HT-M01

Mini LoRa Gateway

Based on Semtech® SX1301

Ultra-small size, USB, SPI, LoRaWAN protocol support
1. Product Overview

The core chip of the HT-M01 LoRa gateway is the SX1301 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.
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:

```bash
sudo raspi-config
```

![Figure 2-2 Raspberry Pi Command Configuration Interface](image)

Interfacing Options → SPI → Yes
After exiting the configuration interface, enter the following commands in sequence:

```bash
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.
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/picoarasdk/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.

![Figure 2-4 ID read out in USB mode](image)

### 2.3 Installing LoRa-gateway-bridge on Raspberry Pi

依次输入如下命令:

```bash
# Firstly, we need add the address of lora-gateway-bridge to the Raspberry Pi software update source
export DISTRIBUT_ID=`lsb_release -si`
export DISTRIBUT_CODENAME=`lsb_release -sc`
sudo echo "deb https://repos.loraserver.io/${DISTRIBUT_ID,,} ${DISTRIBUT_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:

```bash
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).
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.

```bash
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:

```bash
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.

```bash
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 keysserver.ubuntu.com --recv-keys 1CE2AFD36DBCCA00
sudo echo "deb https://repos.loraserver.io/ubuntu xenial testing" | sudo tee /etc/apt/sources.list.d/oraserver.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:

```toml
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] 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
```
3.5 Installing the LoRa App Server

```bash
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:

  ```toml
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.

  ```toml
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:8000"

# 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
```

![Image](http://heltec.taobao.com)

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.

![Figure 4-1 Checking Service Health Status](image)

Raspberry PI use following command:

```bash
sudo systemctl status lrgateway  # Check if mosquitto 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.

Ubuntu server and Raspberry PI are in the 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).
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.
➤ Enter the name of the company/organization/individual you want in the Organization:

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

➤ 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.
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.
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).
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.

```c
#define IEEE_OUI 0x22, 0x0a, 0x0a
#define HT_M01_DEVICE_EUI (IEEE_OUI, 0x12, 0x34, 0x56)
#define HT_M01_APPLICATION_EUI (0x12, 0x34, 0x56, 0x78, 0x9a, 0xbc)
```

Fig. 4-10
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.
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:

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

Get the result shown in Figure 4-15:

![Mosquitto log information view](image)

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:

```json
{
  "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": []
}
```
After decrypting this “garbage” (as shown in Figure 4-16), it is exactly the same as the data we sent on the node.

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