Task 1: Installing Node-RED on a Raspberry Pi through Remote Login.

he **Raspberry Pi** is a low-cost, credit-card-sized **single-board computer (SBC)** developed in the UK to promote coding and DIY computing.

Key Features:
✓ **Runs Linux** (Raspberry Pi OS, Ubuntu, etc.)
✓ **GPIO Pins** (For electronics & IoT projects)
✓ **Versatile Uses**:

- Home automation (Node-RED, Home Assistant)
- Media center (Kodi, Plex)
- Web server / VPN
- Robotics & AI (TensorFlow, OpenCV)
- Learning programming (Python, Scratch)

Popular Models:

- **Raspberry Pi 5** (Latest, fastest)
- **Raspberry Pi 4** (Still widely used)
- **Raspberry Pi Zero 2 W** (Ultra-cheap, WiFi-enabled)

@ Draw raspberry pi diagram and Pinout

Connect to Your Raspberry Pi via SSH

If you're using a Linux/macOS terminal or Windows (PowerShell/WSL), use:

bash

Copy

ssh pi@<raspberry_pi_ip_address>

Replace <raspberry_pi_ip_address> with your Pi's actual IP.
(Default password: raspberry unless changed.)

Update System Packages

Before installing Node-RED, ensure your system is up to date:

sudo apt update && sudo apt upgrade -y

Install Node-RED

The recommended method is using the official script:

bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)

This installs:

- Node.js (if not present)
- Node-RED and its dependencies

Optional: Auto-start Node-RED on Boot

To run Node-RED as a service:

sudo systemctl enable nodered.service

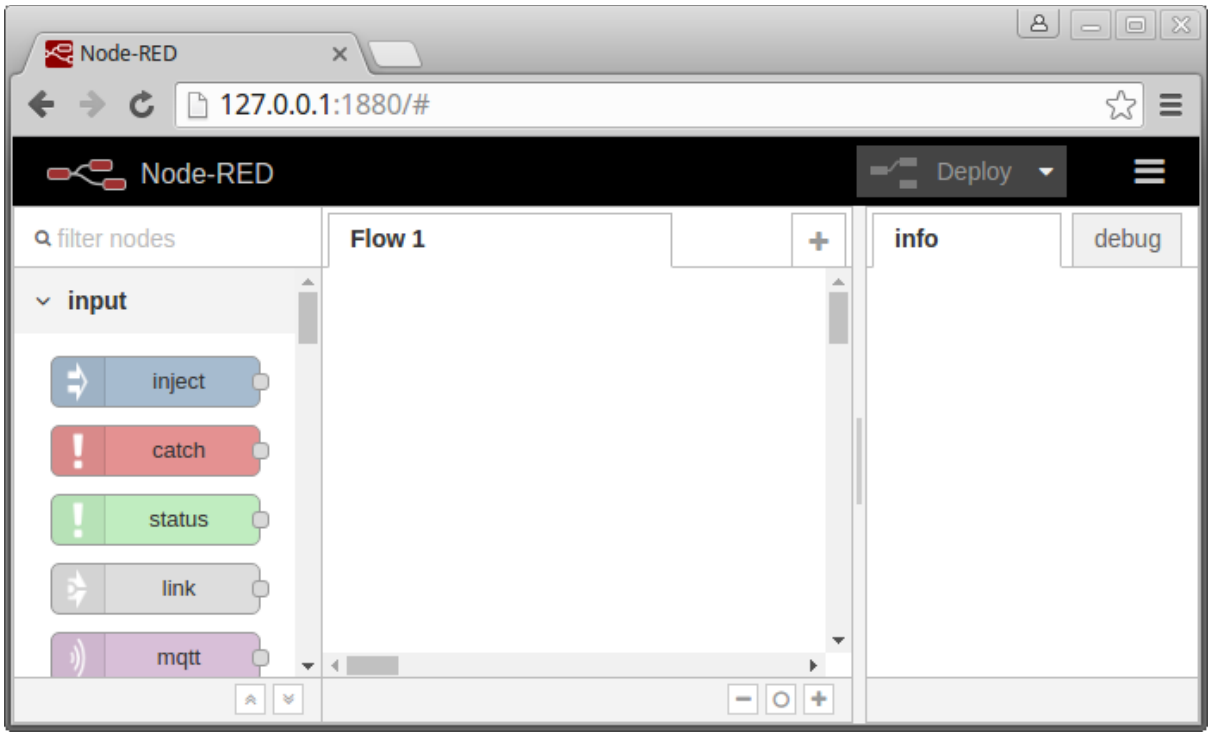
Start it immediately:

sudo systemctl start nodered.service

Access Node-RED Dashboard

After installation, Node-RED runs on port 1880. Open a browser and navigate to:

http://<raspberry_pi_ip_address>:1880



Task 2: Create a simple Node-RED flow that takes input from an inject node and displays output in a debug node. Add a function node to modify the payload in the flow.

To create a simple Node-RED flow that takes input from an Inject node, modifies it with a Function node, and displays the output in a Debug node, follow these steps:

1. **Access the Node-RED Editor**: Open your web browser and navigate to the Node-RED editor, typically at http://localhost:1880 if running locally.
2. **Add Nodes to the Flow**:
    - **Inject Node**: Drag an Inject node from the palette onto the workspace. This node will allow you to manually trigger the flow.
    - **Function Node**: Drag a Function node onto the workspace. This node will be used to modify the payload.
    - **Debug Node**: Drag a Debug node onto the workspace. This will display the output in the debug sidebar.
3. **Connect the Nodes**:
    - Wire the Inject node to the Function node.
    - Wire the Function node to the Debug node.
4. **Configure the Inject Node**:
    - Double-click on the Inject node to open its properties.
    - Set the payload type (e.g., a string or number) that you want to inject. For example, you can set it to a static string like "Hello, World!" or leave it as a timestamp.
5. **Configure the Function Node**:
    - Double-click on the Function node to open its edit dialog.
    - Enter JavaScript code to modify the payload. For example, if you want to append some text to a string, you can use:

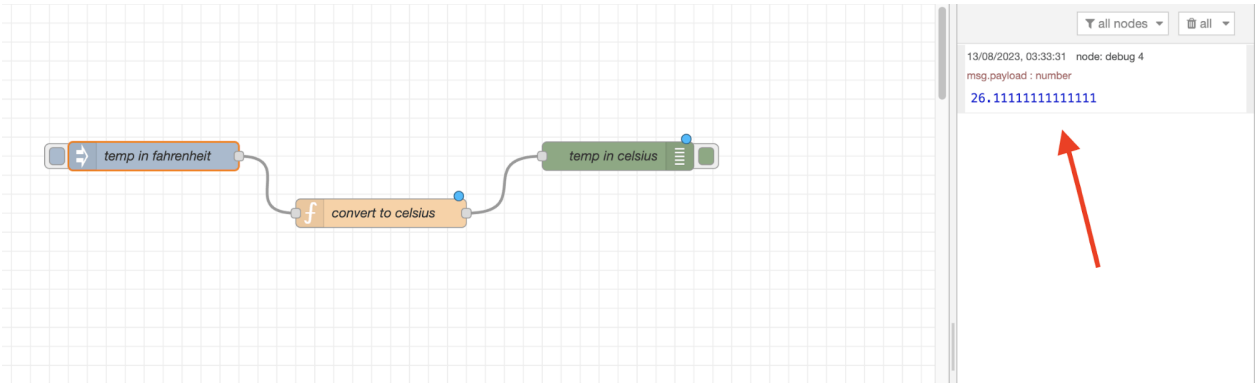
Javascript in function node:

```
// Get the input temperature in Fahrenheit from msg.payload
    var fahrenheit = msg.payload;
// Convert Fahrenheit to Celsius
    var celsius = (fahrenheit - 32) * (5 / 9);
```

```
// Store the result in msg.payload
msg.payload = celsius;
// Return the modified message
return msg;
```

Click "Done" to save your changes.

6. **Deploy the Flow**: Click on the "Deploy" button in the top right corner of the editor to save and activate your flow.
7. **Test Your Flow**:
   - With the Debug sidebar open, click on the button of the Inject node to trigger it.
   - Observe the output in the Debug sidebar; it should show your modified payload.



Task 3: Integrate an MQTT node into a Node-RED flow and subscribe to a topic.

1. Install and Configure Node-RED

- Ensure Node-RED is installed on your system. If not, install it using npm install -g --unsafe-perm node-red or Docker (docker run -it -p 1880:1880 --name mynodered nodered/node-red)
- .
- Open the Node-RED editor in your browser (e.g., http://localhost:1880).

2. Add MQTT Nodes

- Drag an **MQTT In** node from the palette onto the workspace. This node will subscribe to a topic.
- Drag a **Debug** node onto the workspace to display messages in the debug sidebar.

3. Configure the MQTT Broker

- Double-click the **MQTT In** node to open its configuration.
- Click the pencil icon next to the **Server** field to add a new MQTT broker.
    - Enter the broker address (e.g., mqtt://broker.hivemq.com) and port (default is 1883).
    - Add credentials if required (username and password).
    - Set QoS (Quality of Service) level as needed (e.g., 0, 1, or 2)
- Save the broker configuration.

4. Subscribe to a Topic

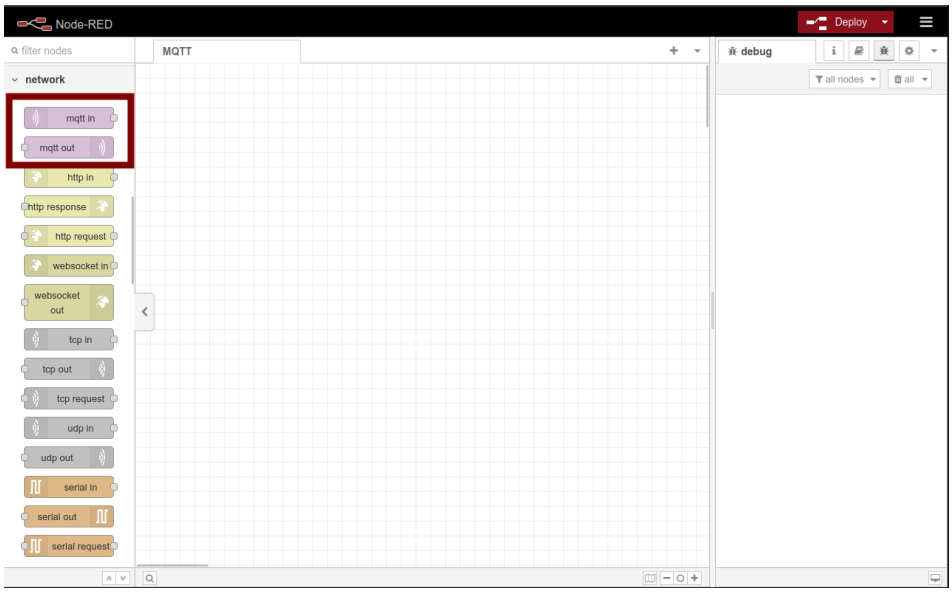
- In the **MQTT In** node, specify the topic you want to subscribe to (e.g., sensor/temperature).
- Save the configuration.

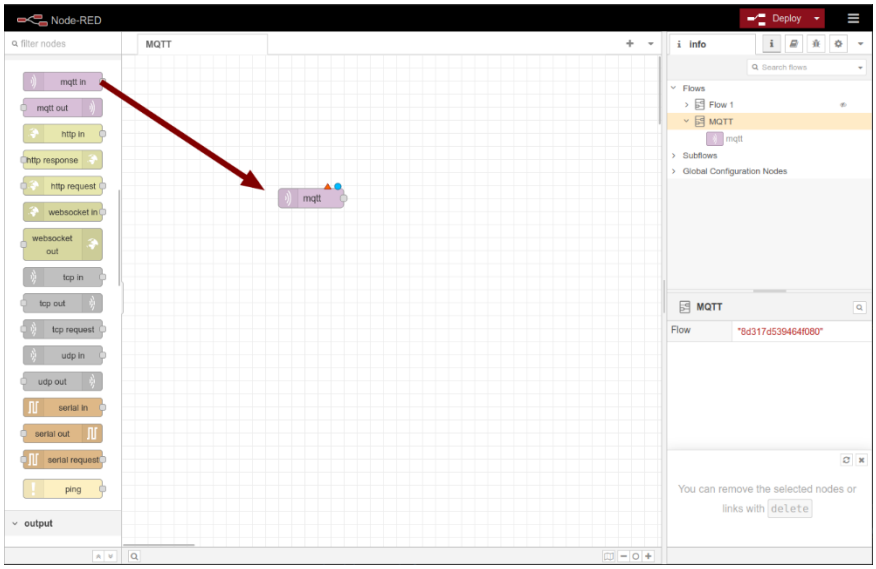
5. Connect Nodes

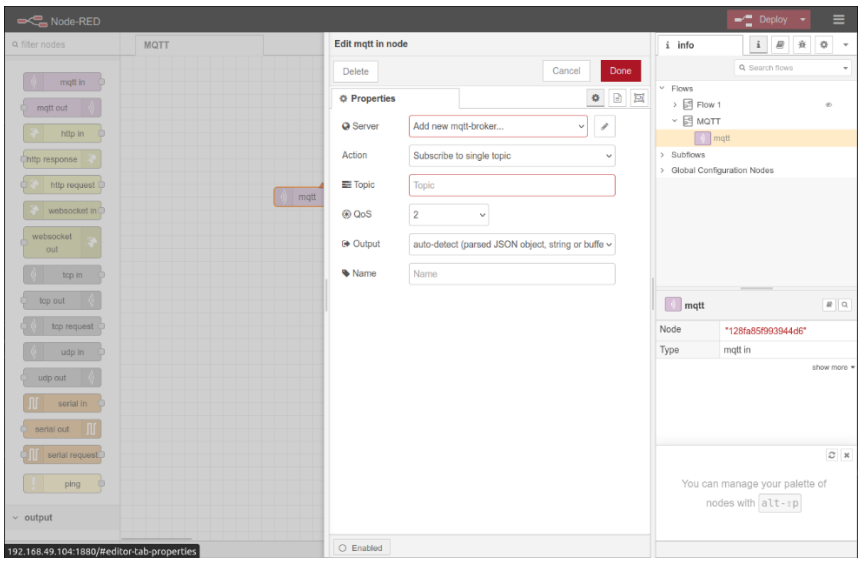
- Wire the **MQTT In** node to the Debug node. Optionally, add a Function node between them for data processing.

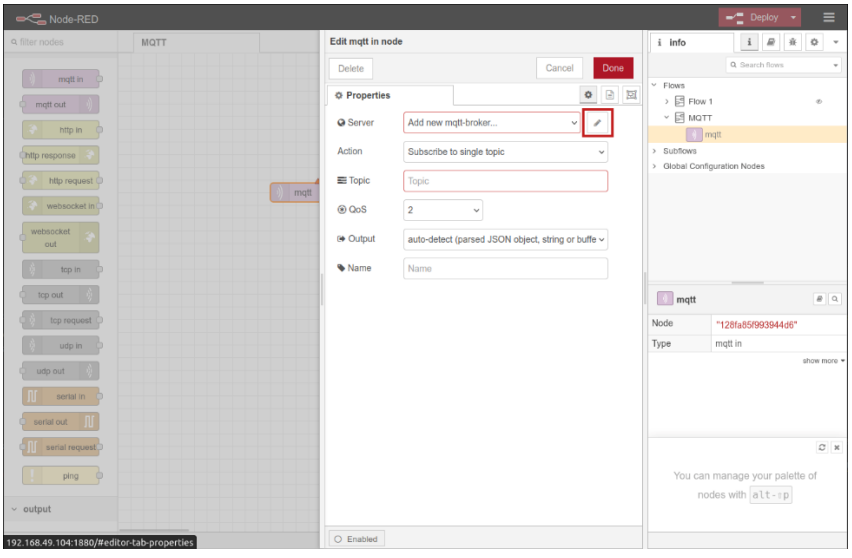
6. Deploy and Test

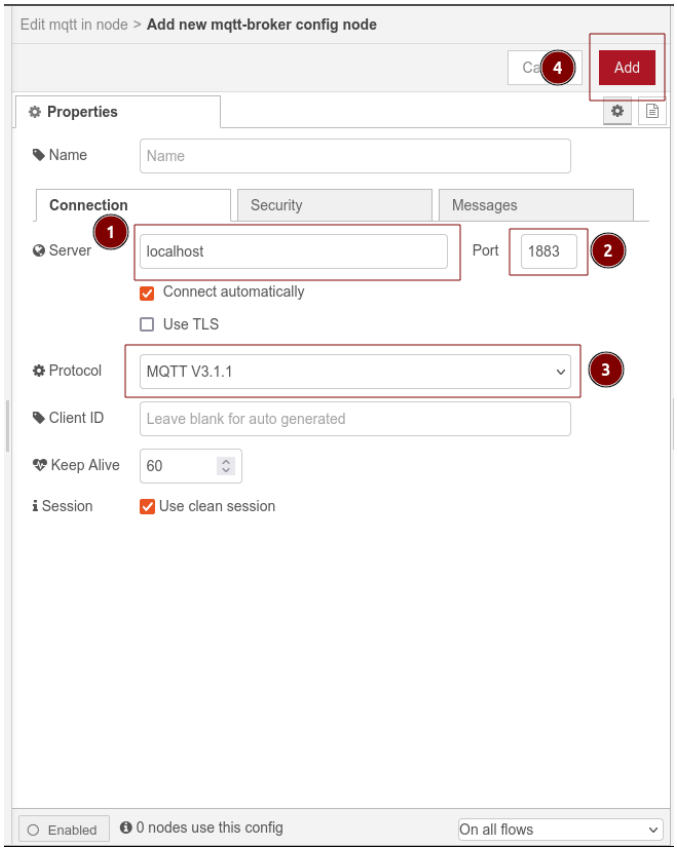
- Click "Deploy" in the top-right corner of Node-RED.
- Publish a message to the subscribed topic using an MQTT client or tool like MQTTX or mosquitto_pub.
- Check the Debug sidebar for incoming messages

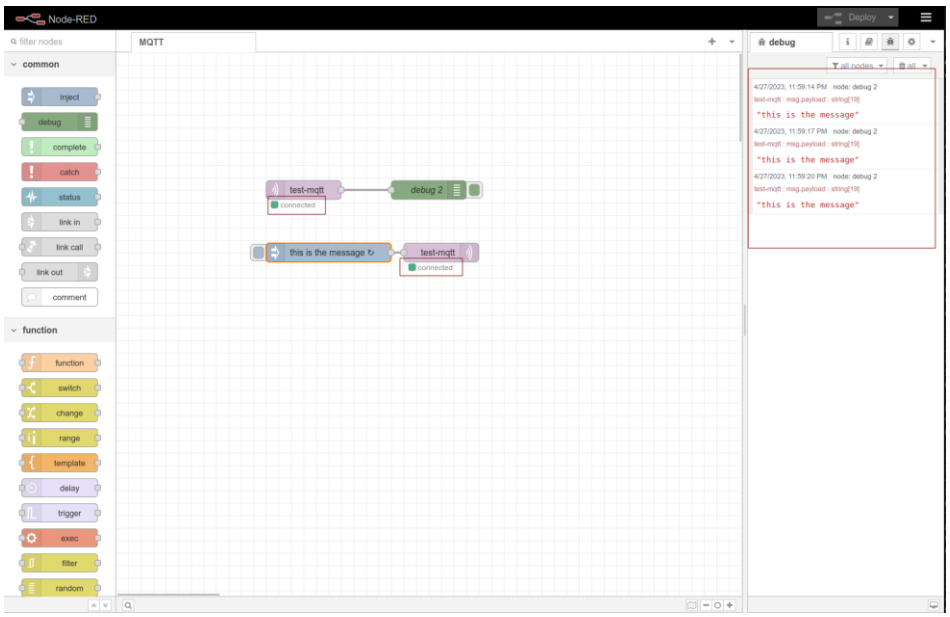



Node-RED
Deploy
filter nodes
MQTT
mqtt in
mqtt out
http in
http response
http request
websocket in
websocket out
tcp in
tcp out
tcp request
udp in
udp out
serial in
serial out
serial request
ping
output
mqtt
info
Flows
Flow 1
MQTT
mqtt
Subflows
Global Configuration Nodes
MQTT
Flow
"8d317d539464f080"
You can remove the selected nodes or links with delete


Node-RED
Deploy
filter nodes
MQTT
Edit mqtt in node
Delete
Cancel
Done
Properties
Server
Add new mqtt-broker...
Action
Subscribe to single topic
Topic
Topic
QoS
2
Output
auto-detect (parsed JSON object, string or buffe
Name
Name
Enabled
info
Flows
Flow 1
MQTT
mqtt
Subflows
Global Configuration Nodes
mqtt
Node
"128fa85f993944d6"
Type
mqtt in
show more
You can manage your palette of nodes with alt-p
192.168.49.104:1880/#editor-tab-properties


Node-RED
Deploy
filter nodes
MQTT
Edit mqtt in node
Delete
Cancel
Done
Properties
Server
Add new mqtt-broker...
Action
Subscribe to single topic
Topic
Topic
QoS
2
Output
auto-detect (parsed JSON object, string or buffe
Name
Name
Enabled
info
Flows
Flow 1
MQTT
mqtt
Subflows
Global Configuration Nodes
mqtt
Node
"128fa85f993944d6"
Type
mqtt in
show more
You can manage your palette of nodes with alt-p
192.168.49.104:1880/#editor-tab-properties





Task 4: Implement an HTTP request node to interact with an external API.

@write lab task where you got time from vclock.tab draw the flow and describe the changes you made in each component

Task 5: Build a basic IoT dashboard using the Node-RED dashboard nodes.

To build a basic IoT dashboard using Node-RED Dashboard nodes, follow these steps:

Step 1: Install the Node-RED Dashboard Module

1. Open the Node-RED editor in your browser (e.g., http://localhost:1880).
2. Go to the **Menu** (top-right corner) > **Manage Palette** > **Install**.
3. Search for node-red-dashboard and click **Install**.
4. Once installed, dashboard nodes (e.g., buttons, sliders, charts) will appear in the palette.

Step 2: Create a Dashboard Layout

1. Click on the **Dashboard** tab (top-right corner of the Node-RED editor).
2. Create a new **Tab** (a page in the dashboard):
    - Click the +tab button.
    - Name it (e.g., "IoT Dashboard").
3. Create a **Group** within the tab:
    - Click the +group button.
    - Name it (e.g., "Sensor Data").

Step 3: Build the Flow

1. Drag and drop the following nodes onto the workspace:
    - **Inject Node**: To simulate data input.
    - **Function Node**: To process or generate dynamic data.
    - **UI Gauge Node**: To display sensor data on the dashboard.
    - **UI Chart Node**: To visualize data trends over time.
2. Configure each node:
    - **Inject Node**:
        - Set it to inject a random number as payload (e.g., between 0 and 100).

Javascript

```
msg.payload = Math.floor(Math.random() * 101); // Random number between 0-100
return msg;
```

 Function Node:

- Add logic to process or format data if needed.

 UI Gauge Node:

- Double-click to configure.
- Assign it to the created Tab/Group (e.g., "IoT Dashboard > Sensor Data").
- Set properties like "Label" (e.g., "Temperature"), range (e.g., 0–100), and units (e.g., °C).

 UI Chart Node:

- Assign it to the same Tab/Group.
- Configure it to display a time-series chart.

To build a basic IoT dashboard using Node-RED Dashboard nodes, follow these steps:

Step 1: Install the Node-RED Dashboard Module

1. Open the Node-RED editor in your browser (e.g., http://localhost:1880).
2. Go to the **Menu** (top-right corner) > **Manage Palette** > **Install**.
3. Search for node-red-dashboard and click **Install**.
4. Once installed, dashboard nodes (e.g., buttons, sliders, charts) will appear in the palette.

Step 2: Create a Dashboard Layout

1. Click on the **Dashboard** tab (top-right corner of the Node-RED editor).
2. Create a new **Tab** (a page in the dashboard):
    - Click the +tab button.
    - Name it (e.g., "IoT Dashboard").

3. Create a **Group** within the tab:
    - Click the +group button.
    - Name it (e.g., "Sensor Data").

Step 3: Build the Flow

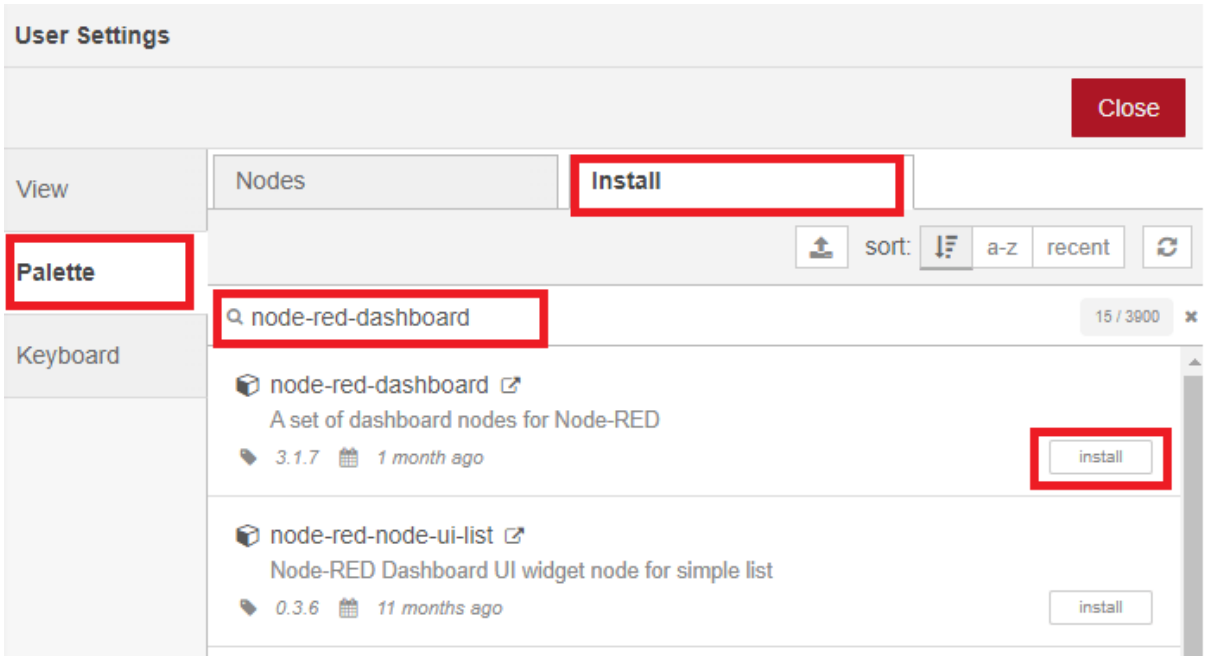
1. Drag and drop the following nodes onto the workspace:
    - **Inject Node**: To simulate data input.
    - **Function Node**: To process or generate dynamic data.
    - **UI Gauge Node**: To display sensor data on the dashboard.
    - **UI Chart Node**: To visualize data trends over time.
2. Configure each node:
    - **Inject Node**:
        - Set it to inject a random number as payload (e.g., between 0 and 100).

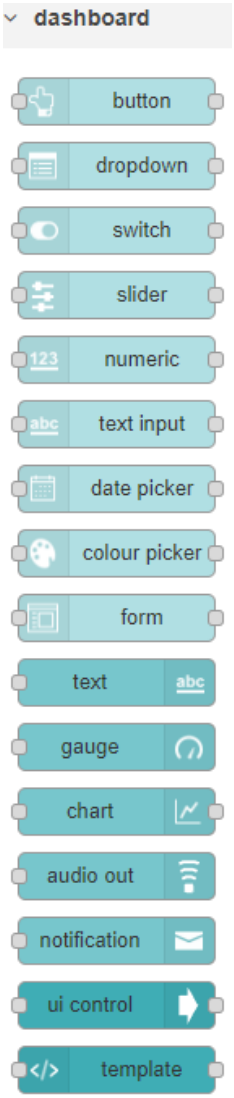
javascript

msg.payload = Math.floor(Math.random() * 101); *// Random number between 0-100*

return msg;

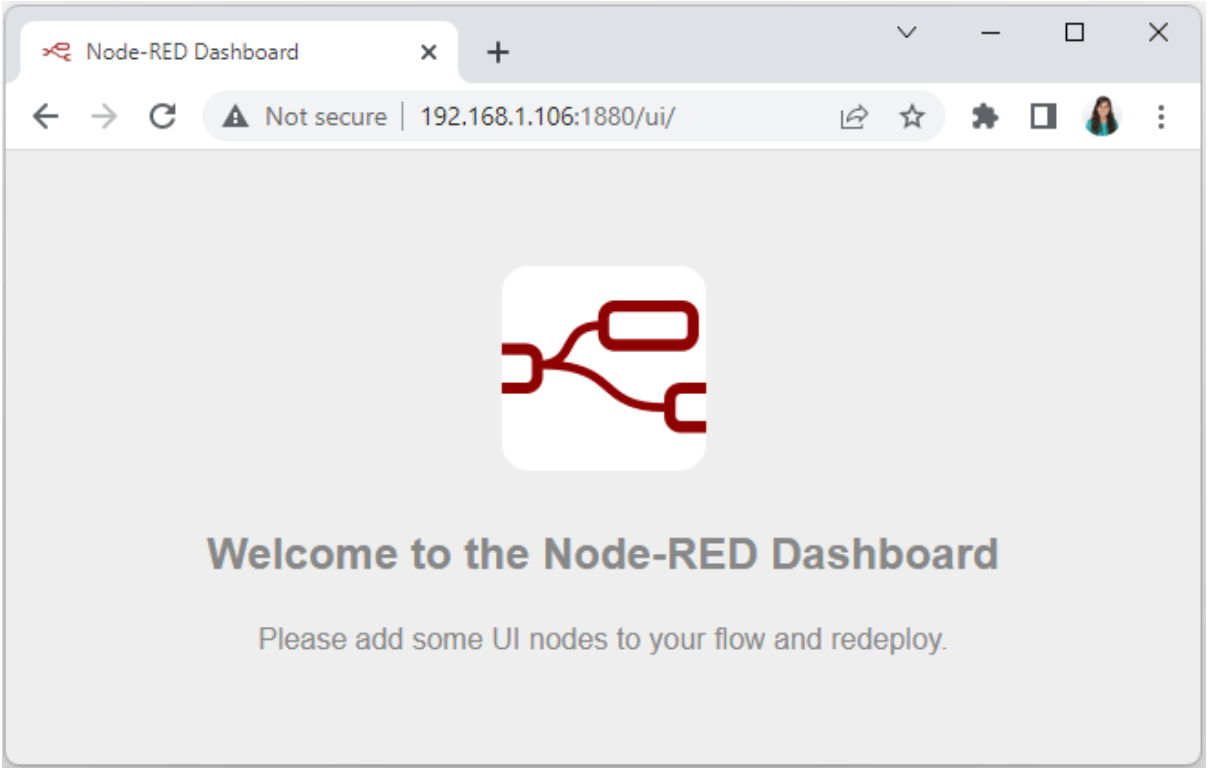
- **Function Node**:
    - Add logic to process or format data if needed.
- **UI Gauge Node**:
    - Double-click to configure.
    - Assign it to the created Tab/Group (e.g., "IoT Dashboard > Sensor Data").
    - Set properties like "Label" (e.g., "Temperature"), range (e.g., 0–100), and units (e.g., °C).
- **UI Chart Node**:
    - Assign it to the same Tab/Group.
    - Configure it to display a time-series chart.

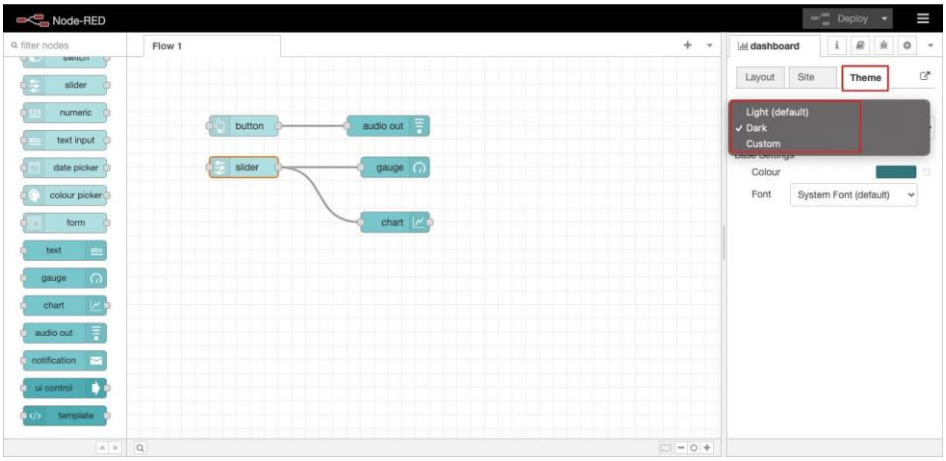

User Settings
Close
View
Palette
Keyboard
Nodes
Install
sort:
a-z
recent
node-red-dashboard
15 / 3900
node-red-dashboard
A set of dashboard nodes for Node-RED
3.1.7
1 month ago
install
node-red-node-ui-list
Node-RED Dashboard UI widget node for simple list
0.3.6
11 months ago
install

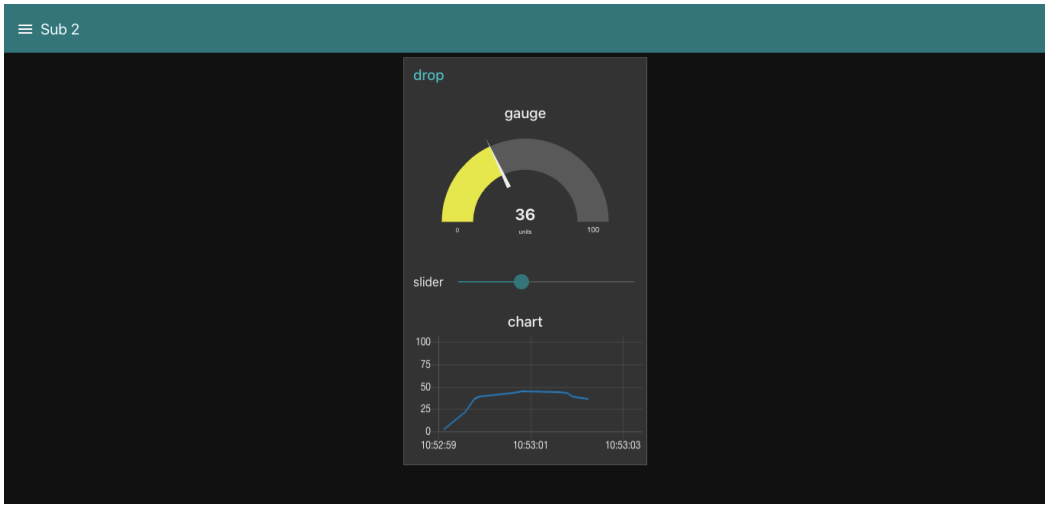

dashboard
button
dropdown
switch
slider
numeric
text input
date picker
colour picker
form
text
gauge
chart
audio out
notification
ui control
template

Step 4: Deploy and View the Dashboard

1. Click **Deploy** in the top-right corner of Node-RED.
2. Access your dashboard at http://<your-node-red-ip>:1880/ui.
3. Click on the Inject button to simulate data updates and observe changes on the Gauge and Chart widgets in real time.







Task 6: Include widgets for displaying sensor data and control buttons.

Step 1: Install Node-RED and Dashboard Nodes

1. **Install Node-RED**:
    - If Node-RED is not installed, install it on your Raspberry Pi:

sudo apt update

sudo apt install -y nodejs npm

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

Start Node-RED:

- node-red-start
- Access the Node-RED editor at http://<Your_RPi_IP>:1880.

2. **Install Dashboard Nodes**:
    - Go to **Menu > Manage Palette > Install**.
    - Search for node-red-dashboard and install it.

Step 2: Connect Sensors to Raspberry Pi

- Use a sensor like DHT11 or DHT22 for temperature and humidity readings.
- Connect the sensor to GPIO pins on the Raspberry Pi.
- Install the required Node-RED GPIO nodes:
- npm install node-red-node-pi-gpio
- npm install node-red-contrib-dht-sensor

Step 3: Create Dashboard Layout

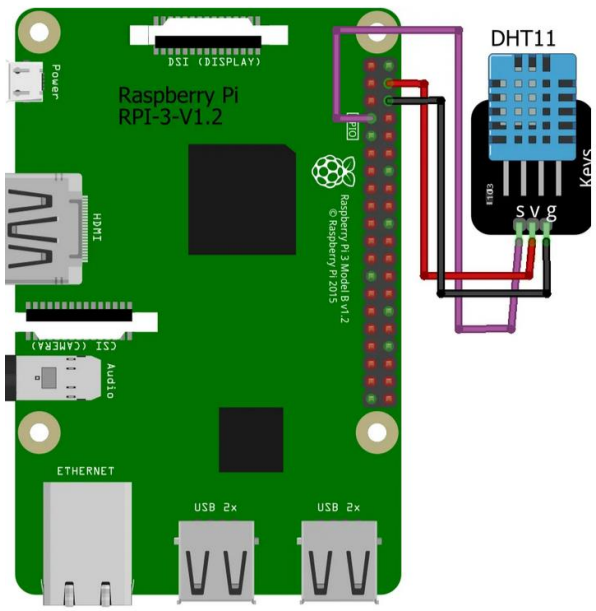
1. Go to the **Dashboard** tab in the Node-RED editor.
2. Create a new **Tab** (e.g., "IoT Dashboard").
3. Add two groups:
    - **Sensor Data**: For displaying sensor readings.
    - **Controls**: For control buttons.

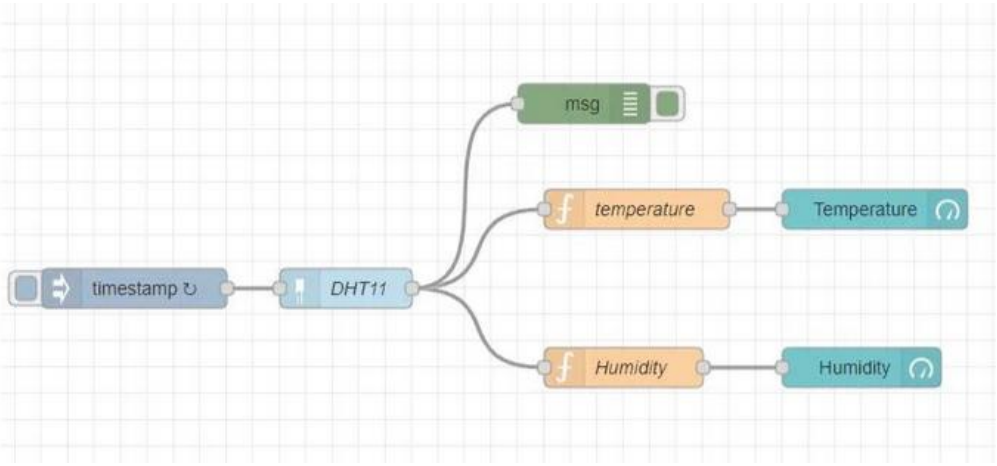
Step 4: Build the Flow

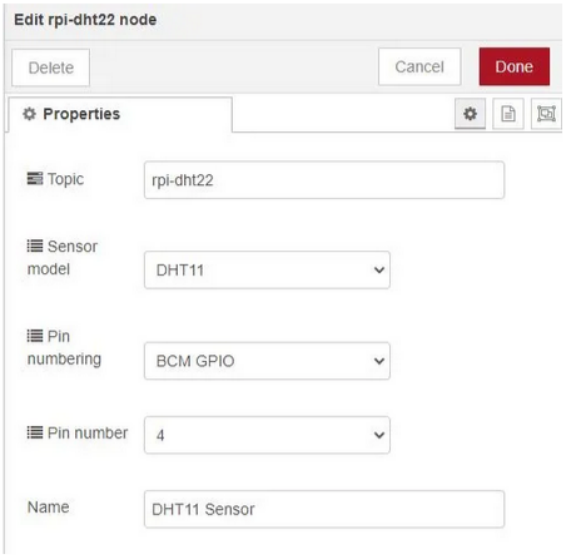
Drag and configure the following nodes:

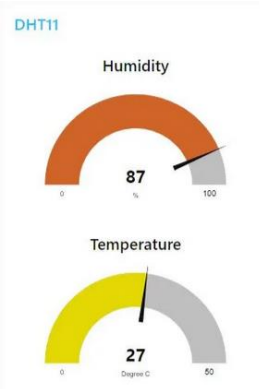
Sensor Data Widgets

1. **DHT Sensor Node**:
    - Drag a DHT sensor node onto the workspace.
    - Configure it to read temperature and humidity from your connected sensor.
    - Set the GPIO pin where the sensor is connected.
2. **UI Gauge Node**:
    - Drag a Gauge node to display temperature readings.
    - Configure it with properties like label (e.g., "Temperature"), range (e.g., 0–50°C), and units (°C).
3. **UI Chart Node**:
    - Drag a Chart node to visualize temperature trends over time.









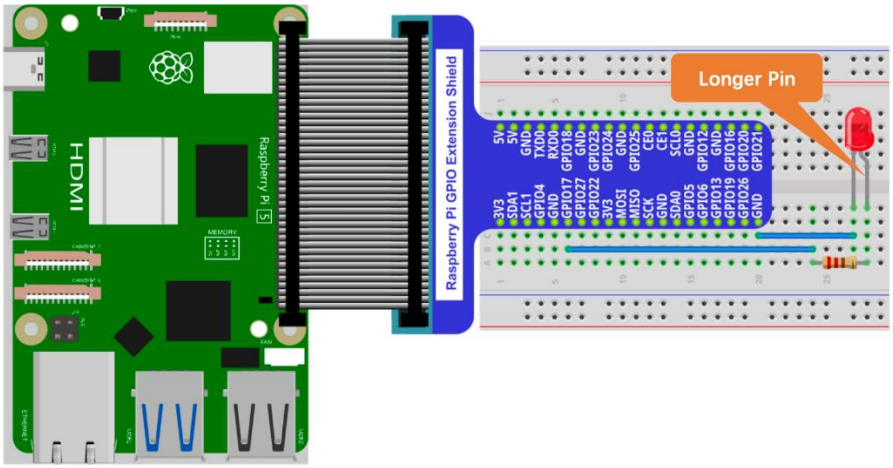
Control Buttons Widgets

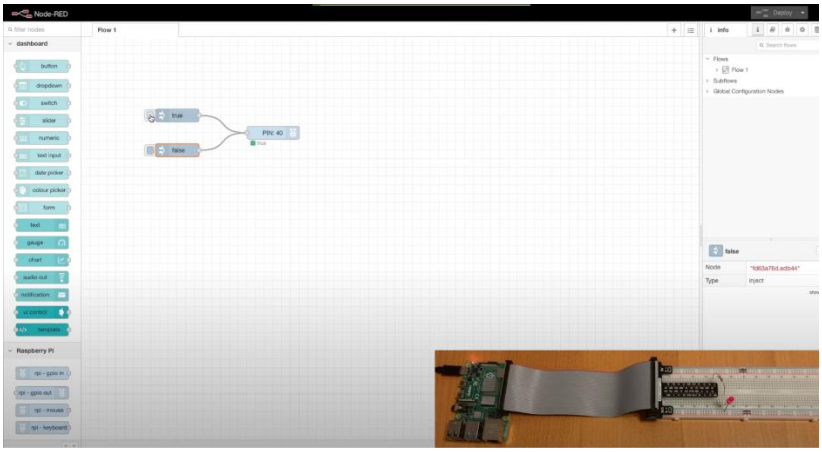
1. **UI Button Node**:
    - Drag a Button node onto the workspace.
    - Configure it to send commands (e.g., turn on/off an LED or relay).
2. **Raspberry Pi GPIO Out Node**:

- Connect the Button node to a GPIO Out node to control devices like LEDs or relays.

Step 5: Deploy and Test

1. Click **Deploy** in the top-right corner of Node-RED.
2. Access your dashboard at http://<Your_RPi_IP>:1880/ui.
3. Test by observing sensor data updates and interacting with control buttons.





Task 7:

Configure security settings for Node-RED, including user authentication.

Node-RED provides several security features including user authentication and HTTPS support. Here's how to configure these settings:

Basic Authentication Setup

1. **Edit your settings.js file** (typically located in ~/.node-red/settings.js)

2. **Enable authentication** by uncommenting and modifying the adminAuth section:

```
adminAuth: {
    type: "credentials",
    users: [
        {
            username: "admin",
            password: "$2a$08$zZWtXTja0fB1pzD4sHCMyOCMYz2Z6dNbM6tl8sJogENOMcxWV9DN.",
            permissions: "*"
        }
    ]
},
```

Generate a password hash using the node-red-admin command:

node-red-admin hash-pw

HTTPS Configuration

To enable HTTPS, add this to your settings.js:

```
https: {
    key: require("fs").readFileSync('privkey.pem'),
    cert: require("fs").readFileSync('cert.pem')
},
```

Advanced Security Options

1. **User permissions** (for multi-user environments):

```
adminAuth: {
    type: "credentials",
    users: [
        {
            username: "user1",
            password: "hashed-password",
            permissions: "read"
        },
        {
            username: "admin",
            password: "hashed-password",
            permissions: "*"
        }
    ]
},
```

Disable the editor for public access: httpAdminRoot: false,

Enable CORS restrictions:

```
httpNodeCors: {
    origin: "https://yourdomain.com",
    methods: "GET,PUT,POST,DELETE"
},
```

Task 8: Implement SSL/TLS for secure communication in a Node-RED instance.

SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are cryptographic protocols designed to provide secure communication over computer networks, particularly the internet. TLS is the successor to SSL, though the terms are often used interchangeably.

Key Purposes of SSL/TLS

1. **Encryption**: Scrambles data in transit to prevent eavesdropping
2. **Authentication**: Verifies the identity of communicating parties
3. **Data Integrity**: Ensures data isn't altered during transmission

How SSL/TLS Works

1. **Handshake Process**:
    - Client connects to SSL-enabled server
    - Server presents its digital certificate
    - Client verifies the certificate
    - Secure session keys are established
2. **Symmetric Encryption**: After handshake, faster symmetric encryption is used for the session

Key Components

1. **Certificates**: Digital documents that verify identity
    - Issued by Certificate Authorities (CAs)
    - Contain public key and identity information
2. **Public/Private Key Pair**:
    - Public key encrypts data
    - Private key decrypts data
3. **Cipher Suites**: Combinations of cryptographic algorithms

To secure your Node-RED instance with SSL/TLS encryption, we need to follow these steps:

1. Obtain SSL/TLS Certificates

Option A: Self-Signed Certificates (for testing)

openssl req -x509 -newkey rsa:4096 -keyout privkey.pem -out cert.pem -days 365 -nodes -subj "/CN=yourdomain.com"

Option B: Let's Encrypt (for production)

sudo apt install certbot

sudo certbot certonly --standalone -d yourdomain.com

2. Configure Node-RED for HTTPS

Edit your settings.js file (typically at ~/.node-red/settings.js):

```
module.exports = {
    // ... existing settings ...

    // Enable HTTPS
    https: {
        key: require("fs").readFileSync('/path/to/privkey.pem'),
        cert: require("fs").readFileSync('/path/to/cert.pem'),
        // For Let's Encrypt:
        // key: require("fs").readFileSync('/etc/letsencrypt/live/yourdomain.com/privkey.pem'),
        // cert: require("fs").readFileSync('/etc/letsencrypt/live/yourdomain.com/fullchain.pe
m')
    },

    // Force HTTPS (redirect HTTP to HTTPS)
    requireHttps: true,

    // Disable the HTTP Admin interface (optional)
    // httpAdminRoot: false,

    // ... other settings ...
};
```

3. Configure Security Headers

Add these to your settings.js for enhanced security:

```
httpStaticHeaders: {
    "X-Frame-Options": "DENY",
    "X-Content-Type-Options": "nosniff",
    "X-XSS-Protection": "1; mode=block",
    "Strict-Transport-Security": "max-age=31536000; includeSubDomains",
    "Content-Security-Policy": "default-src 'self'; script-src 'self' 'unsafe-inline'; style-src
'self' 'unsafe-inline'; img-src 'self' data:"
},
```

5. Verify Your Configuration

After restarting Node-RED and your web server, verify with:

Check if Node-RED is listening on HTTPS

curl -v -k https://localhost:1880

Test SSL configuration (for public domains)

openssl s_client -connect yourdomain.com:443 -servername yourdomain.com | openssl x509 -noout -text