Using the Module System in Node-Red

Getting Started

Node-Red consists of a server and client components. The server can be downloaded from https://nodered.org on any platform. This example uses Windows. Once the server is running then the client code runs in any browser, as long as you can connect to the machine running the Node-Red server. If you already have the Node Red server installed, make sure it's running and then go to the client section.

Server Installation & Launch

Node.js should first be installed, which can be downloaded from https://nodejs.org - once installed you can run the following commands in either Powershell or Command Prompt (this is a Windows example, use terminal on Linux or Mac) to check the applications can be found…

```
C:>node --version
```

```
C:>npm --version
```

Node Red can now be install and it adds node-red on the system path…

```
C:>npm install -g --unsafe-perm node-red
```

Once the server is installed, you can start the Node-Red server…

```
C:>node-red
```

This will start Node Red in the command prompt so you will need to keep this open to keep the server running.

Client Launch

Open a browser and navigate to the Node-Red server - if you are running the browser on the same machine that's running the server you can use…

http://localhost:1880

Otherwise use the IP address of the Node-Red server.

Client Setup

Modules can be loaded into Node-Red which can extend the Node-Red functionality. There is currently no SDRplay specific modules, but these examples show how the web-socket interface to SDRconnect's module system can be used.

Node Modules

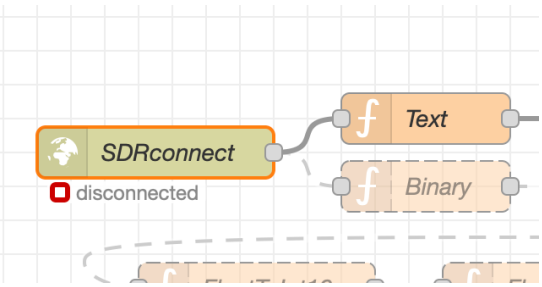
This example uses dashboard elements that can be installed using…

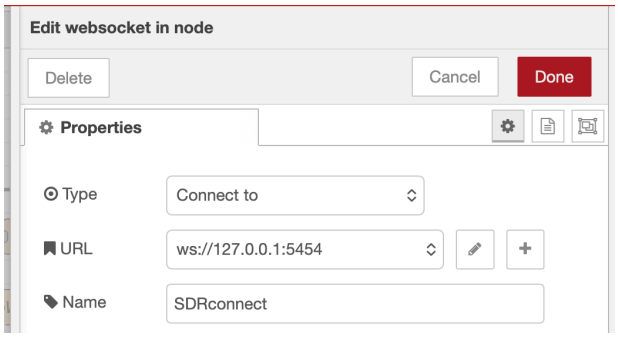
```
npm install @flowfuse/node-red-dashboard
```

You can install other components like this, just remember to restart the Node-Red server after installing the components so they appear in the client browser window.

SDRconnect Module Interface

As previously mentioned, the SDRconnect module system is communicated to via web-sockets. There are web-socket modules in Node-Red that we can use for this. Use a "websocket in" module to receive information from SDRconnect and a "websocket out" module to send data to SDRconnect…

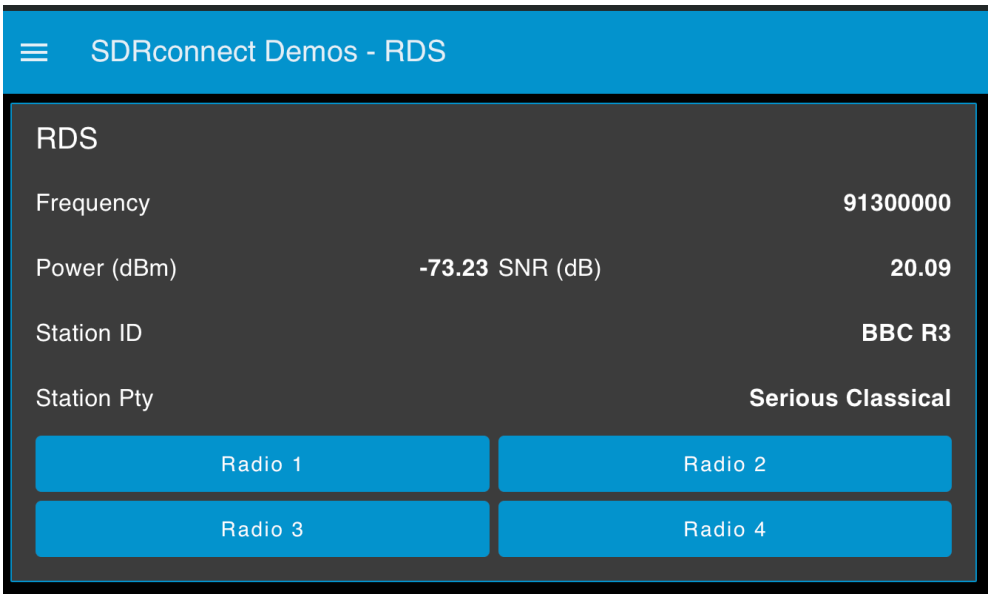


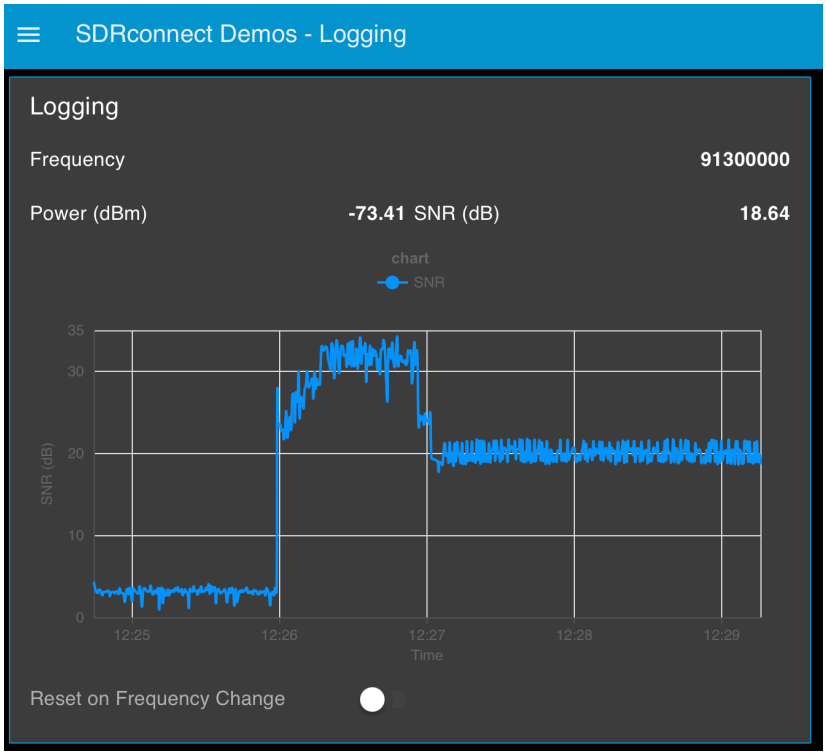


Note: the SDRconnect websocket server is on port 5454. When SDRconnect is started (and the Websocket Server is enabled in Preferences) then the disconnected message below the web-socket instance in Node-Red changes to connected.

Examples

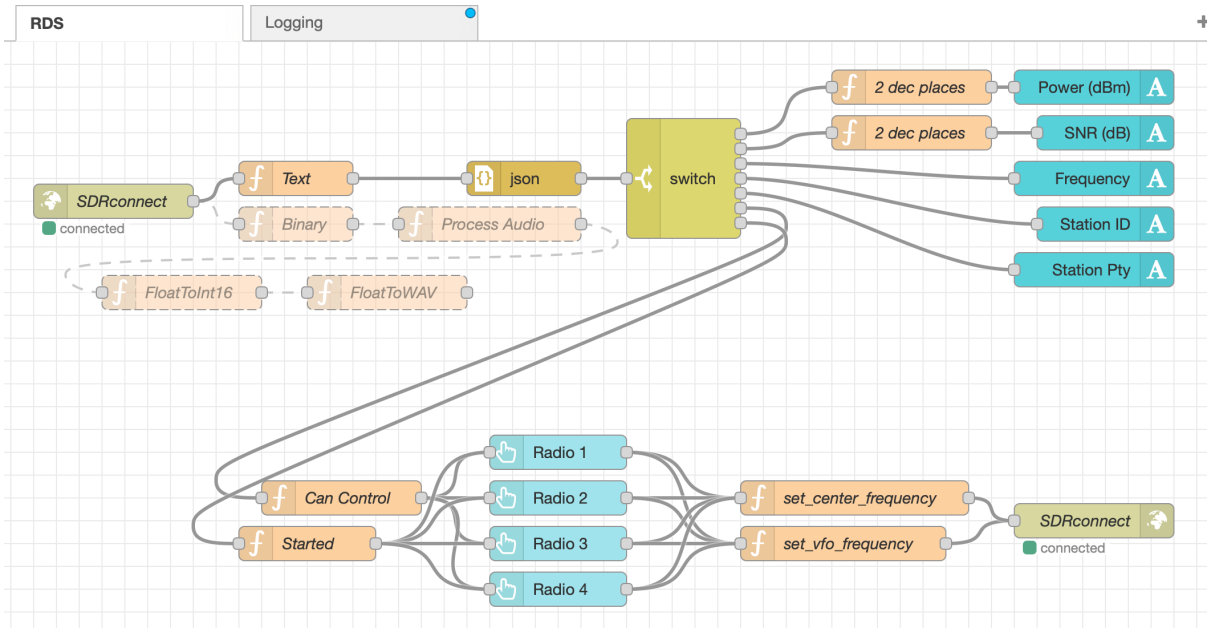
There are two examples of the sorts of things you can do with the module system in this document. One is an RDS display tool and the other is a SNR logging tool. In Node-Red they are both in the same project and you can select between them via the menu that appears in the dashboard. To open the dashboard, click on the drop down arrow in the top right corner and select Dashboard 2.0 and then click on the Open Dashboard button. A new browser tab with the dashboard will open...

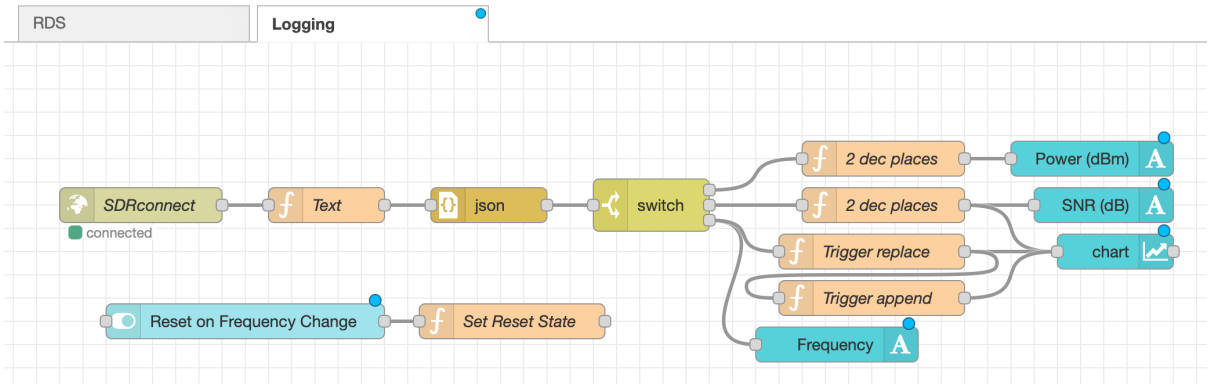




Code

Node-Red is a graphical tool and below are the two projects that are mentioned above…





The code is stored in a json container and is therefore very portable. The json for the two example projects above can be downloaded from: https://www.sdrplay.com/resources/node_red_example.zip

Extract the json file from the .zip file and then import it by clicking on the menu (top right) and selecting Import. To save your project click on the same menu and select Export.

Note: when specifying the IP address for the SDRconnect instance in the Node-Red websocket component, remember that it's the Node-Red server that's connecting to SDRconnect and not the browser client. So use 127.0.0.1 if Node-Red server and SDRconnect are on the same machine.