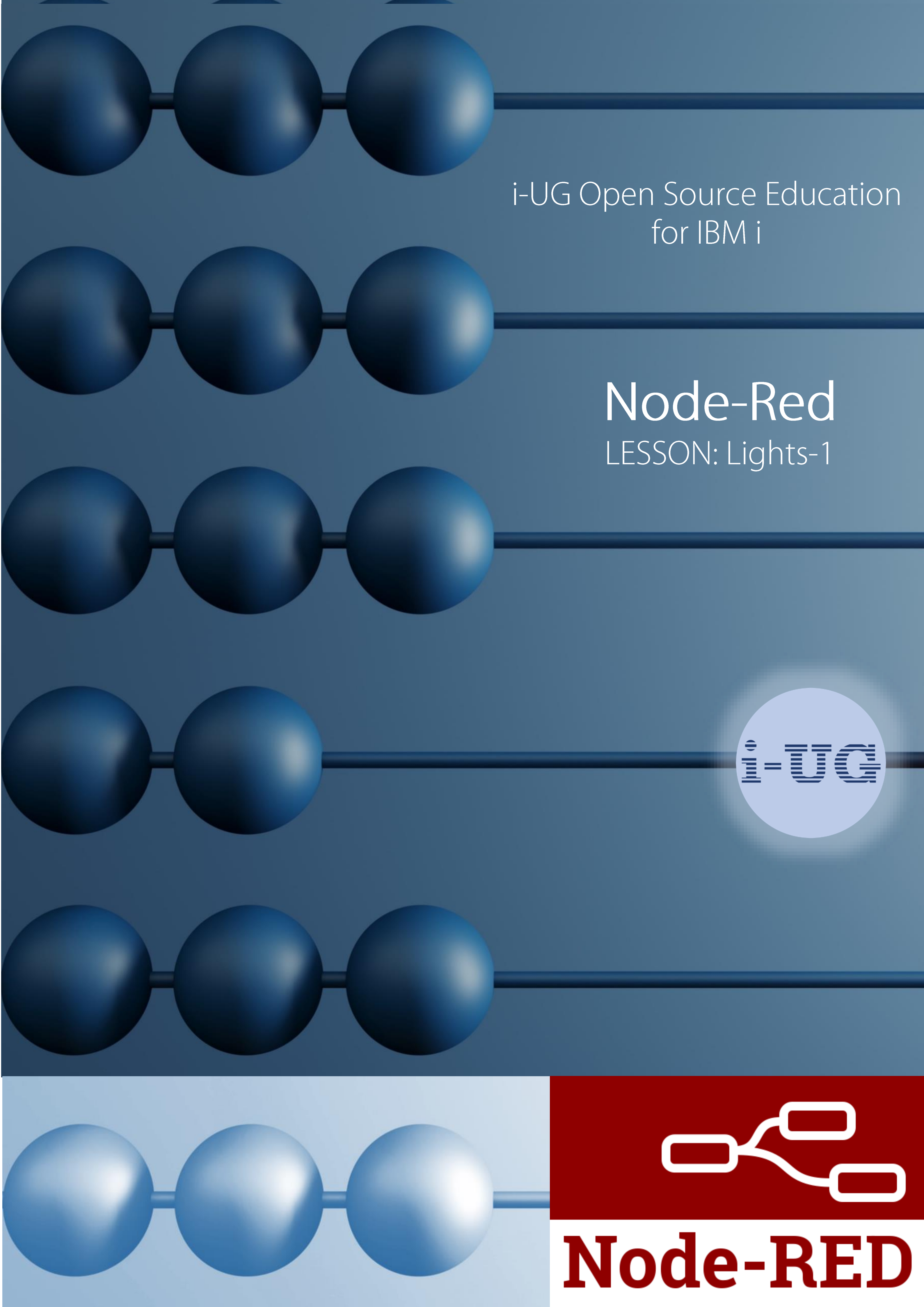
i-UG Open Source Education for IBM i

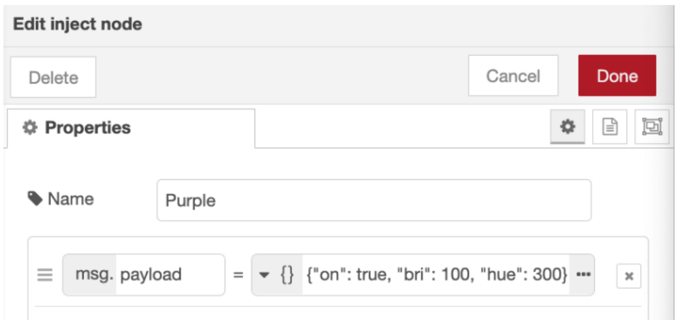
Node-Red

LESSON: Lights-1



1. Changing the colours and brightness of the light

So, switching a light on and off (over hundreds of miles away) was a cinch. However, the light strip we are using is capable of changing the hue, saturation and brightness, but for the sake of simplicity we will be changing the brightness and hue only. This will involve the same nodes as we have already used but with some adjustments.


Edit inject node

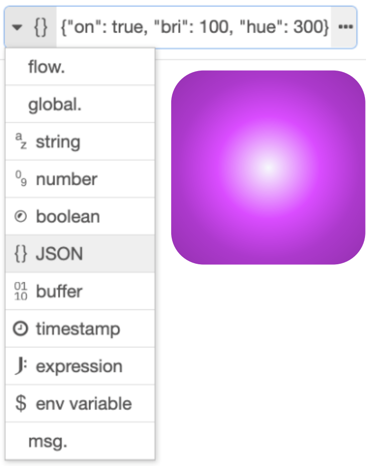
Delete | Cancel | Done

Properties

Name: Purple

msg. payload = {} {"on": true, "bri": 100, "hue": 300}

We'll start by making the light go purple and setting the brightness to full. The Philips Hue brightness settings range from 0 (off) to 100 (full brightness), and the hue (or colours) range from 1 to 359, giving a high range of different colours. To give the strip the commands for the settings we want, we must enter them into an inject node in the form of a JSON object.



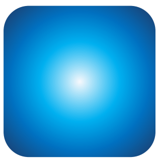
Drag in and open the inject node, in the message payload box, click the dropdown arrow and change the data type to JSON.

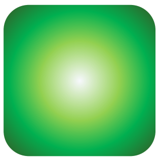
Click into the text box and type in the following command:

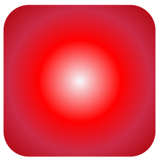
{"on": true, "bri": 100, "hue": 300}

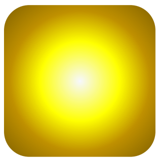
"on" is the setting to tell the light to turn on, and the Boolean statement **true** confirms that command, false would keep the light off. **"bri"** is the brightness setting, the number that follows is the value you want it set at. **"hue"** is the colour setting, and the number that follows is the value we are setting the colour to.

Your task now is to make a range of inject nodes of several different colours, so you are able to switch between them with the press of an inject button. See if you can find the "hue" setting for each of the following colours.

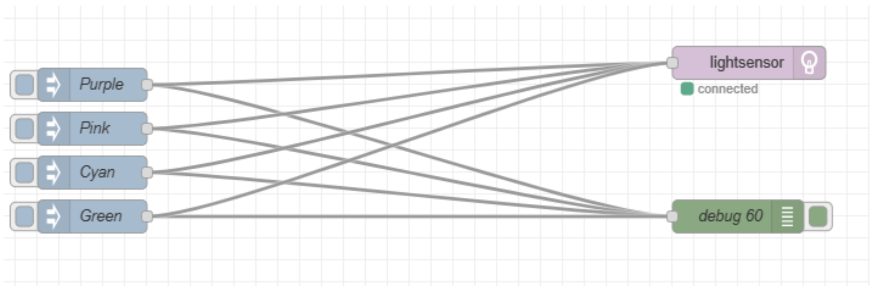

Hue Value:


Hue Value:

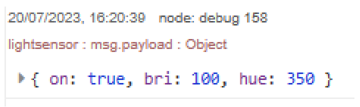

Hue Value:


Hue Value:

Once you have created this flow, it should look something like the image below.



Check out the Debug in the Sidebar.


```
20/07/2023, 16:20:39   node: debug 158
lightsensor : msg.payload : Object
▸{ on: true, bri: 100, hue: 350 }
```

2. Creating a randomized colour generator

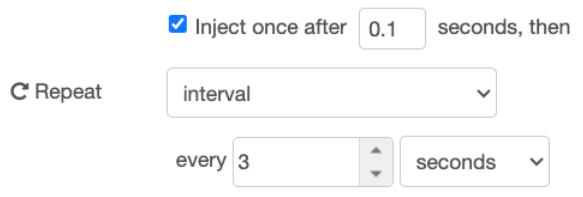
Now to create something a little more complex, a flow that will randomise the colour. To do this we will require the introduction of two new nodes that will come in useful later.

Random Node:
This node will generate a random number between two given values when prompted by an inject node.

Gate Node:
This node can halt the payload of a flow depending on Boolean logic, acting as a gateway for information.

Debug Node:
This will allow us to see the payload from a node.

The main principal of this flow will be to use the code that we have created already to provide the lighting strip with the information needed to produce coloured light, however we are making the colour (hue) a variable element of data. Every time this data refreshes, the light strip will instantly reflect this by changing to the new colour. The *random node* will act as a surrogate to provide the hue data, but you can imagine the random variable being replaced with intentional data.

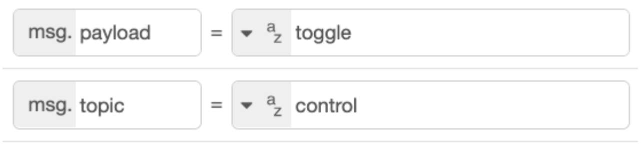


[✓] Inject once after 0.1 seconds, then

Repeat: interval

every 3 seconds

To start, drag in two *inject nodes*. For one of these nodes, it should be a simple timestamp but set it to repeat by changing the repeat setting to 'interval' and change to number count to a setting of your choice, for demonstrational purposes we will simply set it to every 3 seconds. This will act as a repeating injection of the data.
The other *inject node* will act as a control toggle to open and close a gate. Set **msg.payload** to toggle and **msg.topic** to control.

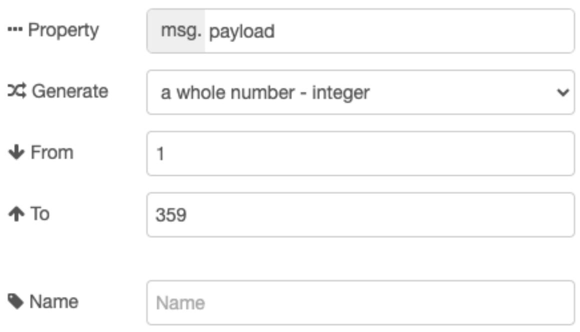


msg. payload = toggle

msg. topic = control

Now bring in a *gate node*, with both *inject nodes* connecting to the *gate node*. No settings need to be changed within the node for now. This will allow us to halt the changing of the colour by blocking the flow of inputs refreshing the data set.

Following this we will bring in our *random node*. Inside the *random node* we will input the following settings:
The range we are using is 1 – 359 as we have 359 hue values built into the light strip, so each number generated will be its own individual hue. To see what hue value has been generated we can attach a *debug node* to the *random node*, the value will be displayed in the debug tab.



Property	msg. payload
Generate	a whole number - integer
From	1
To	359
Name	Name

We need to introduce another new node that is going to be pivotal going forward and used in nearly every flow you make.

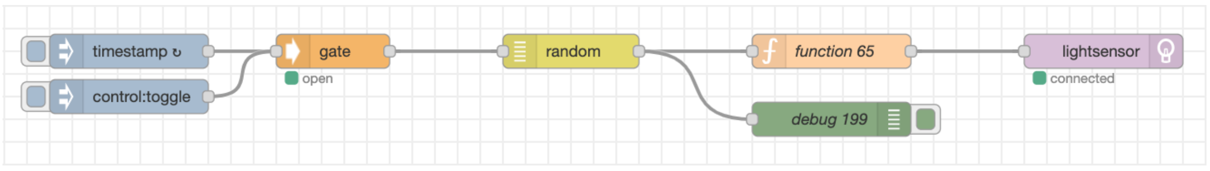
Function Node:
This node allows us to input code into our flow, making it more useful and specific.

Using the *function node* does require some coding ability, but don't worry if that sounds like an ability you do not possess (YET!) as we will walk you through the code needed to make our flows function.

Connect a *function node* to the *random node*, open it up and we will start on the code required.

```
1   let value = msg.payload
2
3   msg.payload = { "on": true, "bri": 100, "hue": (value) }
4   return msg;
```

This code is simpler than it may first seem. In line 1 we are setting the payload from the random node (the number for the hue) as a data element called **value**. In line 3 we are defining what the payload coming from this node forward will be, which for us is the data needed for the light strip, now including our variable. Once you've done this, connect the *MQTT node* and you should have a flow that resembles this.



If all is working correctly you can connect to the camera, and you should see the lights changing colours randomly. Though this example is simple, it demonstrates how you can take a variable set of data to apply an effect, what other uses could you think of for this kind of flow?

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, 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-Node-Red-Lessons