Bohunt School *(Wokingham)*

Internet of Things (IoT) and Node-RED

This practical session should be a bit of fun for you. It involves learning how to control a SonOff WiFi-enabled mains switch.

SonOff S20 mains switch

Here's a photo of a SonOff S20 device.

The one shown here is for the UK market as it plugs into a square 3-pin domestic mains socket.

As can be seen in this photo, the front face of the device has a 3-pin socket into which you can plug all sorts of mains-driven devices.

For example, a table lamp, a TV or radio or even high powered devices like an electric kettle.

In this exercise we will use a table lamp.

Here's a rear view of the SonOff S20.

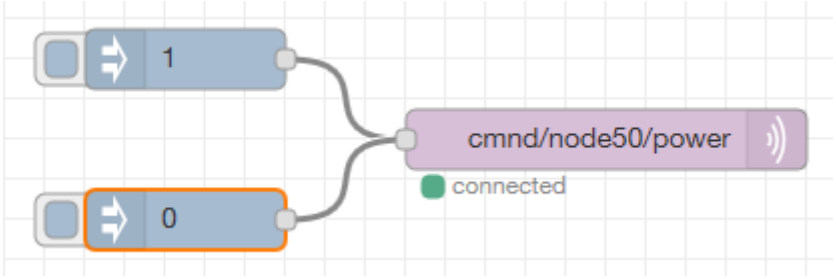
You can clearly see the 3-pin moulded plug.

This particular device has been re-flashed with the Tasmota firmware (rather than the ESP Easy firmware that is used with the WeMos D1 Minis).

The blue label shows the node as 'node50' this means it will have an IP address of 192.168.1.50

Switching the SonOff S20 device On and Off

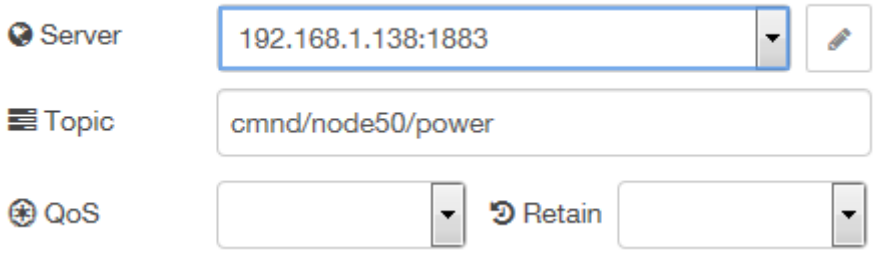
Here's a very simple/basic flow to turn the SonOff S20 On and Off.



The settings for the 'MQTT Output' node are shown on the next page.

MQTT settings

Double-click the MQTT node and check the settings match with this image.



You may need to alter the 'Server' setting to match your Raspberry Pi.

Check/enter the 'Topic' settings as: cmnd/node50/power

The 'Topic' specifies you are going to send a *command to node50 to control the relay that switches power to the 3-pin socket on the front of the device*.

If you look back to page-1 you will see that the msg.payload that comes from the 'inject' nodes and goes to the input of the 'MQTT' node is a 1 or a 0.

If the msg.payload is a 0 then this will turn the S20 off.

If the msg.payload is a 1 then this will turn the S20 on.

Testing time

Using the image on page-1 as a guide build the basic 'flow'. Then ask Mr D for a SonOff S20 and a table lamp.

Perform the following steps with extreme care - the mains can KILL you !!!

1. **Find a mains socket on the wall.**
2. **Make sure the socket is switched OFF.**
3. **Carefully plug the S20 into the mains socket.**
4. **Carefully plug the table lamp into the front face of the S20.**
5. **Then and only then, switch on the mains socket on the wall.**
6. **Click the 'inject' nodes and see if the table lamp comes ON goes OFF.**
7. **If you encounter any problems please seek assistance from Mr D.**

Single or Repeated MQTT messages

The 'flow' shown on page-1 uses two 'inject' nodes. Each time you click one of the 'inject' nodes a single MQTT message is sent to the SonOff S20. That means the amount of traffic over the network is very, very light.

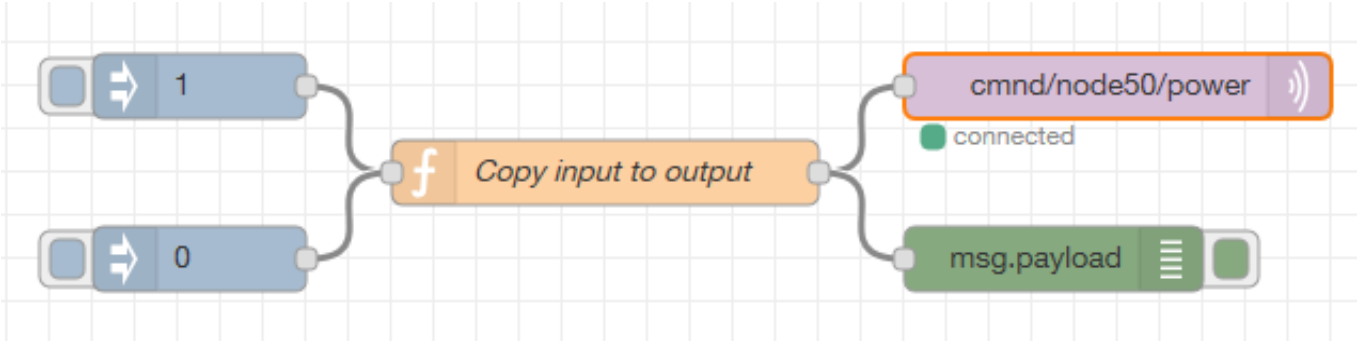
Let's assume you have decided to use the S20 with say the Ultrasonic Distance Measuring Device (see topic I3 for further details) to switch a light on when the ultrasonic beam is interrupted at a certain distance. What would happen in this situation is that a MQTT message would be sent on a regularly basis whilst an obstacle was detected by the beam. That means the amount of traffic over the network would increase.

So for example, if the Ultrasonic beam was triggered every second, then there would a corresponding MQTT transfer to the SonOff S20 device every second. This means there could be lots of transfers telling the S20 device to turn-on when it has already been switched-on by a previous MQTT transfer.

Likewise, there could be lots of transfers telling the S20 device to turn-off when it has already been switched-off by a previous MQTT transfer.

Using a variable to remember the S20's state

Using the following image as a guide, construct this Node-RED flow.



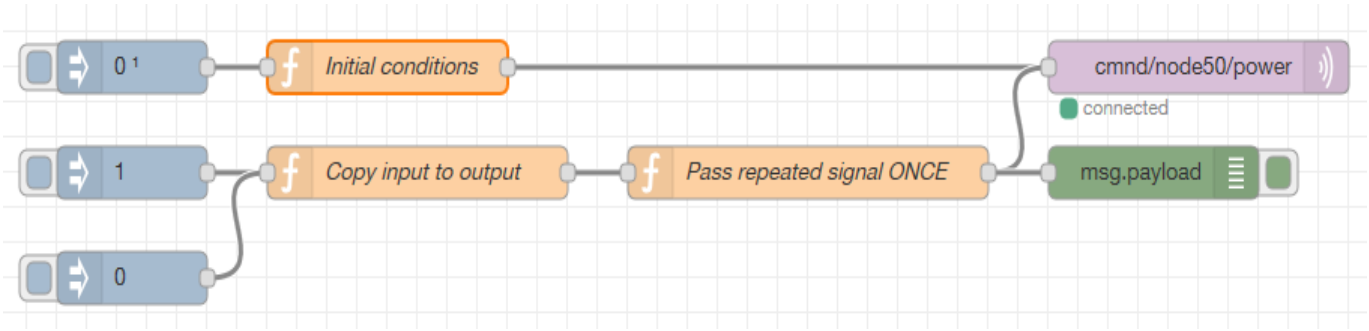
This is a simple test-bench to show that every time an 'inject' node is clicked a signal is sent to the 'MQTT Out' to operate the SonOff 320.

The screen-shot on the right shows the activity on the green 'Debug' node. In this situation it is showing that a numerical '1' is coming out of the 'function' node every time the top 'inject' node is clicked.

This means that repeated commands are being sent to the S20 - when in fact we only need to instruct the device to turn on once.

Note: The 'function' node just returns the original message.

Using the following image as a guide, modify your Node-RED flow.



The 'function' node at the top of the flow sets the initial conditions.

Initial conditions

Function

```
1  flow.set("sonoff_s20_status","off");
2  msg.payload = 0;
3  return msg;
4
5
```

The flow-variable named 'sonoff_s20_status' is assigned the text string "off". This happens as soon as the 'flow' starts after you deploy your new flow.

Note:
There are three types of variables. The first type is a local variable that exists in a piece of JavaScript. You can spot it by looking for the keyword 'var'.

The next type of variable is the 'flow-variable'. This type of variable exists within the current flow. It is not known on any other flows.

The third type of variable is the 'global-variable'. As its name implies it can exist across all flows - very useful if you need to pass information from one flow to another.

If you turn the Raspberry Pi off, then the values stored in any of the above variables will be lost. A later topic will explain how to overcome this problem.

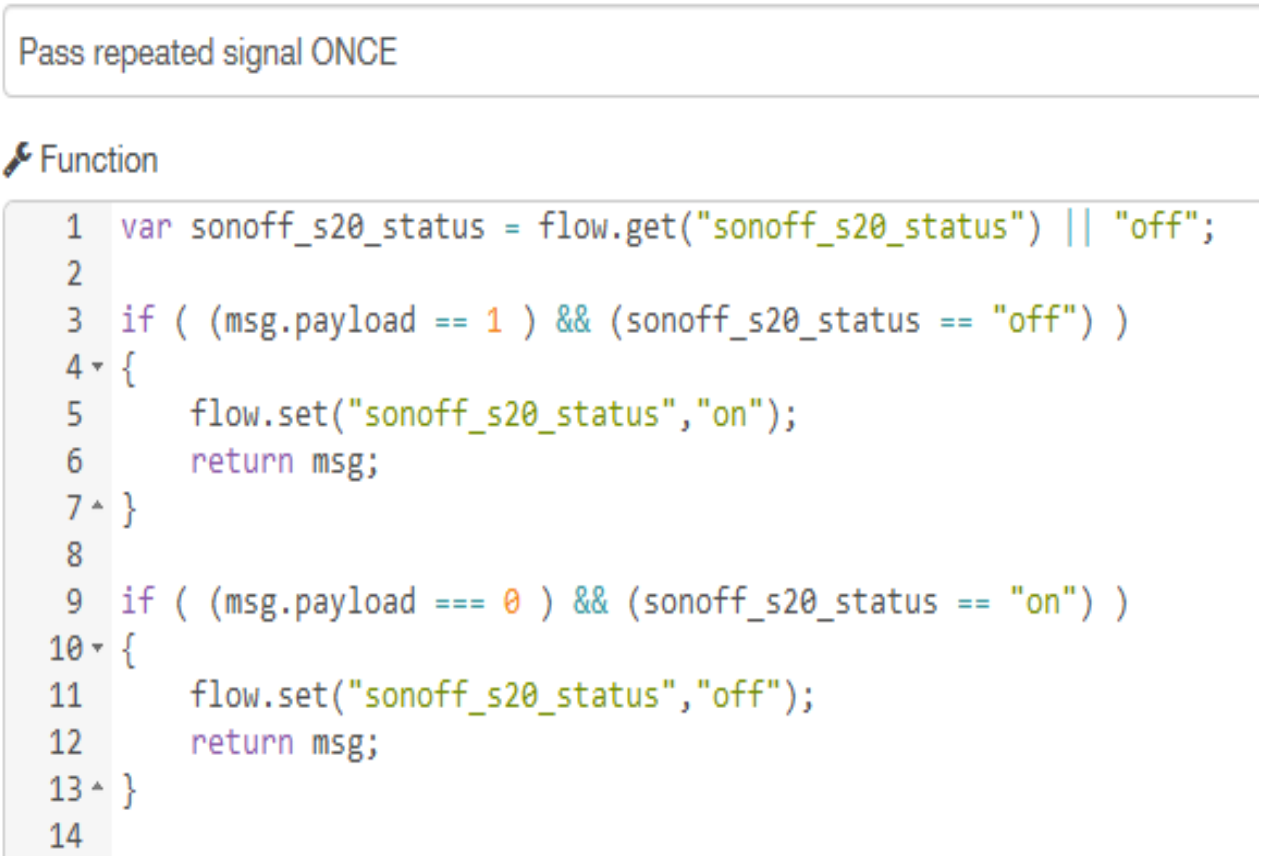
The 'function' node named 'Copy input to output' does exactly that.

Function

```
1  return msg;
```

Although this 'function' node was needed in the original flow (to join the two wires together that emerge from the 'inject' nodes), it's not really needed in this situation and could be omitted.

The JavaScript within the 'Pass repeated signal once' node is shown below.



Line 1 reads the value of the 'flow-variable' "sonoff_s20_status" into a local variable also called 'sonoff_s20_status'. If for some reason the variable did not exist, then it is set to the default value of "off".

Line 3 checks two conditions. The incoming msg.payload and the value of the variable called 'sonoff_s20_status'. If the incoming message is a '1' AND the status is "off", then Line-5 sets the flow-variable "sonoff_s20_status" to "on". Line-6 sends the message payload (which is equal to 1) to the 'MQTT Input' node. Please note that 'msg.payload' will retain its value from Line-3.

Note: JavaScript uses the double && symbols to indicate the AND operation.

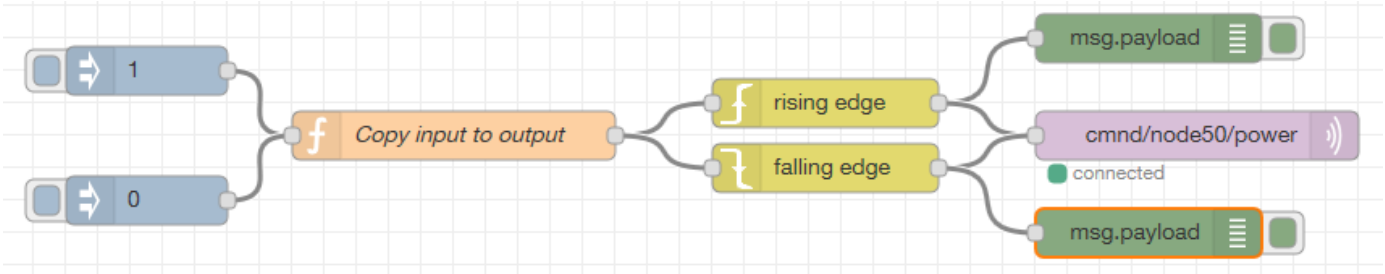
The JavaScript code on lines 10 to 15 perform a similar function except it is checking if the incoming msg.payload is 0 and the status is "on".

Please have a go at understanding the JavaScript, but if you get stuck have a word with Mr D.

Next thing to do is test that the flow performs correctly. You should be able to repeatedly click one of the 'inject' nodes and by looking at the 'debug' node see that only one signal is generated. You should also be able to repeatedly click the other 'inject' node and see that only one signal is generated.

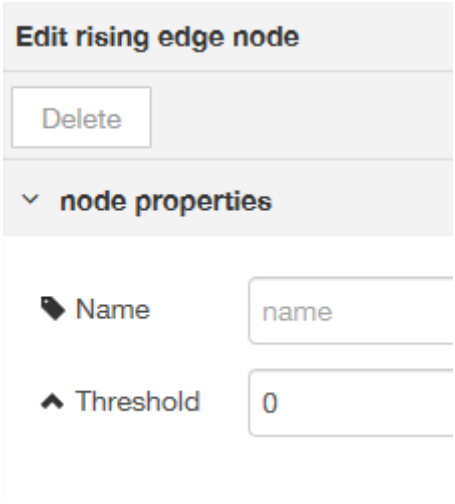
Using the rising edge and falling edge detectors

Using this image as a guide, construct the following Node-RED flow.



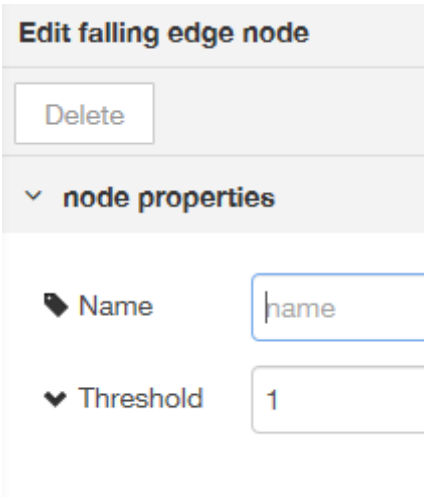
The mustard coloured 'rising edge' and 'falling edge' nodes can be found in the palette category named 'functions'.

The setting for the rising edge detector is shown below.



The 'Threshold' is set to '0'. This means if a signal higher than this value appears on the input, then it will be passed to the output.

Likewise, the 'Threshold' setting for the falling edge detector is set to a '1'.



This means if a signal lower than this value appears on the input, then it will be passed to the output.

Using the two detectors in parallel means you can detect both the rising and falling edges.

The next and final step is to test your flow and make sure it gives the same results as the flow you made on page-4.

Congratulations you now know how to control a SonOff S20 mains switch.