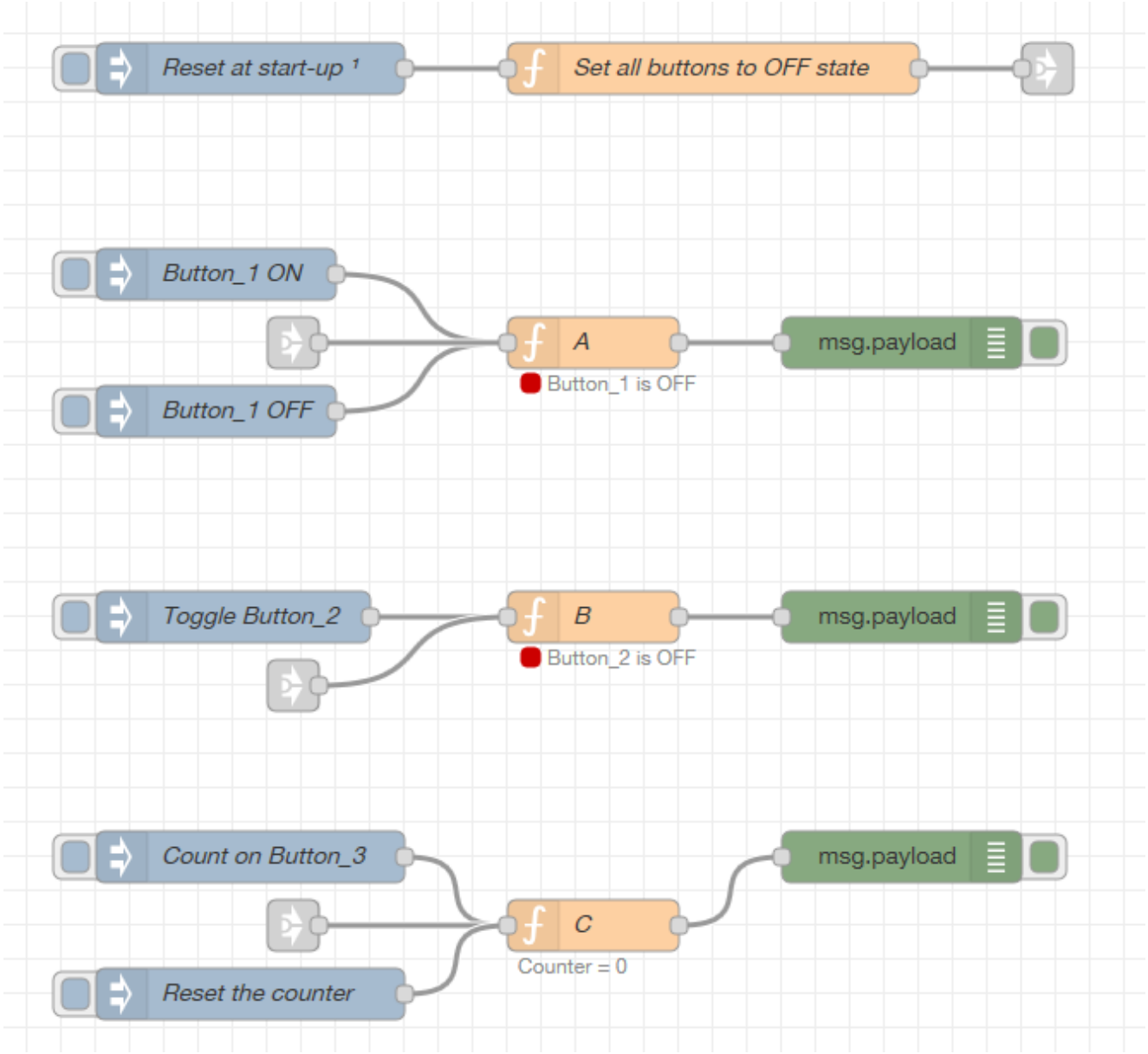
Bohunt School *(Wokingham)*

Having some FUN with Node-RED buttons

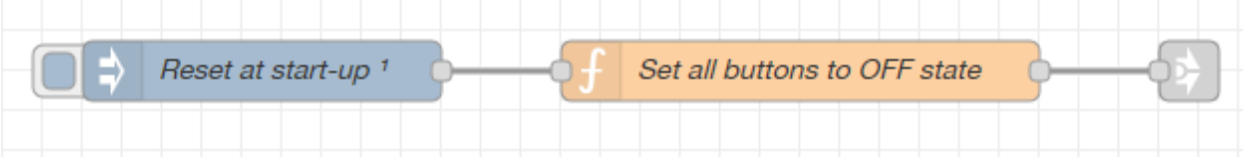
This practical session should be a bit of fun for you. It shows how to create different types of 'button actions' in a Node-RED flow

Complete Node-RED flow

Here's a screen-shot of the complete Node-RED flow.

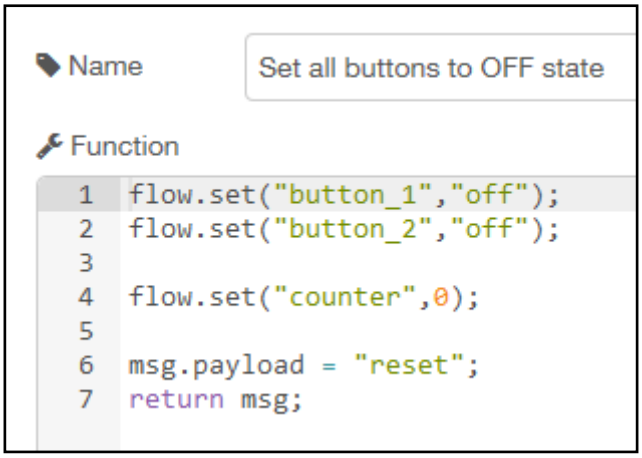


Although this flow may look rather complicated (it isn't), it can be subdivided into the Initial Reset section and the flows for various types of buttons.



The 'inject' node labelled 'Reset at start-up' ensures the 'function' node is executed when Node-RED is started or whenever a 'full redeploy' is performed.

Here's the listing of the JavaScript inside the 'function' node.



The series of 'flow.set' commands are used to store a value in each flow variable. The first two commands stores a text value into the variables, while the third command stores a numerical value into a variable.

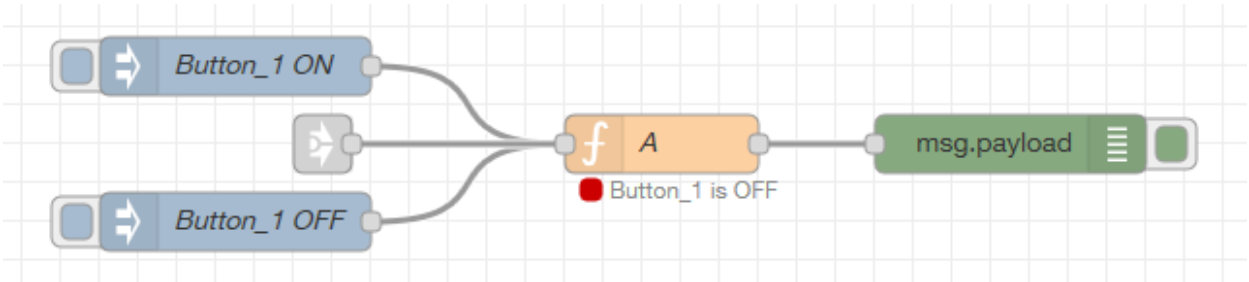
You may remember there are three types of variables: a variable declared with the keyword 'var' that only exists within a 'function node. As soon as you leave the 'function' node the variable is destroyed. A 'flow variable' (which appears in the JavaScript above) is available throughout the Node-RED flow in your current 'tab'. There is also a 'global variable' and as the name suggests - it is available throughout ALL your flows.

One thing to note is all three types of variables will be lost if the Raspberry Pi is rebooted or if there is a power-failure. Node-RED supports other forms of variables that will persist after a reboot/power failure. See Mr D for details.

The flow ends by sending "reset" as the message-payload out of the function.

Two buttons to create an ON OFF action

This flow shows two buttons that have an ON and OFF action.



A 'Link' node (from the flow on page-1) will cause the flow variable labelled "button_1" in node A to be set to the value "off". Pressing either of the 'inject' nodes will change the state of the flow variable.

Button_1 ON **sets the flow variable to "on" - I'll leave you to work out what you think happens when the *Button_1 Off* 'inject' node is pressed.**

Actually when either of the two 'inject' nodes are pressed they send a specific message-payload. *Button_1 ON* sends "on" and *Button_1 OFF* sends "off.

Here's a listing of the JavaScript inside 'function' node A.

Name A

Function

```
1  if ( (msg.payload == "off") || (msg.payload == "reset") ) {
2      flow.set("button_1","off");
3      node.status({fill:"red",shape:"dot",text:"Button_1 is OFF"});
4
5      msg.payload = 0; // Set this to what you want to pass on
6      return msg;
7  }
8  else if (msg.payload == "on") {
9      flow.set("button_1","on");
10     node.status({fill:"green",shape:"dot",text:"Button_1 is ON"});
11
12     msg.payload = 1; // Set this to what you want to pass on
13     return msg;
14 }
15
```

The first 'if' statement checks if the message-payload is "off" or "reset" and if it is true it will set the flow variable "button_1" to "off". A node.status command has been included to give an visual indication of what happens.

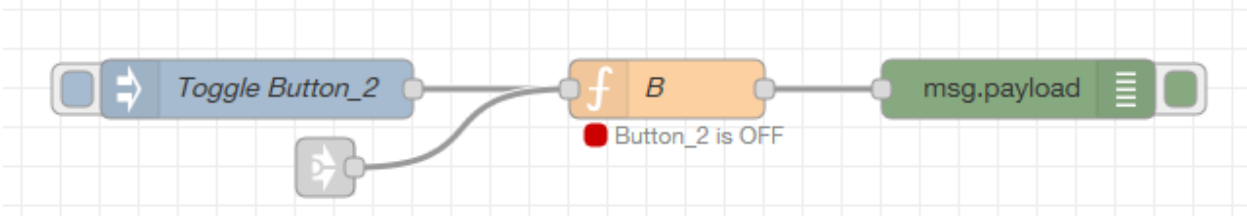
Line-5 will output a message-payload (in this case a numeric 0) - you can change this if a downstream flow needs a different value (e.g. "off").

Lines 8-14 do a similar check for the input message-payload equal to "on".

The above flow reacts to the two 'inject' nodes which are used to represent a button. You could replace then with a 'dashboard' button (see page-6).

Single button that performs a 'toggle' action

This flow shows a single 'inject' node that works as a toggle action. Each time you press the 'inject' node the state of 'Button_2' is inverted.



Here's a listing of the JavaScript inside 'function' node B.

Name B

Function

```
1  if (msg.payload == "change") {
2      var state = flow.get("button_2") || "off";
3      if (state == "off") {
4          state = "on";
5          node.status({fill:"green",shape:"dot",text:"Button_2 is ON"});
6          msg.payload = 1; // Set this to what you want to pass on
7      }
8      else if (state == "on") {
9          state = "off";
10         node.status({fill:"red",shape:"dot",text:"Button_2 is OFF"});
11         msg.payload = 0; // Set this to what you want to pass on
12     }
13     flow.set("button_2",state);
14     return msg;
15 }
16
17 else if (msg.payload == "reset") {
18     flow.set("button_2","off");
19     node.status({fill:"red",shape:"dot",text:"Button_2 is OFF"});
20     msg.payload = 0; // Set this to what you want to pass on
21     return msg;
22 }
23
```

The first 'if' statement checks if the incoming message-payload is "change" and if it is true it will read the state of the flow variable "button_2" (on Line-2). This is followed by another 'if' 'else' construct that inverts the flow variable. For example, if the state of the flow variable was "off" it is set to "on" and vice-versa.

The last part of the flow checks if the incoming message-payload is "reset". If it is, then this indicates the 'inject' node labelled *'Reset at start-up'* has been pressed or the Node-RED flow has been re-deployed. In either case the flow variable "button_2" is set to "off".

As in the previous piece of JavaScript you can change the value or type of the message sent out via the outgoing message-payload. (Lines 6, 11 and 20).

Index a counter each time a button is pressed

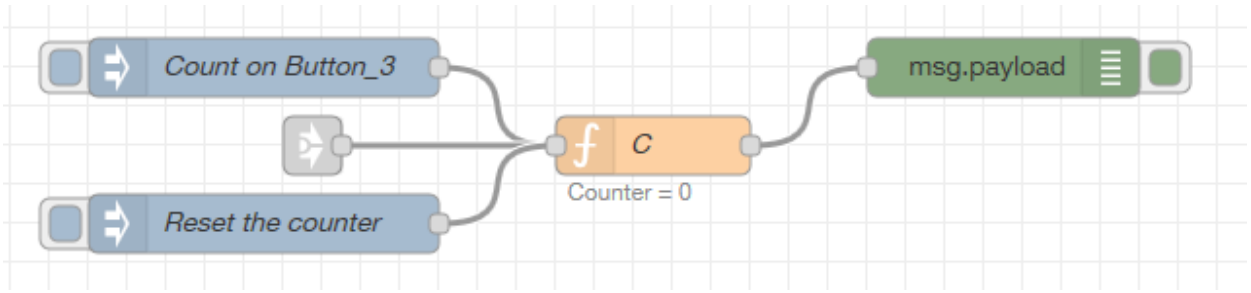
On the next page is a flow that will cause a decimal counter to be incremented each time the 'inject' node is pressed.

A second 'inject' node has been included to reset the counter.

Bohunt School *(Wokingham)*

Having some FUN with Node-RED buttons

Here's the flow for the decimal counter.



Here's a listing of the JavaScript inside 'function' node C.

Name C

Function

```
1  if (msg.payload == "increment") {
2      var counter = flow.get("counter") || 0;
3      counter = counter + 1;
4      node.status({text:"Counter = "+counter});
5      flow.set("counter",counter);
6      msg.payload = counter;
7      return msg;
8  }
9
10 else if (msg.payload == "reset") {
11     flow.set("counter",0);
12     node.status({text:"Counter = 0"});
13     msg.payload = 0;
14     return msg;
15 }
16
```

The first 'if' statement checks if the incoming message-payload is "increment" and if it is true it reads & increments the value of the flow variable "counter".

The second 'if' statement checks if the incoming message-payload is "reset" and if it is true it resets the flow variable "counter" to zero.

The value of the flow variable "counter" is sent out via the outgoing message-payload on Lines 6 and 13.

Dashboard buttons

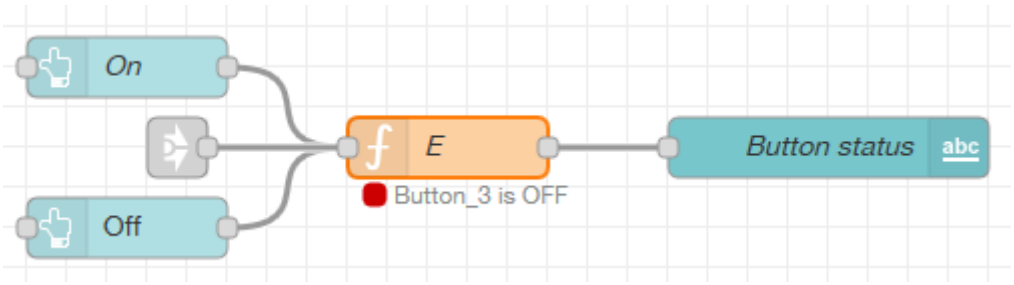
Up to this point a series of 'inject' nodes have been used to simulate the behaviour of one or more 'buttons' and the node.status command has be used to show the status of a variable (as in Lines 4 and 12 in the above JavaScript).

In real situations you would probably create a Dashboard so you could interact with your flow using graphical widgets like buttons, switches and sliders.

Bohunt School *(Wokingham)*

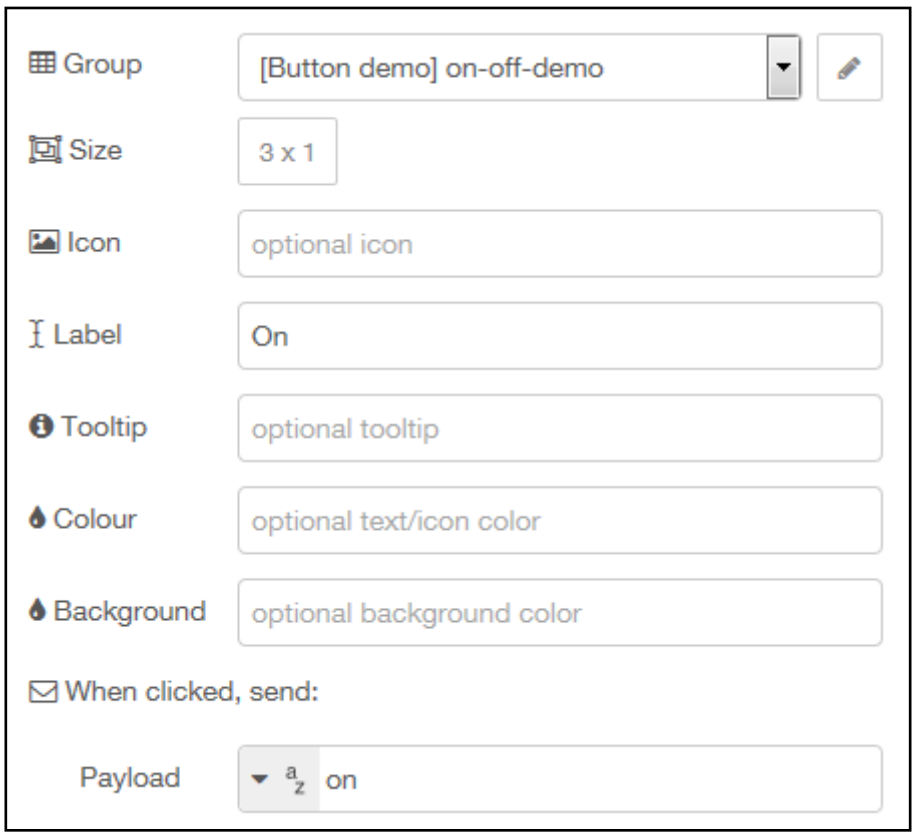
Having some FUN with Node-RED buttons

Here's a flow to show how the 'button widget' is used in the Dashboard.



If you compare this flow with the one that appears on page-2, you will see that they are very similar - in fact all that's different is the 'inject' and 'debug' nodes have been replaced with the 'button' and 'text' Dashboard widgets.

Just like the 'inject' nodes the 'button' widget can be set-up to send a specific message-payload when it is clicked. This is shown in the following screen-shot.



Have a look at the Payload and Label fields in the above screen-shot. The values can be altered to suit your particular application. Various other fields enable the size of a button, its iconic-shape, text colour and background colour to be defined.

If you have not used the Dashboard before, have a word with Mr D (or someone who has used the Dashboard) about how to create Dashboard 'tabs' and how to organise the stacking order for the widgets you decide to use.

Here's a listing of the JavaScript inside 'function' node E.

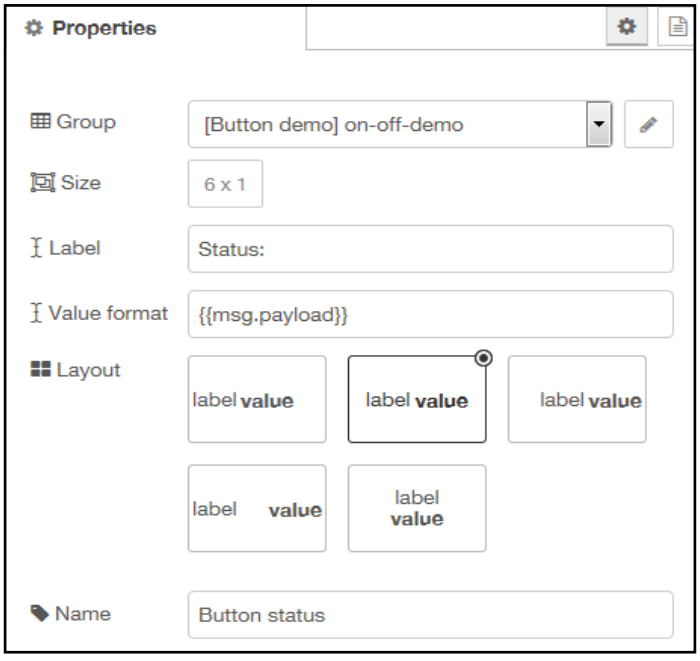
Name: E

Function

```
1  if ( (msg.payload == "off") || (msg.payload == "reset") ) {
2      flow.set("button_3","off");
3      node.status({fill:"red",shape:"dot",text:"Button_3 is OFF"});
4
5      msg.payload = "Button_3 is OFF"; // Set this to what you want to pass on
6      return msg;
7  }
8  else if (msg.payload == "on") {
9      flow.set("button_3","on");
10     node.status({fill:"green",shape:"dot",text:"Button_3 is ON"});
11
12     msg.payload = "Button_3 is ON"; // Set this to what you want to pass on
13     return msg;
14 }
15
```

If you compare this listing with the one that appears on page-3, you should see that they are very similar. In fact the only differences are the references to the name of the button (e.g. "button_3" rather than "button_1").

Take a look at lines 5 and 12 that define the outgoing message-payload. This information is sent to the 'text' widget (shown below) to show the status of the button.



Link to the Node-RED flow

Here's a LINK to a working solution (you can import into Node-RED) if you encounter any problems getting your flows to work correctly.

Congratulations you now know how to create some buttons.