ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

Intelligent and Communicating Systems, ICS
2^{nd} Year Specialty SIQ G02, 2CS SIQ2

LAB report n°10

Title:

Iot System based Platform Broker MQTT - Node Red

Studied by:
First Name: Nour el Imane
Last Name: HEDDADJI
E-mail: jn_heddadji@esi.dz

A. Theory

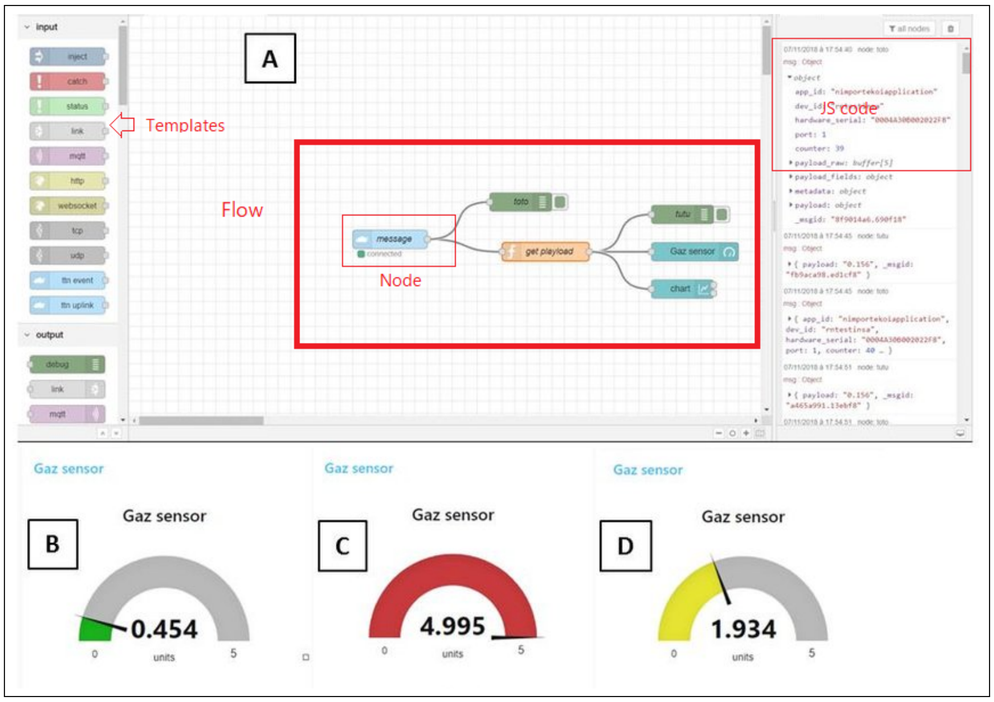
1. Node-RED

Node-RED is an **open-source, flow-based programming tool** initially developed by IBM and currently managed by the Node-RED project.

It serves as a visual canvas for simplifying the creation of Internet of Things (IoT) applications. With a **web-based editor** featuring drag-and-drop functionality, developers seamlessly connect predefined code blocks, or 'nodes,' to perform tasks.

These nodes, encompassing inputs, processing, and outputs, are visually linked within the editor to form a 'flow.'

One of the key benefits of Node-RED is that it simplifies the process of connecting disparate systems and devices by providing pre-built nodes for popular platforms and services such as Twitter, Slack, IBM Watson, and Amazon Web Services.



```
07/11/2018 à 17:54:40  node: toto
msg : Object
▾object
  app_id: "nimportekoiapplication"
  dev_id: [illegible]
  hardware_serial: "0004A30B002022F8"
  port: 1
  counter: 39
 ▸payload_raw: buffer[5]
 ▸payload_fields: object
 ▸metadata: object
 ▸payload: object
  _msgid: "8f9014a6.690f18"
07/11/2018 à 17:54:45  node: tutu
msg : Object
 ▸{ payload: "0.156", _msgid:
"fb9aca98.ed1cf8" }
07/11/2018 à 17:54:45  node: toto
msg : Object
 ▸{ app_id: "nimportekoiapplication",
dev_id: "rntestinsa",
hardware_serial: "0004A30B002022F8",
port: 1, counter: 40 … }
07/11/2018 à 17:54:51  node: tutu
msg : Object
 ▸{ payload: "0.156", _msgid:
"a465a991.13ebf8" }
07/11/2018 à 17:54:51  node: toto
```

Figure 1: Node-Red interface

1.1. Basics of Node-RED

To seamlessly integrate IoT sensors and visualize real-time data using Node-RED, follow these straightforward steps:

1. **Install Node-RED:**
   - Begin by installing Node-RED on your local machine.
2. **Node-RED editor:**
   - Access the Node-RED editor by navigating to the URL `http://localhost:1880` in your web browser. This web-based editor serves as the hub for building your IoT flows.
3. **Create a Dashboard:**
   - Navigate to the ”Manage Palette” menu, select ”Install” tab, and search for ”node-red-dashboard.”
4. **Create a New Flow:**
   - Click on the ”New Flow” button within the editor to create dedicated space for your IoT application’s logic.
5. **Add Nodes:**
   - Enhance your flow by adding nodes. Utilize the drag-and-drop functionality to effortlessly place nodes onto the canvas, each representing a specific function such as input, processing, or output.
6. **Configure Nodes:**
   - By double-clicking on it, opening a configuration dialog.
7. **Connect Nodes:**
   - Establish seamless communication between nodes by connecting them. Simply drag a wire from the output of one node to the input of another.
8. **Deploy Your Flow:**
   - Click on the ”Deploy” button. This action activates your configured flow. The dashboard updates in real-time, offering a dynamic display of your sensor data.

1.2. Comparisons between Node-RED and other popular IoT platforms

Aspect	Node-RED	OpenHAB
Programming Model	Visual programming with a web-based editor.	Rule-based automation with configuration relying on text-based files.
Ease of Use	Less coding and a drag-and-drop interface.	Harder to use due to text-based configuration files.
Extensibility	Rich library of pre-built nodes for various platforms such as Twitter, Slack etc	Supports a wide range of smart home devices and protocols.
Integration	Seamless integration with different devices and services.	Designed for home automation, providing compatibility with various devices and protocols.
Configuration	Simplified configuration through a visual interface.	Configuration relies on text-based files, providing granular controls.
Community Support	Growing smaller community.	Established and thriving user community contributing to a vast library of bindings and extensions.

Table 1: Comparison of Node-RED and OpenHAB

2. MQTT Protocol

MQTT, or Message Queuing Telemetry Transport, stands out as a lightweight protocol tailored for IoT devices with limited resources. Developed by IBM, it employs a publish-subscribe model for efficient communication.

2.1. Publish-Subscribe Model

The MQTT Publish-Subscribe pattern has four main components: Publisher, Subscriber, Broker, and Topic.

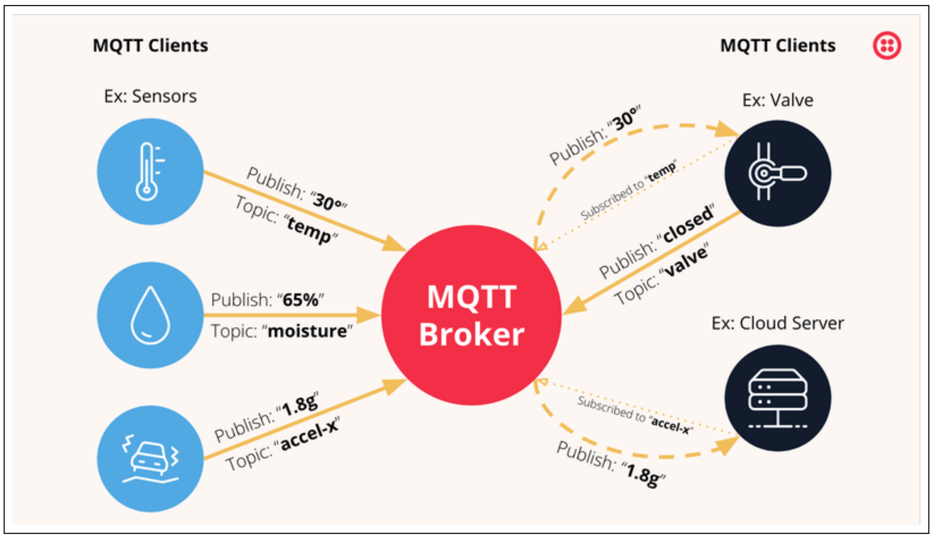


Figure 2: MQTT : publish-subscribe pattern

Publisher

The role of the publisher is to send messages to a specific topic. It can publish data to one topic at a time, and it does not need to worry about the online status of subscribers when sending a message.

Subscriber

Subscribers receive messages by subscribing to a particular topic and have the flexibility to subscribe to multiple topics concurrently. MQTT also supports **load-balancing** subscriptions among multiple subscribers through shared subscriptions.

Broker

The broker acts as an **intermediary**, receiving messages from publishers and forwarding them to the relevant subscribers. Additionally, the broker manages client requests for connecting, disconnecting, subscribing, and unsubscribing.

Topic

In MQTT, messages are routed based on topics. Topics are hierarchical and typically structured with slashes (/) between levels, akin to URL paths. For instance, a topic might be "sensor/1/temperature." Multiple subscribers can subscribe to the same topic, and the broker efficiently forwards all messages on that topic to these subscribers. Similarly, multiple publishers can send messages to the same topic, with the broker routing these messages in the order they are received to the subscribed clients.

MQTT supports **topic wildcards**, allowing subscribers to simultaneously subscribe to multiple topics with a single subscription. This feature enhances flexibility, enabling subscribers to receive messages from diverse topics through a unified subscription.

3. Comparison with Other Protocols

3.1. AMQP (Advanced Message Queuing Protocol)

- AMQP is a more complex messaging protocol compared to MQTT and is specifically designed for use in enterprise applications.
- It supports advanced features like message routing, transactional messaging, and message fragmentation.
- However, AMQP is less lightweight than MQTT, making it less suitable for use in constrained environments.

3.2. CoAP (Constrained Application Protocol)

- CoAP is designed specifically for use in constrained environments and aims to be simple and lightweight, minimizing message overhead.
- It supports resource discovery, request-response interactions, and block-wise transfers.
- Despite its simplicity, CoAP is less flexible than MQTT and lacks some of the advanced features present in AMQP.

B. Activity

1. Getting Familiar with MQTT

1.1. Installation of MQTT-Cli

To begin, visit the MQTT-Cli website (`https://hivemq.github.io/mqtt-cli/docs/installation/`) to download and install the MQTT-Cli tool on your operating system. Follow the provided instructions for installation.

1.2. Publish to a Topic

The general command to publish a topic is:

```
mqtt-cli.exe sub-h address -t your_topic -m "Your message here"
```

In this activity, we will be publishing two topics, ”Light” and ”Temperature” to our local machine.

1. Open a command prompt or terminal window.
2. Navigate to the folder containing mqtt-cli.exe.
3. Use the following command to publish an empty message to the ”Light” topic on the local MQTT broker:

```

    mqtt−cli.exe pub −t Light −h localhost −p 1883 −m:empty

```

4. Use the following command to publish an empty message to the ”Temperature” topic on the local MQTT broker:

```

    mqtt−cli.exe pub −t Temperature −h localhost −p 1883 −m:empty

```

1.3. Subscribe to a Topic

The general command to subscribe to a topic is:

```
mqtt-cli.exe sub -h address -t your_topic
```

Now, let's experiment with subscribing to a single or multiple topics.

1. Open a new command prompt or terminal window.
2. Use the following command to subscribe to **the "Light" topic**:

```
mqtt−cli.exe sub −t Light −h localhost −p 1883
```

3. Open a third command prompt or terminal window.
4. Use the following command to subscribe to **both "Light" and "Temperature" topics**:

```
mqtt−cli.exe sub −t Light −t Temperature −h localhost −p 1883
```

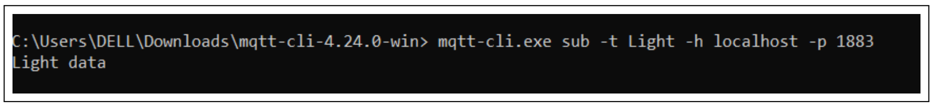
1.4. Test Your MQTT Broker

Now, let's publish on both topics and observe the results:

1. In the first command prompt, publish a message to the "Light" topic:

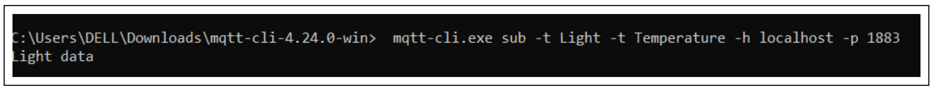
```
mqtt−cli.exe pub −t Light −h localhost −p 1883 −m "Light data"
```

2. Observe the second and third command prompt subscribed to "Light". You should see the message related to the "Light" topic.



```
C:\Users\DELL\Downloads\mqtt-cli-4.24.0-win> mqtt-cli.exe sub -t Light -h localhost -p 1883
Light data
```

Figure 3: Subscriber n:01



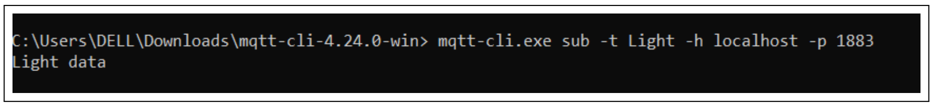
```
C:\Users\DELL\Downloads\mqtt-cli-4.24.0-win>  mqtt-cli.exe sub -t Light -t Temperature -h localhost -p 1883
Light data
```

Figure 4: Subscriber n:02

3. In the first command prompt, publish a message to the ”Temperature” topic:

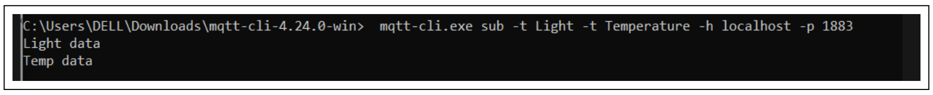
```

mqtt−cli.exe pub −t Temperature −h localhost −p 1883 −m "Temp data"

```

4. Observe the second and third command prompt. You should see messages related to ”Temperature” topic in **the third prompet only.**



```
C:\Users\DELL\Downloads\mqtt-cli-4.24.0-win> mqtt-cli.exe sub -t Light -h localhost -p 1883
Light data
```

Figure 5: Subscriber n:01



```
C:\Users\DELL\Downloads\mqtt-cli-4.24.0-win>  mqtt-cli.exe sub -t Light -t Temperature -h localhost -p 1883
Light data
Temp data
```

Figure 6: Subscriber n:02

2. Getting Familiar with Node-red

2.1. Installation of Node-RED

To start working with Node-RED, you need to install it on your local machine. Follow these steps:

1. Open a command prompt or terminal window.
2. Install Node-RED using the following command:

```

npm install −g node−red

```

3. Once the installation is complete, start Node-RED:

```

node–red

```

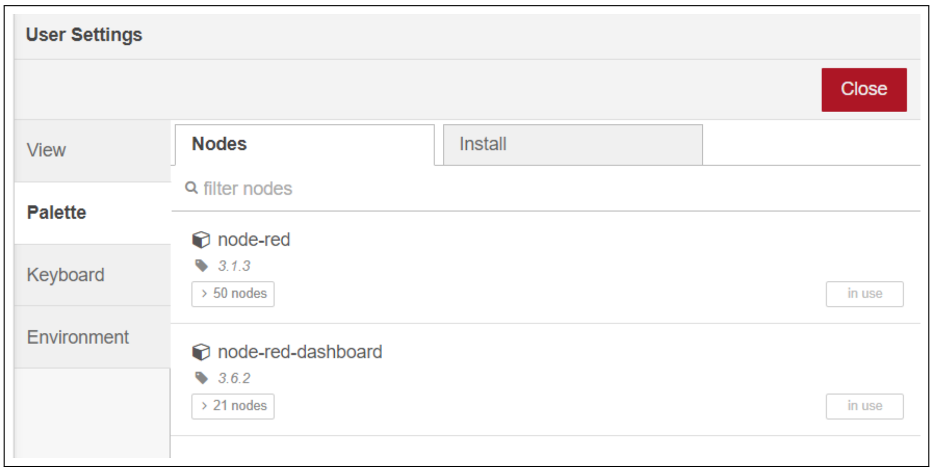
4. Open your web browser and navigate to `http://localhost:1880`.

2.2. Creating a Flow in Node-RED

Now, let's create a flow to subscribe to MQTT topics and visualize the data.

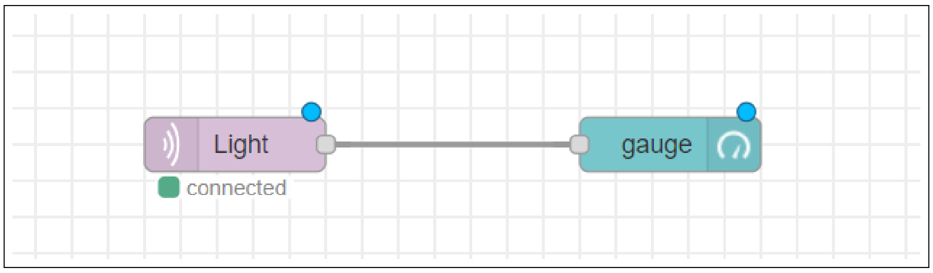
MQTT Nodes Installation

1. Click on the three horizontal lines in the top-right corner of the Node-RED interface to open the menu.
2. Select "Manage palette."
3. Go to the "Install" tab and search for "node-red-dashboard." Click "Install".



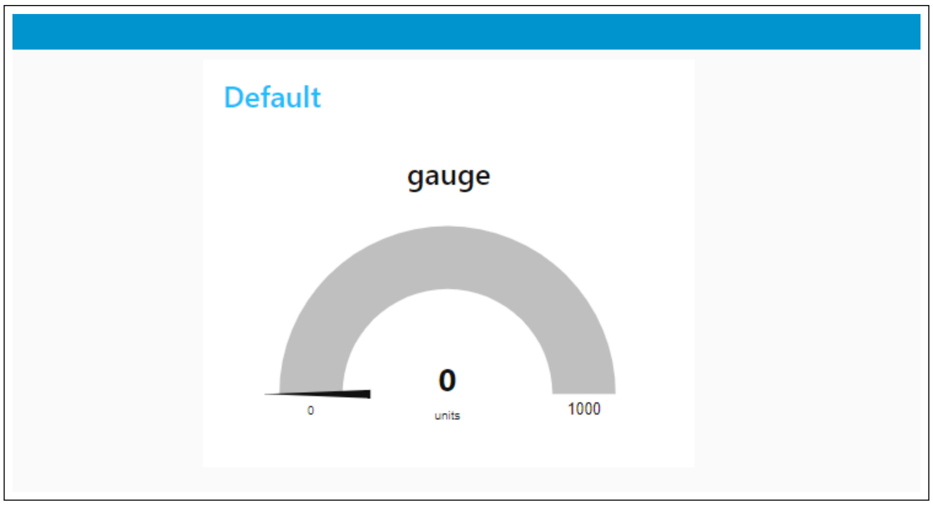
Building the Flow

1. Drag and drop "MQTT In" node from the palette to the flow.
2. Double-click on the node to configure it. Set the topic to "Light" and click "Done."
3. Drag and drop **"Gauge"** node from the dashboard nodes to the flow.
4. Connect the "MQTT In" node to the **"Gauge"** node.
5. Deploy the flow by clicking the "Deploy" button.



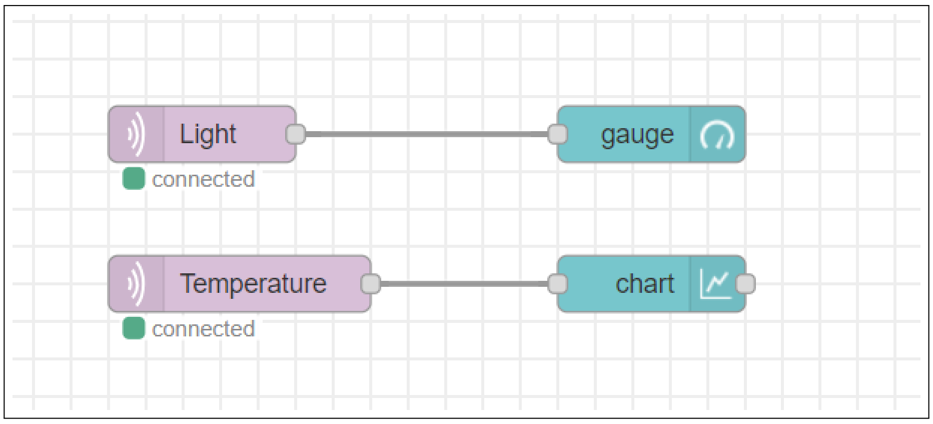
Visualization in Dashboard

1. Open a new tab in your web browser and navigate to `http://localhost:1880/ui`.
2. You should see a Gauge visualization updating with the ”Light” topic data.



Adding a Chart Node

1. Drag and drop another ”MQTT In” node from the palette to the flow.
2. Double-click on the new node to configure it. Set the topic to ”Temperature” and click ”Done.”
3. Drag and drop a ”Chart” node from the dashboard nodes to the flow.
4. Connect the new ”MQTT In” node to the ”Chart” node.
5. Deploy the flow by clicking the ”Deploy” button.



Visualization in Dashboard

1. Open a new tab in your web browser and navigate to `http://localhost:1880/ui`.
2. You should see both a Gauge for ”Light” and a Chart for ”Temperature” updating with data.

2.3. Testing the System

Now, let’s publish data to MQTT topics and observe the visualizations in Node-RED.

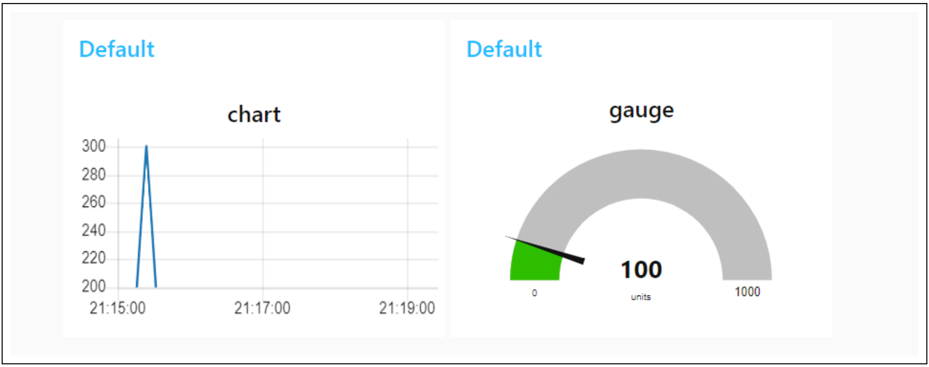
1. Open a new command prompt or terminal window.
2. Use the following command to publish a message to the ”Light” topic:

```

mqtt-cli.exe pub -t Light -h localhost -p 1883 -m 100

```

3. Use the following command to publish to the ”Temperature” topic:

```

    mqtt-cli.exe pub -t Temperature -h localhost -p 1883 -m 200
    mqtt-cli.exe pub -t Temperature -h localhost -p 1883 -m 300
    mqtt-cli.exe pub -t Temperature -h localhost -p 1883 -m 200

```

4. Observe the Node-RED dashboard, and you should see both the Gauge and Chart updating with data.



3. Using Arduino-based client MQTT and a cloud Mosquito server)

In this activity, we will use an Arduino MKR 1010 to collect data from LDR and Force sensors and **publish** the information to the `https://test.mosquitto.org/` Mosquitto server on the cloud. We will use the PC as a client to receive data (Subscriber). To do that follow these steps:

3.1. Hardware Setup

Connect LDR to analog pin A0 and Force sensor to analog pin A1 on the Arduino MKR 1010.

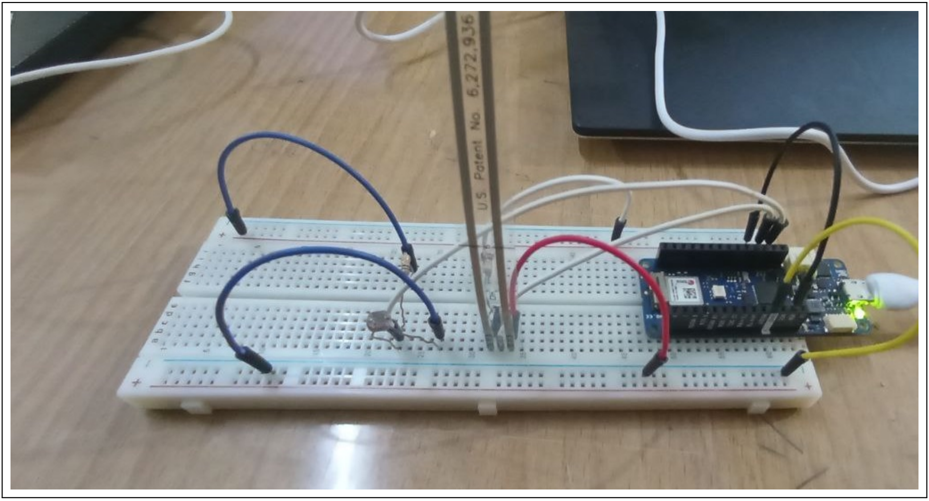


Figure 7: Real view

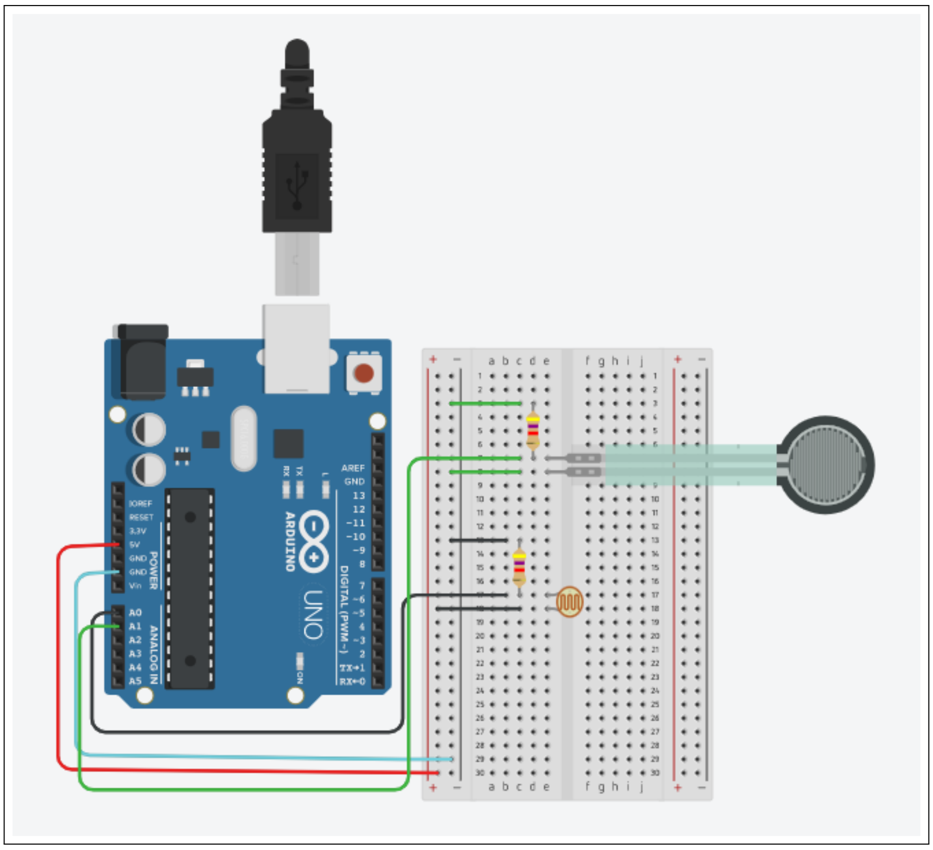


Figure 8: Circuit View

3.2. Arduino Code Setup

In this section, we establish a connection to a WiFi network and connect to an MQTT broker. The Arduino code reads analog values from the LDR and force sensor. Subsequently, it publishes these values to specific MQTT topics, namely "Light" and "Force," at regular intervals. This facilitates the seamless transmission of sensor data over the network.

To observe the results, we attempt to read these values on our PC using the MQTT CLI tool, which has been previously installed.

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

// WiFi credentials
const char* ssid = "YourWiFiSSID";
const char* password = "YourWiFiPassword";

// MQTT broker information
const char* mqtt_server = "test.mosquitto.org";
const int mqtt_port = 1883;

// WiFi and MQTT clients
WiFiClient espClient;
PubSubClient client(espClient);

```

```
// Analog pins for LDR  and force sensor
int ldr_pin = A0;
int force_pin = A1;

// Function to set up WiFi connection
void setup_wifi() {
  delay(10);
  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.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

// Arduino setup function
void setup() {
  Serial.begin(115200);
  pinMode(ldr_pin, INPUT);
  pinMode(force_pin, INPUT);
  setup_wifi();
  client.setServer(mqtt_server, mqtt_port);
}

// Function to reconnect to MQTT broker
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");

    if (client.connect("ArduinoClient")) {
      Serial.println(" connected");
      client.publish("outTopic", "hello world");
      client.subscribe("inTopic");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

// Arduino main loop
void loop() {
  if (!client.connected()) {
    reconnect();
  }

  client.loop();

  // Read analog values from LDR and force sensor
  int ldr_value = analogRead(ldr_pin);
  int force_value = analogRead(force_pin);

  // Publish analog values to MQTT topics
  client.publish("Light", String(ldr_value).c_str());
  client.publish("Force", String(force_value).c_str());

  // Delay for 1 second before the next iteration
  delay(1000);
}
```

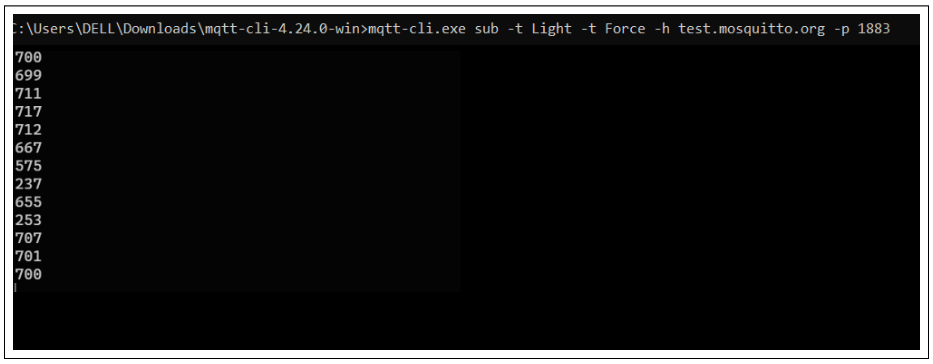
Listing 1: Arduino Code

3.3. Result Observation

Let's subscribe to the topics and observe the results:

```

   mqtt-cli.exe sub -t Light -t Force -h test.mosquitto.org -p 1883
```

You should see the Light and Force topics data being published from the Arduino MKR 1010.



```
:\Users\DELL\Downloads\mqtt-cli-4.24.0-win>mqtt-cli.exe sub -t Light -t Force -h test.mosquitto.org -p 1883
700
699
711
717
712
667
575
237
655
253
707
701
700
```

4. Using a Mosquitto Broker Installed on Raspberry Pi

4.1. Installing Mosquitto Broker on Raspberry Pi

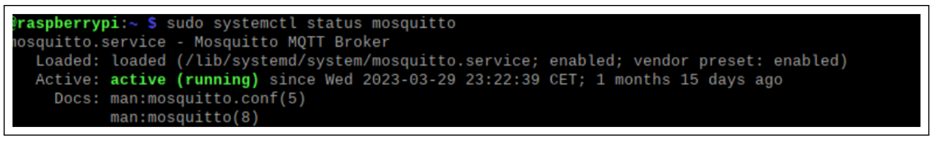
To set up the Mosquitto MQTT broker on your Raspberry Pi, Use this command :

```

   sudo apt install -y mosquitto


```

Mosquitto should start automatically after installation. You can verify its status:

```
sudo systemctl status mosquitto
```

The output should indicate that Mosquitto is active and running.



```
@raspberrypi:~ $ sudo systemctl status mosquitto
mosquitto.service - Mosquitto MQTT Broker
   Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2023-03-29 23:22:39 CET; 1 months 15 days ago
     Docs: man:mosquitto.conf(5)
           man:mosquitto(8)
```

4.2. Hardware Configuration

For the hardware setup, we will maintain the existing circuit configuration as previously detailed.

4.3. Software

In this section, we will utilize the Mosquitto broker installed on our Raspberry Pi. Instead of relying on a cloud-based broker, we will be using the MQTT broker hosted on our local Raspberry Pi. This necessitates a modification in our Arduino code to communicate with the broker hosted on the Raspberry Pi's address. To achieve this, follow these steps:

1. Identify the IP address of your Raspberry Pi on your local network. You can find this information using the following command on the Raspberry Pi terminal:

   ```
   hostname -I
   ```

   The output will display the IP address of your Raspberry Pi.

2. Open the Arduino code, and locate the following lines:

   ```
   const char* mqtt_server = "test.mosquitto.org";
   const int mqtt_port = 1883;
   ```

   Replace "test.mosquitto.org" with the IP address of your Raspberry Pi.

```
const char* mqtt_server = " 192.168.43.204";
const int mqtt_port = 1883;
```

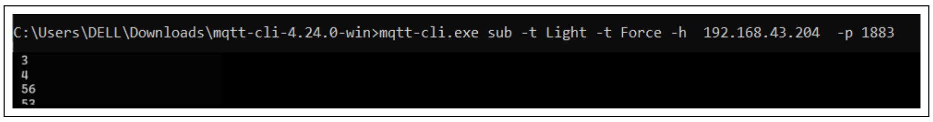
3. Save the changes and upload the modified Arduino code to your device.

4.4. Result Observation

Let's subscribe to the topics and observe the results:

```
mqtt-cli.exe sub -t Light -t Force -h  192.168.43.204 -p 1883
```

You should see the Light and Force topics data being published from the Arduino MKR 1010.



```
C:\Users\DELL\Downloads\mqtt-cli-4.24.0-win>mqtt-cli.exe sub -t Light -t Force -h  192.168.43.204  -p 1883
3
4
56
```

5. Using Node-RED on Raspberry Pi

In Section 2., we familiarized ourselves with Node-RED by installing it and creating flows on our PC. Now, we will replicate the process on our Raspberry Pi, integrating it with the local Mosquitto MQTT broker.

5.1. Installation of Node-RED

Follow these steps to install Node-RED on your Raspberry Pi:

1. Open a command prompt or terminal window on your Raspberry Pi.
2. Execute the following command to install Node-RED:

```
sudo npm install -g --unsafe-perm node-red
```

3. Once the installation is complete, start Node-RED:

```

    node-red


```

4. Open your web browser and navigate to **`http://localhost:1880`** to access the Node-RED editor.

5.2. Creating Our flows

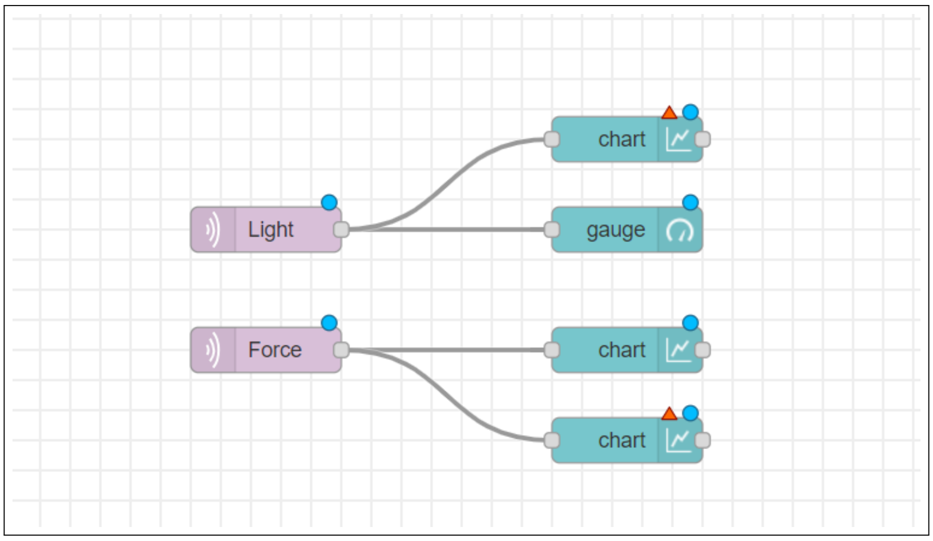
Light Sensor Visualization

1. Drag and drop an ”MQTT In” node onto the Node-RED flow.
2. Configure the node by double-clicking on it, setting the topic to ”Light,” and clicking ”Done.”
3. Add a **”Gauge”** node from the dashboard nodes and connect it to the ”MQTT In” node.
4. Integrate a ”Chart” node into the flow and connect it to the ”MQTT In” node.

Force Sensor Visualization

Follow similar steps for the force sensor visualization:

1. Drag and drop another ”MQTT In” node onto the Node-RED flow.
2. Configure the new node by setting the topic to ”Force.”
3. Add a **”Gauge”** node from the dashboard nodes and connect it to the new ”MQTT In” node.
4. Integrate a ”Chart” node into the flow and connect it to the ”MQTT In” node.



5. Now we deploy the flow.

Viewing the Visualizations

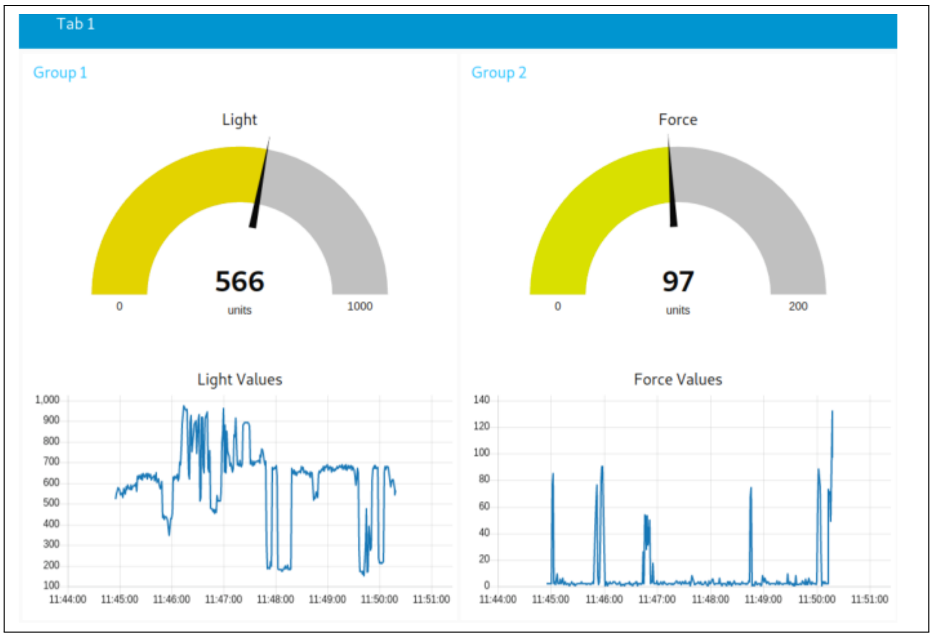


Figure 9: results

6. Controlling LED Intensity Using Node-RED

6.1. Hardware

A led is connected to the Raspberry Pi **GPIO19 pin**.

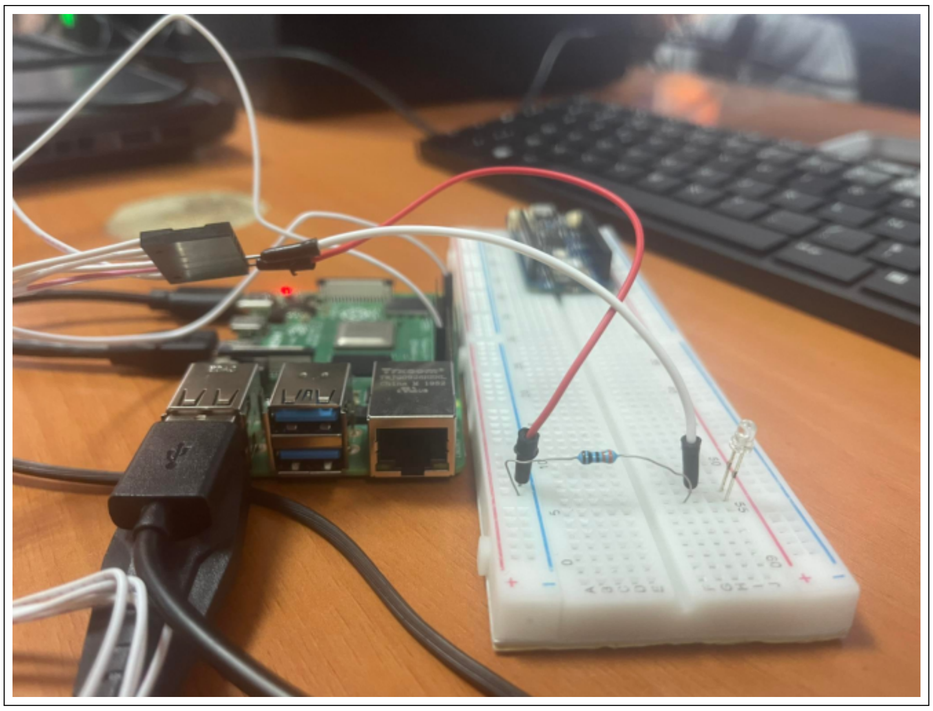
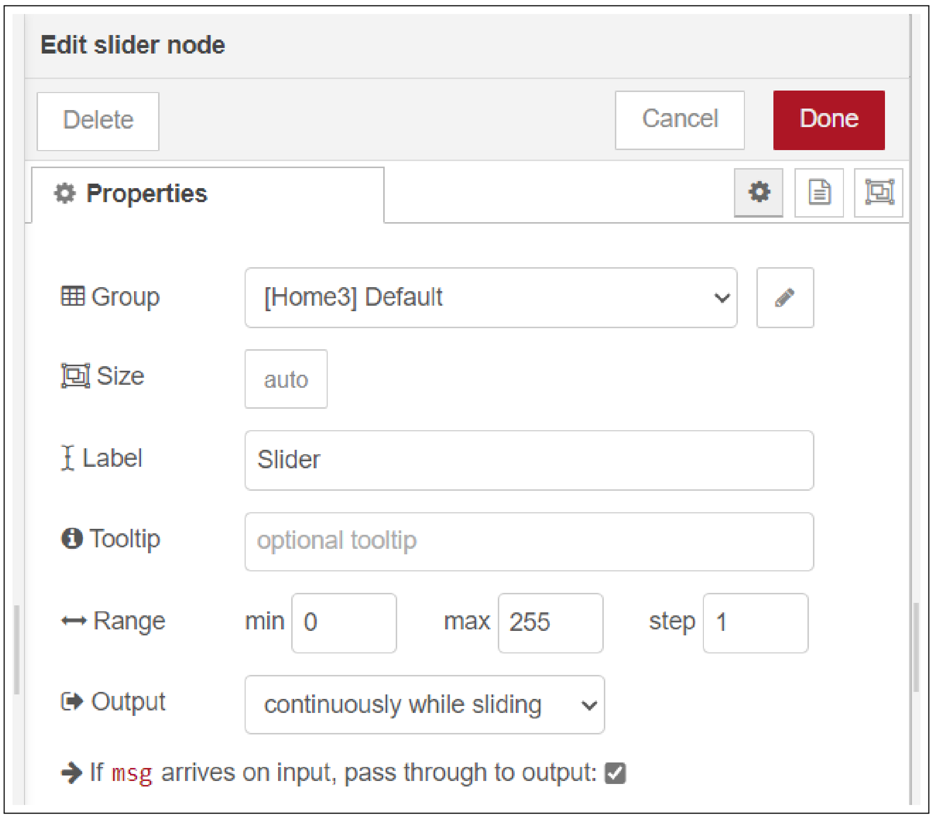


Figure 10: Blinked led connection

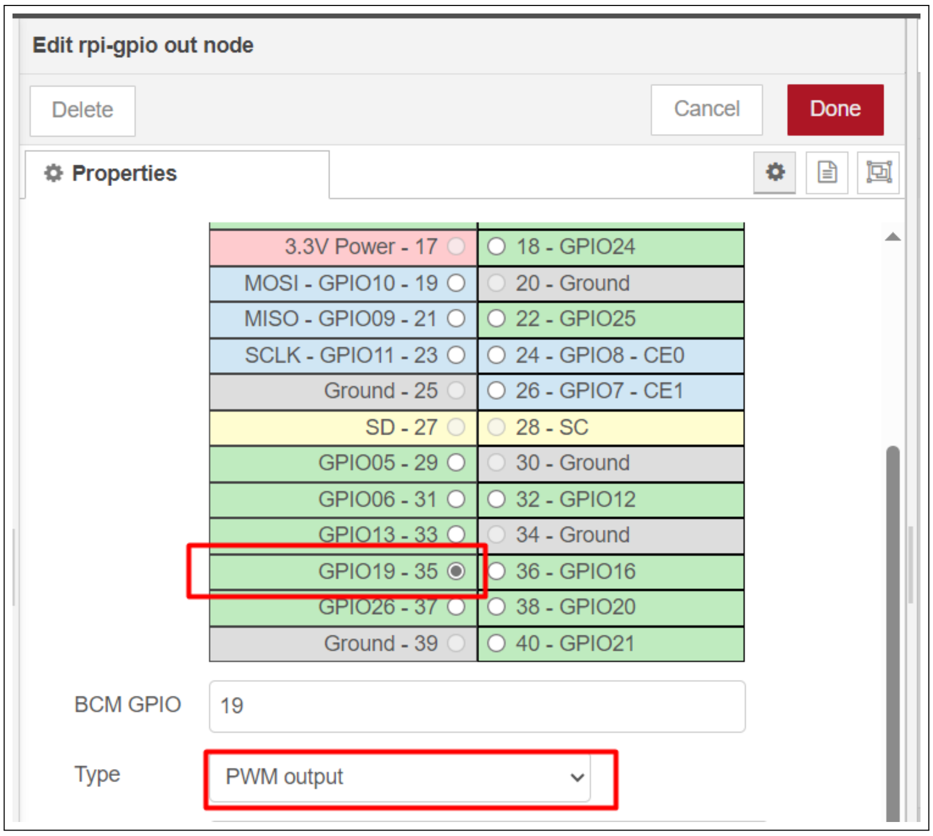
6.2. Building the Node-RED Flow

1. Drag and double-click the **”Slider”** node to configure its properties.


Edit slider node

Group: [Home3] Default
Size: auto
Label: Slider
Tooltip: optional tooltip
Range: min 0 max 255 step 1
Output: continuously while sliding
If msg arrives on input, pass through to output: ✓

2. Drag and drop a **"GPIO Out"** node onto the workspace.
3. Connect the output of the **Slider node** to the input of the **GPIO Out node** and configure its properties.

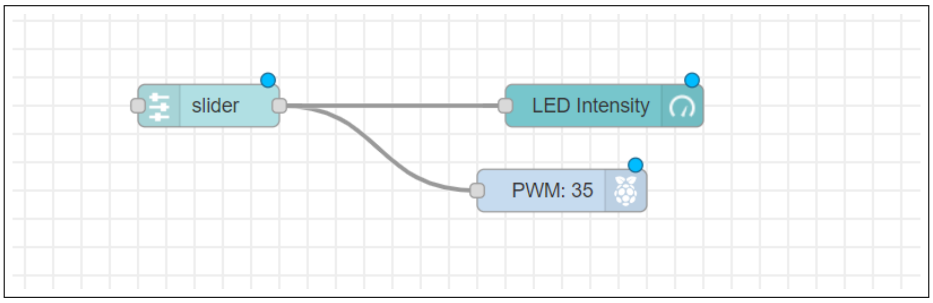


3.3V Power - 17	18 - GPIO24
MOSI - GPIO10 - 19	20 - Ground
MISO - GPIO09 - 21	22 - GPIO25
SCLK - GPIO11 - 23	24 - GPIO8 - CE0
Ground - 25	26 - GPIO7 - CE1
SD - 27	28 - SC
GPIO05 - 29	30 - Ground
GPIO06 - 31	32 - GPIO12
GPIO13 - 33	34 - Ground
[✓ GPIO19 - 35]	36 - GPIO16
GPIO26 - 37	38 - GPIO20
Ground - 39	40 - GPIO21

BCM GPIO: 19

Type: PWM output

4. Drag and drop a **"Gauge"** node onto the workspace and xonnect the output of the **"Slider"** node to the input of the Gauge node.



6.3. Testing the LED Intensity Controller

With the flow control diagram nodes configured, you may enable the project by pressing the Deploy button on the Node-RED editor.

Once deployed, the dimming control dashboard and the interactive widgets will be visible and ready to operate the LED intensity.

The Node-RED gauge will show the LED intensity level based on the **"Slider"** widget's adjusted position.

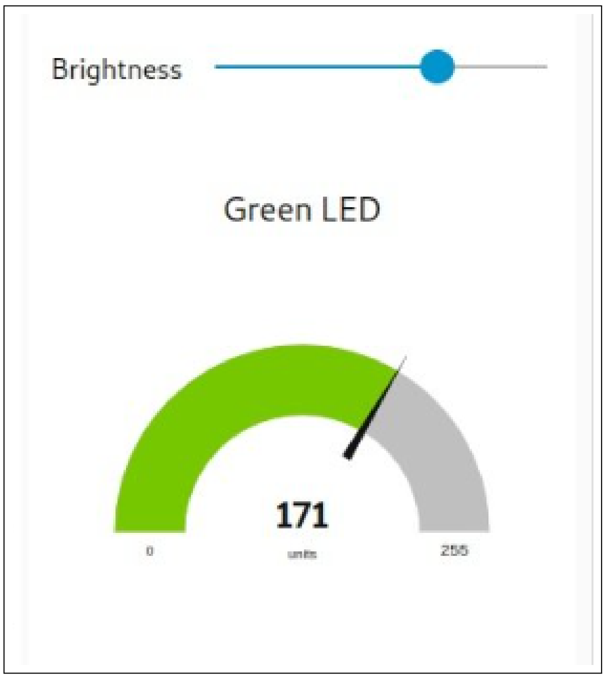


Figure 11: LED Brightness Control.

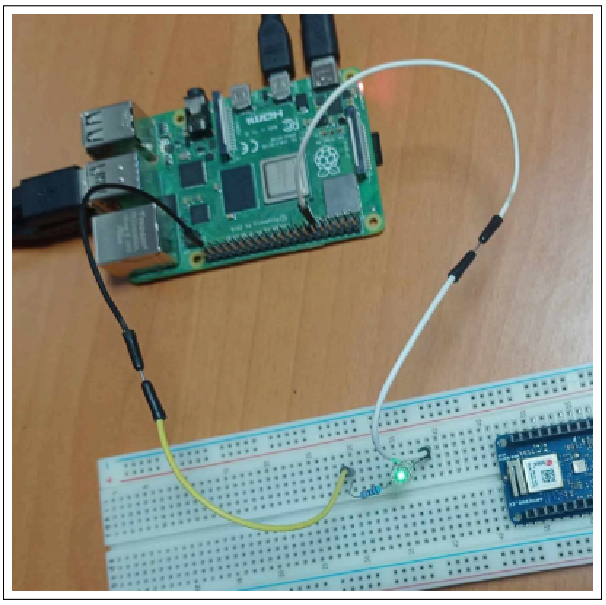


Figure 12: Green LED ON

7. User Management in Node-RED Dashboard

To implement user management in your Node-RED Dashboard, follow the steps below to control access levels, define user credentials, and handle user logins.

7.1. Initial Settings

To establish different levels of access for users in the Node-RED Dashboard, we have defined two user profiles:

User 1 - Light Dashboard Access Only

- Username: `user1`
- Password: `password1`
- Access Level: 0

User 1 is configured with access limited to the **"Light" dashboard,** providing a focused view on light-related data.

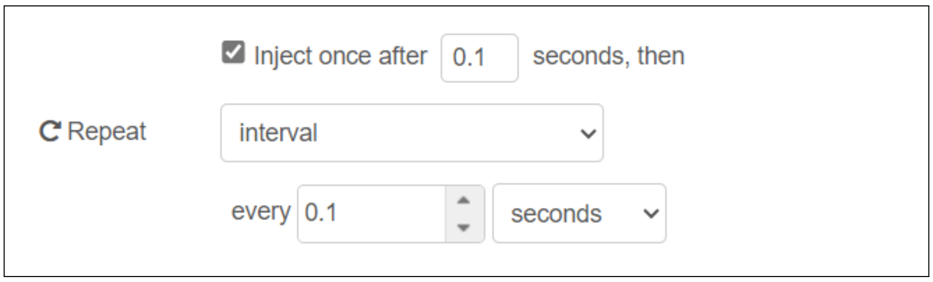
User 2 - Light and Force Dashboard Access

- Username: `user2`
- Password: `password2`
- Access Level: 1

User 2 enjoys broader access, allowing interaction with both the **"Light" and "Force"** dashboards, providing a comprehensive overview of light and force-related data.

7.2. Node-red Flow Configuration

1. Drag an **"Inject"** node onto the Node-RED workspace.
2. Configure the **"Inject"** node to trigger every 0.1 seconds.

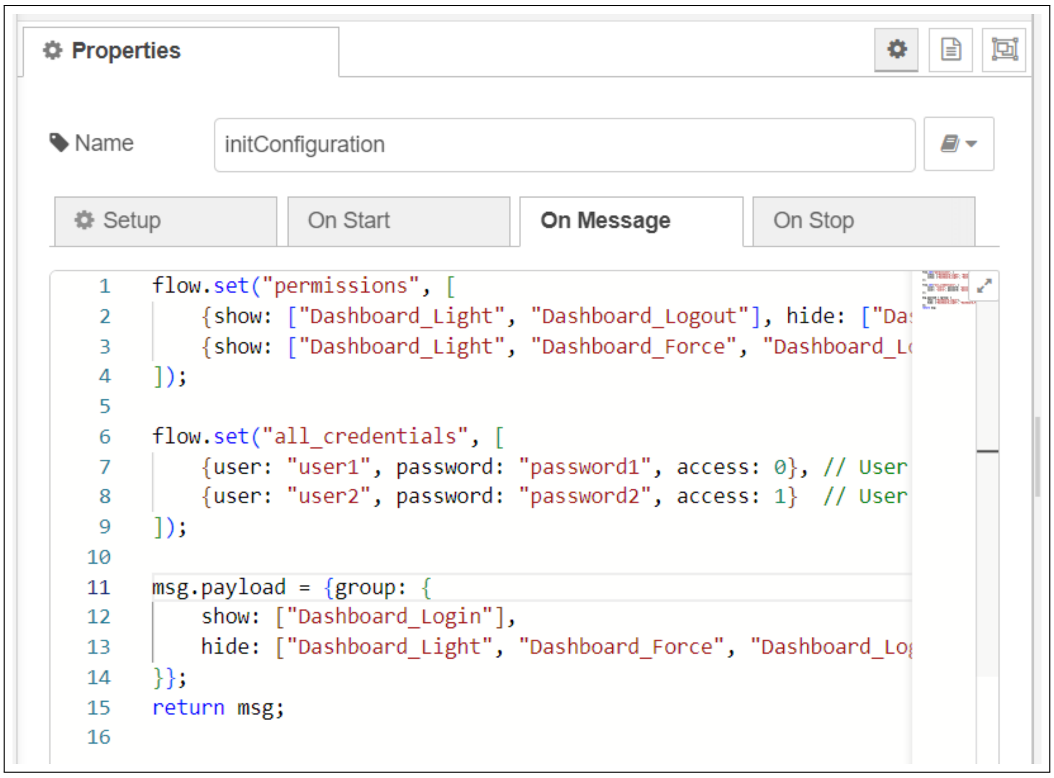


Inject once after 0.1 seconds, then
Repeat: interval
every 0.1 seconds

3. Add a **"Function"** node and set it to initialize the configuration for all users using the following code:

```

    flow.set("permissions", [
        {show: ["Dashboard_Light"], hide: ["Dashboard_Login"]},
        {show: ["Dashboard_Light", "Dashboard_Force], hide: ["
   Dashboard_Login"]}
    ]);

    flow.set("all_credentials", [
        {user: "user1", password: "password1", access: 0}, // User 1
   can access only Light topics
        {user: "user2", password: "password2", access: 1}  // User 2
   can access Light and Force topics
    ]);
    msg.payload = {group: {
        show: ["Dashboard_Login"],
        hide: ["Dashboard_Light", "Dashboard_Force"]
    }};
    return msg;
```

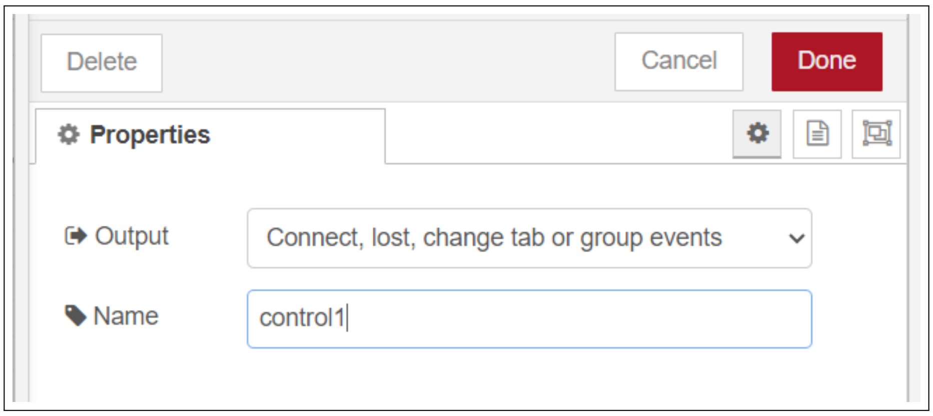
This code initializes variables for user permissions and credentials. The **permissions** array defines which tabs users are allowed to see, while the **all credentials** array stores usernames, passwords, and access levels.



Name: initConfiguration

```
flow.set("permissions", [
    {show: ["Dashboard_Light", "Dashboard_Logout"], hide: ["Da
    {show: ["Dashboard_Light", "Dashboard_Force", "Dashboard_Lo
]);

flow.set("all_credentials", [
    {user: "user1", password: "password1", access: 0}, // User
    {user: "user2", password: "password2", access: 1}  // User
]);

msg.payload = {group: {
    show: ["Dashboard_Login"],
    hide: ["Dashboard_Light", "Dashboard_Force", "Dashboard_Lo
}};
return msg;
```

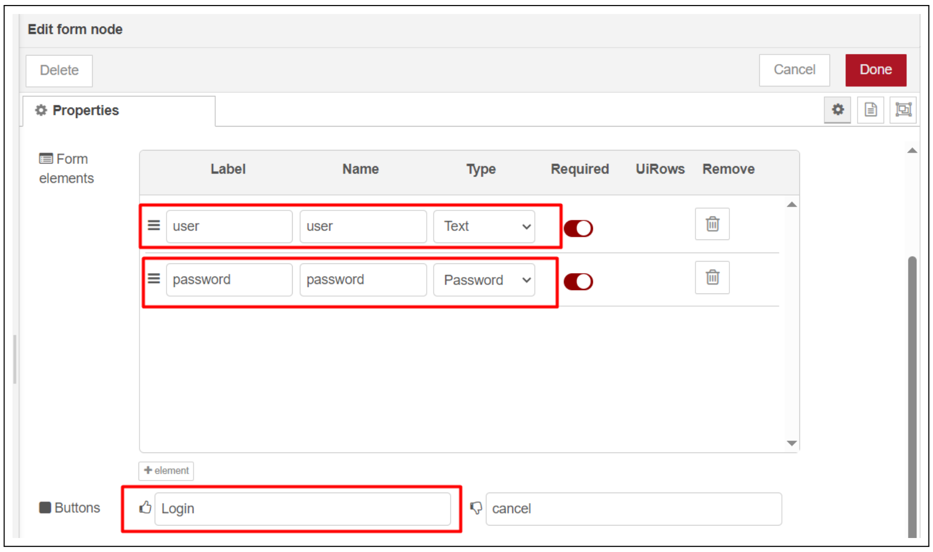
4. Attach a ui control node with the Output: **Connect, lost, change tab or group events**.



Output: Connect, lost, change tab or group events

Name: control1

5. After configuring the system, add a **form node** and set it like this:



Edit form node

Delete | Cancel | Done

Properties

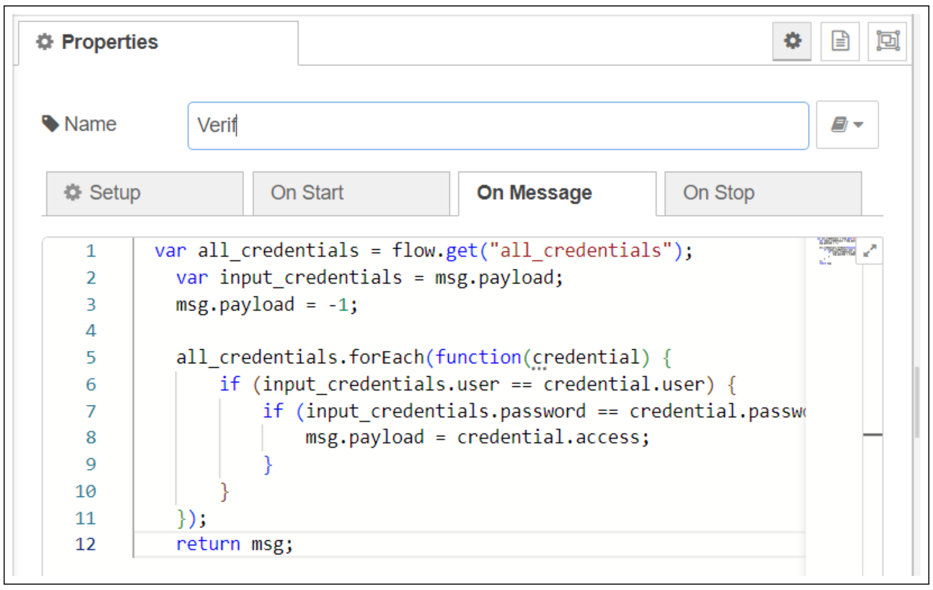
Form elements

Label	Name	Type	Required	UiRows	Remove
user	user	Text	●		🗑
password	password	Password	●		🗑

+ element

Buttons: Login | cancel

6. Add another **"Function"** node to grant access based on user credentials:



Properties

Name: Verif

Setup | On Start | On Message | On Stop

```
var all_credentials = flow.get("all_credentials");
  var input_credentials = msg.payload;
  msg.payload = -1;

  all_credentials.forEach(function(credential) {
      if (input_credentials.user == credential.user) {
          if (input_credentials.password == credential.passw
              msg.payload = credential.access;
          }
      }
  });
  return msg;
```

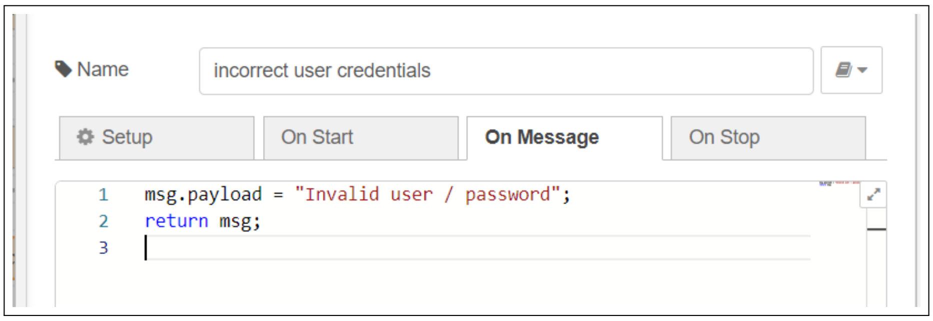
```

    var all_credentials = flow.get("all_credentials");
    var input_credentials = msg.payload;
    msg.payload = -1;

    all_credentials.forEach(function(credential) {
        if (input_credentials.user == credential.user) {
            if (input_credentials.password == credential.password) {
                msg.payload = credential.access;
            }
        }
    });
    return msg;
```

This function node checks **user credentials** against the stored credentials and **sets msg.payload to the user's access level.**

7.3. filter out correct and incorrect users.

Attach a **Switch** node with appropriate settings to filter out correct and incorrect users.

1. In the first output of the "Switch" node, add a "Function" node to handle incorrect user credentials:

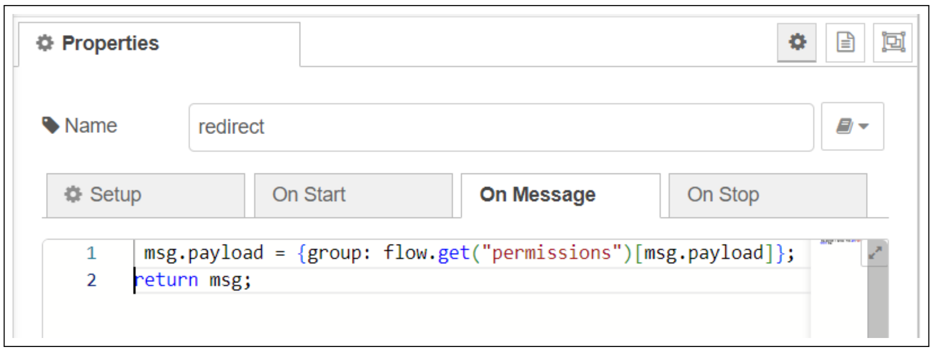


Name: incorrect user credentials

```
msg.payload = "Invalid user / password";
return msg;
```

2. Use a **"Notification"** node to alert the user about incorrect credentials.

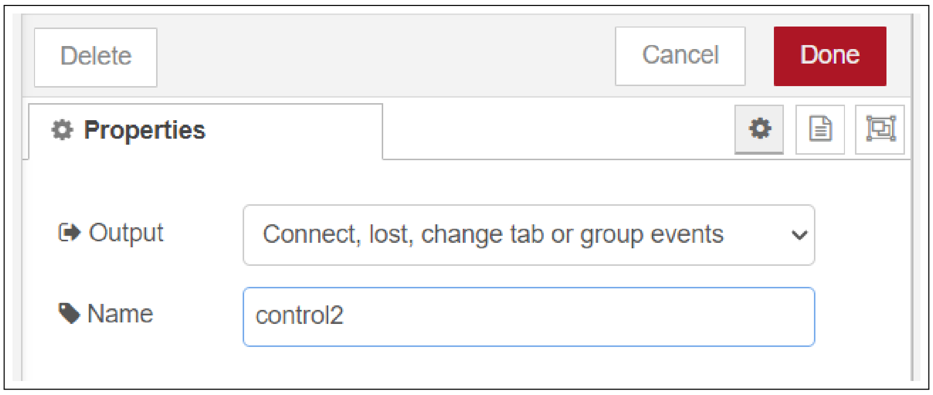
3. In the second output of the **"Switch"** node, add a **"Function"** node to set the user's group based on permissions



Name: redirect

```
msg.payload = {group: flow.get("permissions")[msg.payload]};
return msg;
```

4. Attach a **"ui control"** node to the last function node with "Connect, lost, change tabs or group events" as the output.

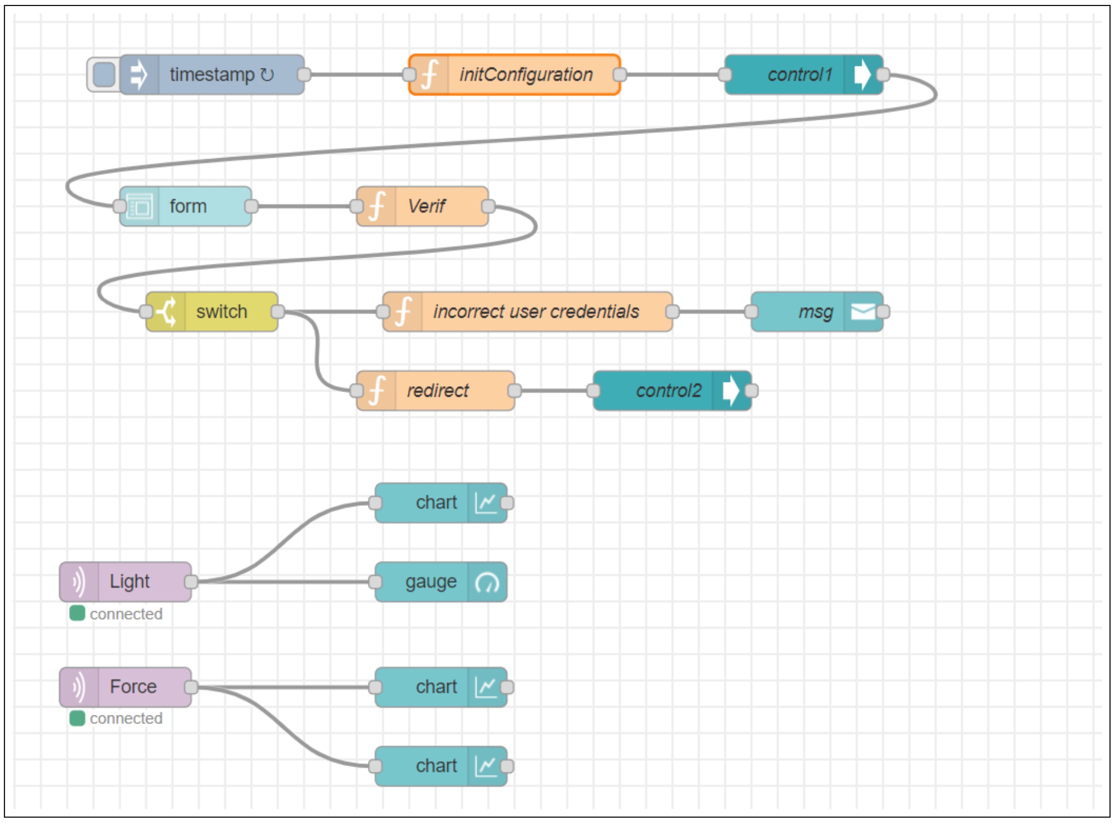


Output: Connect, lost, change tab or group events

Name: control2

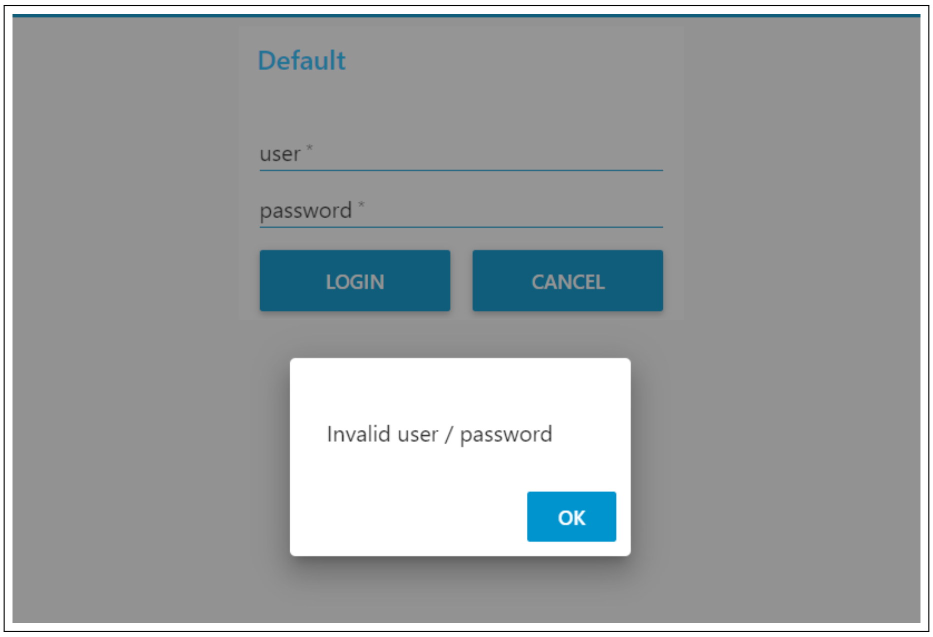
7.4. Full node-red flow

Here's our final flow .



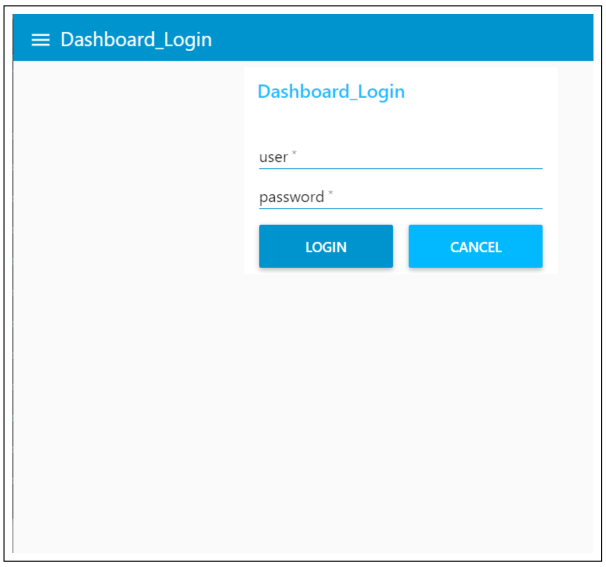
7.5. Testing the System

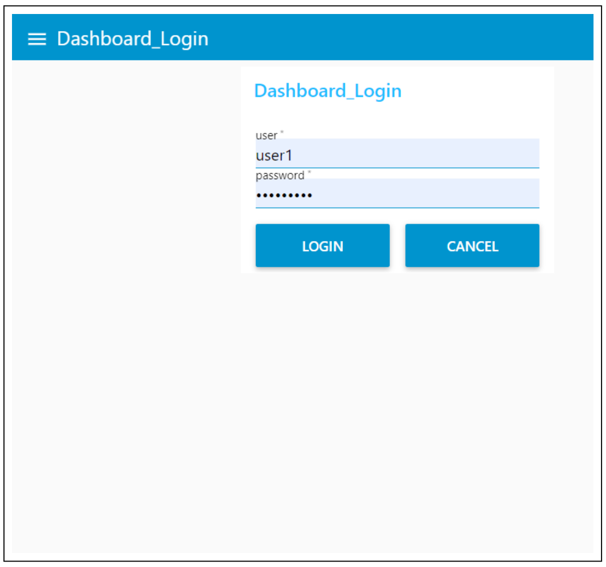
Invalid Credentials

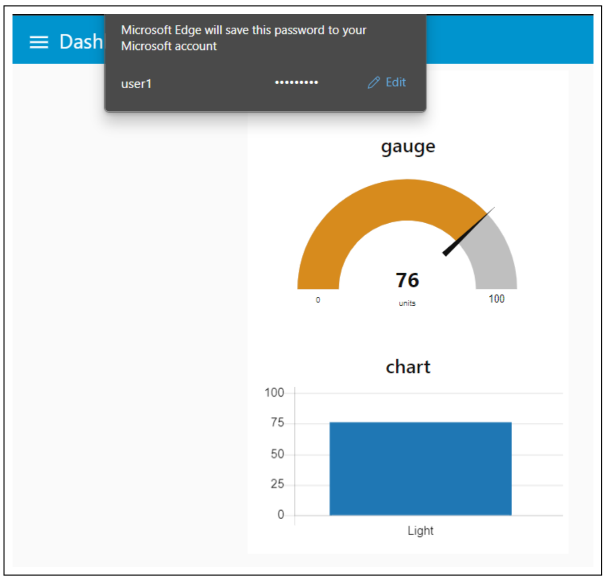


Invalid user / password

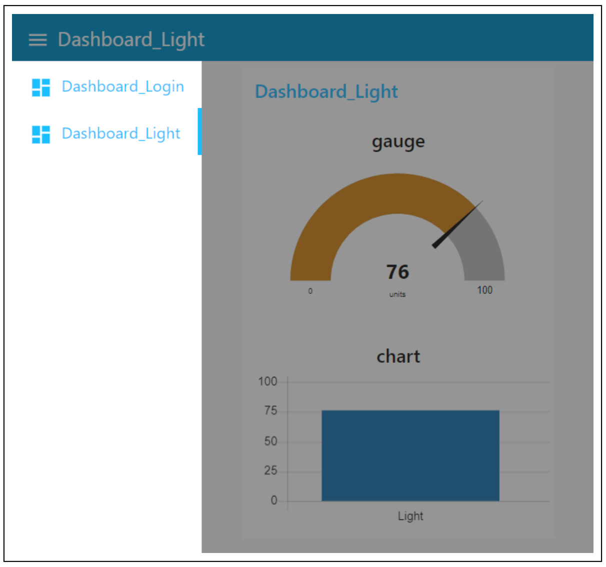
7.6. User1



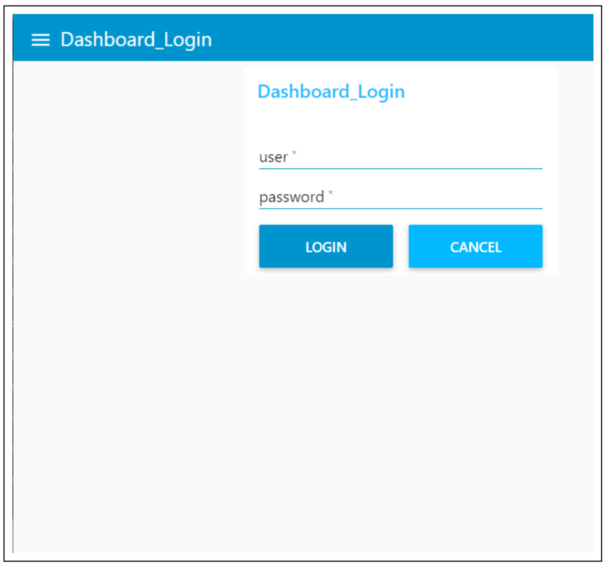


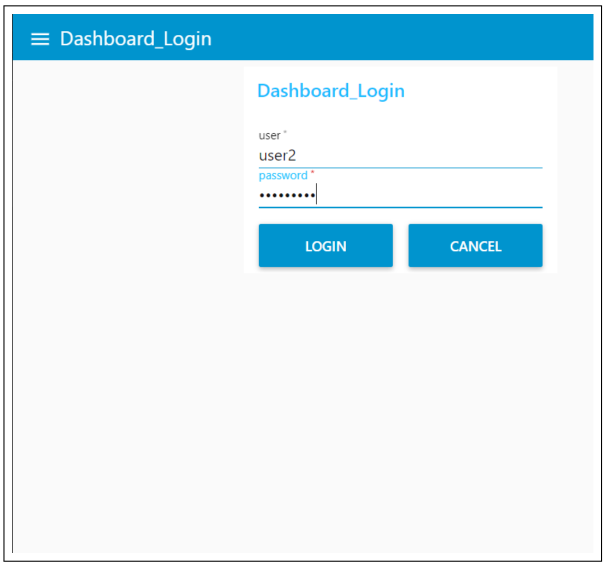

Microsoft Edge will save this password to your Microsoft account

user1 ••••••••• Edit

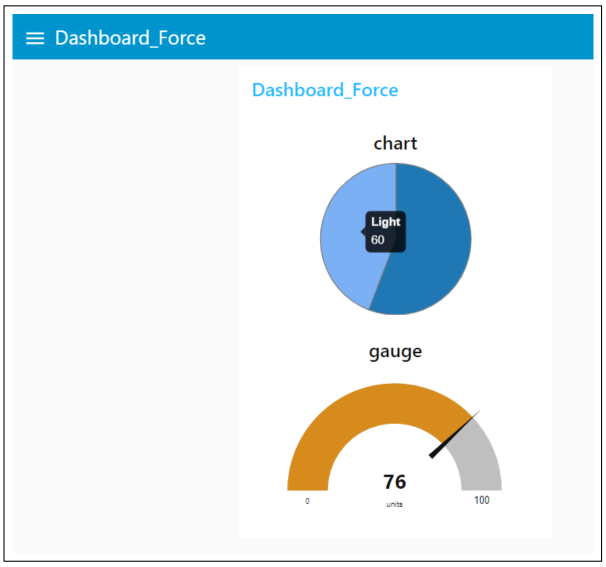


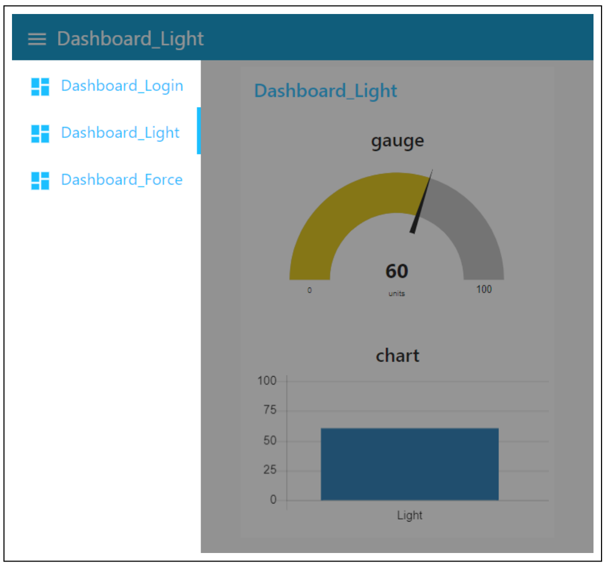
7.7. User 2





user2





C. Conclusion

In conclusion, this comprehensive lab journey delved into the realms of MQTT, Node-RED, and Arduino, offering a holistic understanding of their roles in IoT development. MQTT, with its lightweight messaging protocol, emerged as a powerful tool facilitating efficient communication between devices and servers in IoT networks. The integration of Node-RED, a visual programming tool, simplified the creation of IoT applications, seamlessly connecting data flows and enabling complex automation workflows.

The exploration extended to practical applications, such as LED intensity control, where Node-RED showcased its versatility in implementing real-world functionalities. The utilization of MQTT for sensor data transmission, both locally with a Mosquitto broker on a Raspberry Pi and on a cloud-based platform, demonstrated the flexibility of these technologies.

Furthermore, user management in Node-RED added an extra layer of security, ensuring that different users had access to specific dashboards based on their credentials. This not only enhances security but also tailors the user experience.

In summary, the amalgamation of MQTT, Node-RED, and Arduino provides a robust framework for developing and deploying diverse IoT applications.