

SX1262 915M LoRa HAT

Instruction

- This product is a Raspberry Pi expansion board based on SX1268/SX1262 chip, wireless serial port module with LoRa modulation function
- With multi-level relay to achieve ultra-long-distance communication, low power consumption wake-up communication, encrypted transmission, etc.
- This product uses a private protocol and does not support LoRaWAN

Features

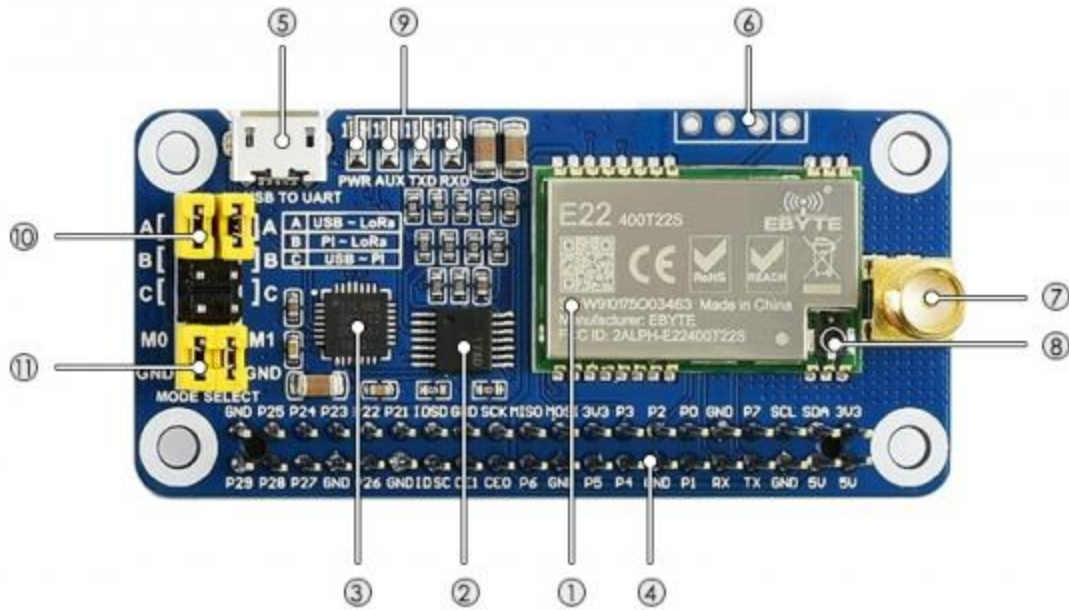
- Standard Raspberry Pi 40PIN GPIO extension header, supports Raspberry Pi series boards
- Onboard CP2102 USB TO UART converter, for serial debugging
- Brings the UART control interface, for connecting host boards like Arduino/STM32
- 4x LED indicators, easy to check the module status
- LoRa spread spectrum modulation technology, up to 81 available signal channel, longer communication distance, more robust to interference
- Auto multi-level repeating, suit for ultra long range communication, allows multi network on the same region
- Low power consumption features like deep sleeping and Wake on Radio, ideal for battery-powered applications
- Customizable communication key which won't be retrieved, greatly improves the security of user data

- Supports LBT, monitoring the signal channel noise before transmitting, greatly improves the success ratio under extreme environment
- Supports RSSI signal intensity indicating, for evaluating signal quality, tuning the network
- Supports wireless parameter configuration, by sending wireless command/data packet, remotely configure or retrieve the module parameter
- Supports fixed-point transmission, broadcast, signal channel monitor
- Comes with development resources and manual (examples for Raspberry Pi/STM32)

Specification

Specification of SX1268 433M LoRa HAT		
Consumption	Transmit Current	110mA (Transient current)
	Receive Current	11mA
	Sleep Current	2uA (LoRa module deep sleep)
MAX Transmit Power		22.0dBm(10, 13, 17, 22dBm Selectable)
Transmit Length		240 Bytes (32, 64, 128, 240 Bytes Selectable)
Buffer		1000 Bytes
Working bands		410.125 ~ 850.125MHz
Receive Sensitivity		147dBm@0.3Kbps (On air)
Interface		UART
Range		5KM(Sunny day; open area; Antenna: AUX 5dBi, Height 2.5m; Air Speed: 2.4kbps)
Working voltage		5V
Logic voltage		3.3V
Working Temperature		40 ~ 85°C

Hardware description



Hardware description of SX1268 LoRa HAT

1. SX1268/SX1262 LoRa module
2. 74HC125V: voltage level translator
3. CP2102: USB TO UART converter
4. Raspberry Pi GPIO connector: for connecting with Raspberry Pi
5. USB TO UART port
6. UART header: for connecting host boards like STM32/Arduino
7. SMA antenna connector
8. IPEX antenna connector
9. Indicators:
 - RXD/TXD: UART RX/TX indicator
 - AUX: auxiliary indicator
 - PWR: power indicator
10. UART selection jumpers
 - A: control the LoRa module through USB TO UART

- B: control the LoRa module through Raspberry Pi
 - C: access Raspberry Pi through USB TO UART
11. LoRa mode selection jumpers
- short M0, short M1: transmission mode
 - short M0, open M1: configuration mode
 - open M0, short M1: WOR mode
 - open M0, open M1: deep sleep mode

【Note】

Mode 0: Transmission mode, Module transmit data when users send data to UART interface. Wireless receiving is enabled to receive data and send to UART interface when idle.

Mode 1: When it is defined to Transmit, user need to add wakeup codes before transmitting, receiving is same as Mode 0.

Mode 2: Wireless transmit and wireless receive are disabled, users can configure configuration according to [#Registers Configuration](#)

Mode 3: Wireless transmit and wireless receive are disabled, module enter deep sleep mode. Module will configure when switching to other modes.

Use Guides

Registers Configuration

If the module is set to configuration mode (Mode 2), you can configure registers according to table below. (Baud rate: 9600, 8N1)

Configure registers



Command format 0xC0 + Begin address + Length + Data

Configure registers	Answer format 0xC1 + Begin address + Length + Data				
	Examples 1: Set channel to 0x11				
		Head	Begin address	Length	Data
	Command	0xC0	0x05	0x01	0x11
	Answer	0xC1	0x05	0x01	0x11
	Examples 2: Set Module address (0x1234), NETID(0x00), UART(9600, 8N1), Air speed(1.2K)				
		Head	Begin address	Length	Data
	Command	0xC0	0x00	0x04	0x12 0x34 0x00 0x61
	Answer	0xC1	0x00	0x04	0x12 0x34 0x00 0x61
	Read registers	Command format 0xC1 + Begin address + Length			
Answer format 0xC1 + Begin address + Length + Data					
Examples 1: Read channel					
		Head	Begin address	Length	Data
Command		0xC1	0x05	0x01	
Answer		0xC1	0x05	0x01	0x12 0x34 0x00 0x61
Examples 2: Read Module address, NETID, Serial port and air speed.					
		Head	Begin address	Length	Data
Command		0xC1	0x00	0x04	
Answer		0xC1	0x00	0x04	0x12 0x34

					0x00	
					0x61	
Configure temporary registers	Command format	0xC2 + Begin address + Length + Data				
	Answer format	0xC1 + Begin address + Length + Data				
	Examples 1: Set channel to 0x11					
		Head	Begin address	Length	Data	
	Command	0xC2	0x05	0x01	0x11	
	Answer	0xC1	0x00	0x01	0x11	
	Examples 2: Set module address(0x1234), NETID(0x00), serial port(9600 8N1), Air speed(1.2K)					
		Head	Begin address	Length	Data	
	Command	0xC2	0x00	0x04	0x12 0x34 0x00 0x61	
	Answer	0xC1	0x00	0x04	0x12 0x34 0x00 0x61	
Wireless configuration	Command format	0xCF 0xCF + Common command				
	Answer format	0xCD 0xCF + Common answer				
	Examples 1: Wireless set channel to 0x11					
		Wireless Head	Head(common)	Begin address	Length	Data
	Command	0xCF 0xCF	0xC2	0x05	0x01	0x11
	Answer	0xCD 0xCF	0xC1	0x05	0x01	0x11
	Examples 2: Wireless set module (0x1234), NETID(0x00), serial port(9600 8N1), Air speed(1.2K)					
		Wireless Head	Head (common)	Begin address	Length	Data

	Command	0xCF 0xCF	0xC2	0x00	0x04	0x12 0x34 0x00 0x61
	Answer	0xCD 0xCF	0xC1	0x00	0x04	0x12 0x34 0x00 0x61
Format error	Answer of Format error					0xFF 0xFF 0xFF

Description of registers

NO.	Read/Write	Name	Description	Note																								
00H	R/W	ADDH	ADDH (Default:0)	High bits and Low bits of module address.																								
01H	R/W	ADDL	ADDL (Default:0)	Note thant when module address is 0xFFFF. It works as broadcasting and listening address and LoRa module doesn't filter address anymore																								
02H	R/W	NETID	NETID (Default: 0)	Network ID, it is used to distinguish network. If you want to communicating between two modules, you need to set their NETID to same ID																								
03H	R/W	REG0	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 25%;"></td> <td style="width: 25%;"></td> <td style="width: 25%;"></td> <td style="width: 25%;"></td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>Baud rate is 1200</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>Baud rate is 2400</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>Baud rate is 4800</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>Baud rate is 9600 (default)</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>Baud rate is 19200</td> </tr> </table>					0	0	0	Baud rate is 1200	0	0	1	Baud rate is 2400	0	1	0	Baud rate is 4800	0	1	1	Baud rate is 9600 (default)	1	0	0	Baud rate is 19200	<p>The baud rates and parity of modules which are intercommunicating can be different.</p> <p>Generally we recommend you to set their baud rate to same value to avoid data blocking or data losing</p>
0	0	0	Baud rate is 1200																									
0	0	1	Baud rate is 2400																									
0	1	0	Baud rate is 4800																									
0	1	1	Baud rate is 9600 (default)																									
1	0	0	Baud rate is 19200																									

			1	0	1	Baud rate is 38400	
			1	1	0	Baud rate is 57600	
			1	1	1	Baud rate is 115200	
							The serial port of modules which are intercommunicating can set to different paramters
			0	0	8N1 (Default)		
			0	1	8O1		
			1	0	8E1		
			1	1	8N1 (same as 00)		
							The air speed of two modules which are intercommunicating must be same. Higher the speed, smaller the latency and shorter the communicating distance.
			0	0	0	Air speed is 0.3K	
			0	0	1	Air speed is 1.2K	
			0	1	0	Air speed is 2.4K (default)	
			0	1	1	Air speed is 4.8K (default)	
			1	0	0	Air speed is 9.6K	
			1	0	1	Air speed is 19.2K	
			1	1	0	Air speed is 38.4K	
			1	1	1	Air speed is 62.5K	
04H	R/W	REG1					

			0		240 bytes (default)	<p>than the length of divided packet. Data are continuous sent to serial port after receiving;</p> <p>If the size of data transmitted are longer than the length of divided packet. Data are divided and sent to serial port after receiving.</p>
			0	1	128 bytes	
			1	0	64 bytes	
			1	1	32 bytes	
						<p>After enabling, You can send command 0xC0 0xC1 0xC2 0xC3 to read register in Transmit Mode or WOR Mode;</p> <p>Register 0x00: RSSI of current ambient noise Register 0x01: The RSSI of last communicating (current noise of channel is dBm = -RSSI/2);</p> <p>Command format: 0xC0 0xC1 0xC2 0xC3 + Begin address + Read length; Answer format: 0xC1 + Address + Read length + Data;</p> <p>For example: Command: 0xC0 0xC1 0xC2 0xC3 0x00 0x01 Answer: 0xC1 0x00 0x01 RSSI (The address</p>
			0	Disable (default)		
			1	enable		

					should begin from 0x00)	
					<p>The relationship between power and current is non-linear.</p> <p>If power is maximum, the power efficiency is highest.</p> <p>Current will not small in the same proportion of power.</p>	
			0	0		22dBm (default)
			0	1		17dBm
			1	0		12dBm
			1	1		10dBm
05H	R/W	REG2	Channel control (CH) 0-83. 84 channels in total		<p>Actually frequency is $850.125 + CH * 1\text{MHz}$. Default 868.125MHz(SX1262), or $410.125 + CH * 1\text{MHz}$. Default 433.125MHz(SX1268)</p>	
06H	R/W	REG3			After enabling, data sent to serial port is added with a RSSI byte after receiving	
			0	Disable (default)		
			1	Enable		
					When point to point transmitting, module will recognize the first three byte as Address High + Address Low + Channel. and wireless transmit it	
			0	Transparent transmitting (default)		
			1	Point to Point Transmitting		
					If target address is not module itself, module will forward data;	
0	Disable (default)					
1	Enable					

					To avoid data echo, we recommend you to use this function in point to point mode, that is target address is different with source address
					Module will listen before transmit wireless data.
		0	Disable(default)		
		1	Enable		This function can be used to avoid interference, however, it also clause longer latency; The MAX LBT time is 2s, after 2s, data is forced to transmit
					This setting only work for Mode 1;
		0	WOR transmit (default) Module is enabled to receive/transmit, and wakeup code is added to transmitted data.		Receiver waits for 1000ms after receive wireless data and forward, and then enter WOR mode again User can send data to serial port and forward via wireless network during this interval, Every serial byte will refresh this interval time (1000ms); You much send the first byte in 1000ms.
		1	WOR Sender Module is disable to send data. Module is working in WOR listen mode. Consumption is reduced		

							<p>This setting only work for Mode 1;</p> <p>Period is equal to $T = (1 + WOR) * 500ms$; MAX 4000ms, MIN 500ms</p> <p>Longer the Period time of WOR listen, lower the average consumption, however, longer the latency</p> <p>The settings of receiver and sender must be same.</p>	
			0	0	0	500ms		
			0	0	1	1000ms		
			0	1	0	1500ms		
			0	1	1	2000ms		
			1	0	0	2500ms		
			1	0	1	3000ms		
			1	1	0	3500ms		
			1	1	1	4000ms		
07H	W	CRYPT_H	High bytes of Key (default 0)				Only write enable, the read result always be 0;	
08H	W	CRYPT_L	Low bytes of key (default 0)				<p>This key is used to encrypting to avoid wireless data intercepted by similar modules;</p> <p>This key is work as calculation factor when module is encrypting wireless data.</p>	
80H ~ 86H	R	PID	Information of module (7 bytes)				7 bytes data of module information	

【Note】

1. When wireless configuring, you need to firstly configure Modules address, NETID, Air speed and Key of both modules to same values. For example, if Modules A: Module address is 1, NETID is 1, Air speed is 2.4Kbps and Key is 1. Module B:

Module address is 2, NETID is 2, Air speed is 62.5Kbps and Key is 2. If you want to use Module A to configure Module B via wireless network, you need to set Module A to be same as Module B. Then you can use 0xCF 0xCF command to configure Module B via wireless network.

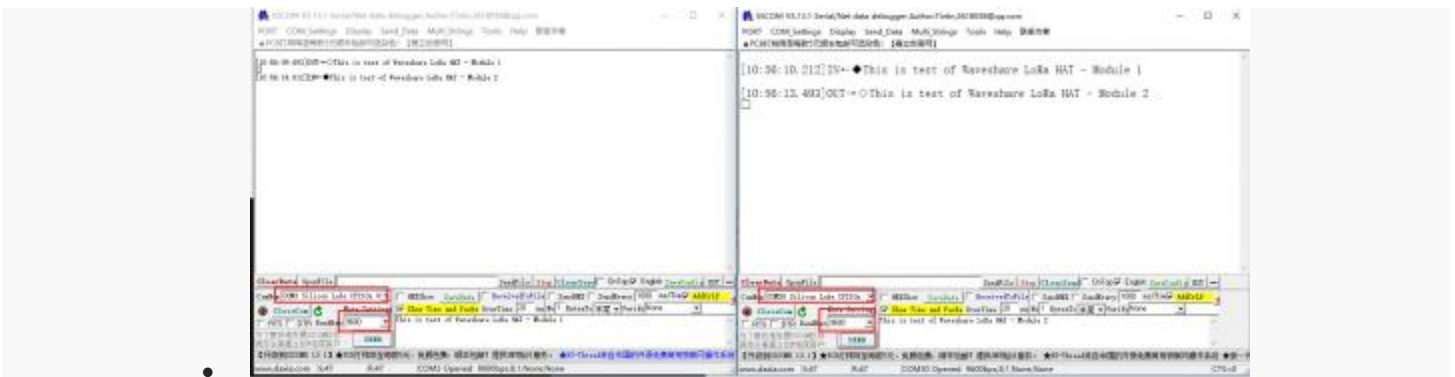
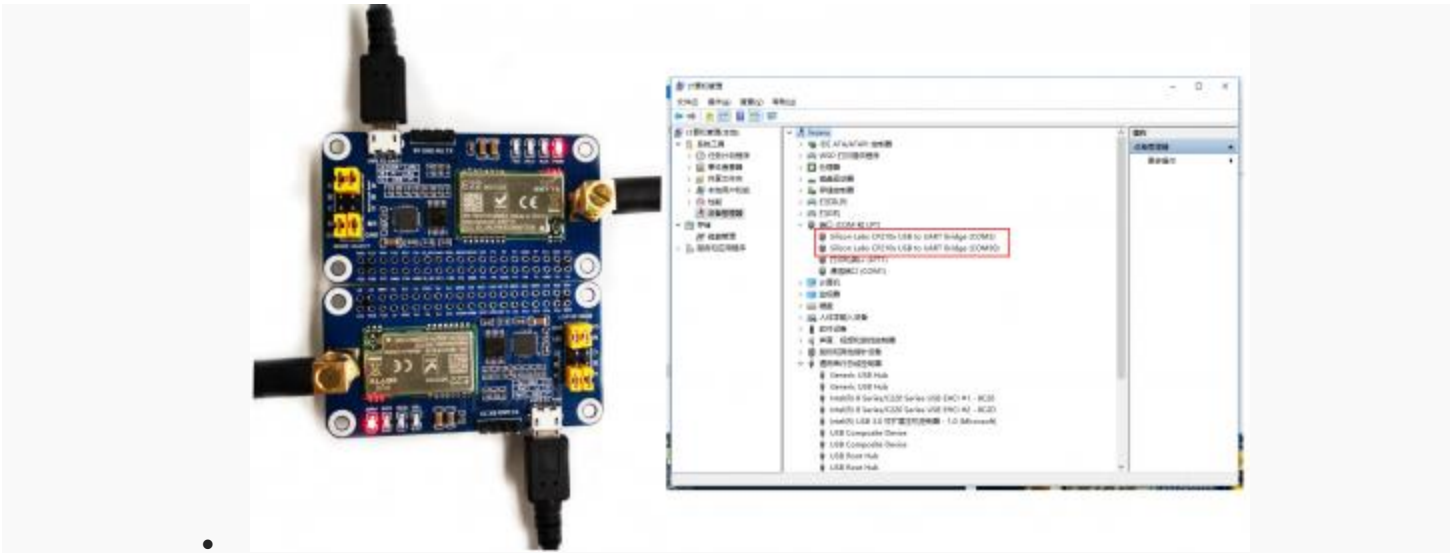
2. After configure temporary registers, LoRa modules work by settings of temporary registers. If LoRa modules restart, the settings of temporary are invalid, LoRa module will re-configure registers according to network settings by 0xC1 commands.

Registers Description

Using with PC

SSCOM connection test

1. To test, you need two SX1268 LoRa HAT (hereafter called LoRa HAT), two micro USB cables.
2. Connect SMA antennas to LoRa HATs. Set jumpers to A and set M0 and M1 to GND.
3. Connect USB to UART interfaces of two LoRa HATs to PC by micro USB cables
4. Check the COM ports on Devices Manager
5. Open SSCOM software, Set serial ports to 9600, 8N1 and try to send data.



Terminal

This test uses a Windows PC to connect to the LoRa HAT, and the jump settings will not change according to the factory location 1. Install python3 in Windows, then enter cmd in the launch bar to search and open the Windows terminal

Enter the path of python3 into the terminal, the default address is generally as shown in the figure below, pay attention to check the user's own python3 path, install pyserial

Use the upper computer software to set up LoRa HAT, unplug the M1 jumper when setting, connect to the M1 jumper and close the serial port

the upper computer after the setting is completed, and set the parameters as follows

Unzip the sample program to the desktop, open the main.py file to modify the COM port, and then run



Working with Raspberry Pi

In this chapter, the example demonstrates 1 use two LoRa HATs to connect two Raspberry Pis for receiving and sending tests, and the example demonstrates 2 using 3 LoRa HATs for relay communication receiving and sending tests.

Hardware connection, install library, enable Raspberry Pi serial port, download sample program

Hardware connection

To test the codes, you need to setup device with RaspberryPi board like below picture,
set jumpers to B and remove M0, M1 jumpers
Powering on Raspberry Pi



Enter the following commands one by one to install the python library, the first command enables the Raspberry Pi serial port

```
sudo raspi-config  
cd /home/pi/documents  
wget https://www.waveshare.com/w/upload/9/9d/SX126X_LoRa_HAT_Demo.zip  
unzip SX126X_LoRa_HAT_Demo.zip
```

2. Enable serial port

Open Terminal of Raspberry Pi

Run command **sudo raspi-config** to open Configure interface

Choose Interfacing Options -> Serial -> No -> Yes

Transparent example

After executing the following command, the node will automatically print to the terminal when it receives the data sent by other nodes. When the node needs to send data to other nodes, press the keyboard i, and then enter the input according to the prompt, as shown in the figure below:

```
cd
~/Documents/SX126X_LoRa_HAT_Code/raspberrypi/python/
sudo python3 main.py
```

A terminal window showing the execution of a Python script on a Raspberry Pi. The prompt is pi@spi4b:~/Documents/sx126x_lora. The user runs 'sudo python3 main.py'. The script outputs instructions: 'Press Esc to exit' and 'Press i to send'. It then receives three messages from address 25 node: 'Hello', 'CPU Temperature:50.147', and 'CPU Temperature:51.121'. Each message is followed by the RSSI value: -59dBm, -60dBm, and -60dBm respectively. The script prompts for input: 'input a string such as 20,Hello World,it will send "Hello World" to node of address 20 please input and press Enter key:'. The user enters '25,Hello World' and a green cursor is visible at the end of the line.

```
pi@spi4b:~/Documents/sx126x_lora $ sudo python3 main.py
Press Esc to exit
Press i to send
receive message from address 25 node
message is b'Hello'
the last receive packet rssi value: -59dBm
receive message from address 25 node
message is b'CPU Temperature:50.147'
the last receive packet rssi value: -60dBm
receive message from address 25 node
message is b'CPU Temperature:51.121'
the last receive packet rssi value: -60dBm
input a string such as 20,Hello World,it will send "Hello World" to node of address 20
please input and press Enter key:25,Hello World
```



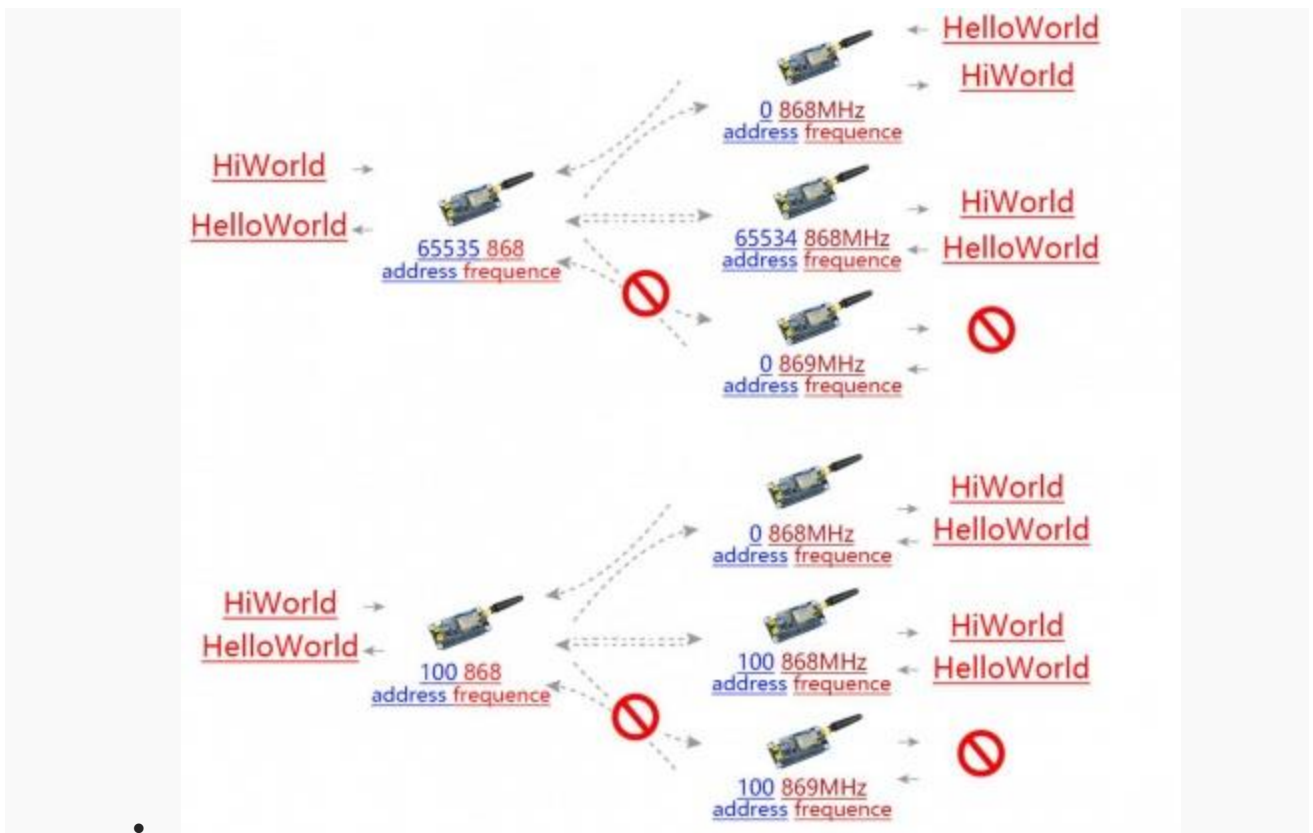
first node

A terminal window showing the execution of a Python script on a Raspberry Pi. The prompt is pi@raspberrypi:~/Documents/test. The user runs 'sudo python3 main.py'. The script outputs instructions: 'Press Esc to exit' and 'Press i to send'. It then receives a message from address 25 node: 'Hello World', followed by the RSSI value: -52dBm. A green cursor is visible at the end of the line.

```
pi@raspberrypi:~/Documents/test $ sudo python3 main.py
Press Esc to exit
Press i to send
receive message from address 25 node
message is b'Hello World'
the last receive packet rssi value: -52dBm
```



second node



transparent transfer diagram

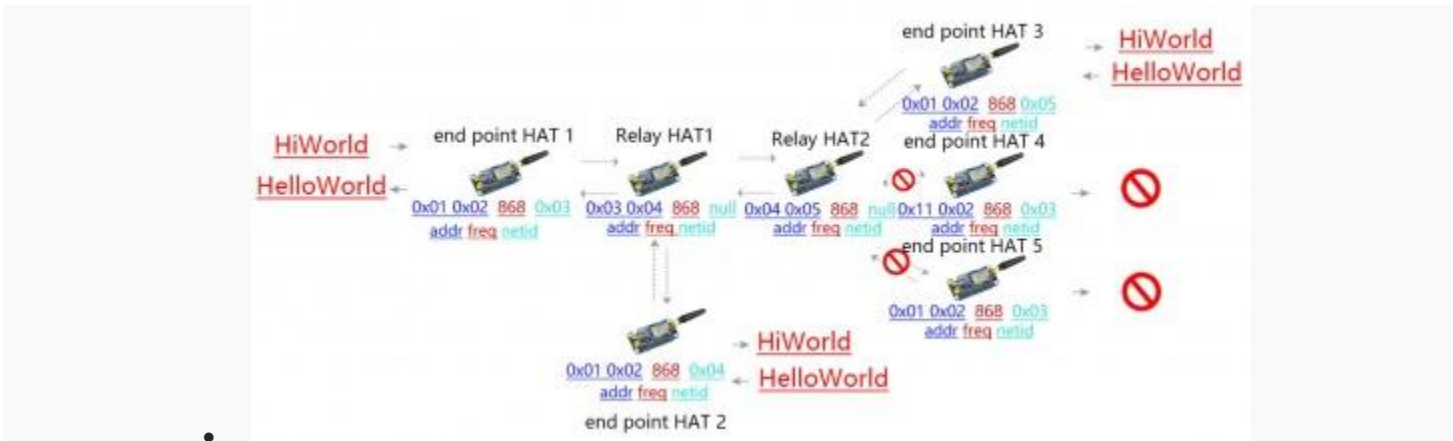
Relay example

After executing the following command, the node will automatically print to the terminal when it receives the data sent by other nodes. When the node needs to send data to other nodes, press the keyboard `i`, and then enter the input according to the prompt, as shown in the figure below:

【Note】 To test Relay example, you require at least three LoRa HATs.

Suppose there is the three LoRa modules, LoRa module A, LoRa module B, and LoRa module C

Use the Windows PC to set the relay LoR module B and LoRa module C. LoRa module A is directly connected to the Raspberry Pi, as shown in the following figure



relay transfer diagram



remove M1 jumper, link to PC and use GUI to set Relay Mode



connect M1 jumper when setting over



lora module B setting



remove M1 jumper link to PC and use GUI to set parameters



connect M1 jumper, when setting over



lora module C setting



lora module A setting

After LoRa module A is connected to the Raspberry Pi, open the main.py file, change line 66, change really=False to

really=True, execute the following command, LoRa module C will print the data from the serial port after receiving the data, and relay the LoRa module B will not print any data from the serial port, enter the following command:

```
cd
~/Documents/SX126X_LoRa_HAT_Code/RaspberryPi/python/
# node = sx126x.sx126x(serial_num =
"/dev/ttyS0",freq=868,addr=0,power=22,rssi=True,air_speed=2400,relay=True)
sudo python3 main.py
```

Using with STM32

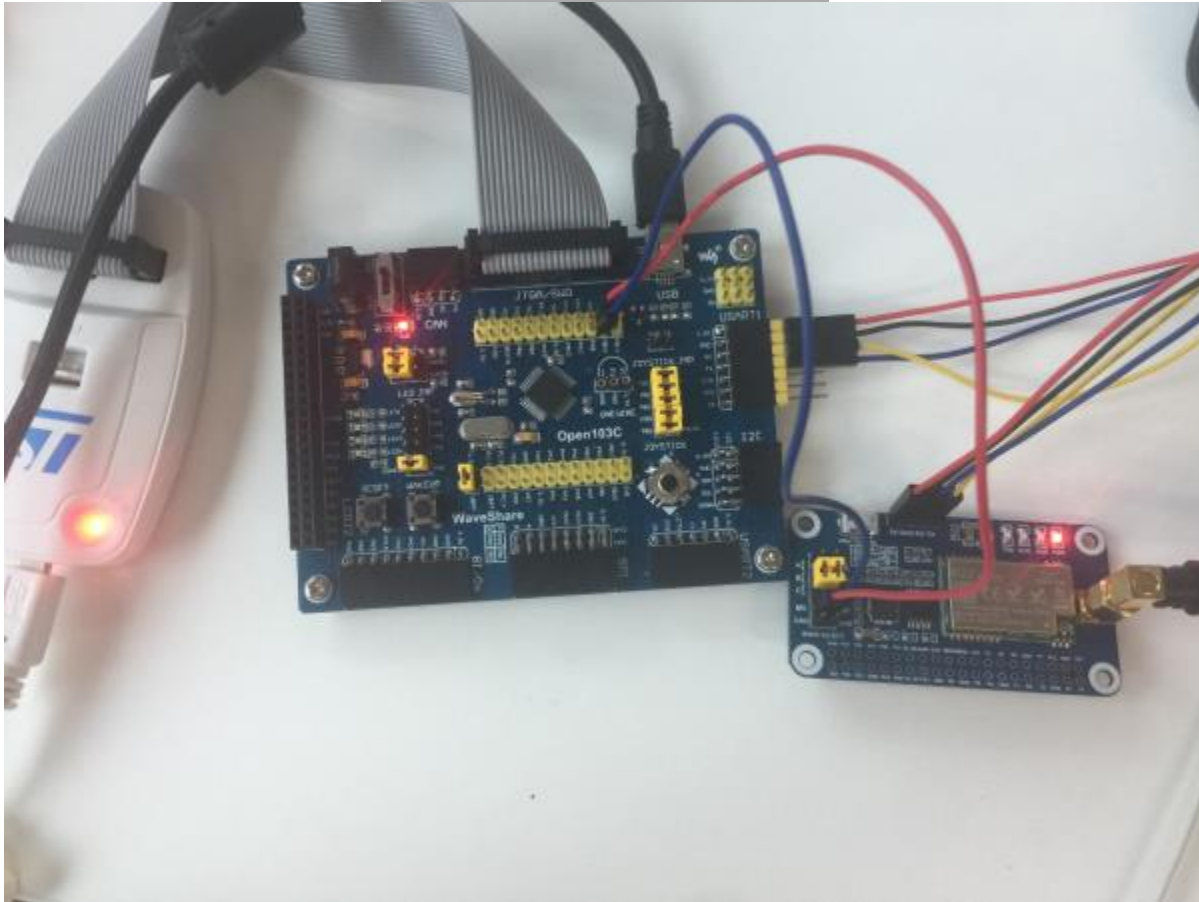
The examples for STM32 are based on [Open103C](#) (STM32F103CBT6). Use HAL Lib.

1. Hardware connection

Set jumpers to B, and connect LoRa HAT to STM32 board pins by pins:

SX1268 LoRa HAT	STM32
5V	5V
GND	GND
RXD	PA10
TXD	PA9

M1	PB15
M0	PB14



2. Expected result

The connection of other LoRa HATs (if required) you can refer to the Raspberry Pi example parts.

Open the project, and modify the definition on main.c file for different communicating modes

```
#define TRASNSPARENT  
//#define RELAY  
//#define WOR
```

Resources

Documents

- [SX1268 LoRa Schematic](#)
- [SX1268 Datasheet](#)

Demo codes

- [SX126X LoRa HAT Code.zip](#)

Softwares

- [Panasonic SDFormatter](#)
- [Win32DiskImager](#)
- [putty](#)
- [SSCOM software](#)
- [RF Setting](#)

If you require technical support, please go to the [Support](#) page and open a tickets.