



i-UG Open Source Education
for IBM i

Node-Red

LESSON 4: Engaging with the
UI

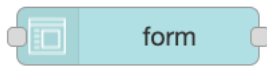


i-UG

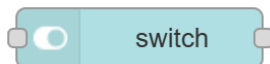
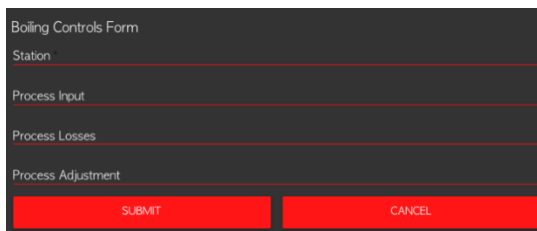
1. The Node-RED UI - Dashboard

As mentioned before, the Dashboard functions as the UI (User Interface) of Node-Red, and can be used as an easier way to navigate and operate interface for using your flows. It can also work as a quick visual indicator of how your flow or work process is running.

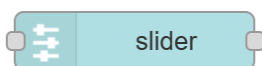
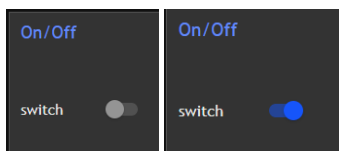
You can have several types of dashboard node, all of which can be found by searching the library for what you are looking for on [Node-Red's](https://nodered.org/docs/user-guide/dashboards) site, however many come with the **node-red-dashboard** package. Some examples we can look into now are:



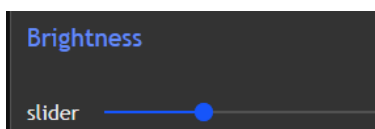
The form node can be useful for inputting data from the Dashboard, which, with the correct coding within a function node, can change and edit the parameters of your task. (We will look at this in detail in the next lesson.).



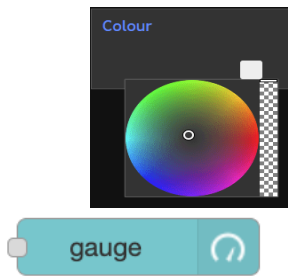
A simple Node that uses a Boolean method to set a switch to on or off



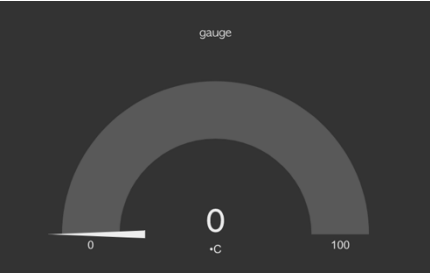
A great tool for setting up a scale and having a slider bar to move up and down through the scale and send the stopping point in the `.msg` over to the next Node.



A powerful generator of hues and colours that can control a multi-light.

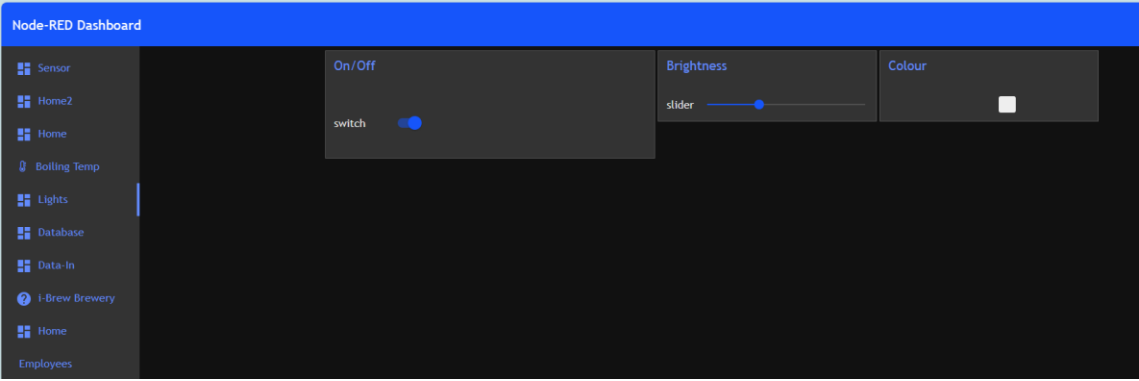


The gauge node is a great visual indicator for progress or item amounts, such as fluid levels or temperature read-outs. They can be edited to fit your exact needs.



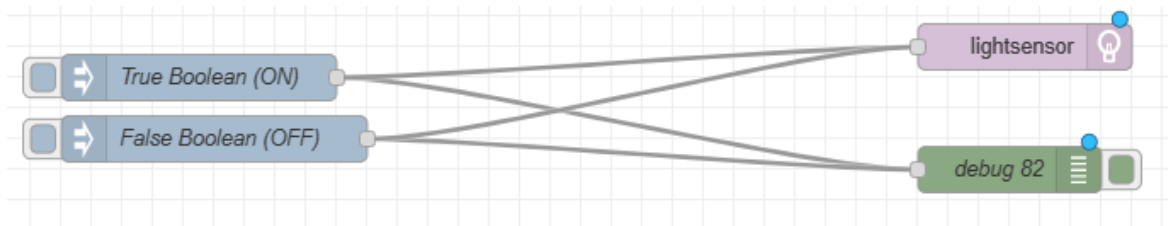
Positioning

When you add a dashboard node such as a gauge, you can decide where in the dashboard it will appear by first deciding what group it will be in. We will use a Group called Lights for the first example. We can then position the widgets where we want them on the Group Page, although, Node-RED already does quite a good job of positioning them. It can look like this: -

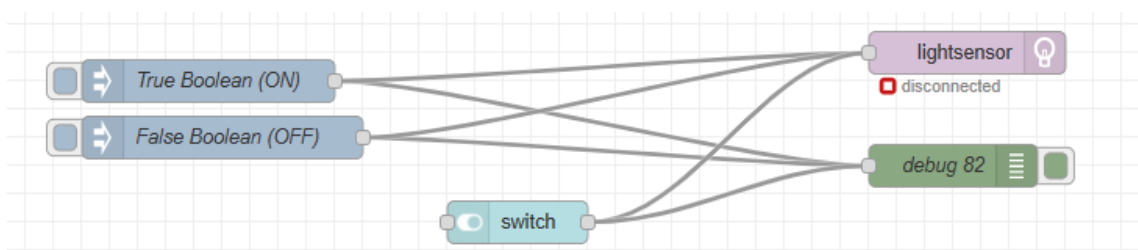


2. UI Switch

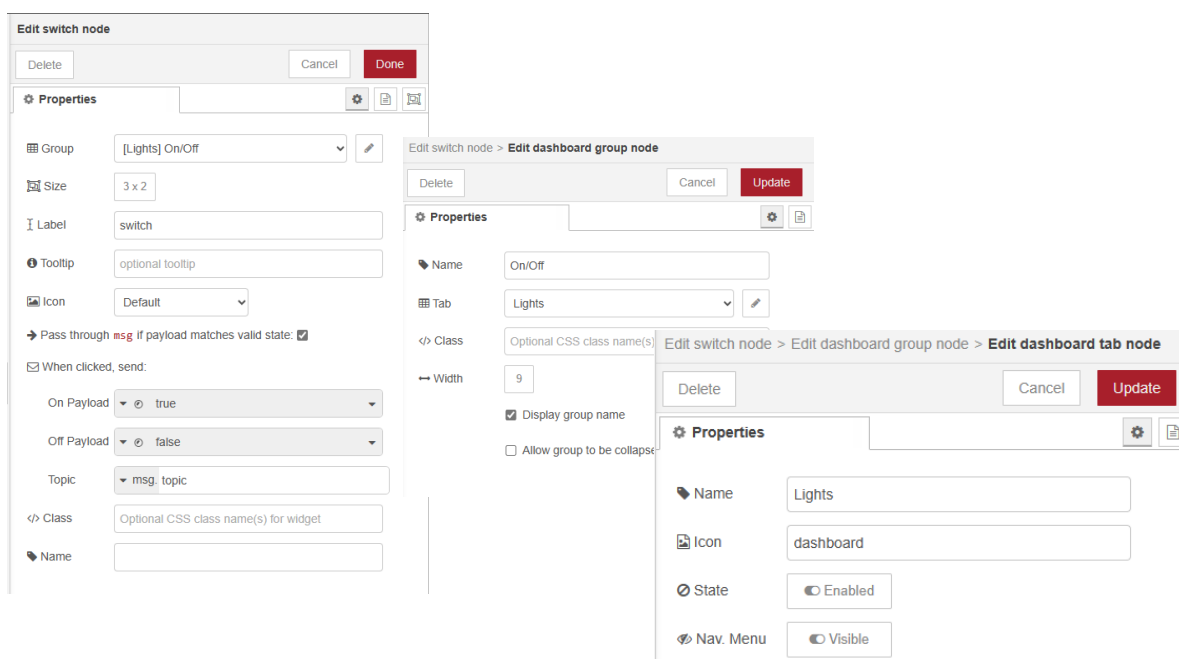
So, let's go back to the Flow where we are controlling the lights (Lesson 2) and we'll add some UI pizzazz to it. This is what it should currently look like: -



Drag in the Switch Node and add it to the Palette.



Double click on the Switch Node to set its properties.



By clicking the pencil next to the group menu, you can create a new group. Once on that panel, you can click the pencil next to the Tab menu, you can create or change which Tab in the UI it's under.

Leave the other parameters as the standard, but note that the `msg.payload` will be set to 'true' or 'false' depending on the switch position. Make sure this new Node is linked into the Flow as above, then Deploy the Flow, and use the Switch on the UI to switch the Manchester lights on and off.

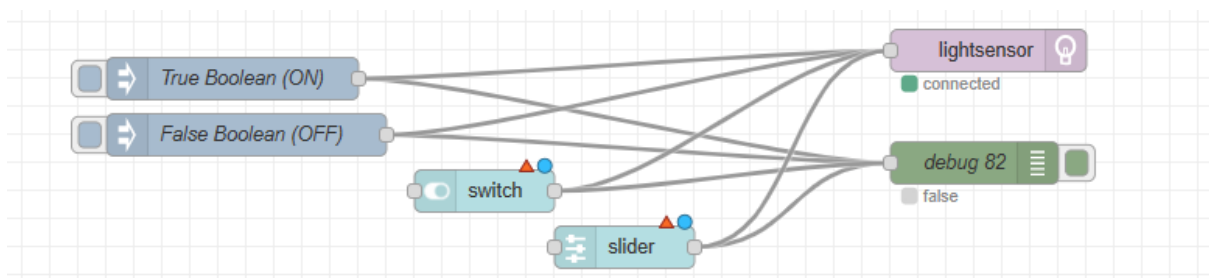
You can view the UI by keying in the same URL as your flow, followed by `/ui`: -

<http://172.16.1.10:188nn/ui>

3. UI Slider

Next, lets add a Slider. The Slider is a little more complex and we will use this to alter the brightness of our light strip.

Drag the Slider Node onto the Palette and, just as with the Switch Node, lets double-click the Node and set up the properties.



Edit slider node

Delete Cancel Done

Properties

Group [Lights] Brightness

Size auto

Label slider

Tooltip optional tooltip

Range min 0 max 100 step 1

Output continuously while sliding

If msg arrives on input, pass through to output: ☒

When changed, send:

Payload Current value

Topic msg.topic

Class Optional CSS class name(s) for widget

Name

Edit slider node > Edit dashboard group node

Delete Cancel Update

Properties

Name Brightness

Tab Lights

Class Optional CSS class name(s) for widget

Width 6

☒ Display group name

☐ Allow group to be collapsed

Edit slider node > Edit dashboard group node > Edit dashboard tab node

Delete Cancel Update

Properties

Name Lights

Icon dashboard

State ☒ Enabled

Nav. Menu ☒ Visible

Once again, by clicking the pencil next to the group menu, you can create a new group. Once on that panel, you can click the pencil next to the Tab menu, you can create or change which Tab in the UI it is under.

Leave the other parameters as the standard, but set the range from 1 to 100 in steps of 1, just like in the example above. The Phillips Light Strip is managed by a hub that is smart enough to realise that this is not a Boolean parameter being sent to it, it is a number, therefore it simply applies this to the brightness of the Light Strip.

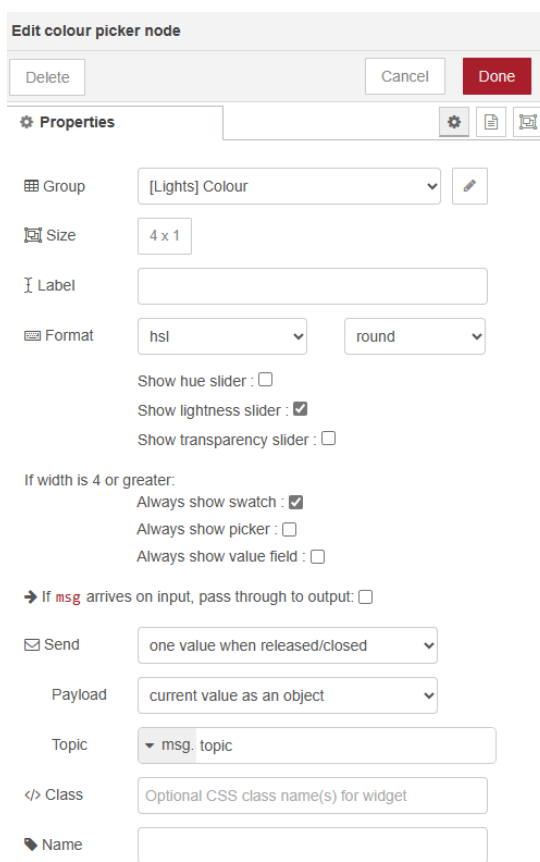
Make sure this new Node is linked into the Flow as above, then Deploy the Flow, and use the Slider on the UI to alter the brightness of the Manchester light strip.

4. UI Colour Picker

Now, let's up the game by adding a Colour Picker. The Colour Picker is again a little more complex. We want it to alter the colour of our Light Strip but there are a few points we will need to control and manage as there is a difference between the capabilities of the Node-RED standard Colour Picker and the requirements of our Phillips Hue system. This is something that you will often encounter, but there is always a way around it, as we will see here.

Drag the Colour Picker Node onto the Palette and also drag on a Function Node and a JSON Node.

Starting with the Colour Picker Node, double-click the Node and set up the properties.



Edit colour picker node

Delete Cancel Done

Properties

Group: [Lights] Colour

Size: 4 x 1

Label:

Format: hsl round

Show hue slider: ☐

Show lightness slider: ☒

Show transparency slider: ☐

If width is 4 or greater:

Always show swatch: ☒

Always show picker: ☐

Always show value field: ☐

→ If msg arrives on input, pass through to output: ☐

Send: one value when released/closed

Payload: current value as an object

Topic: msg topic

Class: Optional CSS class name(s) for widget

Name:

There is a little more complexity here, starting with just what we want the Colour Picker to send to the next node in order to meet the needs of the device.

In the format drop down box, we can select a range of standards, **rgb** (Red/Green/Blue), **hsl** (Hue/Saturation/Lightness) etc.

Our device requires the information in the **hsl** format, but it doesn't end there...

For now, select the **hsl** option.

Make sure that the 'Show lightness slider' box is checked and that the 'Always show picker' box is checked, Uncheck the 'Always show swatch' box, if it is checked.

This Node will send the following to the next Node:

-

The Hue (colour) as msg.payload.h

The Saturation as msg.payload.s

The Lightness (Brightness) as msg.payload.l

We will need the hue and the lightness next.

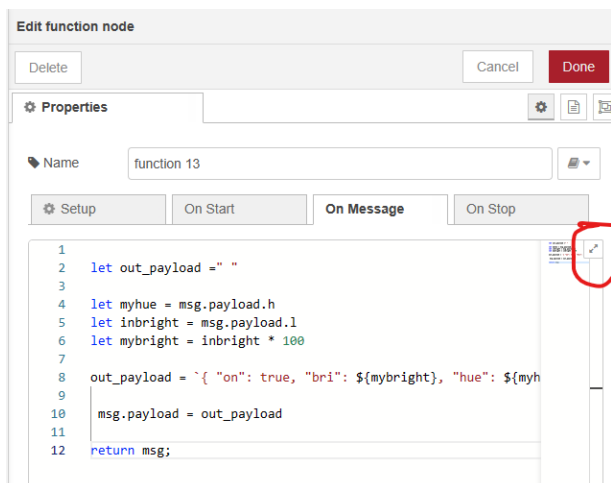
As before, by clicking the pencil next to the group menu, you can create a new group. Once on that panel, you can click the pencil next to the Tab menu, you can create or change which Tab in the UI it is under.

Now let's get the data that the Colour Picker gives us and change it to what we need: Open the newly dragged-in Function node and double-click to open it.

In this node, we will need to use a little code (JS) to build a string containing all of the components that our light strip needs in order to change to the colour and brightness that you select in the Colour Picker Node.

5. Data Manipulation

As you will see later in the Debug node, the Colour Picker sends a JSON object broken down into hue (h), Saturation (s) and lightness (l). We are terming the lightness as brightness for the Phillips Hue. We will skip Saturation for now. We need to change this slightly.



Your pane will be empty.

Use the expansion button (marked here in red circle) so we have a clearer view: -



Key this code into your Node (You can get it from GitHub if you like). Here is what the code is doing: -

```
1
2 let out_payload = ""
3
```

Set up a variable called **out_payload**. Set it to blanks. We will populate this variable with the string to send on.

```
4 let myhue = msg.payload.h
5 let inbright = msg.payload.l
6 let mybright = inbright * 100
7
```

Set up three new variable to use as work fields. We populate these variable with some of the Colour Picker data from msg.payload.h (hue) and .l (Lightness) which is coming in to this Node.

A limitation of either the Colour Picker or the light strip means that we need to multiply the value of the hue by 100 in order to work for us (Line 6). Like we accepted earlier, every device will have its own characteristics.


```

7
8  ✓ out_payload = `{ "on": true, "bri": ${mybright}, "hue": ${myhue} }`
9  |

```

This piece of code is constructing a text string to pass on to the next Node. The interesting bit is that we are going to replace some standard fields with substitution variables from our code above.

Between the two quote (') marks, and all shown in red, is simply a text string, but it is written as a JSON object. JSON objects are constructed of name/value pairs. They have a text to describe the data in double quote then a Colon followed by the data that we want to send.

`"on": true`

It is followed by a comma to denote that we want to add another name/value pair. By surrounding all of the pairs by curly brackets {} we are describing an array.

Substitution variables are denoted by a Dollar sign \$ followed immediately by the name of a variable wrapped in its own curly brackets {}. So, in the name/value pair for the brightness, I always want to call the *name* "bri" but I want the *value* to be whatever comes from the Colour Picker. – in this case the variable **mybright**. (remember, we multiplied this value by 100 as we set up the variable). We are doing the same with hue (myhue).

So, now we have a string that is laid out in a JSON format... but it is not a JSON object yet. We will do this with the following Node.

```

9
10 msg.payload = out_payload
11
12 return msg;

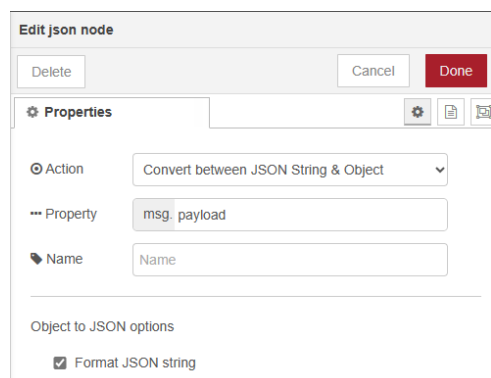
```

The final piece of the code in this Function Node is to set the `msg.payload` to the value if the work variable – `out_payload`.

And we send that on to the next Node at the `return` statement, as always.

6. Conversion

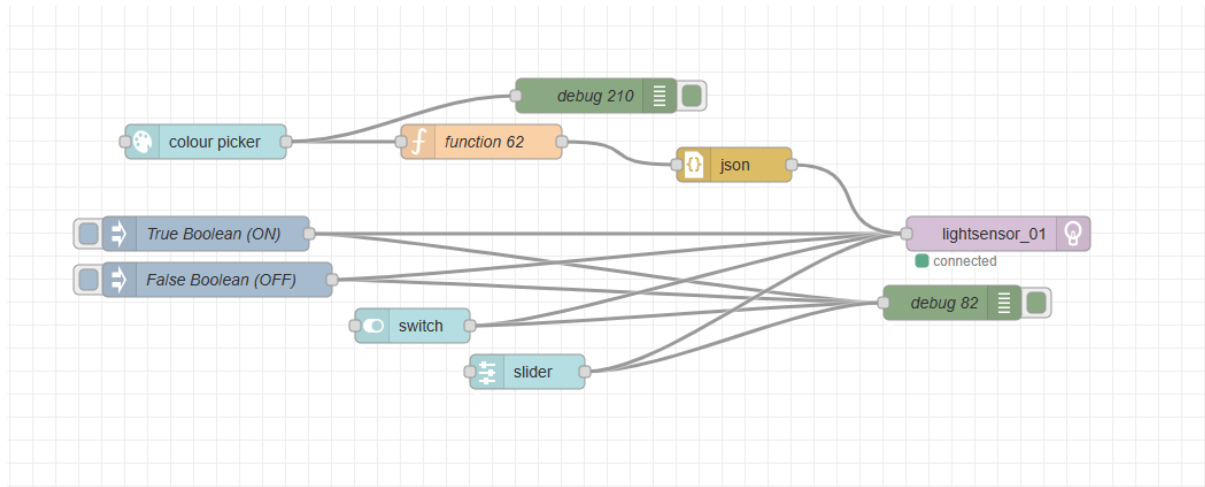
So, we can now use a very useful Node, the JSON node, to convert our string into a JSON object, ready for being sent on to the light strip. This is a simple one to set up but very useful and effective. Drag in the JSON Node if you haven't already, and double click it.



Just make sure that you select the 'Convert between JSON string and Object'.

You can check the Format JSON string box, but it is not really necessary, the Node defaults to this.

You should end up with a Flow that looks like this: -



Make sure the Nodes are all attached as in the diagram above, then Deploy. Use the new Colour Picker on the UI to choose colours and brightness and watch the lights in Manchester follow your every choice.

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