  - Under the "Install" tab, search for `node-red-dashboard` and install it.
- **Step 2: Dashboard Layout**
  - Click on the arrow icon in the top-right corner and select **Dashboard**.
  - Under the **Layout** tab, create a new tab called **Room**.
  - Inside the **Room** tab, create two new links (groups): **Lamp** and **Sensor**.

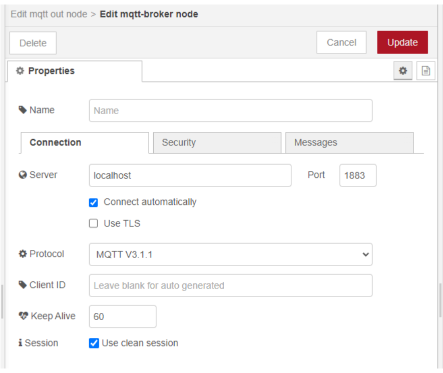
- **Step 3: Creating the Flow**
  - From the **Dashboard** section in the left sidebar, drag the following nodes to the flow:
    - **Switch** – Controls the ESP8266 output.
    - **Chart** – Displays the temperature sensor readings.
    - **Gauge** – Displays humidity sensor readings.
  - Drag the following MQTT nodes from the **Network** section:
    - Two **MQTT Input** nodes – Subscribe to the temperature and humidity topics to receive sensor data.
    - One **MQTT Output** node – Publishes a message to the ESP8266 based on the switch state.
- **Step 4: Connecting MQTT Broker to Node-RED**
  - Double-click on the **MQTT Output** node to open its properties.
  - Click on **Add new mqtt-broker** option.
  - In the **Server** field, enter **localhost** (or your Raspberry Pi's IP address if not using the same machine). Here, we enter `localhost`.
  - For now, ensure no changes are made in the Security tab when configuring the MQTT client.
  - Once done, click `Add`.

Notice:
- Check the port number of the MQTT broker (ie. 1883).



```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
mosquitto is already the newest version (2.0.11-1).
mosquitto-clients is already the newest version (2.0.11-1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~ $ sudo systemctl enable mosquitto.service
Synchronizing state of mosquitto.service with SysV service script with /lib
emd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable mosquitto
pi@raspberrypi:~ $ mosquitto -v
1679397109: mosquitto version 2.0.11 starting
1679397109: Using default config.
1679397109: Starting in local only mode. Connections will only be possible
clients running on this machine.
1679397109: Create a configuration file which defines a listener to allow r
 access.
1679397109: For more details see https://mosquitto.org/documentation/authen
ion-methods/
1679397109: Opening ipv4 listen socket on port 1883.
1679397109: Error: Address already in use
1679397109: Opening ipv6 listen socket on port 1883.
1679397109: Error: Address already in use
pi@raspberrypi:~ $
```

Figure 1: Starting Mosquitto on Terminal



Name: Name

Server: localhost Port: 1883

☑ Connect automatically

☐ Use TLS

Protocol: MQTT V3.1.1

Client ID: Leave blank for auto generated

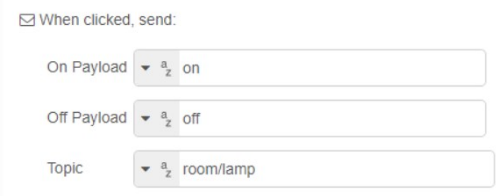
Keep Alive: 60

Session: ☑ Use clean session

Figure 2: Setting MQTT Server on NODE-RED

- the protocol being used
- MQTT Protocol Message Fields like **KeepAlive, Clean Session, etc.**

- **Step 5: Configuring the Switch Node**
  - The **Switch** node sends an `on` string message when it's on and `off` when it's off.
  - Set the topic as `room/lamp`, to which the ESP8266 will be subscribed to control the lamp.
  - Change the payload to type **string**.



☑ When clicked, send:

On Payload: on

Off Payload: off

Topic: room/lamp

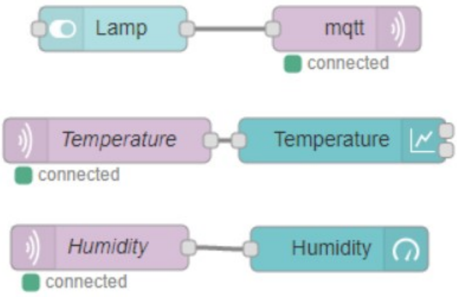
- **Step 6: Configuring MQTT Output Node**
  - The **MQTT Output** node is connected to the Mosquitto MQTT broker.
  - Publish to the topic `room/lamp`, set the **QoS** (Quality of Service) level, and configure the **Retain** message flag based on your requirement. Here, we set QoS as 2 and Retain as 'true'.

Notice: You can also set the Topic, QoS or Retain through `message properties`.

- **Step 7: Configuring MQTT Input Nodes**
  - One **MQTT Input** node subscribes to the `room/temperature` topic to receive temperature data from the ESP8266.
  - Another **MQTT Input** node subscribes to the `room/humidity` topic to receive humidity data.

Notice:
  * Field options of **Action, QoS, Output** format.
  * **'Action'** field has 2 options: 'Subscribe to single topic' & 'Dynamic Subscription'.
  * In the case of 'Dynamic Subscription', the output format would become slightly complex, requiring further parsing of the values as required.

- **Step 8: Configuring the Chart & Gauge Node**
- **Step 9: Wiring the Nodes**
  - Wire the **Switch**, **Chart**, **Gauge**, and the MQTT nodes as shown in the diagram.



- **Step 10: Deploying the Flow**
  - Once the flow is ready, click the **Deploy** button in the top-right corner.
  - The Node-RED application is ready. To see how your dashboard looks, go to http://127.0.0.1:1880/ui

Part 4: Building the Circuit

- **Connections of DHT11:**
  - GND pin of DHT11 to GND of NodeMCU
  - VCC pin to 3V3
  - DATA pin to D1
- **Connections of LED:**
  - Shorter Leg (Cathode(-)) of LED to GND pin of NodeMCU
  - Longer Leg (Anode(+)) to D2 pin

Part 5: Preparing your Arduino IDE

- **Install the PubSubClient Library** using Download Link. Then, in the Arduino IDE, go to Sketch > Include Library > Add .ZIP library and select the library .zip folder you've just downloaded.
- **Install DHT Sensor Library:** Open your Arduino IDE and go to Sketch > Include Library > Manage Libraries and search to install **Adafruit Unified Sensor** & **DHT sensor library by Adafruit**.

Part 6: Upload the code

- Modify the hotspot & MQTT broker credentials.
- Enter the IP Address of Raspberry Pi (Hotspot IP Address) as the IP address of MQTT server. You can find that by typing `ipconfig` in the terminal.

1.4 Code

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

// Uncomment one of the lines bellow for whatever DHT sensor type you're using!
#define DHTTYPE DHT11   // DHT 11
//#define DHTTYPE DHT21   // DHT 21 (AM2301)
//#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321

// Change the credentials below, so your ESP8266 connects to your router
const char* ssid = "SUPER-HP";
const char* password = "admin123";

// MQTT broker credentials (set to NULL if not required)
const char* MQTT_username = "admin";
const char* MQTT_password = "admin123";

// Change the variable to your Raspberry Pi IP address, so it connects to your MQTT broker
const char* mqtt_server = "192.168.137.64";
//For example
```

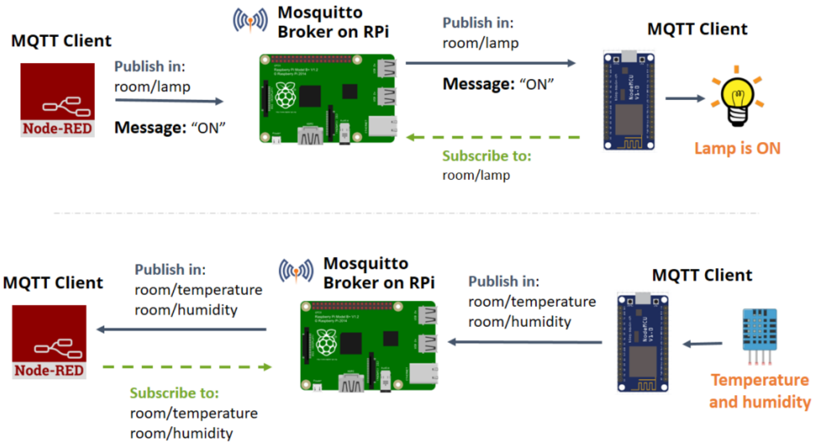
```
//const char* mqtt_server = "192.168.1.106";

// Initializes the espClient. You should change the espClient name if you have multiple ESPs running in your home automation system
WiFiClient espClient;
PubSubClient client(espClient);

// DHT Sensor - GPIO 5 = D1 on ESP-12E NodeMCU board
const int DHTPin = 5;

// Lamp - LED - GPIO 4 = D2 on ESP-12E NodeMCU board
const int lamp = 4;

// Initialize DHT sensor.
DHT dht(DHTPin, DHTTYPE);

// Timers auxiliar variables
long now = millis();
long lastMeasure = 0;

// This functions connects your ESP8266 to your router
void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("WiFi connected - ESP IP address: ");
  Serial.println(WiFi.localIP());
}

// This function is executed when some device publishes a message to a topic that your ESP8266 is subscribed to
// Change the function below to add logic to your program, so when a device publishes a message to a topic that
// your ESP8266 is subscribed you can actually do something
void callback(String topic, byte* message, unsigned int length) {
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  String messageTemp;

  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageTemp += (char)message[i];
  }
  Serial.println();

  // Feel free to add more if statements to control more GPIOs with MQTT

  // If a message is received on the topic room/lamp, you check if the message is either on or off. Turns the lamp GPIO according to the message
  if(topic=="room/lamp"){
      Serial.print("Changing Room lamp to ");
      if(messageTemp == "on"){
        digitalWrite(lamp, HIGH);
        Serial.print("On");
      }
      else if(messageTemp == "off"){
        digitalWrite(lamp, LOW);
        Serial.print("Off");
      }
  }
  Serial.println();
}

// This functions reconnects your ESP8266 to your MQTT broker
// Change the function below if you want to subscribe to more topics with your ESP8266
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    /*
     YOU MIGHT NEED TO CHANGE THIS LINE, IF YOU'RE HAVING PROBLEMS WITH MQTT MULTIPLE CONNECTIONS
     To change the ESP device ID, you will have to give a new name to the ESP8266.
     Here's how it looks:
       if (client.connect("ESP8266Client")) {
     You can do it like this:
       if (client.connect("ESP1_Office")) {
     Then, for the other ESP:
       if (client.connect("ESP2_Garage")) {
      That should solve your MQTT multiple connections problem
    */
    if (client.connect("ESP8266Client", MQTT_username, MQTT_password)) {
      Serial.println("connected");
      // Subscribe or resubscribe to a topic
      // You can subscribe to more topics (to control more LEDs in this example)
      client.subscribe("room/lamp");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
```

```
}

// The setup function sets your ESP GPIOs to Outputs, starts the serial communication at a baud rate of 115200
// Sets your mqtt broker and sets the callback function
// The callback function is what receives messages and actually controls the LEDs
void setup() {
  pinMode(lamp, OUTPUT);

  dht.begin();

  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

}

// For this project, you don't need to change anything in the loop function. Basically it ensures that you ESP is connected to your broker
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  if(!client.loop())
    client.connect("ESP8266Client", MQTT_username, MQTT_password);

  now = millis();
  // Publishes new temperature and humidity every 30 seconds
  if (now - lastMeasure > 30000) {
    lastMeasure = now;
    // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
    float humidity = dht.readHumidity();
    // Read temperature as Celsius (the default)
    float temperatureC = dht.readTemperature();
    // Read temperature as Fahrenheit (isFahrenheit = true)
    float temperatureF = dht.readTemperature(true);

    // Check if any reads failed and exit early (to try again).
    if (isnan(humidity) || isnan(temperatureC) || isnan(temperatureF)) {
      Serial.println("Failed to read from DHT sensor!");
      return;
    }

    // Publishes Temperature and Humidity values
    client.publish("room/temperature", String(temperatureC).c_str());
    client.publish("room/humidity", String(humidity).c_str());
    //Uncomment to publish temperature in F degrees
    //client.publish("room/temperature", String(temperatureF).c_str());

    Serial.print("Humidity: ");
    Serial.print(humidity);
    Serial.println(" %");
    Serial.print("Temperature: ");
    Serial.print(temperatureC);
    Serial.println("  C ");
    Serial.print(temperatureF);
    Serial.println("  F ");
  }
}
```

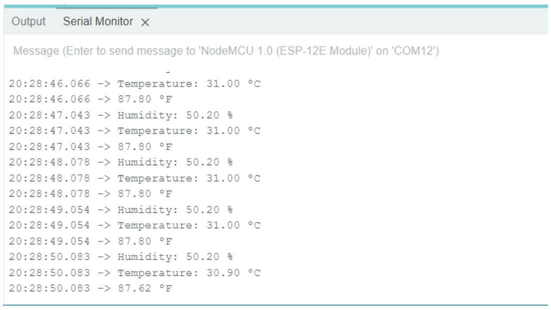
[language:C++]

1.5 Observations & Results

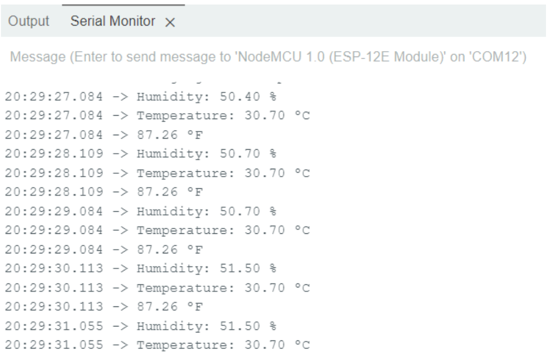
- Node-RED was successfully installed and configured on the Raspberry Pi.
  Note: In case of any error, try to reboot or update and upgrade Raspberry Pi
- The Mosquitto MQTT broker was installed and configured to allow anonymous connections.
- Node-RED Dashboard was installed, allowing visualization of the MQTT data.
- Successful communication between devices using the MQTT protocol was established.



- Serial Monitor Output


Output
Serial Monitor ×
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM12')

```
20:28:46.066 -> Temperature: 31.00 °C
20:28:46.066 -> 87.80 °F
20:28:47.043 -> Humidity: 50.20 %
20:28:47.043 -> Temperature: 31.00 °C
20:28:47.043 -> 87.80 °F
20:28:48.078 -> Humidity: 50.20 %
20:28:48.078 -> Temperature: 31.00 °C
20:28:48.078 -> 87.80 °F
20:28:49.054 -> Humidity: 50.20 %
20:28:49.054 -> Temperature: 31.00 °C
20:28:49.054 -> 87.80 °F
20:28:50.083 -> Humidity: 50.20 %
20:28:50.083 -> Temperature: 30.90 °C
20:28:50.083 -> 87.62 °F
```


Output
Serial Monitor ×
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM12')

```
20:29:27.084 -> Humidity: 50.40 %
20:29:27.084 -> Temperature: 30.70 °C
20:29:27.084 -> 87.26 °F
20:29:28.109 -> Humidity: 50.70 %
20:29:28.109 -> Temperature: 30.70 °C
20:29:28.109 -> 87.26 °F
20:29:29.084 -> Humidity: 50.70 %
20:29:29.084 -> Temperature: 30.70 °C
20:29:29.084 -> 87.26 °F
20:29:30.113 -> Humidity: 51.50 %
20:29:30.113 -> Temperature: 30.70 °C
20:29:30.113 -> 87.26 °F
20:29:31.055 -> Humidity: 51.50 %
20:29:31.055 -> Temperature: 30.70 °C
```

- The data sent via MQTT was displayed on the Node-RED dashboard, verifying both the broker's and Node-RED's functioning.

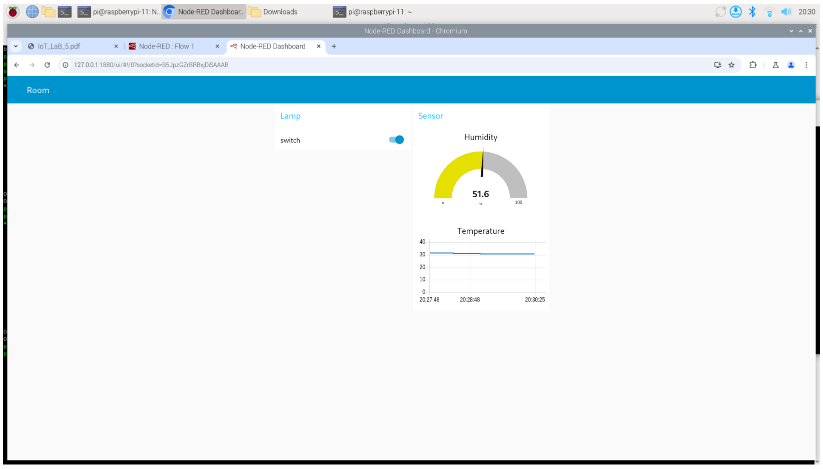


Figure 3: With Switch ON

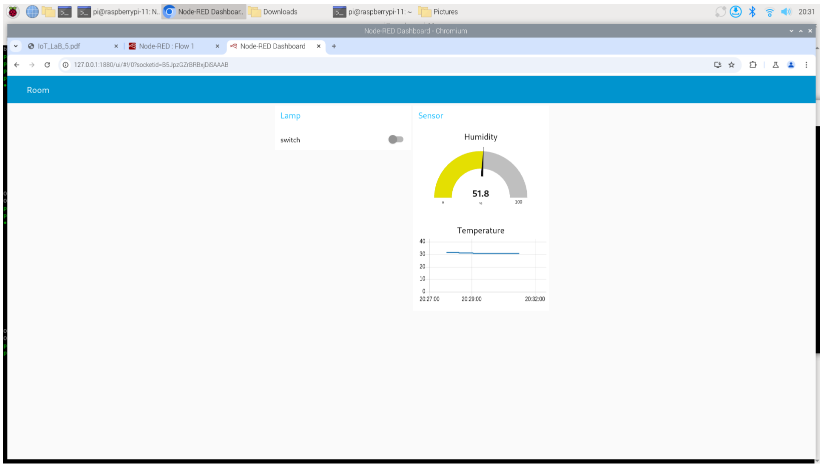


Figure 4: With Switch OFF