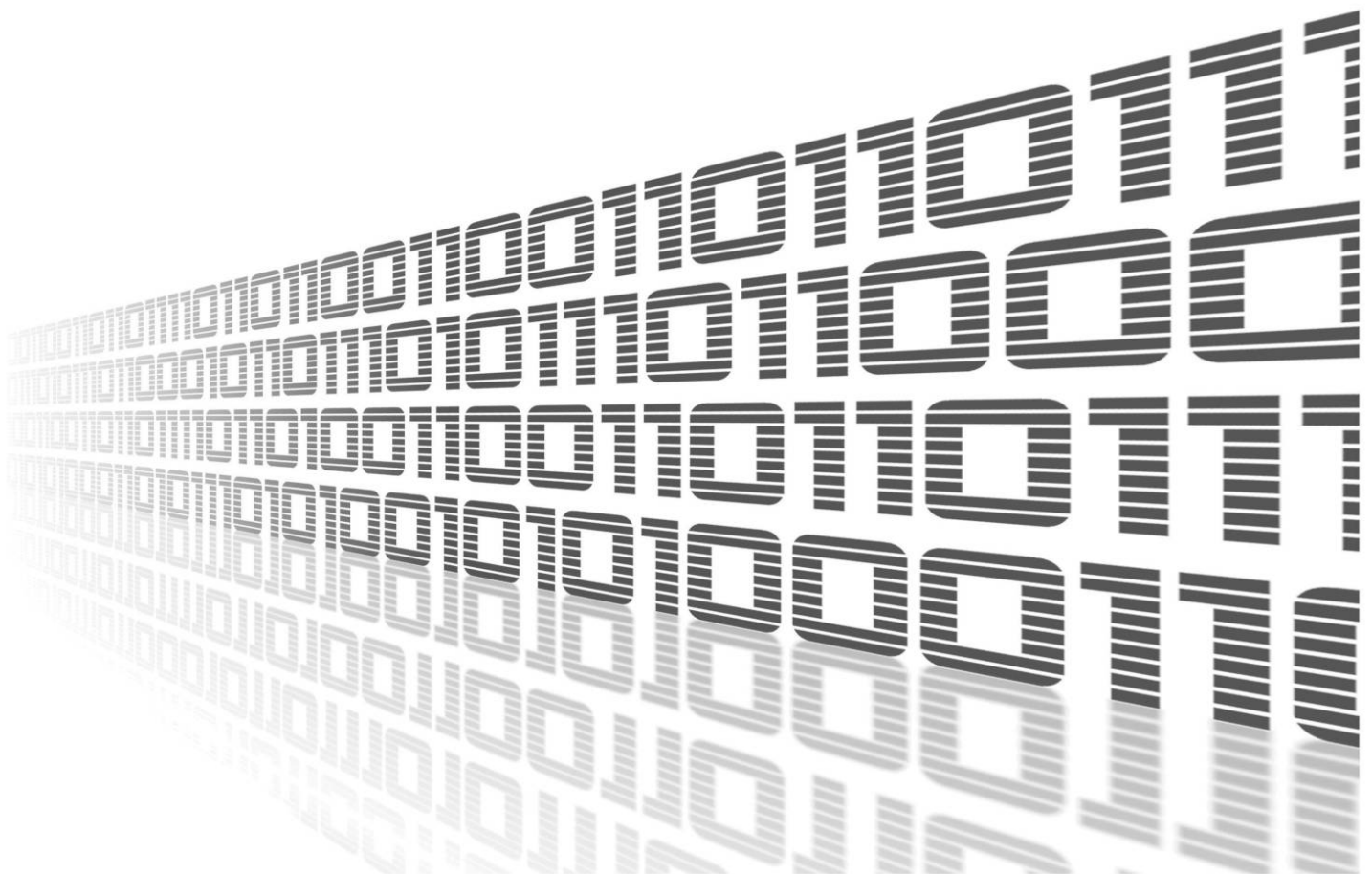


ADVANTECH



Node-RED



© 2024 Advantech Czech s.r.o. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photography, recording, or any information storage and retrieval system without written consent. Information in this manual is subject to change without notice, and it does not represent a commitment on the part of Advantech.

Advantech Czech s.r.o. shall not be liable for incidental or consequential damages resulting from the furnishing, performance, or use of this manual.

All brand names used in this manual are the registered trademarks of their respective owners. The use of trademarks or other designations in this publication is for reference purposes only and does not constitute an endorsement by the trademark holder.

Used symbols



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



Attention – Problems that can arise in specific situations.



Information – Useful tips or information of special interest.



Example – Example of function, command or script.

Contents

1. Changelog	1
1.1 Node-RED	1
1.2 Node-RED AWS	3
1.3 Node-RED Azure	3
1.4 Node-RED BACnet	3
1.5 Node-RED Bluetooth	4
1.6 Node-RED Dashboard	4
1.7 Node-RED dnp3	4
1.8 Node-RED Filesystem	5
1.9 Node-RED FTP	5
1.10 Node-RED GPS	5
1.11 Node-RED gZIP	6
1.12 Node-RED KNX	6
1.13 Node-RED LwM2M	7
1.14 Node-RED Modbus	7
1.15 Node-RED OPC UA	8
1.16 Node-RED other	8
1.17 Node-RED PLC EtherNet-IP	8
1.18 Node-RED PLC Melsec	8
1.19 Node-RED Splunk	9
2. Node-RED Router App Description	10
2.1 Node-RED Dependency and Extensions	11
2.2 Node-RED Router App	13
3. Node-RED Configuration	14
3.1 Run and Access the Node-RED engine	14
3.2 Flow Files Path	15
3.3 Web Static Files FS Path	15
3.4 Flow Webeditor	16
3.5 Node-RED Status (Log)	16
4. Available Nodes and Version	17
4.1 Node-RED Version	17
4.2 Available Nodes	17
4.3 Router Nodes	19
4.4 Router Apps for Additional Nodes	20
4.4.1 Router App Node-RED / AWS	20
4.4.2 Router App Node-RED / Azure	21
4.4.3 Router App Node-RED / BACnet	22
4.4.4 Router App Node-RED / Bluetooth	22
4.4.5 Router App Node-RED / Dashboard	23
4.4.6 Router App Node-RED / dnp3	24
4.4.7 Router App Node-RED / Filesystem	24
4.4.8 Router App Node-RED / FTP	25

4.4.9 Router App Node-RED / GPSd	25
4.4.10 Router App Node-RED / gzip	26
4.4.11 Router App Node-RED / KNX	26
4.4.12 Router App Node-RED / LwM2M	27
4.4.13 Router App Node-RED / Modbus	27
4.4.14 Router App Node-RED / OPC UA	28
4.4.15 Router App Node-RED / PLC - EtherNet/IP	28
4.4.16 Router App Node-RED / PLC - Melsec	29
4.4.17 Router App Node-RED / Splunk HEC	29
5. Advanced topics	30
5.1 File locations	30
5.2 How to distribute your own flows to the routers	30
5.3 Persistent variable storage	31
5.4 Node "router"	31
5.5 Building the Custom Nodes for Node.js/Node-RED	32
5.5.1 Preparing JavaScript Only Nodes on PC	32
5.5.2 Preparing Nodes Using Other Languages on PC	33
5.5.3 Installing nodes directly on router	35
6. Flow Examples	36
6.1 Example 1: User LED Turn On and Off	37
6.2 Example 2: HTTP Endpoint	38
6.3 Example 3: Ping Statistics	39
6.4 Node-RED Bluetooth Examples	42
6.4.1 Example 4: Reading from supported BLE sensor in Node-RED	42
6.4.2 Example 5: Reading from and writing to unsupported BLE sensor in Node-RED	45
6.5 Node-RED Modbus Examples	49
6.5.1 Example 6: Writing to the User-Defined Modbus Server	50
6.5.2 Example 7: Reading from the User-Defined Modbus Server	51
6.5.3 Example 8: Writing a sequence to the User-Defined Modbus Server	52
6.5.4 Example 9: Reading a sequence from the User-Defined Modbus Server	53
6.5.5 Example 10: Reading Coil status from a Modbus RTU Relay using the Modbus Protocol	55
6.5.6 Example 11: Writing Coil status to a Modbus RTU Relay using the Modbus Protocol	56
6.5.7 Example 12: Reading from a Real sensor using the Modbus Protocol	57
6.6 Resources	58
7. Related Documents	59

List of Figures

1 An example of the Node-RED editor running in an Advantech router	10
2 Node.js to make Node-RED work and additional nodes router apps	12
3 Node-RED router app menu	13
4 Node-RED router app configuration – run Node-RED	14
5 Node-RED login in the router	15
6 Node-RED Status: Version information and Log	16
7 List of available nodes in main Node-RED Router App	18

8	AWS nodes	20
9	Azure node	21
10	BACnet nodes	22
11	Additional Bluetooth nodes	22
12	Dashboard nodes	23
13	Additional dnp3 nodes	24
14	Additional Filesystem nodes	24
15	Additional FTP and SFTP nodes	25
16	Additional GPSd node in input category	25
17	Additional gzip node	26
18	Additional knx nodes	26
19	LwM2M nodes	27
20	Modbus nodes in	27
21	Additional OPC UA nodes	28
22	PLC EtherNet/IP nodes	28
23	PLC Melsec nodes	29
24	Additional Splunk event collector nodes	29
25	Persistent storage	32
26	Router node	32
27	Installation with npm tool on the router	35
28	Top right menu – import from clipboard	36
29	Import nodes via JSON	36
30	Example 1 – user LED flow	37
31	Example 1 – user LED turned on	37
32	Example 2 – imported nodes to deploy	38
33	Example 2 – HTTP response	38
34	Example 3 – Web static file FS path configuration	39
35	Example 3 – ping statistics flow	40
36	Example 3 – ping statistics chart and download	41
37	Example 4 – Ruuvi Tag	42
38	Example 4 – Status	42
39	Example 4 – Bluetooth router app	44
40	Example 4 – Presentation of the environment conditions with Dashboard UI	44
41	Example 5 – Xiaomi Flower Care	45
42	Example 5 – Status	45
43	Example 5 – Nodes	47
44	Modbus Flex Server	49
45	Example 6 – Write Voltage Nodes	50
46	Example 7 – Read Voltage Nodes	51
47	Example 8 – Writing Sequence Nodes	52
48	Example 9 – Reading Sequence Nodes	53
49	Example 9 – Voltage Chart	54
50	Example 10 – Reading Coil Nodes	55
51	Example 11 – Writing Coil Nodes	56
52	Example 12 – Real Sensor Nodes	57

List of Tables

1	Configuration items description	14
---	---------------------------------	----

2	Router nodes	19
3	Relay control commands	47
4	Relay control commands	47

1. Changelog



This Router App has been tested on a router with firmware version 6.3.10. After updating the router's firmware to a higher version, make sure that a newer version of the Router App has not also been released, as it is necessary to update it as well for compatibility reasons.

1.1 Node-RED

1.0.0 (2016-12-19)

- First release.

1.0.1 (2017-03-13)

- Added new modules - Dashboard, SNMP, Ping.

1.0.2 (2017-10-02)

- Modified for build from sources in build system as standard router modules.
- Split Node.js and Node-RED to separate modules.
- Updated Node-RED to 0.17.5.
- Moved nodes to /usr/lib.
- Added new router nodes - LED and SMS Sender.

1.1.0

- Fixed port setting.
- Added loglevel setting to GUI.
- Changed logging to logs rotate.

1.1.1

- Added script for reports.

1.1.2

- Fixed labels in the ping and serial nodes.

1.1.3 (2018-08-10)

- Fixed generally i18n for 3rd party nodes texts.
- Added iptables rule to work with firmware "Enable filtering of locally destined packets" option on.
- Added a new option "Web static files path" for file access via Node-RED's web to module GUI configuration.
- Moved the Modbus-TCP node to a separated module together with a new Modbus-Serial node.

2.0.0 (2019-11-13)

- Introduced off-line building.
- Added license info page to module webconfig.
- Fixed install script to avoid needless Node-RED restarts.
- Updated version Node-RED 0.20.5, Dashboard 2.14.0, Ping 0.0.16, SNMP 0.0.21.
- Optimized size - removed unused files (docs, examples, test. . .).
- Polished log - solved significant warnings, removed insignificant warnings.
- Added nodes: timed-counter, input split, string.
- Prepared for new GCC 7.4.
- Prepared for new kernel 4.14.
- Prepared for V4 platform.
- Fixed not working tail and watch nodes (watch requires FW 6.2.1 or later).
- Added nodes for XBus, SMS receiving and HW monitoring.
- Reworked binary I/O and LED nodes.

2.0.1 (2020-09-29)

- Updated CSS and HTML code to match firmware 6.2.0+.

2.1.0 (2021-05-06)

- Added nodes Big Timer, Credentials, Float, Loop.
- Removed node Sentiment.
- Added the configuration option for the flows storage place.
- Added the configuration option for enabling/disabling the flow editor.
- Added the hidden configuration option for the credential storage secret.
- Fixed Binary Input node (switching logic).
- Fixed the issue with internal path info when nodes are loaded from the "nodes" folder.
- Fixed some missing licenses.
- Updated String node to 1.0.0.
- Moved license information to Node.js UM.
- Flows are included in router's reports.

2.2.2 (2022-03-10)

- Updated to Node-RED 2.2.2.
- Changed numbering.

- Changed path to 3rd party nodes.
- Added buffer-parser node.
- Added e-mail node (previously it was part of Node-RED).
- Added router node to global.

2.2.2-1 (2022-05-10)

- Added Configuration nodes.

2.2.2-2 (2022-09-08)

- Added the edge-trigger nodes.
- Modified searching the palette nodes to find it in a root node_modules folder in a symlinked tree.

2.2.2-3 (2023-01-09)

- Node-RED init script sets a minimal TLS version in compliance with the system http service settings.

3.0.2 (2023-04-18)

- Updated Node-RED to 3.0.2.
- Updated nodes email (1.19.1), bigtimer (2.8.5), ping (0.3.3), serialport (1.0.3), snmp (2.0.0), timed-counter (0.1.0).

3.0.2-1 (2024-01-26)

- Added description and summary files
- Recompiled with ModulesSDK 2.1.0

1.2 Node-RED AWS

0.7.0 (2022-01-19)

- The first release (node-red-contrib-aws 0.7.0).

1.3 Node-RED Azure

0.2.5 (2022-01-19)

- The first release (node-red-contrib-azure-iot-device 0.2.5).

0.2.6 (2023-04-25)

- Updated to 0.2.6 version.

1.4 Node-RED BACnet

0.2.7 (2023-04-28)

- The first release (node-red-contrib-bacnet 0.2.7).

1.5 Node-RED Bluetooth

0.9.1 (2021-07-28)

- First release (node-red-contrib-noble-bluetooth 0.9.1, node-red-contrib-ruuvitag node 0.1.0).

0.9.1-1 (2023-04-28)

- Only rebuilt for Node.js 18.x to fit API.

1.6 Node-RED Dashboard

3.1.6 (2022-03-09)

- First release of Node-RED Dashboard as standalone Router App Previously it was included in main Node-RED Router App
- Added other widgets: context menu, digital display, led, level, list, media, multistate switch, svg, table, timeline, upload.

3.4.0 (2023-04-20)

- Updated Dashboard to 3.4.0.
- Updated UI Table (0.4.3) and UI Upload (0.7.0).

1.7 Node-RED dnp3

1.0.0 (2022-01-24)

- First release of dnp3 node with OpenDNP3 3.1.1.

1.0.1 (2022-08-16)

- Updated underlying OpenDNP3 lib to 3.1.2.

1.1.0 (2023-04-05)

- Added timestamp support to input msg of outstation out node.
- Fixed keep-alive parameter.
- Fixed crash on control command for unexisting datapoint.
- Modified general unsolicited configuration.
- Fixed time sync parameter.
- Fixed channel closing.
- Added the info logs for channel opening/closing.
- Fixed crash when TCP is already opened.

1.1.1 (2023-05-02)

- Rebuilt for Node.js 18.x to fit API.

1.8 Node-RED Filesystem

1.0.0 (2022-04-05)

- First release (node-red-contrib-filesystem node ver. 1.0.0).

1.9 Node-RED FTP

1.0.0 (2017-10-18)

- First release with node-red-contrib-ftp 0.0.2.

1.0.1 (2018-08-05)

- Fixed restarting in install/deinstall script.

1.0.2 (2019-03-22)

- Fixed install script to avoid needless Node-RED restarts.

2.0.0 (2019-11-13)

- Upgraded to node-red-contrib-ftp 0.0.5 with SFTP support.
- Added FTP function mkdir, rmdir.
- Added SFTP node (node-red-contrib-better-sftp 0.0.7).
- Optimized size - removed unused files (docs, examples, test. . .).

0.0.6 (2021-08-11)

- Updated FTP node to 0.0.6.
- Updated FTPS node to 0.0.12.
- Changed numbering.
- Changed folder.

0.0.8 (2023-04-25)

- Updated FTP node to 0.0.8.

1.10 Node-RED GPS

1.0.0 (2017-10-18)

- First release (node-red-contrib-gpsd 1.0.1).

1.0.1 (2018-08-05)

- Fixed restarting in install/deinstall script.

1.0.2 (2019-03-22)

- Fixed install script to avoid needless Node-RED restarts.

2.0.0 (2020-01-28)

- Updated to 1.0.2.Optimized size - removed unused files (docs, examples, test. . .).
-

1.0.4 (2021-07-28)

- Updated to 1.0.4.
- Changed numbering.
- Changed folder.
- Changed category.

1.0.7 (2023-04-25)

- Updated to 1.0.7.

1.11 Node-RED gZIP

1.0.0 (2017-10-18)

- First release (gzip node ver. 0.0.3).

1.0.1 (2018-08-05)

- Fixed restarting in install/deinstall script.

1.0.2 (2019-03-22)

- Fixed install script to avoid needless Node-RED restarts.

2.0.0 (2019-11-13)

- Optimized size - removed unused files (docs, examples, test. . .).

0.0.3 (2021-07-28)

- Changed numbering.
- Changed folder.

1.12 Node-RED KNX

1.0.0 (2019-03-08)

- First release (node-red-contrib-knxjs node ver. 1.0.6).

2.0.0 (2019-11-13)

- Nothing changed, renumbered for new Node.js version only.

1.0.6 (2021-07-28)

- Changed numbering.

- Changed folder.
- Changed category.

1.3.35 (2022-03-21)

- Switched from knxjs to knx-ultimate nodes.

1.4.15 (2023-04-25)

- Updated to version 1.4.15.

1.13 Node-RED LwM2M

2.12.3 (2022-08-25)

- The first release (node-red-contrib-lwm2m 2.12.3).

2.12.3-1 (2023-05-02)

- Rebuilt for Node.js 18.x to fit API.

1.14 Node-RED Modbus

1.0.0 (2018-08-05)

- First release (modbus-serial node ver. 0.0.8, modbus-tcp node ver. 1.2.1 from master 2018-08-05) (Modbus-TCP node was moved from the main Node-RED module).

1.0.1 (2019-03-22)

- Fixed install script to avoid needless Node-RED restarts.

2.0.0 (2020-05-12)

- Modbus serial updated to 0.0.11.
- Modbus TCP updated to 1.2.3.
- Optimized size - removed unused files (docs, examples, test. . .).

5.21.2 (2022-03-09)

- Changed to different node node-red-contrib-modbus 5.14.1.
- Changed numbering.
- Changed folder.
- Changed category.

5.26.0 (2023-04-25)

- Updated to version 5.26.0.

1.15 Node-RED OPC UA

2.0.0 (2021-04-11)

- First release (node-red-contrib-opcua 0.2.113).

0.2.263 (2022-03-21)

- Updated to 0.2.262.
- Changed numbering.
- Changed folder.
- Changed category.

0.2.304 (2023-04-25)

- Updated to version 0.2.304.

0.2.307 (2023-06-20)

- Minor updated to version 0.2.307 to fix bug.

1.16 Node-RED other

1.0.0 (2022-01-31)

- First release of ugly non public nodes.
- Node for modifying flow itself.

1.0.1 (2022-09-14)

- Changed a deploy type from "flows" to "full".

1.0.2 (2022-10-13)

- Fixed to use httpAdminRoot setting API paths.

2.0.0 (2023-02-23)

- Added node for starting/restarting daemons. When Node-RED (re)start daemon directly, libuv changes owning of Node-RED resources to these daemons, what causes problems. Additionally this node also allows to restart Node-RED itself.

1.17 Node-RED PLC EtherNet-IP

1.1.3 (2022-01-05)

- The first release (node-red-contrib-cip-ethernet-ip 1.1.3).

1.18 Node-RED PLC Melsec

1.2.1 (2022-01-04)

- The first release (node-red-contrib-mcprotocol 1.2.1).

1.19 Node-RED Splunk

1.0.0 (2018-09-21)

- First release (Node-RED http-event-collector node ver. 0.1.7).

1.0.1 (2019-03-22)

- Fixed install script to avoid needless Node-RED restarts.

2.0.0 (2019-11-13)

- Optimized size - removed unused files (docs, examples, test. . .).

0.2.0 (2021-07-28)

- Updated to 0.2.0.
- Changed numbering.
- Changed folder.

2. Node-RED Router App Description

Router app *Node-RED* is not contained in the standard router firmware. Uploading of this router app is described in the Configuration manual (see Chapter [Related Documents](#)). There is a dependency for *Node-RED* router app to be installed in the router – follow the instructions in Chapter [2.1](#). **This router app is only compatible with v3 and v4 platform routers!**

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 Router App* for Advantech routers allows Node-RED to run in v3 and v4 routers as a software router app. 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](#), 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 reuse. 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/>

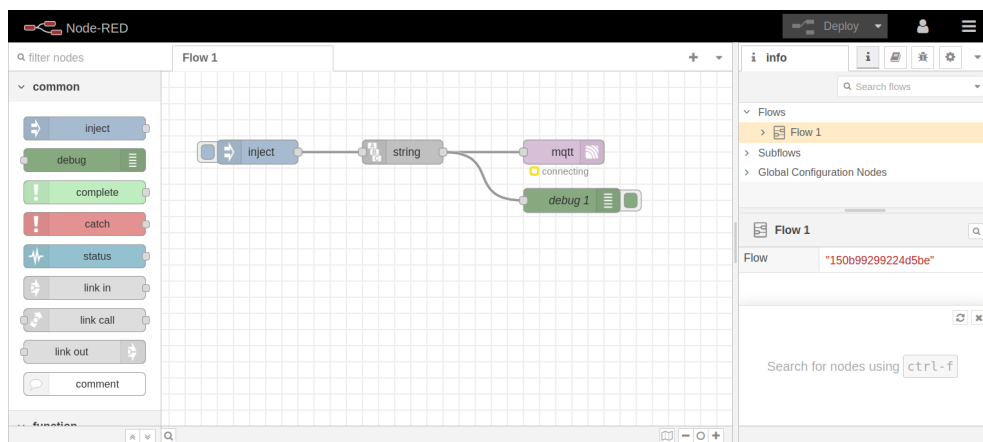


Figure 1: An example of the Node-RED editor running in an Advantech router

Flows definitions are part of the router report automatically, so if you want to send the report to Support, it is unnecessary to attach flow definitions separately. The report does not contain data declared by the node as confidential (credentials).

2.1 Node-RED Dependency and Extensions



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

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

- AWS
- Azure
- BACnet
- Bluetooth
- Dashboard
- dnp3
- Filesystem
- FTP
- GPSd
- gzip
- KNX
- LwM2M
- Modbus
- OPC UA
- PLC - EtherNet/IP
- PLC - Melsec
- Splunk event collector.

See the figure below. These additional nodes are described in Chapter 4.4. If you need an other node not available through Advantech Router App, see chapter 5.5.



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



Please always use 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 router apps

2.2 Node-RED Router App

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

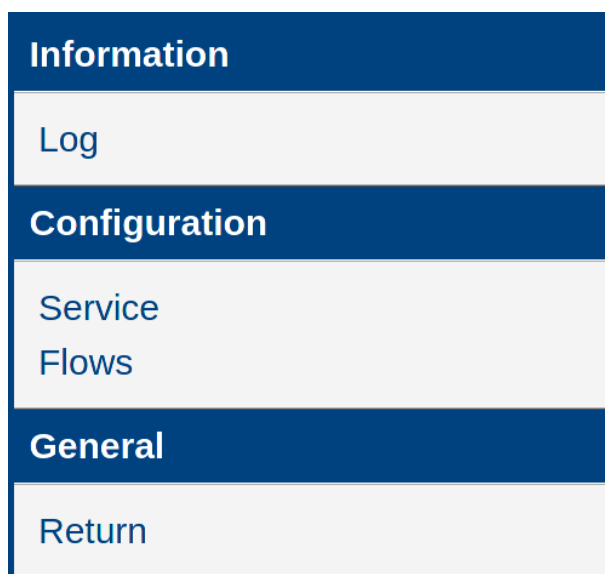


Figure 3: Node-RED router app menu

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 the flow editor. The *Return* item in the *General* section is to return to the higher menu of the router.

3. Node-RED Configuration

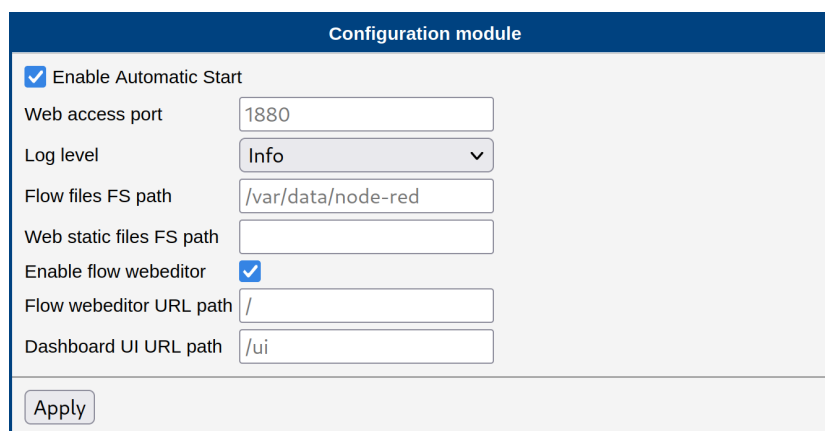


Figure 4: Node-RED router app configuration – run Node-RED

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. The default port is 1880.
Log level	Choose the severity of the information you want to see in the log. See the chapter 2.4 for details.
Flow files FS path	Specify the path, where the flow definitions will be stored. See the chapter 2.2 for details. The default value is /var/data/node-red.
Web static files FS path	Specify the path from which Node-RED web server serves files. See the chapter 2.3 for details.
Enables flow webeditor	You can uncheck this option on the production routers so the end users can not see/access Flows editor. See the chapter 3.4 for details.
Flow webeditor URL path	The path part of URL through which user can access the flow editor. See the chapter 3.4 for details.
Dashboard UI URL path	The path part of URL through which user can access the Dashboard web GUI. See the chapter 3.4 for details.

Table 1: Configuration items description

3.1 Run and Access the Node-RED engine

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

1. In router app's menu, go to *Configuration, Node-RED* and check the *Enable Automatic Start*. Press the *Apply* button. Node-RED will start and 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 its status information on the *Log* page – "Started flows" row indicates fully started Node-RED engine. You can also enable or disable the flow editor - a useful way to disable the end user's access to the flow editor.

2. Click on menu item *Flows* to access the flow editor or you can manually enter the address https://<router_ip>:<port_number>/<webadmin_path> (Make user 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.

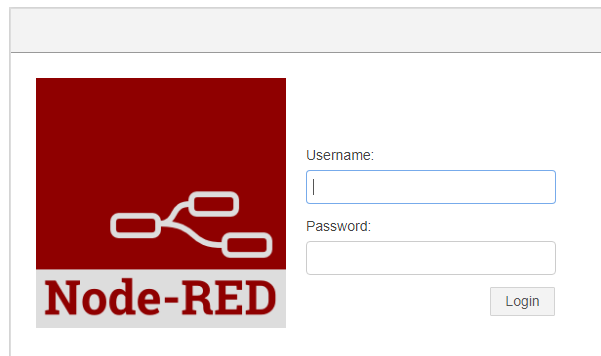


Figure 5: Node-RED login in the router

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

Admin and User profiles have the 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>

3.2 Flow Files Path

You can specify the path where the flow definitions will be stored. This is the GUI equivalent to *flowFile* option in the *setting.js* file without the file name. The 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 a huge benefit of being able to archive flow as a classic Router App and install it as usual afterward. See chapter [5.2](#) for more details.

3.3 Web Static Files FS Path

There is a *Web static files FS 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 [5.1](#)). It's a path which the Node-RED web server serves files from. It's useful for static web files (e.g., images) or downloading collected data as a file. See the example of both in Chapter [6.3](#).



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

3.4 Flow Webeditor

It is possible to customize access to Node-RED GUI. You can prohibit a user access to Node-RED flow editor on production routers by unchecking *Enables flow webeditor*. When there is a Dashboard GUI for a running flow, admin can set a custom URL path for access. Set *Dashboard UI URL path* to `/` and GUI will be available e.g. on <https://192.168.1.1:1880/> For convenient developing in this case a developer can set Flow webeditor URL path to e.g. `/admin` to not have conflict between Dashboard and webeditor paths and the webeditor then will be accessible on <https://192.168.1.1:1880/admin>.

3.5 Node-RED Status (Log)

You can choose various Log levels 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 router app 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. Users can find a version of all other individual nodes in the Node.js Router App details on the Licenses page.

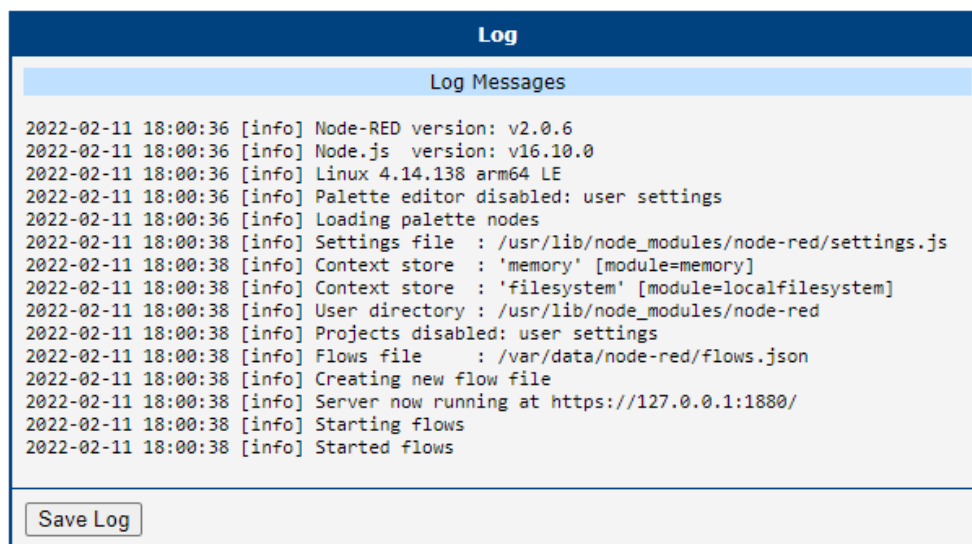


Figure 6: Node-RED Status: Version information and Log

4. Available Nodes and Version

4.1 Node-RED Version

For more information about the software versions, see the *Log* page in the *Node-RED* router app or the *Licenses* page in the *Node.js* router app. The current Router Apps contain *Node-RED* version 3.0.2 and *Node.js* version 18.15.0.

4.2 Available Nodes

Advantech ships Node-RED and all related nodes in Router App packages and other router's additional software. The main Router App Node-RED contains the Node-RED itself and these commonly useable nodes:

- Node packages required by Nore-RED
- Router nodes – special set designed for Advantech routers – see Chapter 4.3
- 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>
- Edge Trigger – <https://flows.nodered.org/node/node-red-contrib-edge-trigger>
- 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>

Other nodes are available in the separated Router Apps described in chapters 4.4 below. Figure 7 lists a complete graphical overview of the nodes available to drag and drop in the Node-RED UI.

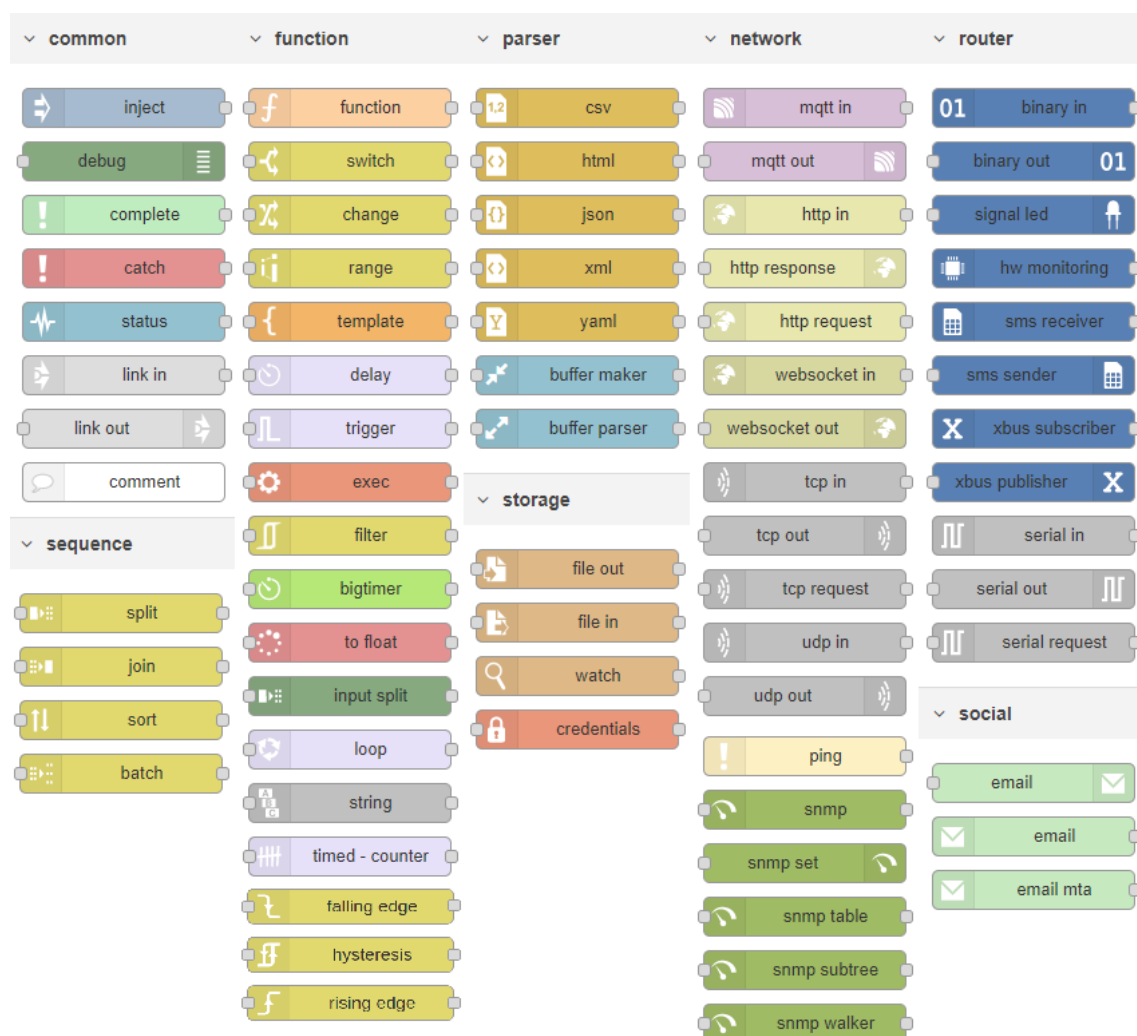


Figure 7: List of available nodes in main Node-RED Router App



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

;other node that Advantech does not provide as a part of the Router Apps, you can try to prepare it for the router yourself (see chapter 5.5) or you can send a request to cellular.info@advantech.com.

4.3 Router Nodes

Advantech provides custom router nodes to access router-specific features. The list of available router-related nodes is in the table below. When clicking on the nodes, more detailed documentation is contained in the Node-RED UI.





















Node	Description
 get configuration 	Gets a router configuration (both Firmware and all Router App configurations).
 set configuration 	Sets a router configuration (both Firmware and all Router App configurations).
 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 temperature 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 	Publishes messages to XBus.

Table 2: Router nodes

4.4 Router Apps for Additional Nodes

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

4.4.1 Router App Node-RED / AWS

Adding this router app to the router will add a collection of Node-RED nodes for AWS. All nodes cover all cloud services, and almost all nodes are direct wrappers for the AWS Javascript API, so consult the API docs for information about available parameters.

See more information in the official documentation:

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



Figure 8: AWS nodes

4.4.2 Router App Node-RED / Azure

Adding this router app to the router will add the Azure IoT device Node-RED node that can be used to connect Node-RED to the Azure IoT platform. It can connect to Azure IoT Hub, Azure IoT Central, and use Azure IoT Edge as a transparent gateway.

The Azure Device node can be used for:

- sending telemetry
- receiving and responding to command
- receiving desired properties
- updating reported properties
- receiving C2D messages

See more information in the official documentation:

<https://flows.nodered.org/node/node-red-contrib-azure-iot-device>

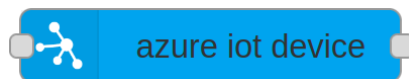


Figure 9: Azure node

4.4.3 Router App Node-RED / BACnet

Adding this router app to the router will add a set of BACnet nodes used to building Automation and Control Networks Protocol toolbox for Node-RED.

See more information in the official documentation:

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

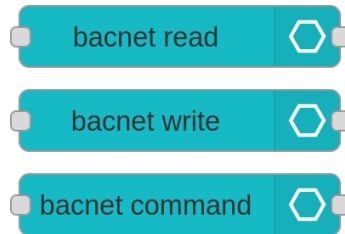


Figure 10: BACnet nodes

4.4.4 Router App Node-RED / Bluetooth

Adding this router app 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 [6.4.1](#) and [6.4.2](#) on how to use this nodes. You can find more about Bluetooth on Advantech routers in the Bluetooth Router App manual. See more information in the official documentation: <https://flows.nodered.org/node/node-red-contrib-noble-bluetooth>



This router app requires the Bluetooth Router App to be installed on the router.



It is necessary to restart Node-RED Router App when user restarts Bluetooth Router App.

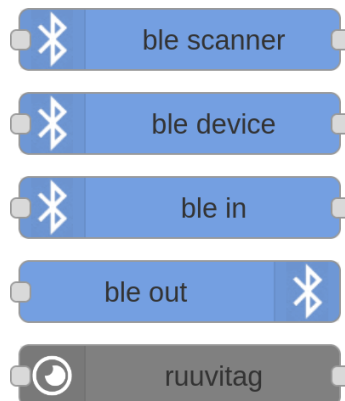


Figure 11: Additional Bluetooth nodes

4.4.5 Router App Node-RED / Dashboard

The dashboard lets you create a web UI to interact with your Node-RED flows/application. For example, it can show live data from the input nodes, or the user can control the behavior of the flow. Adding this router app 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>

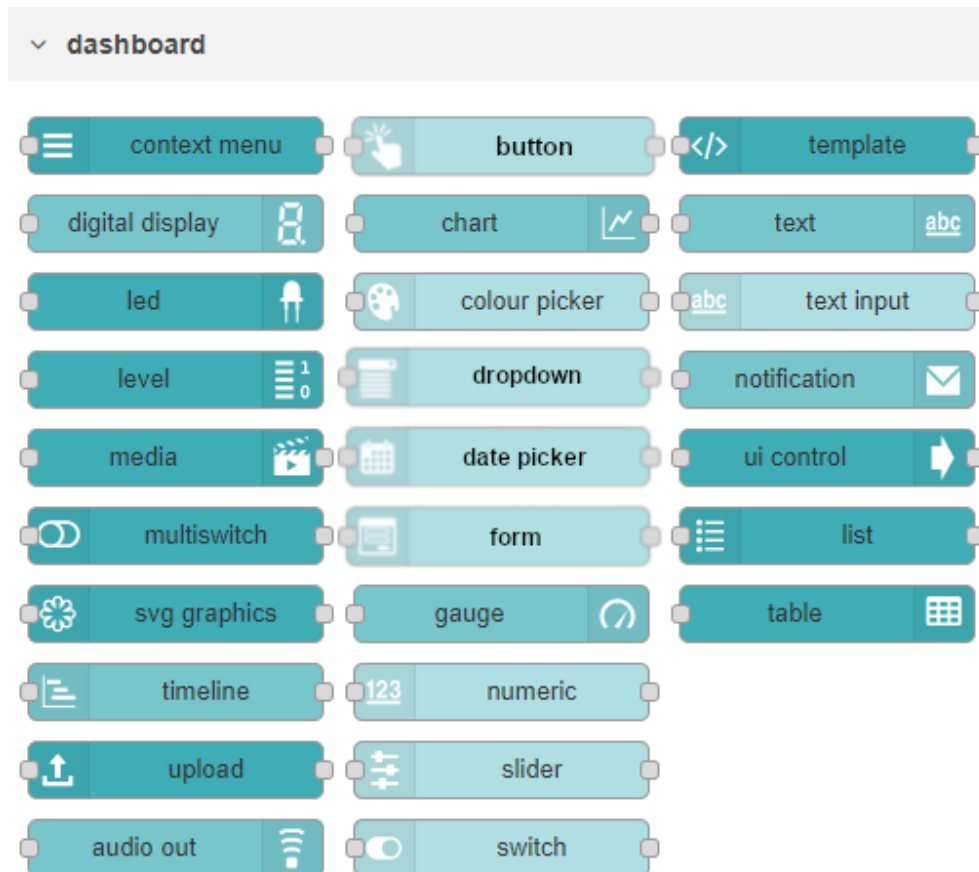


Figure 12: Dashboard nodes

4.4.6 Router App Node-RED / dnp3

Router can act as a dnp3 outstation with these nodes. It can send the binary input/output, double binary input, analog input/output, counter and octet string datapoints to a master and receive the binary output and analog output datapoints from the master.

For download visit our *Router App* section at: <https://icr.advantech.com/products/software/user-modules/category/Node-RED#node-red-dnp3-node>

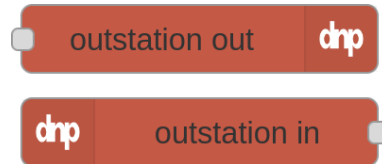


Figure 13: Additional dnp3 nodes

4.4.7 Router App Node-RED / Filesystem

These nodes provide operations over the filesystem similar to the shell commands cp, mv, ln, rm mkdir, ls, stat. See more information in the official documentation:

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

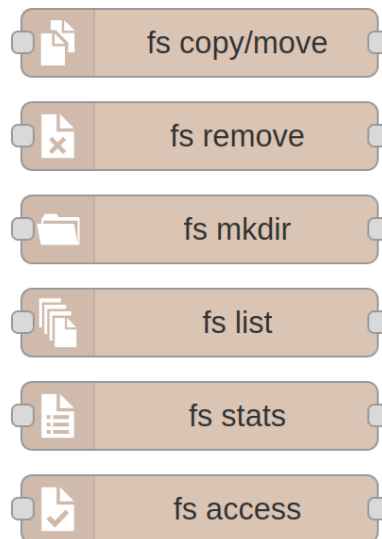


Figure 14: Additional Filesystem nodes

4.4.8 Router App Node-RED / FTP

Adding this router app 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, and 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>

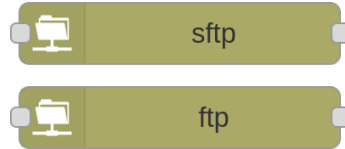


Figure 15: Additional FTP and SFTP nodes

4.4.9 Router App Node-RED / GPSd



This router app requires the GPS router app to be installed on the router.

Adding this router app 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>

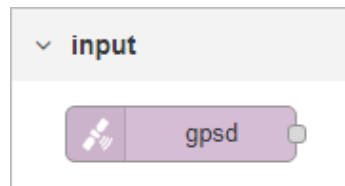


Figure 16: Additional GPSd node in input category

4.4.10 Router App Node-RED / gzip

Adding this router app 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>

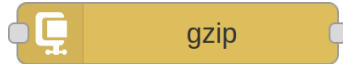


Figure 17: Additional gzip node

4.4.11 Router App Node-RED / KNX

Adding this router app 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>

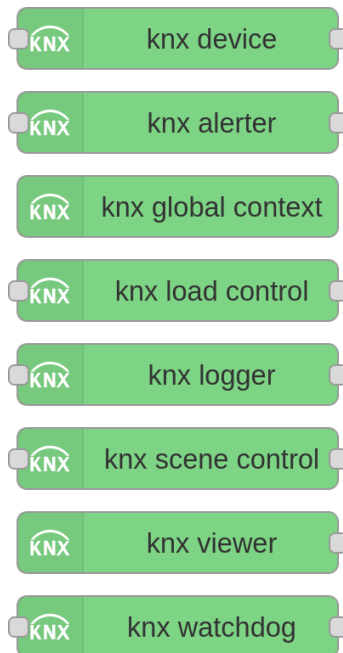


Figure 18: Additional knx nodes

4.4.12 Router App Node-RED / LwM2M

This node offers *OMA* LwM2M client functionalities and allows you to create your *OMA* LwM2M client applications on top of Node-RED. See more information in the official documentation: <https://flows.nodered.org/node/node-red-contrib-lwm2m>

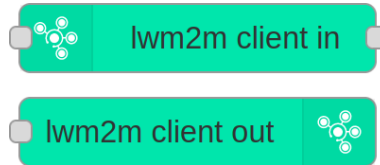


Figure 19: LwM2M nodes

4.4.13 Router App Node-RED / Modbus

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

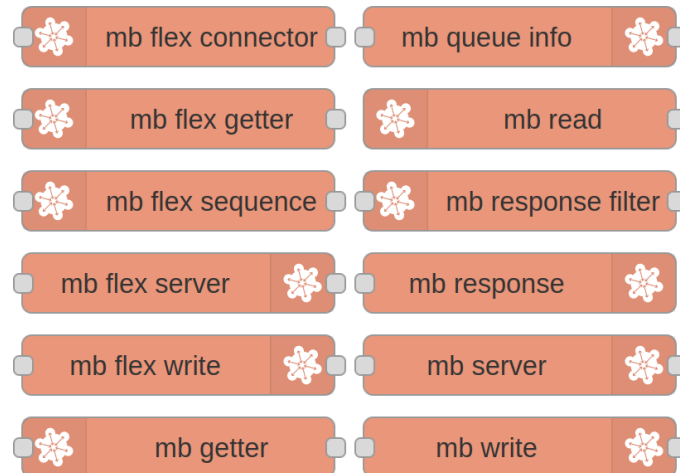


Figure 20: Modbus nodes in

4.4.14 Router App Node-RED / OPC UA

OPC Unified Architecture (OPC UA) is a machine to machine communication protocol for industrial automation. Adding this router app 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>

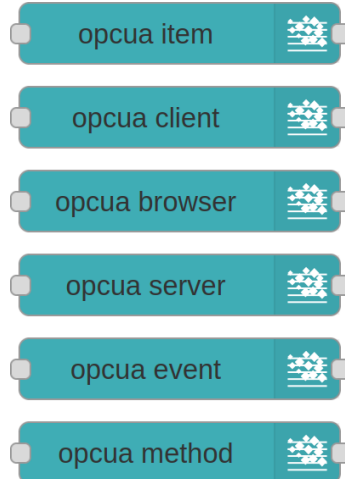


Figure 21: Additional OPC UA nodes

4.4.15 Router App 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>

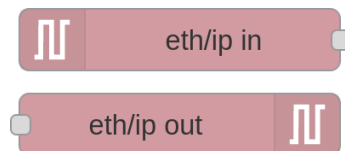


Figure 22: PLC EtherNet/IP nodes

4.4.16 Router App 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>

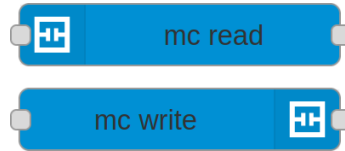


Figure 23: PLC Melsec nodes

4.4.17 Router App Node-RED / Splunk HEC

Splunk HEC is a HTTP event collector. Adding this router app 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>

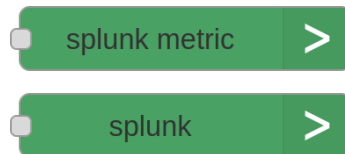


Figure 24: Additional Splunk event collector nodes

5. Advanced topics

5.1 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 history length depends on the Log level set).

/var/data/node-red/flows.json - File contains definition of your flows. Path part is configurable, see [3.2](#)

/var/data/node-red/flows_cred.json - File contains credentials (confidential data) of your flows. The file is encrypted. Path part is configurable, see [3.2](#)

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

5.2 How to distribute your own flows to the routers

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

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

For example, you could directly copy the files with flow definitions over SFTP. The path to those files can be set in the Configuration (Flow files path item). Default path is `/var/data/node-red`. The main file is called *flows.json*. After the file is uploaded, the Node-RED needs to be restarted.

If your flows contain 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 were created is needed. The secret is generated during the first launch of Node-RED, or you could choose your own. This option is not accessible via the GUI. The secret can be physically found in the `/opt/nodered/etc/setting` file in the `MOD_NODERED_CREDSECRET` item. If you are running a router installation in bulk, you are probably loading the prepared configuration. The needed secret for Node-RED will be easily distributed if the configuration is prepared on the router where the flows are developed.

The best way to distribute flows to multiple devices is to create your own Router App (previously Router app). This way you can install and update your Node-RED application exactly 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 flow definitions to `flows` (see above). The `etc` should contain the text files *name* and *version* with the name and version of your application. Compress everything using the tar command:

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

More information about creating Router Apps/User Modules can be found on our portal <https://icr.advantech.com/devzone/developing-user-modules> or in the [Programming of Router Apps Application Note](#)

Your own Router App can be used for distributing your own nodes, too (See chapter 5.5 for details on how to prepare your own node). The finished node can then be compressed with the other files to a .tgz. User needs to create a symbolic link to your node on the router, for example `/usr/lib/node_modules/node-red/nodes/mynode` → `/opt/myapp/nodes/mynode`. User should use the script `/opt/myapp/etc/install` for creating symbolic link and `/opt/myapp/etc/uninstall` for deleting. Make sure these scripts are executable.

Example of the install script:

```
#!/bin/sh

if [ ! -L /usr/lib/node_modules/node-red/nodes/mynode -a -L
/usr/lib/node_modules/node-red ]; then
ln -s /opt/myapp/nodes/mynode /usr/lib/node_modules/node-red/nodes/mynode
fi

# If it isn't firmware update and it isn't called from Node-RED install script
if [ -e /etc/settings -a "$1" != "norestart" ]; then
/opt/nodered/etc/init restart
fi
```

and the uninstall script:

```
#!/bin/sh

rm -f /usr/lib/node_modules/node-red/nodes/mynode

/opt/nodered/etc/init restart
```

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

5.3 Persistent variable storage

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

5.4 Node "router"

Our node *router* is added to the global context. See the Node.js application note for more information about this node. Users can use this node to obtain router data, like a serial number. The example below shows how to get the serial number to the message payload.

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

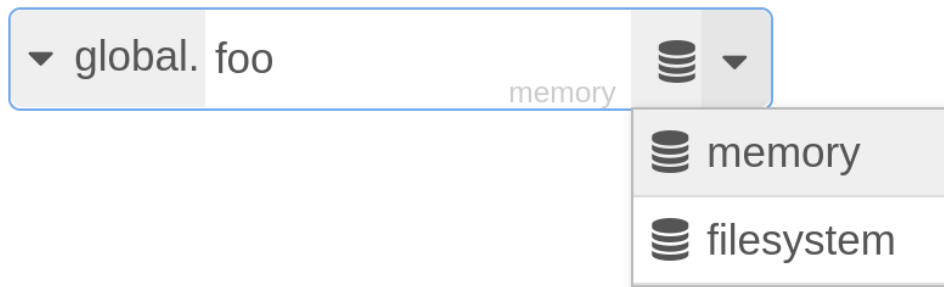


Figure 25: Persistent storage

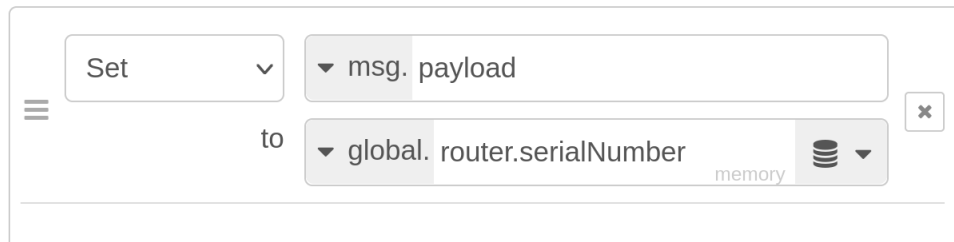


Figure 26: Router node

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

5.5 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 to build and install a node is using the npm command. However, our router is an embedded device with limited resources, and some nodes require a complex building environment and high performance because of languages other than JavaScript. Advantech provides the npm tool as a standalone Router App. You can install it on the router, but its usage is limited.

Another way is to prepare a node on a PC with Linux and then copy it to the router. The process is easy, but in some cases, you need to have a PC with the same major version of Node.js as is on the router.



During installation, you can encounter two kinds of nodes: those written in JavaScript only and those written in JavaScript and other language(s). Some node authors provide prebuilt binaries so you can install later type quickly just as easily as nodes written only in JavaScript. But when you see a message:

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

in npm output, you need to compile the binaries, and chapter 4.5.2 is only one option for you.

5.5.1 Preparing JavaScript Only Nodes on PC

These nodes need an 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 the 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 a better idea is to place it in `/opt` folder and create a link from the mentioned folders to avoid overriding while upgrading firmware. Packing the result node to `.tgz` file as the common Router App is also possible.

5.5.2 Preparing Nodes Using Other Languages on PC

Some nodes have 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 the same toolchains we use for building the router's router apps. See Environment setup chapter in [Developing Router Apps Guideline](#)¹ to learn how to install it.

When you have installed all prerequisites, invoke the following commands for the 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 the V4 platform, invoke the following:

```
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
```

¹<https://sendfiles.advantech-bb.cz/download.php?id=42&token=W0TUuyz5bd4SIg60b2frqH9dCI9pAtn1>

where NODE is again a real node name and continues 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.

5.5.3 Installing nodes directly on router



Applications installed in the `/usr/lib` directory will be deleted upon updating the router firmware.



Be aware that if the `/usr/lib` folder fills the entire storage, it may render the router non-functional, and a factory reset might not resolve the issue. Ensure not to install too many applications in this folder.

You can manage nodes with the installed npm Router App in the router. A common command is:

```
npm install -g <node_name>
```

For example:

```
npm install -g node-red-contrib-combine
```

It will install node to the main node modules folder `/usr/lib/node_modules` (see chapter 5.1). Node-RED can find palette nodes in this location. It is possible to install with npm to a different location but note that other folders are symlinks to `/opt`, and they are overwritten on updates.

```
# npm install -g node-red-contrib-combine

changed 1 package, and audited 2 packages in 6s

found 0 vulnerabilities
#
```

Figure 27: Installation with npm tool on the router

Note that the npm tool requires an internet connection.



Note that we only support nodes provided by Advantech as the Router Apps, and there will be no support for the Node.js/Node-RED nodes installed by the npm tool.

6. Flow Examples

The 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 in the example. Import the code via top right menu in the Node-RED UI, by choosing *Import*, *Clipboard*.

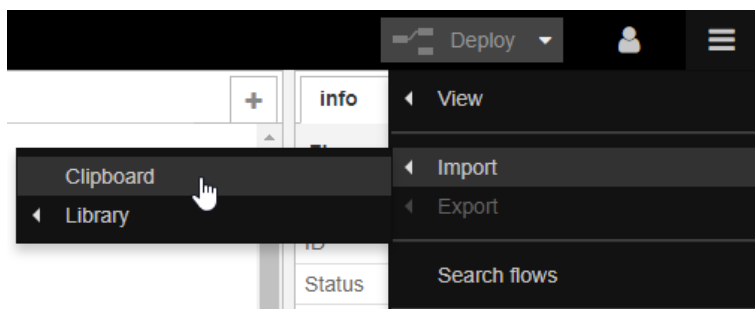


Figure 28: Top right menu – import from clipboard

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

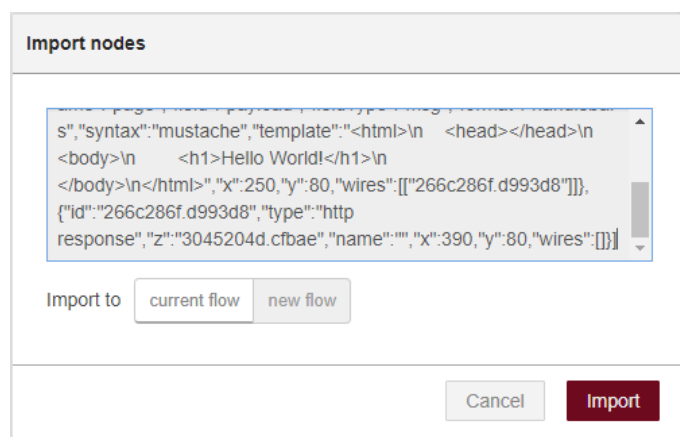


Figure 29: Import nodes via JSON

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

6.1 Example 1: User LED Turn On and Off

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

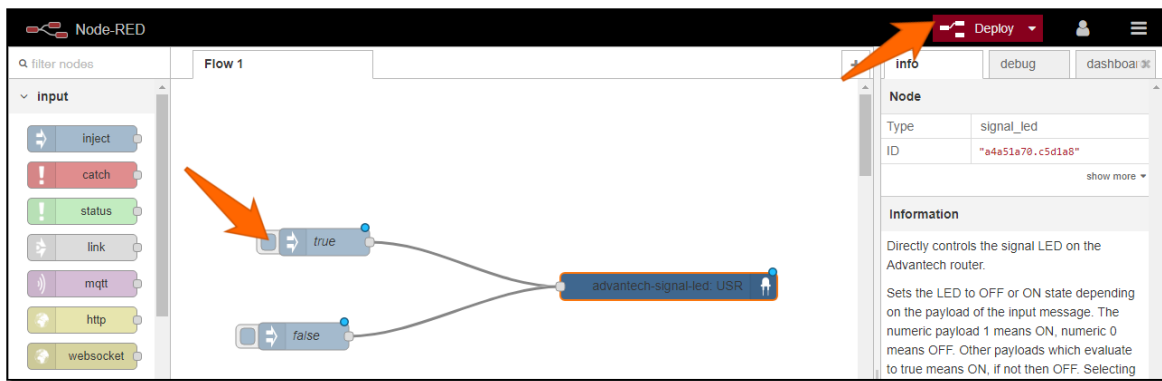


Figure 30: Example 1 – user LED flow

Click *Deploy* red button in the upper right corner. Now the flow is running. Click on the small rounded button in front of the 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 31: Example 1 – user LED turned on

6.2 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 in the following Figure – an HTTP endpoint that responds to GET requests with some static content (such as an HTML page) is created.

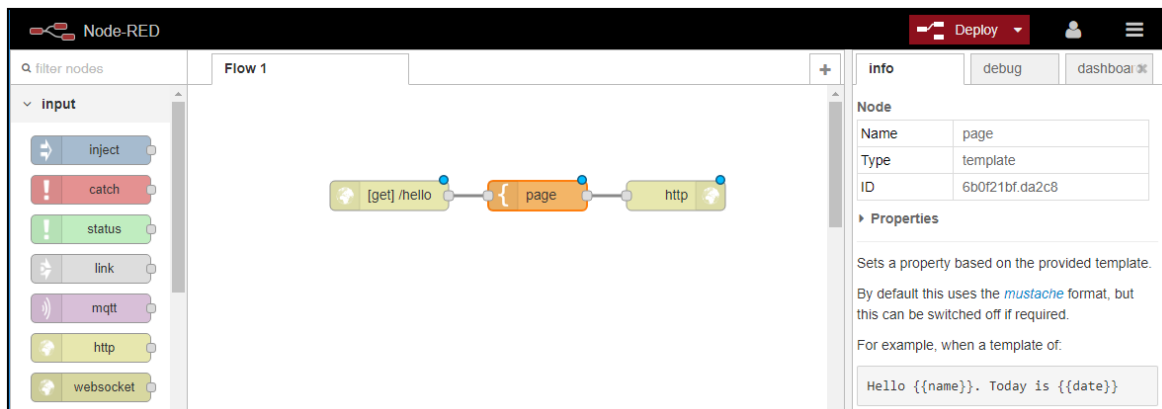


Figure 32: Example 2 – imported nodes to deploy

The nodes "HTTP in" (where URL can be defined), "template" (where HTML page can be defined), and "HTTP response" are used. Next, 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-address-of-the-router>:<configured-port>/hello>), you will see the defined HTML page as the response:

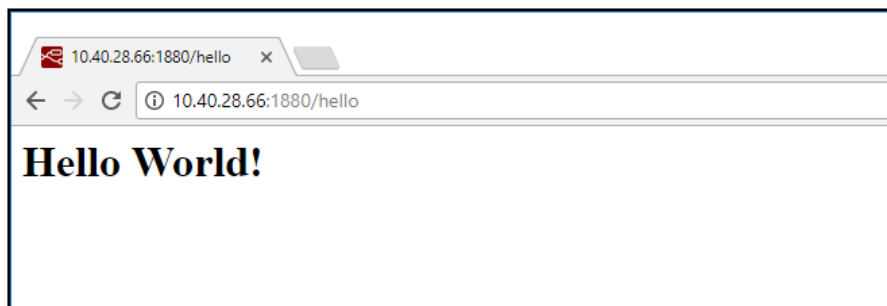
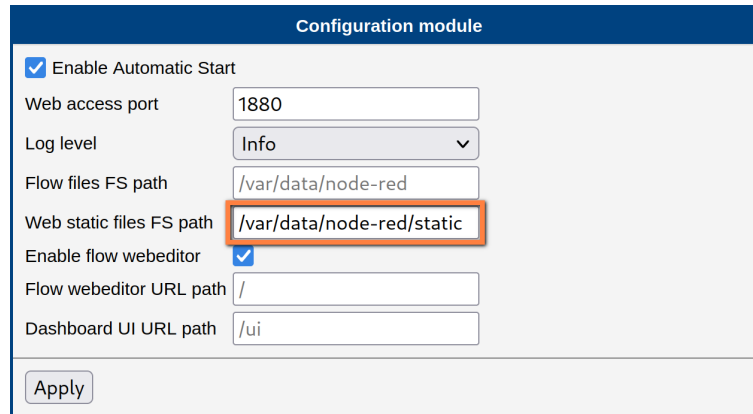


Figure 33: Example 2 – HTTP response

6.3 Example 3: Ping Statistics

This is an example of how to use the *Web static file FS path* option in Node-RED configuration (explained in Chapter 5.1). It pings on the `www.advantech.com` domain, stores the results in a file, shows them in a chart, and enables the download of the collected data file.

First, add the path `/var/data/node-red/static` to the Node-RED configuration as seen in the Figure:



The screenshot shows the 'Configuration module' interface. It has a title bar 'Configuration module' and a list of settings. The 'Web static files FS path' field is highlighted with an orange border. The settings are as follows:

Setting	Value
Enable Automatic Start	<input checked="" type="checkbox"/>
Web access port	1880
Log level	Info
Flow files FS path	/var/data/node-red
Web static files FS path	/var/data/node-red/static
Enable flow webeditor	<input checked="" type="checkbox"/>
Flow webeditor URL path	/
Dashboard UI URL path	/ui

An 'Apply' button is located at the bottom left of the configuration area.

Figure 34: Example 3 – Web static file FS path configuration

Next, create the folder structure, so the given path exists. You can log in to the router (via SSH) and run this command,



```
mkdir /var/data/node-red/static
```

or connect to the router via FTP and create folder `/var/data/node-red/static`. Then download the button image from:

https://upload.wikimedia.org/wikipedia/commons/8/8d/Download_alt_font_awesome.svg

and upload it to the router as `/var/data/node-red/static/download.svg` file. Or download it directly in the router, e.g. with command:



```
curl https://upload.wikimedia.org/wikipedia/commons/8/8d/Download_alt_font_awesome.svg  
-o /var/data/node-red/static/download.svg
```

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

```
[{"id":"6a06b9c496562116","type":"tab","label":"Flow 1","disabled":false,"info":"","env":[]},
{"id":"2c84ad78.fa6f92","type":"ui_chart","z":"6a06b9c496562116","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":"10","removeOlderPoints":"","removeOlderUnit":"60","cutout":0,
"useOneColor":false,"useUTC":true,"colors":
["#ff7f0e","#aec7e8","#1f77b4","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],
"outputs":1,"useDifferentColor":
false,"className":"","x":690,"y":220,"wires":[[]]},{
"id":"d9e078ec.f791b","type":"file","z":
"6a06b9c496562116","name":"","filename":"/var/data/node-red/static/pings.dat","filenameType":"str",
"appendNewline":true,"createDir":true,"overwriteFile":
false,"x":500,"y":160,"wires":[[]]},{
"id":"21f9f5c8.52674a","type":"ping","z":"6a06b9c496562116",
"mode":null,"name":"","
"host":"www.advantech.com","timer":"10","inputs":0,"x":230,"y":160,"wires":[["d9e078ec.f791b"]]},
{"id":"20279b4717c1ddd3",
"type":"ui_template","z":"6a06b9c496562116","group":"ae1455d7.c2f2c8","name":"Download button",
"order":1,"width":"2","height":"1",
"format":"<a href=\"/pings.dat\"><md-button> <img src=\"/download.svg\" width=\"20\"
height=\"20\"/><n</md-button><n</a>>",
"storeOutMessages":true,"fwdInMessages":true,"resendOnRefresh":false,
"templateScope":"local","className":"","x":650,"y":280,"wires":[[]]},{
"id":"a81ff01f8dd4f3f5",
"type":"watch","z":"6a06b9c496562116","name":"Watch File","files":"/var/data/node-red/static/pings.dat",
"recursive":"","x":200,"y":220,"wires":
[[["7d8a9b63f5b77eea"]]],{
"id":"7d8a9b63f5b77eea","type":"file_in","z":"6a06b9c496562116","name":
"Read File","filename":"filename","filenameType":"msg","format":"utf8","chunk":false,
"sendError":false,"encoding":"none","allProps":false,
"x":360,"y":220,"wires":[["56294e2ca54d88b8"]]],{
"id":"56294e2ca54d88b8","type":"change","z":
"6a06b9c496562116","name":
"Get Last Line","rules":[{"t":"set","p":"payload","pt":"msg","to":"$split(msg.payload,
\\\"\\n\\\")[-2]","tot":"jsonata"}],
"action":"","property":"","from":"","to":"","reg":false,"x":530,"y":220,"wires":
[[["2c84ad78.fa6f92"]]],{
"id":"ae1455d7.c2f2c8","type":"ui_group","name":"Networking monitor",
"tab":"8912138.b569e7","order":1,"disp":true,"width":"8"},
{"id":"8912138.b569e7","type":"ui_tab","name":"Advantech demo","icon":"dashboard"}]
```



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

The following flow of nodes will appear:

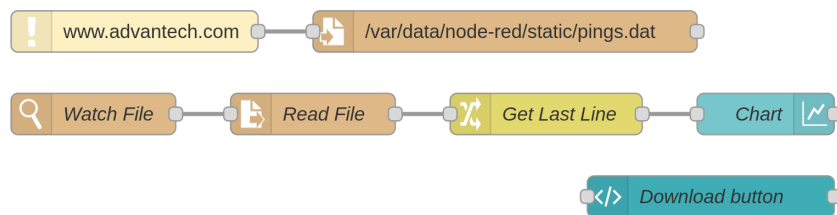


Figure 35: Example 3 – ping statistics flow

Deploy the flow with a red button in the upper right corner. Now you can access the ping statistics at this address: https://<router_ip>:<port_number>/<dashboard_path>

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/data/node-red/static/pings.dat` file. *Web static files FS path* is used when writing – saving the ping results to the data file, reading – viewing the statistics page (button image), and downloading the data file.

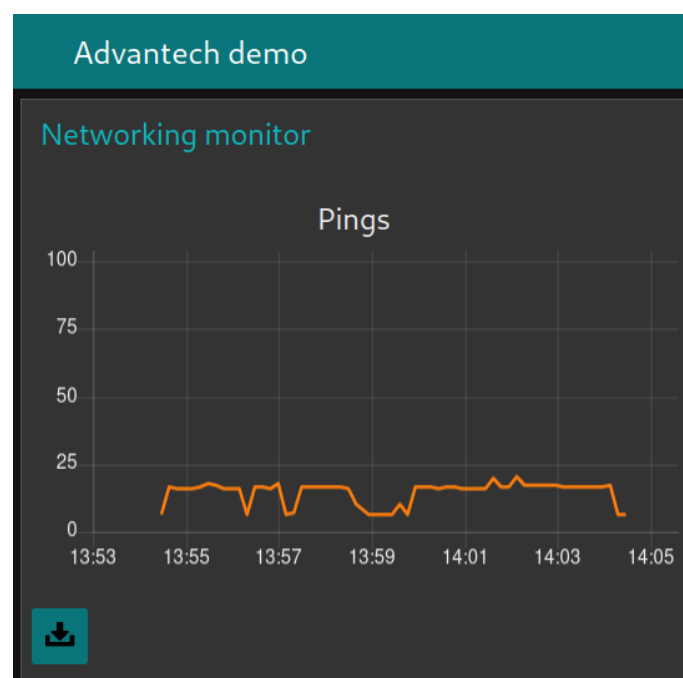


Figure 36: Example 3 – ping statistics chart and download

6.4 Node-RED Bluetooth Examples

6.4.1 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 37: 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 38: Example 4 – Status

We offer a specialized node for the Ruuvi Tag sensors that take care of data decoding. So you can 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":""},{id":"be2fd8f3.23c9e8","type":"BLE scanner","z":"2945eb57.8f6a34","name":"","services":"","servicesType":"str","autostart":true,"continuous":true,"duplicates":true,"timeout":"","x":250,"y":160,"wires":[["22af1566.e8bde2"]]},{"id":"22af1566.e8bde2","type":"ruuvitag","z":"2945eb57.8f6a34","name":"","x":420,"y":160,"wires":[["3693f0e2.1ef39"]]},{"id":"3693f0e2.1ef39","type":"switch","z":"2945eb57.8f6a34","name":"","property":"peripheral","propertyType":"msg","rules":[{"t":"eq","v":"e92232c53f2b","vt":"str"},{"t":"eq","v":"e92232c54a78","vt":"str"},{"t":"eq","v":"e92232c5412c","vt":"str"}],"checkall":"true","repair":false,"outputs":3,"x":560,"y":160,"wires":[["358dabb5.a5d1ac","8464c887.e37e4"],["6619f6f5.3502c","5bf803f2.6134a4"],["7df2dec6.c0c068","2ca03fd5.6eb208"]]},{"id":"358dabb5.a5d1ac","type":"ui_gauge","z":"2945eb57.8f6a34","name":"","group":"14d04008.aa63f","order":0,"width":0,"height":0,"gtype":"gage","title":"Temperature","label":"units","format":"{{msg.payload.temperature}}","min":10,"max":40,"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":1,"width":0,"height":0,"gtype":"gage","title":"Humidity","label":"units","format":"{{msg.payload.humidity}}","min":0,"max":50,"colors":["#ffffff","#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":0,"width":0,"height":0,"gtype":"gage","title":"Temperature","label":"units","format":"{{msg.payload.temperature}}","min":10,"max":40,"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":1,"width":0,"height":0,"gtype":"gage","title":"Humidity","label":"units","format":"{{msg.payload.humidity}}","min":0,"max":50,"colors":["#ffffff","#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":"gage","title":"Temperature","label":"units","format":"{{msg.payload.temperature}}","min":10,"max":40,"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":"gage","title":"Humidity","label":"units","format":"{{msg.payload.humidity}}","min":0,"max":50,"colors":["#ffffff","#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.

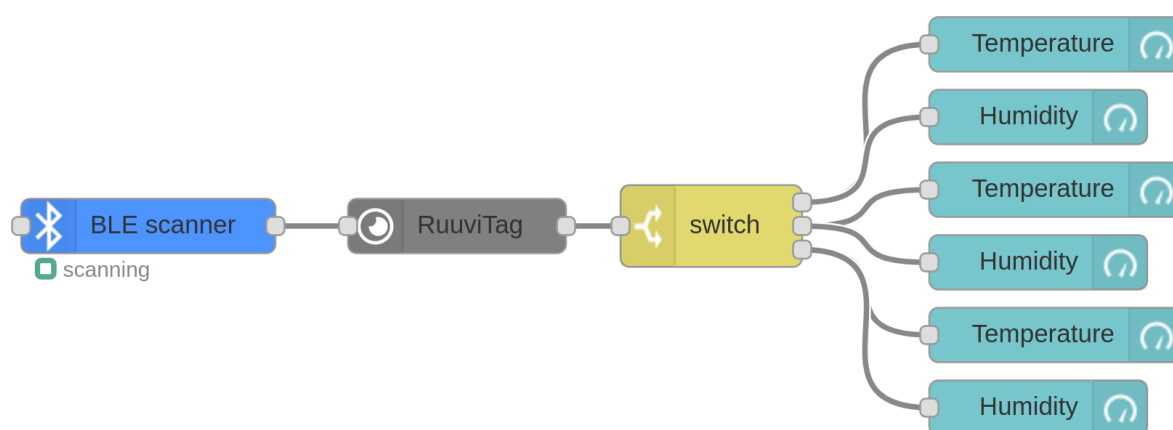


Figure 39: Example 4 – Bluetooth router app

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

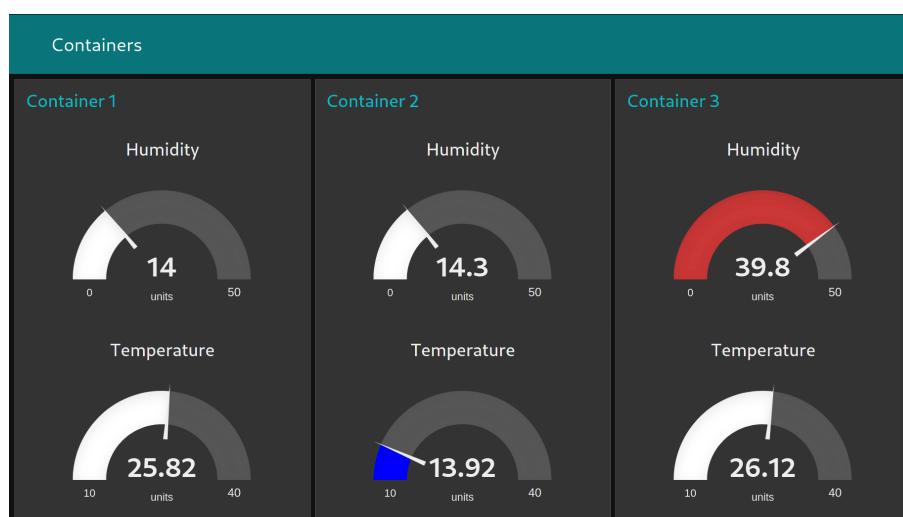


Figure 40: Example 4 – Presentation of the environment conditions with Dashboard UI

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 of how to use Bluetooth with Advantech routers.

6.4.2 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 41: 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 42: 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": "ac17368.d75d", "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", "d9bdfcb.738508"}]}, {"id": "b773f45.1638d88", "type": "BLE in", "z": "30182cd3.1eaddc", "topic": "read", "characteristic": "00001a020000100800000805f9b34fb", "name": "", "x": 370, "y": 40, "wires": [{"f4e7939bd.3e098"}]}, {"id": "ded8cf21.e47b18", "type": "BLE in", "z": "30182cd3.1eaddc", "topic": "read", "characteristic": "00001a010000100800000805f9b34fb", "name": "", "x": 490, "y": 160, "wires": [{"f1fe8af43.36f789"}]}, {"id": "eed2cad7.3cb43", "type": "BLE out", "z": "30182cd3.1eaddc", "characteristic": "00001a000000100800000805f9b34fb", "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}\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", "servicesType": "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", "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": [{"1a4f3520.7f10c3"}]}, {"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": "d9bdfcb.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:

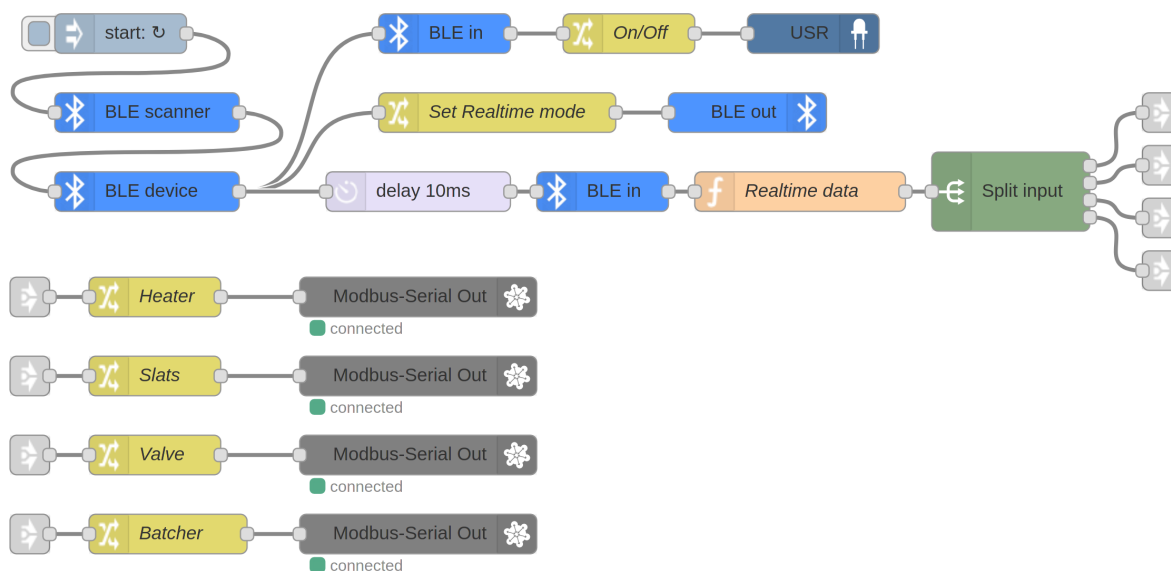


Figure 43: Example 5 – Nodes

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 only counts on one sensor, so there are no address checks and duplicity allowed. Real-time data are read with a slight delay so that the sensor can 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 they can clarify some other aspects of using Bluetooth with Advantech routers.

6.5 Node-RED Modbus Examples



Please note that you will need the Node-RED Modbus Node Module.

The Modbus Server is only helper part of bellow examples. It is possible to install its node to other router or to use an different Modbus server (e.g. a simulator on PC). It is only necessary to fit communication configuration on the both client in the example and the server.

The required configuration before attempting examples 5.5.1-5.5.4. Adds the Modbus Server node to the flow and configures the Modbus Client, which the Modbus nodes will use to connect.



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

[illegible]

The following node will appear:

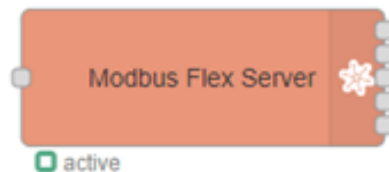


Figure 44: Modbus Flex Server

6.5.1 Example 6: Writing to the User-Defined Modbus Server

An example of a flow to write to the TCP Modbus Server. This example writes to the holding registers.

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



```
[{"id":"15e99f6dc3820308","type":"hw_monitor","z":"5e1b0393890554df","param":"voltage","when":"0","low":"","high":"","name":"","x":670,"y":60,"wires":[["fe35a226cfc35727"]]},{"id":"fe35a226cfc35727","type":"change","z":"5e1b0393890554df","name":"voltage * 1000","rules":[{"t":"set","p":"payload","pt":"msg","to":"$number(payload)*1000","tot":"jsonata"}],"action":"","property":"","from":"","to":"","reg":false,"x":840,"y":60,"wires":[["5c6e5d717d6c4e1d"]]},{"id":"5c6e5d717d6c4e1d","type":"modbus-write","z":"5e1b0393890554df","name":"Write voltage","showStatusActivities":false,"showErrors":false,"unitid":"1","dataType":"HoldingRegister","adr":"0","quantity":"1","server":"6e4a5e2fc9f3826f","emptyMsgOnFail":false,"keepMsgProperties":false,"x":1030,"y":60,"wires":[[],[]]}
```

The following flow of nodes will appear:



Figure 45: Example 6 – Write Voltage Nodes

Writing is triggered every time the voltage changes in the device's power supply. Since the registers can store numbers between 0 – 65535 and the voltages on the device can range from +9 to +36 V DC with three-digit precision, the number can be multiplied by 1000 and then stored.

First, voltage is written to the address '0' using the Modbus-write node. Node configuration uses the Modbus function code 6, which writes a single value to a specified address in the holding registers. After the voltage changes again, the old value in the register rewrites.

6.5.2 Example 7: Reading from the User-Defined Modbus Server

An example of a flow to read from the TCP Modbus Server. This example reads the value which we wrote in 6.5.1.

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

```
[{"id":"fa93b02f75982874","type":"inject","z":"5e1b0393890554df","name":"Read last value","props":[{"p":"payload"}],"repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"","payloadType":"str","x":720,"y":120,"wires":[["13a16208e9af985c"]]}, {"id":"13a16208e9af985c","type":"modbus-getter","z":"5e1b0393890554df","name":"Read voltage","showStatusActivities":false,"showErrors":false,"logIOActivities":false,"unitid":"1","dataType":"HoldingRegister","adr":"0","quantity":"1","server":"6e4a5e2fc9f3826f","useIOFile":false,"ioFile":"","useIOForPayload":false,"emptyMsgOnFail":false,"keepMsgProperties":false,"x":890,"y":120,"wires":[["d5e2d57441cccbf1"]]}, {"id":"d5e2d57441cccbf1","type":"change","z":"5e1b0393890554df","name":"voltage / 1000","rules":[{"t":"set","p":"payload","pt":"msg","to":"payload[0]/1000","tot":"jsonata"}],"action":"","property":"","from":"","to":"","reg":false,"x":1080,"y":120,"wires":[["17ea3fce6db98c28"]]}, {"id":"17ea3fce6db98c28","type":"debug","z":"5e1b0393890554df","name":"Output","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload","targetType":"msg","statusVal":"","statusType":"auto","x":1230,"y":120,"wires":[]}]
```

The following flow of nodes will appear:



Figure 46: Example 7 – Read Voltage Nodes

Press the button to read the value in the Modbus holding registers. Reading uses the Modbus-read node. Node configuration uses the Modbus function code 3, which reads a specified number of holding registers from the starting address. In this case, the configuration reads one register from address '0'. Since the stored value in the register was multiplied by 1000, we divided the number before displaying it because of three-digit precision.

6.5.3 Example 8: Writing a sequence to the User-Defined Modbus Server

An example of a flow to write a sequence of values. This example demonstrates how to make modifications to 6.5.1 to write values to the registers in a different way.

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



```
[{"id":"b8a5f7c30afe0013","type":"hw_monitor","z":"5e1b0393890554df","param":"voltage","when":"0","low":"","high":"","name":"","x":350,"y":280,"wires":[["ff3d51e58d027413"]]}, {"id":"a15b41cd4d503053","type":"modbus-write","z":"5e1b0393890554df","name":"Write voltage","showStatusActivities":false,"showErrors":false,"unitid":"1","dataType":"MHoldingRegisters","adr":"1","quantity":"2","server":"6e4a5e2fc9f3826f","emptyMsgOnFail":false,"keepMsgProperties":false,"x":730,"y":280,"wires":[["2e4d4cc32fdeca64"],[]]}, {"id":"ff3d51e58d027413","type":"function","z":"5e1b0393890554df","name":"float to dec array","func":"// example: \n// input: 12.095 \n// output: [16705, 34079]\n\nvar float = msg.payload;\nvar buffer = new ArrayBuffer(4);\nvar floatView = new Float32Array(buffer);\nvar uint16View = new Uint16Array(buffer);\n\nfloatView[0] = float;\nmsg.payload = [uint16View[1], uint16View[0]];\nreturn msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":530,"y":280,"wires":[["a15b41cd4d503053"]]}]
```

The following flow of nodes will appear:



Figure 47: Example 8 – Writing Sequence Nodes

This example focuses on the transformation of measured values. This time, instead of multiplying the calculated value, we store it in a Single-precision floating-point format. This way, keeping negative values is possible. Another way to accurately store negative numbers is by defining an offset, adding it before storing the value, and subtracting it before displaying it. The best practice is to hold the specified offset in another register and read it in computing.

How does the "float to dec array" function node work? Let's say the float value '12.078' flows in. This node transforms the value into an array of length 2 with dec values ranging from 0 to 65535. This way, the registers will use 32 bits of registers for storage.

Values are written to the addresses '0' and '1' using the Modbus-write node. Node configuration uses the Modbus function code 16, which presets multiple registers from a specified address and fixed length in the holding registers.

6.5.4 Example 9: Reading a sequence from the User-Defined Modbus Server



Please note that you need the Node-RED module Dashboard installed for this example to work correctly.

Example of a flow to read a sequence of values. This example reads the values written to the registers in 6.5.3.



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

```
{
  "id": "001d41f9baabeb9c",
  "type": "inject",
  "z": "5e1b0393890554df",
  "name": "Read last float value",
  "props": [
    {
      "p": "payload"
    }
  ],
  "repeat": "1",
  "crontab": "",
  "once": true,
  "onceDelay": "1",
  "topic": "",
  "payload": "",
  "payloadType": "str",
  "x": 440,
  "y": 480,
  "wires": [
    [
      "753cc8d5c5bec8a1"
    ]
  ],
  "id": "73c3804fc367a0b8",
  "type": "toFloat",
  "z": "5e1b0393890554df",
  "name": "",
  "toFixed": "3",
  "x": 810,
  "y": 560,
  "wires": [
    [
      "775095c1191539c3"
    ]
  ],
  "id": "2136931f8d721d92",
  "type": "ui_chart",
  "z": "5e1b0393890554df",
  "name": "",
  "group": "9efd6e054f0a3a18",
  "order": 0,
  "width": 0,
  "height": 0,
  "label": "Voltage",
  "chartType": "line",
  "legend": "false",
  "xformat": "HH:mm:ss",
  "interpolate": "linear",
  "nodata": "",
  "dot": true,
  "ymin": "",
  "ymax": "",
  "removeOlder": "2",
  "removeOlderPoints": "",
  "removeOlderUnit": "60",
  "cutout": 0,
  "useOneColor": false,
  "useUTC": true,
  "colors": [
    "#1f77b4",
    "#aec7e8",
    "#ff7f0e",
    "#2ca02c",
    "#98df8a",
    "#d62728",
    "#ff9896",
    "#9467bd",
    "#c5b0d5"
  ],
  "outputs": 1,
  "useDifferentColor": false,
  "className": "",
  "x": 1080,
  "y": 560,
  "wires": [
    [
      "753cc8d5c5bec8a1"
    ]
  ],
  "id": "753cc8d5c5bec8a1",
  "type": "modbus-getter",
  "z": "5e1b0393890554df",
  "name": "Read voltage float",
  "showStatusActivities": false,
  "showErrors": false,
  "logIOActivities": false,
  "unitid": "1",
  "dataType": "HoldingRegister",
  "adr": "1",
  "quantity": "2",
  "server": "6e4a5e2fc9f3826f",
  "useIOFile": false,
  "ioFile": "",
  "useIOForPayload": false,
  "emptyMsgOnFail": false,
  "keepMsgProperties": false,
  "x": 670,
  "y": 480,
  "wires": [
    [
      "4c6c0d9df75ead1"
    ],
    [
      "81a3ee19a532262d"
    ]
  ],
  "id": "81a3ee19a532262d",
  "type": "change",
  "z": "5e1b0393890554df",
  "name": "dec to bin",
  "rules": [
    {
      "t": "set",
      "p": "payload",
      "pt": "msg",
      "to": "$pad($formatBase($number(payload), 2), -16, '0')\t",
      "tot": "jsonata",
      "action": "",
      "property": "",
      "from": "",
      "to": "",
      "reg": false,
      "x": 540,
      "y": 560,
      "wires": [
        [
          "bed734d91af051d9"
        ]
      ],
      "id": "4c6c0d9df75ead1",
      "type": "split",
      "z": "5e1b0393890554df",
      "name": "split array",
      "spl": "\n",
      "splType": "str",
      "arraySpl": 1,
      "arraySplType": "len",
      "stream": false,
      "addname": "",
      "x": 400,
      "y": 560,
      "wires": [
        [
          "81a3ee19a532262d"
        ]
      ],
      "id": "bed734d91af051d9",
      "type": "join",
      "z": "5e1b0393890554df",
      "name": "join bits",
      "mode": "custom",
      "build": "string",
      "property": "payload",
      "propertyType": "msg",
      "key": "topic",
      "joiner": "",
      "joinerType": "str",
      "accumulate": false,
      "timeout": "",
      "count": "2",
      "reduceRight": false,
      "reduceExp": "",
      "reduceInit": "",
      "reduceInitType": "",
      "reduceFixup": "",
      "x": 680,
      "y": 560,
      "wires": [
        [
          "73c3804fc367a0b8"
        ]
      ],
      "id": "775095c1191539c3",
      "type": "switch",
      "z": "5e1b0393890554df",
      "name": "> 0",
      "property": "payload",
      "propertyType": "msg",
      "rules": [
        {
          "t": "gt",
          "v": "0",
          "vt": "num"
        }
      ],
      "checkall": true,
      "repair": false,
      "outputs": 1,
      "x": 930,
      "y": 560,
      "wires": [
        [
          "2136931f8d721d92"
        ]
      ],
      "id": "eae94d7a961ee00d",
      "type": "inject",
      "z": "5e1b0393890554df",
      "name": "reset chart",
      "props": [
        {
          "p": "payload"
        }
      ],
      "repeat": "",
      "crontab": "",
      "once": false,
      "onceDelay": 0.1,
      "topic": "",
      "payload": [
        []
      ],
      "payloadType": "json",
      "x": 920,
      "y": 520,
      "wires": [
        [
          "2136931f8d721d92"
        ]
      ],
      "id": "9efd6e054f0a3a18",
      "type": "ui_group",
      "name": "Voltage monitor",
      "tab": "48418b79.0f5834",
      "order": 1,
      "disp": true,
      "width": "6",
      "collapse": false,
      "className": ""
    }
  ]
}
```

The following flow of nodes will appear:

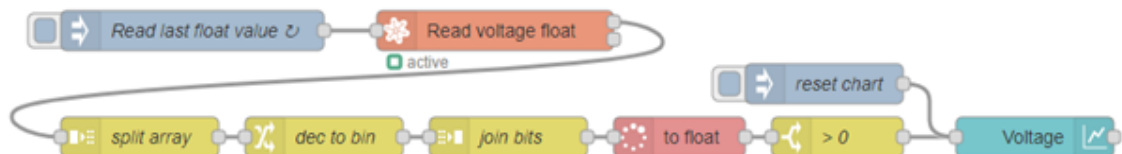


Figure 48: Example 9 – Reading Sequence Nodes

This example focuses on decoding the Single-precision floating-point format sequence in the Modbus Server, written in the last example. The Modbus-read node performs the reading. Node configuration uses the Modbus function code 3, which reads a specified number of holding registers from the starting address. In this case, the configuration reads two registers from address '0'. The node returns an array of two values, splits them, converts the decimal values to binary, and then joins the two split values. Then, thanks to the "to float" node, values are converted from binary to float.

Although this example could end with a debug node printing out the value, we expanded it to include an actual use case by adding a chart that displays the voltage over the last 5 minutes. Achieving this requires temporarily storing the value. We configured the "inject" node to inject the flow every second. After the "to float" node, we filter out invalid values using the "switch" node. Adding values to the chart is as easy as connecting them to the output.

The chart can be accessed at address:

<https://<ip-adress-of-the-router>:<configured-port>/ui>

The chart should look like this:

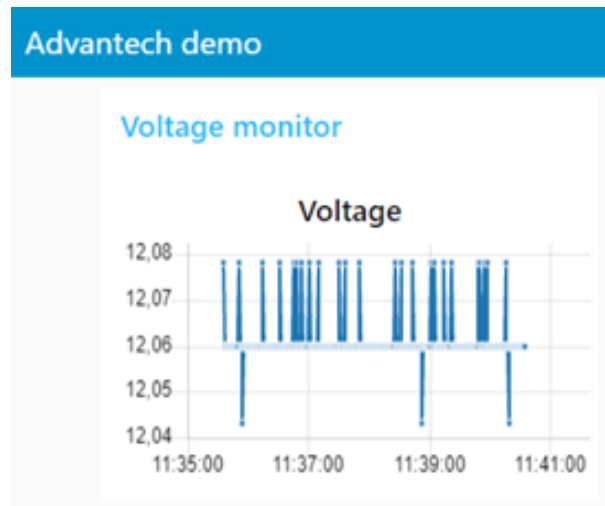


Figure 49: Example 9 – Voltage Chart

6.5.5 Example 10: Reading Coil status from a Modbus RTU Relay using the Modbus Protocol

The primary purpose of the Modbus protocol is to facilitate communication in industrial systems. This example demonstrates reading from a Modbus RTU Relay — the relay links to the `/dev/ttyS1` serial port.

The relay used in this example: <https://www.waveshare.com/modbus-rtu-relay.htm>

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



```
[{"id":"557beabce6821e78","type":"inject","z":"5e1b0393890554df","name":"Read Coil statuses","props":[{"p":"payload"}],"repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"","payloadType":"date","x":470,"y":1220,"wires":[["909c4b126be4dfbb"]]}, {"id":"909c4b126be4dfbb","type":"modbus-getter","z":"5e1b0393890554df","name":"Relay read Coils","showStatusActivities":true,"showErrors":true,"logIOActivities":false,"unitid":"1","dataType":"Coil","adr":"0","quantity":"8","server":"6e1c00798d661717","useIOFile":false,"ioFile":"","useIOForPayload":false,"emptyMsgOnFail":false,"keepMsgProperties":false,"x":680,"y":1220,"wires":[["e4e2479f8a53143d"]]}, {"id":"e4e2479f8a53143d","type":"modbus-response","z":"5e1b0393890554df","name":"","registerShowMax":20,"x":910,"y":1220,"wires":[["6e1c00798d661717"]]}, {"id":"6e1c00798d661717","type":"modbus-client","name":"RTU","clienttype":"simpler","bufferCommands":false,"stateLogEnabled":false,"queueLogEnabled":false,"tcpHost":"127.0.0.1","tcpPort":"1188","tcpType":"DEFAULT","serialPort":"/dev/ttyS1","serialType":"RTU-BUFFERD","serialBaudrate":"9600","serialDatabits":"8","serialStopbits":"1","serialParity":"none","serialConnectionDelay":"100","serialAsciiResponseStartDelimiter":"0x3A","unit_id":"1","commandDelay":"1","clientTimeout":"1000","reconnectOnTimeout":true,"reconnectTimeout":"2000","parallelUnitIdsAllowed":true}]
```

The following flow of nodes will appear:



Figure 50: Example 10 – Reading Coil Nodes

The Modbus Response node will report the Coil status upon pressing the button. The Modbus-read node performs the reading. Configuration of the node is to use Modbus function code 1, which reads a specified number of coil statuses from the starting address. In this case, the configuration reads eight coils from address '0', the only permitted read operation of the hardware. Node returns an array of eight Boolean values.

6.5.6 Example 11: Writing Coil status to a Modbus RTU Relay using the Modbus Protocol

This example demonstrates how to write to the Modbus RTU Relay, showcasing two different forms of writing: single and multiple values.

They relay used in this example: <https://www.waveshare.com/modbus-rtu-relay.htm>

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



```
[{"id":"ab51a98b8cff324d","type":"modbus-write","z":"5e1b0393890554df","name":"Relay write Coils","showStatusActivities":true,"showErrors":true,"unitid":"1","dataType":"MCoils","adr":"0","quantity":"8","server":"6e1c00798d661717","emptyMsgOnFail":false,"keepMsgProperties":false,"x":730,"y":1040,"wires":[["c055b132ecc35311"],[]]},{id":"c055b132ecc35311","type":"modbus-response","z":"5e1b0393890554df","name":"","registerShowMax":20,"x":950,"y":1040,"wires":[[]]},{id":"91b38740c3c8a991","type":"inject","z":"5e1b0393890554df","name":"Write Coil statuses [0,0,0,0,1,1,1,1]","props":[{"p":"payload"}],"repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"[0,0,0,0,1,1,1,1]","payloadType":"json","x":460,"y":1040,"wires":[["ab51a98b8cff324d"]]}, {"id":"410ce3cd5646aad7","type":"inject","z":"5e1b0393890554df","name":"Write Coil statuses [1,1,1,1,0,0,0,0]","props":[{"p":"payload"}],"repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"[1,1,1,1,0,0,0,0]","payloadType":"json","x":460,"y":1080,"wires":[["ab51a98b8cff324d"]]}, {"id":"2cef4b3c069dddb0","type":"modbus-write","z":"5e1b0393890554df","name":"Relay write Coil","showStatusActivities":true,"showErrors":true,"unitid":"1","dataType":"Coil","adr":"0","quantity":"1","server":"6e1c00798d661717","emptyMsgOnFail":false,"keepMsgProperties":false,"x":620,"y":940,"wires":[["b2aeed12262a9777"],[]]},{id":"5a7026bf24f970f7","type":"inject","z":"5e1b0393890554df","name":"Write Coil 0 True","props":[{"p":"payload"}],"repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"true","payloadType":"bool","x":400,"y":940,"wires":[["2cef4b3c069dddb0"]]}, {"id":"9cf498a5de01a01e","type":"inject","z":"5e1b0393890554df","name":"Write Coil 0 False","props":[{"p":"payload"}],"repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"false","payloadType":"bool","x":400,"y":980,"wires":[["2cef4b3c069dddb0"]]}, {"id":"b2aeed12262a9777","type":"modbus-response","z":"5e1b0393890554df","name":"","registerShowMax":20,"x":870,"y":940,"wires":[[]]},{id":"6e1c00798d661717","type":"modbus-client","name":"RTU","clienttype":"simpler","bufferCommands":false,"stateLogEnabled":false,"queueLogEnabled":false,"tcpHost":"127.0.0.1","tcpPort":"1188","tcpType":"DEFAULT","serialPort":"/dev/ttyS1","serialType":"RTU-BUFFERED","serialBaudrate":"9600","serialDatabits":"8","serialStopbits":"1","serialParity":"none","serialConnectionDelay":"100","serialAsciiResponseStartDelimiter":"0x3A","unit_id":"1","commandDelay":"1","clientTimeout":"1000","reconnectOnTimeout":true,"reconnectTimeout":"2000","parallelUnitIdsAllowed":true}]
```

The following flow of nodes will appear:

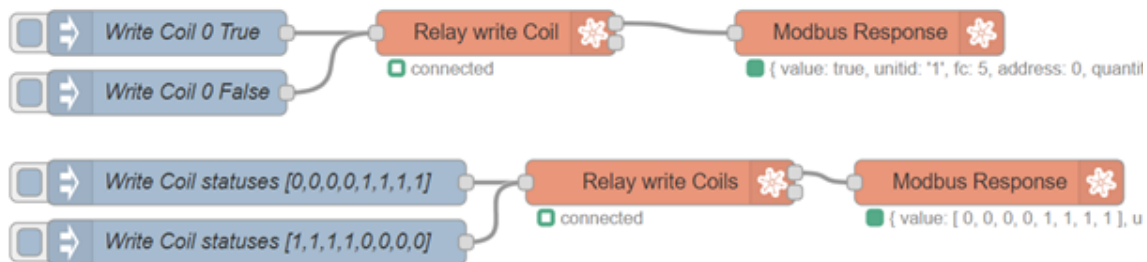


Figure 51: Example 11 – Writing Coil Nodes

The flow consists of multiple buttons, which set different values on coils inside the Modbus registers upon pressing. The "Relay write Coil" node is set up with function code 5, "force single coil," to force the predefined Coil (here, the Coil on address 0) to take a value from the "inject" node.

The "Relay write Coils" node is set up with function code 15, "force multiple coils," which forces Coils from starting address to passed values. The "inject" node passes eight values, which the Modbus-write node expects.

6.5.7 Example 12: Reading from a Real sensor using the Modbus Protocol

This example demonstrates how to read, convert and display read values in the correct format from an actual industrial sensor.

The sensor used in this example:

<https://www.seeedstudio.com/RS485-Temperature-and-Humidity-Sensor-p-5235.html>

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



```
[{"id":"e607447884854120","type":"inject","z":"5e1b0393890554df","name":"Read sensor","props":[{"p":"payload"}],"repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"","payloadType":"date","x":390,"y":700,"wires":[["19e80878f8c78e3c"]]}, {"id":"19e80878f8c78e3c","type":"modbus-getter","z":"5e1b0393890554df","name":"Read registers","showStatusActivities":false,"showErrors":false,"logIOActivities":false,"unitid":"1","dataType":"HoldingRegister","adr":"0","quantity":"3","server":"6e1c00798d661717","useIOFile":false,"ioFile":"","useIOForPayload":false,"emptyMsgOnFail":false,"keepMsgProperties":false,"x":580,"y":700,"wires":[["0108ead1ab83428b"]]}, {"id":"0108ead1ab83428b","type":"change","z":"5e1b0393890554df","name":"(temp / 100)°C","rules":[{"t":"set","p":"payload","pt":"msg","to":"payload/100&\"°C\"","tot":"jsonata"}],"action":"","property":"","from":"","to":"","reg":false,"x":780,"y":740,"wires":[["4a7a960f.6d3108"]]}, {"id":"4a7a960f.6d3108","type":"change","z":"5e1b0393890554df","name":"(hum / 100)%RH","rules":[{"t":"set","p":"payload","pt":"msg","to":"payload/100&\"%RH\"","tot":"jsonata"}],"action":"","property":"","from":"","to":"","reg":false,"x":780,"y":780,"wires":[["4170818c5d915ca6"]]}, {"id":"4170818c5d915ca6","type":"change","z":"5e1b0393890554df","name":"(dew / 100)%RH","rules":[{"t":"set","p":"payload","pt":"msg","to":"payload/100&\"%RH\"","tot":"jsonata"}],"action":"","property":"","from":"","to":"","reg":false,"x":820,"y":820,"wires":[["4a7a960f.6d3108"]]}, {"id":"4a7a960f.6d3108","type":"join","z":"5e1b0393890554df","name":"","mode":"custom","build":"array","property":"payload","propertyType":"msg","key":"topic","joiner":"\\n","joinerType":"str","accumulate":false,"timeout":"","count":"3","reduceRight":false,"reduceExp":"","reduceInit":"","reduceInitType":"","reduceFixup":"","x":990,"y":780,"wires":[["2a0d287b.23a8e8"]]}, {"id":"2a0d287b.23a8e8","type":"function","z":"5e1b0393890554df","name":"message","func":"const currentTime = new Date();\\nconst [temperature, humidity, dewPoint] = msg.payload;\\nmsg.payload = `At ${currentTime} (UTC), the temperature was ${temperature} with humidity ${humidity} and dew point ${dewPoint}`;\\nreturn msg;\\n","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":1140,"y":780,"wires":[["42e91c83.8d5084"]]}, {"id":"42e91c83.8d5084","type":"debug","z":"5e1b0393890554df","name":"","output":"Output","active":true,"tostobar":true,"console":false,"status":false,"complete":"payload","targetType":"msg","statusVal":"","statusType":"auto","x":1090,"y":840,"wires":[["6e1c00798d661717"]]}, {"id":"6e1c00798d661717","type":"modbus-client","name":"RTU","clientType":"simpler","bufferCommands":false,"stateLogEnabled":false,"queueLogEnabled":false,"tcpHost":"127.0.0.1","tcpPort":"1188","tcpType":"DEFAULT","serialPort":"/dev/ttyS1","serialType":"RTU-BUFFERD","serialBaudrate":"9600","serialDatabits":"8","serialStopbits":"1","serialParity":"none","serialConnectionDelay":"100","serialAsciiResponseStartDelimiter":"0x3A","unit_id":"1","commandDelay":"1","clientTimeout":"1000","reconnectOnTimeout":true,"reconnectTimeout":"2000","parallelUnitIdsAllowed":true}]
```

The following flow of nodes will appear:

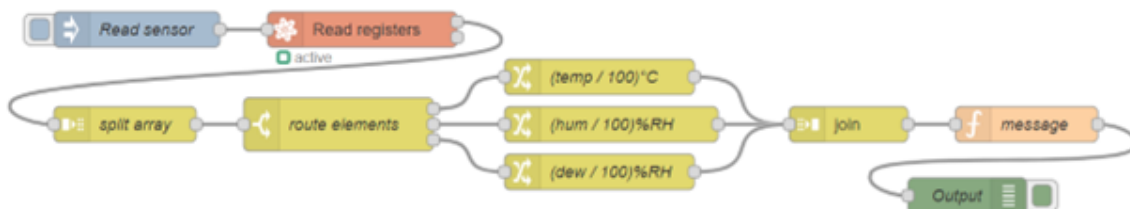


Figure 52: Example 12 – Real Sensor Nodes

The flow consists of a button, which reads registers inside the Modbus registers upon pressing. The Modbus-read node performs the reading. Configuration of the node is to use Modbus function code 3, which reads a specified number of holding registers from the starting address. In this case, the configuration reads three registers from address '0'. Node returns an array of three values: split, routed, converted, joined, and formed to an output.

Example of output:

On Thu Apr 13, 2023, 11:48:04 GMT+0000 (GMT) (UTC), the temperature was 23.16°C with humidity of 37.39%RH and dew point of 7.83°C

6.6 Resources

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

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

7. Related Documents

- [1] Router Apps: icr.advantech.com/router-apps
- [2] JS Foundation: <https://nodered.org/>
- [3] Advantech Czech: **Node.js Application note** (APP-0080-EN)
- [4] Advantech Czech: **Bluetooth Application note** (APP-0098-EN)
- [5] Advantech Czech: **Programming of Router Apps Application Note** (APP-0025-EN)

You can obtain product-related documents on *Engineering Portal* at icr.advantech.com address.

To get your router's *Quick Start Guide*, *User Manual*, *Configuration Manual*, or *Firmware* go to the [Router Models](#) page, find the required model, and switch to the *Manuals* or *Firmware* tab, respectively.

The *Router Apps* installation packages and manuals are available on the [Router Apps](#) page.

For the *Development Documents*, go to the [DevZone](#) page.