Home Assistant

Node-RED

dScript

User Manual Version 4.12

Table of Contents

Documentation history

v4.12	First Version

Scope of this Document

The document will show how to control the dS range of relay modules using Home Assistant and NodeRED. We will use the dS3484 in this example although there is nothing specific to that module. This document applies to the full range of dScript modules.

This document assumes you have Home Assistant with the NodeRED integration already setup. If that is not the case then you will need to install those. Search YouTube for Home Assistant and NodeRED, there are a lot of videos there to help you.

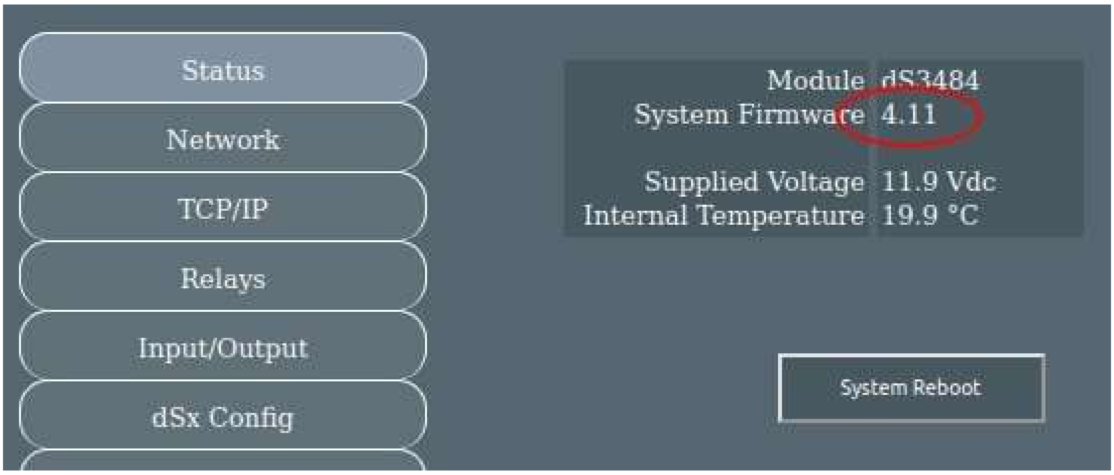
Your Home Assistant/NodeRED server should always have the same IP address. Either set this up as a static IP or set your Router/DHCP server to always assign the same address to Home Assistant.

We are using the latest version of home assistant at the time of writing – 2023.2.0

Getting started

Start by taking your dScript module (dS3484 in this case) and plugging in the Ethernet cable to connect the to your network, the USB cable to your computer and the 12v jack plug from your adapter. Switch on and wait for the blue LED to flash 3 times. Load up the dScript Editor and go to Module→IP Address to get the modules IP address. Then in your browser go to <IP Address>/_config.htm. For example on our computer:
192.168.1.146/_config.htm

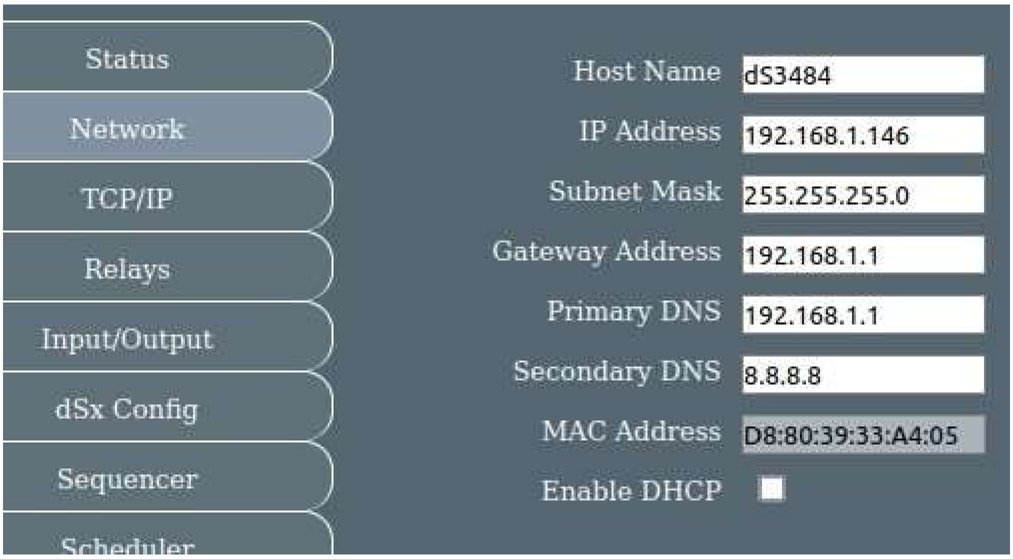
Now check that you have version 4.11 or later installed. If not you will need to update the firmware following the instructions in your modules user manual.



Module	dS3484
System Firmware	4.11
Supplied Voltage	11.9 Vdc
Internal Temperature	19.9 °C

Configuring the dS module

Go to the Network Configuration Tab and set a static IP for the dS module. In our case we have chosen 192.168.1.146



Host Name dS3484

IP Address 192.168.1.146

Subnet Mask 255.255.255.0

Gateway Address 192.168.1.1

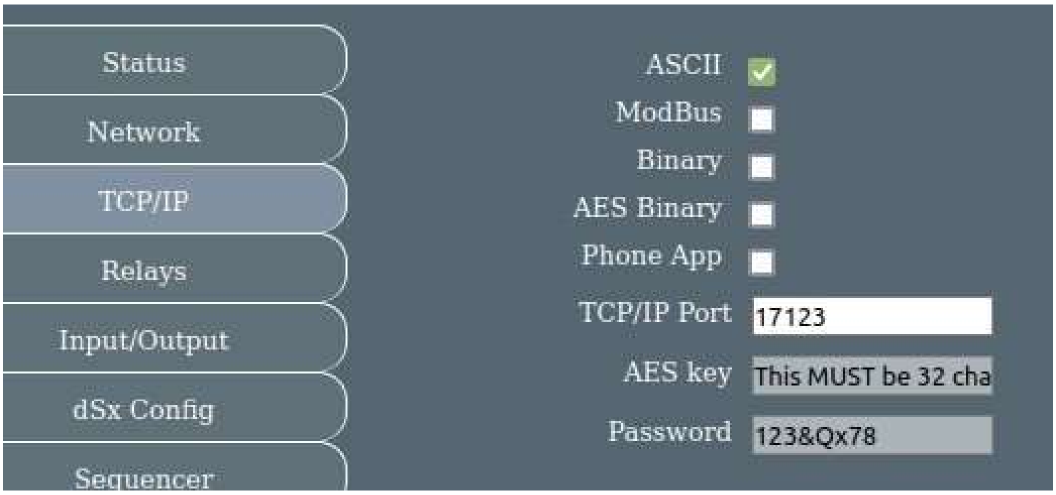
Primary DNS 192.168.1.1

Secondary DNS 8.8.8.8

MAC Address D8:80:39:33:A4:05

Enable DHCP □

Next go to the TCP/IP tab and select ASCII for the control protocol and the port number the dS module will listen on. This will be used to control the relays.



ASCII ☑

ModBus □

Binary □

AES Binary □

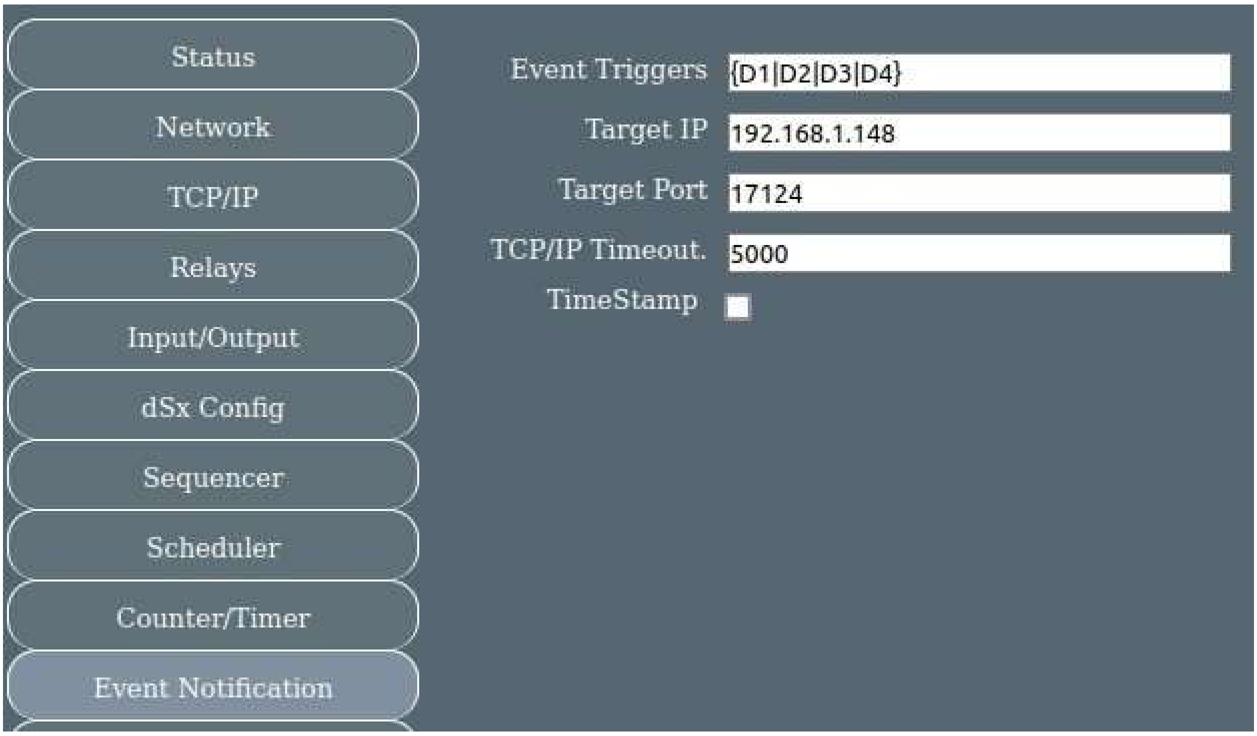
Phone App □

TCP/IP Port 17123

AES key This MUST be 32 cha

Password 123&Qx78

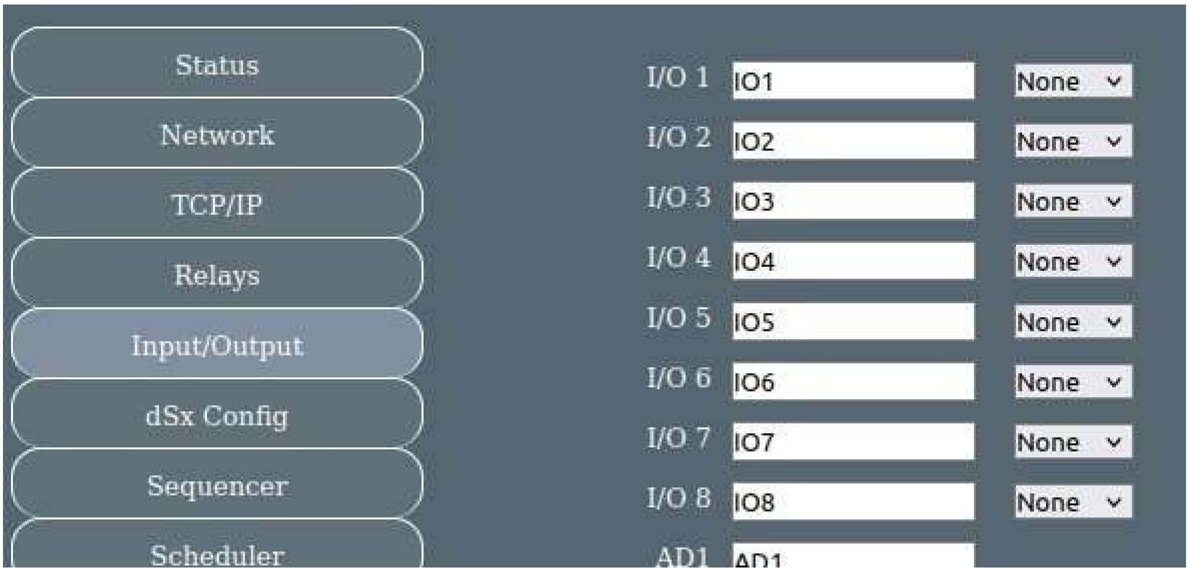
On the Event Notification tab, select the events you wish to trigger a notification message to be sent to Home Assistant/NodeRED.



Event Triggers {D1|D2|D3|D4}
Target IP 192.168.1.148
Target Port 17124
TCP/IP Timeout. 5000
TimeStamp □

In this case we want the first four inputs to trigger a notification message.
The target IP is the address for your Home Assistant/NodeRED server, in our case it is at 192.168.1.148. The port is whatever you wish to use for notification messages.
Leave the timeout at 5000 (5 seconds). Reducing it does NOT speed up anything.
The TimeStamp box should be unchecked as Home Assistant has comprehensive data logging already.

On the Input/Output tab, make sure the link boxes are set to None.

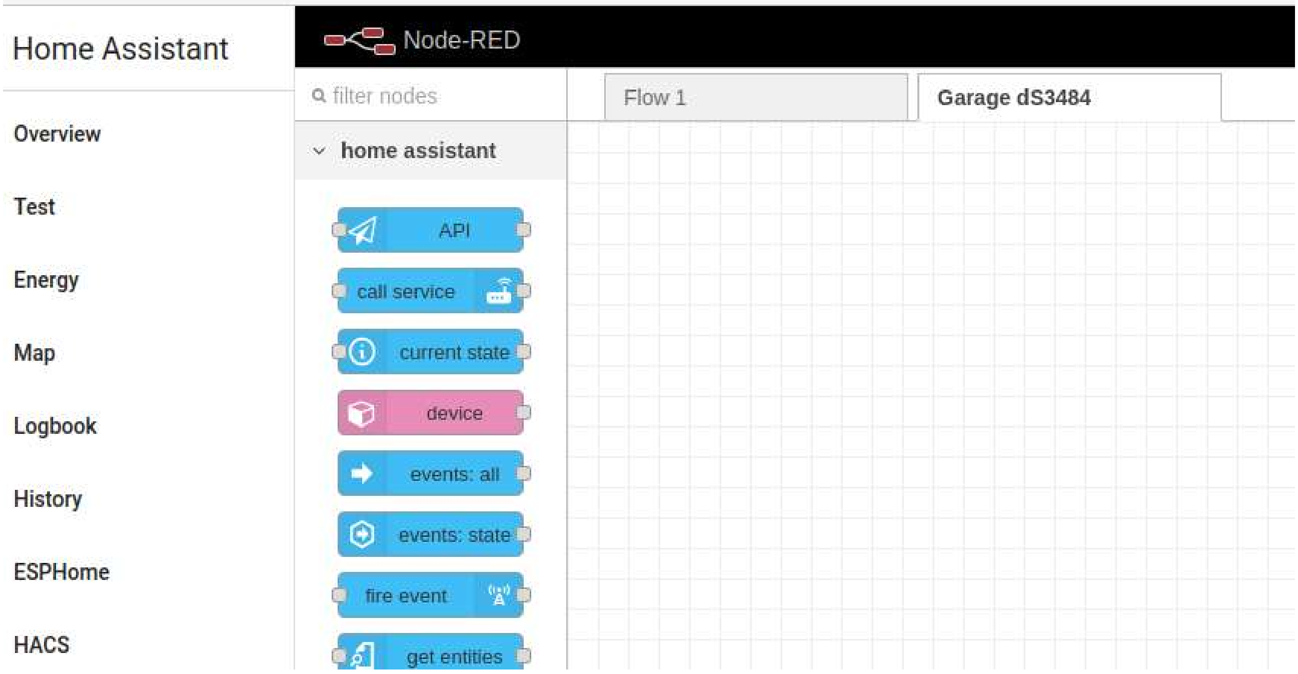


I/O 1	IO1	None
I/O 2	IO2	None
I/O 3	IO3	None
I/O 4	IO4	None
I/O 5	IO5	None
I/O 6	IO6	None
I/O 7	IO7	None
I/O 8	IO8	None
AD1	AD1	

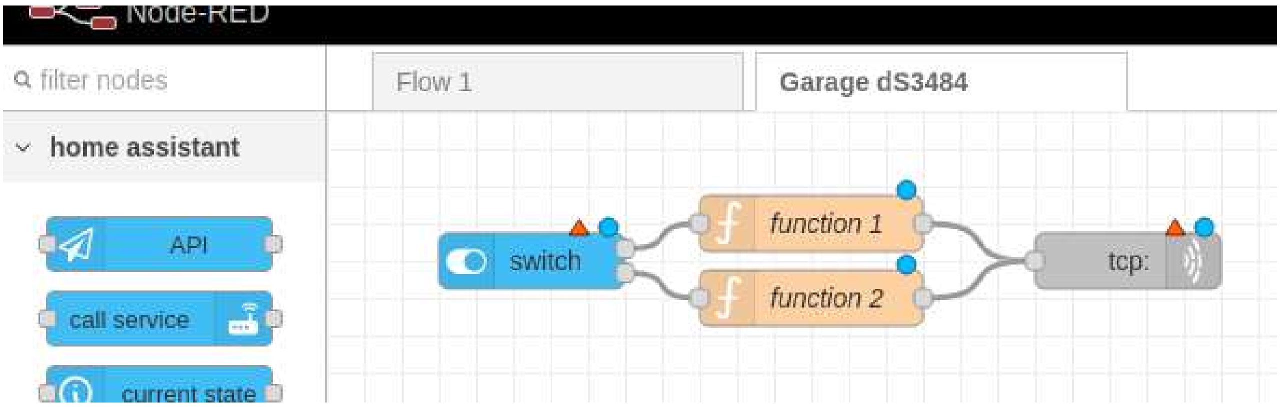
Wait for the red Update Pending indicator to go out and reboot the dS module. You will then find it at the new static IP address you just set up.

Configuring the Relays in NodeRED

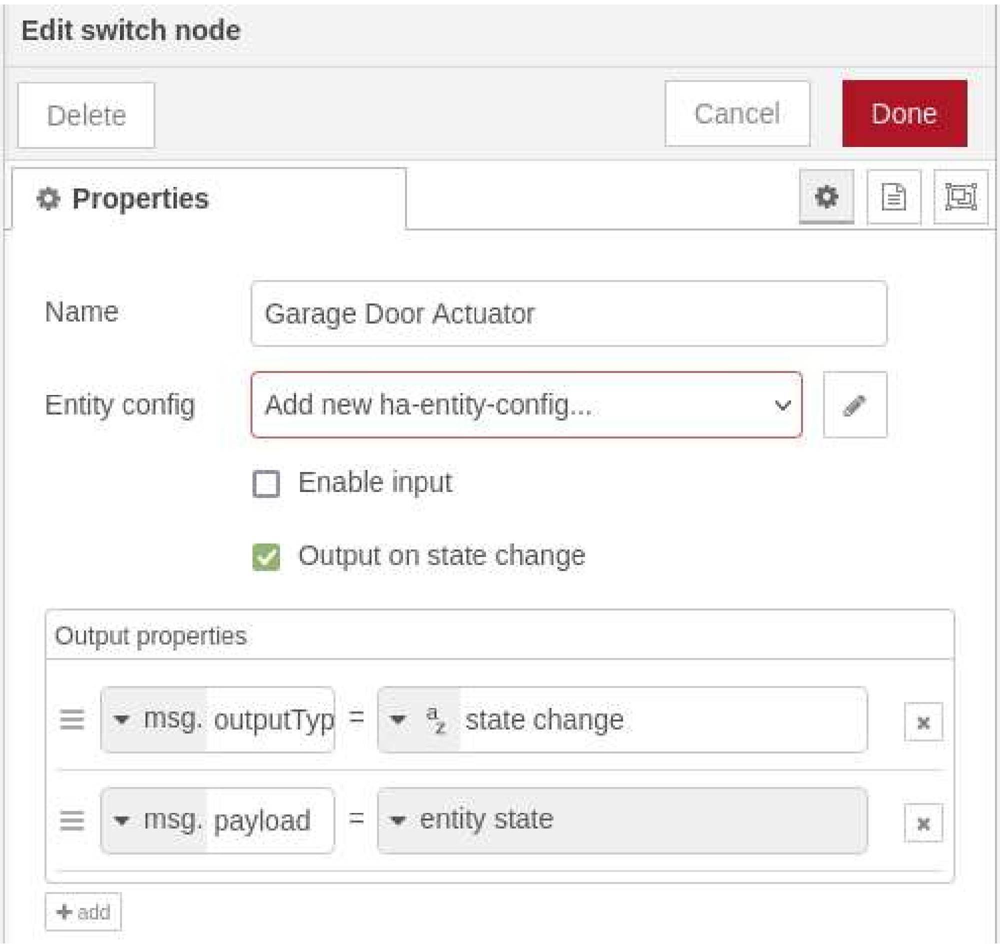
In Home Assistant, select NodeRED, and create a new flow. Rename it as you please, here we have called it Garage dS3484.



The first thing we will do is get a relay working. From Home Assistant Entities, drag a switch node onto the flow. From Function, drag a couple of function nodes and from Network drag a tcp out node. Connect them like this:



Now we configure these nodes. Double click the switch node to bring up the edit form. Call this Node “Garage Door Actuator” and click the output on state change box.



Edit switch node

Name: Garage Door Actuator

Entity config: Add new ha-entity-config...

□ Enable input

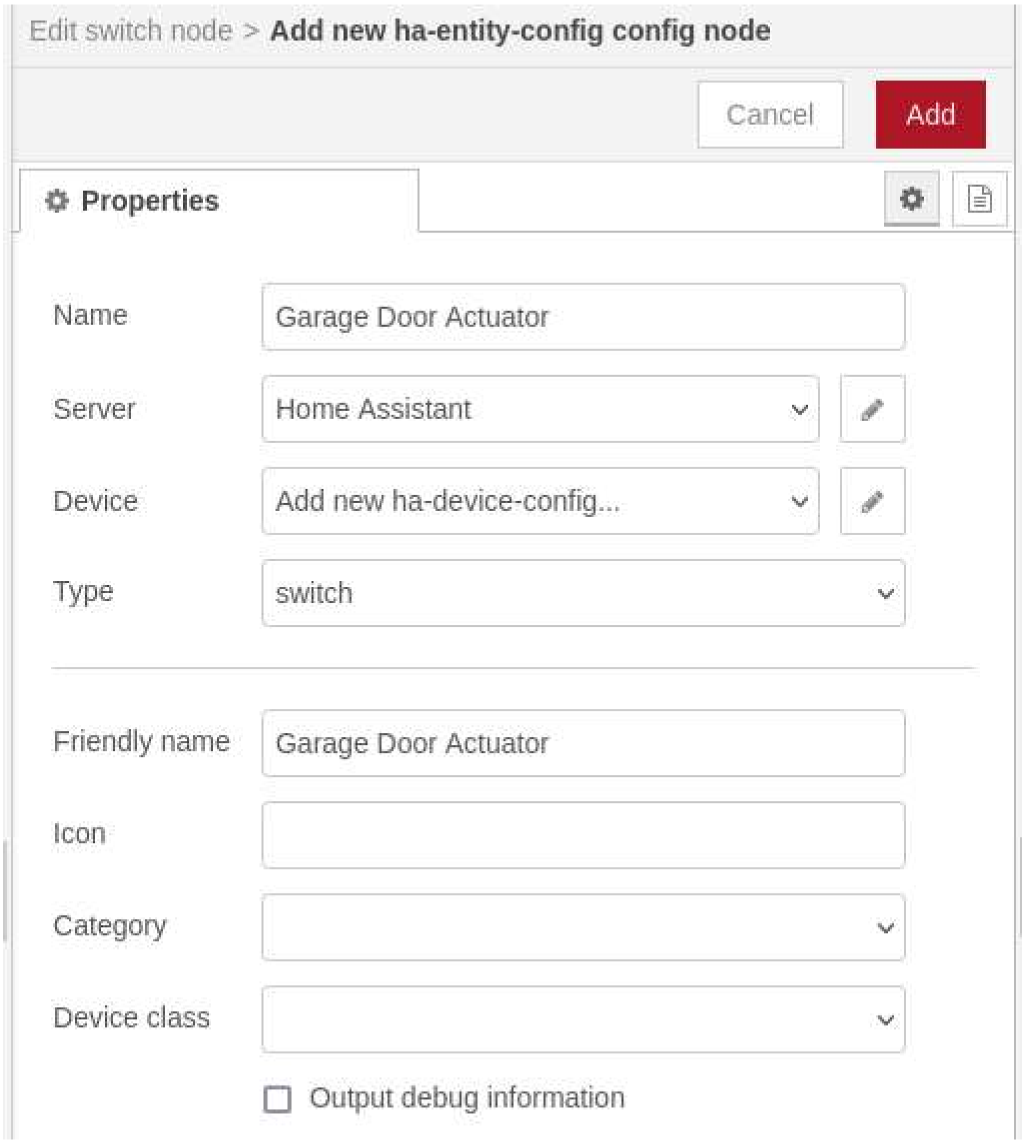
☑ Output on state change

Output properties

msg. outputTyp = state change

msg. payload = entity state

Next we will create the entity and as this is the first one, the device as well.
Click the edit symbol beside the Add new ha-entity-config to bring up the following entity config page.



Edit switch node > **Add new ha-entity-config config node**

Name: Garage Door Actuator

Server: Home Assistant

Device: Add new ha-device-config...

Type: switch

Friendly name: Garage Door Actuator

Icon:

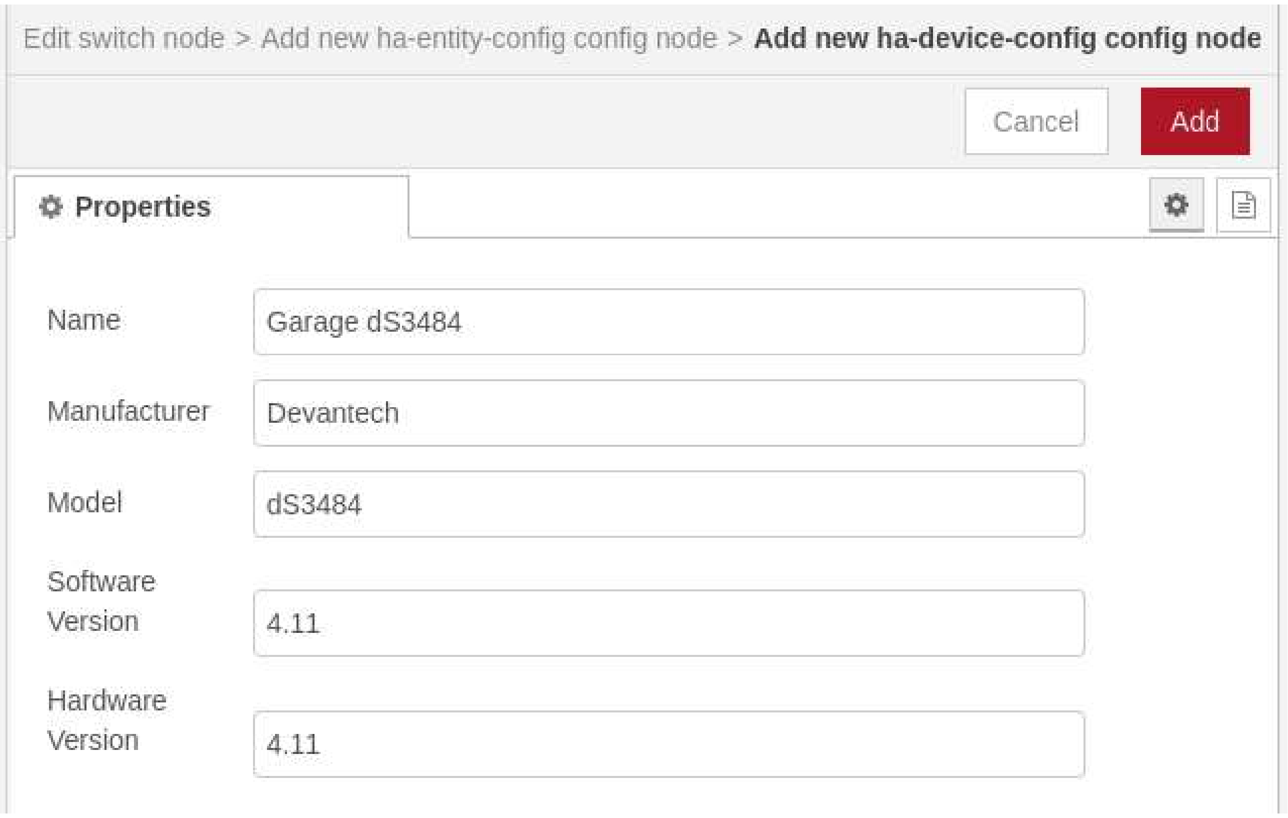
Category:

Device class:

□ Output debug information

Select the entity name and the friendly name, in this case the same as our node name.
As this is the first entity for this device (the dS3484), we will also create the device. Click on the edit symbol next to the Add new ha-device-config box.

Enter the following device details and click Add.



Edit switch node > Add new ha-entity-config config node > **Add new ha-device-config config node**

Name: Garage dS3484

Manufacturer: Devantech

Model: dS3484

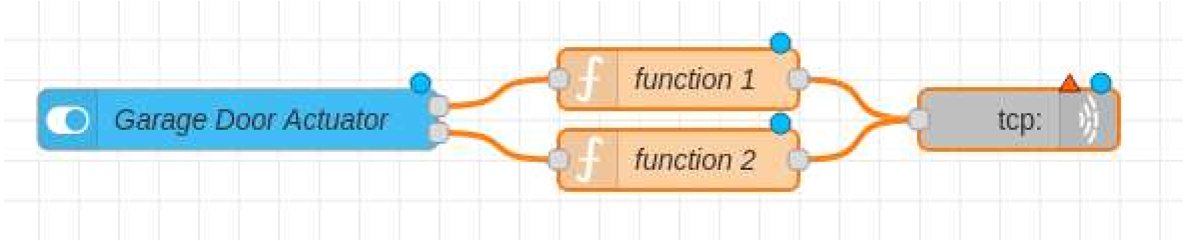
Software Version: 4.11

Hardware Version: 4.11

Now you are back to the entity config page, click Add.

And now you are back to the node config page, click Done.

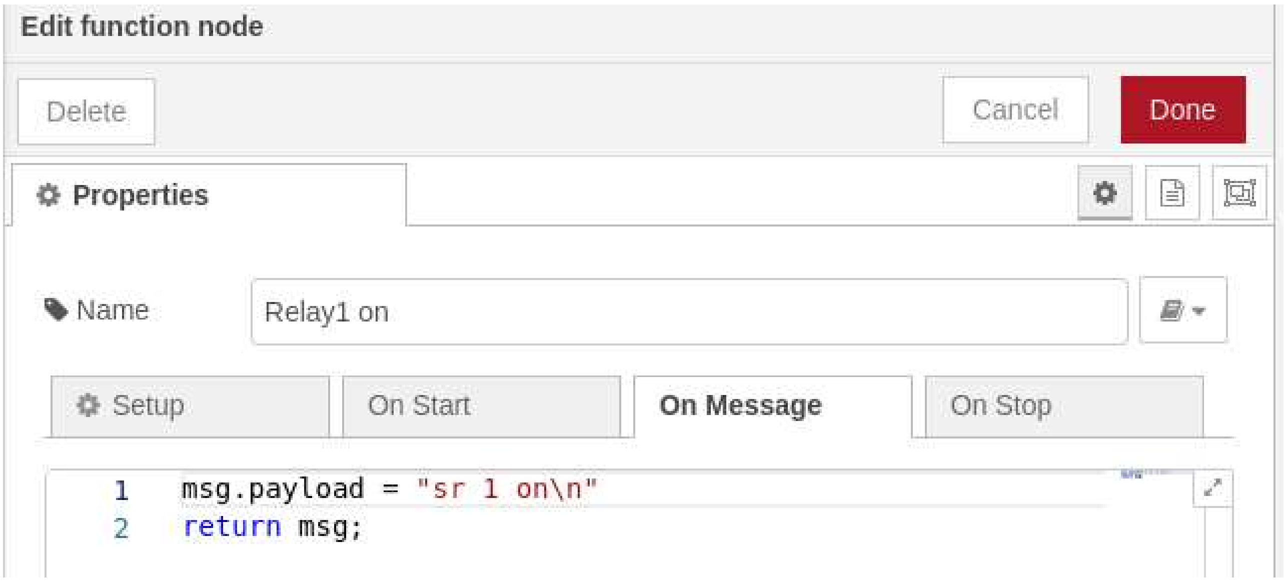
Your flow now looks like this:



Don’t deploy just yet, we will configure the function nodes next.

The purpose of the function nodes is to translate the boolean true and false signals coming from Home Assistant through the switch node into the ASCII commands the dScript module requires.

Double click the function1 node. Give the node a descriptive name such as Relay1 on. In the code window, set the msg payload to our dScript ASCII command, “sr 1 on\n”.



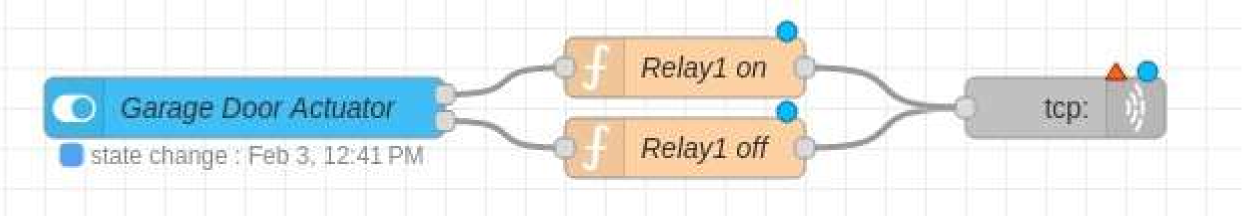
Edit function node

Name: Relay1 on

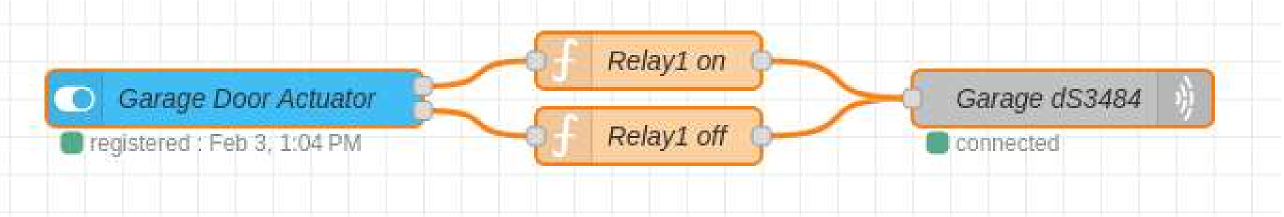
```
msg.payload = "sr 1 on\n"
return msg;
```

Click Done.

Do the same for function2, changing on to off in both the name and the code. The flow should now look like this.

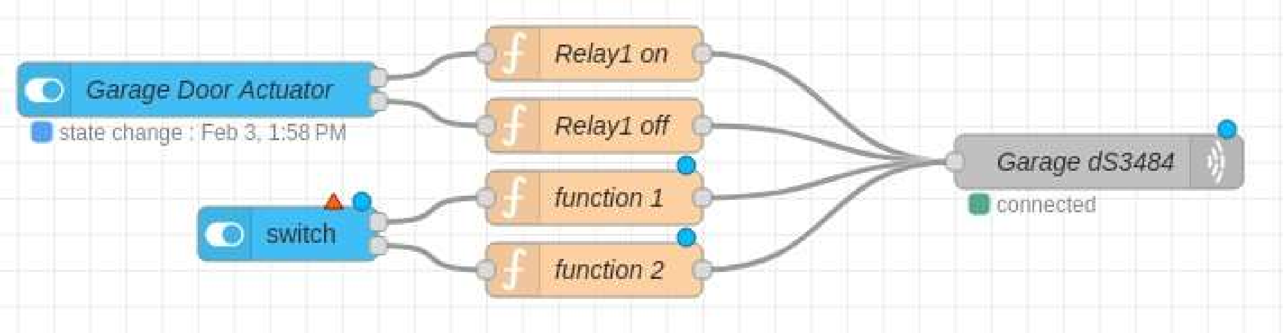


Next we configure the tcp node. Double click it and enter the same port number as on the dScript modules TCP/IP tab, 17123 in our case.
Enter the same static IP as you set up on the dScript modules Network tab, 192.168.1.146 in our case. Set the node name to describe the device.
Click Done. Click Deploy.

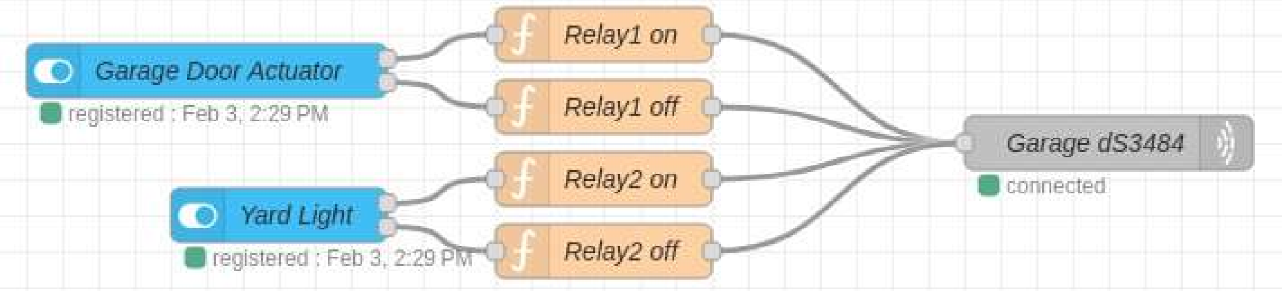


At this point you can test the relay.
Go to home Assistant→Settings→Devices & Services→Integrations and select entities on the Node-RED Companion window. There you will see the Garage Door Actuator listed. Click on it and select the INFO tab. There you will see a slide switch which will turn your Relay On/Off on the dScript module.

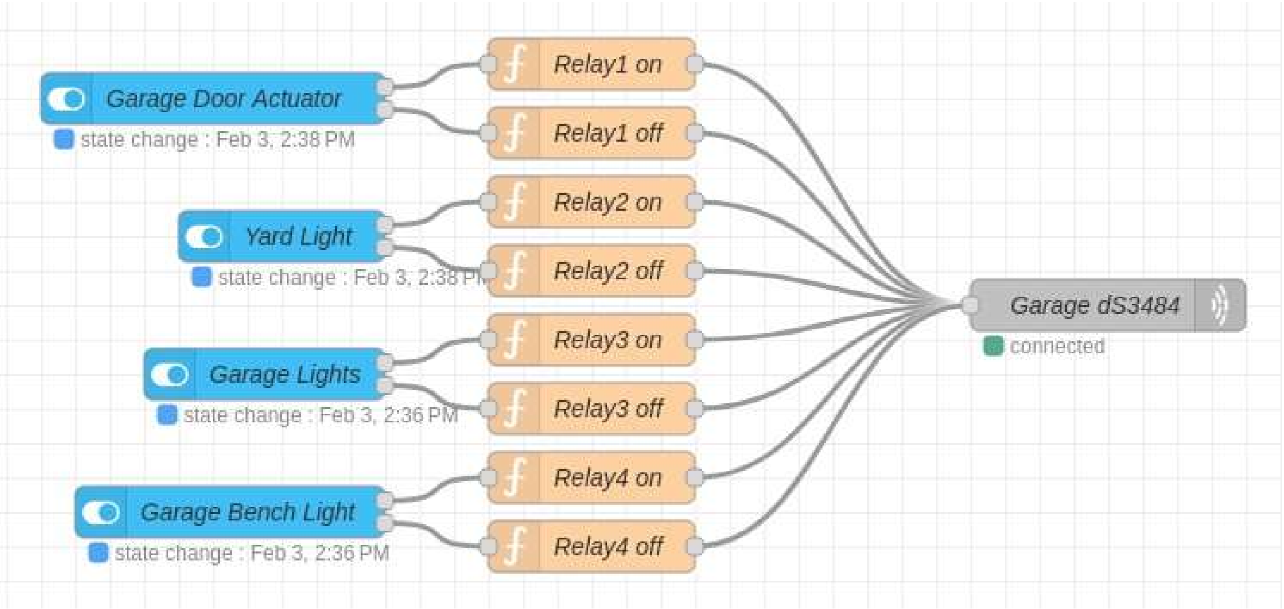
OK. Back to NodeRED. We will add another Relay, this time for a Yard Floodlight outside the garage. Drag in another switch node and two more function nodes. Connect them to the existing tcp node.



Set up the switch node as before, this time calling it Yard Light. You won't need to create the device this time, you will find the Garage dS3484 listed in the device dropdown box. Click Deploy when done and test as before.



You can add two more relays the same way, calling them Garage Lights and Garage Bench Light.

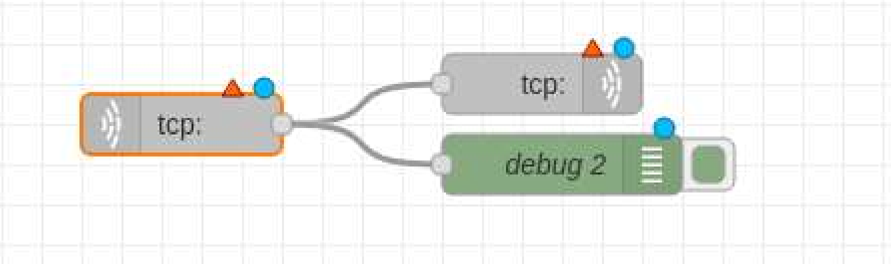


Adding Inputs

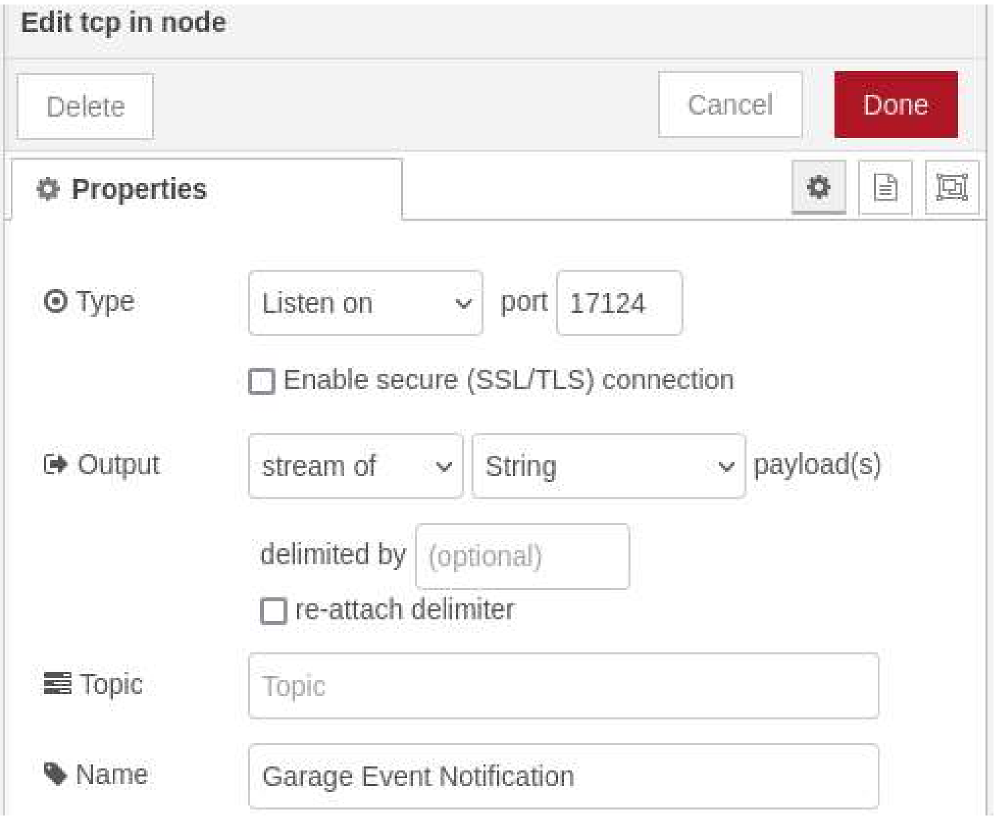
In the dScript modules Event Notification tab, we selected D1, D2, D3 and D4 (Inputs 1-4) as the notification trigger. This means if any of those inputs change state a notification message is sent to NodeRED.

Lets have a look at that message and see exactly what we get. Add the following below the relays.

A tcp in node, a tcp out node and a debug node.



Double click the tcp in node and enter the same port number you entered on the dScript modules Event Notification tab, 17124 in our case. Change the Output from Buffer to String. Set a descriptive name for the node and click Done.



Edit tcp in node

Type: Listen on port 17124

□ Enable secure (SSL/TLS) connection

Output: stream of String payload(s)

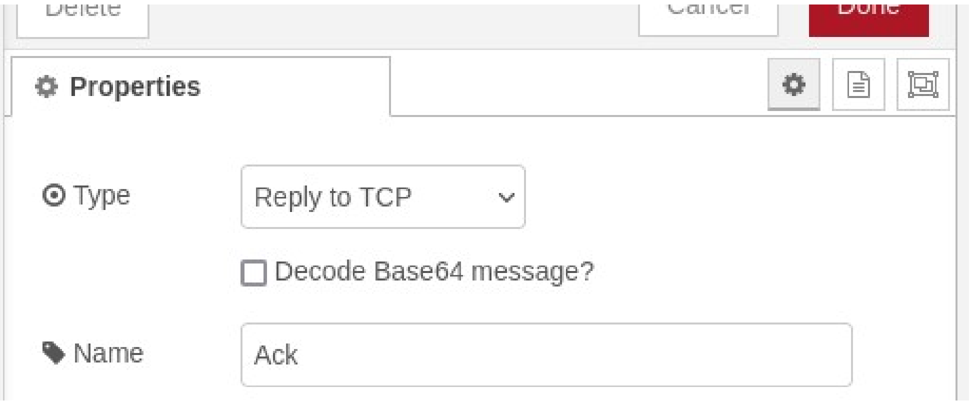
delimited by (optional)

□ re-attach delimiter

Topic: Topic

Name: Garage Event Notification

Select the tcp out node and set the type to Reply to TCP. Name the node as Ack. The dScript modules require a response when they send out a TCP message, this provides that response.



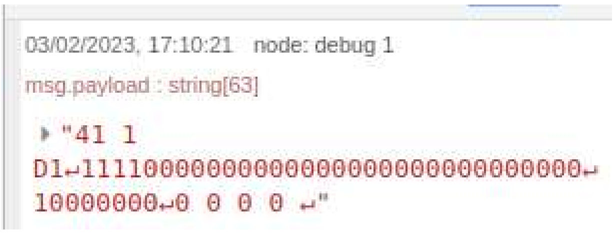
Type: Reply to TCP

□ Decode Base64 message?

Name: Ack

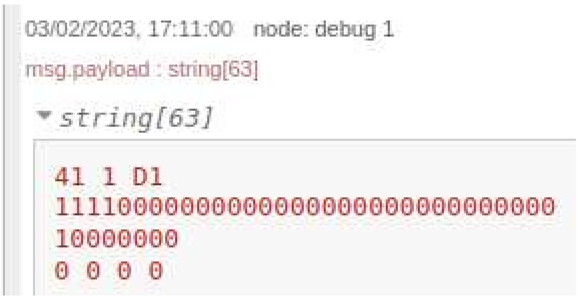
Deploy the flow.

Select the debug tab on NodeRED and clear it. Now apply input 1 on the dScript module and you should see the following response.



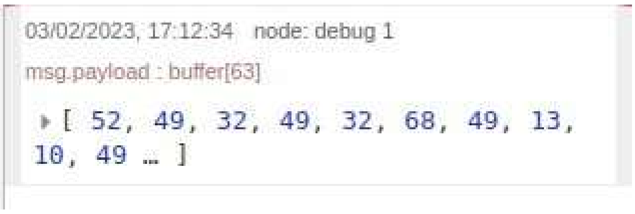
```
03/02/2023, 17:10:21   node: debug 1
msg.payload : string[63]
▸ "41 1
D1↵11110000000000000000000000000000↵
10000000↵0 0 0 0 ↵"
```

Click on the numbers to format them in a more readable way.



```
03/02/2023, 17:11:00   node: debug 1
msg.payload : string[63]
▾ string[63]
41 1 D1
11110000000000000000000000000000
10000000
0 0 0 0
```

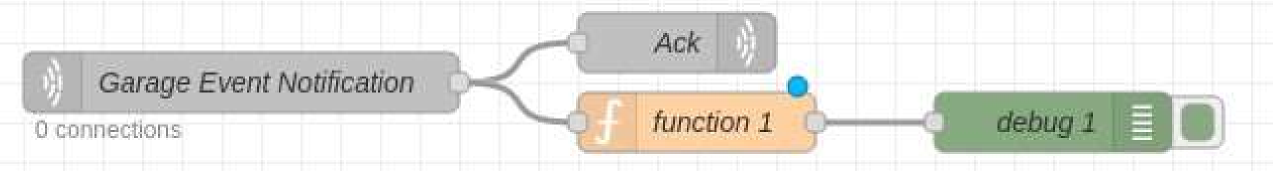
If you see this:



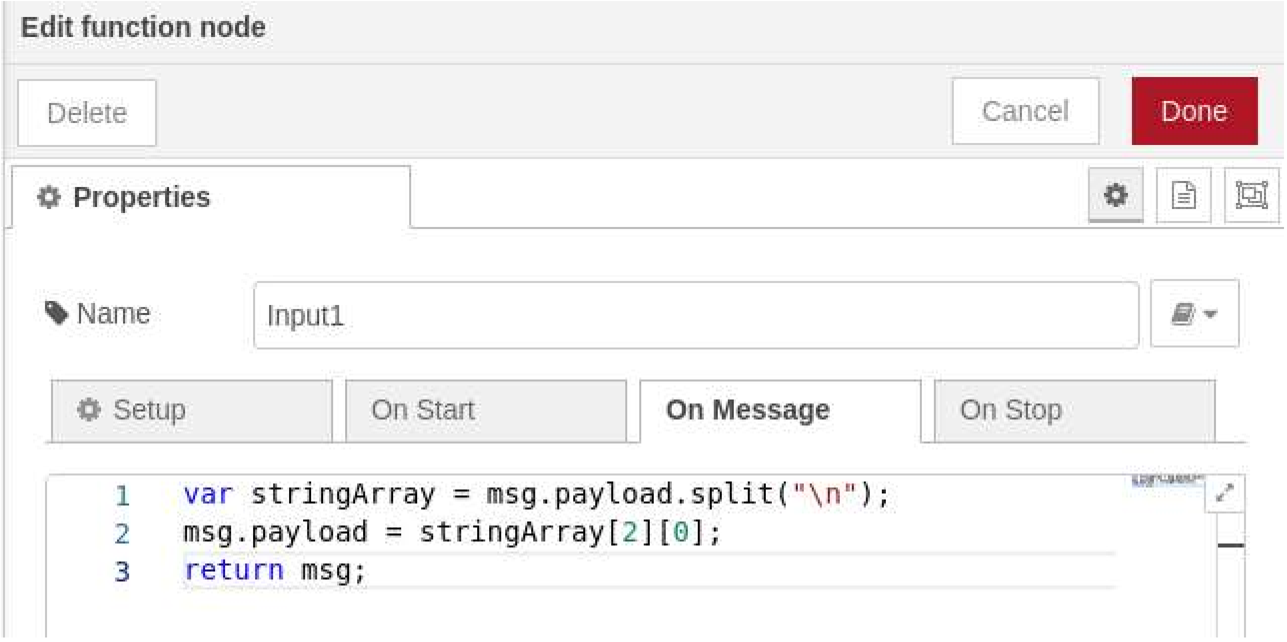
```
03/02/2023, 17:12:34   node: debug 1
msg.payload : buffer[63]
▸ [ 52, 49, 32, 49, 32, 68, 49, 13,
10, 49 … ]
```

go back to the tcp in node and check you have Output set to stream of String (not buffer).

See the dScript modules manual for information on what all these numbers mean. The ones of interest here are the 8 digits on the third line. These are, left to right, inputs 1 to 8. We will need to extract these from the data string. This is done with a function node, add one in like this:



Name the function Input1 and add the follow code:

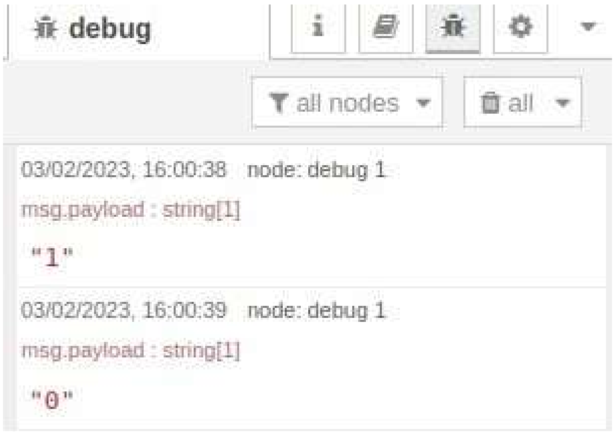


Name: Input1

```
var stringArray = msg.payload.split("\n");
msg.payload = stringArray[2][0];
return msg;
```

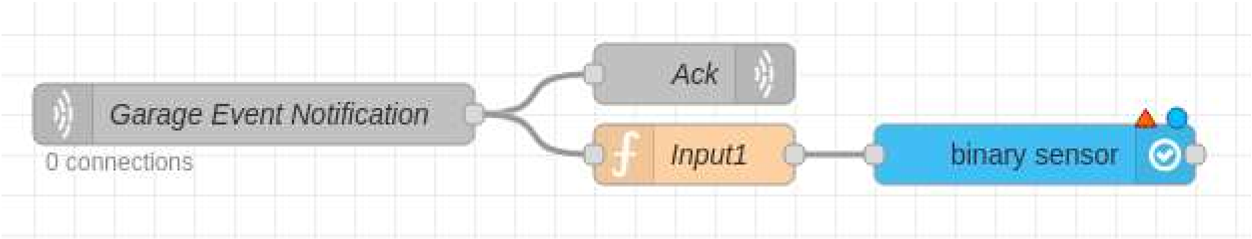
This splits the message into lines and extracts the first digit which is input1 from line 3. Note the array indexes start at zero, so line 3 is index 2 and the first digit is index 0.
The output will be 0 or 1.
Click Done and Deploy.

You will see the following when you apply and remove the input on the dScript module.

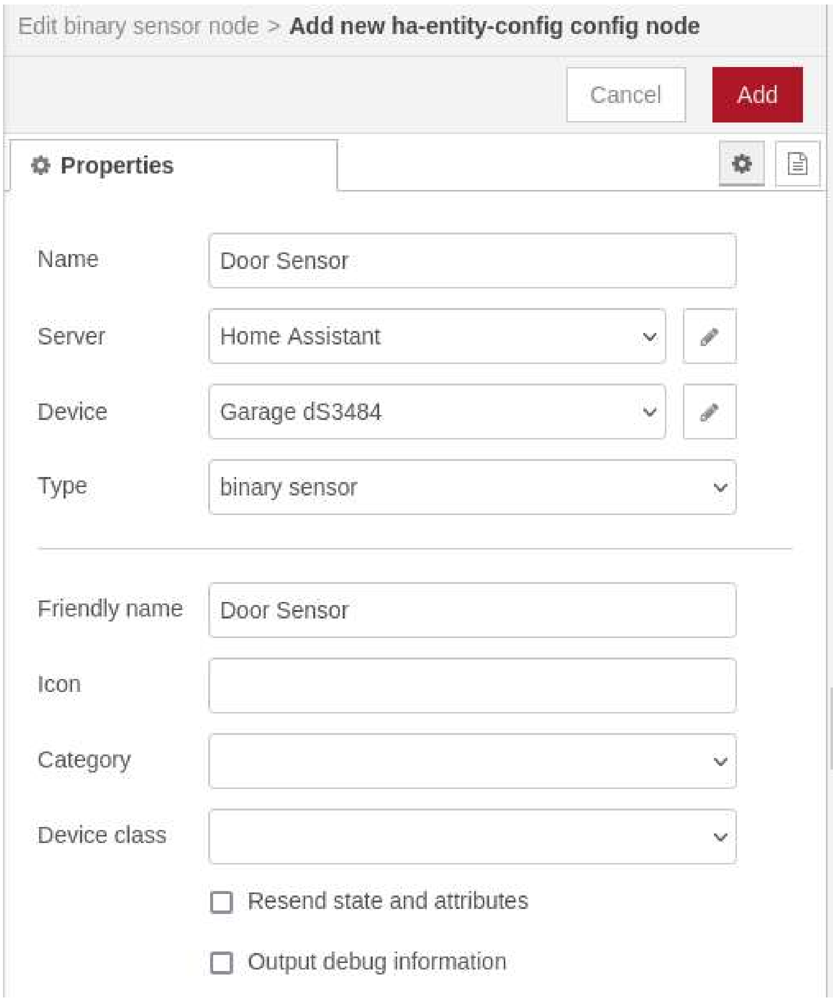


```
03/02/2023, 16:00:38   node: debug 1
msg.payload : string[1]
"1"
03/02/2023, 16:00:39   node: debug 1
msg.payload : string[1]
"0"
```

So far only NodeRED is aware of the input, now we need to get it into Home Assistant.
Delete the debug node and replace it with a binary sensor node.



Double click the binary sensor node and enter a name.
Click the edit symbol to Add new ha-entity-config. Give it a name and friendly name.
In the Add new ha-device-config box select our Garage dS3484 device.


Edit binary sensor node > **Add new ha-entity-config config node**

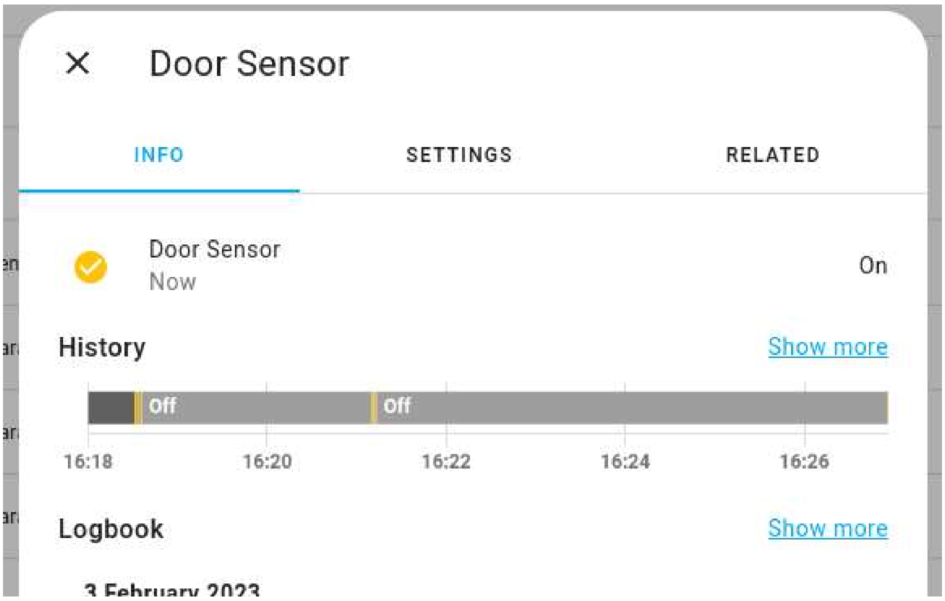
Name	Door Sensor
Server	Home Assistant
Device	Garage dS3484
Type	binary sensor
Friendly name	Door Sensor
Icon	
Category	
Device class	

□ Resend state and attributes

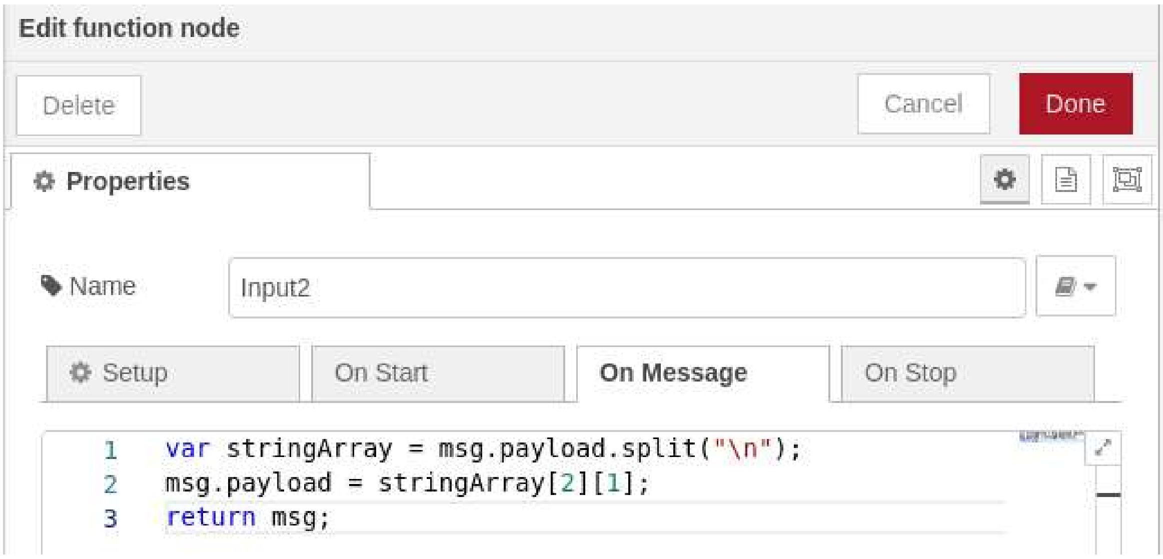
□ Output debug information

Click Add to get back to the node page, then Done to finish. Deploy.
You can now test your input.

Go to home Assistant→Settings→Devices & Services→Integrations and select entities on the Node-RED Companion window. There you will see the Door Sensor listed. Click on it and select the INFO tab. There you will see the state of Input1 on the dScript module. Apply and remove the input to see door sensor change state.



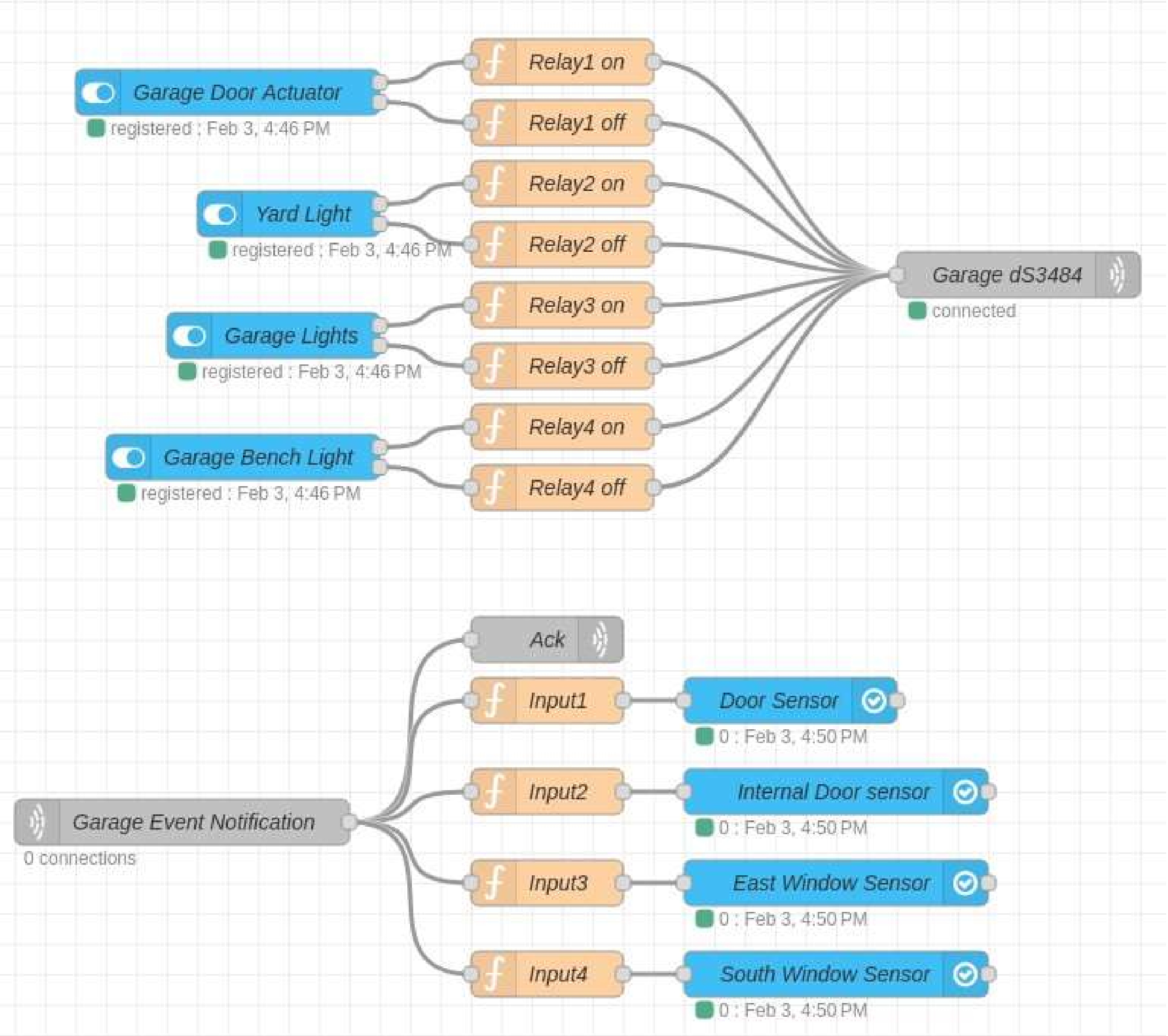
Go back to NodeRED and add inputs 2,3 and 4. Each time change the functions name and the second array index in the code. Here is Input2 function.



Name: Input2

```
var stringArray = msg.payload.split("\n");
msg.payload = stringArray[2][1];
return msg;
```

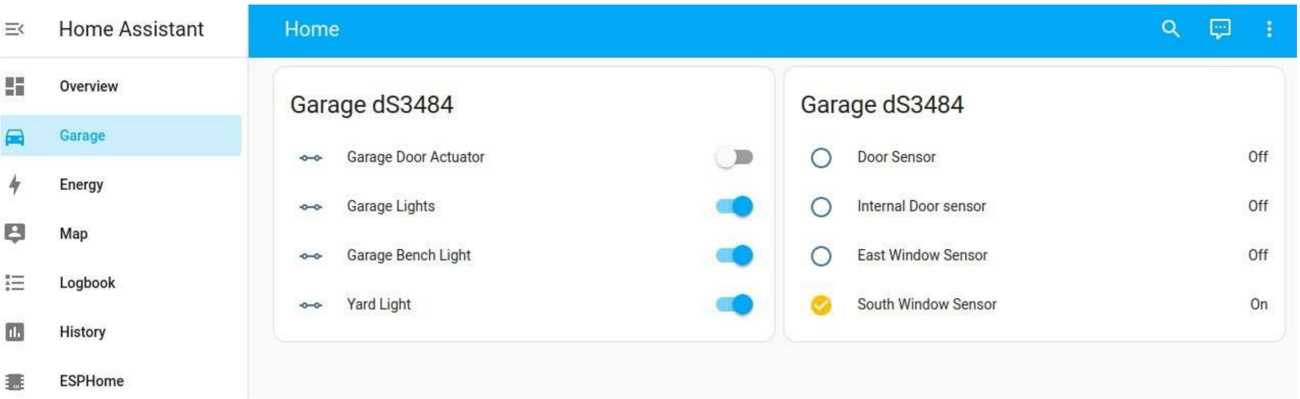
Your final flow should look like this:



Dashboard

If you want these on a Home Assistant dashboard, go to Settings→Dashboard and click ADD DASHBOARD. Set Title to Garage and select a Car as the Icon.

Go to Settings→Devices & Services→Integrations and select devices on the Node-RED Companion window. Select Garage dS3484. There you will see all your Controls and Sensors listed for the dS3484, with an ADD TO DASHBOARD link. Add them and go to your Garage Dashboard.



Garage dS3484

Garage Door Actuator	○
Garage Lights	●
Garage Bench Light	●
Yard Light	●

Garage dS3484

Door Sensor	Off
Internal Door sensor	Off
East Window Sensor	Off
South Window Sensor	On

That’s it, have fun.

Notes