

# Modbus and Bluetooth data sending to Node-RED

□

## Contents

- [1 Introduction](#)
- [2 Configuration overview and prerequisites](#)
- [3 Node-RED installation and setup](#)
- [4 MQTT Gateway](#)
  - [4.1 Using Teltonika device as an MQTT broker](#)
  - [4.2 Node-RED setup when using Teltonika device as an MQTT broker](#)
  - [4.3 Using Node-RED as an MQTT broker](#)
    - [4.3.1 Adjusting Node-RED configuration](#)
    - [4.3.2 Adjusting Teltonika device configuration](#)
  - [4.4 MQTT gateway using MQTT explorer and Node-RED broker](#)
    - [4.4.1 Device setup](#)
    - [4.4.2 MQTT explorer setup and testing](#)
- [5 Modbus Data to Server via MQTT protocol](#)
  - [5.1 Device configuration](#)
  - [5.2 Node-RED setup](#)
- [6 Bluetooth Data to Server via MQTT protocol](#)
  - [6.1 Device configuration](#)
  - [6.2 Node-RED setup](#)
- [7 Modbus Data to Server via HTTP protocol](#)
  - [7.1 Device configuration](#)
  - [7.2 Node-RED setup](#)
  - [7.3 Testing the configuration](#)
- [8 Summary](#)
- [9 References](#)

## Introduction

The information in this page is updated in accordance with the [RUTXXX\\_R\\_00\\_07\\_03](#) firmware version.

Node-RED is a flow-based programming tool developed by IBM Emerging Technology and written in Node.js. It provides a browser-based editor for wiring together hardware devices, APIs, and online services using a visual programming interface. More information about Node-RED could be found [here](#).

This article provides an extensive configuration example with details on how to use Node-RED with Teltonika Routers via MQTT and HTTP protocols.

# Configuration overview and prerequisites

Before we begin, let's overview the configuration that we are attempting to achieve and the prerequisites that make it possible.

Prerequisites:

- Teltonika RUTXXX router or TRBXXX gateway. We are going to use the RUTX11 in this example.
  - At least one end device (PC, Laptop, Tablet, Smartphone) to configure the devices.
  - Linux Virtual Machine to host Node-RED server.
- 

There are a couple of different use cases with Node-RED and Teltonika devices:

1. [MQTT gateway using either Node-RED or Teltonika device as MQTT broker.](#)
2. [Modbus data to server using either Node-RED or Teltonika device as MQTT broker.](#)
3. [Bluetooth data to Node-RED server.](#)
4. [HTTP data to Node-RED server.](#)

## Node-RED installation and setup

We are going to set up Node-RED in Linux virtual machine. For Node-RED to work, you would need to install Node.js version 14.00 or higher, if you already have Node.js installed, verify Node.js version using this command:


```
node -v
```

If you do not have Node.js installed, run these commands to install it:


```
sudo apt install curl
curl -fsSL https://deb.nodesource.com/setup_14.x | sudo -E bash -
sudo apt-get install -y nodejs
```

Once we have Node.js installed, we can install Node-RED. Use this command to install:

```
sudo npm install -g --unsafe-perm node-red
```

Use the command *node-red* to start a local server. Here is how the terminal should look like if the server starts correctly: 

---

Once you have the local server running, use the local IP and port number in your internet browser. In this case, we are using **127.0.0.1:1880**: 

For MQTT usage, we are going to need MQTT-specific nodes. Use the side menu to navigate to Manage Palette section and install these nodes:

- node-red-contrib-aedes
- node-red-contrib-mqtt-broker



## MQTT Gateway

MQTT Gateway allows to query the device from MQTT Clients, in this section we will configure it to work with the Node-RED.

### Using Teltonika device as an MQTT broker

---

Here is connection topology for this use case:



Router setup when using the router as a broker consists of two parts. First, we are going to set up a local broker. In order to do so, navigate to **Services -> MQTT -> Broker** and enable the broker.



Once done, navigate to **Services -> Modbus -> MQTT Gateway**. Follow these steps for the configuration:

1. **Enable:** on
2. **Host:** since we are using a local broker on the router - input 127.0.0.1

Leave everything else as default, or adjust according to your needs



### Node-RED setup when using Teltonika device as an MQTT broker

---

For this example, we are going to need **MQTT-in** and **MQTT-out** nodes. We are not going to need a broker node for now, since we are using our router as an MQTT broker. In order to send request and get response messages, we are also going to need **inject** and **debug** nodes. Drag required nodes onto the screen and click on them to configure accordingly:

- **MQTT-out node** - click on this node and then click on the **pencil** icon to configure Node-RED for Teltonika broker usage.



In the next screen configure settings accordingly:

1. **Name:** RUT\_broker for this example
2. **Server:** 192.168.10.1 (Router's LAN IP address)
3. **Port:** Default port 1883
4. Press **Update**



---

Now we are going to go back to MQTT-out node configuration and adjust settings as shown below:

1. **Server:** RUT\_broker
2. **Topic:** request
3. Press **Done**



---

**MQTT-in node** - configure settings as shown below:

1. **Server:** RUT\_broker
2. **Action:** Subscribe to single topic
3. **Topic:** response
4. **QoS:** 0
5. **Output:** auto-detect
6. Press **Done**



- 
- **Inject node** - here we only need to input a payload. To find required message format, refer to this article: [Modbus TCP Master MQTT Gateway](#)

For this example, we will try to get router's temperature, so we are going to use payload in this format: **0 65432 0 192.168.10.1 502 5 1 3 6 2**



There is no need to configure **debug** node. Here is how the finished flowchart should look like:



---

In order to test the configuration, press **Deploy** in the top right corner:



Now press on **Inject** node and look for incoming message on the right side of the screen:



Here we can see that the router's temperature is 38.0 degrees.

## Using Node-RED as an MQTT broker

---

Here we are using Node-RED as MQTT Broker, so we are going to do necessary changes in order for

this functionality to work.

Here is the connection topology for this use case:



### Adjusting Node-RED configuration

If you would like to use Node-RED as a broker, there are a couple of changes needed to be made. Similarly as shown before, click on **MQTT Subscriber** node and click edit in the Server section. Input name **(1)** and address **127.0.0.1 (2)** in order to use Node-RED broker and press **Done**.



In order for the broker to be available, add broker node to the flowchart. Now the flowchart should look like this:



### Adjusting Teltonika device configuration

Navigate to **Services -> Modbus -> MQTT Gateway**. Change the host IP to your Linux Virtual Machine IP address (192.168.10.139 in this example):



To test the configuration, **deploy** the nodes again and send the message by clicking on the **inject** node.

## MQTT gateway using MQTT explorer and Node-RED broker

---

In this example, we are going to use the same Node-RED broker configured in previous parts of this article, only this time we are going to use [MQTT Explorer](#) app to Publish and Subscribe to Modbus data.

Here's connection topology for this use case:



### Device setup

---

Navigate to **Services -> Modbus -> MQTT Gateway**. Follow these steps for the configuration:

1. **Enable:** on
2. **Host:** input **192.168.10.139**, or if you would like to use local broker on the device - input **127.0.0.1**

Leave everything else as default, or adjust according to your needs



## MQTT explorer setup and testing

---

Open MQTT Explorer application and in the default screen input the **Host IP (1)** of MQTT Broker, then press **Connect (2)**.



Once done, locate Publish section on the right side. You can use the same settings as in previous sections of MQTT Gateway:

1. **Topic:** request
2. **Data:** 0 65432 0 192.168.10.1 502 5 1 3 6 2



If everything is set up correctly, then you should be able to see response from the router:



## Modbus Data to Server via MQTT protocol

In this section, we are going to upload Modbus data to server on Node-RED. First of all, we are going to need Modbus data source, in this example we are using the same device as Modbus TCP Slave and Master.

Here's connection topology for this use case:



### Device configuration

---

To start with, navigate to **Services -> Modbus -> Modbus TCP slave**. Press **enable** to enable the instance and you can leave everything else as **default**:



Next, navigate to **Services -> Modbus -> Modbus TCP master**. Adjust settings here accordingly:

1. **Enabled:** on
2. **Name:** Local\_slave (input any preferred name)
3. **Slave ID:** 1
4. **IP address:** 127.0.0.1 (since we are using local TCP slave - input the IP address of your slave if you are using external device as Modbus Slave)
5. **Port:** 502



Add new request and adjust settings according to your needs. For this example, we are going to pull device name from the registers, so our configuration looks like this:

1. **Name:** test (input any preferred name)
2. **Data type:** ASCII (to get a string of text)
3. **Function:** Read holding registers (3)
4. **First register number:** 72
5. **Register count/values:** 3
6. **Brackets:** use brackets

Press enable to enable the Modbus request.



---

To finish up the device configuration, navigate to **Services -> Data to server**. Add new data sender and configure settings accordingly:

1. **Enable:** on
2. **Name:** Data\_to\_node (input any preferred name)
3. **Data source:** MODBUS data
4. **Protocol:** MQTT
5. **JSON format:** {"Router name": %a} - adjust this according to your needs
6. **URL/Host/Connection string:** 192.168.10.139 (virtual machine address)
7. **Port:** 1883
8. **Topic:** rutx (input any preferred topic, make sure to subscribe on the same topic)



## Node-RED setup

---

Node-RED setup to get Modbus data is simple, we are going to need one **MQTT Subscriber** node, one **MQTT broker** node so we could use broker on Node-RED and we will use **debug** node to read MQTT messages. Here is how the flowchart should look like:



Adjust MQTT Subscriber node settings accordingly:

1. **Server:** Local (from previous example)
2. **Action:** Subscribe to single topic
3. **Topic:** rutx

Leave everything else as default.



Deploy the nodes and look for incoming data in the debug window - based on **Period** time in Data to server settings. Here is incoming message with router name:



If you would like to use MQTT broker on the Teltonika device, change **Server: Local** to Server: **RUT\_Broker** (which uses the IP 192.168.10.1).

# Bluetooth Data to Server via MQTT protocol

In this section, we are going to upload Bluetooth data to server on Node-RED. The configuration is going to be similar to Modbus data to server. More information about Bluetooth functionality and usage with beacons could be found in [Teltonika EYE device pairing and data sender configuration example](#).

Here's connection topology for this use case:



## Device configuration

---

First of all we are going to need a Bluetooth device which is paired to Teltonika device. For this example, we are using Teltonika [EYE Beacon](#) and we have it paired to the device. You can pair Bluetooth device by navigating to **Services -> Bluetooth** and clicking **Scan**. Then select your device and click **Pair**. Here's how a paired device looks like:



After that, navigate to **Services -> Data to server**. Add new data sender and configure settings accordingly:

1. **Enable:** on
2. **Name:** BT\_data (input any preferred name)
3. **Data source:** Bluetooth data
4. **Protocol:** MQTT
5. **JSON format:** {"Data": "%b", "Hour": "%d"} - adjust this according to your needs
6. **URL/Host/Connection string:** 192.168.10.139 (virtual machine address)
7. **Port:** 1883
8. **Topic:** beacon (input any preferred topic, make sure to subscribe on the same topic)



## Node-RED setup

---

The setup is the same as with Modbus data to server, just change MQTT Subscriber settings to subscribe on the **beacon** topic. if everything is setup correctly, you will be able to see incoming messages on the right side of the screen:



# Modbus Data to Server via HTTP protocol

In this section, we will adjust the configuration in order to get the data via HTTP protocol.

Here's connection topology for this use case:





## Device configuration

---

Look into Modbus data to server via MQTT section for Modbus Slave and Master setup. Navigate to navigate to **Services -> Data to server**. Add new data sender and configure settings accordingly:

1. **Enable:** on
2. **Name:** tcp (input any preferred name)
3. **Data source:** MODBUS data
4. **Protocol:** HTTP(S)
5. **JSON format:** {"Data": %a} - adjust this according to your needs
6. **URL/Host/Connection string:** 192.168.10.139:8080 (virtual machine address and port number)



## Node-RED setup

---

For this configuration we are going to need two nodes: **TCP** and **debug**. Drag in both nodes and click on TCP node. Adjust settings accordingly:

1. **Type:** Listen on port **8080**
2. **Output:** stream of **String** payload



Flowchart for this example should look like this:



## Testing the configuration

---

If configured correctly, you should be able to see incoming Modbus data to the Node-RED debug window:



## Summary

In this article, several use cases are discussed, such as MQTT gateway using Node-RED or Teltonika device as MQTT broker, Modbus data to server, Bluetooth data to server, HTTP data to Node-RED server, and MQTT gateway using MQTT Explorer. The article provides steps on how to install Node-RED on a Linux virtual machine, install MQTT-specific nodes, and set up a local server. The article also explains how to use the Teltonika device as an MQTT broker and set up Node-RED to work with it.

# References

[Node-RED](#)

[Modbus TCP Master MQTT Gateway](#)

[Teltonika EYE device pairing and data sender configuration example](#)

[MQTT Explorer](#)