Chengdu Ebyte Electronic Technology Co.,Ltd

Wireless Modem

User Manual



E870-LxxxLG12-O Series User Manual

All rights to interpret and modify this manual belong to
Chengdu Ebyte Electronic Technology Co., Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Contents

Features 3
1 Product Introduction 4
1.1 Basic Parameters 4
1.2 Dimensions & Interface Description 4
2 Hardware Connection 5
3 User Instructions 6
3.1 Basic Tools and Command Usage 6
3.1.1 MobaXterm 6
3.1.2 VIM Basics 10
3.1.3 View logs for currently running containers 11
3.1.4 View network adapter information 12
3.1.5 View Gateway Service Unit Logs 13
3.2 Network Service Configuration 14
3.2.1 Ethernet 14
3.2.2 WIFI 16
3.3 Script Usage 17
3.3.1 Gateway Script 17
3.3.2 Server Script 17
3.3.3 Node-RED Script 18
3.4 Gateway Parameter Configuration 19
3.4.1 Script Usage 19
3.4.2 Configuration file 19
3.5 Server Parameter Configuration 20
3.5.1 Script Usage 20
3.5.2 Chirpstack Parameter Specifications 20
3.6 Node Device Communication 21
3.6.1 Node Configuration 22
3.6.2 Communication Testing 22
3.7 Node-RED LoRaWAN Node 25
3.7.1 Node Configuration 25

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.7.2 Chirpstack-related nodes 25
3.8 Use of Peripheral Devices 27
3.8.1 LED 27
3.8.2 Button 28
3.9 Cross-compilation 28
3.9.1 Download Toolchain 28
3.9.2 Unzip, install, and verify the toolchain 29
3.9.3 Cross-Compilation Demonstration 30
Revision history 32
About us 32

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Features

- Supports 1 wired WAN port, supporting 10/100/1000Mbps;
- Supports 1 button (user-configurable);
- Supports 8 LED indicators (all seven LEDs except PWR are user-configurable);
- Supports Shell configuration method;
- Built-in Chirpstack server;
- Built-in Node-RED;
- Supports firmware secondary development;
- Fully open-source;

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

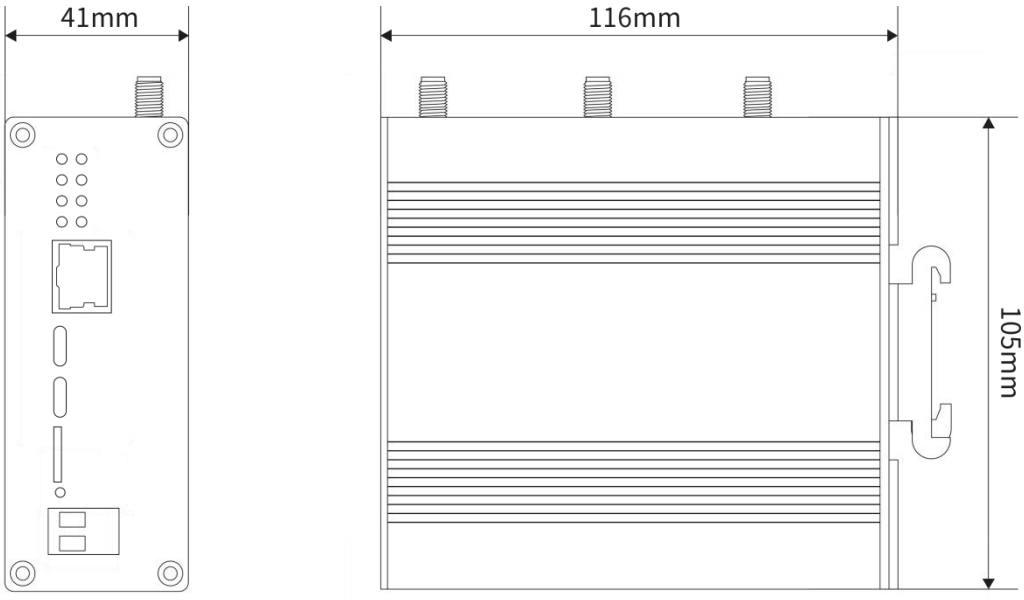
1 Product Introduction

1.1 Basic Parameters

	Item	Indicator
Hardware Specifications	Product Dimensions (H*W*D)	116×105×41mm
	Product Weight	417g
	Operating Temperature	-40℃~+85℃
	Storage temperature	-40℃~+85℃
	Operating Humidity	5%~95%
	Storage humidity	1%~95%
	Operating Voltage	8V~28V

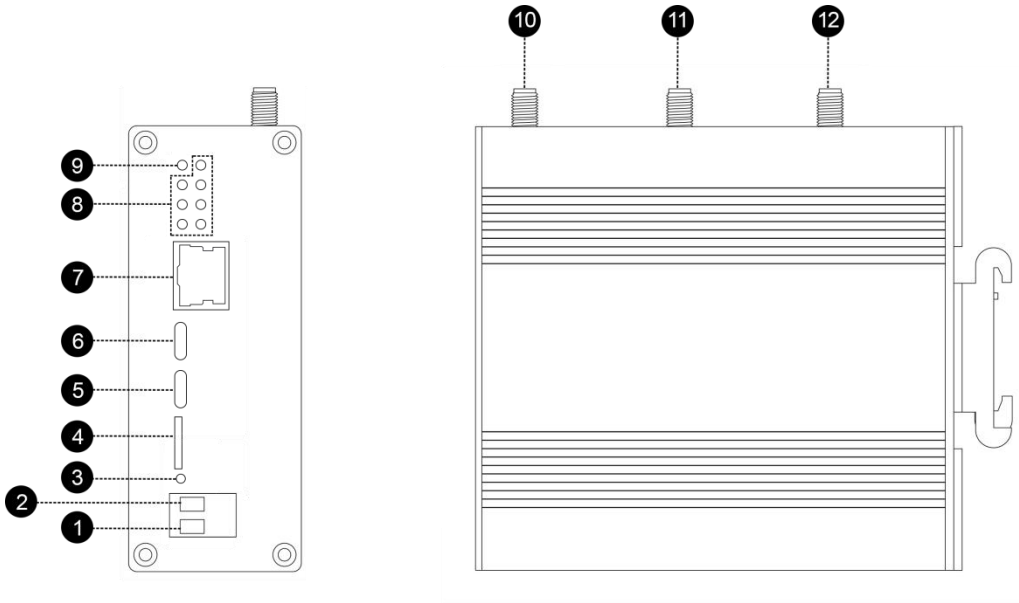
E870-L470LG12-O supports CN470 region files.
E870-L868LG12-O supports EU868, RU864, and IN865 region files.
E870-L915LG12-O supports US915, AU915, AS923, and KR920 region files.

1.2 Dimensions & Interface Description



Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



The hardware interface is described as follows:

Interface Serial Number	Name	Note
1	DC-IN+	Power Supply Range DC: 8–28V, Positive Terminal
2	DC-IN-	Power Supply Range DC: 8–28V, negative terminal
3	KEY	1. Firmware programming enabled 2. User-defined functions
4	SIM card slot	Only 4G-enabled devices are supported.
5	USB2	USB debugging interface
6	USB1	USB programming interface
7	WAN	WAN port, 10/100/1000 Mbps
8	LED1-LED7	User-defined LED
9	PWR	Power indicator light, normally lit
10	WIFI Antenna	WiFi Antenna (SMA Connector)
11	4G Antenna	4G antenna (SMA connector), compatible only with 4G-enabled devices
12	LoRa Antenna	LoRa Antenna (SMA Connector)

2 Hardware Connection

Connect the Wi-Fi antenna and LoRa antenna to the SMA connectors labeled on the device panel. After completing these steps, use the device's included power supply to power on the gateway. Once powered on, the PWR indicator on the panel will remain lit, indicating the device is receiving normal power.

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3 User Instructions

In this chapter, we will introduce and explain the functional configuration.

3.1 Basic Tools and Command Usage

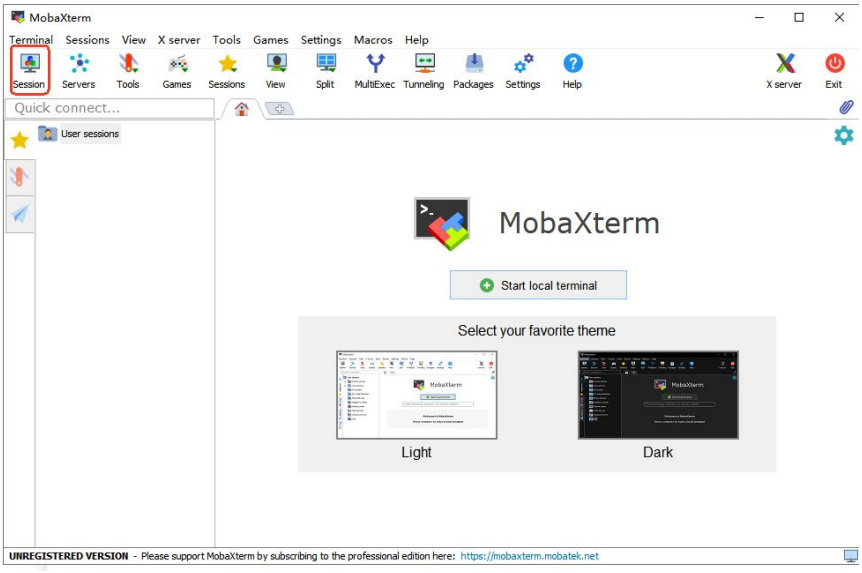
3.1.1 MobaXterm

MobaXterm is an enhanced remote computing toolkit that integrates multiple functionalities including SSH, X11, RDP, VNC, FTP, SFTP, terminal, and Unix commands. At its core, it serves as a portable Unix environment emulator, enabling you to manage remote servers on Windows with the same efficiency as using Linux.

In the E870-LxxxLG12-O series open-source gateways, devices can be connected via Serial or SSH for remote viewing and configuration of device-related parameters.

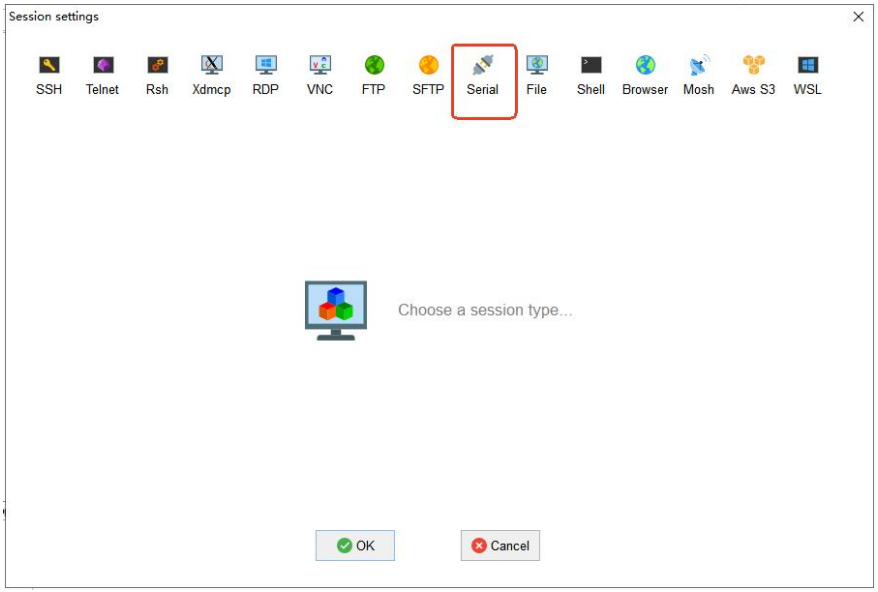
(1)Serial：

① Connect the device's debug port using a USB cable. Launch the configuration tool—MobaXterm software (available for download from the official website; users must install it themselves). Click “Session” in the upper-left corner.

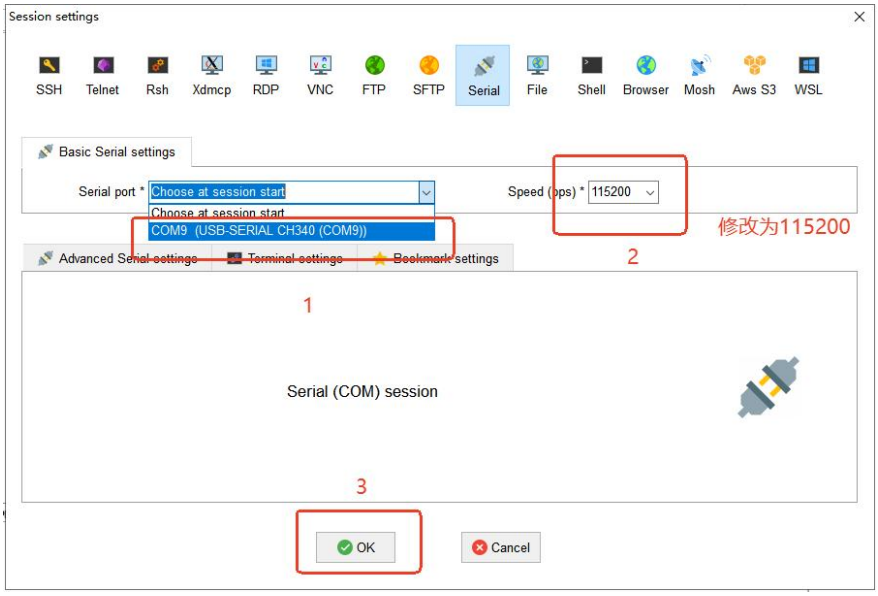


② Click “Serial”

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



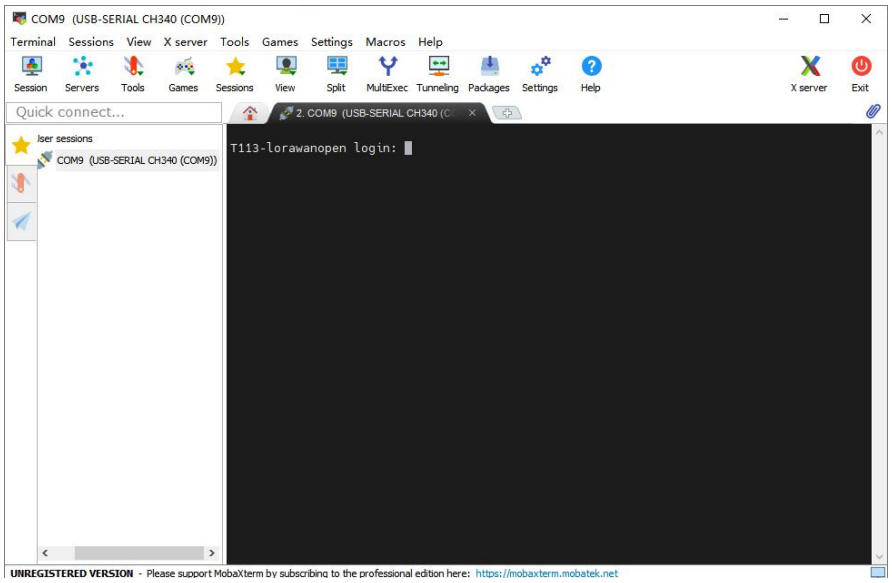
③ Select the port number, change the baud rate to 115200, and click OK after completing the settings.



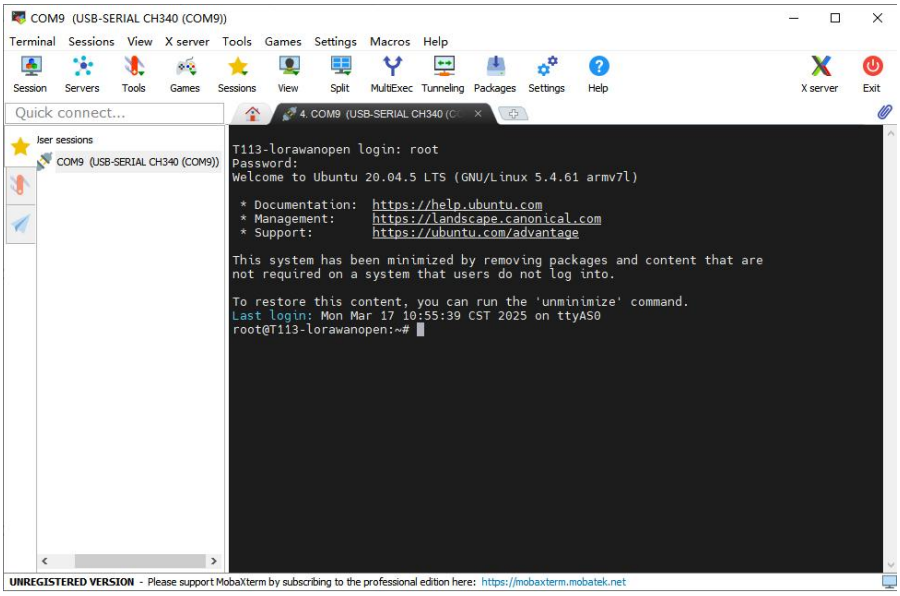
④ After completing the previous step, enter this interface (initially without the “T113-lorawanopen login:” prompt; press Enter to display it). Enter the username and password sequentially; both are “root”.

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



⑤ After entering your account credentials, wait for the configuration screen to load. Once the screen shown below appears, you may begin configuring and using the relevant features.

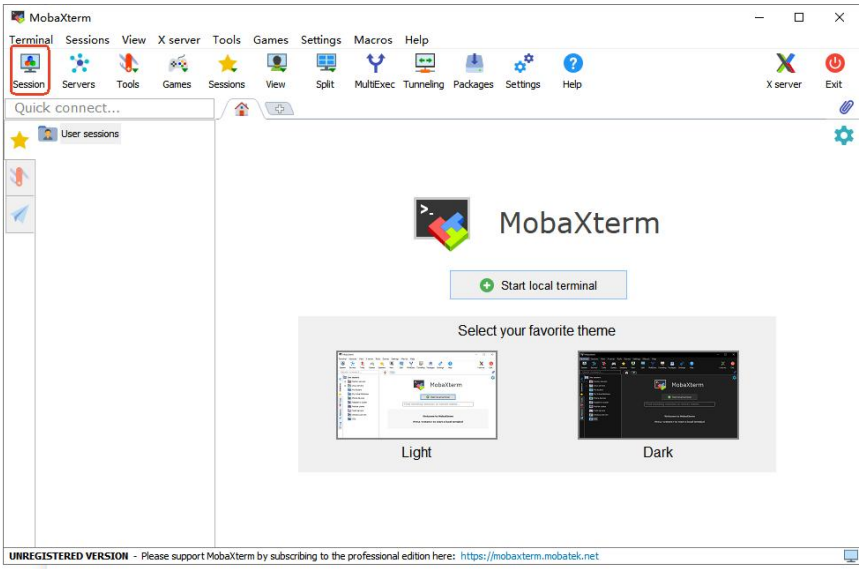


(2) SSH (if you can confirm the gateway's IP address, you can also access the configuration interface via an SSH tool):

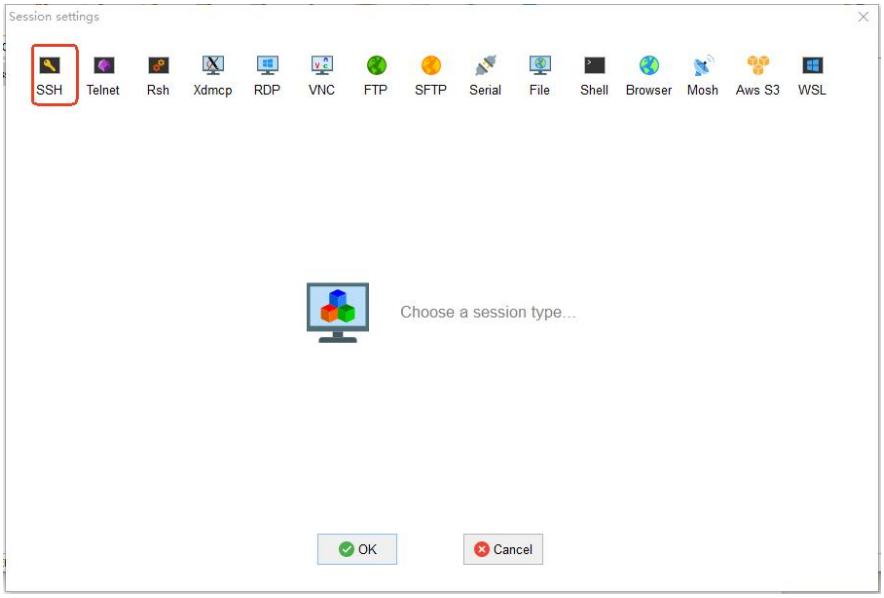
① Open the configuration tool—MobaXterm software (downloadable from the official website; users must install it themselves), then click “Session” in the upper-left corner.

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



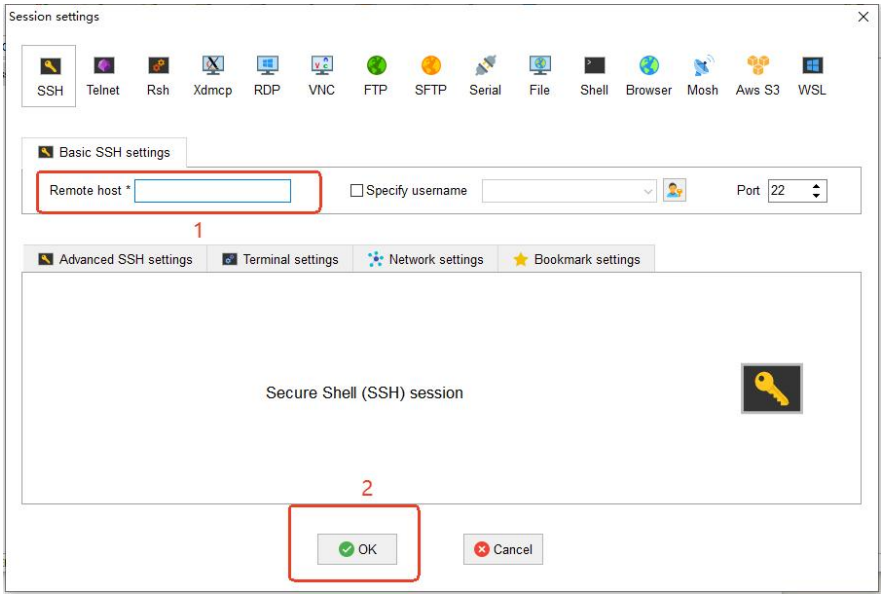
②click “SSH”



③ Enter the device IP address and click OK.

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



④ After completing the previous step, enter this interface and input the username and password in sequence. Both the username and password are “root”.

⑤ After entering the username and password, wait for the configuration to load. When the screen shown below appears, you may begin configuring and using the relevant functions.

3.1.2 VIM Basics

Vim is a terminal text editor renowned for its pure keyboard operation, extreme efficiency, and limitless customization. It exists on nearly all Unix/Linux systems and serves as the “standard tool” for programmers, system administrators, and hackers.

In the E870-LxxxLG12-O series open-source gateways, certain parameter files can be modified using the Vim editor. We commonly employ two modes: command mode and insert mode.

(1) Command Pattern:

After opening a file with a Vim command, you are now in command mode. In command mode, keystrokes are interpreted as commands rather than character input. For example, pressing `i` does not input the character ‘i’, but instead executes the corresponding command—switching to insert mode. Below are several commonly used commands in normal mode:

i -- Switch to insert mode (begin typing at the current cursor position)

x -- Delete the character at the current cursor position

a -- Enter insert mode, begin typing at the next position after the cursor

u -- Undo the last operation

:w -- Save the file

:q -- Exit Vim editor (without saving changes)

:wq -- Save and exit Vim editor

:q! -- Force quit Vim editor (without saving changes)

(2) Input Mode:

Pressing ‘i’ in command mode enters input mode. In input mode, you can modify parameters as needed. After making changes, press ESC to return to command mode. Within input mode, use the arrow keys to move the cursor through

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

the text, and Page Up/Page Down to scroll up/down through pages.

3.1.3 View logs for currently running containers

Docker Compose is a tool for defining and running multi-container Docker applications, with common commands including starting, stopping, and restarting services. Here we primarily explain how to use Docker Compose commands to view running container logs.

Note that you must first navigate to the specified directory using the command `cd /root/lorawanopen/chirpstack` before executing the following commands:

1. docker-compose ps: View the status of containers currently managed by Compose.

```
root@T113-lorawanopen:~/lorawanopen/chirpstack# docker-compose ps
NAME                                          IMAGE                                          COMMAND
       SERVICE                               CREATED        STATUS         PORTS
chirpstack-chirpstack-1                       chirpstack/chirpstack:4                        "/usr/bin/chirpst
ack…"   chirpstack                           7 months ago   Up 9 minutes   0.0.0.0:8080->8080/tcp, :::8080->
8080/tcp
chirpstack-chirpstack-gateway-bridge-1        chirpstack/chirpstack-gateway-bridge:4   "/usr/bin/chirpst
ack…"   chirpstack-gateway-bridge            7 months ago   Up 9 minutes   0.0.0.0:1700->1700/udp, :::1700->
1700/udp
chirpstack-chirpstack-gateway-bridge-basicstation-1   chirpstack/chirpstack-gateway-bridge:4   "/usr/bin/chirpst
ack…"   chirpstack-gateway-bridge-basicstation   7 months ago   Up 9 minutes   0.0.0.0:3001->3001/tcp, :::3001->
3001/tcp
chirpstack-chirpstack-rest-api-1              chirpstack/chirpstack-rest-api:4         "/usr/bin/chirpst
ack…"   chirpstack-rest-api                   7 months ago   Up 9 minutes   0.0.0.0:8090->8090/tcp, :::8090->
8090/tcp
chirpstack-mosquitto-1                        eclipse-mosquitto:2                      "/docker-entrypoi
nt.…"   mosquitto                            7 months ago   Up 9 minutes   0.0.0.0:1883->1883/tcp, :::1883->
1883/tcp
chirpstack-postgres-1                         postgres:14-alpine                       "docker-entrypoin
t.s…"   postgres                             7 months ago   Up 9 minutes   5432/tcp
chirpstack-redis-1                            redis:7-alpine                           "docker-entrypoin
t.s…"   redis                                7 months ago   Up 9 minutes   6379/tcp
root@T113-lorawanopen:~/lorawanopen/chirpstack#
```

2. docker-compose logs: View the log output of the service.

```
root@T113-lorawanopen:~/lorawanopen/chirpstack# docker-compose ps
NAME                                          IMAGE                                          COMMAND
       SERVICE                               CREATED        STATUS         PORTS
chirpstack-chirpstack-1                       chirpstack/chirpstack:4                        "/usr/bin/chirpst
ack…"   chirpstack                           7 months ago   Up 9 minutes   0.0.0.0:8080->8080/tcp, :::8080->
8080/tcp
chirpstack-chirpstack-gateway-bridge-1        chirpstack/chirpstack-gateway-bridge:4   "/usr/bin/chirpst
ack…"   chirpstack-gateway-bridge            7 months ago   Up 9 minutes   0.0.0.0:1700->1700/udp, :::1700->
1700/udp
chirpstack-chirpstack-gateway-bridge-basicstation-1   chirpstack/chirpstack-gateway-bridge:4   "/usr/bin/chirpst
ack…"   chirpstack-gateway-bridge-basicstation   7 months ago   Up 9 minutes   0.0.0.0:3001->3001/tcp, :::3001->
3001/tcp
chirpstack-chirpstack-rest-api-1              chirpstack/chirpstack-rest-api:4         "/usr/bin/chirpst
ack…"   chirpstack-rest-api                   7 months ago   Up 9 minutes   0.0.0.0:8090->8090/tcp, :::8090->
8090/tcp
chirpstack-mosquitto-1                        eclipse-mosquitto:2                      "/docker-entrypoi
nt.…"   mosquitto                            7 months ago   Up 9 minutes   0.0.0.0:1883->1883/tcp, :::1883->
1883/tcp
chirpstack-postgres-1                         postgres:14-alpine                       "docker-entrypoin
t.s…"   postgres                             7 months ago   Up 9 minutes   5432/tcp
chirpstack-redis-1                            redis:7-alpine                           "docker-entrypoin
t.s…"   redis                                7 months ago   Up 9 minutes   6379/tcp
root@T113-lorawanopen:~/lorawanopen/chirpstack# clear
root@T113-lorawanopen:~/lorawanopen/chirpstack# docker-compose logs
chirpstack-rest-api-1  | Starting ChirpStack REST API server
chirpstack-rest-api-1  | Starting ChirpStack REST API server
chirpstack-rest-api-1  | Starting ChirpStack REST API server
chirpstack-rest-api-1  | Starting ChirpStack REST API server
chirpstack-rest-api-1  | Starting ChirpStack REST API server
chirpstack-rest-api-1  | Starting ChirpStack REST API server
chirpstack-rest-api-1  | Starting ChirpStack REST API server
chirpstack-rest-api-1  | Starting ChirpStack REST API server
chirpstack-rest-api-1  | Starting ChirpStack REST API server
chirpstack-rest-api-1  | Starting ChirpStack REST API server
chirpstack-rest-api-1  | Starting ChirpStack REST API server
chirpstack-rest-api-1  | Starting ChirpStack REST API server
chirpstack-rest-api-1  | Starting ChirpStack REST API server
chirpstack-rest-api-1  | Starting ChirpStack REST API server
chirpstack-rest-api-1  | Starting ChirpStack REST API server
```

To view the logs for a specific service individually, use the `docker-compose logs xx` command (where `xx` is the

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

specific service whose logs you want to view). For example, to view the logs for Chirpstack, use the command: docker-compose logs chirpstack

```
root@T113-lorawanopen:~/lorawanopen/chirpstack# docker-compose logs chirpstack
chirpstack-1  | 2025-04-25T06:04:23.008509Z  INFO chirpstack::cmd::root: Starting ChirpStack LoRaWAN Network Server ver
chirpstack-1  | 2025-04-25T06:04:23.008885Z  INFO chirpstack::storage::postgres: Setting up PostgreSQL connection pool
chirpstack-1  | 2025-04-25T06:04:23.010823Z  INFO chirpstack::storage: Applying schema migrations
chirpstack-1  | 2025-04-25T06:04:23.542476Z  INFO chirpstack::storage: Setting up Redis client
chirpstack-1  | 2025-04-25T06:04:23.544558Z  INFO chirpstack::region: Setting up regions
chirpstack-1  | 2025-04-25T06:04:23.546641Z  INFO setup{common_name=AS923 region_id=as923}: chirpstack::region: Configu
chirpstack-1  | 2025-04-25T06:04:23.551566Z  INFO setup{common_name=AS923_2 region_id=as923_2}: chirpstack::region: Con
chirpstack-1  | 2025-04-25T06:04:23.553596Z  INFO setup{common_name=AS923_3 region_id=as923_3}: chirpstack::region: Con
chirpstack-1  | 2025-04-25T06:04:23.555710Z  INFO setup{common_name=AS923_4 region_id=as923_4}: chirpstack::region: Con
chirpstack-1  | 2025-04-25T06:04:23.555997Z  INFO setup{common_name=AU915 region_id=au915_0}: chirpstack::region: Confi
chirpstack-1  | 2025-04-25T06:04:23.557022Z  INFO setup{common_name=CN470 region_id=cn470_0}: chirpstack::region: Confi
chirpstack-1  | 2025-04-25T06:04:23.557354Z  INFO setup{common_name=CN470 region_id=cn470_10}: chirpstack::region: Conf
chirpstack-1  | 2025-04-25T06:04:23.557899Z  INFO setup{common_name=CN779 region_id=cn779}: chirpstack::region: Configu
chirpstack-1  | 2025-04-25T06:04:23.559902Z  INFO setup{common_name=EU433 region_id=eu433}: chirpstack::region: Configu
chirpstack-1  | 2025-04-25T06:04:23.560250Z  INFO setup{common_name=EU868 region_id=eu868}: chirpstack::region: Configu
chirpstack-1  | 2025-04-25T06:04:23.560613Z  INFO setup{common_name=IN865 region_id=in865}: chirpstack::region: Configu
chirpstack-1  | 2025-04-25T06:04:23.563324Z  INFO setup{common_name=ISM2400 region_id=ism2400}: chirpstack::region: Con
chirpstack-1  | 2025-04-25T06:04:23.563542Z  INFO setup{common_name=KR920 region_id=kr920}: chirpstack::region: Configu
chirpstack-1  | 2025-04-25T06:04:23.563781Z  INFO setup{common_name=RU864 region_id=ru864}: chirpstack::region: Configu
chirpstack-1  | 2025-04-25T06:04:23.563960Z  INFO setup{common_name=US915 region_id=us915_0}: chirpstack::region: Confi
chirpstack-1  | 2025-04-25T06:04:23.564390Z  INFO setup{common_name=US915 region_id=us915_1}: chirpstack::region: Confi
chirpstack-1  | 2025-04-25T06:04:23.565000Z  INFO chirpstack::backend::joinserver: Setting up Join Server clients
chirpstack-1  | 2025-04-25T06:04:23.565163Z  INFO chirpstack::backend::roaming: Setting up roaming clients
chirpstack-1  | 2025-04-25T06:04:23.565204Z  INFO chirpstack::adr: Setting up adr algorithms
chirpstack-1  | 2025-04-25T06:04:23.565380Z  INFO chirpstack::integration: Setting up global integrations
chirpstack-1  | 2025-04-25T06:04:23.565425Z  INFO chirpstack::integration::redis: Initializing Redis integration
chirpstack-1  | 2025-04-25T06:04:23.565455Z  INFO chirpstack::integration::mqtt: Initializing MQTT integration
chirpstack-1  | 2025-04-25T06:04:23.652683Z  INFO chirpstack::integration::mqtt: Connecting to MQTT broker server_uri=t
chirpstack-1  | 2025-04-25T06:04:23.652943Z  INFO chirpstack::gateway::backend: Setting up gateway backends for the dif
chirpstack-1  | 2025-04-25T06:04:23.652993Z  INFO chirpstack::gateway::backend: Setting up gateway backend for region r
chirpstack-1  | 2025-04-25T06:04:23.653310Z  INFO chirpstack::integration::mqtt: Starting MQTT event loop
chirpstack-1  | 2025-04-25T06:04:23.661672Z  INFO chirpstack::integration::mqtt: Subscribing to command topic command_t
chirpstack-1  | 2025-04-25T06:04:23.682681Z  INFO chirpstack::gateway::backend::mqtt: Connecting to MQTT broker region_
chirpstack-1  | 2025-04-25T06:04:23.683256Z  INFO chirpstack::gateway::backend::mqtt: Starting MQTT event loop
chirpstack-1  | 2025-04-25T06:04:23.685041Z  INFO chirpstack::gateway::backend: Setting up gateway backend for region r
chirpstack-1  | 2025-04-25T06:04:23.694809Z  INFO chirpstack::gateway::backend::mqtt: Subscribing to gateway event topi
```

3.1.4 View network adapter information

The ifconfig command is a classic tool used in Linux and Unix systems to configure and view network interfaces.

Common commands:

1. ifconfig: View information about all network interfaces.

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
root@T113-lorawanopen:~/lorawanopen/chirpstack# ifconfig
br-39187a48a78d: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.18.0.1  netmask 255.255.0.0  broadcast 172.18.255.255
        inet6 fe80::42:e0ff:fed8:fe81  prefixlen 64  scopeid 0x20<link>
        ether 02:42:e0:d8:fe:81  txqueuelen 0  (Ethernet)
        RX packets 351  bytes 12405 (12.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 371  bytes 24671 (24.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:aa:82:04:fd  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.3.71  netmask 255.255.255.0  broadcast 192.168.3.255
        inet6 fe80::50:46ff:fe1d:9501  prefixlen 64  scopeid 0x20<link>
        ether 02:50:46:1d:95:01  txqueuelen 1000  (Ethernet)
        RX packets 558  bytes 67677 (67.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 33  bytes 2898 (2.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 36

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 559  bytes 30658 (30.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 559  bytes 30658 (30.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2. ifconfig xx: View information for a specified network interface, where xx is the interface to be examined. For example, to view information for eth0, simply use the command `ifconfig eth0`.

```
root@T113-lorawanopen:~/lorawanopen/chirpstack# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.3.71  netmask 255.255.255.0  broadcast 192.168.3.255
        inet6 fe80::50:46ff:fe1d:9501  prefixlen 64  scopeid 0x20<link>
        ether 02:50:46:1d:95:01  txqueuelen 1000  (Ethernet)
        RX packets 639  bytes 78084 (78.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 33  bytes 2898 (2.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 36

root@T113-lorawanopen:~/lorawanopen/chirpstack#
```

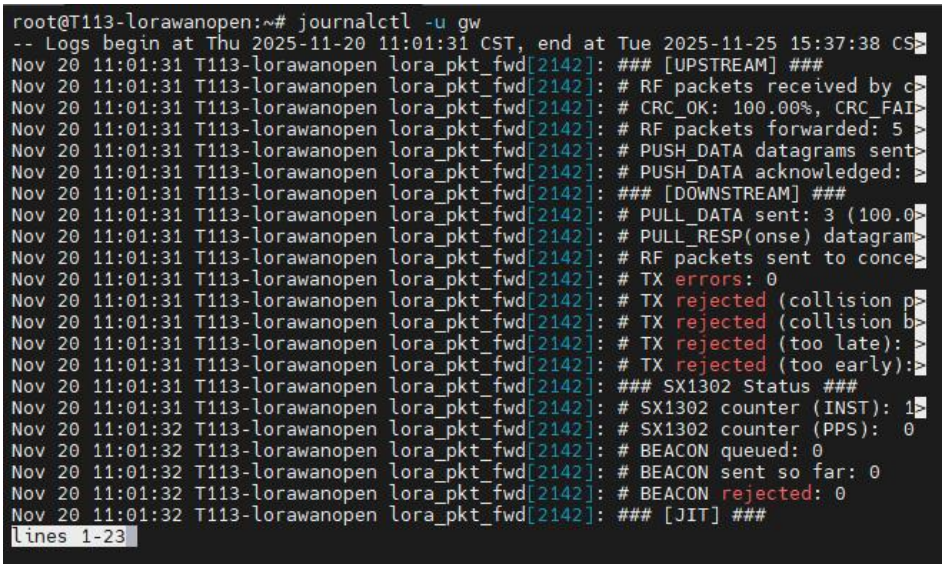
3.1.5 View Gateway Service Unit Logs

journalctl is a powerful tool in Linux systems for querying and displaying systemd logs. The basic command format for journalctl is: journalctl [options] [matching criteria...]. Common commands are as follows:

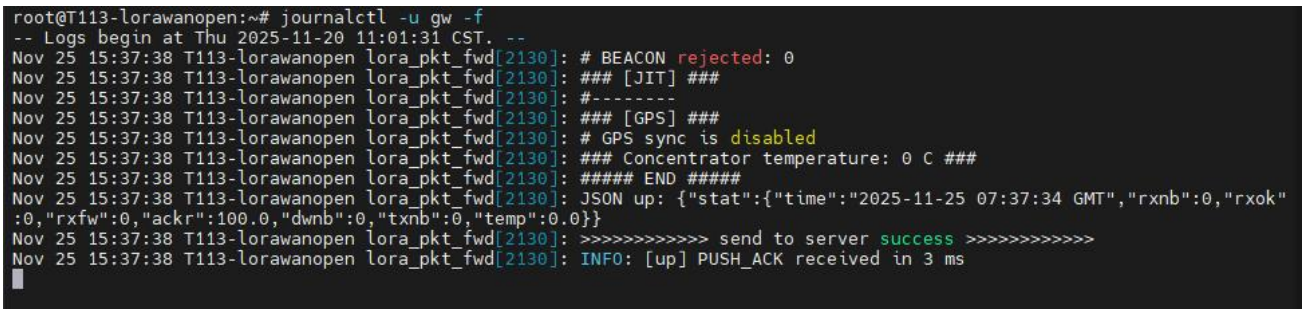
1. journalctl -u xx: View logs for a specific service, where xx is the name of the service to be viewed. For example, to view gateway logs, use the command journalctl -u gw.

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



2. journalctl -u xx -f: View the logs of a specific service in real time, where xx is the name of the service you want to monitor.



3.2 Network Service Configuration

3.2.1 Ethernet

Ethernet interface: Activates automatically upon connecting an Ethernet cable.

Ethernet is managed by default via the systemd-networkd service, with its configuration file located at /etc/systemd/network/01-eth0.network. Ethernet can be configured for either DHCP or static IP mode. Directly modify the configuration file using the command vim /etc/systemd/network/01-eth0.network. See below for details.

After making changes, check the systemd-networkd service status with the command: `systemctl status systemd-networkd`. The following status indicates the systemd-networkd service is running and enabled.

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
root@T113-lorawanopen:~# systemctl status systemd-networkd
● systemd-networkd.service - Network Service
     Loaded: loaded (/lib/systemd/system/systemd-networkd.service; enabled; ven>
     Active: active (running) since Tue 2023-03-28 01:54:10 CST; 1 years 11 mon>
TriggeredBy: ● systemd-networkd.socket
       Docs: man:systemd-networkd.service(8)
   Main PID: 2015 (systemd-network)
     Status: "Processing requests..."
      Tasks: 1 (limit: 1106)
     Memory: 2.2M
     CGroup: /system.slice/systemd-networkd.service
             └─2015 /lib/systemd/systemd-networkd

Mar 17 13:20:03 T113-lorawanopen systemd-networkd[2015]: veth96e1d54: Gained IP>
Mar 17 13:20:03 T113-lorawanopen systemd-networkd[2015]: veth107eda6: Gained IP>
Mar 17 13:20:04 T113-lorawanopen systemd-networkd[2015]: vethf439e74: Gained IP>
```

Disable systemd-networkd service command: `systemctl disable systemd-networkd`. After disabling, this service will not start automatically at system boot and must be started manually.

Enable systemd-networkd service command: `systemctl enable systemd-networkd`. After enabling, this service will start automatically at system boot.

Configuration Guide for DHCP and Static IP:

(1) Configuring Ethernet for DHCP

Edit the parameters in the file `vim /etc/systemd/network/01-eth0.network` as shown below to enable DHCP mode. When the service runs, the Ethernet interface will automatically enter DHCP mode and obtain a random available IP address. After saving the modified parameters, restart the network service using the command `systemctl restart systemd-networkd`. Use the command `ifconfig eth0` to view the current Ethernet IP address.

```
[Match]
Name=eth0

[DHCP]
ClientIdentifier=mac

[Network]
DHCP=ipv4
```

(2) Configuring a Static IP Address for Ethernet

Use the command `vim /etc/systemd/network/01-eth0.network` to modify the parameters as shown below, setting it to static IP mode. When the service runs, the Ethernet interface will use the fixed IP address (Note: The subnet specified in the `Address` configuration matches the subnet of the network the device connects to via the Ethernet cable). After saving the modified parameters, restart the network service using the command `systemctl restart systemd-networkd`. Use the command `ifconfig eth0` to view the current Ethernet IP address.

```
root@T113-lorawanopen:~# vim /etc/systemd/network/01-eth0.network
[Match]
Name=eth0

[Network]
Address=192.168.1.124/24
Gateway=192.168.1.1
DNS=192.168.1.1
```

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

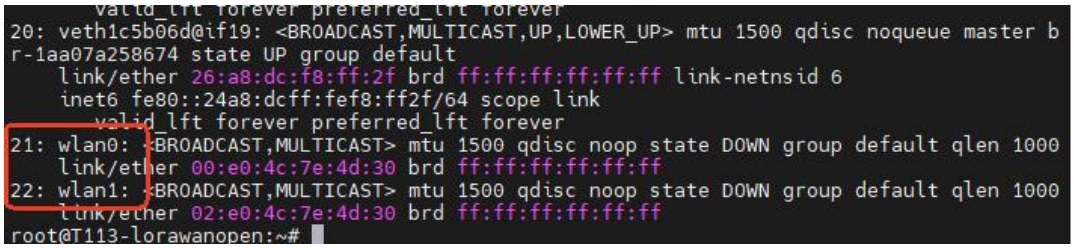
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.2.2 WIFI

When using the WiFi function, connect the WiFi antenna to ensure a stable WiFi signal. This device only supports 2.4GHz WiFi.

Using WiFi requires installing the driver. The relevant driver is pre-installed at the factory. Driver location: /root/lorawanopen/network/wifi/rtl8723du/8723du.ko

Load the driver using the following command: insmod /root/lorawanopen/network/wifi/rtl8723du/8723du.ko. After loading, use the ip addr command to verify if the corresponding network interface is active. The appearance of wlan0 and wlan1 indicates successful driver installation.



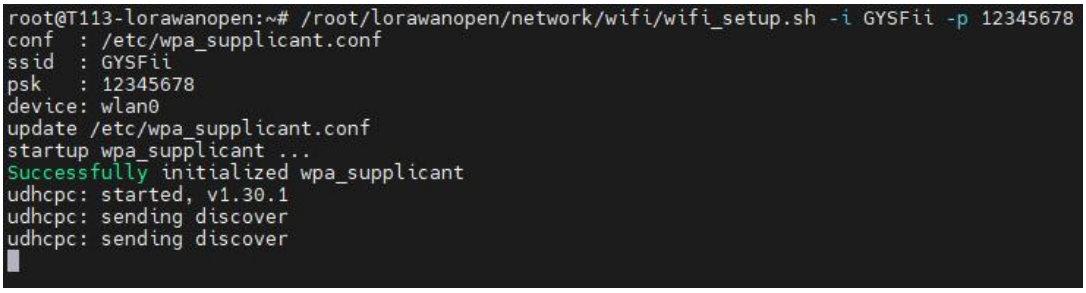
To remove an installed WiFi driver, use the command `rmmod 8723du`. After removal, the corresponding wlan0 and wlan1 interfaces will also be removed.

Connecting to WiFi (two methods):

(1) Connect with parameters:

/root/lorawanopen/network/wifi/wifi_setup.sh -i <wifi name> -p <wifi password>

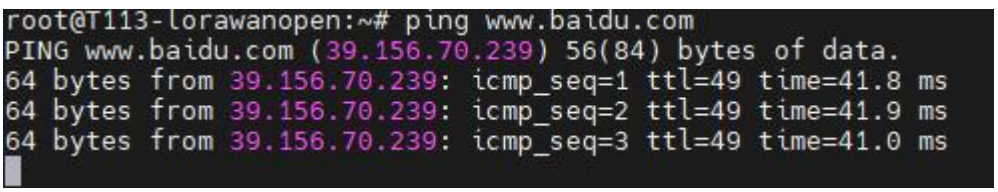
After connecting to WiFi, wait a short while and use `ifconfig wlan0` to check if the wlan0 network interface has an IP address.



(2) Parameter-free connection: /root/lorawanopen/network/wifi/wifi_setup.sh

Note: Before using the parameter-free connection, first establish a connection with parameters. After connecting with parameters, the username and password will be stored in /etc/wpa_supplicant.conf. Subsequent parameter-free connections will read the WiFi credentials from this file by default. After connecting to WiFi, wait a short while and use `ifconfig wlan0` to check if the wlan0 network interface has an IP address.

After connecting to the network via Ethernet/WiFi, you can verify internet access by pinging www.baidu.com. If the result matches the image below, internet access is functioning normally.



Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.3 Script Usage

In this section, we will introduce and explain the gateway, server, and Node-RED scripts.

3.3.1 Gateway Script

Script file: /root/lorawanopen/gateway/gw.sh

Quick command: gw

Script help (prompt description): gw -h (gw --help)

```
root@T113-lorawanopen:~# gw -h

Usage:
  gw {start|stop|restart|status|enable|disable|switch}

Commands:
  status     Show the status of gateway service
  start      Start gateway service
  stop       Stop gateway service
  restart    Restart gateway service
  enable     Enable gateway service on boot
  disable    Disable gateway service on boot
  switch     Switch the gateway regional parameters [1. CN470 2. EU868 3. IN865 4. RU864 5. AS923 6. AU915 7. KR920 8. US915]

Example:
  gw status
  gw switch
```

gw status: View the gateway's operational status.

gw start: Start the gateway service.

gw stop: Stop the gateway service.

gw restart: Restart the gateway service.

gw enable: Enable the gateway service and set it to start automatically at boot.

gw disable: Disable the gateway service and prevent it from starting automatically at boot.

gw switch: Switch the regional frequency band for gateway operation. Eight bands are available. A gateway service restart is required for changes to take effect (the selected band must be supported by the current device for normal operation).

Note: The default eight regional frequency bands correspond as follows:

(1) CN470 → CN470_0

(2) EU868 → EU868

(3) IN865 → IN865

(4) RU864 → RU864

(5) AS923 → AS923

(6) AU915 → AU915_0

(7) KR920 → KR920

(8) US915 → US915_0

If other sub-bands are required, customers must modify the band configuration file (.json file) themselves. The file location is detailed in section “3.2.2 Configuration File”.

3.3.2 Server Script

Script file: /root/lorawanopen/chirpstack/cs.sh

Quick command: cs

Script help: cs -h (cs --help)

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
root@T113-lorawanopen:~/lorawanopen/chirpstack# cs -h

Usage:
  cs {start|stop|restart|status|enable|disable}

Commands:
  status     Show the status of chirpstack service
  start      Start chirpstack service
  stop       Stop chirpstack service
  restart    Restart chirpstack service
  enable     Enable chirpstack service on boot
  disable    Disable chirpstack service on boot
  switch     Switch the server regional parameters [1. CN470 2. EU868 3. IN865 4. RU864 5. AS923 6. AU915 7. KR920 8. US915]

Example:
  cs status
  cs stop
```

cs status: View server running status.

cs start: Start the server.

cs stop: Stop the server.

cs restart: Restart the server.

cs enable: Enable the server for automatic startup.

cs disable: Disable the server and prevent automatic startup.

cs switch: Switch the server's region band (8 available). Changes take effect only after restarting the server (requires stopping and restarting; a direct restart only restarts the service without applying configuration changes).

Note: The default 8 region bands correspond as follows:

(1) CN470 → CN470_0

(2) EU868 → EU868

(3) IN865 → IN865

(4) RU864 → RU864

(5) AS923 → AS923

(6) AU915 → AU915_0

(7) KR920 → KR920

(8) US915 → US915_0

To use other sub-bands, customers must manually modify the band configuration file (.json). The file location is detailed in section “3.2.2 Configuration File”.

3.3.3 Node-RED Script

Script file: /root/lorawanopen/nodered/nr.sh

Shortcut command: nr

Script help: nr -h (nr --help)

```
root@T113-lorawanopen:~/lorawanopen/nodered# nr -h

Usage:
  nr {start|stop|restart|status|enable|disable}

Commands:
  status     Show the status of Node-RED service
  start      Start Node-RED service
  stop       Stop Node-RED service
  restart    Restart Node-RED service
  enable     Enable Node-RED service on boot
  disable    Disable Node-RED service on boot

Example:
  nr status
  nr start
```

nr status: View Node-RED running status.

nr start: Start Node-RED.

nr stop: Stop Node-RED.

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

nr restart: Restart Node-RED.

nr enable: Enable Node-RED for automatic startup.

nr disable: Disable Node-RED and turn off automatic startup.

3.4 Gateway Parameter Configuration

3.4.1 Script Usage

The gateway control script for the LoRaWAN open-source gateway is located at `/root/lorawanopen/gateway/gw.sh`. Its system shortcut command is `gw`. For details on using the script, refer to Section 3.3: Script Usage.

3.4.2 Configuration file

Configuration file location: /root/lorawanopen/gateway/.gwsrc/config/.

The gateway's current regional frequency band configuration is determined by the symbolic link /root/lorawanopen/gateway/.gwsrc/global_conf.json, which points to a specific configuration file within the config folder. This switching is implemented by the gw script. The lorawanopen directory structure is as follows (sections marked “##” denote official LoRaWAN open-source content; customers should review these independently):

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
├── .factory                        #内部测试使用
├── chirpstack                      #chirpstack相关目录
|   ├── .gitattributes              ##
|   ├── .github                     ##
|   ├── .gitignore                  ##
|   ├── LICENSE                     ##
|   ├── Makefile                    ##
|   ├── README.md                   ##
|   ├── configuration               ##
|   ├── cs.sh                       #lorawan开源网关：chirpstack主控制脚本
|   └── docker-compose.yml          ##
├── gateway                         #网关相关目录
|   ├── .gw_sx1302_hal              #SX1302_hal程序完整源码（通过.gwsync脚本从开发平台同步）
|   ├── .gwsrc                      #SX1302_hal程序运行所需文件
|   |   ├── config                  #不同区域频段配置文件目录
|   |   |   ├── 470_CN470.json      ##
|   |   |   ├── 868_EU868.json      ##
|   |   |   ├── 868_IN865.json      ##
|   |   |   ├── 868_RU864.json      ##
|   |   |   ├── 915_AS923.json      ##
|   |   |   ├── 915_AU915.json      ##
|   |   |   ├── 915_KR920.json      ##
|   |   |   └── 915_US915.json      ##
|   |   ├── global_conf.json -> /root/lorawanopen/gateway/.gwsrc/config/868_EU868.json    #全局配置文件软链接，
由gw.sh脚本直接修改
|   |   ├── lora_pkt_fwd            ##
|   |   └── reset_lgw.sh            ##
|   ├── .gwsync                     #从开发平台拉取SX1302_hal程序完整源码到本地
|   └── gw.sh                       #lorawan开源网关：网关主控制脚本
├── network                         #网络配置相关目录
|   └── wifi                        #WiFi相关目录
|       ├── rtl8723du               #RTL8723DU目录
|       |   └── 8723du.ko           #RTL8723DU WiFi驱动内核模块
|       └── wifi_setup.sh           #WiFi启动执行脚本
└── nodered                         #Node-RED相关目录
    ├── nr.sh                       #lorawan开源网关：Node-RED主控制脚本
    ├── package-lock.json           ##
    ├── package.json                ##
    └── working_dir                 #Node-RED工作目录
```

3.5 Server Parameter Configuration

3.5.1 Script Usage

In the LoRaWAN open-source gateway, Chirpstack is used as the server. The control script is located at /root/lorawanopen/chirpstack/cs.sh, with the system shortcut command being cs. For details on script usage, refer to Section 3.3: Script Usage.

3.5.2 Chirpstack Parameter Specifications

ChirpStack in the LoRaWAN open-source gateway is installed via Docker and managed by Docker Compose. The configuration file is located at /root/lorawanopen/chirpstack/docker-compose.yml.

1. Regional Frequency Parameters:

(This step can be automated via a cs script)

When manually switching server regional frequencies is required, open the configuration file using vim /root/lorawanopen/chirpstack/docker-compose.yml. Locate the chirpstack-gateway-bridge service and modify the regional frequency-related fields within the environment variables for the MQTT topic:

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
chirpstack-gateway-bridge:
  image: chirpstack/chirpstack-gateway-bridge:4
  restart: unless-stopped
  ports:
    - "1700:1700/udp"
  volumes:
    - ./configuration/chirpstack-gateway-bridge:/etc/chirpstack-gateway-bridge
  environment:
    - INTEGRATION__MQTT__EVENT_TOPIC_TEMPLATE=cn470_0/gateway/{{ .GatewayID }}/event/{{ .EventType }}
    - INTEGRATION__MQTT__STATE_TOPIC_TEMPLATE=cn470_0/gateway/{{ .GatewayID }}/state/{{ .StateType }}
    - INTEGRATION__MQTT__COMMAND_TOPIC_TEMPLATE=cn470_0/gateway/{{ .GatewayID }}/command/#
  depends_on:
    - mosquitto
```

After making changes, use the `cs` script to stop and then start the service for the changes to take effect. (Note: The `restart` function only restarts the Chirpstack service; modified configurations will not be applied.)

2. Adapting to Regional Frequency Bands:

To enable or add support for a specific regional frequency band in Chirpstack, open the configuration file using the following command: vim /root/lorawanopen/chirpstack/configuration/chirpstack/chirpstack.toml Locate the `enabled_regions` array within the `[network]` section and add the desired frequency bands.

```
# Network related configuration.
[network]

  # Network identifier (NetID, 3 bytes) encoded as HEX (e.g. 010203).
  net_id="000000"

  # Enabled regions.
  #
  # Multiple regions can be enabled simultaneously. Each region must match
  # the 'name' parameter of the region configuration in '[[regions]]'.
  enabled_regions=[
    "as923",
    "as923_2",
    "as923_3",
    "as923_4",
    "au915_0",
    "cn470_0",
    "cn470_10",
    "cn779",
    "eu433",
    "eu868",
    "in865",
    "ism2400",
    "kr920",
    "ru864",
    "us915_0",
    "us915_1",
  ]
```

3.6 Node Device Communication

The device has been shipped with three node devices already configured within the built-in Chirpstack service. Their information is as follows:

Type	Default Band	DevEUI	APPEUI	APPKEY
915	US915	0000000000000924	0000000000000000	7b530db8623ed2a0cfceb02cb1eb2a3e
470	CN470	0000000000000925	0000000000000000	f06675ca596fdb5c54c63b4d6f82748e
868	EU868	0000000000000926	0000000000000000	d9352a74891c5903fff8b014af59ee00

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.6.1 Node Configuration

The above information must be configured into the node device via AT commands before network access. The following example node uses the E78-400TBL-02 (E78-470LN22S(6601) test board) for validation testing. Configuration steps are as follows:

1. Configure DevEUI: AT+CDEVEUI=0000000000000925
2. Configure APPEUI: AT+CAPPEUI=0000000000000000
3. Configure APPKEY: AT+CAPPKEY=f06675ca596fdb5c54c63b4d6f82748e
4. Change region parameters: AT+REGIONCFG=CN470
5. Node network registration: AT+CJOIN=1,1,3,3 (After sending the registration command, wait briefly. Successful registration first returns “OK”, followed by “+CJOIN:OK” after a short delay. Failure displays “FAIL”. Once registered, the node remains online without power cycling and does not require re-registration.)

See the full process below:



3.6.2 Communication Testing

Before communication, ensure the node is connected to the network (this communication test connects to the gateway's built-in server; if not using the gateway's built-in server, connect to the desired server first).

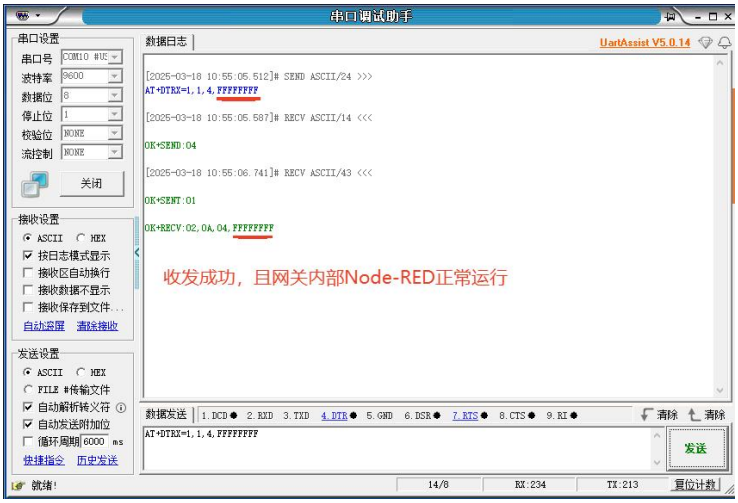
1. Node Device Reporting:

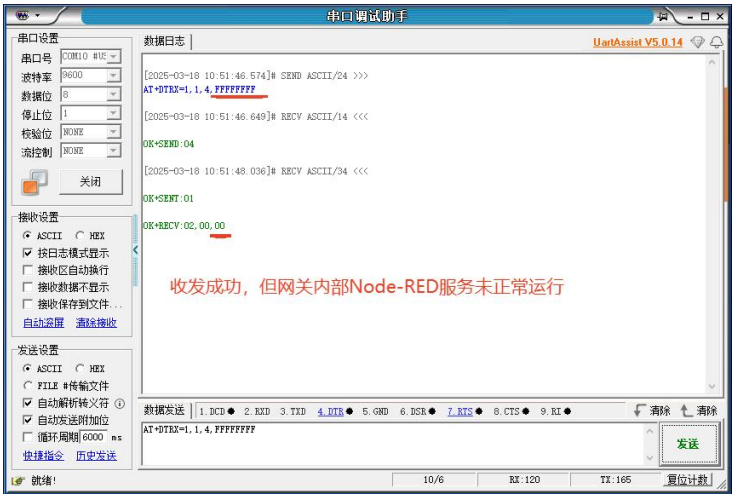
Send the AT command “AT+DTRX=1,1,4,FFFFFFFF” to the node device via the serial port. If transmission and reception are normal, it will display “OK+RECV:xxxxxxx”. If transmission and reception are successful and the RECV message contains the sent information, it indicates that the internal Node-RED service on the gateway is

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

functioning properly. If transmission/reception succeeds but the returned message does not contain the sent data, communication is normal but the internal Node-RED service is not functioning properly. If transmission/reception fails, the display shows “ERR+SENT:xx”, indicating communication failure and that either the server or gateway program within the gateway is not running correctly.







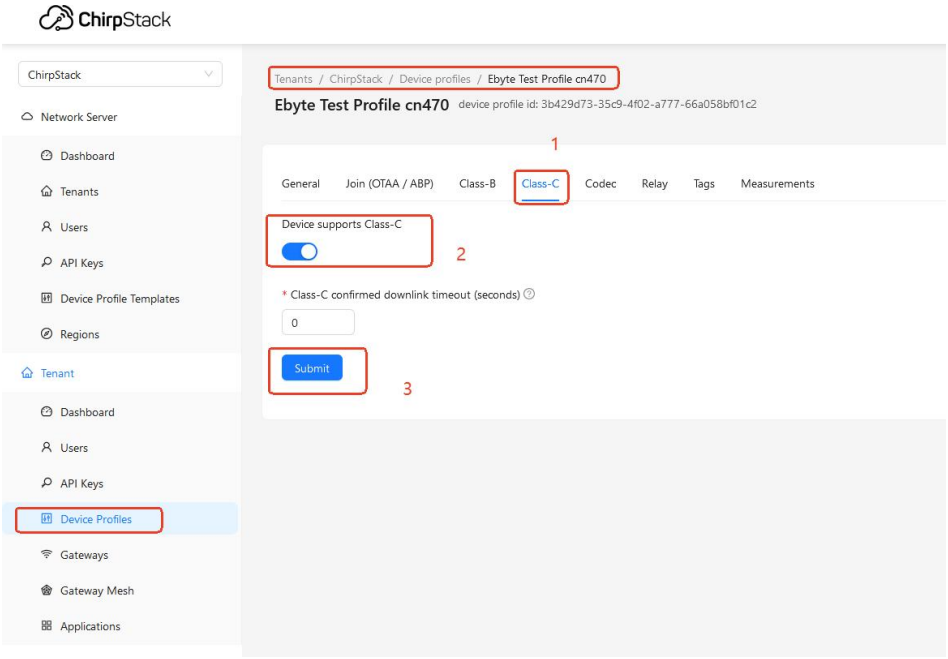
2. Server Deployment:

The device comes pre-installed with a Chirpstack server. Access the Chirpstack server interface by querying the device's current IP address followed by “:8080”. For example, if the device's IP address is 192.168.1.10 (refer to Section 3.2 “Network Service Configuration” for IP address query methods), enter 192.168.1.10:8080 in your

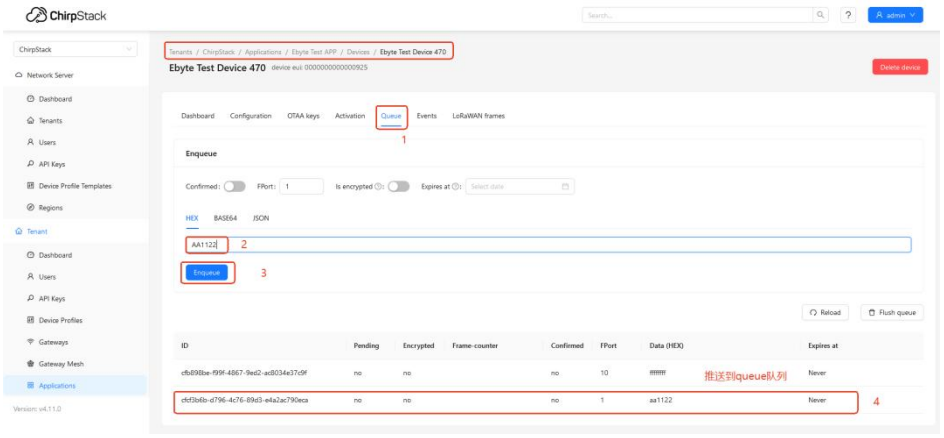
Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

browser to access the Chirpstack server interface. The default username and password are both admin. To test server-initiated push notifications, set the node device to CLASS C mode using the AT command “AT+CCLASS=2”. Additionally, enable CLASS C mode on the gateway for the node.



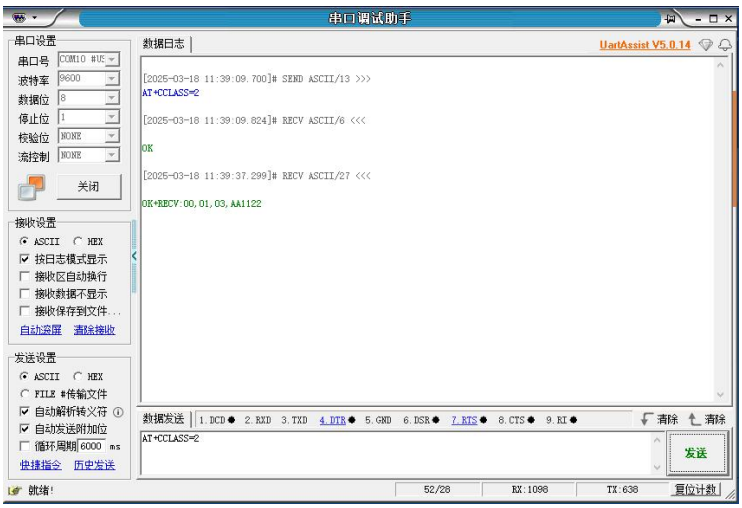
In Chirpstack, select the device you wish to actively send to, enter the content to be sent in the Queue field, then click “Enqueue.” The content will be pushed to the queue.



The contents in the queue are actively sent by the server to the gateway, which then pushes them to the node device. If serial port software is connected, RECV messages can be observed.

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



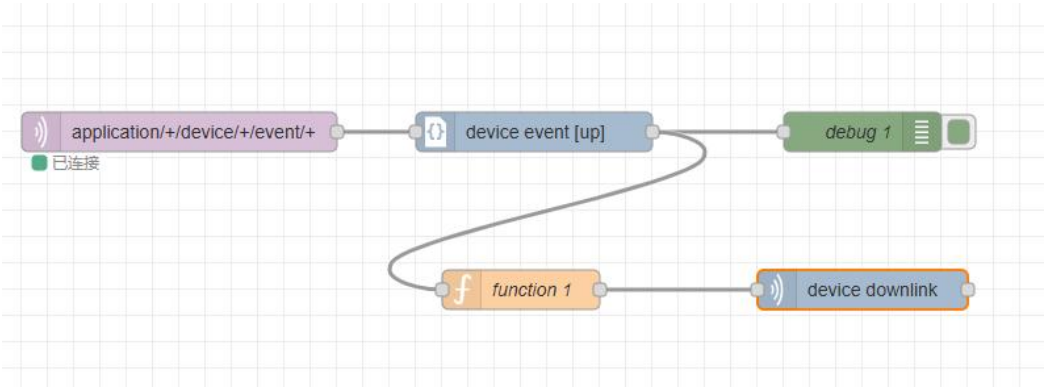
3.7 Node-RED LoRaWAN Node

3.7.1 Node Configuration

The Node-RED control script for the LoRaWAN open-source gateway is located at /root/lorawanopen/nodered/nr.sh. Its system shortcut command is nr. For details on using the script, refer to Section 3.3: Script Usage.

3.7.2 Chirpstack-related nodes

The device comes pre-installed with Node-RED. Access the Chirpstack server interface by entering the device's current IP address followed by “:1880”. For example, if the device's IP address is 192.168.1.10 (refer to Section 3.2 “Network Service Configuration” for IP address lookup methods), enter 192.168.1.10:1880 in your browser to access the Node-RED interface. The LoRaWAN open-source gateway's Node-RED comes pre-installed with Chirpstack-related nodes. Pre-configured examples:



Test result: After sending a message, the node device receives an identical response message.

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
[2025-03-18 10:55:05.512]# SEND ASCII/24 >>>
AT+DTRX=1,1,4,FFFFFFFF

[2025-03-18 10:55:05.587]# RECV ASCII/14 <<<

OK+SEND:04

[2025-03-18 10:55:06.741]# RECV ASCII/43 <<<

OK+SENT:01

OK+RECV:02,0A,04,FFFFFFFF
```

1. Device Event Node:

The Device Event Node serves as an event trigger node, capable of decoding and filtering MQTT events. It is typically used in conjunction with the MQTT In Node.

Uplink: Events triggered when a device sends data to the network.

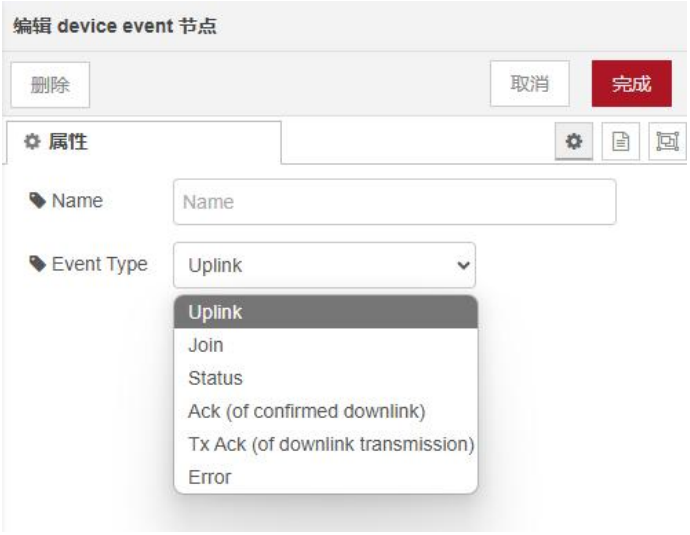
Join: Events sent when a device establishes its initial connection via the LoRaWAN network. This is usually triggered when the device joins the network.

Status: Events indicating updates to the device's status, such as connection status information.

Ack (of confirmed downlink): Event confirming the device has received a downlink message. Typically triggered for acknowledgment-type downlink messages to ensure successful data reception.

Tx Ack (of downlink transmission): Event confirming the device has successfully received and responded to a downlink packet from the gateway.

Error: Event triggered when the device encounters an error, potentially including communication failures, transmission issues, etc.



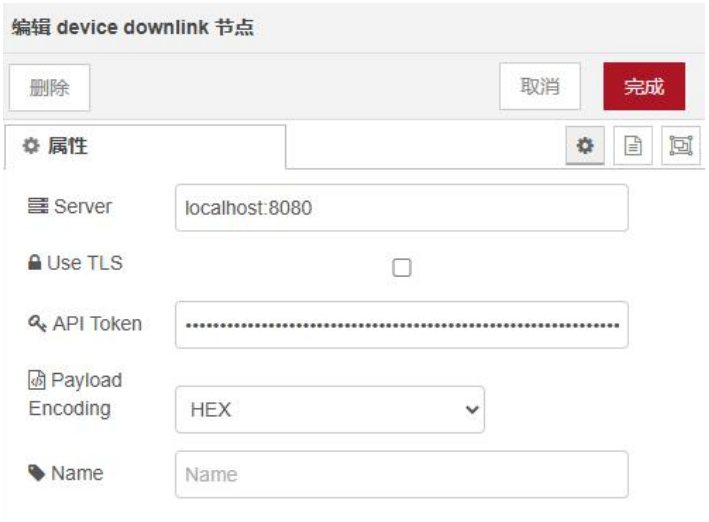
2. Device Downlink Node:

The Device Downlink Node can send messages to the LoRaWAN network via the ChirpStack API. Messages are delivered to the queue.

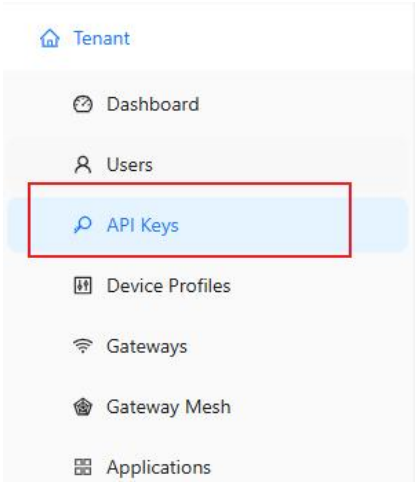
The Server connects to the ChirpStack server. Here, it connects directly to the local server within the LoRaWAN open-source gateway, hence the port number is 8080.

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



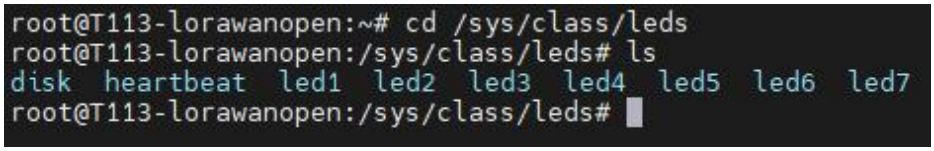
To obtain the API Token, first navigate to the Tenant section on the left side of the Chirpstack management page. Under API Keys, click Add API Key to generate a random token. Copy this token and paste it into the node information.



3.8 Use of Peripheral Devices

3.8.1 LED

The LoRaWAN open-source gateway features 8 LEDs, one of which serves as the power indicator (PWR). The remaining LEDs are user-configurable. Their path is: /sys/class/leds/. Access the LED device folder using the command “cd /sys/class/leds/”. Control each LED via the brightness file within its respective folder. (Note: disk and heartbeat are pre-configured by the driver. User-configurable LEDs are LED1 through LED7. Refer to the device label for specific correspondences.)



Example:

Command to turn on LED1: echo 1 > /sys/class/leds/led1/brightness

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Command to turn off LED1: echo 0 > /sys/class/leds/led1/brightness

3.8.2 Button

The LoRaWAN open-source gateway features one user button, connected to pin PE14 (pin number 142). Users can customize its functionality by detecting the pin's electrical state. The pin management path is: /sys/class/gpio/. Use the command “cd /sys/class/gpio/” to navigate to the button device directory.

```
root@T113-lorawanopen:/sys/class/leds# cd /sys/class/gpio/
root@T113-lorawanopen:/sys/class/gpio# ls
export  gpio112  gpiochip0  unexport
root@T113-lorawanopen:/sys/class/gpio#
```

To use a pin, you must export it via the `export` command. The instruction is: `echo 142 > /sys/class/gpio/export` After exporting, a new folder for pin 142 will appear in the gpio directory:

```
root@T113-lorawanopen:/sys/class/gpio# echo 142 > /sys/class/gpio/export
root@T113-lorawanopen:/sys/class/gpio# ls
export  gpio112  gpio142  gpiochip0  unexport
root@T113-lorawanopen:/sys/class/gpio# cd gpio142
root@T113-lorawanopen:/sys/class/gpio/gpio142# ls
active_low  device  direction  edge  power  subsystem  uevent  value  waiting_for_supplier
root@T113-lorawanopen:/sys/class/gpio/gpio142#
```

Example:

Set KEY to input mode:

echo in > /sys/class/gpio/gpio142/direction

Read KEY state:

cat /sys/class/gpio/gpio142/value

(Note: The pin defaults to high level, where value is 1. When the key is pressed, it goes to low level, where value is 0.)

```
root@T113-lorawanopen:~# echo in > /sys/class/gpio/gpio142/direction
root@T113-lorawanopen:~# cat /sys/class/gpio/gpio142/value
1
```

3.9 Cross-compilation

Development host environment architecture: x86_64. LoRaWAN open-source gateway architecture: ARMv7.

3.9.1 Download Toolchain

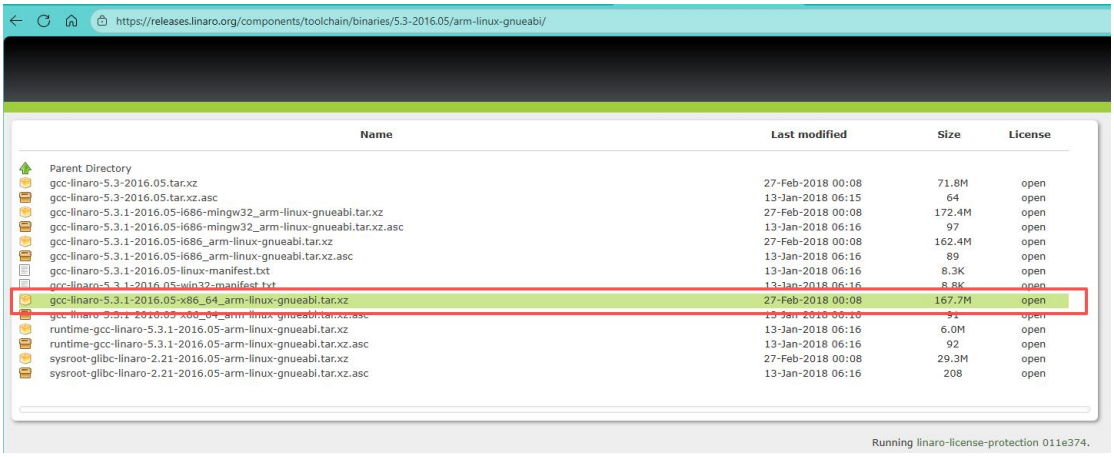
The LoRaWAN open-source gateway system is compiled using the Linaro toolchain version 5.3. To prevent library incompatibility issues, the toolchain version 5.3 is also selected here. Linaro cross-compilation tool download:

https://releases.linaro.org/components/toolchain/binaries/5.3-2016.05/arm-linux-gnueabi/

Note: The above address is the official download source.

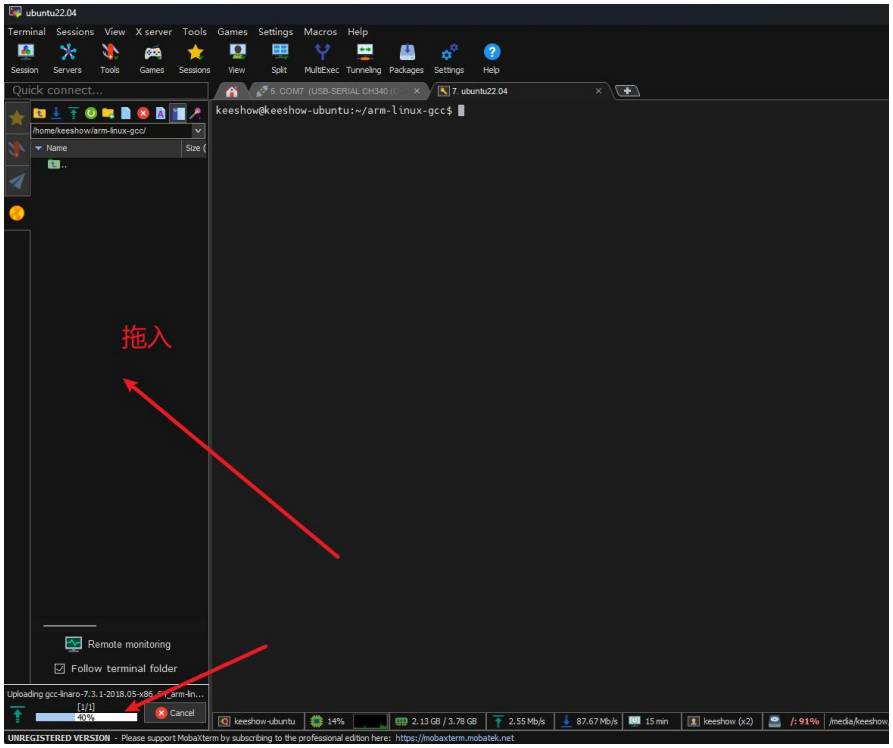
Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

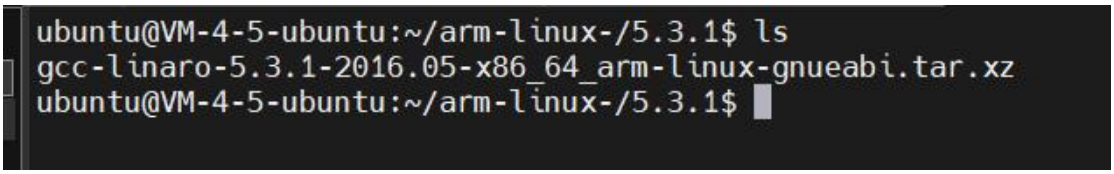


3.9.2 Unzip, install, and verify the toolchain

Unzip, install, and verify the toolchain. Place the downloaded toolchain archive on the development host (if using MobaXterm, simply drag and drop it; the file transfer progress will be visible in the lower-left corner). Once the transfer is complete, you will see the toolchain archive.



After the transfer is complete, you can see the compressed archive of the toolchain.



Use the decompression command to extract the toolchain archive: tar -xvf gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi.tar.xz

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/vstring_util.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/codecvt_specializations.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/typelist.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/rb_tree
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/pool_allocator.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/functional
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/string_conversions.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/debug_allocator.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/iterator
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/mt_allocator.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/alloc_traits.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/rope
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/concurrence.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/hash_map
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/stdio_filebuf.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/random
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/new_allocator.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/cast.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/numeric_traits.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/enc_filebuf.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/aligned_buffer.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/numeric
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/pointer.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/arm-linux-gnueabi/include/c++/5.3.1/ext/throw_allocator.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/include/
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/include/mpf2mpfr.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/include/mpc.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/include/gmpxx.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/include/gdb/
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/include/gdb/jit-reader.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/include/dejagnu.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/include/mpfr.h
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/include/gmp.h
ubuntu@VM-4-5-ubuntu:~/arm-linux-/5.3.1$ ls
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi  gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi.tar.xz
ubuntu@VM-4-5-ubuntu:~/arm-linux-/5.3.1$
```

After extraction is complete, navigate to the bin directory within the toolchain folder. You will find the following toolchain programs:

```
ubuntu@VM-4-5-ubuntu:~/arm-linux-/5.3.1$ cd gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/bin/
ubuntu@VM-4-5-ubuntu:~/arm-linux-/5.3.1/gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/bin$ ls
arm-linux-gnueabi-addr2line  arm-linux-gnueabi-elfedit    arm-linux-gnueabi-gcc-ranlib  arm-linux-gnueabi-ld       arm-linux-gnueabi-readelf
arm-linux-gnueabi-ar         arm-linux-gnueabi-g++        arm-linux-gnueabi-gcov        arm-linux-gnueabi-ld.bfd   arm-linux-gnueabi-size
arm-linux-gnueabi-as         arm-linux-gnueabi-gcc        arm-linux-gnueabi-gcov-tool   arm-linux-gnueabi-nm       arm-linux-gnueabi-strings
arm-linux-gnueabi-c++        arm-linux-gnueabi-gcc-5.3.1  arm-linux-gnueabi-gdb         arm-linux-gnueabi-objcopy  arm-linux-gnueabi-strip
arm-linux-gnueabi-c++filt    arm-linux-gnueabi-gcc-ar     arm-linux-gnueabi-gfortran    arm-linux-gnueabi-objdump  gdbserver
arm-linux-gnueabi-cpp        arm-linux-gnueabi-gcc-nm     arm-linux-gnueabi-gprof       arm-linux-gnueabi-ranlib   runtest
ubuntu@VM-4-5-ubuntu:~/arm-linux-/5.3.1/gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/bin$
```

Use `pwd` to locate the toolchain program directory, then add this path to your environment variables with the following command:

export PATH=$PATH:/home/ubuntu/arm-linux-/5.3.1/gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/bin

After configuration, verify the toolchain version. Successful execution indicates a successful installation. Use the following command:

arm-linux-gnueabihf-gcc -v

```
ubuntu@VM-4-5-ubuntu:~/arm-linux-/5.3.1/gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/bin$ arm-linux-gnueabi-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gnueabi-gcc
COLLECT_LTO_WRAPPER=/home/ubuntu/arm-linux-/5.3.1/gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/bin/../libexec/gcc/arm-linux-gnueabi/5.3.1/lto-wrapper
Target: arm-linux-gnueabi
Configured with: /home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg/target/arm-linux-gnueabi/snapshots/gcc-linaro-5.3-2016.05/co
nfigure SHELL=/bin/bash --with-mpc=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg/target/arm-linux-gnueabi/_build/builds/des
tdir/x86_64-unknown-linux-gnu --with-mpfr=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg/target/arm-linux-gnueabi/_build/bui
lds/destdir/x86_64-unknown-linux-gnu --with-gmp=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg/target/arm-linux-gnueabi/_bui
ld/builds/destdir/x86_64-unknown-linux-gnu --with-gnu-as --with-gnu-ld --disable-libstdcxx-pch --disable-libmudflap --with-cloog=no --with-ppl=no --with-isl=n
o --disable-nls --enable-c99 --with-tune=cortex-a9 --with-arch=armv7-a --with-fpu=vfpv3-d16 --with-float=softfp --with-mode=thumb --disable-multilib --enable-
multiarch --with-build-sysroot=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg/target/arm-linux-gnueabi/_build/sysroots/arm-l
inux-gnueabi --enable-lto --enable-linker-build-id --enable-long-long --enable-shared --with-sysroot=/home/tcwg-buildslave/workspace/tcwg-make-release/label/d
ocker-trusty-amd64-tcwg/target/arm-linux-gnueabi/_build/builds/destdir/x86_64-unknown-linux-gnu/arm-linux-gnueabi/libc --enable-languages=c,c++,fortran,lto --
enable-checking=release --disable-bootstrap --build=x86_64-unknown-linux-gnu --host=x86_64-unknown-linux-gnu --target=arm-linux-gnueabi --prefix=/home/tcwg-bu
ildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg/target/arm-linux-gnueabi/_build/builds/destdir/x86_64-unknown-linux-gnu
Thread model: posix
gcc version 5.3.1 20160412 (Linaro GCC 5.3-2016.05)
ubuntu@VM-4-5-ubuntu:~/arm-linux-/5.3.1/gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/bin$
```

3.9.3 Cross-Compilation Demonstration

This demonstrates writing a simple program on the development host that can control the flashing of LED1 on the LoRaWAN open-source gateway. The demonstration code is as follows:

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(void) {
    FILE *fp;
    int state = 0;

    while(1) {
        fp = fopen("/sys/class/leds/led1/brightness", "w");
        if (fp == NULL) {
            perror("无法打开LED1");
            return 1;
        }

        state = !state;
        fprintf(fp, "%d", state);
        fclose(fp);

        sleep(2);
    }

    return 0;
}
```

After writing the program, compile it using the arm-linux-gnueabihf-gcc tool with the following command: arm-linux-gnueabi-gcc led1_blink.c -o led1_blink

```
ubuntu@VM-4-5-ubuntu:~/arm-linux-/5.3.1$ ls
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi  gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi.tar.xz  led1_blink.c
ubuntu@VM-4-5-ubuntu:~/arm-linux-/5.3.1$ arm-linux-gnueabi-gcc led1_blink.c -o led1_blink
ubuntu@VM-4-5-ubuntu:~/arm-linux-/5.3.1$ ls
gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi  gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi.tar.xz  led1_blink  led1_blink.c
ubuntu@VM-4-5-ubuntu:~/arm-linux-/5.3.1$
```

After compilation completes, place the executable program into the LoRaWAN open-source gateway system (if lacking execution permissions, use `chmod` to modify them). Then run the program, and you will observe the LoRaWAN open-source gateway's LED1 begin to flash. This indicates successful cross-compilation and library version compatibility.

```
root@T113-lorawanopen:~# ls
led1_blink  lorawanopen
root@T113-lorawanopen:~# chmod +x led1_blink
root@T113-lorawanopen:~# ./led1_blink &
[1] 6913
root@T113-lorawanopen:~#
```

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Revision history

Version	Date	Description	Issued by
1.0	2025/6/11	Initial version	ZYD
1.1	2025/11/25	Updated User Guide	ZYD

About us

Technical support: service@cdebyte.com
Documents and RF Setting download link: www.cdebyte.com
Thank you for using Ebyte products! Please contact us with any questions or suggestions: info@cdebyte.com
---
Web: www.cdebyte.com
Address: B5 Mould Park, 199# Xiqu Ave, High-tech District, Sichuan, China

Chengdu Ebyte Electronic Technology Co.,Ltd.

Copyright ©2012–2025，Chengdu Ebyte Electronic Technology Co.,Ltd.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000