https://wiki.teltonika-networks.com/view/OpenRemote

OpenRemote

Contents

Introduction

OpenRemote an intuitive user-friendly 100% open source IoT platform. You can build a complete IoT device management solution including: device management and auto provisioning, customisation of asset types, automation via when-then, flow, javascript and groovy rules, data analytics, connectivity via several protocol agents and manager APIs (e.g. MQTT broker, HTTP/REST, WS), Multi-tenancy (realms), Users and roles management, Edge gateway, Front-end UI web components and consoles, and an Insights dashboard builder.

In this article you will find instructions on how to setup OpenRemote Manager as a MQTT Broker and how to send telemetry data using Teltonika device to the OpenRemote platform.

Prerequisites

You will need:

- OpenRemote system
- A router from RUTx, TCRx or TRBx series

Hardware Description

Hardware descriptions can be found in different Quick Start Guides (QSG). There you will find an overview of the various components on the front and back of a device, hardware installation instructions, first login information, device specifications, and general safety information. Link: Quick Start Guides

Set up your Development Environment

Teltonika Networks devices comes with our created firmware, therefore no additional development or scripting is required for this unit to support OpenRemote.

Setting up OpenRemote manager

Refer to the online OpenRemote documentation at Quckstart. Follow the steps in the section **Quickstart** to create your own environment with full access.

NOTE: In examples, port `1883` was used for MQTT protocol. To use this port you need to add it to your `docker-compose` file.

OpenRemote as MQTT broker and subscriber

Create a service user

The service user will give programmatic access to the MQTT client.

- Go to the users page and create a new service user (second panel on the page).

- Name the service user 'mqtt_user' and give the user the read and write role for the sake of convenience. It is advised to configure a more restricted role for your service users.

- Click 'Create', a secret will be generated automatically.
- Open the 'mqtt_user' user to see and copy the secret.

Create an asset with attributes

We will create a Thing asset (MQTT subscriber), but feel free to use an AssetType that matches your physical device.

- On the assets page, click the + in the asset tree on the left.

- Select the Thing asset type, give it a friendly name and click 'Add'.

- Add new attribute in the Edit asset mode and save configuration:
  - type: Custom Attribute
  - name: writeAttribute
  - Valuetype: JSON

RUT as MQTT Publisher

What data do you want to publish?

Depending on your RUT model, you can have the following options of data sources to send over MQTT:

- Bluetooth
- Wifi scanner
- MODBUS

Publishing Bluetooth/Wifi scanner/Modbus data

Before sending data from those sources to the MQTT Broker, the router needs to know what data from each source to send:

- For Bluetooth, first you have to pair the device you want to get data from
- For WiFi scanner data, you have to enable the wifi scanner functionality first
- For MODBUS data, you need to set what data from what MODBUS slave the router has to get

If you need any help on setting up each functionality, the Teltonika Wiki has topics for each one of them: Bluetooth, WiFi Scanner, MODBUS

First example: Publishing RUT MODBUS to OpenRemote manager using MQTT

In this example the RUT device will act as MODBUS TCP Master and MODBUS TCP Slave, so the device will make requests (Master) and answer to himself (Slave). The received reply, will be sent over MQTT. You can also send data from another MODBUS Slave devices connected to the router.

Enabling MODBUS TCP Slave

Enabling the MODBUS Slave option on the router allows it to answer any requests coming from a MODBUS Master. To do that, go to the router configuration page→Services→MODBUS→MODBUS TCP Slave. Then clock the "Enable" slider and save.

Enabling MODBUS TCP Master

Enabling the MODBUS Master option on the router allows it to make specific requests to any slave in the MODBUS network. To do that, go to the router configuration page→Services→MODBUS→MODBUS TCP Master. Click on the "Add" button, and do the following configuration on the page:

"Name" : Any name

"Slave ID" : The slave ID you've set on the slave configuration, by default its 1

"IP address" : The LAN IP address of the router, by default its 192.168.1.1

"Port" : 502

"Period" : 10

"Timeout" : 5

Then scroll down the page a bit, on the "Add new request" section, give any name to your request and click on the "Add" button. Then do the following configuration

"Data type" : Data type of the data you are going to receive, in this case, the router reports its device system uptime

"Function" : MODBUS Protocol function, in this case, we are going to Read holding registers

"First register number" : Depends on your MODBUS device, in this case, its 2

"Register count / Values" : Depends on your MODBUS device, in this case, its 2

Save and apply settings.

Configuring Data to Server

Go to the router configuration page→Services→Data to server. Click on the "Add" button, and the configuration is basically the same as described on the "Publishing Bluetooth/Wifi scanner/Modbus data" section of this article, just change the data source to "MODBUS data" and format the data as you wish. You should have something similar to this.

"Name" : Any name

"Data source" : Modbus data

"Protocol" : MQTT

"JSON format" : Use **Quotation marks** and insert between them JSON segment.

"Segment count" : All

"URL / Host / Connection string" : Host IP of your OpenRemote system.

"Topic" : Define the correct topic. For directly writing an attribute value: {realm}/{clientID}/writeattributevalue/{attributeName}/{assetID}. So in our case this will be `master/client123/writeattributevalue/writeAttribute/79hl5XRczN4mjIaQWIMazv`

"Client ID" : Use the client ID that is used in the Topic field

"Period" : Data sending frequency (in seconds)

"Use credentials" : Enable

"Username" : `master:mqtt_user` ({realm}:{user})

"Password" : The secret generated for the MQTT service user (you can find it on the `mqtt_user` Users page)

Checking if it works

Then, you can open the "OpenRemote" system to check if your setup works, if everything was setup correctly, you should see the values update in attributes section. In the example we could see that the data "Slave name - %N and registry data (JSON object) - %a" is being updated.

Second example: Publishing Bluetooth data (from Eye Sensor) to OpenRemote manager using MQTT

In this example the RUT device will collect Bluetooth data from EYE Sensor. The received data will be sent over MQTT to OpenRemote manager.

Enabling Bluetooth service

Enabling the Bluetooth service on the router allows it to pair Bluetooth device and collect data from it. To do that, go to the router configuration page→Services→Bluetooth. Then click the "Enable" slider and click on the "Scan" button:

Select device and pair it:

Configuring Data to Server

Go to the router configuration page→Services→Data to server. Click on the "Add" button, and the configuration is basically the same as described on the "Publishing Bluetooth/Wifi scanner/Modbus data" section of this article, just change the data source to "Bluetooth data" and format the data as you wish. You should have something similar to this.

"Name" : Any name

"Data source" : Bluetooth data

"Protocol" : MQTT

"JSON format" : In this case was used - %b

"Segment count" : 1

"Send as object" : Enable

"URL / Host / Connection string" : Host IP of your OpenRemote system

"Topic" : Define the correct topic. For directly writing an attribute value: {realm}/{clientID}/writeattributevalue/{attributeName}/{assetID}. So in our case this will be `master/client123/writeattributevalue/writeAttribute/79hl5XRczN4mjIaQWIMazv`

"Client ID" : Use the client ID that is used in the Topic field

"Period" : Data sending frequency (in seconds)

"Use credentials" : Enable

"Username" : `master:mqtt_user` ({realm}:{user})

"Password" : The secret generated for the MQTT service user (you can find it on the `mqtt_user` Users page)

Checking if it works

Then, you can open the "OpenRemote" system to check if your setup works, if everything was setup correctly, you should see the values update in attributes section. In the example we could see that the data from bluetooth device EYE Sensor "Device data (JSON object) - %b" is being updated.

References to OpenRemote

1. https://openremote.io/
2. https://github.com/openremote/openremote#readme
3. https://github.com/openremote/openremote/wiki
4. Forum https://forum.openremote.io/