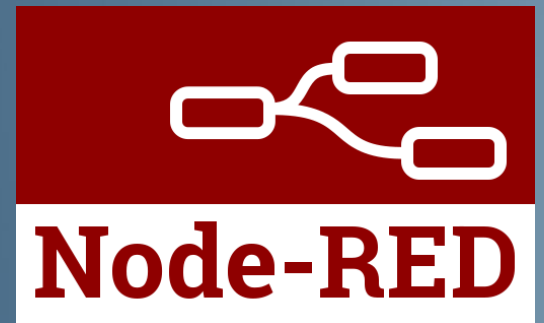


i-UG Open Source Education
for IBM i



LESSON: Lights-1



1. Controlling a simple LED light strip

Node-Red is capable of many IoT (Internet of Things) functions, but for now, we'll start with a simple example. This lesson will go through the functions of remotely changing the properties of a light, turning it on/off, changing colours and making changes dependant on parameters. First things first, let's take a look at the lights, which are situated with an IBM Power System in Manchester, UK.

2. Let's see the Light Strip

The light strip is on a Network in Manchester UK. We'll set up a browser to look at this light strip so we can see the results of our actions.

To do this, we will follow the instructions in the document on this link: -

[Viewing the Phillips Hue light strip.](#)

3. A brief foray into MQTT

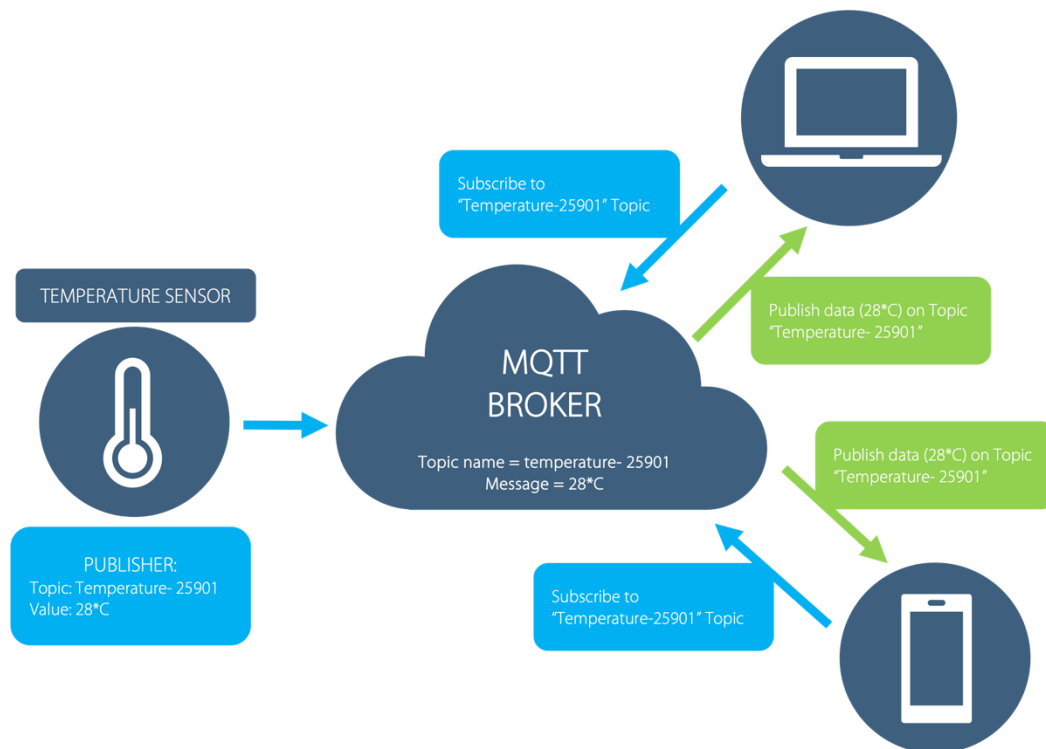
Now that we can see it, we can explain to you that setting up any particular Light strip (or device or switch) will vary greatly between manufacturers, products and device type. For this reason, we have elected not to set this as part of the Course. Instead, we will set up an MQTT link and send our instructions to this link.

a. What is an MQTT Link?

MQTT (Message Queue Telemetry Transport) is a "lightweight, 'publish-subscribe', machine to machine network protocol for message queue/message queuing service. It is designed for connections with remote locations that have devices with resource constraints or limited network bandwidth, such as in the Internet of Things (IoT). It typically runs over TCP/IP."

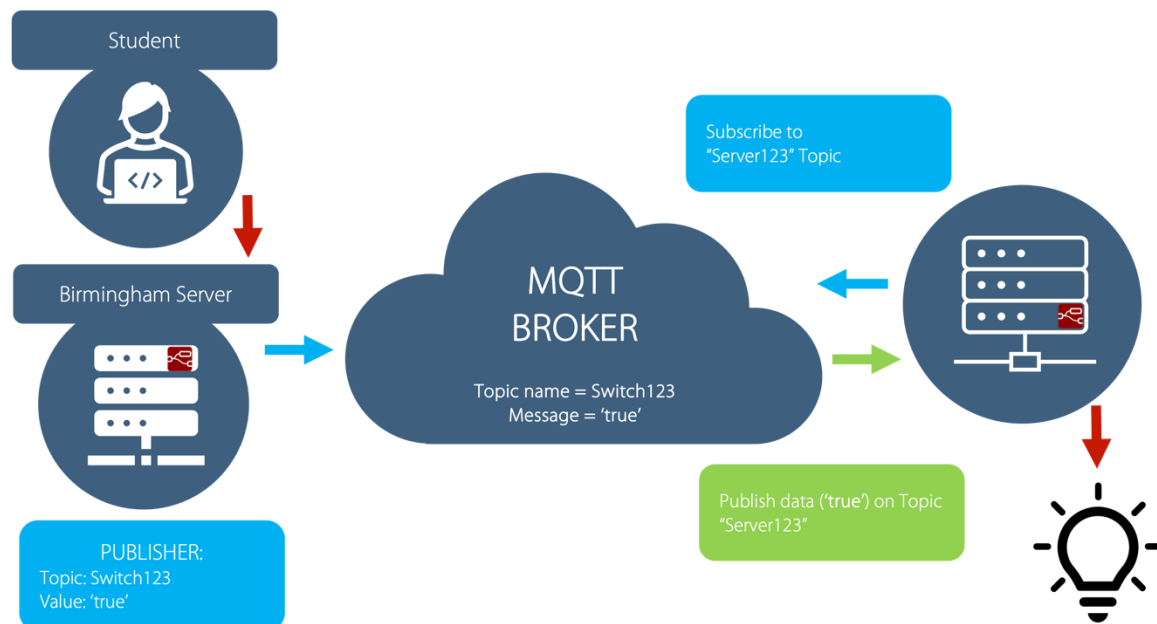
In simple terms, it allows us to easily describe a device that, on the one hand, we can send information from (PUBLISHER) which we can use to send out information about a condition or send an instruction, and also use to read information from the device that we can use (SUBSCRIBER).

In this lesson, we will just touch on MQTT, as we need to make switching the light on and off as simple as we can. However, MQTT will be VERY important as you progress with the IoT. It is explained more fully in a later module.



b. How we will use the MQTT service:

In our example for this lesson, we will use the MQTT Broker in a very simple way. We will **PUBLISH** the code to set the light in the Node-RED MQTT node on the Birmingham Server in a specific **TOPIC** and the Manchester server will be listening on its Node-RED node (**SUBSCRIBING**) for the same **TOPIC**.

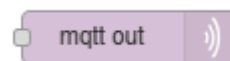


Let's see how we get along...

c. Setting up the link to Manchester

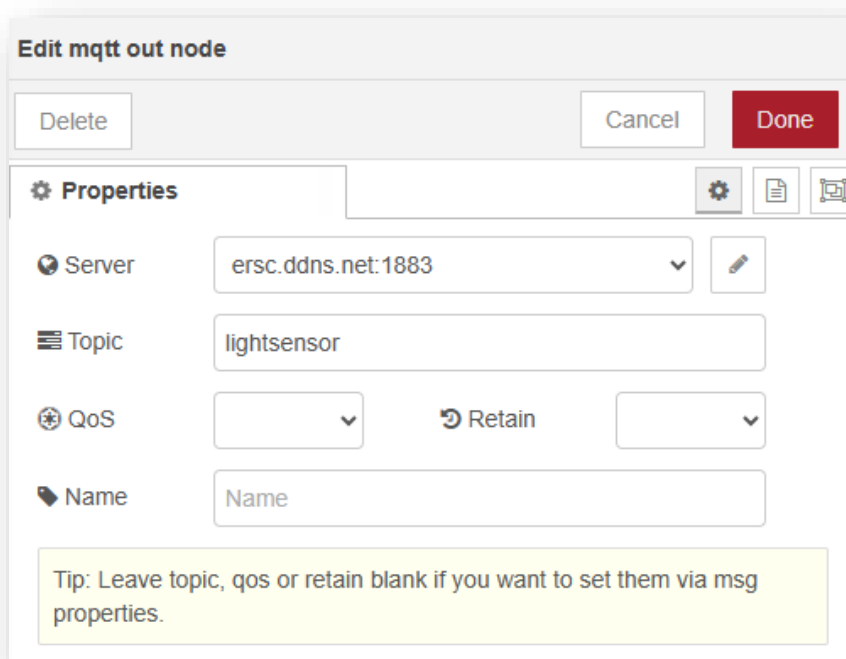
We are using a Philips Smart Hue Hub and Light strip for our example. The light strip itself is located on our Manchester office. As we have described above, the way we will switch on and off the light – all with the correct codes – is that we will transmit the code to do the work, over the MQTT Service.

Drag the **mqtt out** node from the Palette on to the Workspace:



This node will take the instruction or code that you want to send to the Light strip over the MQTT service to the Manchester Server where the instruction will be applied to the Phillips Hue Light strip. If you get it right, the Light strip should behave the way you have intended.

We need to tell the mqtt out both which service we are registered with and what the Topic name is. This is done by double-clicking the mqtt out node.

A dialog box titled "Edit mqtt out node" with a light gray background. At the top, there are three buttons: "Delete", "Cancel", and "Done". Below the buttons is a "Properties" section with a gear icon and three icons (gear, document, and monitor). The "Server" field has a dropdown menu showing "ersc.ddns.net:1883" and a pencil icon. The "Topic" field has a text input with "lightsensor". The "QoS" field has a dropdown menu with a downward arrow. The "Retain" field has a checkbox and a dropdown menu with a downward arrow. The "Name" field has a text input with "Name". At the bottom, there is a yellow tip box that says "Tip: Leave topic, qos or retain blank if you want to set them via msg properties." To the right of the dialog box is a square button with a pencil icon.

From here, click on the Pencil icon so we can get the detail in. This will position us on the Connection tab:

Edit mqtt out node > Edit mqtt-broker node

Delete Cancel Update

Properties

Name

Connection Security Messages

Server: ersc.ddns.net 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

In this Connection tab, we will fill in the Server where the MQTT service we are using is running, select the Protocol level and, where necessary, state who the Client is for recognition.

Server: ersc.ddns.net

Protocol: MQTTV3.1.1 (Use Drop-down)

Client ID: Leave Blank for this application

Name: lightsensor

Next, select the Security tab:

Edit mqtt out node > Edit mqtt-broker node

Delete Cancel Update

Properties

Name

Connection Security Messages

Username essistomikron

Password

Here is where we enter the credentials we have set up at this MQTT provider.

Username: essistomikron

Password: xCYfGVEx-8wfhFXfn (Probably best to copy and paste here...)

Now click on Update,

You will return to the node's Properties. Here you will need to update the Topic that we will use.

In the Topic box, key: **lightsensor_nn** (where 'nn' is your Student ID)

Just to be clever, we are going to change the symbol that currently defines the MQTT node as MQTT and set it to a Light bulb instead.

Click on the Appearance icon: -

Edit mqtt out node

Delete Cancel Done

Appearance

Label ☒ Show

Icon

Port labels

Search icons

node-red

Inputs

1.

Outputs

use default

You will be presented with this:

Select the dropdown next to the Icon and select the Light Bulb icon instead.

Then press Done. Your mqtt out node will now

look like this:



This should now set the light on correctly when the code is sent. If you have trouble, copy the MQTT OUT details from the GitHub

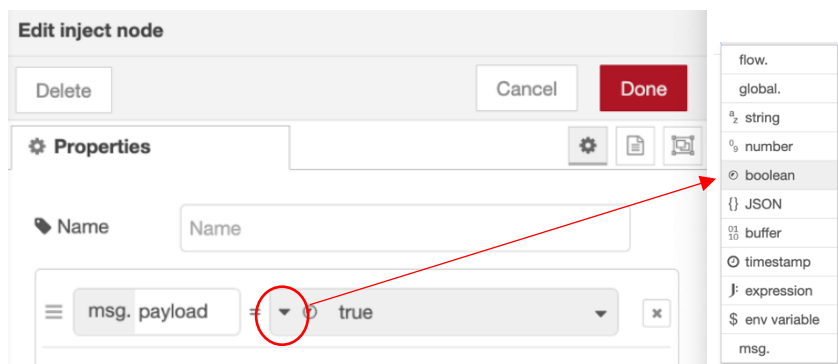
4. Turning the light on and off

a. Light Strip

Now we need to bring in the nodes to make changes to the light. The most basic way of engaging the node is to bring in an 'inject' node.

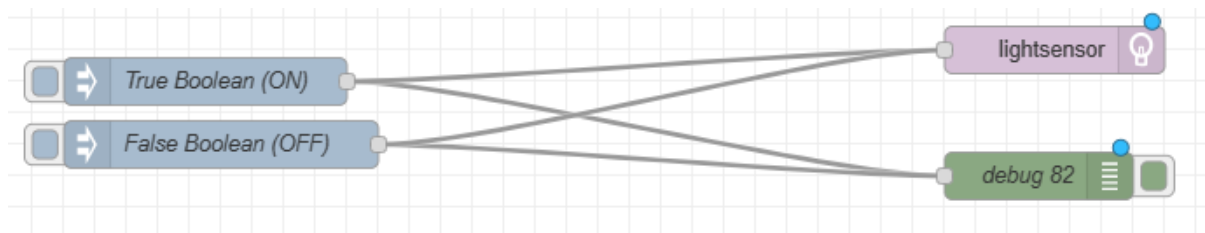
Within the node we need to change the message payload to a Boolean value (True or False). This will act as an on or off switch, depending on the true or false status, when connected to the bridge node via our MQTT Service. (we can ignore the message topic or remove it)

Create 2 Inject nodes. In each one you will need to change the data type to Boolean. Once you have done this, the right-hand drop-down will give only two options – true or false. Set one of each value – 'true' or 'false'. Make sure you give each one a memorable name in the Name box too.



Connect each of them to the lightsensor (mqtt) node we have on the Palette.

We will take the opportunity here of introducing the Debug node too. Drag in a Debug node and connect each of the Inject nodes to it too. You should end up with a small flow looking something like this:



The small blue spots will disappear once you hit the  button.

Use the button attached to each of the inject nodes to inject the command.

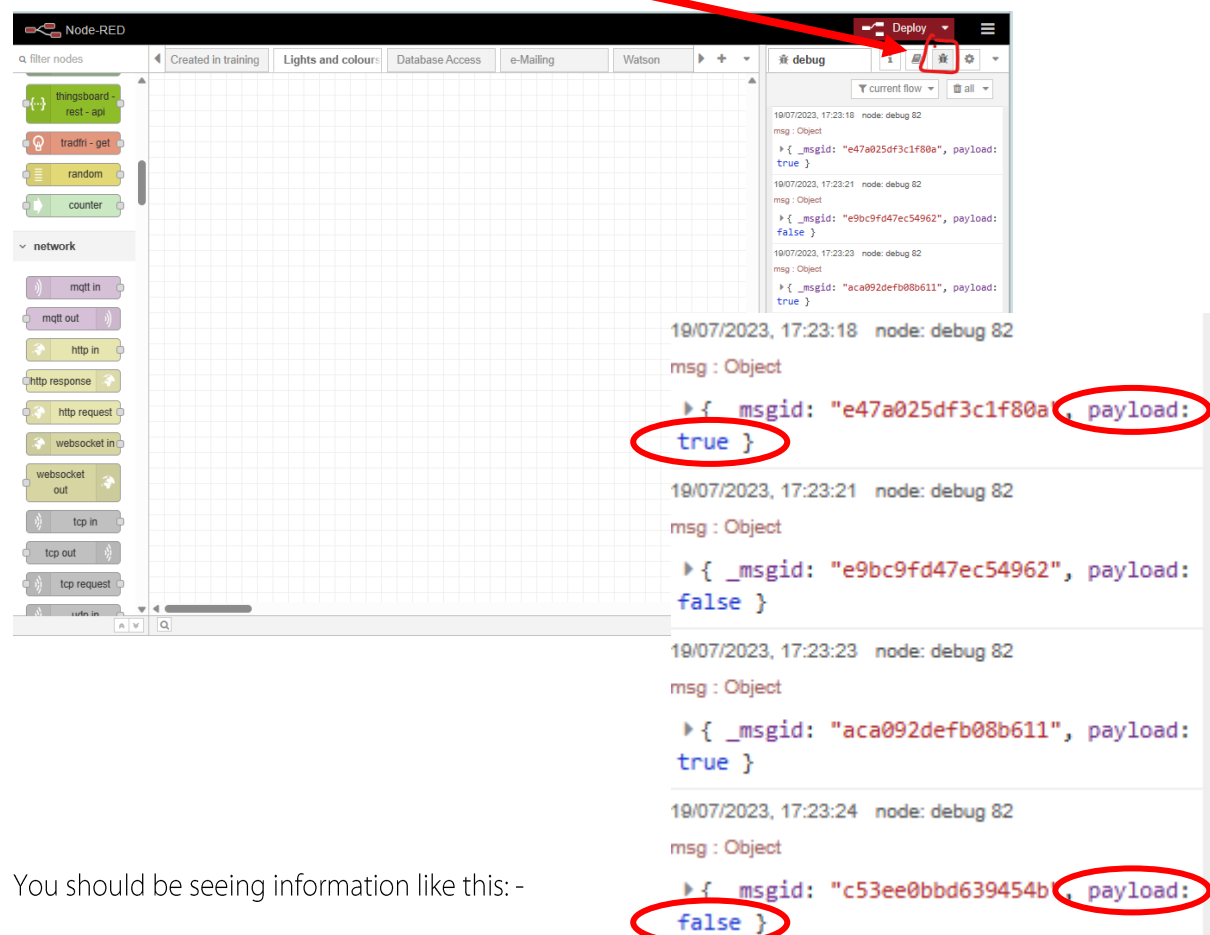
This will now work as our basic switch mechanism, turning the light on and off using Boolean logic, this structure will work as an underpinning of the more advanced flows we will create as we progress.

Try it now. You should see the lights draped around the Manchester server go on and off (in your other browser).



b. Debug

Take time to look at the Debug data in the right hand Sidebar. To set it to look at the Debug data, click this debug icon at the top of the Sidebar:

The screenshot shows the Node-RED interface with the Debug sidebar open. The sidebar displays a list of messages with their timestamps, node IDs, and payloads. The payloads are circled in red to highlight the Boolean values being sent.

```

19/07/2023, 17:23:18 node: debug 82
msg: Object
  { _msgid: "e47a025df3c1f80a", payload: true }

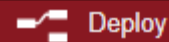
19/07/2023, 17:23:21 node: debug 82
msg: Object
  { _msgid: "e9bc9fd47ec54962", payload: false }

19/07/2023, 17:23:23 node: debug 82
msg: Object
  { _msgid: "aca092defb08b611", payload: true }

19/07/2023, 17:23:24 node: debug 82
msg: Object
  { _msgid: "c53ee0bbd639454b", payload: false }
  
```

You should be seeing information like this: -

All messages have a Message ID, but the important information for YOU is that in the message the Inject node sent to the lightsensor node, you used 'payload' and you wanted to either send a Boolean true, to switch on the light strip or a Boolean false to switch it off



If you have problems, first check that you have deployed the flow (hit the button). Next, check the contents of the nodes in your flow against the contents of the lesson flow in GitHub.

Once it works, congratulate yourself. You have just mastered your first flow, successfully engaged an MQTT Broker to move IoT data between two servers in two different cities and controlled a device- in this case a Phillips Hue Light Strip. In doing so you engaged two powerful IBM Power servers to do your bidding.

Not bad.

And with that, the lesson is concluded. If you didn't manage to make one of the flows work then you can copy one of the flows from our GitHub repository [here](https://github.com/JoshRyanEOG/i-UG_Education_Node-Red), and see where you went wrong. These functions, although they seem simple, are the backbone of many useful applications of Node-Red for major and small businesses.

https://github.com/JoshRyanEOG/i-UG_Education_Node-Red