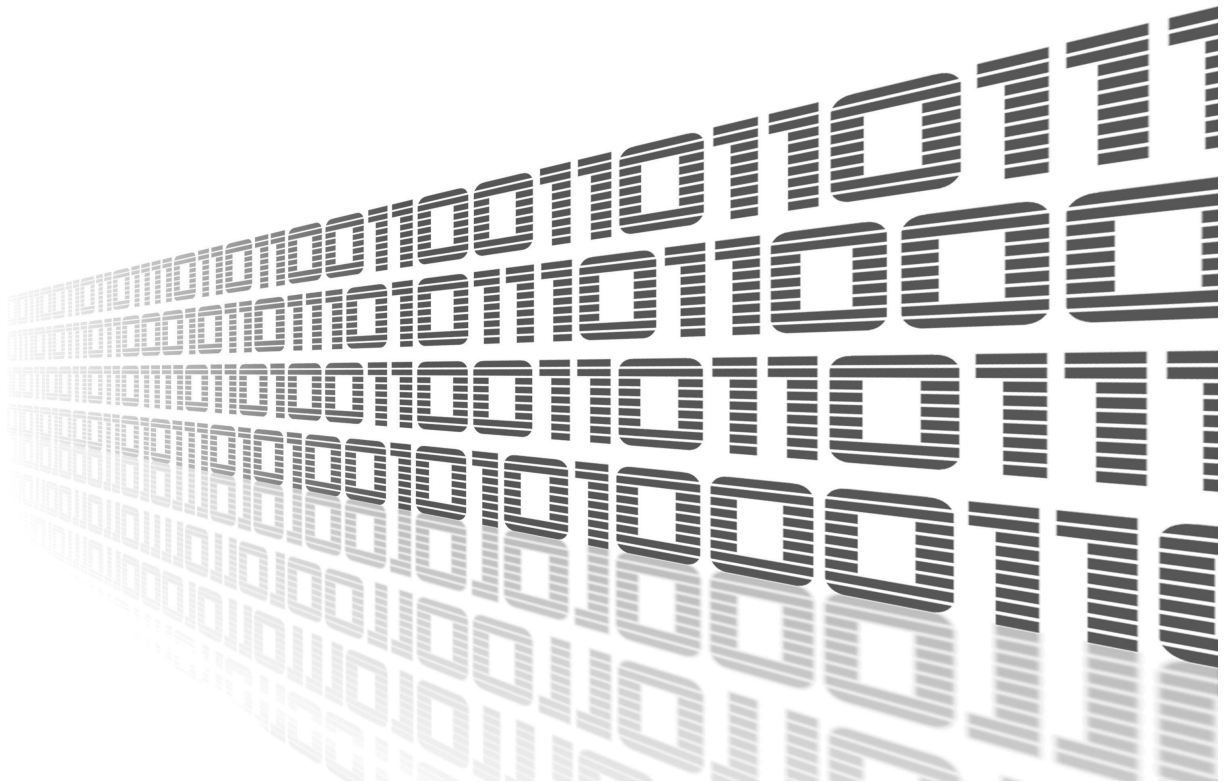




User Module

Node-RED

APPLICATION NOTE



ADVANTECH

Used symbols



Danger – Information regarding user safety or potential damage to the router.



Attention – Problems that may arise in specific situations.



Information or notice – Useful tips or information of special interest.



Example – Example of function, command or script.



Contents

1	Node-RED User Module Description	1
1.1	Node-RED Dependency and Extensions	2
1.2	Node-RED User Module	4
2	Node-RED Configuration	5
2.1	Run and Access the Node-RED engine	5
2.2	Flow Files Path	6
2.3	Web Static Files Path	7
2.4	Node-RED Status (Log)	7
3	Available Nodes and Version	8
3.1	Node-RED Version	8
3.2	NPM and Available Nodes	8
3.3	Router Nodes	10
3.4	User Modules for Additional Nodes	11
3.4.1	User Module Node-RED / Bluetooth	11
3.4.2	User Module Node-RED / Dashboard	12
3.4.3	User Module Node-RED / FTP	13
3.4.4	User Module Node-RED / GPSd	13
3.4.5	User Module Node-RED / gzip	14
3.4.6	User Module Node-RED / KNX	14
3.4.7	User Module Node-RED / Modbus	15
3.4.8	User Module Node-RED / OPC UA	16
3.4.9	User Module Node-RED / PLC - EtherNet/IP	16
3.4.10	User Module Node-RED / PLC - Melsec	17
3.4.11	User Module Node-RED / Splunk HEC	17
4	Advanced topics	18
4.1	File locations	18
4.2	How to distribute your own flows to the routers	18
4.3	Persistent variable storage	19
4.4	Node "router"	20
4.5	Building the Custom Nodes for Node.js/Node-RED	20
4.5.1	JavaScript Only Nodes	21
4.5.2	Nodes Using Other Languages	21
5	Flow Examples	23
5.1	Example 1: User LED Turn On and Off	24
5.2	Example 2: HTTP Endpoint	25

5.3 Example 3: Ping Statistics 26

5.4 Example 4: Reading from supported BLE sensor in Node-RED 28

5.5 Example 5: Reading from and writing to unsupported BLE sensor in Node-RED 31

5.6 Resources 34

6 Related Documents 35

List of Figures

1	An example of the Node-RED editor running in a v3 router	1
2	Node.js to make Node-RED work and additional nodes user modules	3
3	Node-RED user module menu	4
4	Node-RED user module configuration – run Node-RED	5
5	Node-RED login in the router	6
6	Node-RED Status: Version information and Log	7
7	List of available nodes	9
8	Additional Bluetooth nodes in Bluetooth category	11
9	Dashboard nodes	12
10	Additional FTP and SFTP nodes in storage category	13
11	Additional GPSd node in input category	13
12	Additional gzip node in functions category	14
13	Additional knx nodes in iot category	14
14	Modbus nodes	15
15	Additional OPC UA node in input category	16
16	PLC EtherNet/IP nodes in automation category	16
17	PLC Melsec nodes in automation category	17
18	Additional Splunk event collector nodes in cloud category	17
19	Persistent storage	19
20	Router node	20
21	Top right menu – import from clipboard	23
22	Import nodes via JSON	23
23	Example 1 – user LED flow	24
24	Example 1 – user LED turned on	24
25	Example 2 – imported nodes to deploy	25
26	Example 2 – HTTP response	25
27	Example 3 – Web static file path configuration	26
28	Example 3 – ping statistics flow	27
29	Example 3 – ping statistics chart and download	27
30	Example 4 – Ruuvi Tag	28
31	Example 4 – Status	28
32	Example 4 – Bluetooth router app	29
33	Example 4 – Containers	30
34	Example 5 – Xiaomi Flower Care	31
35	Example 5 – Status	31
36	Example 5 – Nodes	33

1. Node-RED User Module Description

Node-RED is a flow-based programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

The *Node-RED User Module* for Advantech routers allows Node-RED to run in v3 routers as a software user module. This enables the use of the router as a Node-RED device.

Node-RED provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click. The lightweight runtime is built on [Node.js](https://nodejs.org/), taking full advantage of its event-driven, non-blocking model. This makes it ideal for running at the edge of the network (in a router) as well as in a cloud. JavaScript functions can be created within the editor using a rich text editor. A built-in library allows you to save useful functions, templates or flows for re-use. The flows created in Node-RED are stored using JSON which can be easily imported and exported for sharing with others or other devices.



For more information and full documentation see: <http://nodered.org/>

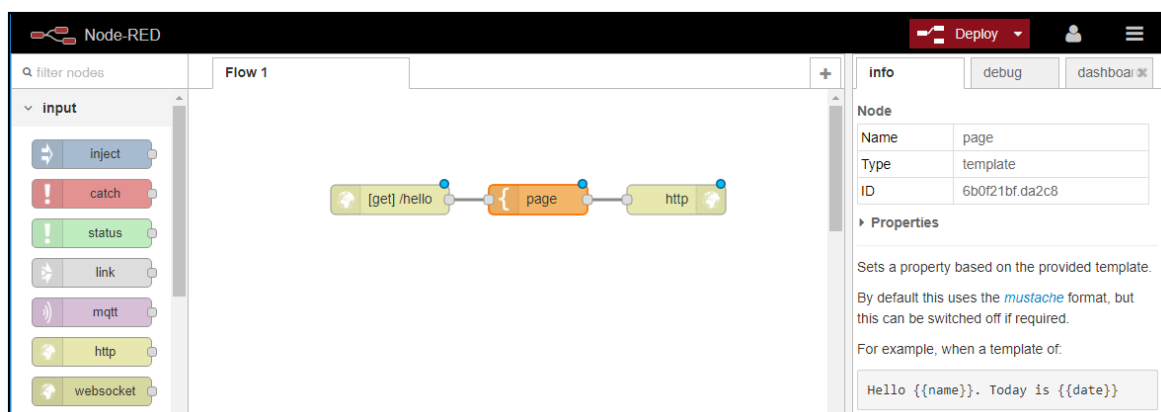


Figure 1: An example of the Node-RED editor running in a v3 router



The *Node-RED* user module is not part of the router's firmware. It can be downloaded from icr.advantech.cz/user-modules. There is dependency for *Node-RED* user module to be installed in the router – follow the instructions in Chapter 1.1. The installation process for the user modules is described in the Configuration Manual (see [1]). **This user module is only compatible with v3 and v4 platform routers!**



Flows definitions are part of the router report automatically, so if want to send report to Support, its unnecessary to attach flows definitions separately. Report does not contain data, which are declared by node as confidential (credentials).

1.1 Node-RED Dependency and Extensions



It is necessary to install the **Node.js** user module prior to *Node-RED*. *Node.js* is required for *Node-RED* to work – it is the separated module and it can be used as a standalone JavaScript server for other purposes.

There is also possibility to install other Node-RED modules. These are just adding more nodes to *Node-RED*. List of available extension modules:

- AWS
- Azure
- Bluetooth
- Dashboard
- FTP
- GPSd
- gzip
- KNX
- Modbus
- OPC UA
- PLC - EtherNet/IP
- PLC - Melsec
- Splunk event collector.

See the figure below. These additional nodes are described in Chapter 3.4.



Node-RED module *MQTT Broker* was removed. If you want MQTT Broker functionality in the Advantech routers, you can now use the separate user module, which is more powerful and has more functions.



Please use always a combination of the latest versions of published Node-RED modules. We can not guarantee the correct functionality for a mix of versions.

User Modules			
GPS	x.x.x	(yyyy-mm-dd)	Delete
Node.js	x.x.x	(yyyy-mm-dd)	Delete
Node-RED	x.x.x	(yyyy-mm-dd)	Delete
Node-RED / AWS	x.x.x	(yyyy-mm-dd)	Delete
Node-RED / Azure	x.x.x	(yyyy-mm-dd)	Delete
Node-RED / Bluetooth	x.x.x	(yyyy-mm-dd)	Delete
Node-RED / Dashboard	x.x.x	(yyyy-mm-dd)	Delete
Node-RED / FTP	x.x.x	(yyyy-mm-dd)	Delete
Node-RED / GPSd	x.x.x	(yyyy-mm-dd)	Delete
Node-RED / gzip	x.x.x	(yyyy-mm-dd)	Delete
Node-RED / KNX	x.x.x	(yyyy-mm-dd)	Delete
Node-RED / Modbus	x.x.x	(yyyy-mm-dd)	Delete
Node-RED / PLC - EtherNet/IP	x.x.x	(yyyy-mm-dd)	Delete
Node-RED / PLC - Melsec	x.x.x	(yyyy-mm-dd)	Delete
Node-RED / OPC UA	x.x.x	(yyyy-mm-dd)	Delete
Node-RED / Splunk HEC	x.x.x	(yyyy-mm-dd)	Delete
<div> New Module <input type="button" value="Choose File"/> <input type="text" value="No file chosen"/> <input type="button" value="Add or Update"/> </div>			

Figure 2: Node.js to make Node-RED work and additional nodes user modules

1.2 Node-RED User Module

When uploaded to the router, the user module is accessible in the *Customization* section in the *User Modules* item of the router's web interface. Click on the title of the user module to see the user module menu as shown below:

Information
Log
Configuration
Service Flows
General
Return

Figure 3: Node-RED user module menu

The *Information* section provides the *Log* page with the Node-RED logging messages. You can auto start/stop Node-RED on any port in the *Configuration* section or access flow editor. The *Return* item in the *General* section is to return to the higher menu of the router.

2. Node-RED Configuration

Figure 4: Node-RED user module configuration – run Node-RED

Item	Description
Enable Automatic Start	Enables Node-RED functionality.
Web access port	Enter port where you want to run Node-RED http server. Flow editor and all other web accessible parts (e.g. HTTP in node or Dashboard nodes) communicate on this port. Default port is 1880.
Log level	Choose severity of the information you want to see in log. See the chapter 2.4 for details.
Flow files path	Specify the path, where the flow definitions will be stored. See the chapter 2.2 for details.
Web static files path	Specify the path, where Node-RED web server serves files from. See the chapter 2.3 for details.
Enables web editor	You can uncheck this option on the production routers so the end users can not see/access Flows editor.

Table 1: Configuration items description

2.1 Run and Access the Node-RED engine

When the user module is uploaded to the router, the following steps allow Node-RED to run in the router:

1. In user module's menu, go to *Configuration, Node-RED*. Check the *Enable Automatic*

Start checkbox and enter port number 1880 (or another desired port) into *Port* input field. Press the *Apply* button. Node-RED will start and it will run continuously whenever the router is running (including after reboots of the router, until deactivated again by unchecking the checkbox and pressing *Apply*). You can see it's status information on the *Log* page – "Started flows" row indicates fully started Node-RED engine. You can also enable or disable flow editor - useful way to disable access to flow editor for end user.

2. Click on menu item *Flows* to access the flow editor or you can manually enter the address https://<router_ip>:<port_number> (Make user to use https as the communication is mandatorily encrypted.)



Warning: It may take 30 seconds or longer to boot the Node-RED UI (depends on the router model, the number of installed nodes and the flow size), so please wait if the login page does not load immediately.

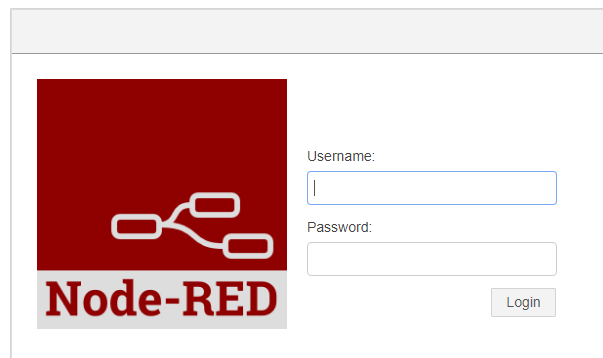


Figure 5: Node-RED login in the router

3. Login using the same credentials as you use for logging into the router. (E.g. user „root“, as used by the router administration or with some other another user's credentials). PAM authentication is used, so it is possible to login with any defined user's details in the given system. You can manage users in router *Administration, Users* menu.

Admin and User profiles have same settings when authenticating into Node-RED, so they have the same rights. If you are interested in using a different authentication process, please refer to the documentation: <https://nodered.org/docs/security>

2.2 Flow Files Path

You can specify the path, where the flow definitions will be stored. This is the GUI equivalent to *flowFile* option in *setting.js* file without file name. Default path was changed and when the user doesn't specify the path, the */var/data/node-red* will be used. Setting this path to */opt/myapp/flows* offers huge benefit of being able to archive flow as classic Router App and install it as usual afterwards. See chapter 4.2 for more details.

2.3 Web Static Files Path

There is a *Web static files path* field on the Node-RED configuration page, see Figure 4. This is the GUI equivalent to *httpStatic* option in *setting.js* file (see Chapter 4.1). It's a path, where Node-RED web server serves files from. It's useful for static web files (e.g. images) or to download collected data as a file. See example of both in Chapter 5.3.



It is necessary to distinguish between URI to local file (writing, storing data) and URI for web access (reading for user access, downloading data).

2.4 Node-RED Status (Log)

You can choose various Log level on the Node-RED configuration page, see Figure 4. Following options are available: *None*, *Fatal*, *Error*, *Warning*, *Info* (the default one), *Debug*, *Trace*.

The Log messages are visible on the *Log* page, in the *Status* section of the user module interface. Here you will find various information, such as: software parts and versions used (which version of Node-RED, Node.js etc.), the settings file, the user directory, the flow JSON file etc. (It is possible to access or edit these, for example via SSH when logged into the router). Information regarding the start/stop flows and error information is also shown in this section. Please see the example below. You can also download these messages and save them to your computer as a text file, simply by clicking the *Save Log* button. User can find version of all other individual nodes in the Node.js User Module details on the Licenses page.

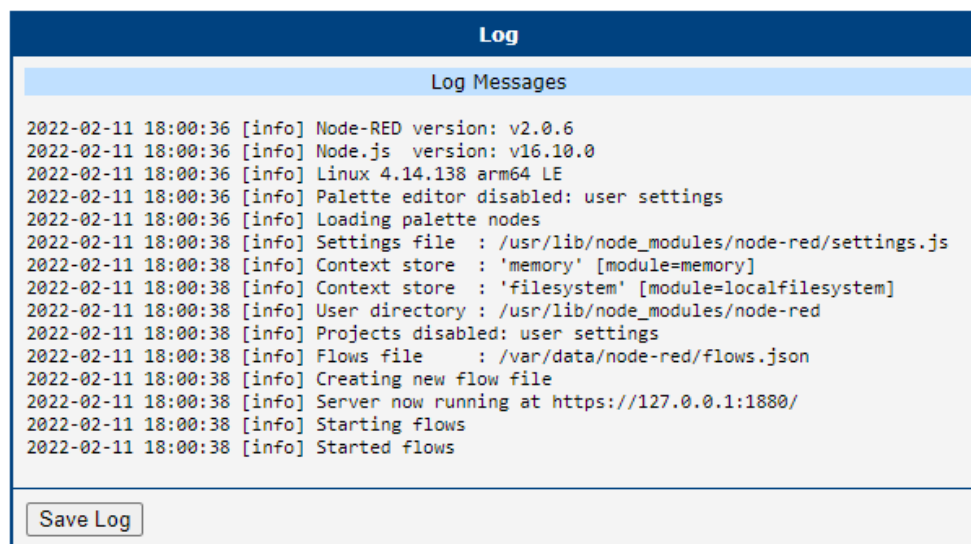


Figure 6: Node-RED Status: Version information and Log

3. Available Nodes and Version

3.1 Node-RED Version

For more information about the software versions, see the *Log* page in the user module. As shown on Figure 6, the *Node-RED* user module has version 2.0.6 and *Node.js* version 16.10.0.

3.2 NPM and Available Nodes



The NPM package system is not installed in the user module because of the router's platform limitations.



Should you need any node not available as part of our user modules, please contact us at: cellular.info@advantech.cz

The Node-RED user module itself has the following node packages installed:

- Node packages required by Nore-RED
- Router nodes – special set designed for Advantech routers – see Chapter 3.3
- Big Time – <https://flows.nodered.org/node/node-red-contrib-bigtimer>
- Buffer Parser – <https://flows.nodered.org/node/node-red-contrib-buffer-parser>
- Credentials – <https://flows.nodered.org/node/node-red-contrib-credentials>
- Float – <https://flows.nodered.org/node/node-red-contrib-tofloat>
- Input Split – <https://flows.nodered.org/node/node-red-contrib-input-split>
- Loop – <https://flows.nodered.org/node/node-red-contrib-loop>
- Ping – <https://flows.nodered.org/node/node-red-node-ping>
- Snmp – <https://flows.nodered.org/node/node-red-node-snmp>
- String – <https://flows.nodered.org/node/node-red-contrib-string>
- Timed Counter – <https://flows.nodered.org/node/node-red-contrib-timed-counter>

Figure 7 lists full graphical overview of the nodes available to drag and drop in the Node-RED UI.

Note: It is possible to install node packages that do not require the compilation.

Attention: It is on your own risk – consider that Advantech Czech does not provide support for any nodes installed by user on his own (not retrieved from us).



Copy the files of the package into the `/usr/lib/node_modules/node-red/nodes/` path in your router and restart the running Node-RED to load the new node. In the case that you see an error message such as: „Tell the author that this fails on your system: <name-of-package>“ this means that the node must be compiled for the given platform and you should contact us on the e-mail given above.

You can also uninstall any unused node package: If you do not use any of the installed nodes (for example you are in need of more disk space), delete the installation folder of this package in the `/usr/lib/node_modules/node-red/nodes/` path in your router. Then restart the user module again.

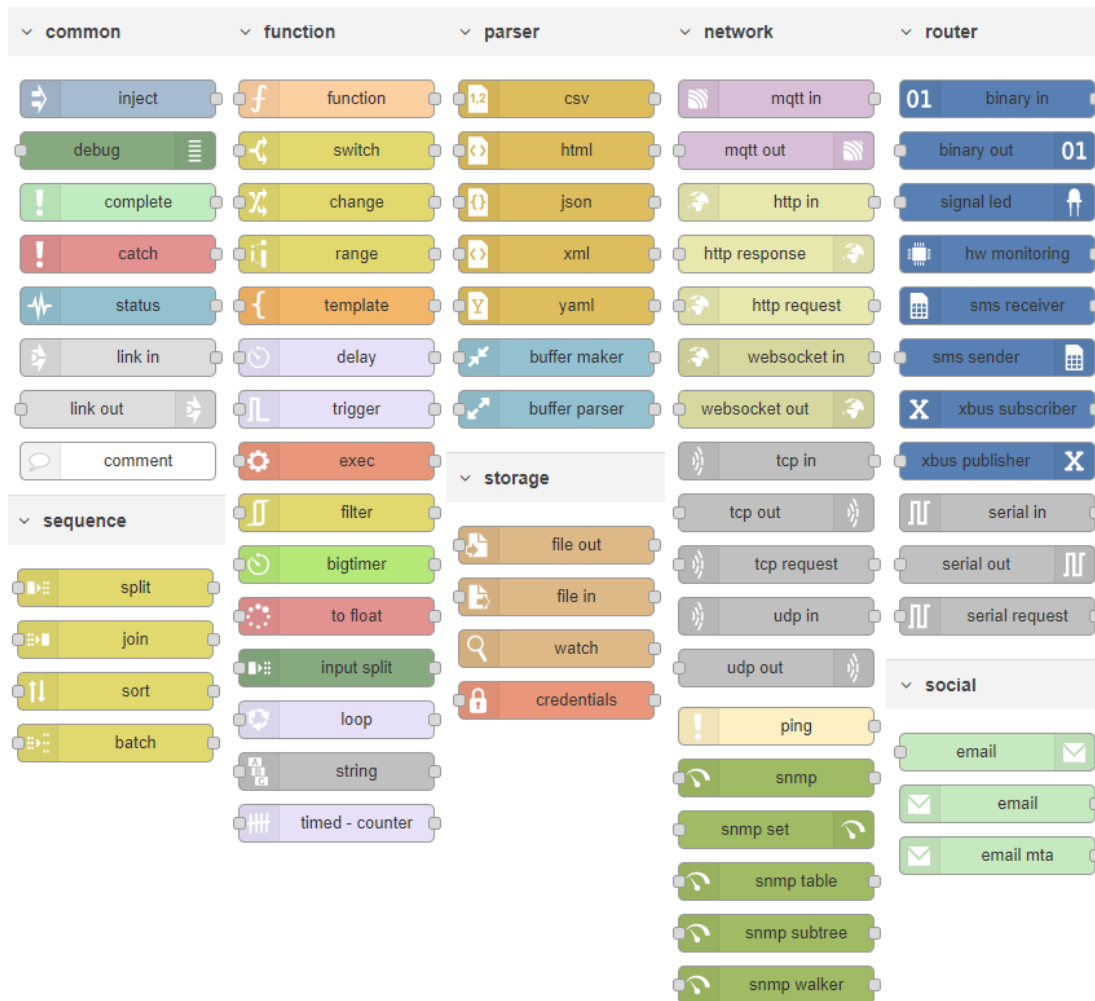


Figure 7: List of available nodes



Please note that router's firmware of version 6.2.1 and above is required for the *watch* node to work properly.

3.3 Router Nodes

The list of available router related nodes is in the table below. More detailed documentation is contained in the Node-RED UI itself when clicking on the nodes.

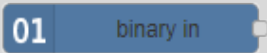
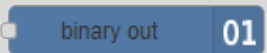
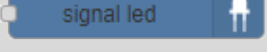
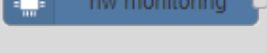
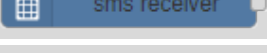

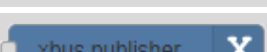

Node	Description
 01 binary in	Binary input node for the router. Sends a message with payload 0 or 1 to the node output when the pin changes state.
 binary out 01	Directly controls a binary output pins on the router.
 signal led	Directly controls the signal LED on the router.
 hw monitoring	Monitores the hardware parameters. You can select between the temeperature in Celsius degrees and the power supply in Volts
 sms receiver	Receives a SMS to the router.
 sms sender	Sends a SMS from the router.
 X xbus subscriber	Subscribes messages from XBus.
 xbus publisher X	Publishes messages to XBus.

Table 2: Router nodes

3.4 User Modules for Additional Nodes

Additional user modules can be uploaded to the router to add more Node-RED nodes. All these work only after Node-RED user module installation.

3.4.1 User Module Node-RED / Bluetooth

Adding this user module to the router will add the BLE Scanner, BLE device, BLE in, BLE out and ruuvitag node to the bluetooth node palette as seen in the figure below. See the examples 5.4 and 5.5 on how to use this nodes. More about Bluetooth on Advantech routers you can find in Bluetooth Router App manual. See more information in the official documentation: <https://flows.nodered.org/node/node-red-contrib-noble-bluetooth>



This user module requires the Bluetooth user module to be installed on the router.

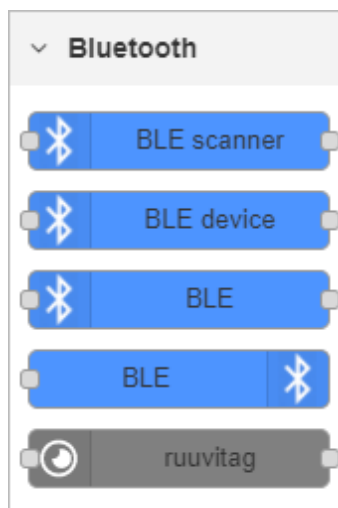


Figure 8: Additional Bluetooth nodes in Bluetooth category

3.4.2 User Module Node-RED / Dashboard

Dashboard allows to create a web UI to interact with your Node-RED flows/application. For example it can show a live data from the input nodes or user can control behaviour of the flow. Adding this user module to the router will add the Dashboard nodes as seen in the figure below. Compared to the basic Dashboard package, it contains several additional nodes. User can find running Dashboard web page on <https://<ip-address-of-the-router>:<configured-port>/ui>. See more information in the official documentation: <https://flows.nodered.org/node/node-red-dashboard>

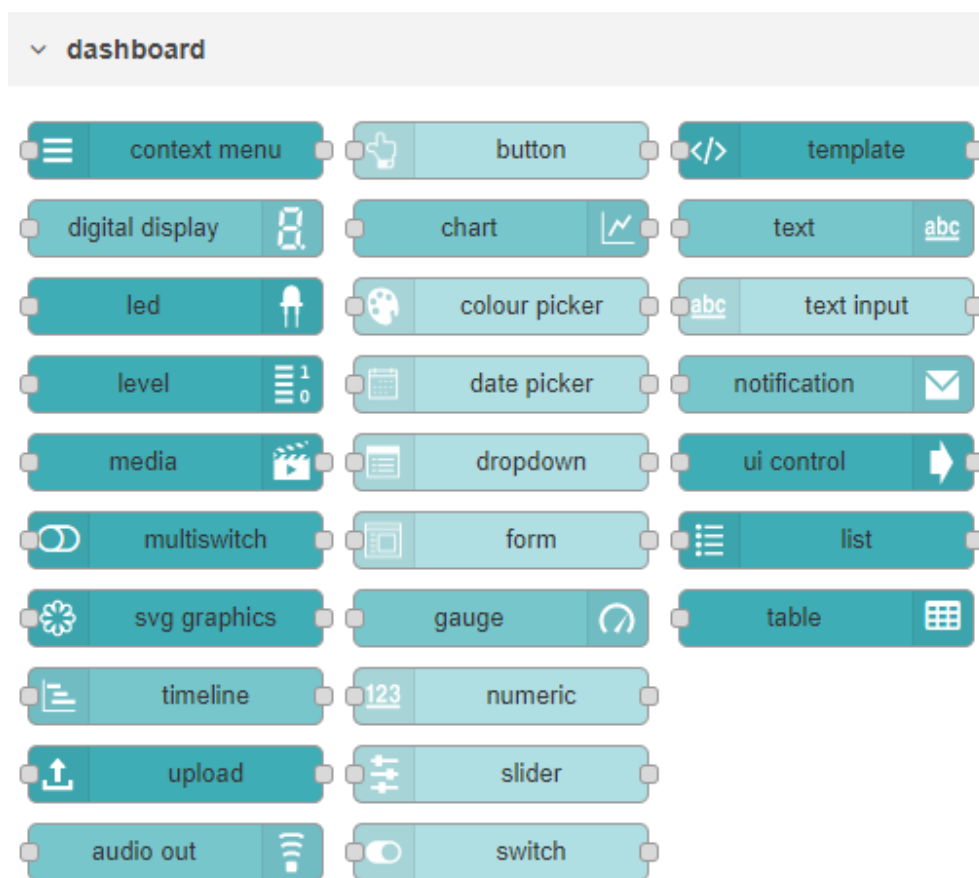


Figure 9: Dashboard nodes

3.4.3 User Module Node-RED / FTP

Adding this user module to the router will add the FTP and SFTP node to the storage node palette as seen in the figure below. You can add an FTP or SFTP connection in the node and operations: list, get, put, delete are available. See more information in the official documentation: <https://flows.nodered.org/node/node-red-contrib-ftp> and <https://flows.nodered.org/node/node-red-contrib-better-sftp>

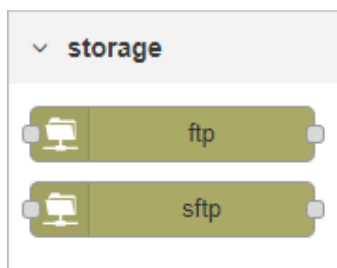


Figure 10: Additional FTP and SFTP nodes in storage category

3.4.4 User Module Node-RED / GPSd



This user module requires the GPS user module to be installed on the router.

Adding this user module to the router will add the GPSd node to the input node palette as seen in the figure below. GPSd is the input node that talks to a gpsd daemon and retrieves data from a GPS in the router. See more information in the official documentation: <https://flows.nodered.org/node/node-red-contrib-gpsd>

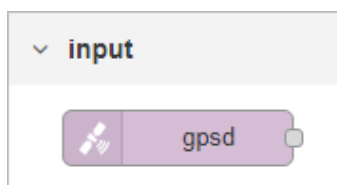


Figure 11: Additional GPSd node in input category

3.4.5 User Module Node-RED / gzip

Adding this user module to the router will add the gzip node to the function node palette as seen in the figure below. If the input is a compressed buffer it tries to decompress to a utf8 string. If the input is a normal string it creates a compressed buffer. See more information in the official documentation: <https://flows.nodered.org/node/node-red-contrib-gzip>

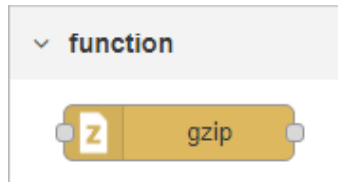


Figure 12: Additional gzip node in functions category

3.4.6 User Module Node-RED / KNX

Adding this user module to the router will add the *knx device*, *knx out* and *knx in* nodes to the iot node palette as seen in the figure below. These nodes add KNXnet/IP support for Node-RED so it is possible to talk to KNX both in raw form and as higher level devices. KNX is an open standard for commercial and domestic building automation. See more information in the official documentation: <https://flows.nodered.org/node/node-red-contrib-knx-ultimate>

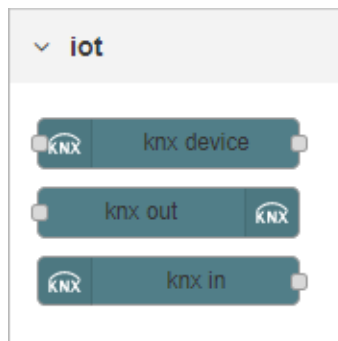


Figure 13: Additional knx nodes in iot category

3.4.7 User Module Node-RED / Modbus

Adding this user module to the router will add node group *automation* as seen in the figure below. Those nodes enable whole lot of modbus functionality. See more information in the official documentation: <https://flows.nodered.org/node/node-red-contrib-modbus>

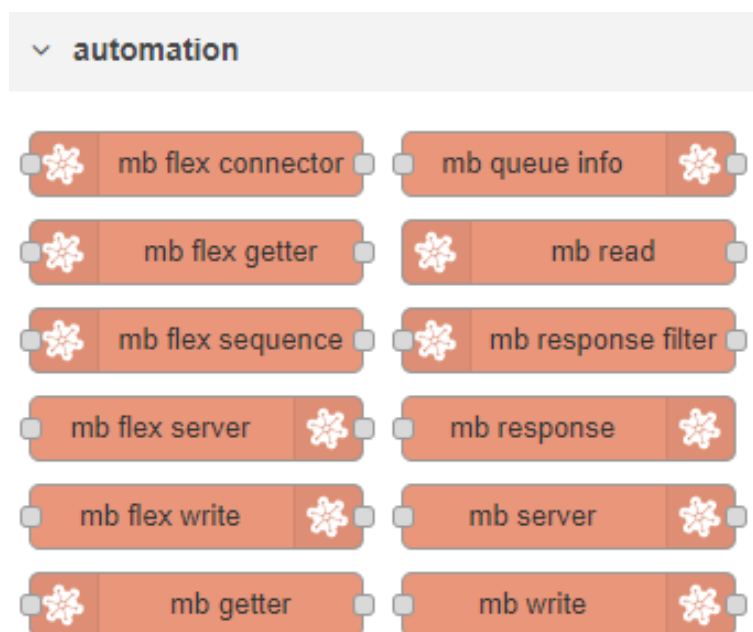


Figure 14: Modbus nodes

3.4.8 User Module Node-RED / OPC UA

OPC Unified Architecture (OPC UA) is a machine to machine communication protocol for industrial automation. Adding this user module to the router will add *opcua item*, *opcua client*, *opcua browser*, *opcua server*, *opcua event*, and *opcua method* to the node palette as seen in the figure below. See more information in the official documentation of the node: <https://flows.nodered.org/node/node-red-contrib-opcua>



Figure 15: Additional OPC UA node in input category

3.4.9 User Module Node-RED / PLC - EtherNet/IP

Node for interaction with PLC using EtherNet/IP Protocol. The protocol EtherNet/IP is widely used by Allen Bradley/Rockwell, Schneider Electric, and Omron PLCs. See more information in the official documentation of the node:

<https://flows.nodered.org/node/node-red-contrib-cip-ethernet-ip>

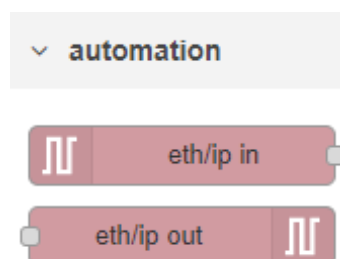


Figure 16: PLC EtherNet/IP nodes in automation category

3.4.10 User Module Node-RED / PLC - Melsec

Nodes to Read from & Write to PLC over Ethernet using MC Protocol. The protocol Melsec is widely used by Mitsubishi PLCs. See more information in the official documentation of the node:

<https://flows.nodered.org/node/node-red-contrib-mcprotocol>

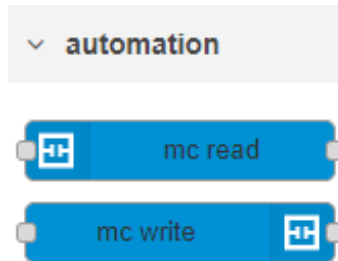


Figure 17: PLC Melsec nodes in automation category

3.4.11 User Module Node-RED / Splunk HEC

Splunk HEC is a HTTP event collector. Adding this user module to the router will add the *splunk hec metric* and *splunk hec* nodes to the cloud palette as seen in the figure below. Here *hec* stands for HTTP event collector and the nodes enable contribution with Splunk tools (www.splunk.com). These nodes allow Node-RED to publish a payload to Splunk's HTTP Event Collector. See more information in the official documentation of the nodes:

<https://flows.nodered.org/node/node-red-contrib-http-event-collector>

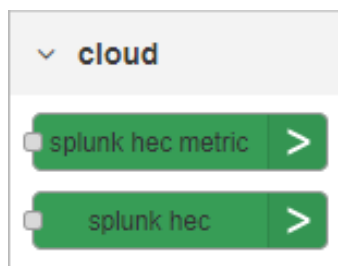


Figure 18: Additional Splunk event collector nodes in cloud category

4. Advanced topics

4.1 File locations

Files related to Node-RED are organized in the router as follows:

/usr/lib/node_modules – Node.js looks for node modules in this folder and its subfolders.

One of the subfolders is "node-red" with Node-RED installation.

/usr/lib/node_modules/node-red/settings.js – File with Node-RED configuration and authentication configuration.

/usr/lib/node_modules/node-red/nodes – Node-RED looks for nodes in this folder. You can put a 3rd party nodes directly into this folder.

/var/log/module-nodered – Logs from Node-RED are written in this file. The log is rotating (maximum size is 1 MB, the length of history depends on the Log level set).

/var/data/node-red/flows.json - File contains definition of your flows. Path part is configurable, see [2.2](#)

/var/data/node-red/flows_cred.json - File contains credentials (confidential data) of your flows. File is encrypted. Path part is configurable, see [2.2](#)

/var/data/node-red/context - Node-RED stores the persistent flow and global variables here.

4.2 How to distribute your own flows to the routers

There are several options how to distribute flows you created to the multiple other routers.

You can of course use export/import functions in the flows editor and perform deploy. This way is useful especially during the development. For distribution to the production routers is this way too tedious and can't be automated.

You could also directly copy the files with flows definitions, for example over SFTP. Path to those files can be set in the Configuration (Flow files path item), default path is `/var/data/node-red`. Main file is called `flows.json`. After the file upload, the Node-RED need to be restarted.

If your flows contains any confidential data (for example password in FTP node), you'll need to add the file `flows_cred.json`. This file is encrypted. For the router to be able to decrypt this file, the secret from the router, where the flows was created is needed. Secret is generated during the first launch of Node-RED, or you could choose your own. This option is not accessible via GUI. The secret can be physically found in the `/opt/nodered/etc/setting` file in the `MOD_NODERED_CREDSECRET` item. If you are running router installation in bulk, you are probably loading prepared configuration. If the configuration is prepared on the router, where the flows are developed, the needed secret for Node-RED will be easily distributed with it.

The best way how to distribute flows to the multiple devices is to create your own Router App (previously User module). This way you can install and updated your Node-RED application in the exactly same manner as the other Router Apps. One simple "trick" is needed - storage folder for flows (Flow files path item in Configuration) is set to `/opt` (for example `/opt/myapp/flows`).

After that you prepare the `myapp` folder with subdirectories `flows` and `etc`. Place the flows definitions to `flows` (see above). The `etc` should contain the text files `name` and `version` with name and version of your application. Compress everything using the tar command:

```
tar -czf myapp.v3.tgz myapp
```

More informations about creating Router Apps/User Modules can be found on our portal <https://icr.advantech.cz/devzone/developing-user-modules>.

Your own Router App can be used for distributing your own nodes too. How to prepare your own node can be found on our portal <https://icr.advantech.cz/support/faq/detail/building-the-custom-nodes-for-node-js-node-red>. Finished node can be then compressed with the other files to a `.tgz`. You need to create symbolic link to your node on the router, for example `/usr/lib/node_modules/node-red/nodes/mynode -> /opt/myapp/nodes/mynode`. For creating symbolic link use the script `/myapp/etc/install`, for deleting symbolic link use `/myapp/etc/uninstall`.

There are other ways how to get your flows to Node-RED, for example Node-RED REST API, but this topic exceeds the focus of this application note.

4.3 Persistent variable storage

The persistent storage for variables is defined in the Node-RED configuration. When the user is adding some item of the *flow* or *global* type, it is possible to choose whether it should be in the memory or in the filesystem. In the case of the filesystem, the variable survive even a router reboot. Items are physically stored in `/var/data/node-red/context`.

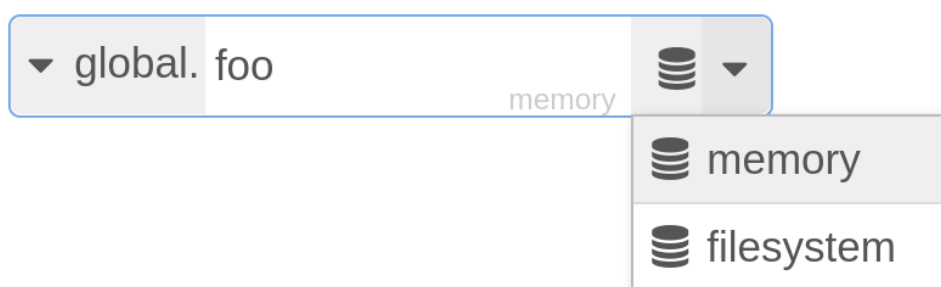


Figure 19: Persistent storage

4.4 Node "router"

Our node *router* was added to the global context. See Node.js application note for more information about this node. Users can use this node to obtain router data like serial number. How to get the serial number to the message payload is shown on the example below.

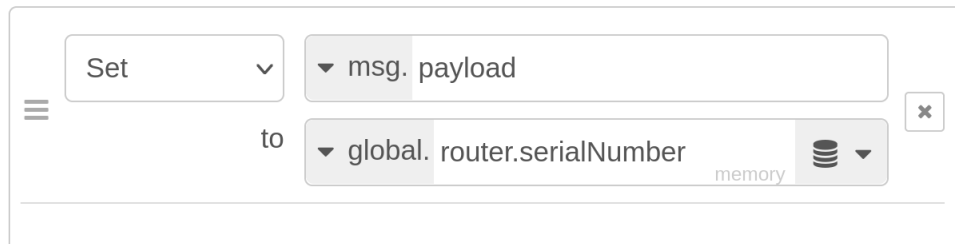


Figure 20: Router node

Alternatively the *router* node can be used in the *function* node, like this:

```
let router = global.get('router')
msg.payload = router.productName
```

4.5 Building the Custom Nodes for Node.js/Node-RED



Advantech is not responsible for the consequences caused by the customer built nodes and it provides no support for the instructions in this chapter.

An official way how to build and install a node is using npm command. However it is not possible to find it on our routers as router is embedded device with limited resources and some nodes require complex building environment and high performance because of other languages than JavaScript.

Fortunately, it is easy to prepare a node on PC with Linux and then copy it to the router. You can meet two kinds of nodes: written in JavaScript only and written in JavaScript and other language(s).



Some node authors provide prebuilt binaries so you can install this one easily as JavaScript only nodes. Look for message:

```
WARN install No prebuilt binaries found
```

in npm output. Such node is second mentioned kind.

4.5.1 JavaScript Only Nodes

This nodes need NPM package manager only. Install NPM via your distribution repository. For example:

```
apt-get install npm
```

for deb based distributions or:

```
dnf install npm (or yum install npm)
```

for rpm based distributions.

Now you can prepare desired node:

```
npm --global-style install NODE
```

where NODE is a node name as you can find it on www.npmjs.org or flows.nodered.org

Once it is finished you can see node_modules folder with desired node folder inside. Copy the node folder to the router to /usr/lib/node_modules/node-red/nodes folder, e.g. via FTP/SFTP, restart Node-RED and node(s) should appear in the palette. Or better idea is to place it to /opt folder create a link from the mentioned folders to avoid overriding while upgrading a firmware. It is also possible to pack result node to .tgz file as the common User Module.

4.5.2 Nodes Using Other Languages

Some nodes has parts written in other languages than JavaScript. In addition, these nodes require GYP meta-build system and a build environment relevant to the language.

GYP package is NPM dependent on most distributions so you don't need to install it separately.

For most common languages C/C++, you can use same toolchains we use for building the router user modules. See Environment setup chapter in Developing User Modules Guideline¹ to learn how to install it.

When you have installed all prerequisites, invoke following commands for V3 platform:

```
export CC="/opt/toolchain/gcc-conel-armv7-linux-gnueabi/  
bin/armv7-linux-gnueabi-gcc"  
export CXX="/opt/toolchain/gcc-conel-armv7-linux-gnueabi/  
bin/armv7-linux-gnueabi-g++"  
npm --arch=arm --global-style install NODE
```

or for V4 platform invoke following:

¹<https://sendfiles.advantech-bb.cz/download.php?id=42&token=W0TUyz5bd4SIg60b2frqH9dCI9pAtn1>

```
export CC="/opt/toolchain/gcc-conel-aarch64-linux-gnueabi/  
    bin/aarch64-linux-gnueabi-gcc"  
export CXX="/opt/toolchain/gcc-conel-aarch64-linux-gnueabi/  
    bin/aarch64-linux-gnueabi-g++"  
npm --arch=arm --global-style install NODE
```

where NODE is again a real node name and continue with copying as described above.



Binding to other languages can use a version dependent API so while building it is strongly recommended to have the same Node.js major version as in the router.

5. Flow Examples

Following examples are importable to Node-RED via JSON code. To import the example to the Node-RED UI, simply copy the JSON code provided at the example. Import the code via top right menu in the Node-RED UI, by choosing *Import, Clipboard*.

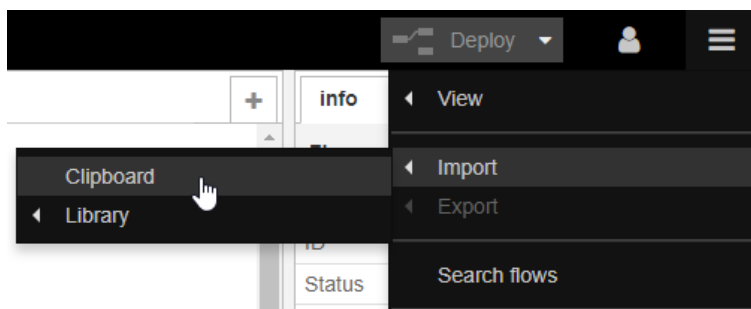


Figure 21: Top right menu – import from clipboard

Paste the JSON code to the input field, as shown in the figure below.

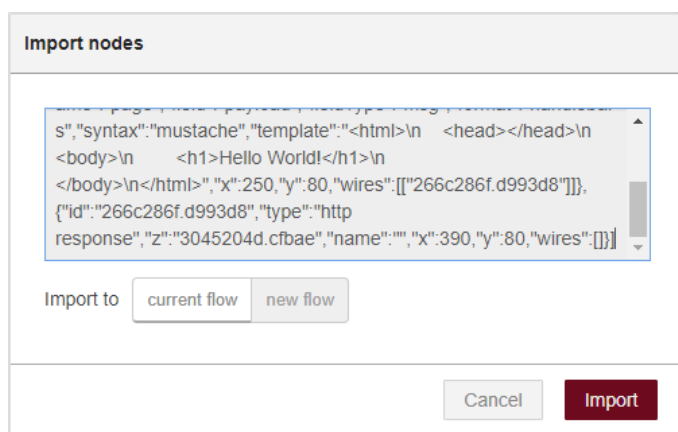


Figure 22: Import nodes via JSON

After import, the JSON code turns into the desired nodes flow.

5.1 Example 1: User LED Turn On and Off

Simple example of the first Node-RED flow to try with Router nodes. Import the following JSON code to get the flow:

```
[{"id":"332fed03.319632","type":"inject","z":"2fdae8a2.1f1e","name":"ON","topic":"","payload":"true","payloadType":"bool","repeat":"","crontab":"","once":false,"x":176.5,"y":33.01666259765625,"wires":[["6b35babe.53a5c4"]]}, {"id":"544bcbf6.9236cc","type":"inject","z":"2fdae8a2.1f1e","name":"OFF","topic":"","payload":"false","payloadType":"bool","repeat":"","crontab":"","once":false,"x":175.5,"y":118.01666259765625,"wires":[["6b35babe.53a5c4"]]}, {"id":"6b35babe.53a5c4","type":"signal_led","z":"2fdae8a2.1f1e","led":"USR","inverting":false,"defaultState":"0","name":"USRLED","x":324.5,"y":75.63333129882812,"wires":[]}]
```

Two *inject* nodes from input nodes list are used and one *signal led* node from Router nodes list. Click twice the node to see or edit its parameters.

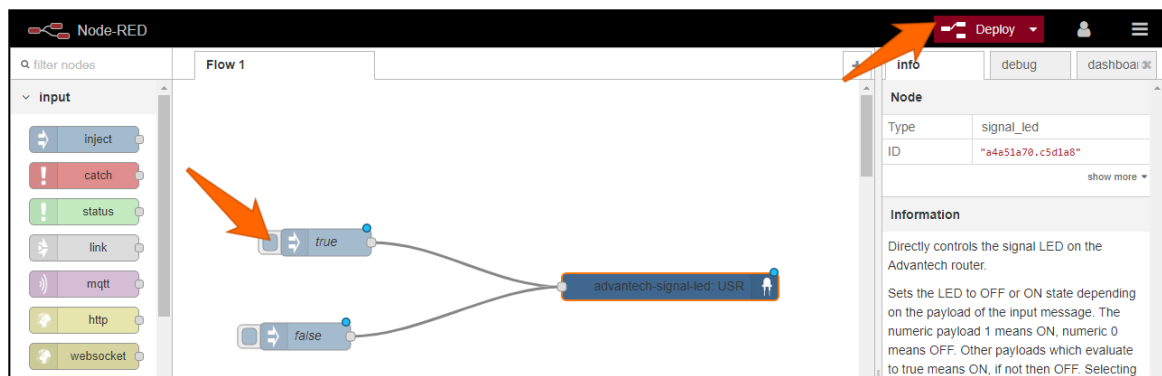


Figure 23: Example 1 – user LED flow

Click *Deploy* red button in the right upper corner. Now the flow is running. Click on the small rounded button in front of true node (orange arrow in the figure). The USR LED on the router will start to shine as in the figure below. You can turn it off by clicking on the false node (sends boolean false to the signal led node).



Figure 24: Example 1 – user LED turned on

5.2 Example 2: HTTP Endpoint

This example is taken from the Node-RED documentation examples:

<https://cookbook.nodered.org/http/create-an-http-endpoint>

Import the JSON code from the link above. It turns into the nodes as seen on the following Figure – an HTTP endpoint that responds to GET requests with some static content (such as an HTML page) is created.

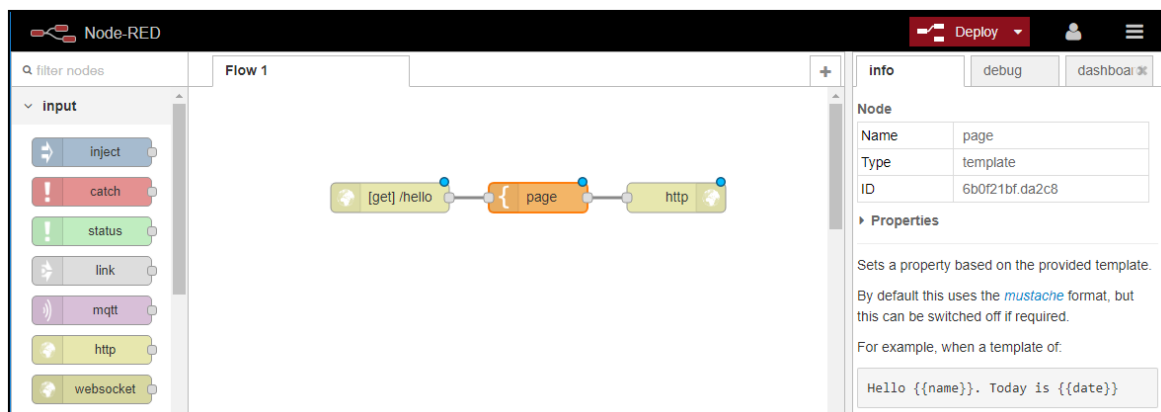


Figure 25: Example 2 – imported nodes to deploy

The nodes "http in" (where URL can be defined), "template" (where HTML page can be defined) and "http response" are used. Click the red *Deploy* button in the upper right corner. This will make the flow live and running.

When you enter a defined URL (from "http in" node) into the web browser (in this case [https://<ip-adress-of-the-router>:1880/hello](https://10.40.28.66:1880/hello)), you will see the defined HTML page as the response:

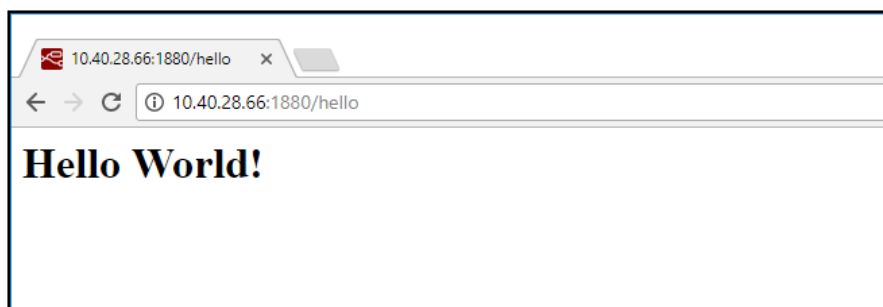


Figure 26: Example 2 – HTTP response

5.3 Example 3: Ping Statistics

This is an example of how to use the Web static file path option in Node-RED configuration (explained in Chapter 4.1). It pings on www.advantech.com domain, stores the results in a file, shows them in a chart and enables download of collected data file.

First, add the path `/var/nodered` to the Node-RED configuration as seen on the Figure:

The screenshot shows the 'Configuration module' interface. It has a checkbox for 'Enable Automatic Start' which is checked. Below it are fields for 'UI access port' (1880), 'Log level' (Info), and 'Web static files path' (/var/nodered). The 'Web static files path' field is highlighted with an orange border. There is an 'Apply' button at the bottom.

Figure 27: Example 3 – Web static file path configuration

Next, create the folder structure so the given path exists. You can login to the router (via SSH) and run these commands,

```
mkdir /var/nodered/
mkdir /var/nodered/data
```

or connect to the router via FTP and create folders `/var/nodered/` and `/var/nodered/data`. Then download the button image from https://upload.wikimedia.org/wikipedia/commons/thumb/8/8d/Download_alt_font_awesome.svg/512px-Download_alt_font_awesome.svg.png and upload it to the router as `/var/nodered/download.svg` file.

Copy this JSON code and import it as the Node-RED flow:

```
[{"id":"2c84ad78.faf92","type":"ui_chart","z":"8ee72921.8c00d8","name":"Chart","group":"ae1455d7.c2f2c8","order":1,"width":0,"height":0,"label":"Pings","chartType":"line","legend":"false","xformat":"HH:mm","interpolate":"linear","nodata":"","dot":false,"ymin":0,"ymax":100,"removeOlder":5,"removeOlderUnit":60,"cutout":0,"useOneColor":false,"colors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"useOldStyle":false,"x":441.5,"y":159.64999389648438,"wires":[[],[]]},{id:"d9e078ec.f791b","type":"file","z":"8ee72921.8c00d8","name":"","filename":"/var/nodered/data/pings.dat","appendNewline":true,"createDir":true,"overwriteFile":"false","x":422.5,"y":101.08331298828125,"wires":[[]]},{id:"69a0b6f4.2b1978","type":"tail","z":"8ee72921.8c00d8","name":"","filetype":"utf-8","split":true,"filename":"/var/nodered/data/pings.dat","x":169.5,"y":160.08331298828125,"wires":[["2c84ad78.faf92"]]}, {"id":"17ec4dc6.6c5392","type":"ui_template","z":"8ee72921.8c00d8","group":"ae1455d7.c2f2c8","name":"Download button","order":0,"width":2,"height":1,"format":"<md-button onclick='document.location='/data/pings.dat'/><img src='/download.svg' width='20' height='20'/></md-button>","storeOutMessages":true,"fwdInMessages":true,"templateScope":"local","x":472.5,"y":202.6500244140625,"wires":[[]]},{id":"21f9f5c8.52674a","type":"ping","z":"8ee72921.8c00d8","name":"","host":"www.advantech.com","timer":10,"x":149.5,"y":100.41668701171875,"wires":[["d9e078ec.f791b"]]}, {"id":"ae1455d7.c2f2c8","type":"ui_group","z":"","name":"Networking monitor","tab":"8912138.b569e7","disp":true,"width":8},{id":"8912138.b569e7","type":"ui_tab","z":"","name":"Advantech demo","icon":"dashboard"}]
```

Please note that you need Node-RED module *Dashboard* installed in order to this example work correctly.

The following flow of nodes will appear:

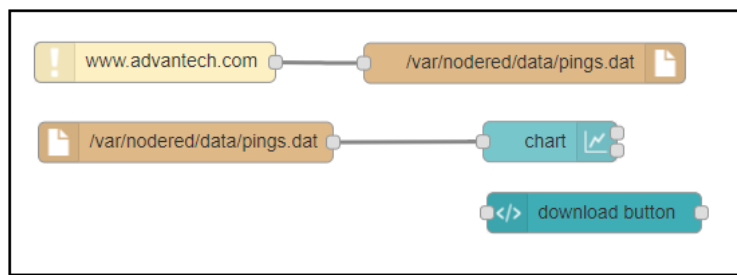


Figure 28: Example 3 – ping statistics flow

Deploy the flow with red button in the right upper corner. Now you can access the ping statistics at this address: <https://<ip-address-of-the-router>:1880/ui>

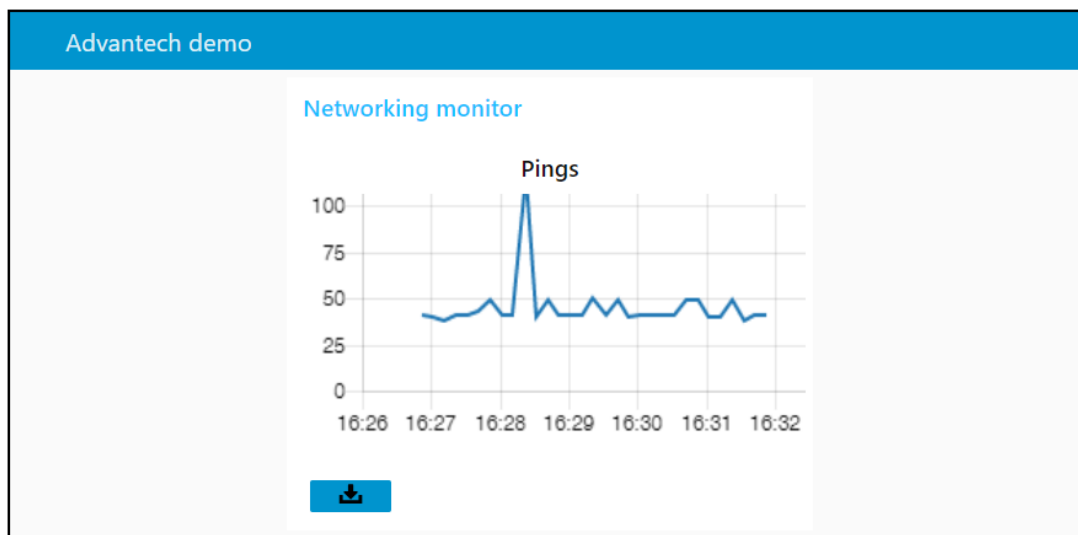


Figure 29: Example 3 – ping statistics chart and download

There is a chart with ping statistics (in milliseconds, the router has to be connected to the Internet) and the button for downloading the `var/nodered/data/pings.dat` file. *Web static files path* is used when writing – saving the ping results to the data file, and reading – viewing the statistics page (button image), downloading data file.

5.4 Example 4: Reading from supported BLE sensor in Node-RED

This example demonstrates the usage of three Ruuvi Tags for temperature and humidity monitoring.



Figure 30: Example 4 – Ruuvi Tag

```
E9:22:32:C5:3F:2B  E0-3 max
Address Type      : random
Name              : E0-3 max
Alias             : E0-3 max
Paired            : no
Trusted           : no
Blocked           : no
Legacy Pairing    : no
RSSI              : -48 dBm
Connected         : no
Manufacturer Data : 0x0499 0x031f161cc4f4fd590075fd000bb9
Services Resolved : no
Advertising Flags : 0x06
```

Figure 31: Example 4 – Status

We offer the a specialized node for the Ruuvi Tag sensors that take care of data decoding. You can simply connect RuuviTag to the BLE scanner node to get the measured values.

The sensors propagate data via the manufacturer data advertising item.

Copy this JSON code and import it as the Node-RED flow:

```
[{"id":"2945eb57.8f6a34","type":"tab","label":"Flow 2","disabled":false,"info":"","z":0,"x":250,"y":160,"wires":[{"id":"be2fd8f3.23c9e8","type":"BLE scanner","z":"2945eb57.8f6a34","name":"","servicesType":"str","autostart":true,"continuous":true,"duplicates":true,"timeout":"","x":250,"y":160,"wires":[{"id":"22af1566.e8bde2","type":"ruuvitag","z":"2945eb57.8f6a34","name":"","x":420,"y":160,"wires":[{"id":"3693f0e2.1ef39","type":"switch","z":"2945eb57.8f6a34","name":"","property":"peripheral","propertyType":"msg","rules":[{"t":"eq","v":"e92232c5412c","vt":"str"},{"t":"eq","v":"e92232c5412c","vt":"str"}],"checkall":"true","repair":false,"outputs":3,"x":560,"y":160,"wires":[{"id":"358dabb5.a5d1ac","type":"ui_gauge","z":"2945eb57.8f6a34","name":"","group":"14d04008.aa63f","order":1,"width":0,"height":0,"gtype":"gauge","title":"Temperature","label":"units","format":"{{msg.payload.temperature}}","min":0,"max":50,"colors":["#0000ff","#ffffff","#ff0000"],"seg1":"15","seg2":"35","x":750,"y":60,"wires":[]},{id":"8464c887.e37e4","type":"ui_gauge","z":"2945eb57.8f6a34","name":"","group":"14d04008.aa63f","order":2,"width":0,"height":0,"gtype":"gauge","title":"Humidity","label":"units","format":"{{msg.payload.humidity}}","min":0,"max":50,"colors":["#0000ff","#e6e600","#ca3838"],"seg1":"25","seg2":"35","x":740,"y":100,"wires":[]},{id":"6619f6f5.3502c","type":"ui_gauge","z":"2945eb57.8f6a34","name":"","group":"b5f6ead5.983858","order":1,"width":0,"height":0,"gtype":"gauge","title":"Temperature","label":"units","format":"{{msg.payload.temperature}}","min":0,"max":50,"colors":["#0000ff","#ffffff","#ff0000"],"seg1":"15","seg2":"35","x":750,"y":140,"wires":[]},{id":"5bf803f2.6134a4","type":"ui_gauge","z":"2945eb57.8f6a34","name":"","group":"b5f6ead5.983858","order":2,"width":0,"height":0,"gtype":"gauge","title":"Humidity","label":"units","format":"{{msg.payload.humidity}}","min":0,"max":50,"colors":["#0000ff","#e6e600","#ca3838"],"seg1":"25","seg2":"35","x":740,"y":180,"wires":[]},{id":"7df2dec6.c0c068","type":"ui_gauge","z":"2945eb57.8f6a34","name":"","group":"87b4700b.33d698","order":0,"width":0,"height":0,"gtype":"gauge","title":"Temperature","label":"units","format":"{{msg.payload.temperature}}","min":0,"max":50,"colors":["#0000ff","#ffffff","#ff0000"],"seg1":"15","seg2":"35","x":750,"y":220,"wires":[]},{id":"2ca03fd5.6eb208","type":"ui_gauge","z":"2945eb57.8f6a34","name":"","group":"87b4700b.33d698","order":1,"width":0,"height":0,"gtype":"gauge","title":"Humidity","label":"units","format":"{{msg.payload.humidity}}","min":0,"max":50,"colors":["#0000ff","#e6e600","#ca3838"],"seg1":"25","seg2":"35","x":740,"y":260,"wires":[]},{id":"14d04008.aa63f","type":"ui_group","z":"","name":"Container 1","tab":"9c703a4f.c1f94","disp":true,"width":6,"collapse":false},{id":"b5f6ead5.983858","type":"ui_group","z":"","name":"Container 2","tab":"9c703a4f.c1f94","disp":true,"width":6,"collapse":false},{id":"87b4700b.33d698","type":"ui_group","z":"","name":"Container 3","tab":"9c703a4f.c1f94","disp":true,"width":6,"collapse":false},{id":"9c703a4f.c1f94","type":"ui_tab","z":"","name":"Containers","icon":"dashboard","disabled":false,"hidden":false}]}
```

! This example requires the extra installed *Bluetooth*, *Node-RED Bluetooth* and *Node-Red Dashboard Router* Apps.

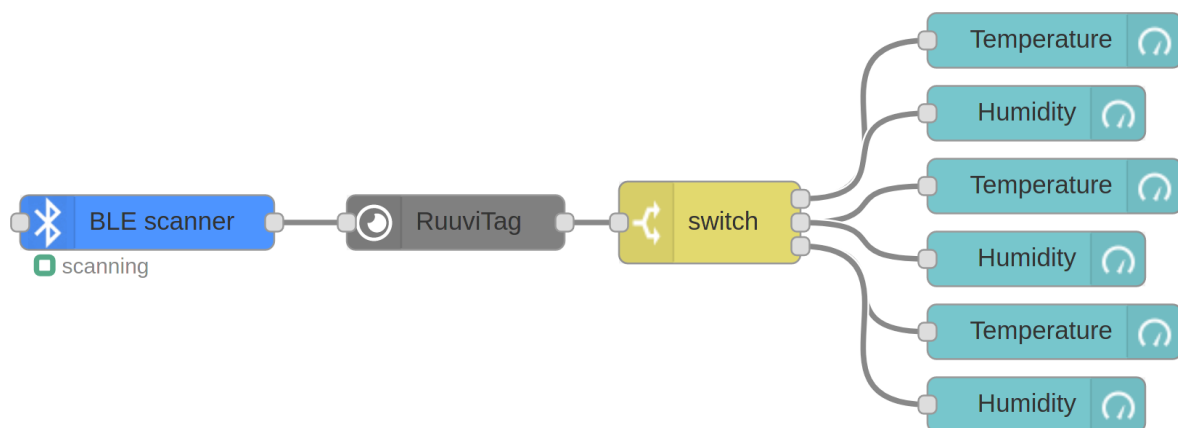


Figure 32: Example 4 – Bluetooth router app

The data are collected continuously. The Switch node splits the data from three different sensors by its addresses to three Dashboard sections. You can see the result at this address: <https://<ip-address-of-the-router>:1880/ui>

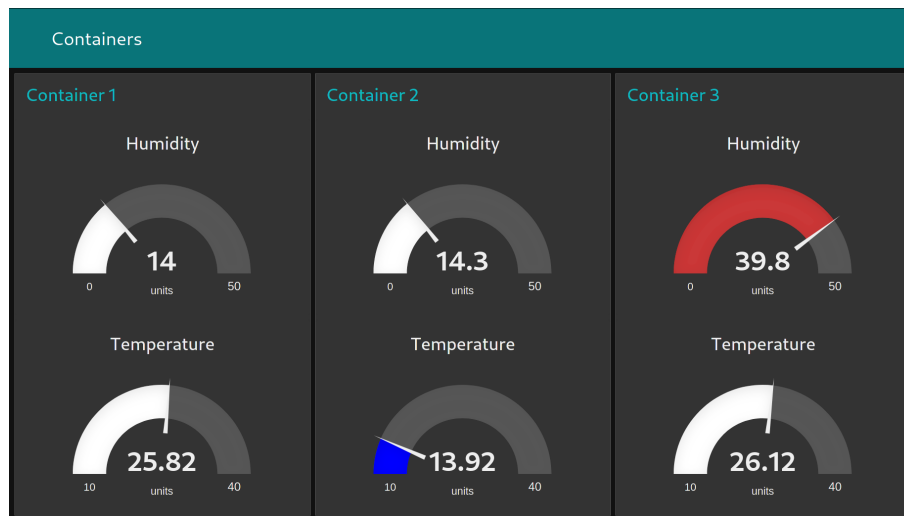


Figure 33: Example 4 – Containers

You can find more Bluetooth examples in the Bluetooth Router App documentation. Those examples are not for Node-RED but it can clarify for you some other aspects how to use Bluetooth with Advantech routers.

5.5 Example 5: Reading from and writing to unsupported BLE sensor in Node-RED

This example is about using the sensor when you don't have a specialized decoding node. It uses Xiaomi Flower Care to control the greenhouse environment.



Figure 34: Example 5 – Xiaomi Flower Care

```
C4:7C:8D:6B:9D:27  Flower care
Address Type      : public
Name              : Flower care
Alias             : Flower care
Paired            : no
Trusted           : no
Blocked           : no
Legacy Pairing    : no
RSSI              : -57 dBm
Connected         : no
UUIDs             : 0000fe95-0000-1000-8000-00805f9b34fb Xiaomi Inc.
Service Data      : 0000fe95-0000-1000-8000-00805f9b34fb 0x71209800f9279d6b8d7cc40d041002f900
Services Resolved : no
Advertising Flags : 0x06
```

Figure 35: Example 5 – Status

The sensor propagates data via the service characteristics.

Copy this JSON code and import it as the Node-RED flow:

```
{
  "id": "30182cd3.1eaddc",
  "type": "tab",
  "label": "Flow 1",
  "disabled": false,
  "info": "",
  "id": "acf17368.dd75d",
  "type": "inject",
  "z": "30182cd3.1eaddc",
  "name": "",
  "topic": "start",
  "payload": "",
  "payloadType": "str",
  "repeat": "300",
  "crontab": "",
  "once": false,
  "onceDelay": 0.1,
  "x": 124,
  "y": 40,
  "wires": [
    [
      "f13061ea.0121c8"
    ]
  ],
  "id": "6e237e59.5aa808",
  "type": "BLE device",
  "z": "30182cd3.1eaddc",
  "name": "",
  "x": 144,
  "y": 160,
  "wires": [
    [
      "b773f45.1638d88",
      "f4e53939.3e3978",
      "d9bd6fcb.738508"
    ]
  ],
  "id": "b773f45.1638d88",
  "type": "BLE in",
  "z": "30182cd3.1eaddc",
  "topic": "read",
  "characteristic": "00001a0200001000800000805f9b34fb",
  "name": "",
  "x": 370,
  "y": 40,
  "wires": [
    [
      "4e7939bd.3e098"
    ]
  ],
  "id": "ded8cf21.e47b18",
  "type": "BLE in",
  "z": "30182cd3.1eaddc",
  "topic": "read",
  "characteristic": "00001a0100001000800000805f9b34fb",
  "name": "",
  "x": 490,
  "y": 160,
  "wires": [
    [
      "1fe8af43.36f789"
    ]
  ],
  "id": "eed2cad7.3cb43",
  "type": "BLE out",
  "z": "30182cd3.1eaddc",
  "characteristic": "00001a0000001000800000805f9b34fb",
  "name": "",
  "x": 600,
  "y": 100,
  "wires": [
    [
      "f4e53939.3e3978"
    ]
  ],
  "type": "change",
  "z": "30182cd3.1eaddc",
  "name": "Set Realtime mode",
  "rules": [
    [
      "t",
      "set",
      "p": "payload",
      "pt": "msg",
      "to": "[160,31]",
      "tot": "bin"
    ]
  ],
  "action": "",
  "property": "",
  "from": "",
  "to": "",
  "reg": false,
  "x": 410,
  "y": 100,
  "wires": [
    [
      "eed2cad7.3cb43"
    ]
  ],
  "id": "1fe8af43.36f789",
  "type": "function",
  "z": "30182cd3.1eaddc",
  "name": "Realtime data",
  "func": "msg.payload = {\n  'temperature': msg.payload.readUInt16LE(0)/10,\n  'brightness': msg.payload.readUInt32LE(3),\n  'moisture': msg.payload.readInt8(7),\n  'conductivity': msg.payload.readUInt16LE(8)\n}\nreturn msg;",
  "outputs": 1,
  "noerr": 0,
  "x": 640,
  "y": 160,
  "wires": [
    [
      "f53466e8.833b18"
    ]
  ],
  "id": "f53466e8.833b18",
  "type": "BLE scanner",
  "z": "30182cd3.1eaddc",
  "name": "",
  "services": [
    "fe95"
  ],
  "serviceType": "str",
  "autostart": false,
  "continuous": false,
  "duplicates": false,
  "x": 144,
  "y": 100,
  "wires": [
    [
      "6e237e59.5aa808"
    ]
  ],
  "id": "4e7939bd.3e098",
  "type": "change",
  "z": "30182cd3.1eaddc",
  "name": "On/Off",
  "rules": [
    [
      "t",
      "set",
      "p": "payload",
      "pt": "msg",
      "to": "msg.payload.readInt8(0) < 40",
      "tot": "str"
    ]
  ],
  "action": "",
  "property": "",
  "from": "",
  "to": "",
  "reg": false,
  "x": 510,
  "y": 40,
  "wires": [
    [
      "6d316b11.624f7c"
    ]
  ],
  "id": "6d316b11.624f7c",
  "type": "signal_led",
  "z": "30182cd3.1eaddc",
  "led": "USR",
  "inverting": false,
  "initial": "0",
  "name": "",
  "x": 650,
  "y": 40,
  "wires": [
    [
      "f53466e8.833b18"
    ]
  ],
  "id": "4399313a.3126f8",
  "type": "Input Split",
  "z": "30182cd3.1eaddc",
  "name": "",
  "inputProps": [
    "payload.temperature",
    "payload.brightness",
    "payload.moisture",
    "payload.conductivity"
  ],
  "outputs": 4,
  "x": 800,
  "y": 160,
  "wires": [
    [
      "7e659451.293304",
      "d1f9663f.e9571",
      "3e469c6.3c0f564",
      "4e0f5d97.82b4c4"
    ]
  ],
  "id": "7e659451.293304",
  "type": "link out",
  "z": "30182cd3.1eaddc",
  "name": "Temperature",
  "links": [
    "7f5b8722.d6b24"
  ],
  "x": 915,
  "y": 100,
  "wires": [
    [
      "d1f9663f.e9571"
    ]
  ],
  "type": "link out",
  "z": "30182cd3.1eaddc",
  "name": "Brightness",
  "links": [
    "4399313a.3126f8",
    "x": 915,
    "y": 140,
    "wires": [
      [
        "3e469c6.3c0f564"
      ],
      "type": "link out",
      "z": "30182cd3.1eaddc",
      "name": "Moisture",
      "links": [
        "a025af7f.9297a8"
      ],
      "x": 915,
      "y": 180,
      "wires": [
        [
          "4e0f5d97.82b4c4"
        ]
      ],
      "type": "link out",
      "z": "30182cd3.1eaddc",
      "name": "Conductivity",
      "links": [
        "ad8db049.e1d3b8"
      ],
      "x": 915,
      "y": 220,
      "wires": [
        [
          "a025af7f.9297a8"
        ]
      ],
      "type": "link in",
      "z": "30182cd3.1eaddc",
      "name": "Valve",
      "links": [
        "3e469c6.3c0f564"
      ],
      "x": 55,
      "y": 360,
      "wires": [
        [
          "7afc3aa4.731944"
        ]
      ],
      "id": "4399313a.3126f8",
      "type": "link in",
      "z": "30182cd3.1eaddc",
      "name": "Slats",
      "links": [
        "d1f9663f.e9571"
      ],
      "x": 55,
      "y": 300,
      "wires": [
        [
          "90162d6b.f87d9"
        ]
      ],
      "id": "ad8db049.e1d3b8",
      "type": "link in",
      "z": "30182cd3.1eaddc",
      "name": "Batcher",
      "links": [
        "4e0f5d97.82b4c4"
      ],
      "x": 55,
      "y": 420,
      "wires": [
        [
          "1a4f3520.7f10c3"
        ]
      ],
      "id": "7f5b8722.d6b24",
      "type": "link in",
      "z": "30182cd3.1eaddc",
      "name": "Heater",
      "links": [
        "7e659451.293304",
        "x": 55,
        "y": 240,
        "wires": [
          [
            "ce4ccd99.f44678"
          ]
        ],
        "id": "ceafabe4.1d6b4",
        "type": "modbusSerial out",
        "z": "30182cd3.1eaddc",
        "port": "88e680e2.274bf",
        "slave": "30",
        "start": "2",
        "dtype": "input",
        "topic": "",
        "name": "",
        "x": 350,
        "y": 360,
        "wires": [
          [
            "7afc3aa4.731944"
          ],
          "type": "change",
          "z": "30182cd3.1eaddc",
          "name": "Valve",
          "rules": [
            [
              "t",
              "set",
              "p": "payload",
              "pt": "msg",
              "to": "msg.payload < 50 ? 1 : 2",
              "tot": "jsonata"
            ]
          ],
          "action": "",
          "property": "",
          "from": "",
          "to": "",
          "reg": false,
          "x": 150,
          "y": 360,
          "wires": [
            [
              "ceafabe4.1d6b4"
            ]
          ],
          "id": "88d99715.18d348",
          "type": "modbusSerial out",
          "z": "30182cd3.1eaddc",
          "port": "88e680e2.274bf",
          "slave": "20",
          "start": "1",
          "dtype": "coil",
          "topic": "",
          "name": "",
          "x": 350,
          "y": 300,
          "wires": [
            [
              "f53466e8.833b18"
            ]
          ],
          "id": "fee97542.aa4",
          "type": "modbusSerial out",
          "z": "30182cd3.1eaddc",
          "port": "88e680e2.274bf",
          "slave": "10",
          "start": "1",
          "dtype": "coil",
          "topic": "",
          "name": "",
          "x": 350,
          "y": 240,
          "wires": [
            [
              "8b30e2c7.b5783"
            ]
          ],
          "type": "modbusSerial out",
          "z": "30182cd3.1eaddc",
          "port": "88e680e2.274bf",
          "slave": "40",
          "start": "1215",
          "dtype": "coil",
          "topic": "",
          "name": "",
          "x": 350,
          "y": 420,
          "wires": [
            [
              "1a4f3520.7f10c3"
            ]
          ],
          "type": "change",
          "z": "30182cd3.1eaddc",
          "name": "Batcher",
          "rules": [
            [
              "t",
              "set",
              "p": "payload",
              "pt": "msg",
              "to": "msg.payload < 80 ? msg.payload*10 : 0",
              "tot": "str"
            ]
          ],
          "action": "",
          "property": "",
          "from": "",
          "to": "",
          "reg": false,
          "x": 150,
          "y": 420,
          "wires": [
            [
              "8b30e2c7.b5783"
            ]
          ],
          "id": "ce4ccd99.f44678",
          "type": "change",
          "z": "30182cd3.1eaddc",
          "name": "Heater",
          "rules": [
            [
              "t",
              "set",
              "p": "payload",
              "pt": "msg",
              "to": "msg.payload < 25 ? 1 : 0",
              "tot": "str"
            ]
          ],
          "action": "",
          "property": "",
          "from": "",
          "to": "",
          "reg": false,
          "x": 150,
          "y": 240,
          "wires": [
            [
              "fee97542.aa4"
            ]
          ],
          "id": "90162d6b.f87d9",
          "type": "change",
          "z": "30182cd3.1eaddc",
          "name": "Slats",
          "rules": [
            [
              "t",
              "set",
              "p": "payload",
              "pt": "msg",
              "to": "msg.payload > 130 ? 1 : msg.payload < 90 ? 3 : 2",
              "tot": "str"
            ]
          ],
          "action": "",
          "property": "",
          "from": "",
          "to": "",
          "reg": false,
          "x": 150,
          "y": 300,
          "wires": [
            [
              "88d99715.18d348"
            ]
          ],
          "id": "d9bd6fcb.738508",
          "type": "delay",
          "z": "30182cd3.1eaddc",
          "name": "",
          "pauseType": "delay",
          "timeout": "10",
          "timeoutUnits": "milliseconds",
          "rate": "1",
          "nbRateUnits": "1",
          "rateUnits": "second",
          "randomFirst": "1",
          "randomLast": "5",
          "randomUnits": "seconds",
          "drop": false,
          "x": 350,
          "y": 160,
          "wires": [
            [
              "ded8cf21.e47b18"
            ]
          ],
          "id": "88e680e2.274bf",
          "type": "modbusSerialConfig",
          "z": "",
          "port": "/dev/ttyS1",
          "baud": "9600",
          "data": "8",
          "parity": "none",
          "stop": "1",
          "name": ""
        ]
      ]
    ]
  ]
}
```

Note: This example requires the extra installed *Bluetooth* and *Node-RED Bluetooth Router* Apps.

The data are checked every 5 minutes. Because we have no specialized node for this sensor, we process its data with general nodes. We need to know the GATT services and characteristics for this. Advantech does not provide this information. You have to ask the sensor vendor. Only the necessary selected information for this example follows:

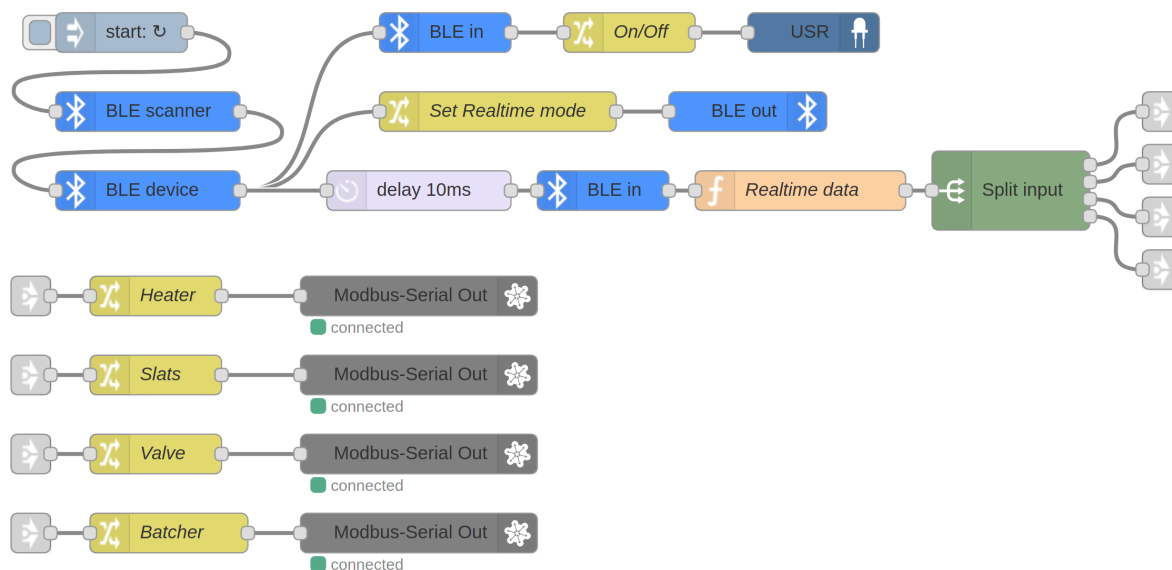


Figure 36: Example 5 – Nodes

Service UUID	Service short UUID	Usage
0000fe95-0000-1000-8000-00805f9b34fb	fe95	device discovery
00001204-0000-1000-8000-00805f9b34fb	1204	data

Table 3: Relay control commands

Characteristics for 1204 service:

Characteristic UUID	Read/Write	Usage
00001a00-0000-1000-8000-00805f9b34fb	write	mode control

Write 2-bytes sequence 0xa01f to read realtime data from next characteristic.

00001a01-0000-1000-8000-00805f9b34fb	read	realtime data
--------------------------------------	------	---------------

The data are Little Endian encoded. Bytes 00-01 contain the temperature in 0.1 °C as uint16, bytes 03-06 brightness in luxes as uint32, byte 07 moisture in % as uint8 and bytes 08-09 conductivity in uS/cm as uint16.

00001a02-0000-1000-8000-00805f9b34fb	read	battery and FW
--------------------------------------	------	----------------

The first byte contains percentage the battery level.

Table 4: Relay control commands

The BLE scanner scans until it finds a sensor with the specific service. This example counts on one sensor only, so there are no address checks and no duplicity allowed. Realtime data are read with a small delay so that the sensor is able to change the mode. Next, they are decoded with the general function node.

The flow controls the greenhouse environment via the Modbus devices (e.g. watering with a Modbus controlled water valve or fertilizing with a Modbus controlled dispenser) according to measured values. It also lights on/off the user LED on the router according to the sensor battery level.

You can find more Bluetooth examples in the Bluetooth Router App documentation. Those examples are not for Node-RED but it can clarify for you some other aspect how to use Bluetooth with Advantech routers.

5.6 Resources

When implementing your own flows, you will find a wealth of resources provided by Node-RED here:

<https://nodered.org/docs/>

6. Related Documents

- [1] Advantech Czech: **SmartStart Configuration Manual** (MAN-0022-EN)
- [2] Advantech Czech: **SmartFlex Configuration Manual** (MAN-0023-EN)
- [3] Advantech Czech: **SmartMotion Configuration Manual** (MAN-0024-EN)
- [4] Advantech Czech: **ICR-3200 Configuration Manual** (MAN-0042-EN)
- [5] User Modules: icr.advantech.cz/user-modules
- [6] JS Foundation: <https://nodered.org/>
- [7] Advantech Czech: **Node.js Application note** (APP-0080-EN)
- [8] Advantech Czech: **Bluetooth Application note** (APP-0098-EN)



[EP] Product-related documents and applications can be obtained on *Engineering Portal* at icr.advantech.cz address.