HT-M01

Mini LoRa Gateway

Based on Semtech® SX1308

Ultra-small size, USB, SPI, LoRaWAN protocol support

1. Product Overview

The core chip of the HT-M01 LoRa gateway is the SX1308 baseband chip produced by Semtech Corporation. It is industrial-grade standard LoRa/LoRaWAN gateway module which with high performance and small size. The HT-M01 series products currently have 433MHz, 470~510MHz, 868MHz and 915MHz and can be customized according to the user's needs. This LoRa Gateway is mainly aimed at wireless sensor networks, Internet of things, embedded applications, it can help customers quickly complete the development of LoRaWAN networking working with our HT-M01 development kits.

1.1 About Development Kits

The development kit is mainly composed of "Raspberry Pi ZERO W" and "Dedicated adapter board". The adapter board is mainly used for exporting the Raspberry Pi SPI bus. It can be directly used to drive the HT-M01 LoRa gateway, and the board contains CH340G USB to serial chip, user-friendly interaction with the Raspberry Pi.

1.2 Application area

- Internet of Things
- Meter reading concentrator
- Industrial Control Concentrator
- Security Alert System Gateway
- LPWAN base station

1.3 Product Features

- Small size of the adapter Raspberry pie ZERO: 66(+10) x 30 x 15 mm
- It can be integrated with Raspberry Pi through dedicated adapter board;
- Embedded 49 pcs LoRa demodulators;
- 10 pcs programmable parallel demodulation paths;
- Automatic adaptive spreading factor, optional for each channel SF7 to SF12;
- Maximum output power 25dBm
- Receiver sensitivity: -142.5dBm@300bps
- Communication Interface: SPI or USB
- 5V power supply;
- Support LoRaWAN Class A, Class B, Class C protocol

2. LoRa Gateway User Guide

The HT-M01 LoRa Gateway supports both USB and SPI modes of operation. The SPI mode can only be used on Linux systems and the USB mode can be used on Linux and Windows®. This document mainly describes how to use the lora gateway in the Raspberry Pi via SPI or USB. The operation of the Windows® part is updated in the following.

The operations involved in this article are mainly the process of setting up a private cloud server. The system framework is shown in Figure 2-1.

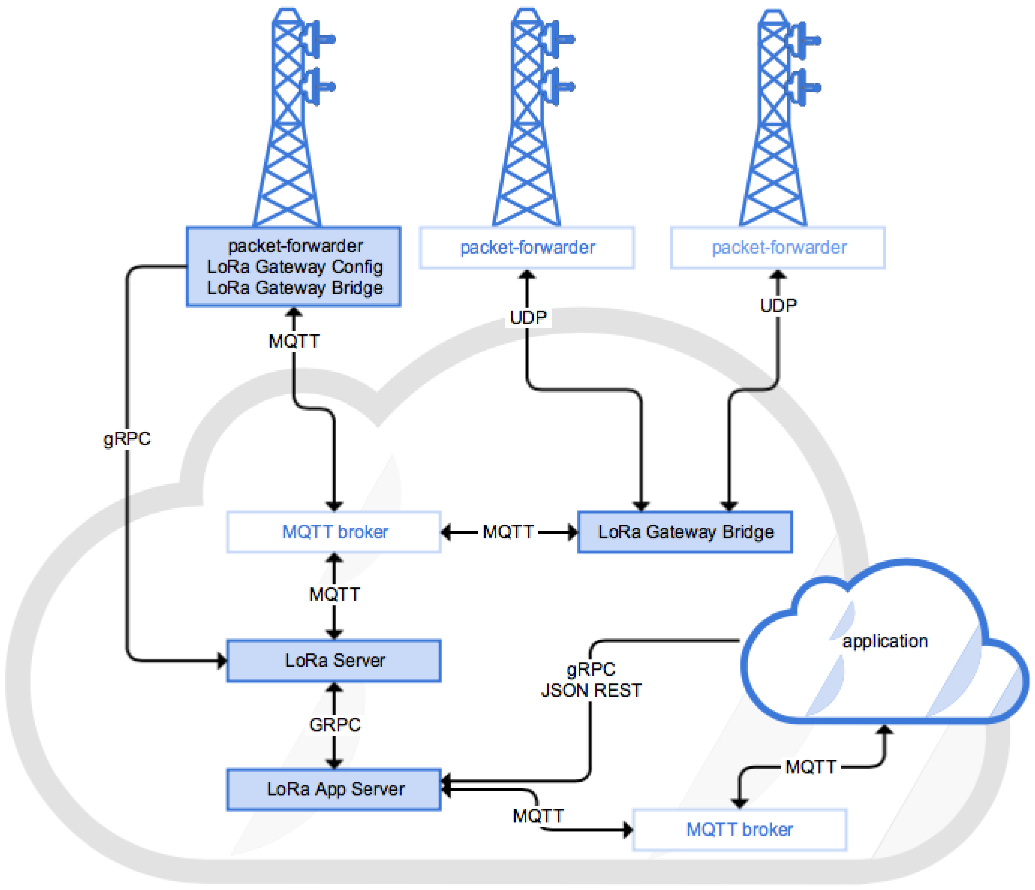


Fig. 2-1 Architecture

The sensors on the End Nodes are driven by the MCU. After reading the data, the data content is sent out through the LoRaWAN protocol. The gateway parses and receives the data sent by the nodes. Then the data Sent to the server through the Internet (UDP protocol). The server is composed of two parts: "web server" and "application server (hereinafter referred to as APP server)". The web server is responsible for summarizing and storing the data in the database that the server depends on. The APP server will read database of the web server and provide data interfaces such as http, MQTT, and so on, making it easy for computers, mobile phones, web pages and other applications to call this data and present it to the user.

Special Note

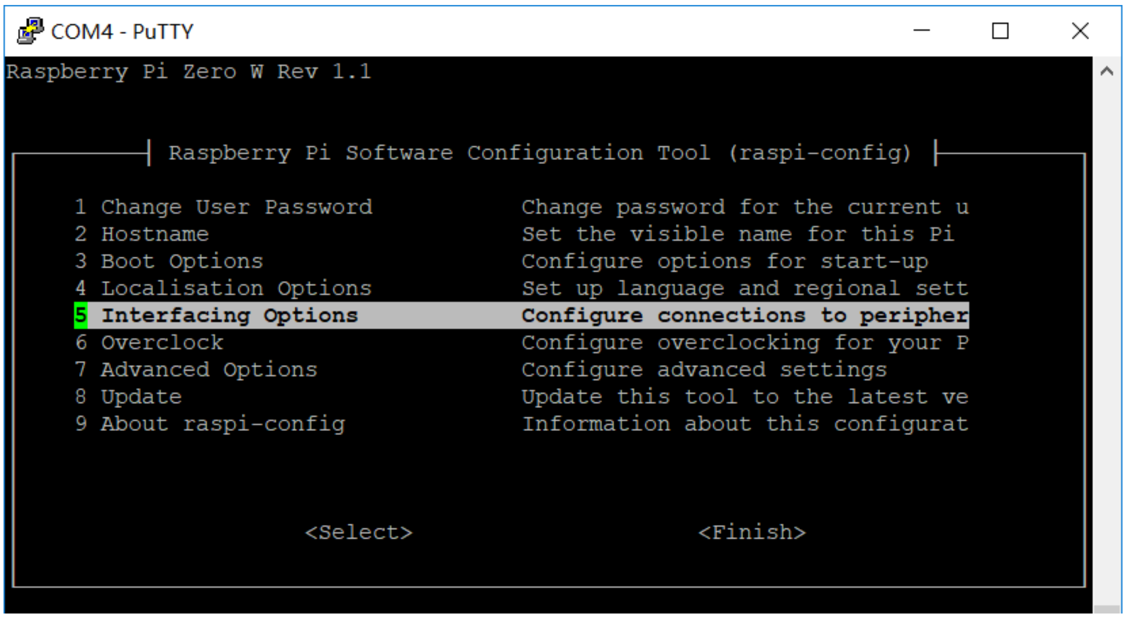
Before proceeding with the following operations and experiments, here are some instructions:

- Make sure your Raspberry Pi can work normally, and you can connect to Wi-Fi;
- This article involves more professional knowledge. We try to express some professional terms in a way that people can receive.
- The commands in this article can be copied and pasted directly. The "#" is followed by the comment of the command, no need to copy; each copy of a line, some longer commands occupy more than one line, when the attention is based on the background color.

2.1 Driving LoRa Gateway through SPI

The Raspberry Pi SPI bus is disabled by default. Enter the following command in putty to open the Raspberry Pi configuration interface (Figure 2-2) and enable the SPI bus:

```
sudo raspi-config
```



```
Raspberry Pi Zero W Rev 1.1

        ┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

   1 Change User Password          Change password for the current u
   2 Hostname                      Set the visible name for this Pi
   3 Boot Options                  Configure options for start-up
   4 Localisation Options          Set up language and regional sett
   5 Interfacing Options           Configure connections to peripher
   6 Overclock                     Configure overclocking for your P
   7 Advanced Options              Configure advanced settings
   8 Update                        Update this tool to the latest ve
   9 About raspi-config            Information about this configurat

                <Select>                     <Finish>
```

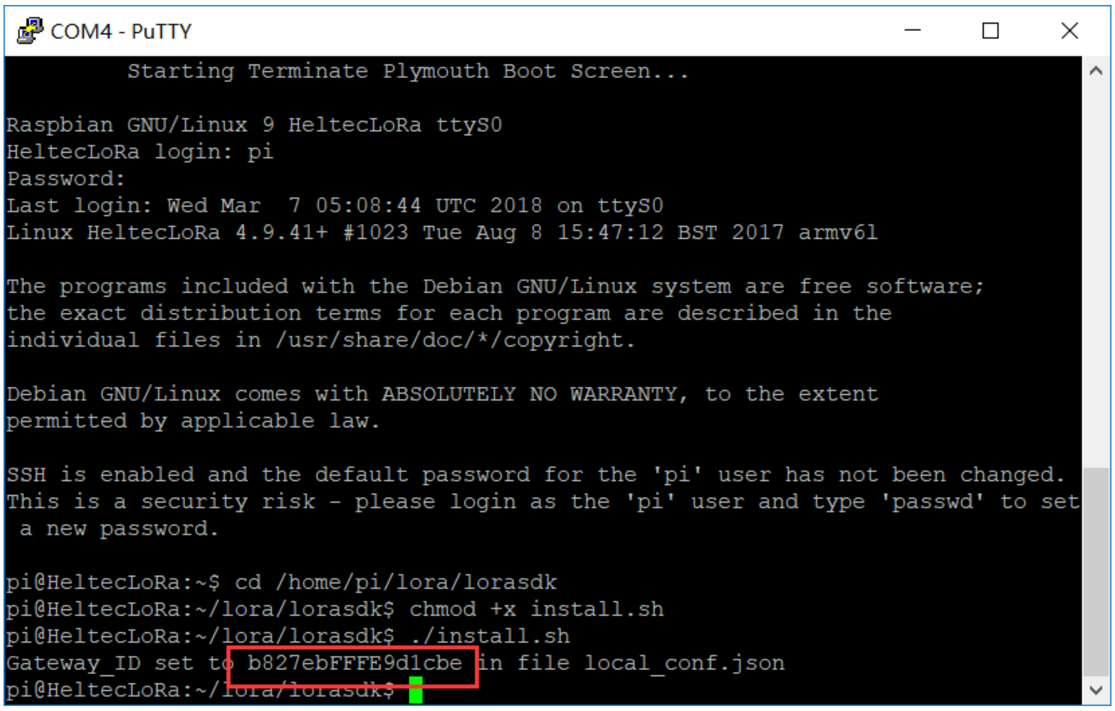
Figure 2-2 Raspberry Pi Command Configuration Interface

Interfacing Options → SPI → Yes

After exiting the configuration interface, enter the following commands in sequence:

```
mkdir lora
cd lora
sudo apt-get update
sudo apt-get install git
git clone https://github.com/Lora-net/lora_gateway.git  # lora Gateway Drivers
git clone https://github.com/Lora-net/packet_forwarder.git  # packet forwarding
software
git clone https://github.com/HelTecAutomation/lorasdk.git  # This is a script
package we made for our own product to implement the semi-automatic installation of
the raspberry pie end lora service
cd /home/pi/lora/lora_gateway
make clean all
cd /home/pi/lora/packet_forwarder
make clean all
cd /home/pi/lora/lorasdk
chmod +x install.sh
./install.sh   #Run the script. After the script is run, it will create a service
named "lrgateway". The purpose is to make the lora driver and data forwarding
program run automatically at startup.
sudo cp -
f /home/pi/lora/lorasdk/global_conf_CN470.json /home/pi/lora/packet_forwarder/lora_
pkt_fwd/global_conf.json
```

After executing the install.sh script, a 16-bit string will be generated based on your hardware. This is the ID of your gateway (Figure 2-3). Keep in mind that the subsequent operations need to be used.



```
        Starting Terminate Plymouth Boot Screen...

Raspbian GNU/Linux 9 HeltecLoRa ttyS0
HeltecLoRa login: pi
Password:
Last login: Wed Mar  7 05:08:44 UTC 2018 on ttyS0
Linux HeltecLoRa 4.9.41+ #1023 Tue Aug 8 15:47:12 BST 2017 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
 a new password.

pi@HeltecLoRa:~$ cd /home/pi/lora/lorasdk
pi@HeltecLoRa:~/lora/lorasdk$ chmod +x install.sh
pi@HeltecLoRa:~/lora/lorasdk$ ./install.sh
Gateway_ID set to b827ebFFFE9d1cbe in file local_conf.json
pi@HeltecLoRa:~/lora/lorasdk$
```

Figure 2-3 Gateway ID

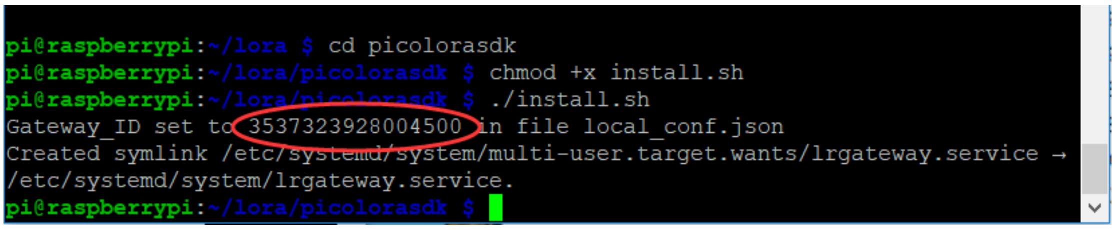
2.2 Driving the LoRa Gateway via USB

Before proceeding with the following operations, use a high-quality Micro USB cable to connect the HT-M01 Gateway to the Raspberry Pi, otherwise it will cause a lot of problems! After the check is correct, enter the following commands in order:

```
mkdir lora
cd lora
sudo apt-get update
sudo apt-get install git
git clone https://github.com/Lora-net/picoGW_hal.git
git clone https://github.com/Lora-net/picoGW_packet_forwarder.git
git clone https://github.com/HelTecAutomation/picolorasdk.git
cd /home/pi/lora/picoGW_hal
make clean all
cd /home/pi/lora/picoGW_packet_forwarder
make clean all
cd /home/pi/lora/picolorasdk
```

```
chmod +x install.sh
./install.sh  #Run the script. After the script is run, it will create a service named "lrgateway". The purpose is to make the lora driver and data forwarding program run automatically at startup.
sudo cp -f /home/pi/lora/picolorasdk/global_conf_CN470.json /home/pi/lora/picoGW_packet_forwarder/lora_pkt_fwd/global_conf.json  #Put the configuration file on the specified path
```

After executing the install.sh script, a 16-bit string will be generated based on your hardware. This is the ID of your gateway (Figure 2-4). Remember that the subsequent operations need to be used.



```
pi@raspberrypi:~/lora $ cd picolorasdk
pi@raspberrypi:~/lora/picolorasdk $ chmod +x install.sh
pi@raspberrypi:~/lora/picolorasdk $ ./install.sh
Gateway_ID set to 3537323928004500 in file local_conf.json
Created symlink /etc/systemd/system/multi-user.target.wants/lrgateway.service →
/etc/systemd/system/lrgateway.service.
pi@raspberrypi:~/lora/picolorasdk $
```

Figure 2-4 ID read out in USB mode

2.3 Installing LoRa-gateway-bridge on Raspberry Pi

依次输入如下命令:

```
# Firstly, we need add the address of lora-gateway-bridge to the Raspberry Pi software update source
export DISTRIB_ID=`lsb_release -si`
export DISTRIB_CODENAME=`lsb_release -sc`
sudo echo "deb https://repos.loraserver.io/${DISTRIB_ID,,} ${DISTRIB_CODENAME} testing" | sudo tee /etc/apt/sources.list.d/loraserver.list

sudo apt-get update
sudo apt-get install lora-gateway-bridge   # Install lora-gateway-bridge
sudo systemctl start lora-gateway-bridge   # start lora-gateway-bridge
sudo systemctl enable lora-gateway-bridge  #Set lora-gateway-bridge autorun
```

The function of “packet_forwarder” is to forward the received lora packet to a fixed IP address and port. The role of “LoRa-gateway-bridge” is to pack the data sent by “packet_forwarder” into json format, and upload to server.

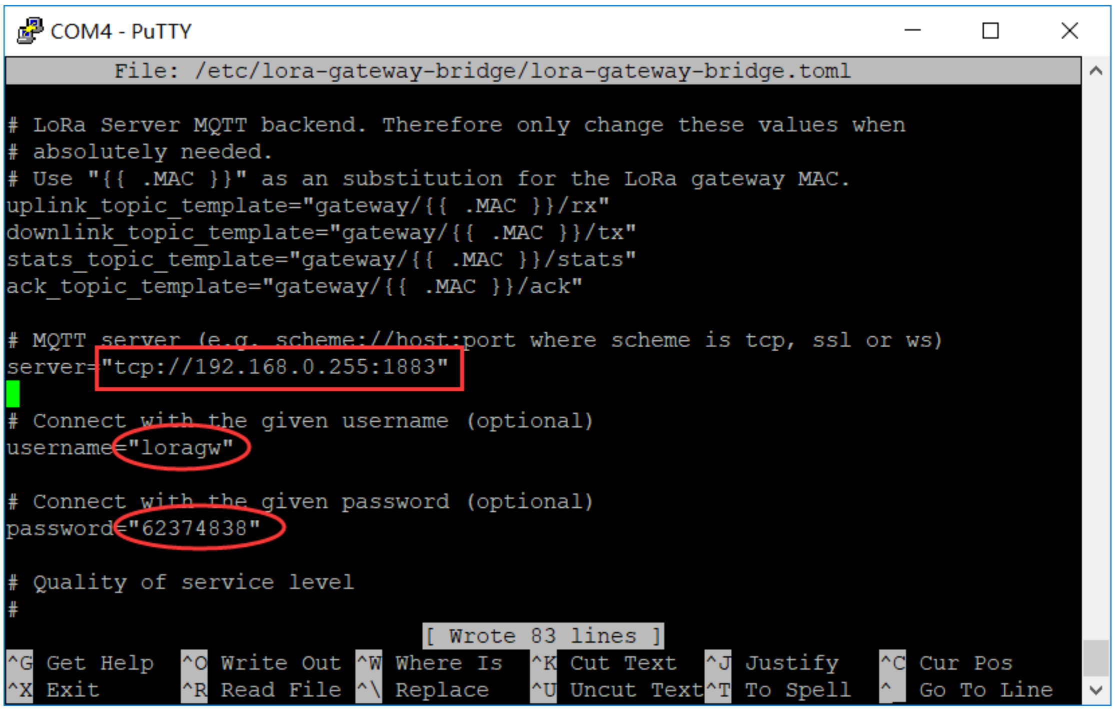
2.4 Configure Raspberry Pi

“Packet_forwarder” will send the received packet to “lora-gateway-bridge” which is installed in the raspberry pie on the machine, so the forwarding address of “packet_forwarder” is set to localhost, the port is 1700, “packet_forwarder” by reading the file of “global_conf. Json” to load the configuration information, we have already packaged some of the major regional configuration files for you in lorasdk (or picolorasdk), which contains the basic settings for each band, the address and port of the packet forwarding, etc., in front of the driver In the installation section, there have been orders to copy it to the proper path.

The configuration information of “lora-gateway-bridge” is implemented by loading the toml file. In the "/etc/lora-gateway-bridge/lora-gateway-bridge.toml" path, open it with the nano editor:

```
sudo nano /etc/lora-gateway-bridge/lora-gateway-bridge.toml
```

As shown in Figure 2-5, the IP address in the red box should be changed to the address of the lora server, port 1883 remains unchanged, username and password are filled in the user created for lora-gateway-bridge (this part of the username and password Create, explained in detail later in the "mosquito configuration" section).



```
        File: /etc/lora-gateway-bridge/lora-gateway-bridge.toml

# LoRa Server MQTT backend. Therefore only change these values when
# absolutely needed.
# Use "{{ .MAC }}" as an substitution for the LoRa gateway MAC.
uplink_topic_template="gateway/{{ .MAC }}/rx"
downlink_topic_template="gateway/{{ .MAC }}/tx"
stats_topic_template="gateway/{{ .MAC }}/stats"
ack_topic_template="gateway/{{ .MAC }}/ack"

# MQTT server (e.g. scheme://host:port where scheme is tcp, ssl or ws)
server="tcp://192.168.0.255:1883"

# Connect with the given username (optional)
username="loragw"

# Connect with the given password (optional)
password="62374838"

# Quality of service level
#
                              [ Wrote 83 lines ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^  Go To Line
```

Figure 2-5 The lora-gateway-bridge configuration file

Now, the configuration of the LoRa gateway part is complete. The latter operation will be performed on the Linux server.

3. Server-side operations

The server-side operations covered in this article are performed in a VMware virtual machine with the operating system Ubuntu 16.0.4.

3.1 Installation Relative Tool Chain

The following software is required for the Lora server:

- **MQTT broker** -- A publish/subscribe protocol that allows users to publish information under topics that others can subscribe to. A popular implementation of the MQTT protocol is Mosquitto.
- **Redis** - A database used to store relatively transient data.

- **PostgreSQL** - The long-term storage database used by the open source packages.

In the terminal of Ubuntu, enter the following command and press Y to confirm it will be automatically installed. The network problem may be wrong in the middle. The "ctrl + c" key combination ends the current operation and the newest command is executed.

```
sudo apt update
sudo apt install mosquitto mosquitto-clients redis-server redis-tools postgresql
```

3.2 Configuration Authentication and Mosquitto

Mosquitto, as the main conduit for messaging between the gateways and the LoRa servers and the applications receiving LoRa data, should be secured to prevent third party access to the data. To set up Mosquitto security:

```
sudo mosquitto_passwd -c /etc/mosquitto/pwd loraroot  # Create a root user. After entering this command, you will be allowed to set and confirm a password. In this experiment, the passwords related to mosquitto are all 62374838.
sudo mosquitto_passwd /etc/mosquitto/pwd loragw  # Create a user named “loragw” for use with lora-gateway-bridge
sudo mosquitto_passwd /etc/mosquitto/pwd loraserver  # This user is used by "loraserver"
sudo mosquitto_passwd /etc/mosquitto/pwd loraappserver  # This user uses "lora-app-server"
sudo chmod 600 /etc/mosquitto/pwd  # Pwd file encryption
```

After setting the user name and password, you need to tell “Mosquitto” where these user password configuration files are located.

```
sudo nano /etc/mosquitto/conf.d/local.conf  # Open the local.conf file with the nano editor, add the following content to it and save and exit;

allow_anonymous false
password_file /etc/mosquitto/pwd
```

After saving this configuration, restart Mosquitto with the new settings:

```
sudo systemctl restart mosquitto
```

3.3 Configure PostgreSQL Database

```
sudo -u postgres psql  # Enter the postgres database command line mode

# set up the users and the passwords (note that it is important to use single
quotes and a semicolon at the end!)
create role loraserver_as with login password 'dbpassword';
create role loraserver_ns with login password 'dbpassword';

# create the database for the servers
create database loraserver_as with owner loraserver_as;
create database loraserver_ns with owner loraserver_ns;

\q  #exit PostgreSQL
```

3.4 Installing LoRa Server

```
#Add the loraserver related package to Ubuntu's apt library
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 1CE2AFD36DBCCA00
sudo echo "deb https://repos.loraserver.io/ubuntu xenial testing" | sudo tee /etc/a
pt/sources.list.d/loraserver.list

sudo apt update
sudo apt install loraserver  #Install loraserver
```

The installation process generally does not have any special problems. After the installation is complete, the loraserver needs to be configured. Its configuration file is in the /etc/loraserver/loraserver.toml path. After opening it with the nano editor, there are many parameters. With the configuration options, you can also explore its features in detail through the comments in the file.

This article explains only a few places that must be modified. Others can be kept as default:

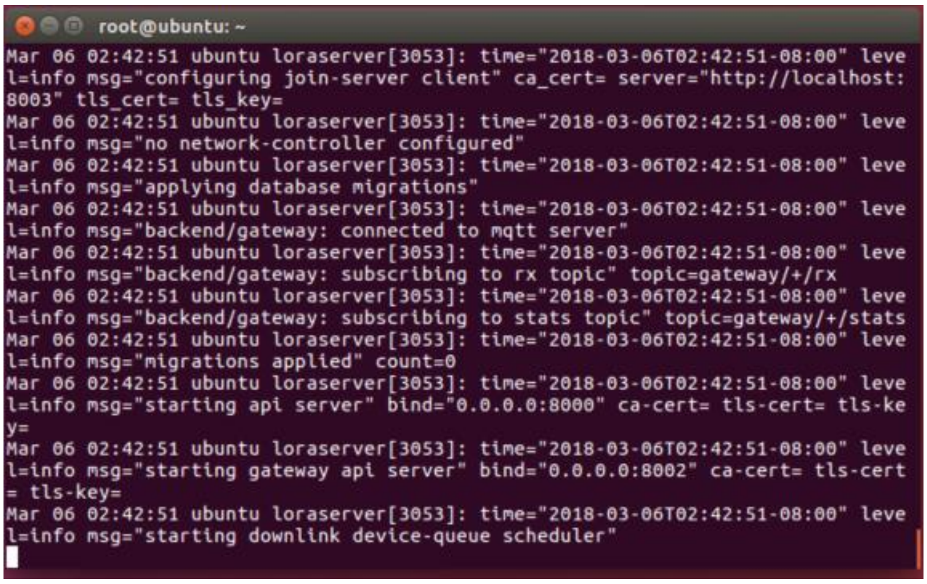
```
postgresql.dsn -- The URL to the postgres database, should look like the following:
dsn="postgres://loraserver_ns:dbpassword@localhost/loraserver_ns?sslmode=disable"
# Be careful with this setting. It is easy to get wrong, and can produce a number
of different error messages.

[network_server.band.name The ISM band to use. E.g.
name="CN_470_510"

network_server.gateway.backend.mqtt.username and
network_server.gateway.backend.mqtt.password, since the MQTT server is publicly
accessible (so LoRa Gateway Bridge instances can send data), it is best to have a
username and password for the server here.
username="loraserver"
password="62374838"
#JWT secret … fill in the encrypted information here, must
jwt_secret="openssl rand -base64 32"
```

In case of an error, we provide a complete "loraserver.toml" file (click to view), which is convenient for users to compare in experiments. At this point, the loraserver program and configuration is complete, open the loraserver service:

```
sudo systemctl start loraserver
```

Verify the operation of loraserver with the following command. If it is running correctly, you will see something similar to Figure 3-1:

```
journalctl –u loraserver -f
```



```
root@ubuntu: ~
Mar 06 02:42:51 ubuntu loraserver[3053]: time="2018-03-06T02:42:51-08:00" leve
l=info msg="configuring join-server client" ca_cert= server="http://localhost:
8003" tls_cert= tls_key=
Mar 06 02:42:51 ubuntu loraserver[3053]: time="2018-03-06T02:42:51-08:00" leve
l=info msg="no network-controller configured"
Mar 06 02:42:51 ubuntu loraserver[3053]: time="2018-03-06T02:42:51-08:00" leve
l=info msg="applying database migrations"
Mar 06 02:42:51 ubuntu loraserver[3053]: time="2018-03-06T02:42:51-08:00" leve
l=info msg="backend/gateway: connected to mqtt server"
Mar 06 02:42:51 ubuntu loraserver[3053]: time="2018-03-06T02:42:51-08:00" leve
l=info msg="backend/gateway: subscribing to rx topic" topic=gateway/+/rx
Mar 06 02:42:51 ubuntu loraserver[3053]: time="2018-03-06T02:42:51-08:00" leve
l=info msg="backend/gateway: subscribing to stats topic" topic=gateway/+/stats
Mar 06 02:42:51 ubuntu loraserver[3053]: time="2018-03-06T02:42:51-08:00" leve
l=info msg="migrations applied" count=0
Mar 06 02:42:51 ubuntu loraserver[3053]: time="2018-03-06T02:42:51-08:00" leve
l=info msg="starting api server" bind="0.0.0.0:8000" ca-cert= tls-cert= tls-ke
y=
Mar 06 02:42:51 ubuntu loraserver[3053]: time="2018-03-06T02:42:51-08:00" leve
l=info msg="starting gateway api server" bind="0.0.0.0:8002" ca-cert= tls-cert
= tls-key=
Mar 06 02:42:51 ubuntu loraserver[3053]: time="2018-03-06T02:42:51-08:00" leve
l=info msg="starting downlink device-queue scheduler"
```

Figure 3-1 Viewing log of loraserver

3.5 Installing the LoRa App Server

```
sudo apt install lora-app-server
```

Similarly, the lora app server needs to be configured. The configuration file is in the path of "/etc/lora-app-server/lora-app-server.toml". After opening it with the nano editor, there are many configuration options for the parameters. You can also explore its features in detail by using the comments in this file. This article explains only a few places that must be modified. Others can be kept as default:

```
postgresql.dsn -- The URL to the postgres database, should look like the following:
dsn="postgres://loraserver_as:dbpassword@localhost/loraserver_as?sslmode=disable"

application_server.integration.mqtt.username and
application_server.integration.mqtt. password since the MQTT server is publicly
accessible (so LoRa Gateway Bridge instances can send data), it is best to have a
username and password for the server here.
username="loraappserver"
password="62374838"
```

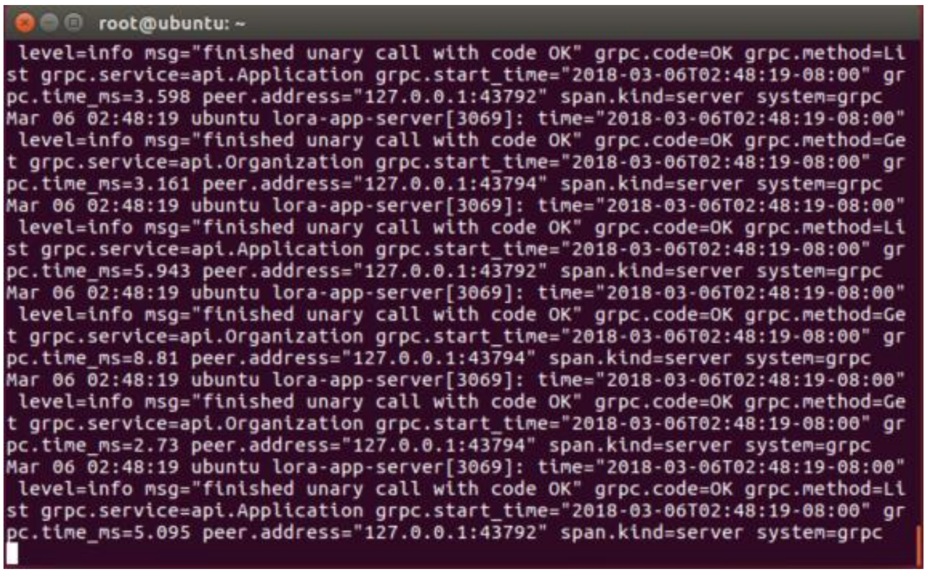
```
application_server.external_api.bind -- The port that serves up the api server.
This should be localhost:8001 as LoRa Server is on the same system.
bind="localhost:8080"

#JWT secret …encrypted information here
jwt_secret="openssl rand -base64 32"
```

In case of an error, we provide a complete "lora-app-server.toml" file (click to view), which is convenient for users to compare in experiments. At this point, the program and configuration of lora-app-server are completed, and the service of lora-app-server is started:

```
sudo systemctl start lora-app-server
```

Use the following command to verify the operating status of lora-app-server. If it is running correctly, something similar to Figure 3-2 will be displayed:

```
journalctl –u lora-app-server -f
```



```
root@ubuntu: ~
 level=info msg="finished unary call with code OK" grpc.code=OK grpc.method=Li
st grpc.service=api.Application grpc.start_time="2018-03-06T02:48:19-08:00" gr
pc.time_ms=3.598 peer.address="127.0.0.1:43792" span.kind=server system=grpc
Mar 06 02:48:19 ubuntu lora-app-server[3069]: time="2018-03-06T02:48:19-08:00"
 level=info msg="finished unary call with code OK" grpc.code=OK grpc.method=Ge
t grpc.service=api.Organization grpc.start_time="2018-03-06T02:48:19-08:00" gr
pc.time_ms=3.161 peer.address="127.0.0.1:43794" span.kind=server system=grpc
Mar 06 02:48:19 ubuntu lora-app-server[3069]: time="2018-03-06T02:48:19-08:00"
 level=info msg="finished unary call with code OK" grpc.code=OK grpc.method=Li
st grpc.service=api.Application grpc.start_time="2018-03-06T02:48:19-08:00" gr
pc.time_ms=5.943 peer.address="127.0.0.1:43792" span.kind=server system=grpc
Mar 06 02:48:19 ubuntu lora-app-server[3069]: time="2018-03-06T02:48:19-08:00"
 level=info msg="finished unary call with code OK" grpc.code=OK grpc.method=Ge
t grpc.service=api.Organization grpc.start_time="2018-03-06T02:48:19-08:00" gr
pc.time_ms=8.81 peer.address="127.0.0.1:43794" span.kind=server system=grpc
Mar 06 02:48:19 ubuntu lora-app-server[3069]: time="2018-03-06T02:48:19-08:00"
 level=info msg="finished unary call with code OK" grpc.code=OK grpc.method=Ge
t grpc.service=api.Organization grpc.start_time="2018-03-06T02:48:19-08:00" gr
pc.time_ms=2.73 peer.address="127.0.0.1:43794" span.kind=server system=grpc
Mar 06 02:48:19 ubuntu lora-app-server[3069]: time="2018-03-06T02:48:19-08:00"
 level=info msg="finished unary call with code OK" grpc.code=OK grpc.method=Li
st grpc.service=api.Application grpc.start_time="2018-03-06T02:48:19-08:00" gr
pc.time_ms=5.095 peer.address="127.0.0.1:43792" span.kind=server system=grpc
```

Figure 3-2 Viewing the logs of lora-app-server

4. Communication experiment

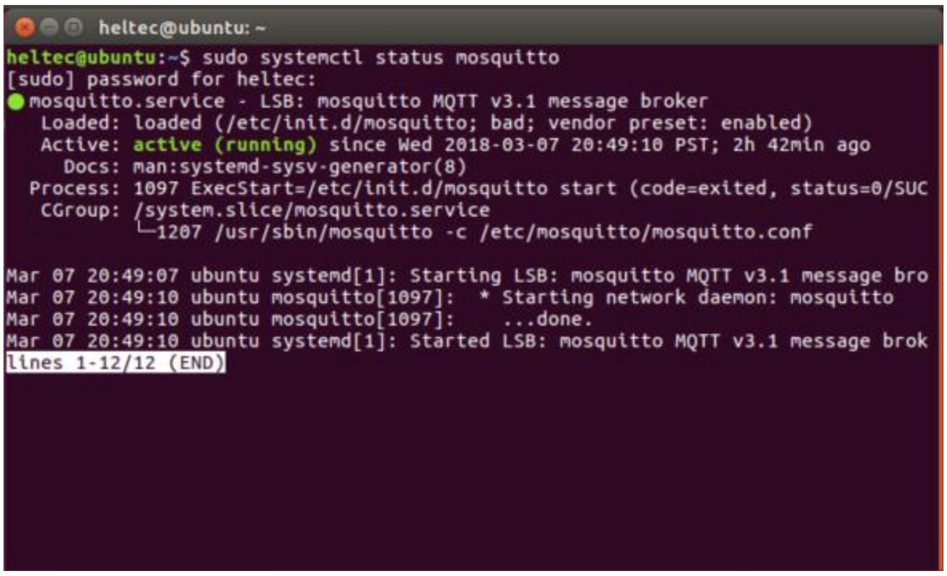
In order to realize the mutual transmission of data between the server and the loRa gateway/node, it is necessary to connect the lora gateway and the lora server through an application. This can be done in the browser's UI. Before starting the experiment, make sure the following key points:

➢ **The server and lora gateway services are all running properly. Use commands to check:**

The server side in turn enters the following command:

```
sudo systemctl status mosquitto         #Check if mosquitto is running
sudo systemctl status loraserver        #Check if loricaserver is running
sudo systemctl status lora-app-server    # Check if lora-app-server is running
```

After executing the above commands in turn, the interfaces that you see are similar to Figure 4-1. There is a green “active (running)” to prove that they are working properly.



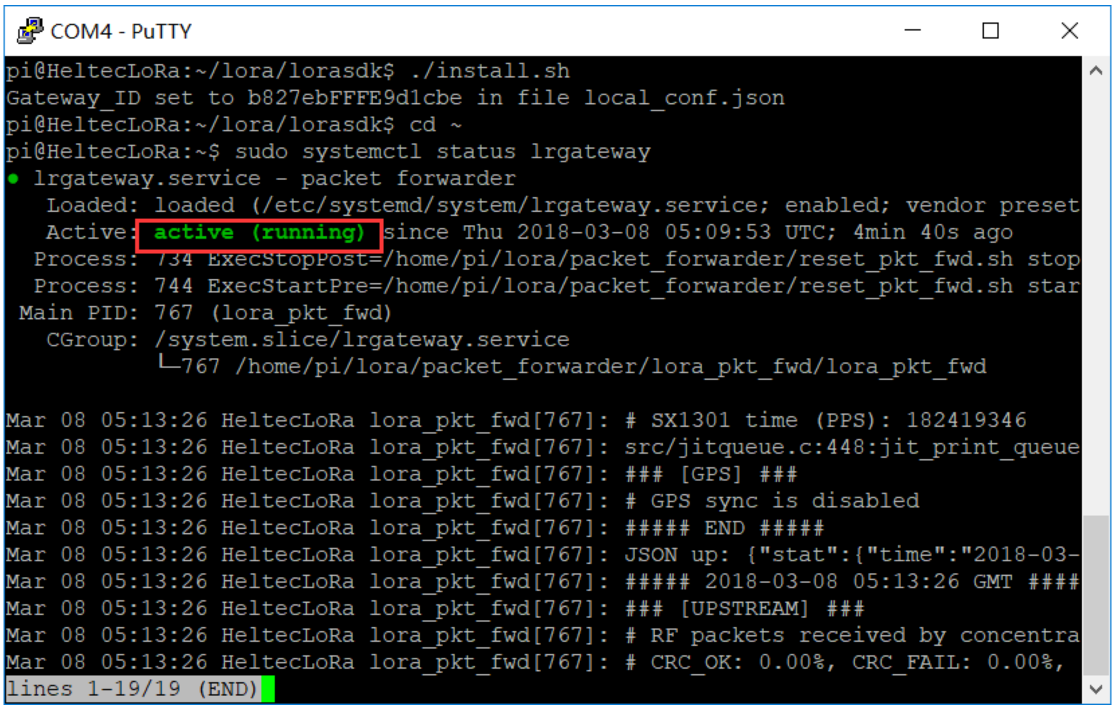
```
heltec@ubuntu:~$ sudo systemctl status mosquitto
[sudo] password for heltec:
● mosquitto.service - LSB: mosquitto MQTT v3.1 message broker
   Loaded: loaded (/etc/init.d/mosquitto; bad; vendor preset: enabled)
   Active: active (running) since Wed 2018-03-07 20:49:10 PST; 2h 42min ago
     Docs: man:systemd-sysv-generator(8)
  Process: 1097 ExecStart=/etc/init.d/mosquitto start (code=exited, status=0/SUC
   CGroup: /system.slice/mosquitto.service
           └─1207 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

Mar 07 20:49:07 ubuntu systemd[1]: Starting LSB: mosquitto MQTT v3.1 message bro
Mar 07 20:49:10 ubuntu mosquitto[1097]:  * Starting network daemon: mosquitto
Mar 07 20:49:10 ubuntu mosquitto[1097]:    ...done.
Mar 07 20:49:10 ubuntu systemd[1]: Started LSB: mosquitto MQTT v3.1 message brok
lines 1-12/12 (END)
```

Figure 4-1 Checking Service Health Status

Raspberry PI use following command：

```
sudo systemctl status lrgateway              # Check if mosquitto is running
sudo systemctl status lora-gateway-bridge    # Check if lora-gateway-bridge is
running
```

After executing the above commands in turn, the interfaces that you see are similar to those in Figure 4-2. There is a green “active (running)” to prove that they are working properly.



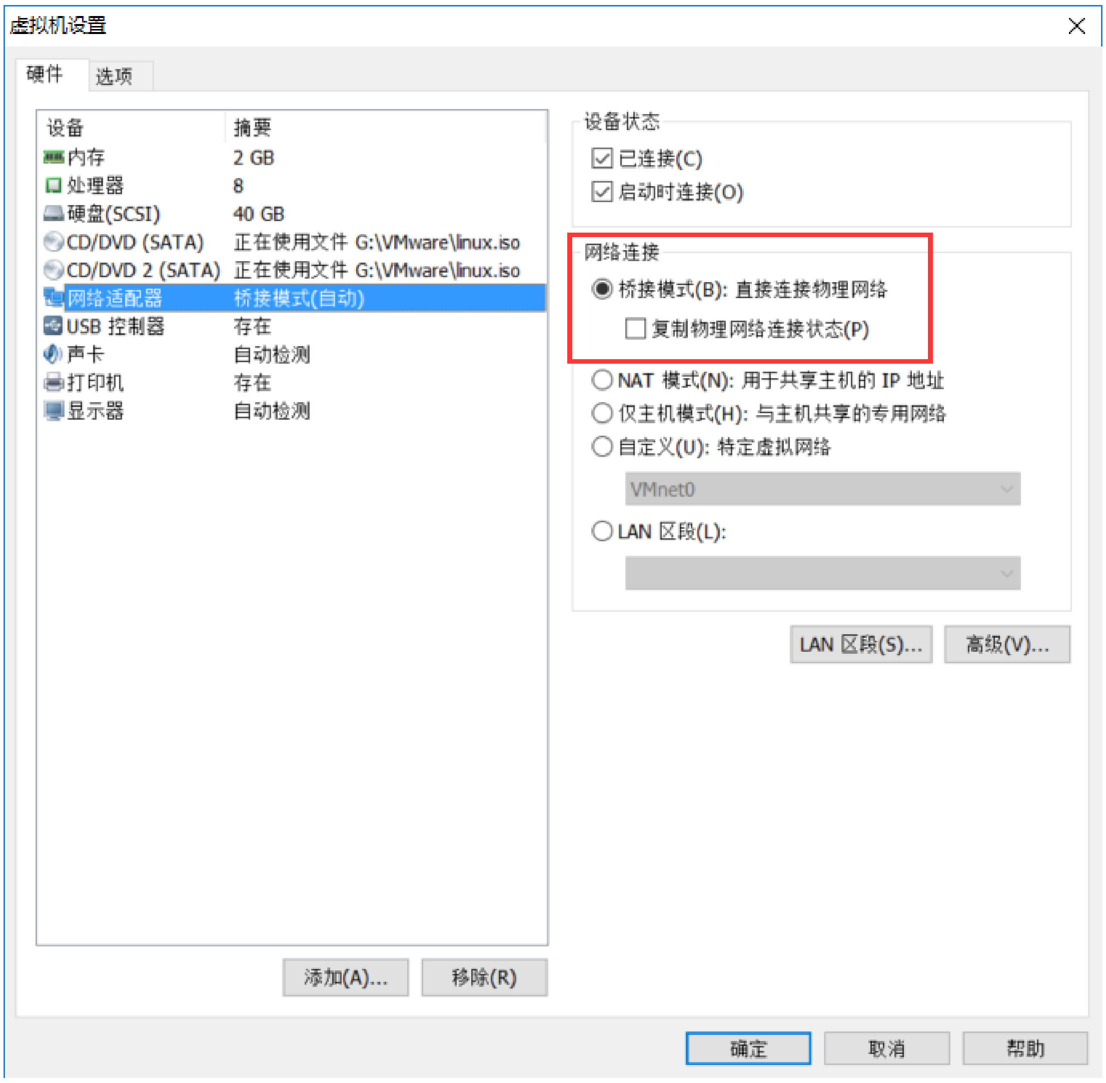
```
pi@HeltecLoRa:~/lora/lorasdk$ ./install.sh
Gateway_ID set to b827ebFFFE9d1cbe in file local_conf.json
pi@HeltecLoRa:~/lora/lorasdk$ cd ~
pi@HeltecLoRa:~$ sudo systemctl status lrgateway
● lrgateway.service - packet forwarder
   Loaded: loaded (/etc/systemd/system/lrgateway.service; enabled; vendor preset
   Active: active (running) since Thu 2018-03-08 05:09:53 UTC; 4min 40s ago
  Process: 734 ExecStopPost=/home/pi/lora/packet_forwarder/reset_pkt_fwd.sh stop
  Process: 744 ExecStartPre=/home/pi/lora/packet_forwarder/reset_pkt_fwd.sh star
 Main PID: 767 (lora_pkt_fwd)
   CGroup: /system.slice/lrgateway.service
           └─767 /home/pi/lora/packet_forwarder/lora_pkt_fwd/lora_pkt_fwd

Mar 08 05:13:26 HeltecLoRa lora_pkt_fwd[767]: # SX1301 time (PPS): 182419346
Mar 08 05:13:26 HeltecLoRa lora_pkt_fwd[767]: src/jitqueue.c:448:jit_print_queue
Mar 08 05:13:26 HeltecLoRa lora_pkt_fwd[767]: ### [GPS] ###
Mar 08 05:13:26 HeltecLoRa lora_pkt_fwd[767]: # GPS sync is disabled
Mar 08 05:13:26 HeltecLoRa lora_pkt_fwd[767]: ##### END #####
Mar 08 05:13:26 HeltecLoRa lora_pkt_fwd[767]: JSON up: {"stat":{"time":"2018-03-
Mar 08 05:13:26 HeltecLoRa lora_pkt_fwd[767]: ##### 2018-03-08 05:13:26 GMT ####
Mar 08 05:13:26 HeltecLoRa lora_pkt_fwd[767]: ### [UPSTREAM] ###
Mar 08 05:13:26 HeltecLoRa lora_pkt_fwd[767]: # RF packets received by concentra
Mar 08 05:13:26 HeltecLoRa lora_pkt_fwd[767]: # CRC_OK: 0.00%, CRC_FAIL: 0.00%,
lines 1-19/19 (END)
```

图 4-1 检查服务的运行状况

➢ **Ubuntu server and Raspberry PI are in a same gateway:**

The loria server used in the operation described in the document is a local server built on the virtual machine. To ensure that the experiment can be performed normally in the local area network, first make sure that the Raspberry Pi and Ubuntu server should be on the same network segment! For example, in the Raspberry Pi, the IP address viewed through the "ifconfig" command is 192.168.0.123, so the IP address of the Ubuntu server should also be 192.168.0.xxx.

By default, the VMware VM uses NAT to access the Internet. In the VM Settings, change it to Bridge Mode and restart (as shown in Figure 4-3).



设备	摘要
内存	2 GB
处理器	8
硬盘(SCSI)	40 GB
CD/DVD (SATA)	正在使用文件 G:\VMware\linux.iso
CD/DVD 2 (SATA)	正在使用文件 G:\VMware\linux.iso
网络适配器	桥接模式(自动)
USB 控制器	存在
声卡	自动检测
打印机	存在
显示器	自动检测

设备状态

- ☑ 已连接(C)
- ☑ 启动时连接(O)

网络连接

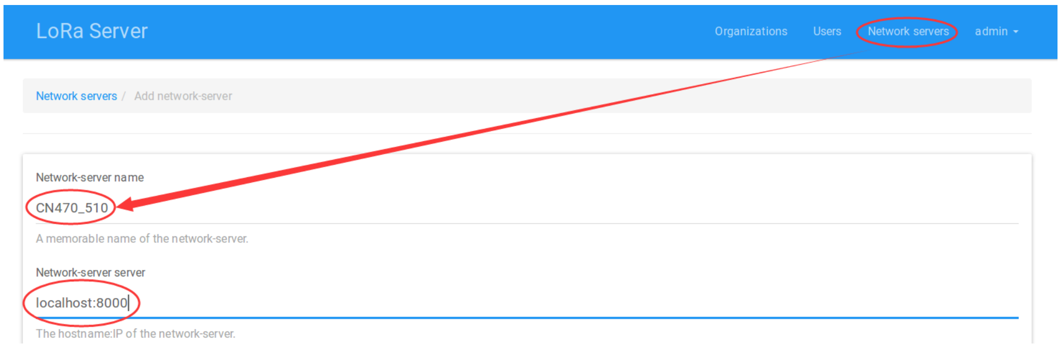
- ● 桥接模式(B): 直接连接物理网络
  - ☐ 复制物理网络连接状态(P)
- ○ NAT 模式(N): 用于共享主机的 IP 地址
- ○ 仅主机模式(H): 与主机共享的专用网络
- ○ 自定义(U): 特定虚拟网络
  - VMnet0
- ○ LAN 区段(L):

Figure 4-3 Setting a virtual machine to access the Internet

4.1 Initialization settings for the application

Open the browser in Ubuntu and enter https://127.0.0.1:8080. The default login and password are both admin. Go to the configuration page, first complete the following initialization configuration:

➢ In Network servers, fill in the working frequency band of your own lora gateway. The name in Network-server name can be set arbitrarily, but it is recommended to correspond to the regional frequency to facilitate differentiation.

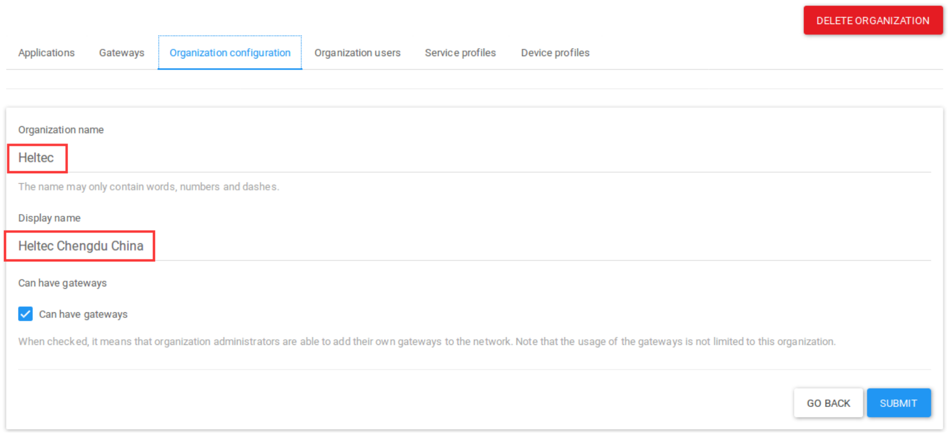

LoRa Server

Network servers / Add network-server

Network-server name
CN470_510
A memorable name of the network-server.

Network-server server
localhost:8000
The hostname:IP of the network-server.

➢ Enter the name of the company/organization/individual you want in the Organization:

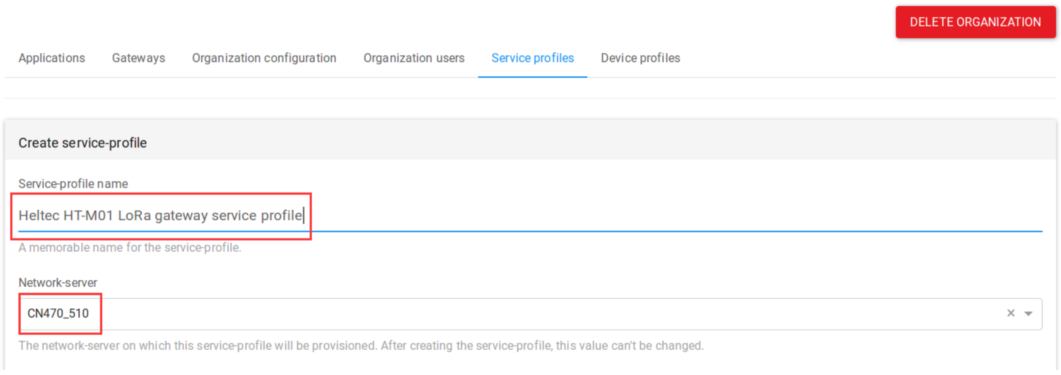


Organization name
Heltec
The name may only contain words, numbers and dashes.

Display name
Heltec Chengdu China

Can have gateways
✓ Can have gateways
When checked, it means that organization administrators are able to add their own gateways to the network. Note that the usage of the gateways is not limited to this organization.

➢ In the Service profile, the red box is required:



Create service-profile

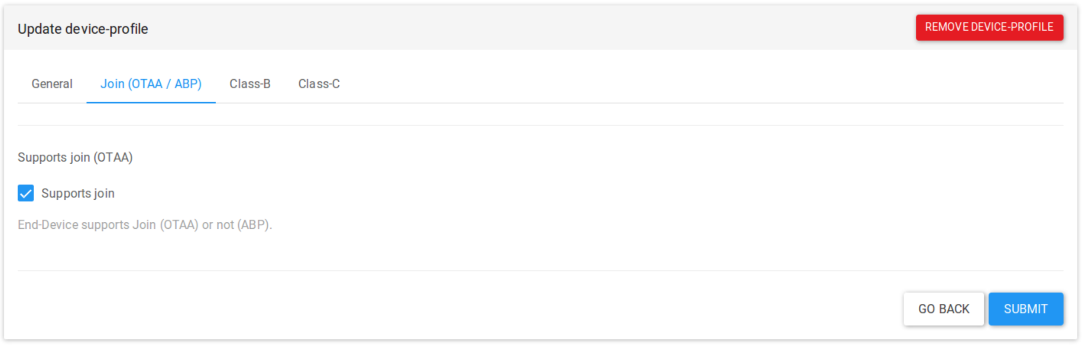
Service-profile name
Heltec HT-M01 LoRa gateway service profile
A memorable name for the service-profile.

Network-server
CN470_510
The network-server on which this service-profile will be provisioned. After creating the service-profile, this value can't be changed.

➢ Device profile, fill in node device information

✓ 1. Device profile name – In order to achieve the "see name and meaning", the name here is generally used to describe the function of the node device, such as "Temperature sensor", "Alarm node" and so on. Do not fill in double-byte characters such as Chinese, Korean, and Japanese.

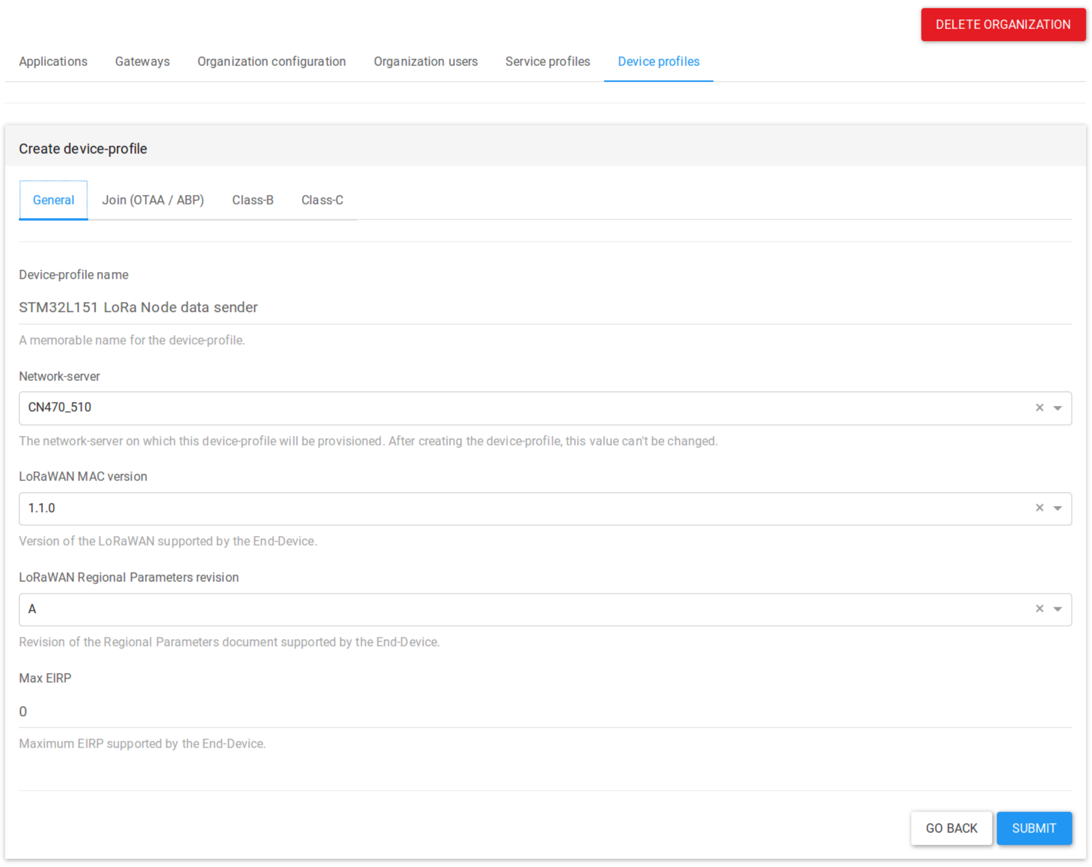
- ✓ LoRaWAN MAC version – This field should be filled in the code running on your node device, the version of the LoRaWAN protocol. As of the writing date of this document, the nodes that we can use to run the LoRaWAN protocol are all running the 1.0.2 version. (For the experiments we have done so far, the higher version of the LoRaWAN protocol is compatible with the lower version. You can directly select the highest version, but this is not recommended and may cause unknown errors and problems).
- ✓ Join (OTAA / ABP) item, check "Supports join".
- ✓ Keep other items as default.


Update device-profile

Supports join (OTAA)

✓ Supports join

End-Device supports Join (OTAA) or not (ABP).



Create device-profile

Device-profile name
STM32L151 LoRa Node data sender
A memorable name for the device-profile.

Network-server
CN470_510
The network-server on which this device-profile will be provisioned. After creating the device-profile, this value can't be changed.

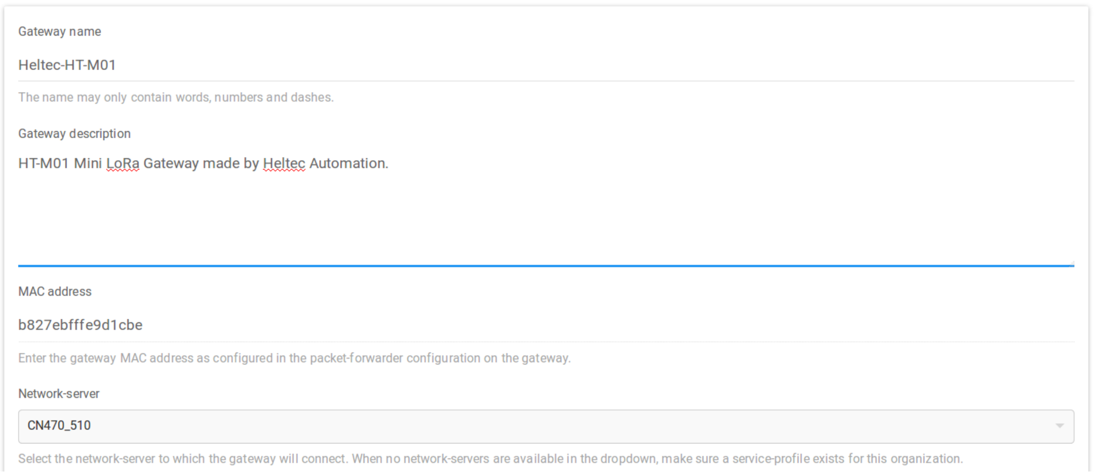
LoRaWAN MAC version
1.1.0
Version of the LoRaWAN supported by the End-Device.

LoRaWAN Regional Parameters revision
A
Revision of the Regional Parameters document supported by the End-Device.

Max EIRP
0
Maximum EIRP supported by the End-Device.

Register a gateway in the server:

"Gateways → CREATE GATEWAY", fill out the four places as shown in Figure 4-9. Note that the Gateway name can only be filled with letters, numbers, and dash! The MAC address is filled with the Gateway_ID obtained in the previous text.



Gateway name
Heltec-HT-M01
The name may only contain words, numbers and dashes.

Gateway description
HT-M01 Mini LoRa Gateway made by Heltec Automation.

MAC address
b827ebfffe9d1cbe
Enter the gateway MAC address as configured in the packet-forwarder configuration on the gateway.

Network-server
CN470_510
Select the network-server to which the gateway will connect. When no network-servers are available in the dropdown, make sure a service-profile exists for this organization.

Figure 4-9

Click "submit" in the lower right corner to complete the configuration of this step. If everything is normal, our HT-M01 gateway can connect to the server and communicate.

Enter the following command in the Raspberry Pi:

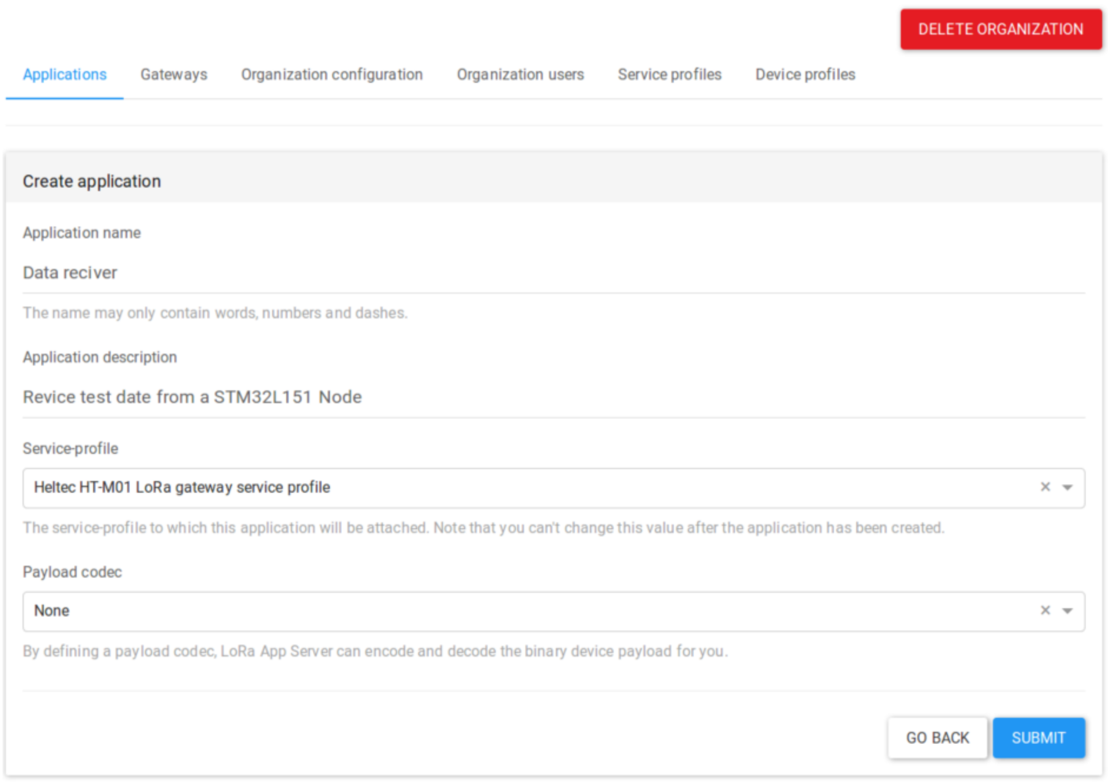
```
journalctl -u lora-gateway-bridge -f -n 50
```

If you get the following log information, it indicates that the gateway has established communication with the server:

```
Mar 08 09:30:46 HeltecLoRa lora-gateway-bridge[427]: time="2018-03-08T09:30:46Z" level=info msg="gateway: sending udp packet to gateway" addr="127.0.0.1:54450" protocol_version=2 type=PullACK

Mar 08 09:30:56 HeltecLoRa lora-gateway-bridge[427]: time="2018-03-08T09:30:56Z" level=info msg="gateway: received udp packet from gateway" addr="127.0.0.1:54450" protocol_version=2 type=PullData
```

4.2 Create an Application

"Applications → CREATE APPLICATION" creates an application class that is one of your many applications. For example: if you try lora technology to monitor the status of a greenhouse, then you should create a The application of "greenhouse-status" is to add nodes such as temperature, humidity, carbon dioxide, and other sensors under this application. As shown in Figure 4-10, the arbitrariness of this part of the set is relatively high, it is still recommended to know the name in the Application name, such as "greenhouse-status", "Alarm" and so on (supports only letters, numbers and dash).



Create application

Application name

Data reciver

The name may only contain words, numbers and dashes.

Application description

Revice test date from a STM32L151 Node

Service-profile

Heltec HT-M01 LoRa gateway service profile

The service-profile to which this application will be attached. Note that you can't change this value after the application has been created.

Payload codec

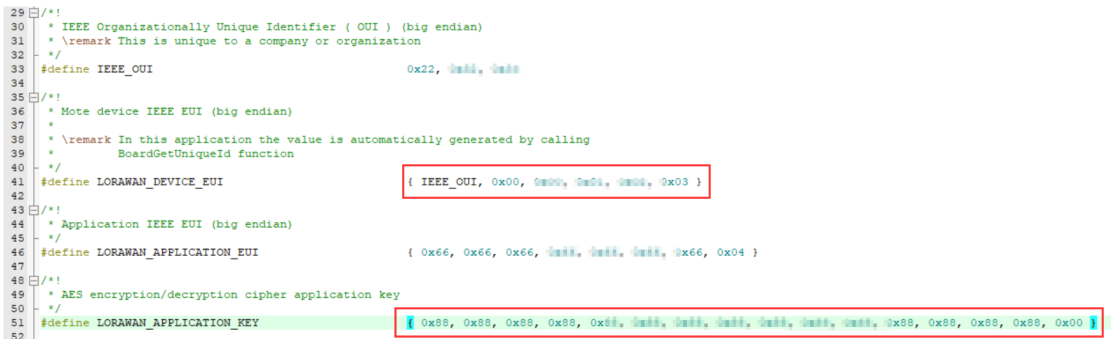
None

By defining a payload codec, LoRa App Server can encode and decode the binary device payload for you.

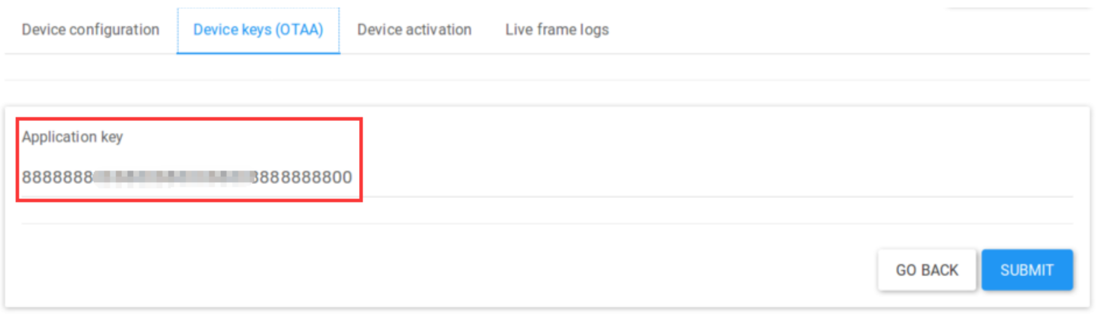
Fig. 4-10

4.3 Create a Node Example

"CREATE DEVICE" need to be in the application where the application the node to be added belongs. The place where you need to pay attention is Device EUI and Application key, which should be consistent with the value of the corresponding place in the "Commissioning.h" file in the code run by this node.

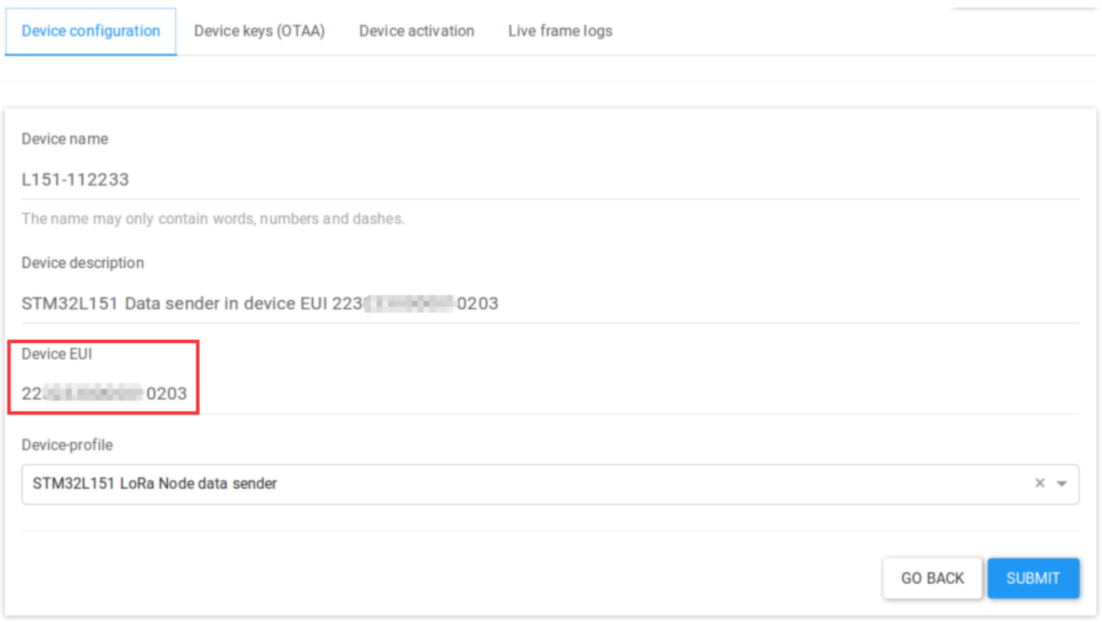


```
/*!
 * IEEE Organizationally Unique Identifier ( OUI ) (big endian)
 * \remark This is unique to a company or organization
 */
#define IEEE_OUI                                0x22, [redacted]

/*!
 * Mote device IEEE EUI (big endian)
 *
 * \remark In this application the value is automatically generated by calling
 *         BoardGetUniqueId function
 */
#define LORAWAN_DEVICE_EUI                      { IEEE_OUI, 0x00, [redacted] 0x03 }

/*!
 * Application IEEE EUI (big endian)
 */
#define LORAWAN_APPLICATION_EUI                 { 0x66, 0x66, 0x66, [redacted] 0x66, 0x04 }

/*!
 * AES encryption/decryption cipher application key
 */
#define LORAWAN_APPLICATION_KEY                 { 0x88, 0x88, 0x88, 0x88, 0x[redacted] 0x88, 0x88, 0x88, 0x88, 0x00 }
```



Application key

8888888[redacted]888888800



Device name

L151-112233

The name may only contain words, numbers and dashes.

Device description

STM32L151 Data sender in device EUI 223[redacted]0203

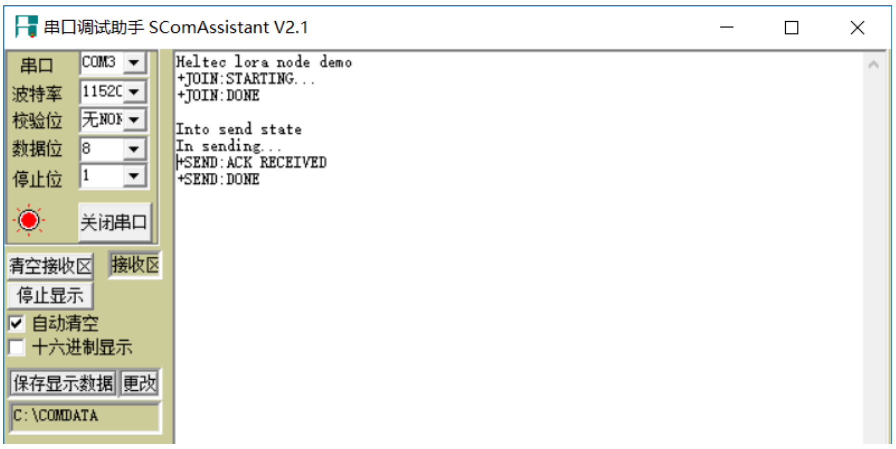
Device EUI

22[redacted]0203

Device-profile

STM32L151 LoRa Node data sender

After submitting, let the node device run. Open the serial port debugging assistant. If some of them are normal, you can see that it can successfully access the network, send data to the LoRa server, and receive an ACK through the debugging information printed out.



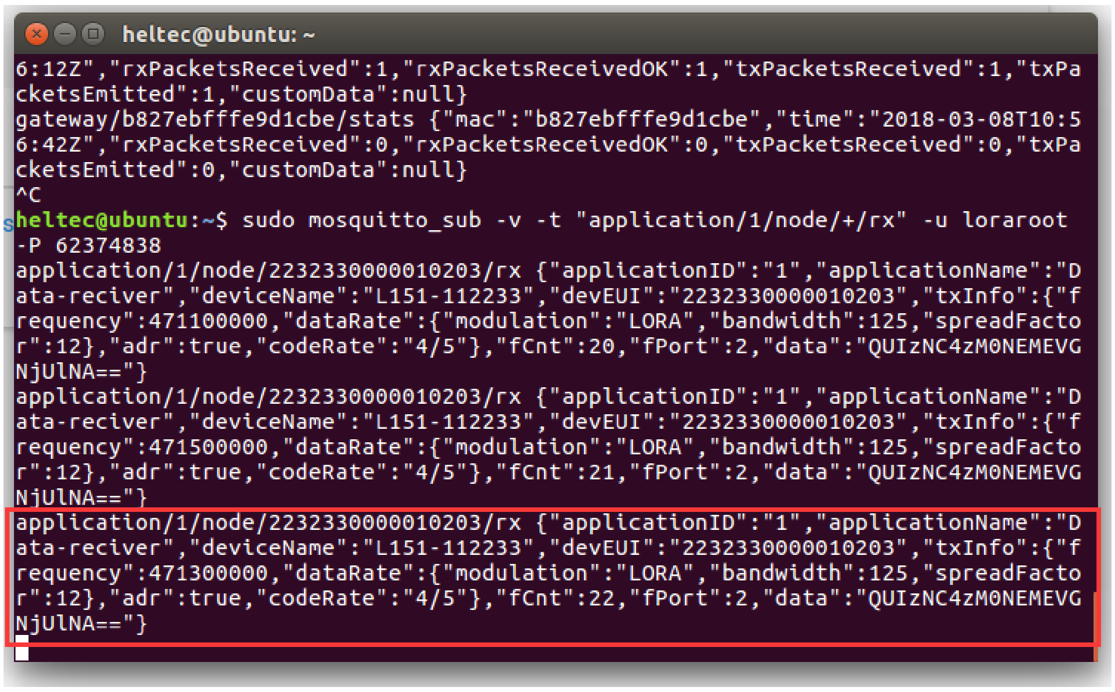
```
Heltec lora node demo
+JOIN:STARTING...
+JOIN:DONE

Into send state
In sending...
+SEND:ACK RECEIVED
+SEND:DONE
```

4.4 Data Analysis

In the previous sending data experiment, the experimental data sent on the node was "AB34.33CD0EF65%4". On the Ubuntu server, enter the following command, we can view the log information received by Mosquitto:

```
sudo mosquitto_sub -v -t "application/1/node/+/rx" -u loraroot -P 62374838
```

Get the result shown in Figure 4-15:



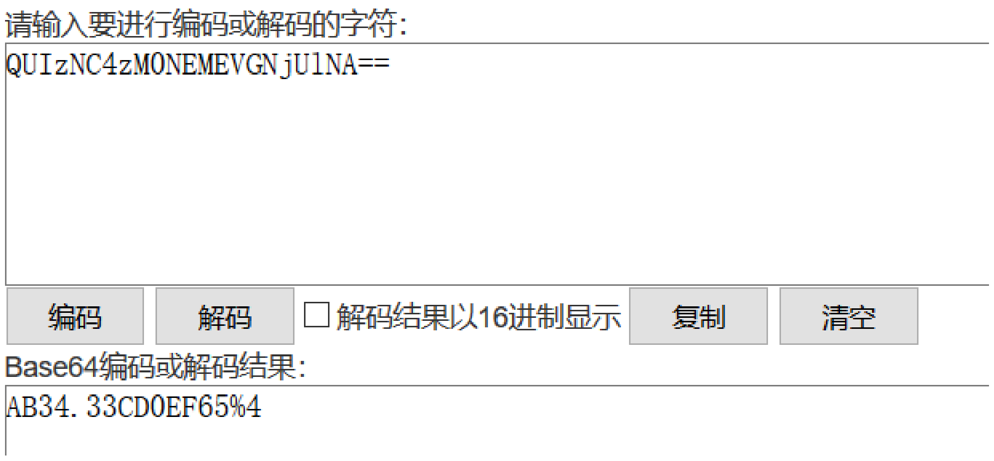
```
heltec@ubuntu: ~
6:12Z","rxPacketsReceived":1,"rxPacketsReceivedOK":1,"txPacketsReceived":1,"txPa
cketsEmitted":1,"customData":null}
gateway/b827ebfffe9d1cbe/stats {"mac":"b827ebfffe9d1cbe","time":"2018-03-08T10:5
6:42Z","rxPacketsReceived":0,"rxPacketsReceivedOK":0,"txPacketsReceived":0,"txPa
cketsEmitted":0,"customData":null}
^C
heltec@ubuntu:~$ sudo mosquitto_sub -v -t "application/1/node/+/rx" -u loraroot
-P 62374838
application/1/node/2232330000010203/rx {"applicationID":"1","applicationName":"D
ata-reciver","deviceName":"L151-112233","devEUI":"2232330000010203","txInfo":{"f
requency":471100000,"dataRate":{"modulation":"LORA","bandwidth":125,"spreadFacto
r":12},"adr":true,"codeRate":"4/5"},"fCnt":20,"fPort":2,"data":"QUIzNC4zM0NEMEVG
NjUlNA=="}
application/1/node/2232330000010203/rx {"applicationID":"1","applicationName":"D
ata-reciver","deviceName":"L151-112233","devEUI":"2232330000010203","txInfo":{"f
requency":471500000,"dataRate":{"modulation":"LORA","bandwidth":125,"spreadFacto
r":12},"adr":true,"codeRate":"4/5"},"fCnt":21,"fPort":2,"data":"QUIzNC4zM0NEMEVG
NjUlNA=="}
application/1/node/2232330000010203/rx {"applicationID":"1","applicationName":"D
ata-reciver","deviceName":"L151-112233","devEUI":"2232330000010203","txInfo":{"f
requency":471300000,"dataRate":{"modulation":"LORA","bandwidth":125,"spreadFacto
r":12},"adr":true,"codeRate":"4/5"},"fCnt":22,"fPort":2,"data":"QUIzNC4zM0NEMEVG
NjUlNA=="}
```

Figure 4-15 Mosquitto log information view

The red box is the contents of a json package received on the server. Let's analyze the information it contains:

```
{
  "applicationID":"1", // The application ID is automatically assigned by the web server
  "applicationName":"Data-reciver", // The name of the application is Data-reciver, set manually on the web server
  "deviceName":"L151-112233",// device name L151-112233, set manually
  "devEUI":"2232330000010203",//节点设备的唯一识别号，在节点上由软件设定
  "rxInfo":[],
  "txInfo":
  {
    "frequency":471300000,
    "dataRate":
    {
      "modulation":"LORA",
      "bandwidth":125,
      "spreadFactor":12
    },
    "adr":true,
    "codeRate":"4/5"
  },
  "fCnt":22,
  "fPort":2,
  "data":" QUIzNC4zM0NEMEVGNjU1NA==""
}
```

After decrypting this “garbage” (as shown in Figure 4-16), it is exactly the same as the data we sent on the node.



```
QUIzNC4zM0NEMEVGNjU1NA==
```

```
AB34.33CD0EF65%4
```

Figure 4-16 base64 codec

5. Conclusion

Reference： https://docs.loraserver.io/overview/

Contact us： www.heltec.cn

Sale email: echo@heltec.cn

Technical support email:： support@heltec.cn