LoRaWAN Deployment Scenario 1: Using your Gateway as a LoRaWAN Server Hub

WisDevice Series
RAK7249/RAK7258

Version 1.0 | July, 2019

www.RAKwireless.com

Visit our website for the latest copy of this manual.

34 PAGES

Table of Contents

1 Overview

The integrated ChirpStack is designed to turn the LoRa Gateway from a device acting as a simple packet forwarder, to a complete solution for monitoring LoRa traffic over applications and devices.

This is a complete solution on its own, you get the packet metadata in its entirety, together with the decrypted payload. However, you can still extend the system by integrating external gateways, creating custom payload encoders/decoders for the purpose for in app visualization.

2 LoRa Network Server Features Set

2.1 General

There is a Built-in LoRa Network Server in the Commercial Gateway line of device. It is perfect for both testing use case scenarios and integration with existing deployments of Nodes and Gateways. We are going to explain its key features and how to configure those. Furthermore this document will address a typical deployment scenario.

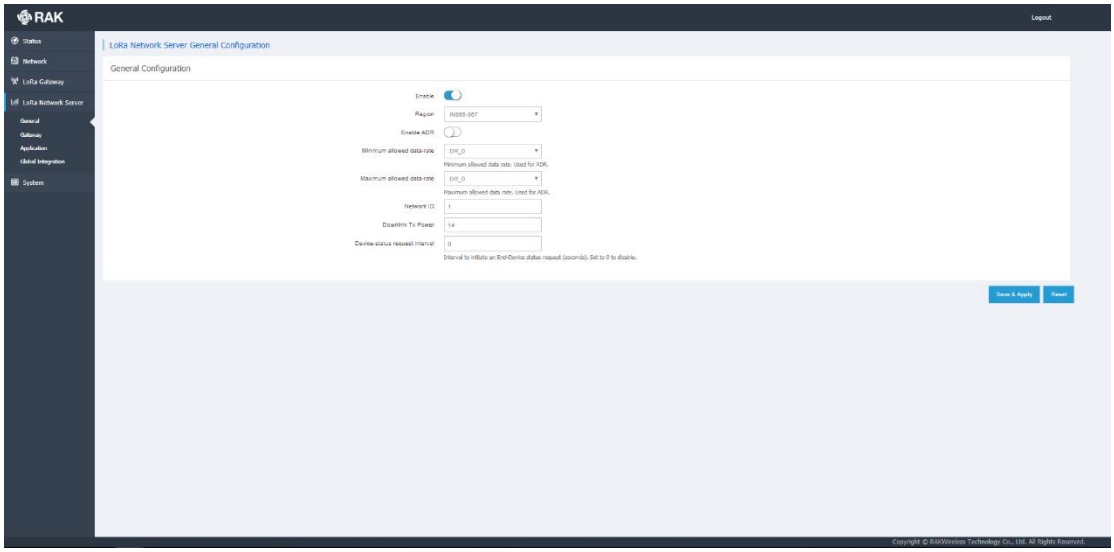

Figure 1 | LoRa Network Server

Enable ADR

If this feature is turned on via the slider, Adaptive Data Rate is enabled. One can set the following parameters:

Minimum allowed data-rate: DR_0 to DR_15

Maximum allowed data-rate: DR_0 to DR_15

By setting the above values, you choose the maximum and minimum allowed rate, meaning it can be anything in between depending the propagation environment. For more details on the possible data rates look up page 16 in the LoRa Regional Parameters document.

Network ID: A number denoting the particular network the gateway is a part of.

Downlink Tx Power: The power in dBm the Gateway will be transmitting with (for example 14 dBm, the limit in the EU)

Device-status request interval: The time in seconds over which the Gateway sends a status request downlink. Enter *0* to disable it.

2.2 Gateway

This is where you can add an external Gateway to forward packets to the Built-in LoRa Server. You can do this for multiple Gateway.

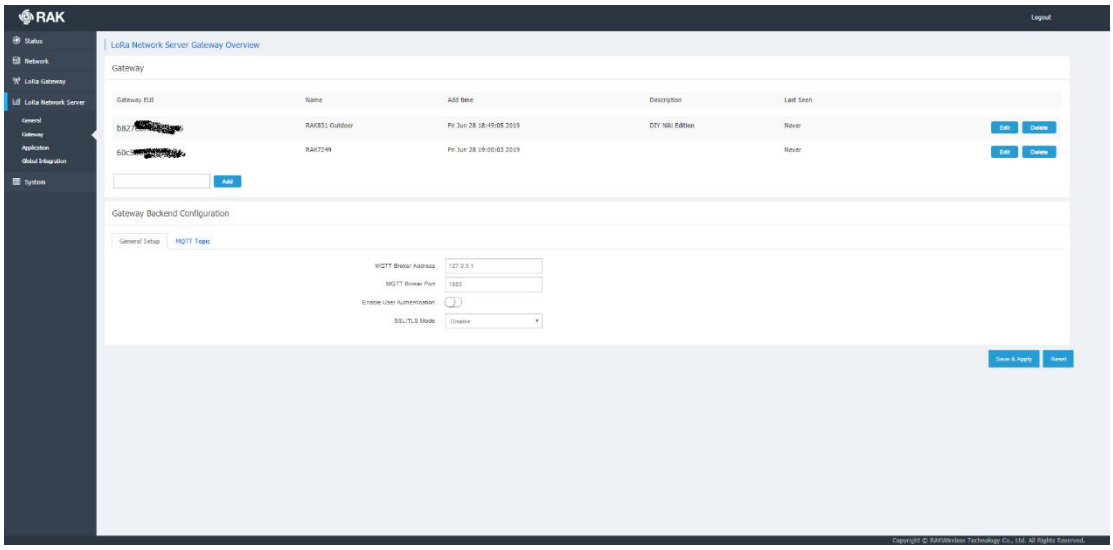

Figure 2 | Gateway tab

General Setup

MQTT Broker Address:

The address of the MQTT broker where the traffic from the External Gateways is directed.

MQTT Broker Port:

The Port of the MQTT broker

Enable User Authentication:

By turning on this slider, you are presented with options for entering an *Username* and *Password*, as well as an *SSL/TLS Mode* (*Self-signed Server Certificate, CA signed server certificate, Self-signed server & client certificate*). Depending on the chosen certificate option you are prompted to choose a *TLS Version* and import, the certificates in the provided fields (see Figure 3 for an example)

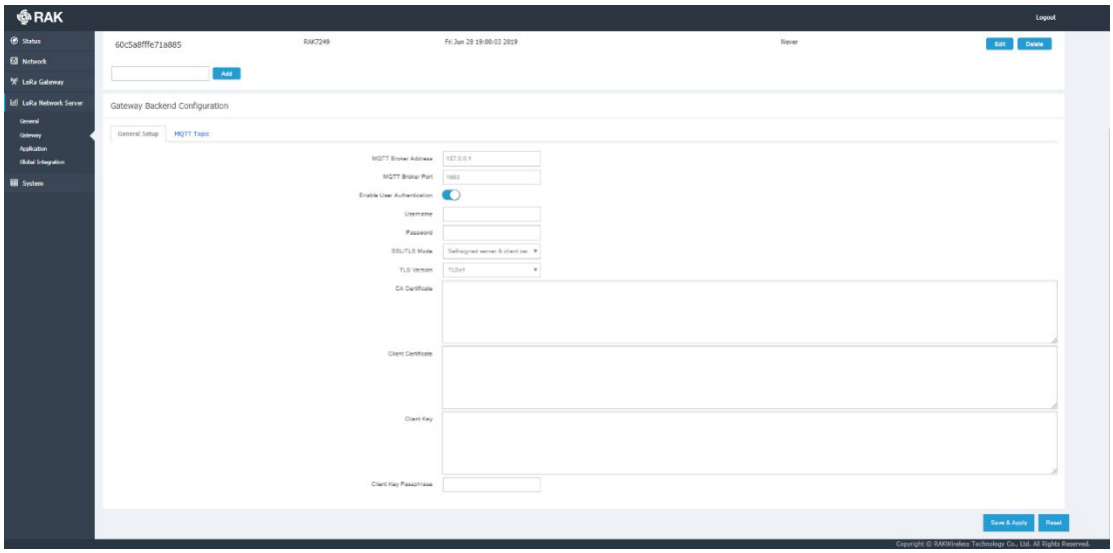

Figure 3 | Gateway Backend Configuration

MQTT Topic

Templates for the following MQTT Topics are provided, in order to be able to acquire the desired data:

Uplink MQTT Topic

Downlink MQTT Topic

Downlink acknowledgment MQTT Topic

Geteway Statistic MQTT Topic

Note: *eui* stands for the particular Gateway EUI, of the device you want to gather data from.

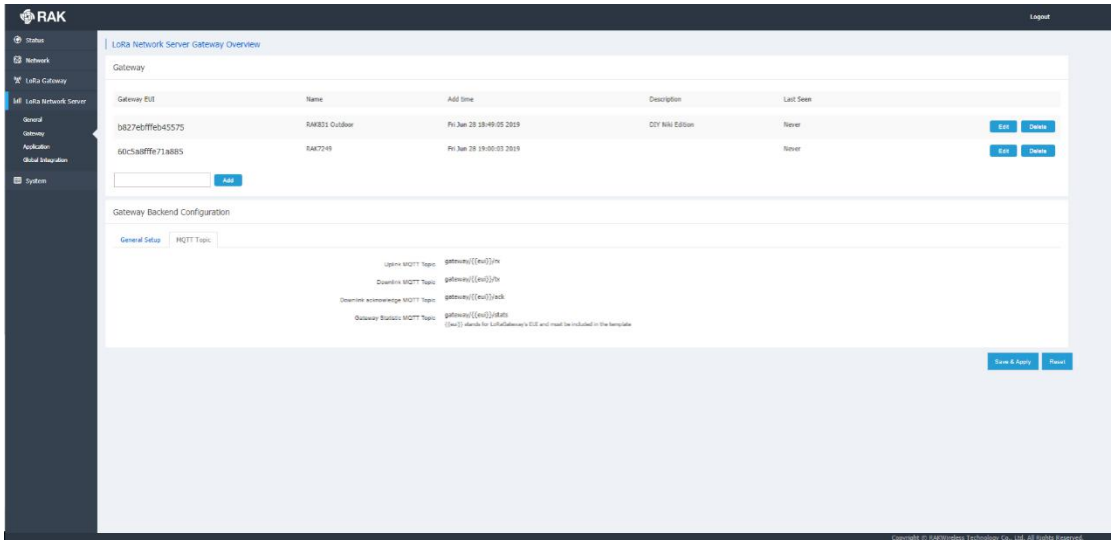

Figure 4 | Adding Gateways

2.3 Application

This is where you add your application. Simply enter a name in the text field and press the “Add” button. You will be directed to a window as shown in Figure 5.

Application Configuration

You can choose to edit the *Application Configuration*, where the *Name* and *Description* fields reside.

Device

In order to register a LoRa node you need to add it as a device, and enter its parameters, so it is recognized. In order to do this you need an appropriate device EUI.

Adding devices can be done in one of the following ways (provided you have their EUIs):

One by one:

Simply enter the EUI in the field and press the “Add” button

Batch:

You need to fill in the following parameters: Start EUI, Step, Count, and Application Key.

The step is a decimal value that represents by how much the value of the EUI will be increased with each consecutive device. This will be done starting from the least significant bit.

The count is the maximum number of devices to be added. Note that if your step is anything different from 1 you will essentially add less devices than the Step value. You will end up with a number of devices that is the Integer Division of the Count by the Step. For example if your Step is 3 and your Count is 10 you will end up with 3 Devices.

The Application Key is an AES-128 value, which is common for all devices under a given application.

Note: When Batch Adding devices they are all configured in Class A, OTAA mode, with Frame counter validation enabled.

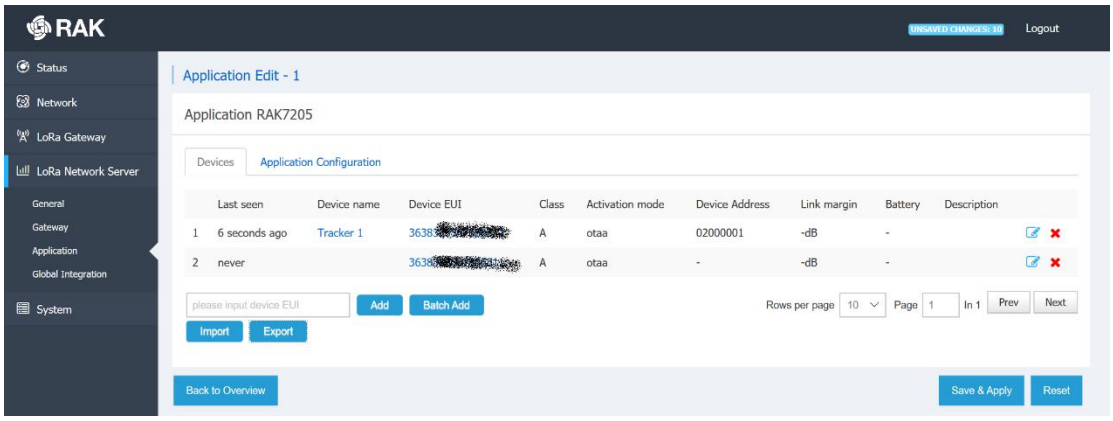

Figure 5 | Adding Devices

Import:

You can import a whole list of devices at once.

Export:

You can export the current device list.

Once you have a device created you will be redirected to the following screen (or you can enter it any time you want by pressing the *Device EUI* number field in the list of devices).

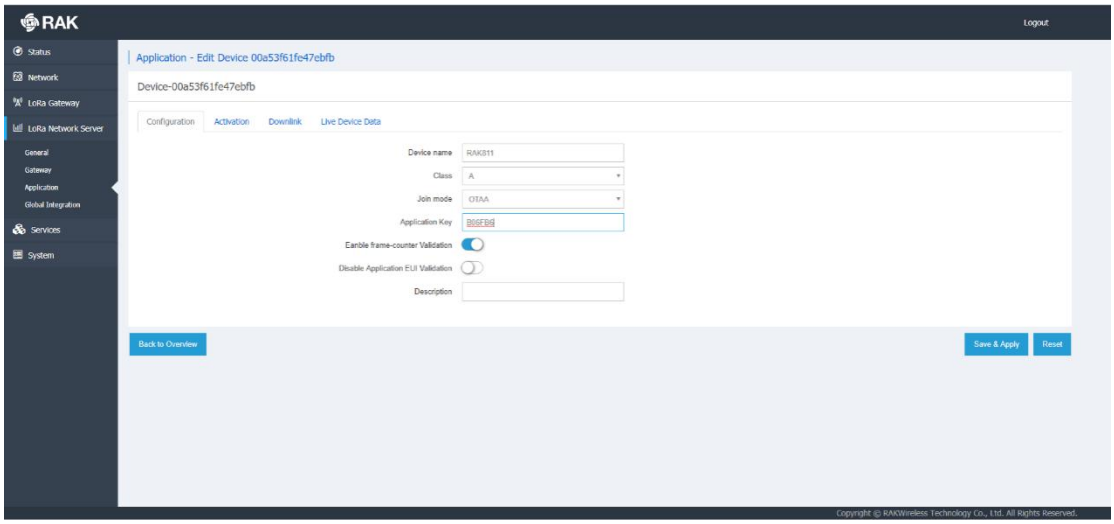

Figure 6 | Device Parameters

Configuration:

The *Device Name*, *Class* (A or C), *Join Mode* (OTAA, ABP), *Application Key*, and *Description* fields are here. Additionally there is a slider for to *Enable frame-counter Validation* and *Disable Application EUI Validation*.

Activation:

Once the LoRa Network Server has authenticated the device (provided the EUI and Application Key are valid), you will see those fields updated with the corresponding values as in Figure 6.

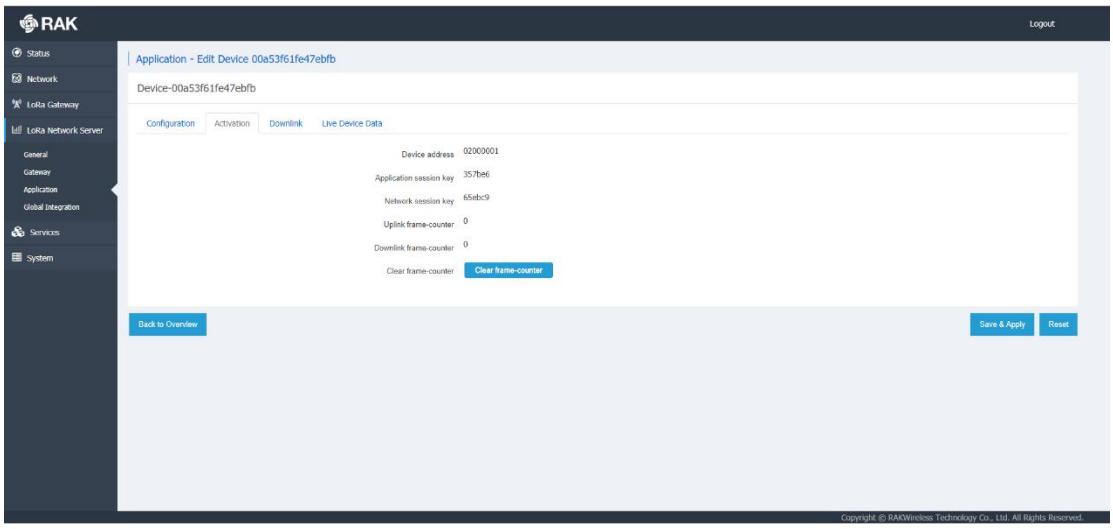

Figure 7 | Device Activation

Live Device Data:

All the packets are displayed here. They each have a timestamp, a type identifier (Uplink, Downlink, Join), followed by the payload data.

If you expand any of the messages you can see the metadata fields, an example is shown in Figure 7:

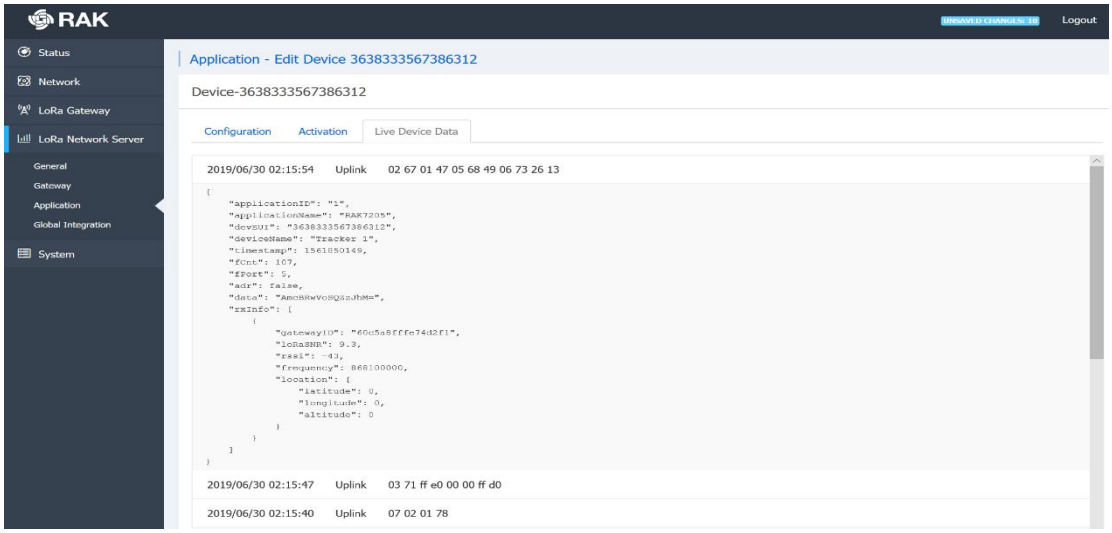

Figure 8 | Live Device Data

2.4 Global Integration

This is the tab where you can configure your LoRa Network Server to integrate with an external Application Server. This is done by pointing to the corresponding MQTT broker:

General Setup

MQTT Broker Address:

The address of the MQTT broker where the traffic from the External Gateways is directed.

MQTT Broker Port:

The Port of the MQTT broker

Enable User Authentication:

By turning on this slider, you are presented with options for entering an *Username* and *Password*, as well as an *SSL/TLS Mode* (*Self-signed Server Certificate, CA signed server certificate, Self-signed server & client certificate*). Depending on the chosen

certificate option you are prompted to choose a *TLS Version* and import, the certificates in the provided fields (see Figure 3 for an example)

MQTT Topic template Setup

Templates for the following MQTT Topics are provided, in order to be able to acquire the desired data:

Uplink MQTT Topic

Downlink MQTT Topic

Downlink acknowledgment MQTT Topic

Geteway Statistic MQTT Topic

Note: *eui* stands for the particular Gateway EUI, of the device you want to gather data from.

3 A Typical Application Case

We will deploy a typical scenario. We will walk through the configuration process of all devices that are required. Additionally we will explain in detail after we are done configuring, what are the benefits of this particular use-case, the message formats, etc.

Configuration of the devices will be done in order as per the topology in the next sub-section.

3.1 Network Topology

- Gateway-A: RAK7249/58 Nexus Gateway (LoRa Server in use)
- Gateway-B: RAK7249/58 External Gateway (MQTT Bridge in use)
- RAK 811 WisNode-LoRa: LoRa node

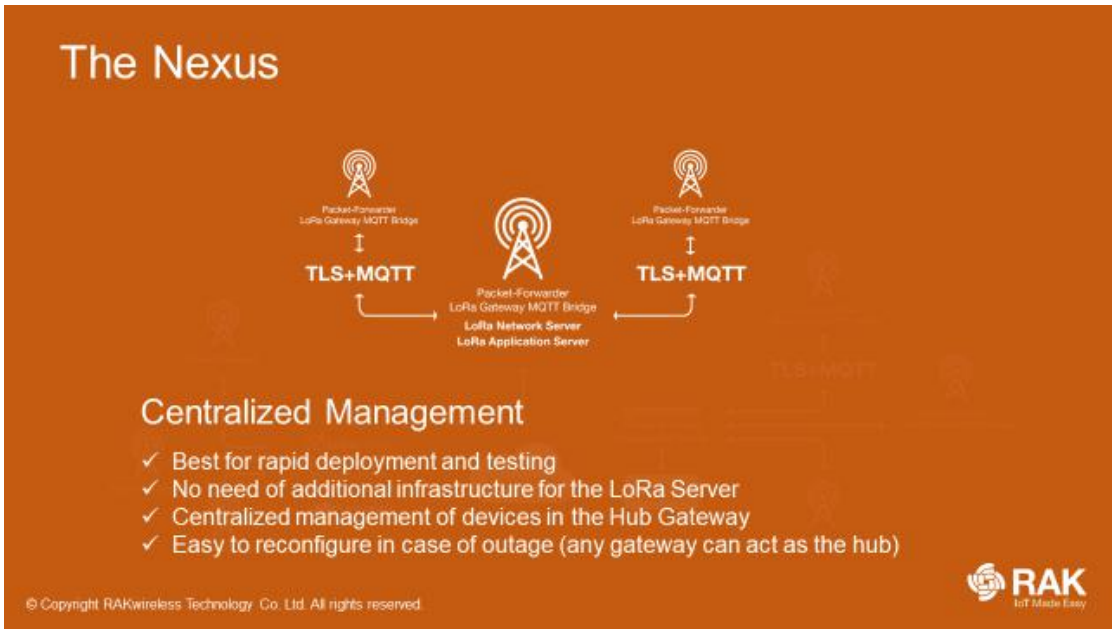

Figure 9 | Network Topology

This is the minimum of devices required. However you can integrated more nodes and also more External Gateways. As an advanced configuration feature we will forward all the traffic from the Nexus Gateway to a MQTT broker, hosted separately. This is not mandatory, however is good practice and required in some cases.

3.2 Gateway-A

3.2.1 Packet Forwarder Setup

Go to the *LoRa Gateway tab -> LoRa Packet Forwarder -> General Setup.* In the drop-down menu for *Protocol* select:
Built-in LoRa Server (refer to Figure 10):

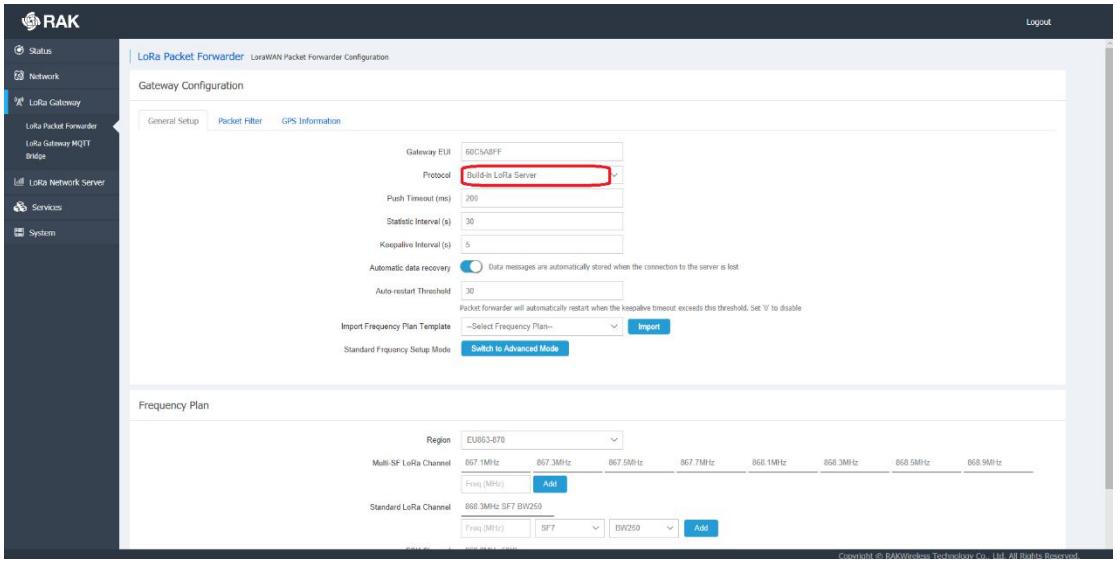

Figure 10 | Gateway Protocol Mode (Built-in LoRa Server)

You can leave the rest of the settings with their default values. Remember to *Save & Apply*.

3.2.2 LoRa Server General Configuration Setup

Go to the *LoRa Network Server tab -> General.* Make sure the *Enable switch* is on.

Select your region (LoRa Band), we are going to use EU863-870 in this example (refer to Figure 11):

The rest of the settings you can use with their default values. You can change them is you like, however this is dependent on your particular case. You can look up what each setting is referring to in **Section 1**.

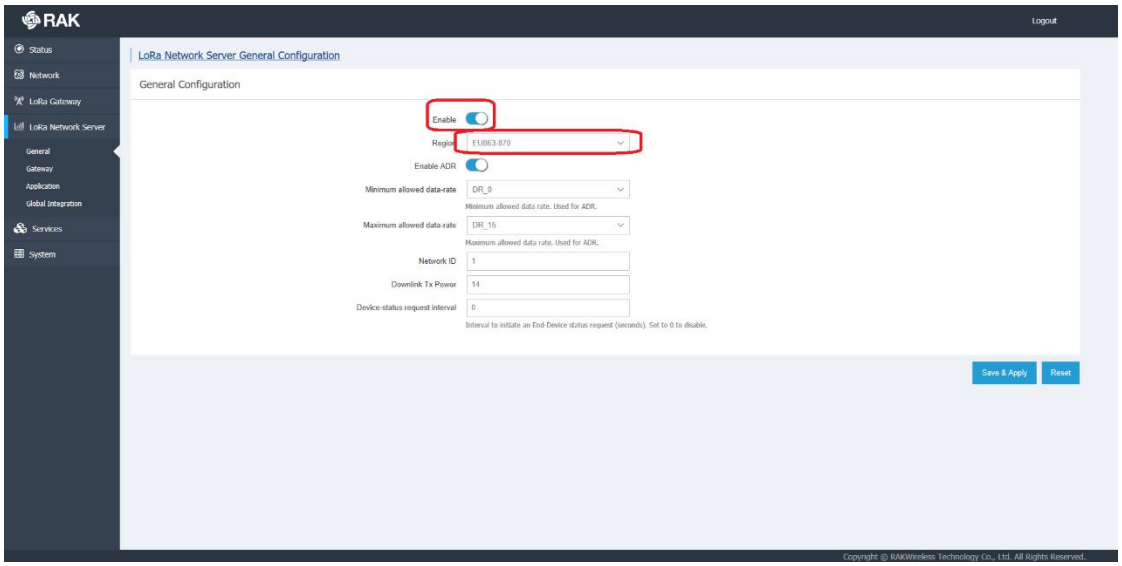

Figure 11 | LoRa Server Enabling and General Configuration

3.2.3 Registering the Gateway with the Built-in LoRa Server

Go to the *LoRa Network Server tab -> Gateway.* Enter the Gateway EUI in the field as in Figure 12.

By pressing the *Add* button you should end up with a screen as in Figure 13.

There you need to fill the parameters: *Name* and *Description* are mandatory.

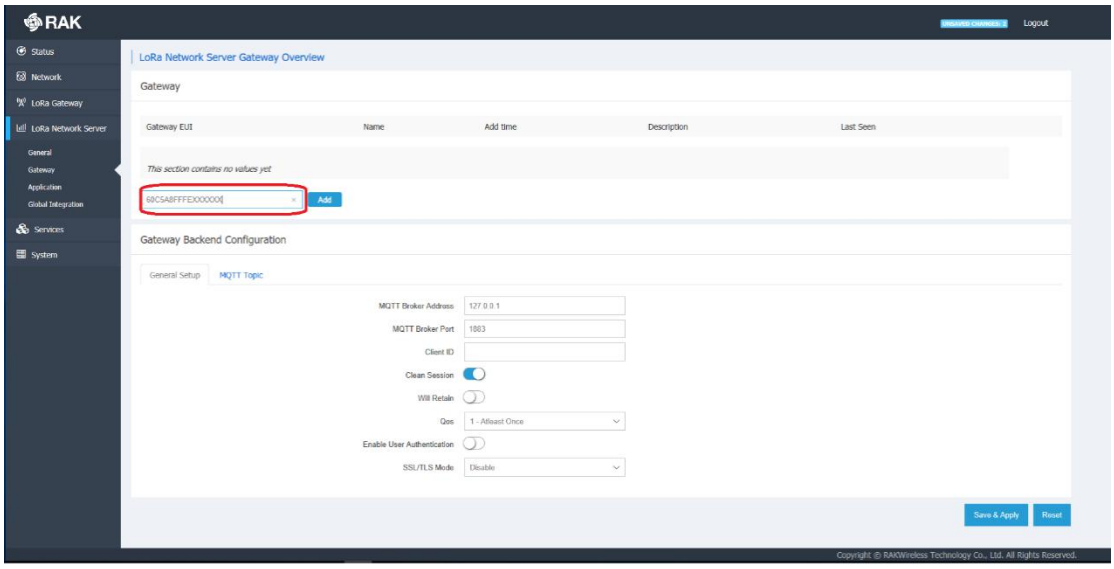

Figure 12 | Adding a Gateway into LoRa Server

Note: The Latitude, Longitude and Altitude parameters are not mandatory. You can leave them for later, or leave them empty if the gateway is not stationary.

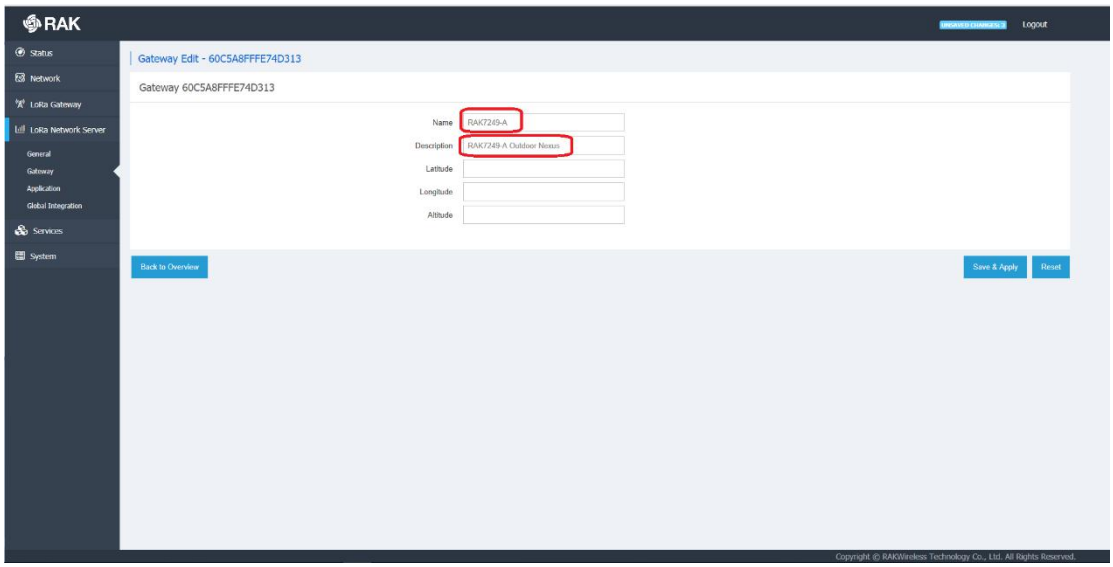

Figure 13 | Gateway Description parameters

If everything is set up correctly, you will see an image similar to Figure 14.

Note: In order to see the Last Seen status update you need to refresh the page. There should be a value of a couple of seconds, if so than everything went well. In case there is a message “Never Seen”, there is an issue and you best redo the configuration.

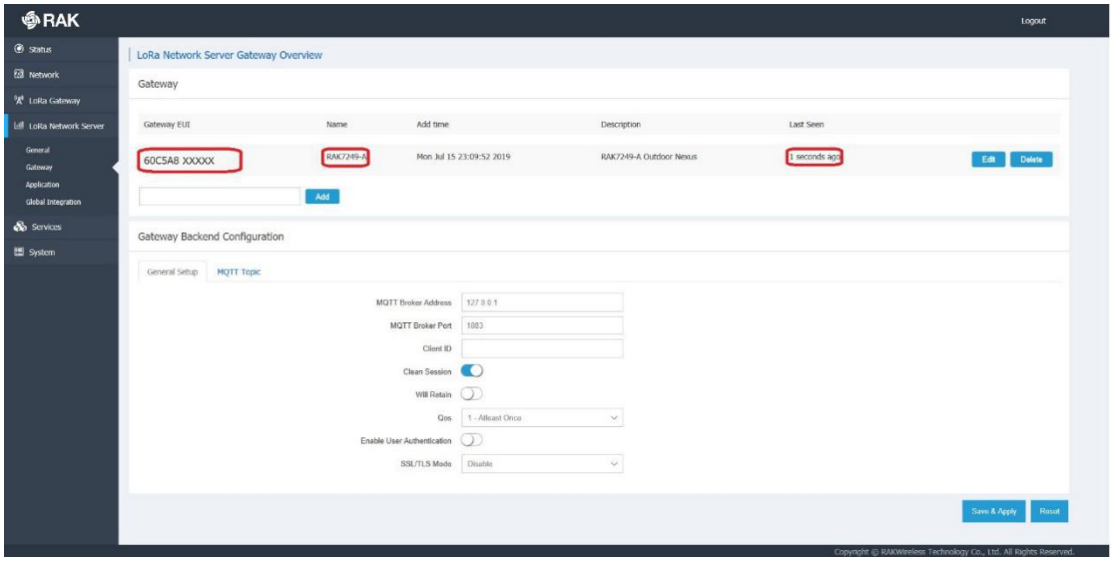

Figure 14 | Gateway Description parameters

3.3 Gateway-B

Now we will point Gateway-B to the built-in LoRa Server of Gateway-A

3.3.1 Packet-forwarder Setup

Go to the *LoRa Gateway tab -> LoRa Packet Forwarder -> General Setup.* In the drop-down menu for *Protocol* select:
LoRa Gateway MQTT Bridge (refer to Figure 15):

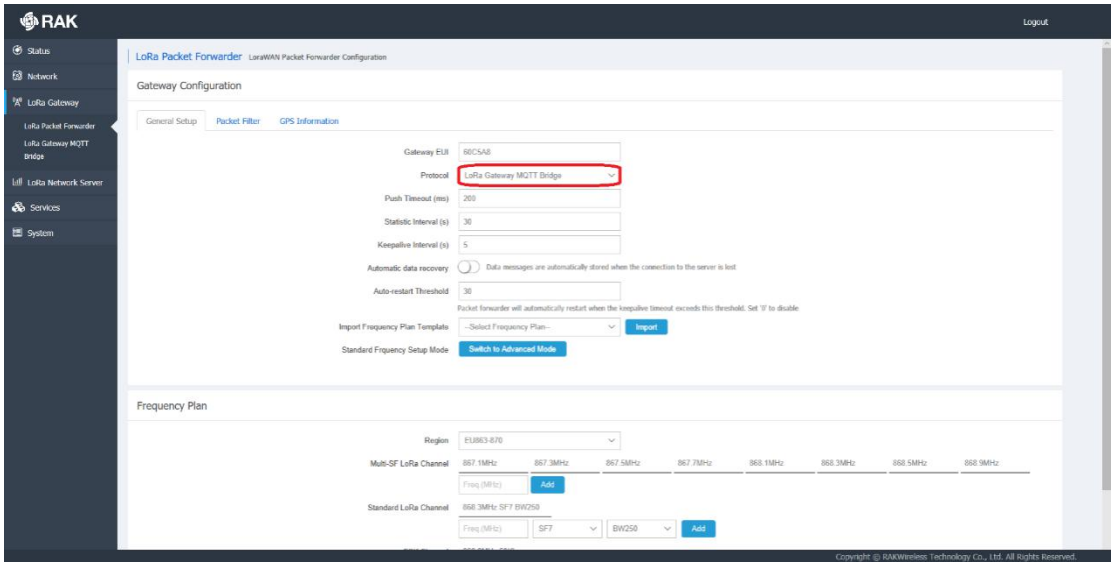

Figure 15 | Gateway Protocol Mode (LoRa Gateway MQTT Bridge)

3.3.2 LoRa Gateway MQTT Bridge

Go to the *LoRa Gateway tab -> LoRa Gateway MQTT Bridge -> General Setup.*

Enable the MQTT Bridge it via the slider and enter the IP address of Gateway-A (the one in Figure 16 is just an example).

The port should be 1883 by default, if it isn't please update the data. Leave the rest of the settings with their default values.

After *Saving* & *Applying*, all LoRa traffic should be redirected via the Bridge of Gateway-B to the MQTT Brocker of Gateway-A.

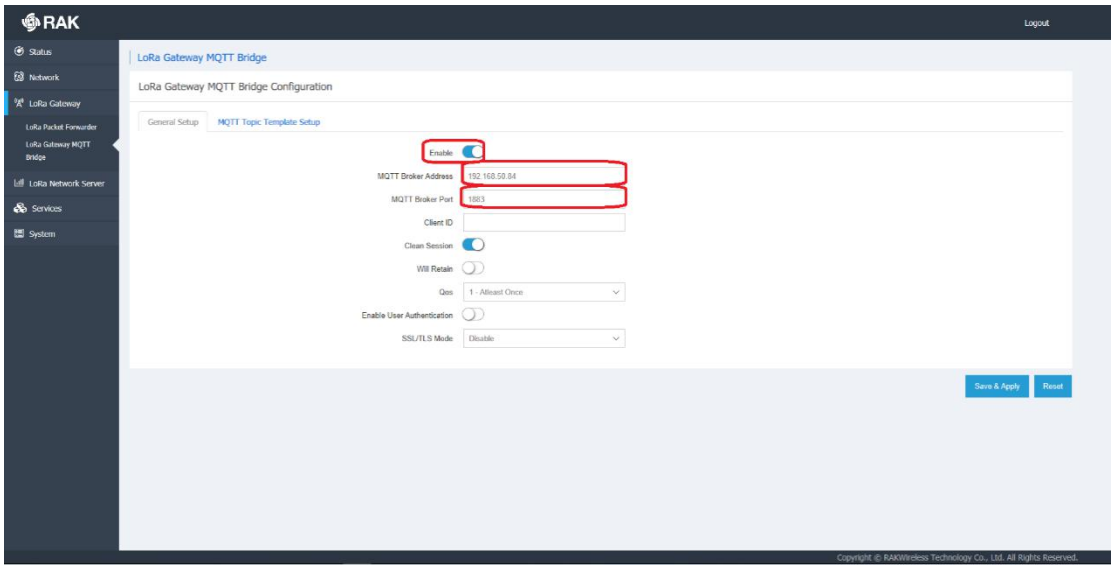

Figure 16 | LoRa Gateway MQTT Bridge Configuration

3.3.3 Registering Gateway-B in Gateway-A's LoRa Network Server

The procedure is the same as when we registered Gateway-A in its built-in LoRa Network Server. Refer to sub-section 2.2.3 on how to repeat the process.

Figure 17 is a representation of what you configuration should look like with the two Gateways added.

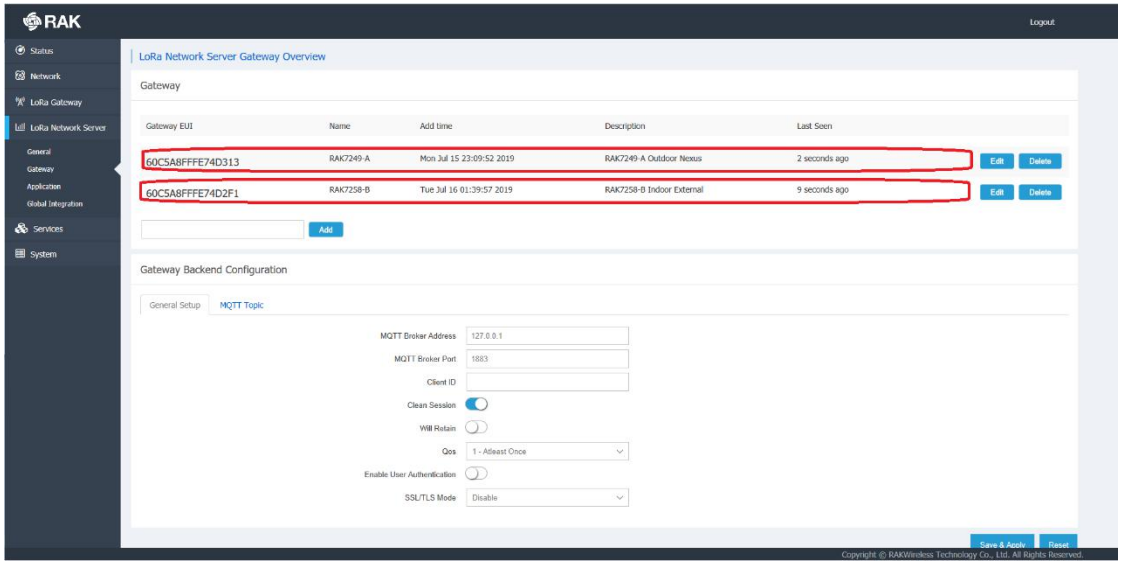

Figure 17 | LoRa Network Server Gateway List

You can add more Gateway in the same manner as we did for the two we are using. This is a convenient way to monitor if they are up (Last Seen field).

3.4 Setting up the External MQTT Broker

3.4.1 Preparing the Raspberry Pi

We are going to use going to use a Raspberry Pi 3B+ for this tutorial, as the device that is going to be hosting Mosquitto (a popular MQTT broker).

First download the latest Raspbian Buster Lite image from the link.

Next flash the image to an SD card with a tool such as Etcher.

We recommend setting up the Raspberry Pi headless. Once done plug the SD card into the slot and power it.

Use your favorite SSH client to connect to the Raspberry Pi (username: pi, password: raspberry).

Now as we have a platform to work with we can begin.

First execute the following command and note the IP address of the interface you will be using to connect to the network. You will need this, as it will be the address for your MQTT Broker when configuring the Gateway:

ifconfig

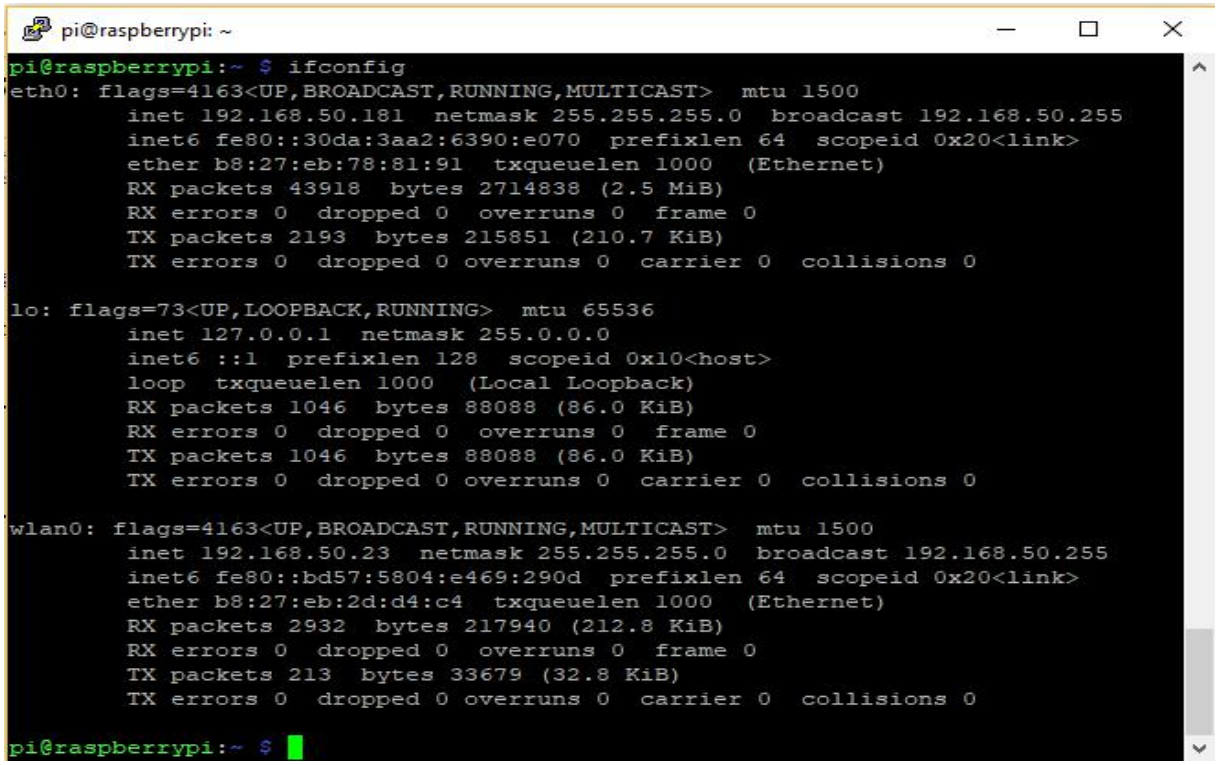

Figure 18 | Raspberry Pi interfaces

3.4.2 Installing Mosquitto

Now it is time to install the MQTT Broker (Mosquitto), via the command:

sudo apt install mosquitto mosquitto-clients

```
Get:2 http://archive.raspberrypi.org/debian buster InRelease [25.1 kB]
Get:3 http://raspbian.raspberrypi.org/raspbian buster/main armhf Packages [13.0
MB]
Get:4 http://archive.raspberrypi.org/debian buster/main armhf Packages [205 kB]
Fetched 13.2 MB in 19s (705 kB/s)
Reading package lists... Done
pi@raspberrypi:~ $ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~ $ sudo apt-get update
Hit:1 http://raspbian.raspberrypi.org/raspbian buster InRelease
Hit:2 http://archive.raspberrypi.org/debian buster InRelease
Reading package lists... Done
pi@raspberrypi:~ $ sudo apt install mosquitto mosquitto-clients
Reading package lists... Done
Building dependency tree
Reading state information... Done
mosquitto is already the newest version (1.5.7-1).
mosquitto-clients is already the newest version (1.5.7-1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~ $
```

Figure 19 | Mosquitto installation

Mosquitto clients help us easily test MQTT through a command line utility. We will use two command windows, one to subscribe to a topic and one to publish a message to it. Those will be explained in detail further in the tutorial.

sudo systemctl enable mosquitto.service

This command is not mandatory, however it is recommended as it creates a *mosquitto service* that will run the broker on startup.

3.4.3 Configuring the Gateway to publish to the MQTT Broker

Now we are going to configure the Gateway to connect to our external MQTT broker.

For the purpose of this example we are going to use the built-in LoRa Server.

First go to the *Packet Forwarder Tab* and choose *Built-in LoRa Server* as your *Protocol*:

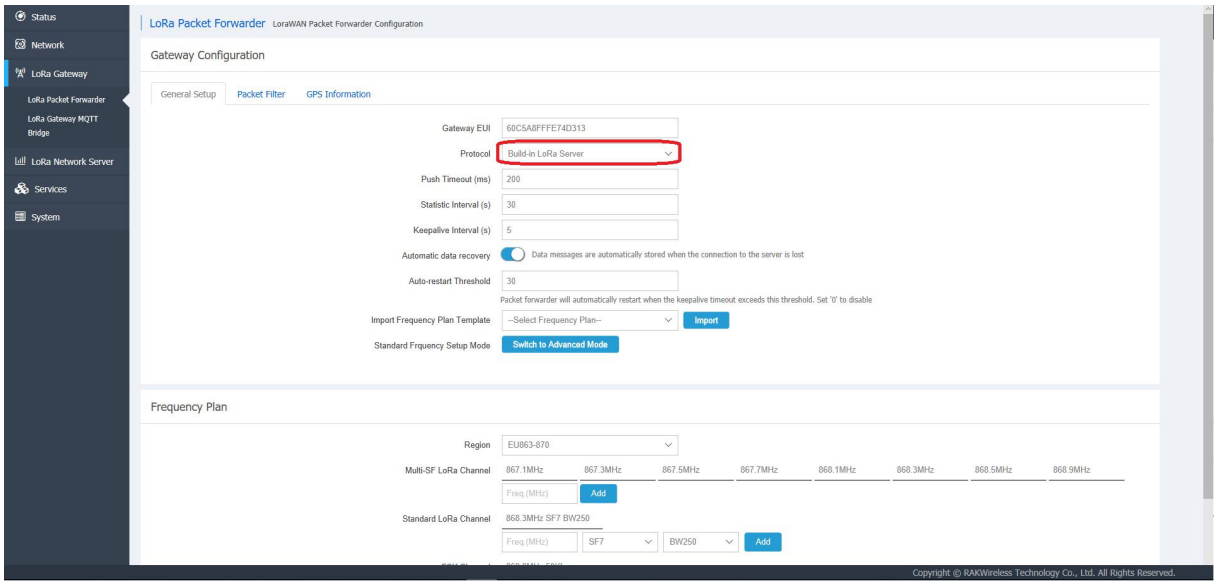

Figure 20 | Protocol selection

Make sure you have the *LoRa Network Server* enabled in the *General* tab:

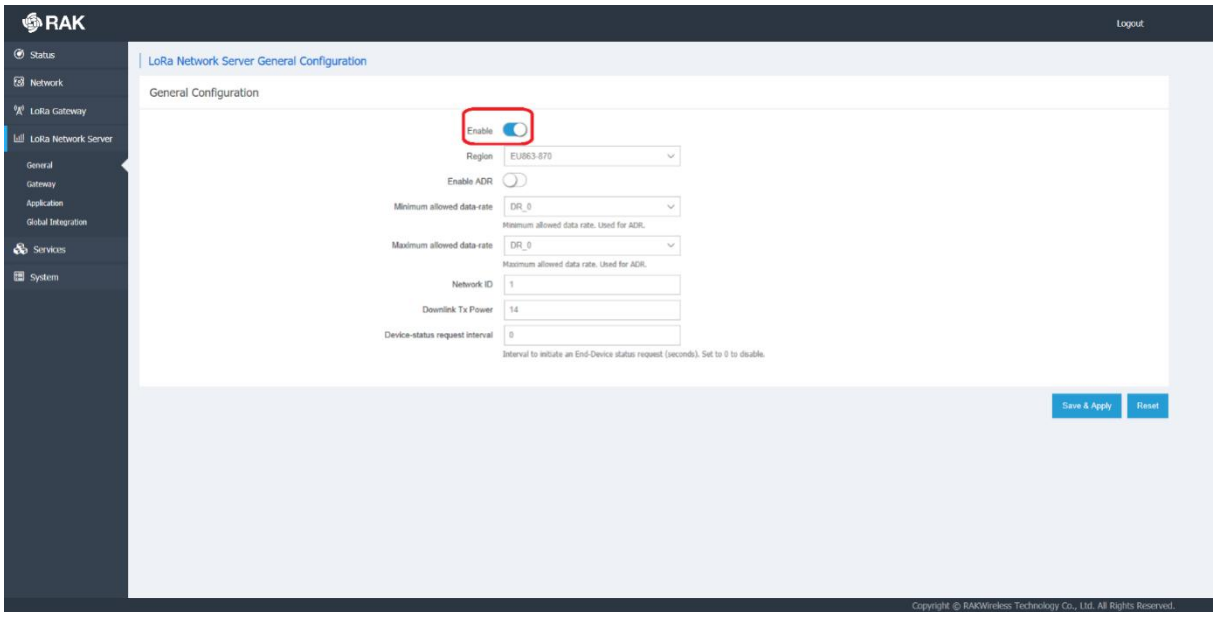

Figure 21 | Built-in LoRa Server activation

Add Your Gateway in the *Gateway* tab if you haven't done so already (You can add multiples Gateways here):

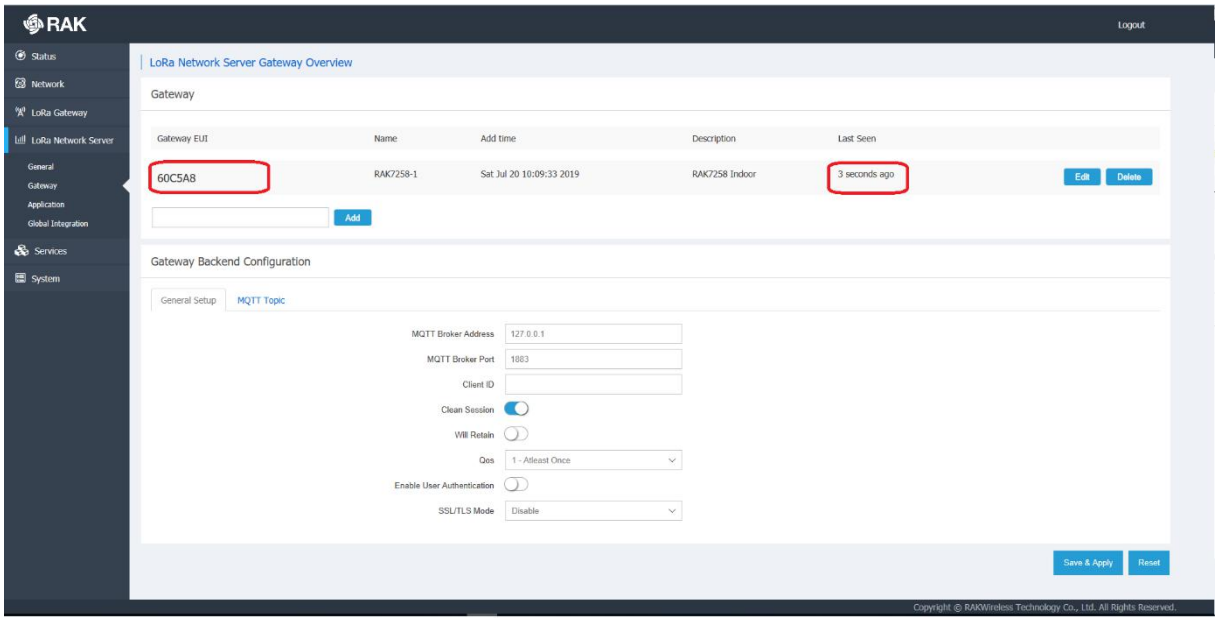

Figure 22 | LoRa Server Gateway configuration

Finally go to the *Global Integration* tab and enter the address where you have your *Mosquitto* instance running in the *MQTT Broker Address* field, leave the *Port* with the default *1883 value.*

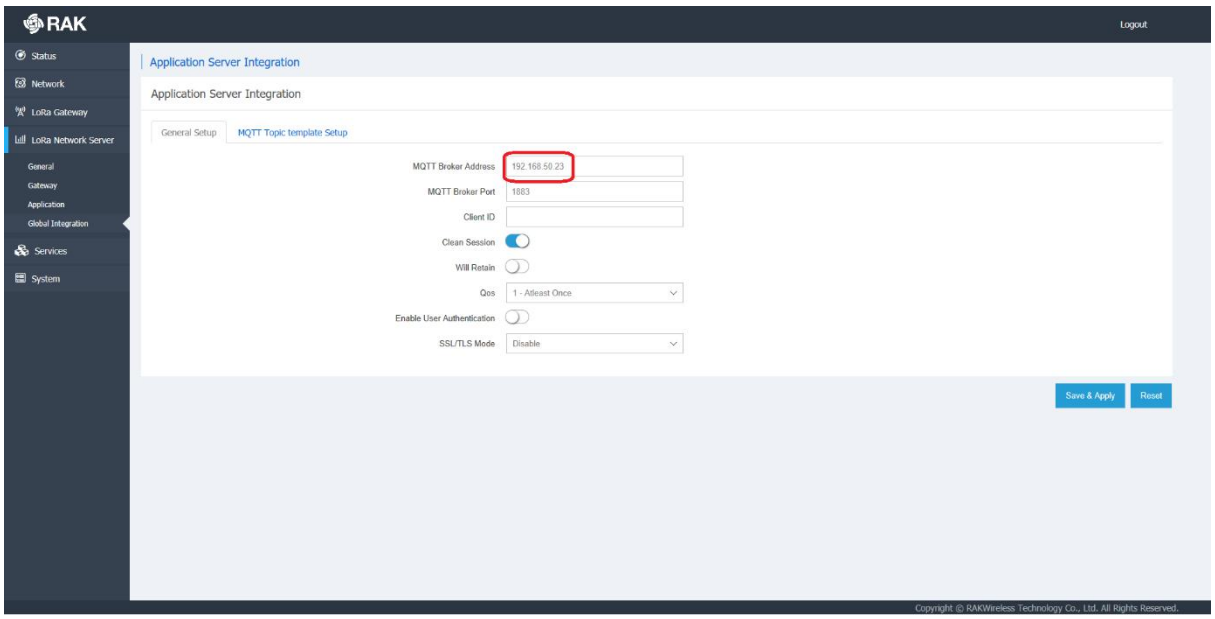

Figure 23 | Setting up the Global Integration

Now your Gateway parameters are set, except for one last part. You need to register your application in order to be able to send and receive data. We are going to use the RAK811 WisNode as an example in the next sub-section.

3.4.4 Registering the Application

Open your Serial tool, connect to the node and open the corresponding port. Reboot it so you can see the device parameters as in Figure X below:

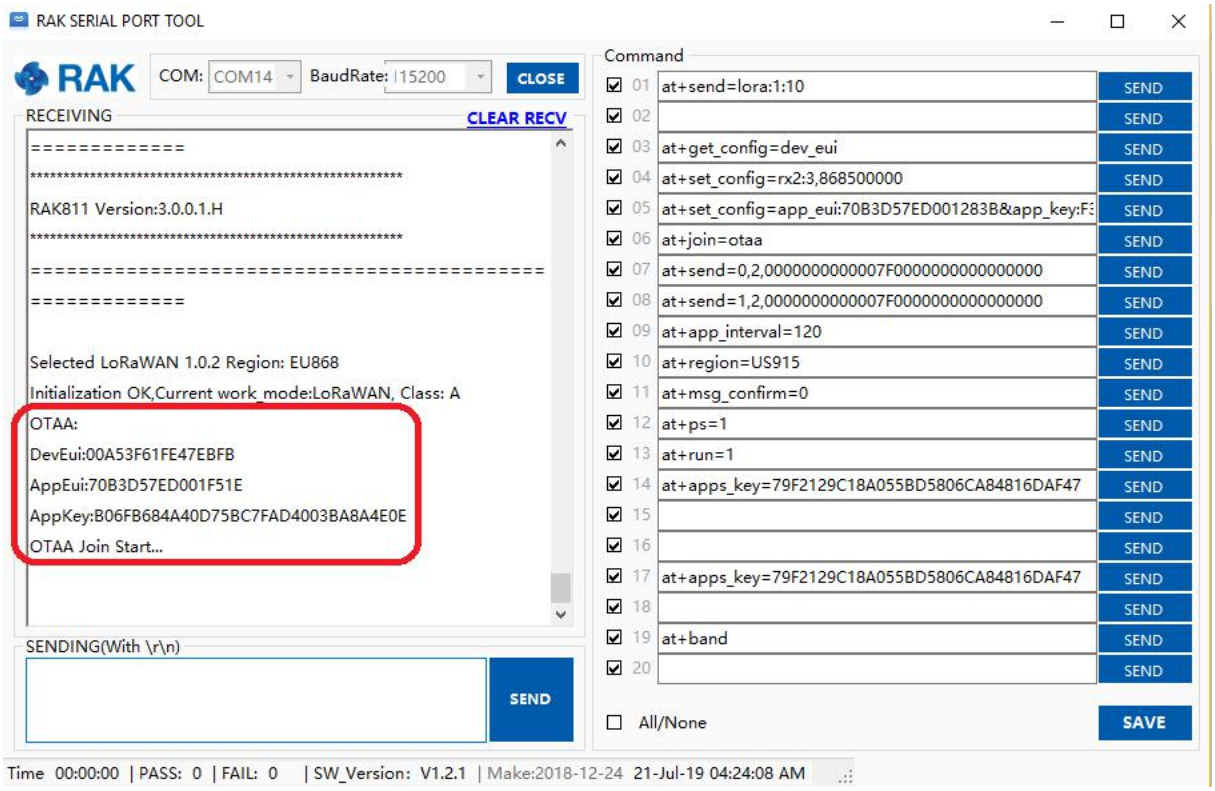

Figure 24 | RAK811 parameters

In case your device is already configured to work in OTAA (this example) it will attempt connecting to the gateway and getting authenticated. As it is not yet registered this will not be successful. We need to do some configuring first.

Execute the command to change the working region/band (EU868 in this example):

at+set_config=lora:region:EU868

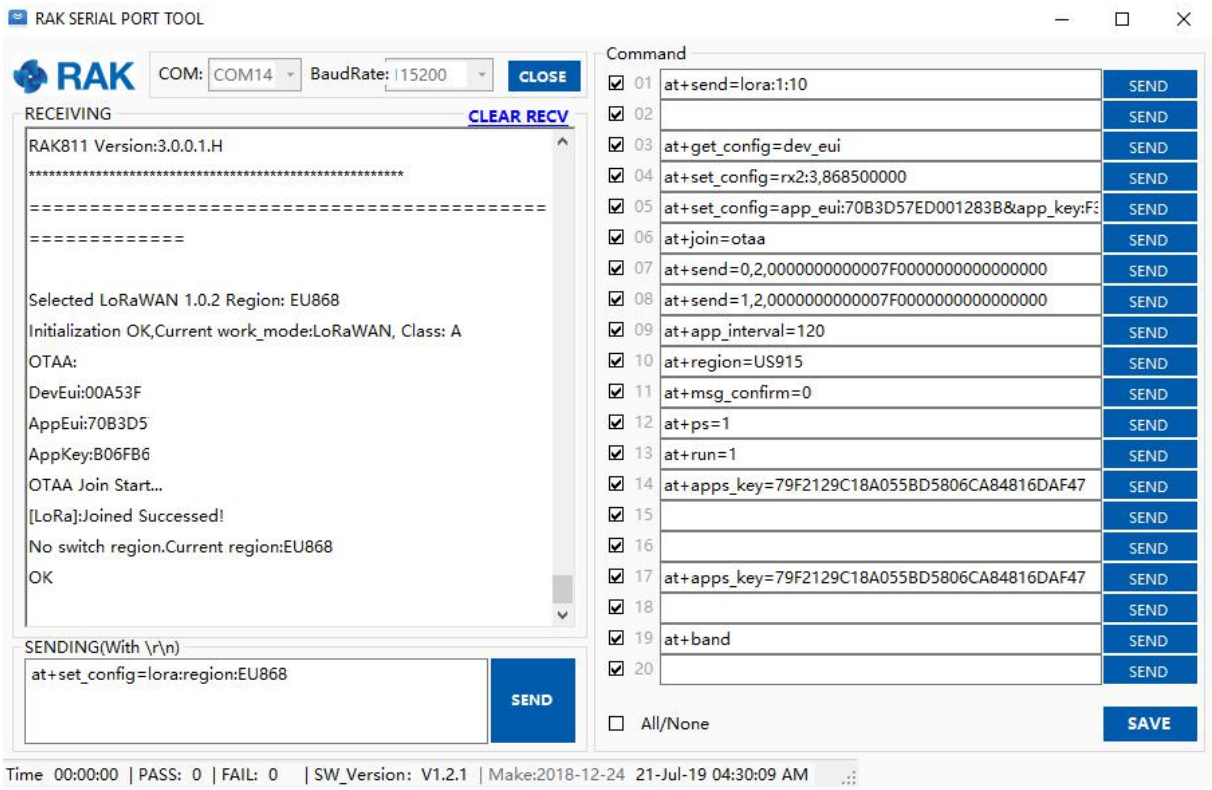

Figure 25 | Setting the region/band

Set the authentication mode to OTAA:

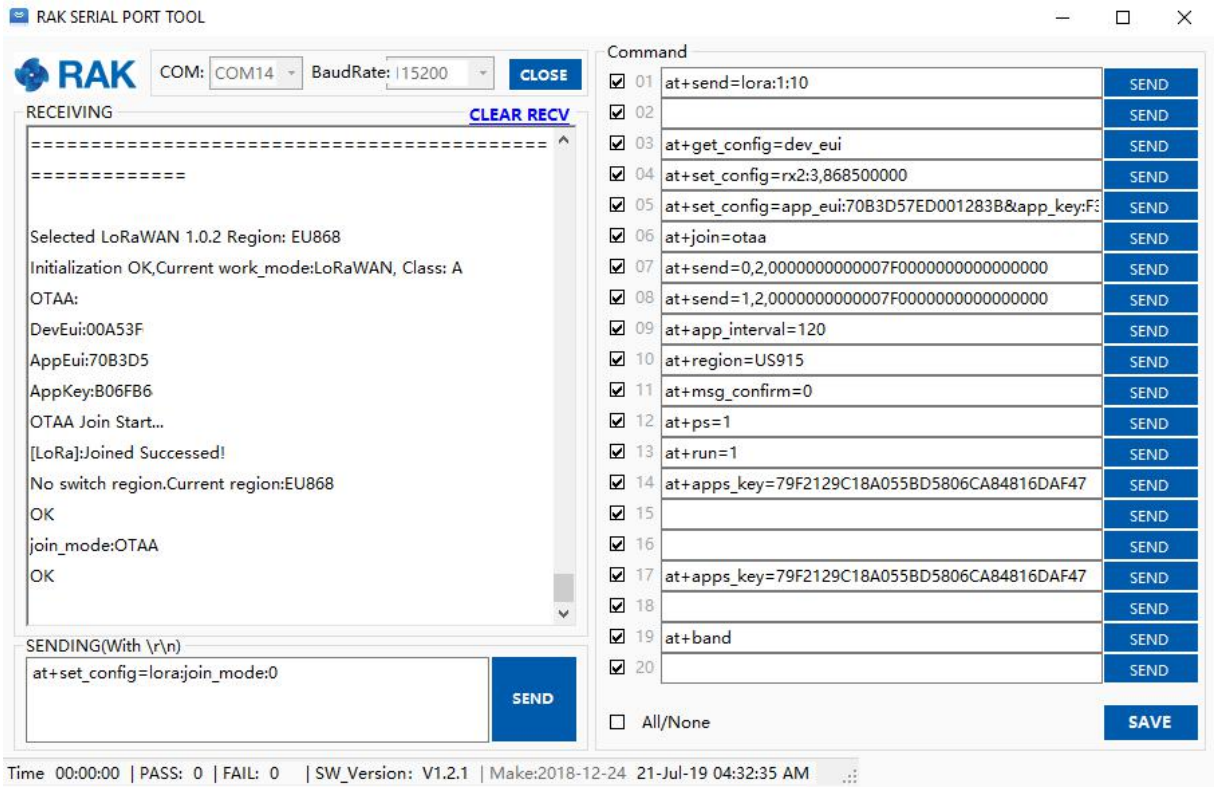

Figure 26 | OTAA mode

Now that your RAK811 is working in the correct region and mode you need to fill in the application parameters in your Gateway. This will register the specific device and allow you to exchange data.

Go to the *Application* tab. Enter a name for your application and press the *Add* button:

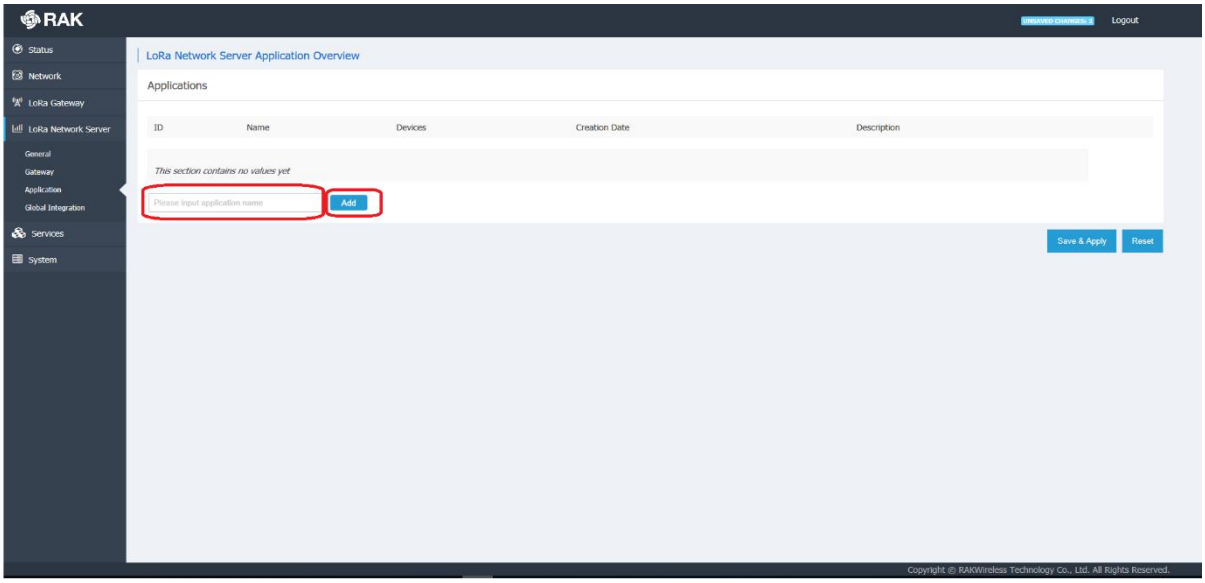

Figure 27 | Application registration

Now go back to your *Serial Tool* and copy the **Application EUI** and **Application Key** (check Figure X below):

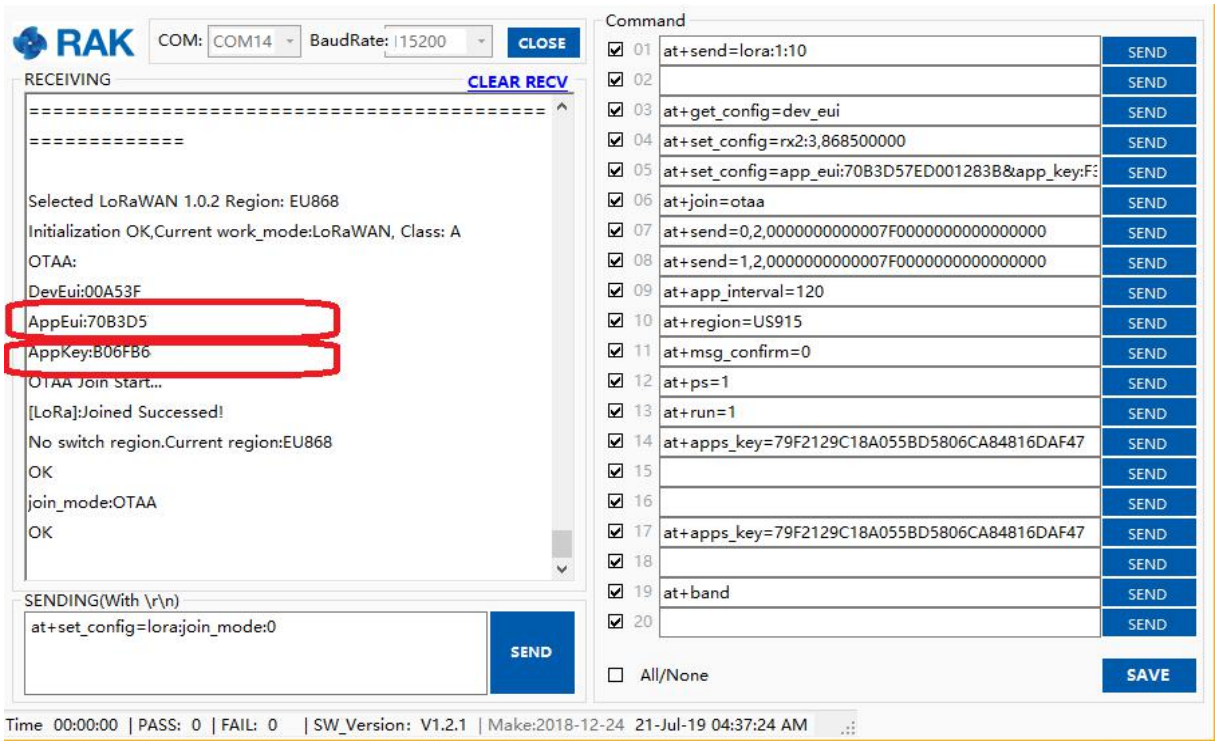

Figure 27 | Application EUI and Key

Input those into the corresponding fields in the *Application Configuration* screen in the Gateway:

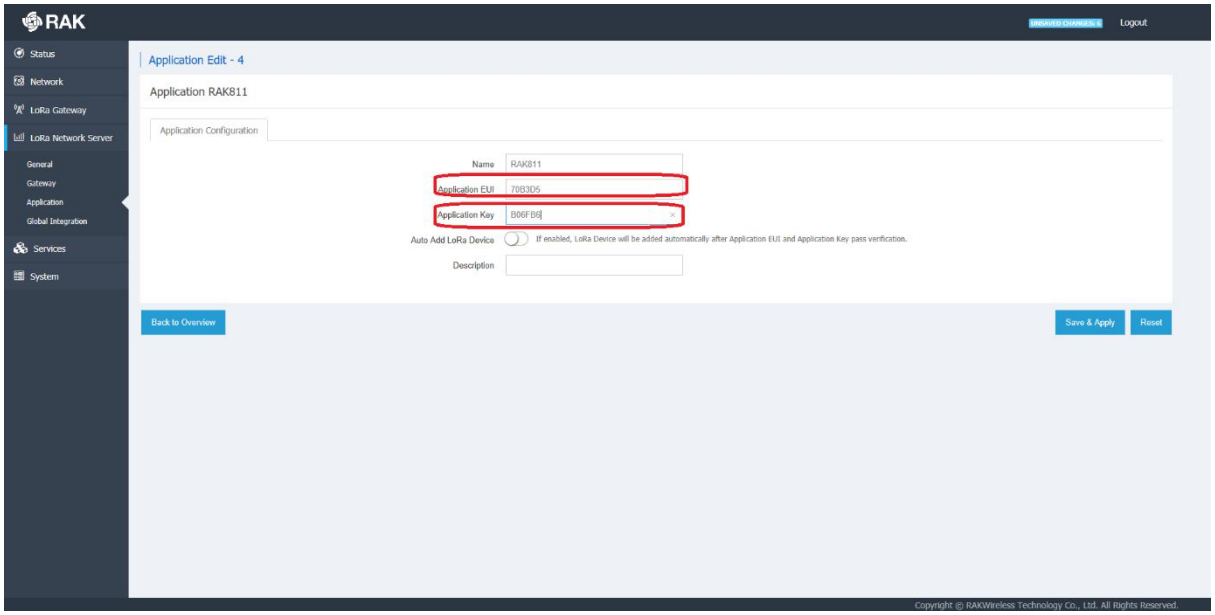

Figure 28 | Application parameters

Save & Apply (Make sure the Auto Add Device Slider is in the off position).

Now you should have an Application created. Press the *Edit* button:

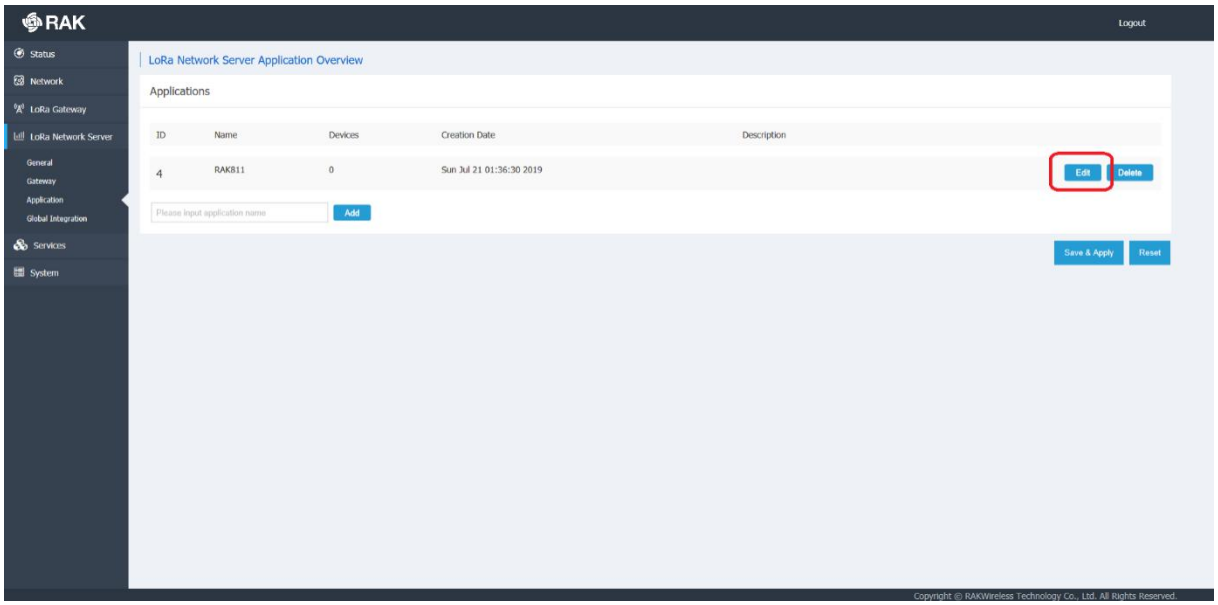

Figure 29 | Editing the Applicaiotn

Now you need to add a Device. Copy the Device EUI from the Serial Tool:

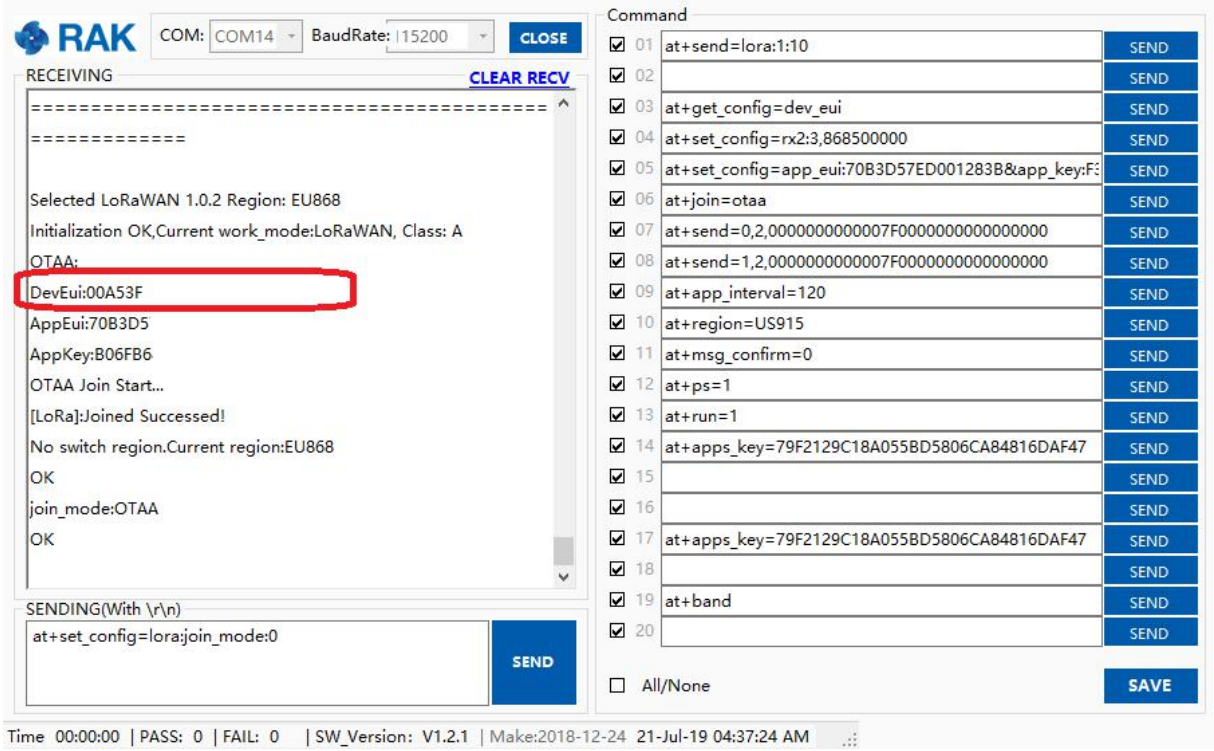

Figure 30 | Device EUI

Enter the *Device EUI* in the corresponding field and press the *Add* button (Figure 31)

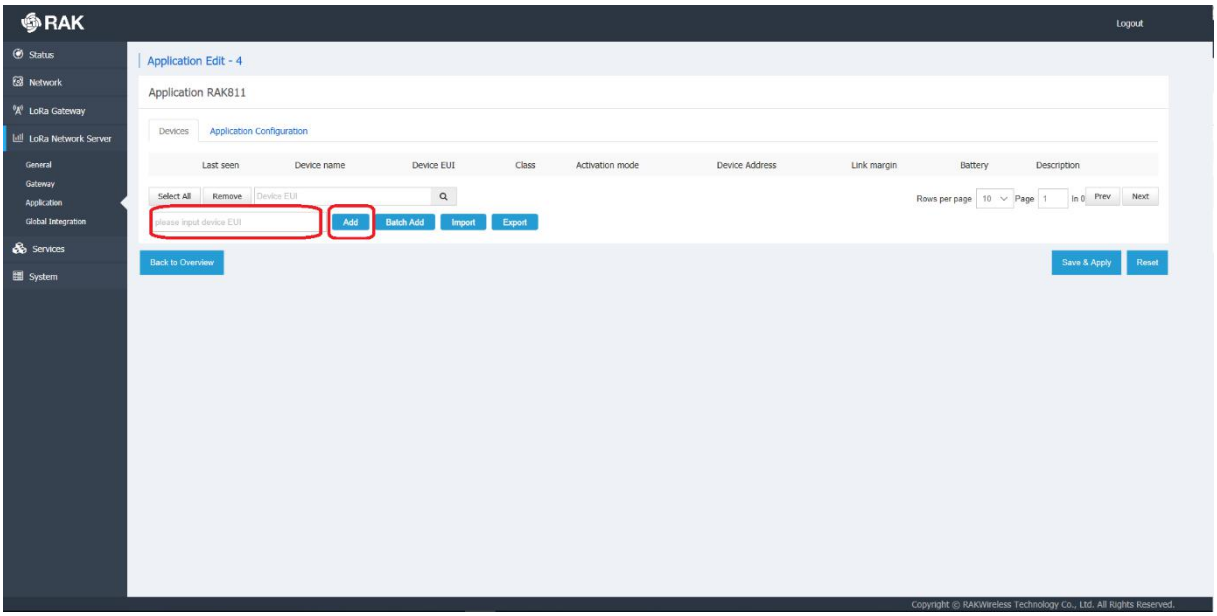

Figure 31 | Adding a Device

Enter a *Device name*, make sure you are in *Class A*, *OTAA mode*. Leave the rest of the parameters with their default settings. *Save & Apply*.

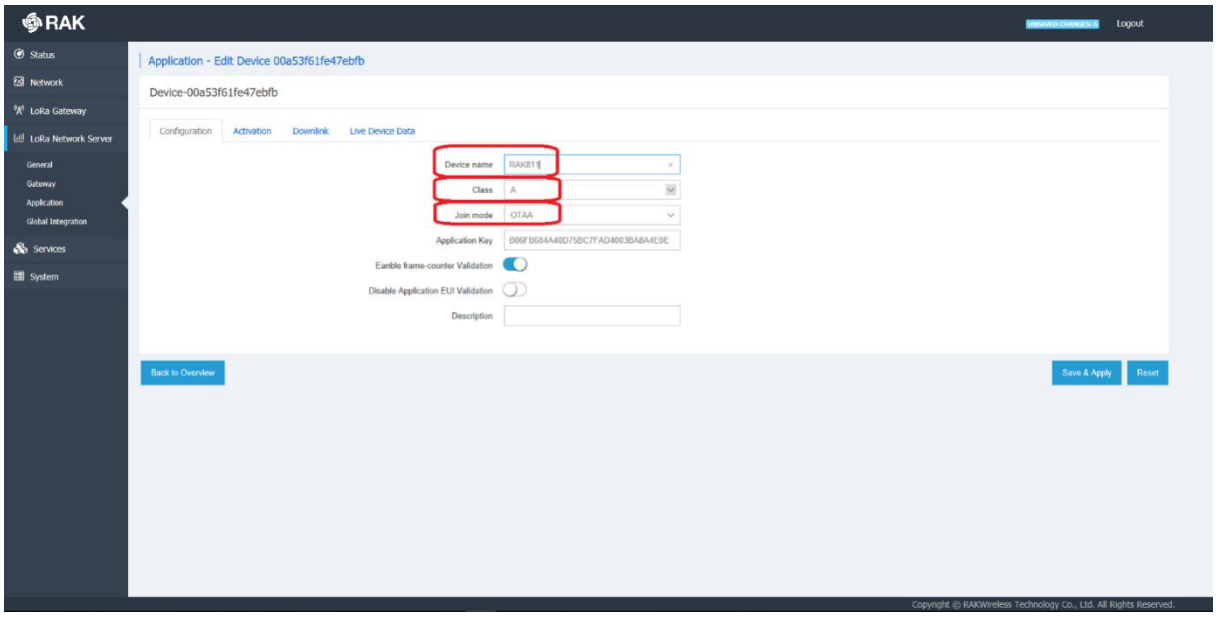

Figure 32 | Device parameters

You should now have your *Device* registered and if you click on the *Device EUI* you will open the corresponding *Device* window. Go to the *Live Device Data* tab. Here you can monitor data that the application is exchanging in real time.

Leave the *Live Device Data* tab open as we want to monitor traffic.

Go the *Serial Tool* and reboot the RAK811 with the onboard button.

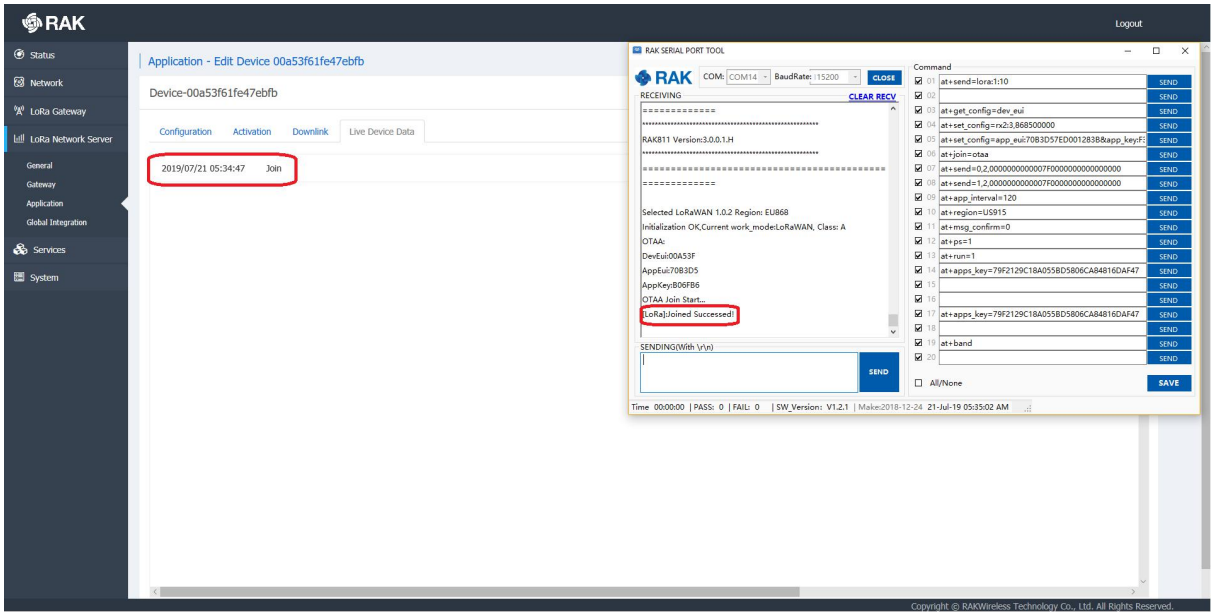

Figure 33 | Successful Joining of the RAK811

Figure 33 represents the output you should get if everything went well. You should see the *Join request* in the *Live Data* tab and the *Join Succeeded!* message in the *Serial Tool*.

3.4.5 Testing and monitoring the traffic

UPLINK

Now your node is authenticated with the built-in LoRa Server. As it is connected to the external MQTT Broker via the Global Integration you can monitor traffic in both the Live Data tab and on the Raspberry Pi where the Mosquitto resides. Let us test this by sending an uplink frame via the RAK811.

First in the command line window of the Raspberry we need to subscribe to the Application/Device we are going to monitor the traffic of. This is done via the following command:

mosquitto_sub -t application/{{application_ID}}/device/{{device_EUI}}/rx -v

{{application_ID}} – is the application ID from the Application tab in the Gateway

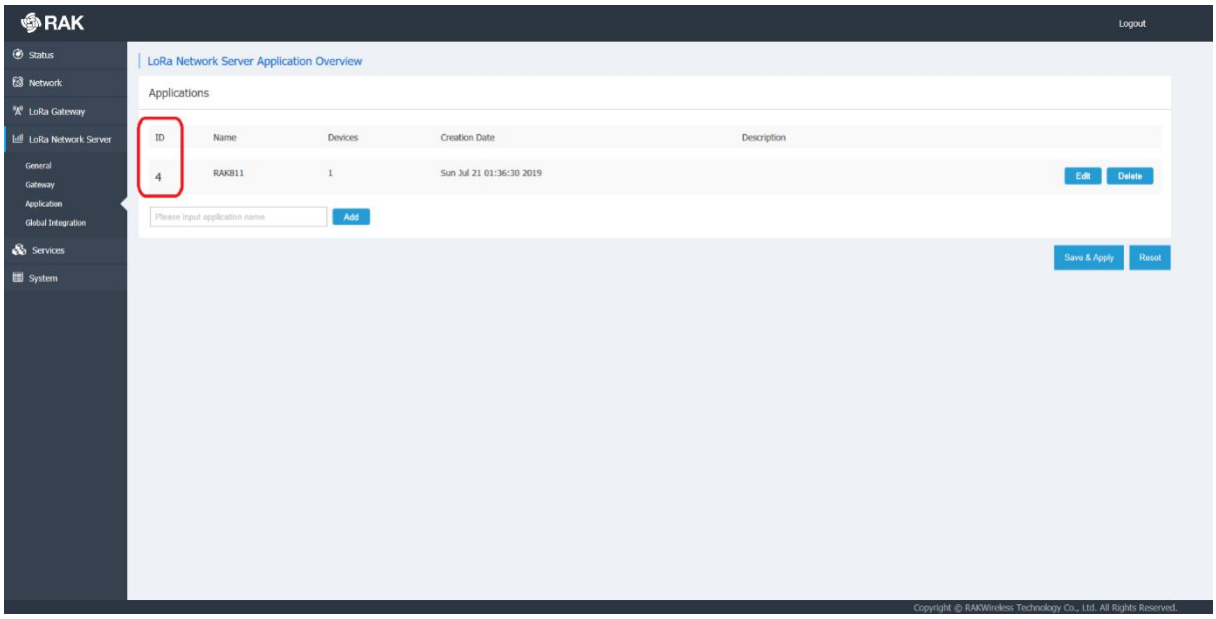

Figure 34 | Application ID

{{device_EUI}} – is the Device EUI of the RAK811

After executing the command you need to send some data via the Serial Tool.

Use the command below to send an uplink frame on *Frame port 1*, with the *Payload 1110*:

at+send=lora:1:1110

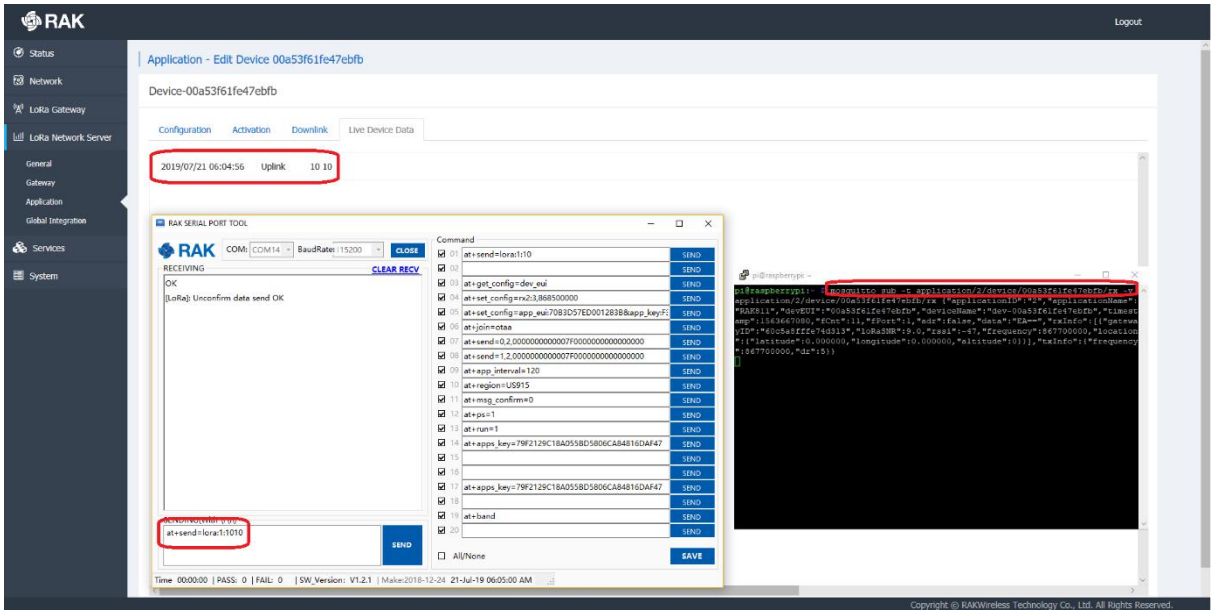

Figure 35 | Test uplink (Application)

Now if you look at the three windows in Figure 35 (Serial Tool, Live Device Data and the CLI of the Raspberry you will see that the message arriving is displayed.

Additionally you can monitor the Gateway traffic itself (all packets not just your application). You can do this via the command:

mosquitto_sub -t gateway/{{eui}}/rx -v

{{eui}} – is the Gateway EUI

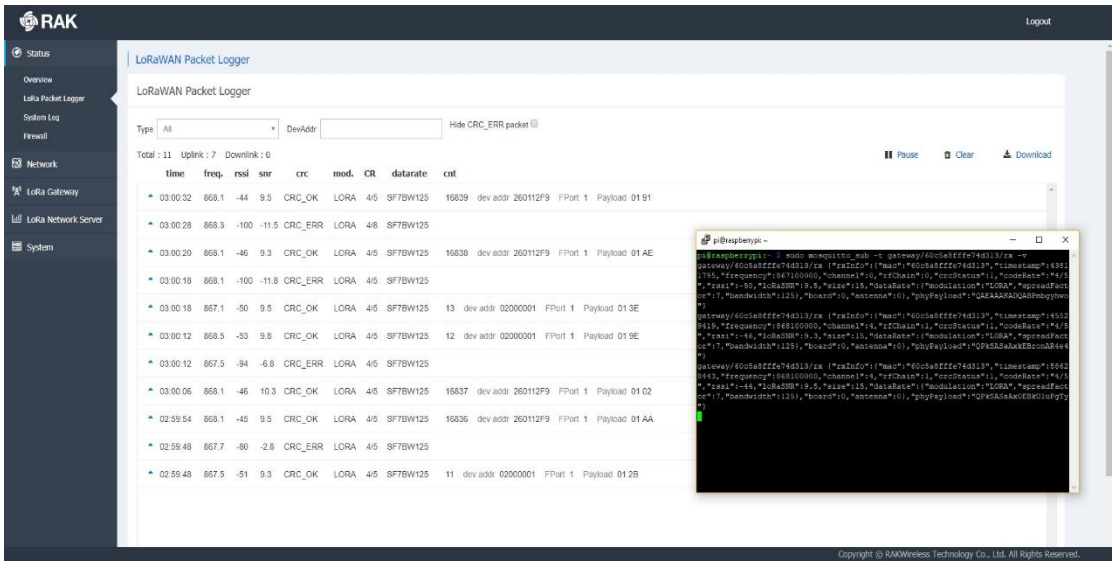

Figure 36 | Test uplink (Gateway)

This is very convenient as you have three ways to monitor and you can see the metadata and payload in both the Gateway and via the MQTT Broker.

DOWNLINK

There is a convenient tool in the Built-in LoRa Server for sending a *Downlink* frame. You can find it in the *Device* tab in the *LoRa Network Server* section. You can choose your *Type of frame* (confirmed/unconfirmed), the *Frame port* and the *Hex Data* (Refer to Figure 37).

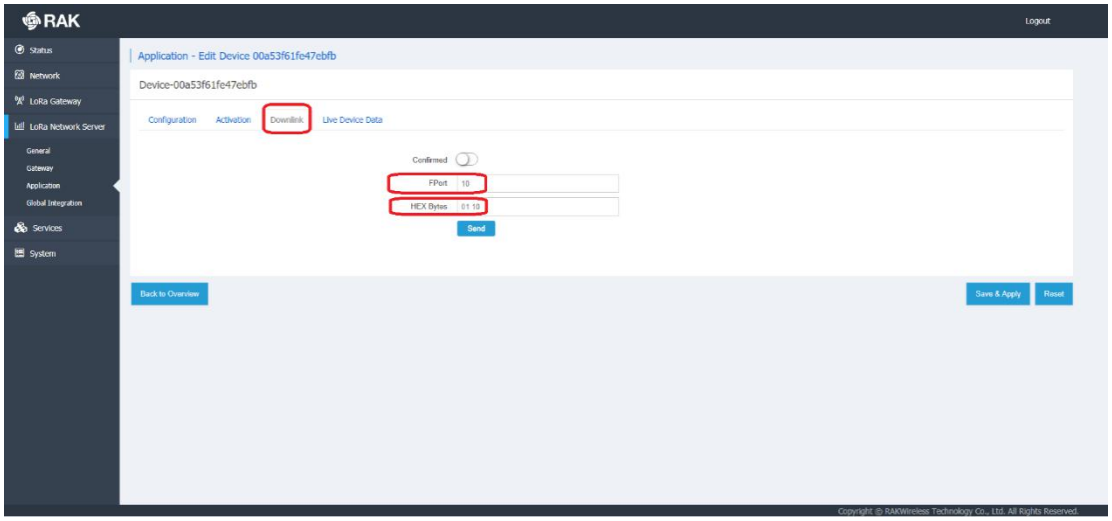

Figure 37 | LoRa Network Server Device Downlink tool

Once you schedule a message for downlink it will be displayed in the *Live Device Data* window. Upon sending the next uplink via the *Serial Tool* you will also see it there, as it needs an uplink frame in order to send the downlink in the *RX1 window* (refer to Figure 37).

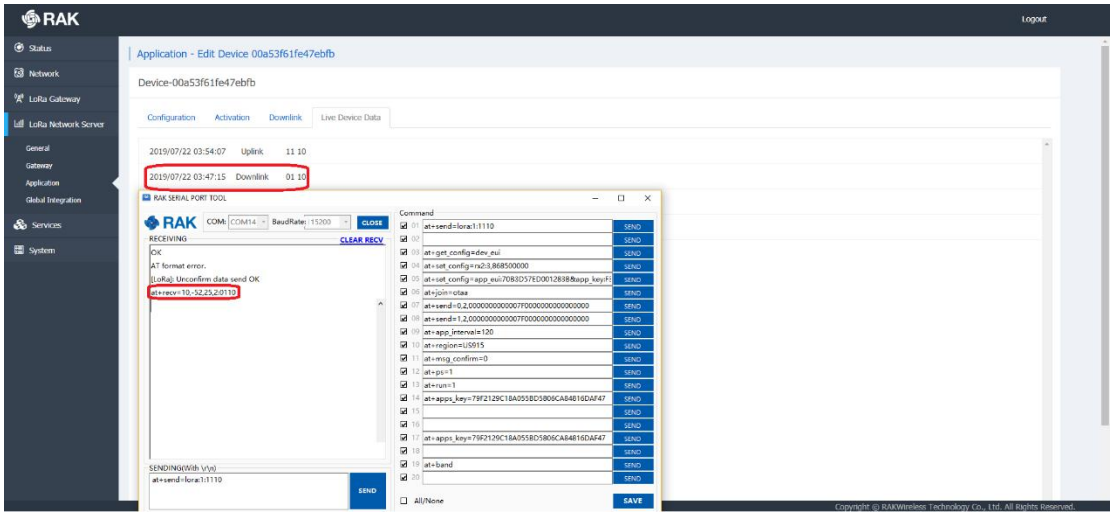

Figure 38 | Received Downlink Frame

Alternately if you want to test the Gateway downlink via the external MQTT Broker you need to first create a *json* file which you will be sensing your data in

Below is what the file formatting structure needs to look like:

```
{
"confirmed": true,
"fPort": 10,
"data": "1001"
}
```

"confirmed": true – This is the LoRa frame type. *True (confirmed)*, *False (unconfirmed)*

"fPort": 10 – the *Frame Port Number*

"data": "TEST" – example data to be sent

Note: You need to have a Base64 encoded HEX data for the above to work!

Create the file, for example with the following command and copy the data in discussed above:

sudo nano test.json

After you have your file created, you need to schedule it for downlink. This means you have to publish it via Mosquitto with the command:

sudo mosquito_pub application/{{application_ID{{/device/{{device_EUI}}/ tx –f test.json

The packet will be scheduler for downlink, which you can see in the Gateway Packet logger.

When the next uplink frame that comes for the Application/Device specified by the application_ID and device_EUI is received, the Gateway will send the data in the RX1 window to the node. You should have a response similar to the one in Figure 38.

4 Contact Information

Please contact us if you need technical support or want to know more information.

Support center: https://forum.rakwireless.com/

Email us: info@rakwireless.com

5 Revision History

Revision	Description	Date
1.0	Initial Release	2019-04-02

6 Document Summary

Prepared by	Checked by	Approved by
Vladislav	Yutao, Penn	