SIM7028 NB-IoT HAT

From Waveshare Wiki
Jump to: navigation, search
SIM7028 NB-IoT HAT

(https://www.waveshare.com/sim7028-nb-iot-hat.htm)

UART, RPI

Overview

Introduction

The SIM7028 NB-IoT HAT is an NB-IoT (Narrow-Band Internet of Things) development board for Raspberry Pi. It is designed for applications that need low delay, low power, and low throughput, and is also suitable for IoT applications such as meters, remote control, asset tracking, remote monitoring, remote healthcare, bicycle-sharing, and so on.

Features

- Compatible with Raspberry Pi Zero/Zero 2/2B/3B/3B+/4B.
- Supports TCP, UDP, LWM2M, COAP, DTLS, DNS, NTP, PING, HTTP(S), MQTT(S), TLS/SSL, etc.
- Onboard Type-C interface for software debugging.
- Onboard UART interface for AR command transmission and firmware update.
- Onboard control pins for connecting with host boards like Arduino/STM32.
- Onboard USB to UART chip, the UART communication pin can be configured via jumper.
- Onboard nano SIM card slot, compatible with NB-IoT specific card.
- 1x LED indicator, easy to monitor the working status. (the STA indicator normally shows the running status of the HAT, the display can be customized to add the STA indicator).
- Baudrate: 2400bps ~ 460900bps (115200bps by default).
- Control via AT commands (3GPP TS 27.007, 27.005 and SIMCOM enhanced AT Commands).
- Supports SIM application toolkit: SAT Class 3, GSM 11.11, USAT.
- Comes with online development resources and manuals (examples for Raspberry Pi/Jetson Nano/Arduino/ESP32).

Communication Parameters

- Band Support:
  - B1/B2/B3/B4/B5/B8/B12/B13/B14/B17/B18/B19/B20/B25/B26/B28/B66/B70/B85
- Transmission Power:
  - Class 3 (0.25W@LTE)
- Data Rate:
  - UL: ≤159Kbps
  - DL: ≤127Kbps

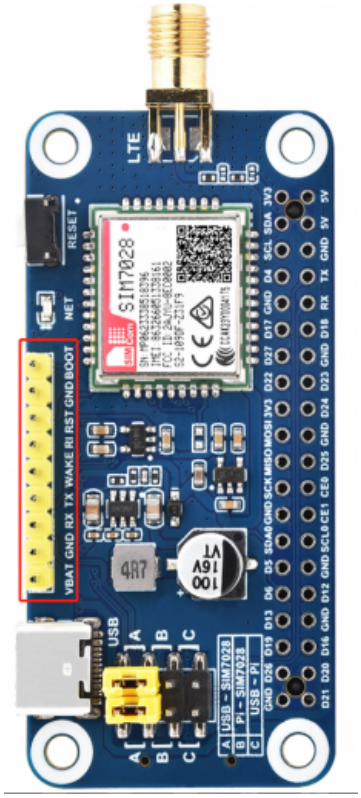
Other Parameters

- Power supply: 2.2V~4.3V
- Logic level: 5V / 3.3V (3.3V by default)
- Standby mode current: 0.8uA

- Sleep mode current: 0.11mA (@DRX=2.56s)
- Operation temperature: -40°C ~ 85°C
- Storage temperature: -45°C ~ 90°C
- Dimensions: 30.2mm x 65mm

Interface Description

Pinout

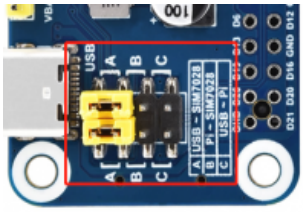


(/wiki/File:SIM7028_yin.png)

Pinout

VBAT	Power input (support 3.7V Li-battery)
GND	Ground
RX	Receive data
TX	Transmit data
WAKE	The low power mode wake-up pin, pull down this WAKE pin to wake up SIM7028. After pulling low this WAKE pin, an AT command must be sent to the module within 10ms, otherwise, it will enter sleep mode.
RI	The RI signal pin is set to a default high level. It will output a low-level pulse of 120ms when receiving a short message or when a URC is sent out.
RESET	Reset pin, pulling it low in the power-on state to reset, but it is invalid in the power-off state.
BOOT	Download control pin, when pulled down, resets the module and allows it to enter download mode.

Jumper Caps

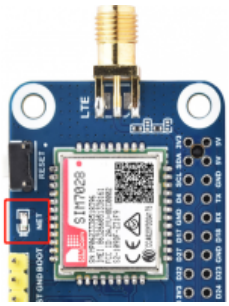


(/wiki/File:SIM7028_230925_03.p
Jumper Caps

A	USB—SIM7028
B	Pi—SIM7028
C	USB—Pi

Indicators

STA	Turn on when power on the module (5V) and GND
NET	64ms on/800ms off——Register the network failed 64ms on/3000ms off——Network registered 64ms on/300ms off——Data transmission off——Power off or PSM/DRX/eDRX sleep mode



(/wiki/File:SIM7028_2309
Indicators

Connect to PCs

Hardware Connection

Before using the SIM7028 module, the user needs to prepare the following items in addition to the Type-C USB cable and LTE antenna:

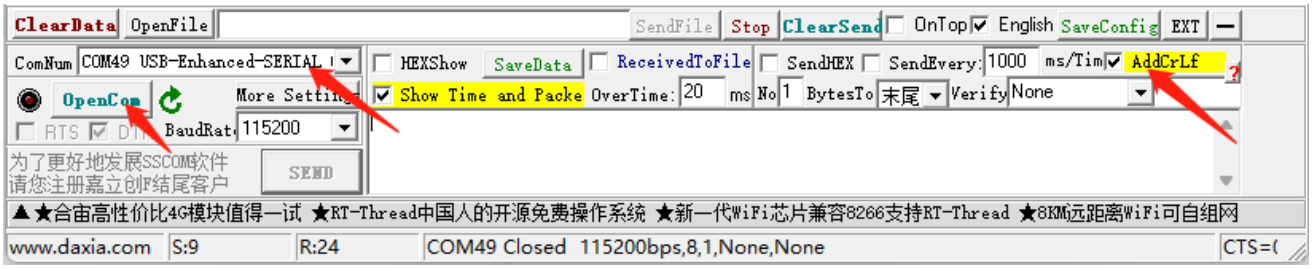
- An NB-IoT dedicated SIM card.

Wiring instructions:
If you plan to use the reserved serial port for TTL communication, you need to disconnect the A jumper cap.
If you are using the Type-C interface, simply keep the A jumper cap connected.

SIM7028-NB-IoT HAT	TTL
5V	5V
GND	GND
RX1	TX
TX1	RX

How to connect:

1. Install the NB-IoT card into the card slot on the back of the module and connect the LTE antenna. (When in use, rotate the LTE antenna to the outside of the board).
2. Connect the Type-C USB cable to the computer's USB port to power the SIM7028 module.
3. Open the provided serial port assistant, select the corresponding serial port, and set the baud rate to 115200. Check the "AddCrLf" option.
4. In the extended settings, you will see the AT commands you want to enter. Click on the corresponding command to send it directly.



(/wiki/File:SIM7028_NB-IoT_HAT_connect01.png)

Basic Networking Test

The following are the common AT commands, you can quickly check whether the AT serial communication of SIM7082 and the network connection are normal.
You should do a simple network test before. Please operate it before checking the network.
A detailed description of the relevant AT commands can be found in the SIM7022 Series_AT Command Manual.

Command	Description	Return value
AT	AT Test Command	OK
ATE	ATE1 turns on echo mode, ATE0 turns off echo	OK
AT+CSQ	Query network signal quality, return signal value	OK
AT+CGMR	Query firmware version	OK
AT+CEREG?	Query network registration status	OK
AT+CGACT?	Query PDP status	OK
AT+COPS?	Query network information	OK
AT+CGCONTRDP	Query network status	OK
AT+CFUN=0	Disable RF	OK
AT+CFUN=1	Enable RF	OK

Communication Test

TCP/IP Communication

Here we introduce the SIM7028 module TCP/IP communication function.
SIM7082 module supports transparent transmission mode (data mode), Push Mode, and Buffered Access Mode in non-transparent transmission mode (Command mode).
SIM7082 TCP/IP is a multi-path client structure by default, supporting 2-ch sockets, TCP or UDP.

Before TCP/IP communication, you should make sure the module's networking is normal according to #Hardware Connection and #Basic Networking Test.
SIM7082 module supports TCP and UDP communication, DNS analysis, and Ping function.

(Note: Detailed description of AT commands can be found in SIM7028 Series TCPIP Application Note V1.04 (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/SIM7028%20Series_TCPIP_Application%20Note_V1.04.pdf), subsequent module firmware upgrade, the corresponding AT commands may be updated.)

【Service】

Related commands:

AT Command	Command Description	Return Value
AT+NETOPEN	Enable Socket service	OK
AT+NETCLOSE	Disable Socket service	OK +NETCLOSE: 0
AT+CIPOPEN	Multi-link Socket service, up to 2 connections	OK
AT+CIPSEND	Send data to the destinated connection (TCP, UDP)	OK
AT+CIPCLOSE	Disable Socket service	OK
AT+SERVERSTART	Enable TCP server	OK
AT+SERVERSTOP	Stop TCP server	OK

【HTTP Client】

Related commands:

AT Command	Command Description	Return Value
AT+HTTPINIT	Enable HTTP service	OK
AT+HTTPPARA=URL, https://www.waveshare.com/ (https://www.waveshare.com/)	Connect to the remote server	OK
AT+HTTPDATA=5,1000	Input data	DOWNLOAD <Type hello OK
AT+HTTPACTION=0	Starting the HTTP request, 0:GET; 1:POST; 2:HEAD; 3:DELETE; 4:PUT	OK +HTTPACTION: 0,200,54
AT+HTTPTERM	Disable HTTP service	OK
AT+HTTPPARA	Setup HTTP parameters	OK
AT+HTTPHEAD	Read HTTP response header information	OK

AT+HTTPREAD	Read HTTP response information	OK

【Self-authored interface testing requests using HTTP POST/GET】

Provide test interface: [POST/GET]: [1] (https://www.waveshare.cloud/api/sample-test/)[2] (http://8.210.171.95:8000/sample-test/).
Here is a simple application case, write POST and GET requests in a single endpoint, and use the "request.method" to differentiate and handle the request methods accordingly.

```
@csrf_exempt
def GET_POST_testapi(request):
    if request.method == 'POST':
        data_value = request.body.decode('utf-8')
        print(data_value)
        if data_value:
            # Storing data into a SQLite database
            data = Data(value=data_value, timestamp=timezone.now())
            data.save()

            max_data_count = 30
            data_count = Data.objects.count()
            if data_count > max_data_count:
                data_to_delete = Data.objects.order_by('timestamp')[:data_count - max_data_count]
                for item in data_to_delete:
                    item.delete()

            return JsonResponse({'message': 'data is saved!'})
        else:
            return JsonResponse({'message': 'Invalid data value'}, status=400)
    elif request.method == 'GET':
        # Access to the latest data
        try:
            latest_data = Data.objects.latest('timestamp')
            response_data = {
                'value': latest_data.value,
                'timestamp': latest_data.timestamp.strftime('%Y-%m-%d %H:%M:%S')
            }
            return JsonResponse(response_data)
        except Data.DoesNotExist:
            return JsonResponse({'message': 'No data available.'})
    else:
        return JsonResponse({'message': 'Invalid request method.'}, status=400)


def get_all_data(request):
    all_data = Data.objects.all().order_by('timestamp')
    timeline = []
    values = []

    for data in all_data:
        try:
            numeric_value = float(data.value)
            timeline.append(data.timestamp.strftime('%Y-%m-%d %H:%M:%S'))
            values.append(numeric_value)
        except ValueError:
            pass  # Ignore values that cannot be converted to numbers

    response_data = {
        'timeline': timeline,
        'value': values
    }

    return JsonResponse(response_data)
```


SSCOM V5.13.1 Serial/Net data debugger,Author:Tintin,2618058@qq.com
PORT COM_Settings Display Send_Data Multi_Strings Tools Help 联系作者 大虾论坛
[19:00:09.020]IN←◆LoadVerifyImageHead read(len=272), Time(ms)->0.
test:VerifyImageHead skip
VerifyImageBody sha256 skip
[19:00:09.211]IN←◆
*ATREADY: 1
[19:00:09.524]IN←◆
SIM Ready
+CEREG: 2
[19:00:14.119]IN←◆
Network Available
+CEREG: 1,"1D30","0A7FD13F",9
[19:00:26.256]OUT→◇AT+HTTPINIT
[19:00:26.262]IN←◆
OK
[19:00:29.406]OUT→◇AT+HTTPPARA="URL","https://www.waveshare.cloud/api/sample-test/"
[19:00:29.417]IN←◆
OK
[19:00:34.790]OUT→◇AT+HTTPACTION=0
[19:00:34.808]IN←◆
OK
[19:00:42.651]IN←◆
+HTTPACTION: 0,200,65
[19:00:44.091]OUT→◇AT+HTTPREAD=0,100
[19:00:44.104]IN←◆
OK
+HTTPREAD: 65
{"value": "hello Waveshare!", "timestamp": "2023-10-11 18:55:51"}
+HTTPREAD: 0
[19:01:40.652]OUT→◇AT+HTTPDATA=18,1000
[19:01:40.661]IN←◆
DOWNLOAD
[19:01:41.852]OUT→◇hello SIM7028 HAT!
[19:01:41.856]IN←◆
OK
[19:01:46.588]OUT→◇AT+HTTPACTION=1
[19:01:46.614]IN←◆
OK
[19:01:53.563]IN←◆
+HTTPACTION: 1,200,29
[19:01:58.327]OUT→◇AT+HTTPREAD=0,100
[19:01:58.340]IN←◆
OK
+HTTPREAD: 29
{"message": "data is saved!"}
+HTTPREAD: 0
[19:02:03.496]OUT→◇AT+HTTPACTION=0
[19:02:03.520]IN←◆
OK
[19:02:10.945]IN←◆
+HTTPACTION: 0,200,67
[19:02:11.885]OUT→◇AT+HTTPREAD=0,100
[19:02:11.903]IN←◆
OK
+HTTPREAD: 67
{"value": "hello SIM7028 HAT!", "timestamp": "2023-10-11 19:01:52"}
+HTTPREAD: 0
ClearData OpenFile SendFile Stop ClearSend OnTop English SaveConfig EXT
ComNum COM49 USB-Enhanced-SERIAL
HEXShow SaveData ReceivedToFile SendHEX SendEvery: 1000 ms/Tim AddCrLf
CloseCom More Settings
Show Time and Packe OverTime: 20 ms No 1 BytesTo 末尾 Verify None
RTS DTR BaudRat 115200
hello SIM7028 HAT!
为了更好地发展SSCOM软件
请您注册嘉立创P结尾客户
SEND
▲★合宙高性价比4G模块值得一试 ★RT-Thread中国人的开源免费操作系统 ★新一代WiFi芯片兼容8266支持RT-Thread ★8KM远距离WiFi可自组网
www.daxia.com S:228 R:600 COM49 Opened 115200bps,8,1,None,None

(/wiki/File:SIM7028_NB-IoT_HAT_001.png)

MQTT Communication

Here we mainly introduce the SIM7028 module and MQTT communication.
For more details about AT commands, you can refer to SIM7028 Series MQTT(S) Application Note V1.03 (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/SIM7028%20Series_MQTT(S)_Application%20Note_V1.03.pdf)

【MQTT Subscribe Topic & Publish Message】

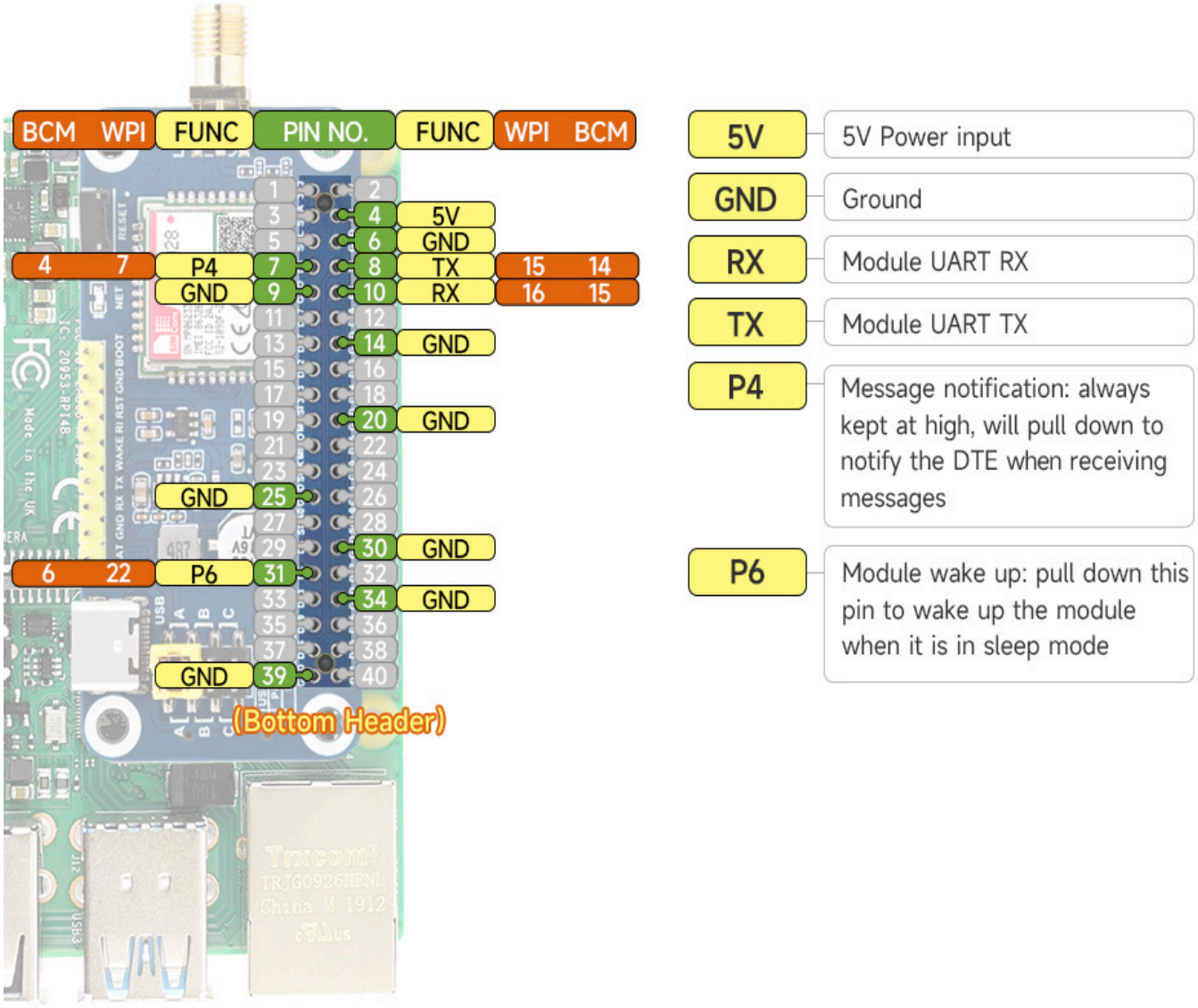
The following is for demonstrating MQTT communication and testing with a private EMQX server connection.
Related commands:

AT Commands	**Command Description**	**Return value**
AT+CMQTTSTART	Enable MQTT service	OK
AT+CMQTTACCQ=0,"Waveshare-Sim7028",0	Apply for MQTT client	OK
AT+CMQTTCONNECT=0,tcp://mqtt.easyiothings.com,20,1	Send MQTT request, connect to the private MQTT server (MQTTS)	OK
AT+CMQTTTOPIC=0,11	Input message publish topic	>sim7028test OK
AT+CMQTTPAYLOAD=0,9	Input the published message	OK >waveshare
AT+CMQTTPUB=0,0,60	Publish message	OK +CMQTTPUB: 0,0
AT+CMQTTSUB=0,4,1	Subscribe to the message topic	>test OK +CMQTTSUB: 0,0 [10:03:39.665] receive ←◆ +CMQTTRXSTART: 0,4,18 +CMQTTRXTOPIC: 0,4 test +CMQTTRXPAYLOAD: 0,18 { "msg": "hello" } +CMQTTRXEND: 0
AT+CMQTTSTOP	Stop MQTT service	OK
AT+CMQTTREL	Release client	OK
AT+CMQTTUNSUBTOPIC	Release subscribed topic	OK
AT+CMQTTUNSUB	Release subscription	OK

Note: the AT response time will be long as the result of the NB-IoT network problems when testing MQTT commands. Please be patient. For more details, you can refer to MQTT.

Working with Raspberry Pi

Hardware Connection



(/wiki/File:SIM7028_NB-IoT_HAT_Rasp01.jpg)

(/wiki/File:SIM7028_NB-

IoT_HAT_Rasp.jpg)
To adjust the jumper cap position to 'B' for the SIM7028 NB-IoT HAT board, please refer to the table below for the Raspberry Pi GPIO pin and module pin connections:

SIM7028 NB-IoT HAT	**Raspberry Pi**
5V	5V
GND	GND
RXD	TXD (Corresponding to BCM 14)
TXD	RXD (Corresponding to BCM 15)
RI	P7 (Corresponding to BCM 4)
WAKE	P31 (Corresponding to BCM 6)

Software Configuration

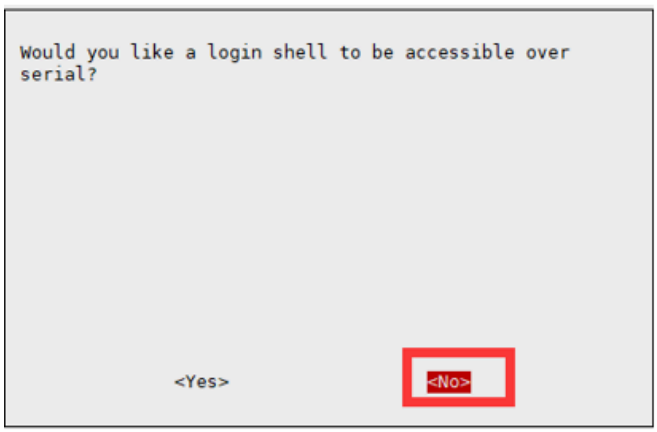
【UART Configuration】

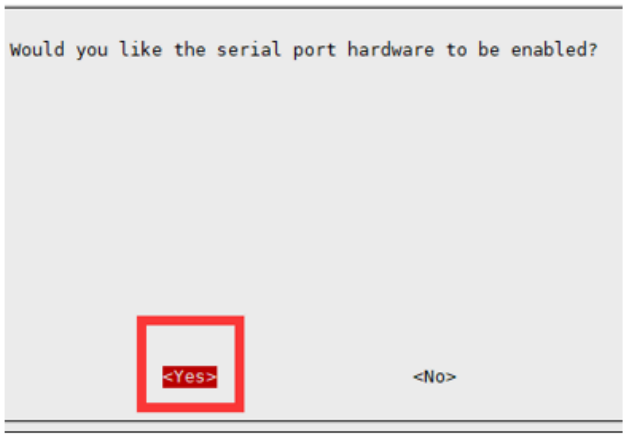
As the serial ports of the Raspberry Pi are for terminal debugging by default, if you need to use the serial port, you must modify the Raspberry Pi setting.

- Execute the following commands to enter the Raspberry Pi configuration:

```
sudo raspi-config
```

- Select Interfacing Option -> Serial -> no -> yes to turn off the UART debugging function.





(/wiki/File:L76X_GPS_Module_rpi_serial.png)

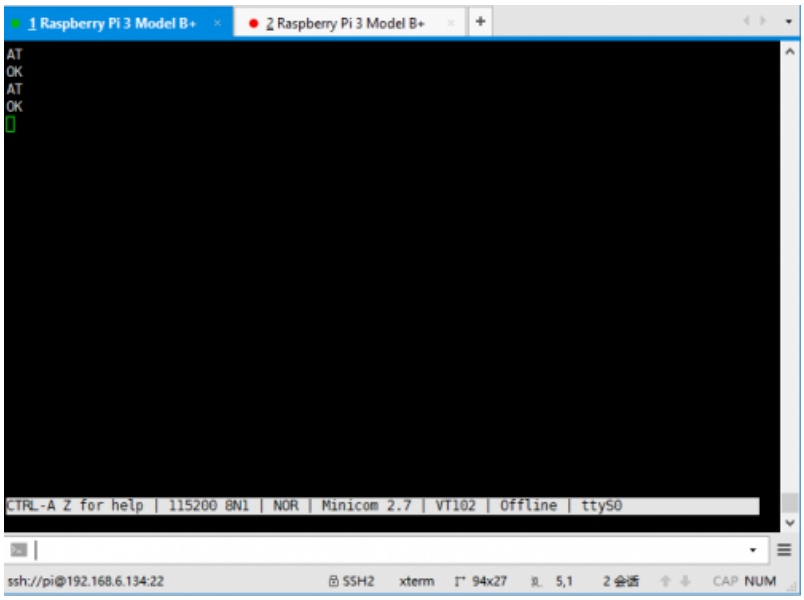
- Reboot it to take effect.

Raspberry Pi Minicom Debugging

Insert the module into the Raspberry Pi, install minicom, and the minicom is the UART debugging tool for Linux:

```
sudo apt-get install minicom
```

Execute minicom -D /dev/ttyS0 to enter the minicom serial port debugging interface.
The default baud rate is 115200, ttyS0 is the serial port of Raspberry Pi 3B/3B+/Zero 2, and ttyAMA0 is the serial port of Raspberry Pi Zero.
Press Ctrl + A, Z and E if you type the data without echo.



(/wiki/File:SIM7028_NB-IoT_HAT_Software01.png)

Raspberry Pi Sample Demo

Download the demo and enter the Raspberry Pi directory:

```
sudo apt-get install unzip -y
wget https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/SIM7028-NB-IoT-HAT-Demo-Code.zip
unzip SIM7028-NB-IoT-HAT-Demo-Code.zip -d SIM7028
cd SIM7028/Raspberry/
sudo pip3 install paho-mqtt pyserial
# HTTP request
sudo python3 AT_http_s.py
# MQTT request: pay attention to modifying the corresponding parameters, this MQTT example is to access the WaveshareCloud platform, please refe
r to the application case for details.
sudo python3 AT_mqtt_s.py
```

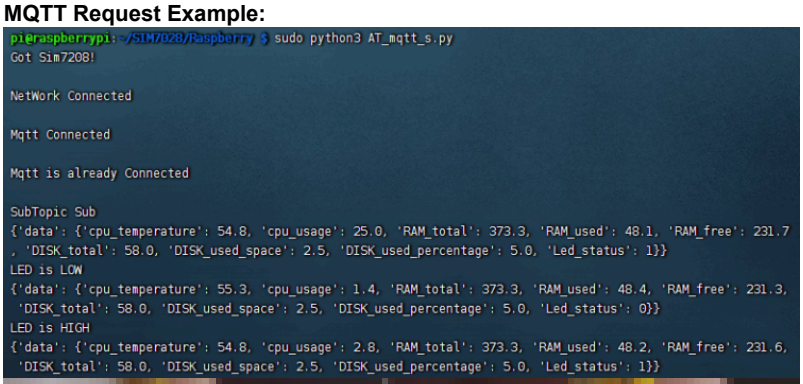
- Sample demo test:

HTTP Request Example:

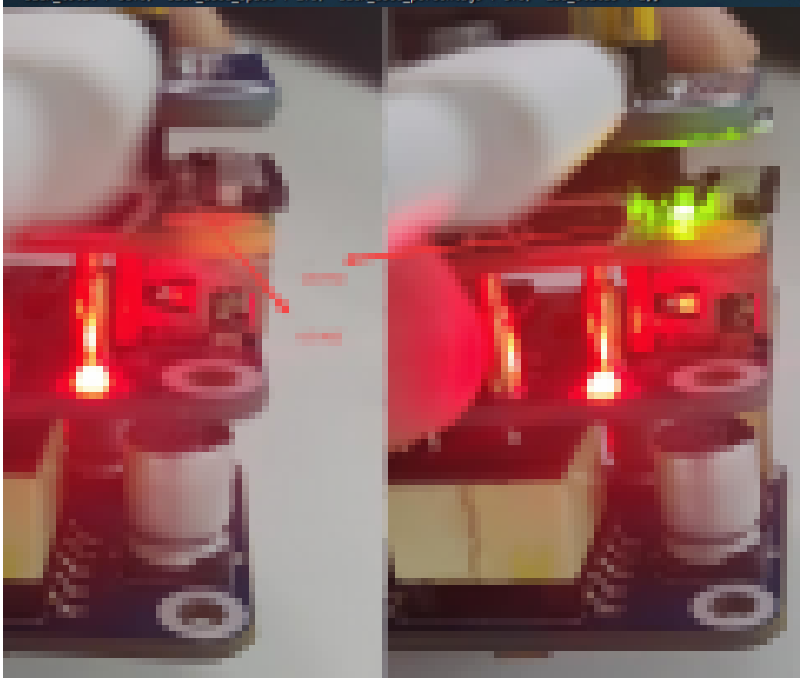
```
pi@raspberrypi:~/SIM7028/Raspberry $ sudo python3 AT_http_s.py
Sending AT command to verify port connection
Send AT command
OK
Serial port connection successful
ERROR
HTTP initialization already done!
Connecting to the interface!
OK
Interface connected to: https://www.waveshare.cloud/api/sample-test/
Interface connection successful!
Enter data? yes/no
yes
Please enter data:         ---------(Press Enter to finish input!)
this is SIM7028 NB-IoT HAT!
DOWNLOAD
OK
Sending request!
Start listening for response!
.........
Request Type: POST
Status Code: 200
Response Length: 29
Status: Request successful
Getting specific response data!
{"message": "data is saved!"}
Request finished
Request finished
Continue entering data? yes/no
no
Sending request!
Start listening for response!
........
Request Type: GET
Status Code: 200
Response Length: 76
Status: Request successful
Getting specific response data!
{"value": "this is SIM7028 NB-IoT HAT!", "timestamp": "2023-10-12 09:25:26"}
Request finished
Closing HTTP
Bye
```

(/wiki/File:SIM7028_Raspberry_Images-1.png)

MQTT Request Example:



(/wiki/File:SIM7028_Raspberry_Images2.png)


(/wiki/File:SIM7028_Raspberry_Images3.png)

Working with Jetson Nano

Hardware Connection

Please refer to the diagram of SIM7028NB-IoT HAT connecting to the Raspberry Pi.

Running Effect

```
sudo apt-get install unzip -y
wget https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/SIM7028-NB-IoT-HAT-Demo-Code.zip
unzip SIM7028-NB-IoT-HAT-Demo-Code.zip -d SIM7028
cd SIM7028/JetsonNano/
sudo pip3 install paho-mqtt pyserial
# HTTP request
sudo python3 AT_http_s.py
#  MQTT request: pay attention to modifying the corresponding parameters, this MQTT example is to access the WaveshareCloud platform, see the ap
plication case for details.
sudo python3 AT_mqtt_s.py
```

HTTP Request Example

```
pi@raspberrypi:~/SIM7028/Raspberry $ sudo python3 AT_http_s.py
Sending AT command to verify port connection
Send AT command
OK
Serial port connection successful
ERROR
HTTP initialization already done!
Connecting to the interface!
OK
Interface connected to: https://www.waveshare.cloud/api/sample-test/
Interface connection successful!
Enter data? yes/no
yes
Please enter data:         ---------(Press Enter to finish input!)
this is SIM7028 NB-IoT HAT!
DOWNLOAD
OK
Sending request!
Start listening for response!
.........
Request Type: POST
Status Code: 200
Response Length: 29
Status: Request successful
Getting specific response data!
{"message": "data is saved!"}
Request finished
Request finished
Continue entering data? yes/no
no
Sending request!
Start listening for response!
........
Request Type: GET
Status Code: 200
Response Length: 76
Status: Request successful
Getting specific response data!
{"value": "this is SIM7028 NB-IoT HAT!", "timestamp": "2023-10-12 09:25:26"}
Request finished
Closing HTTP
Bye
```

(/wiki/File:SIM7028_Raspberry_Images-1.png)

MQTT Request Example

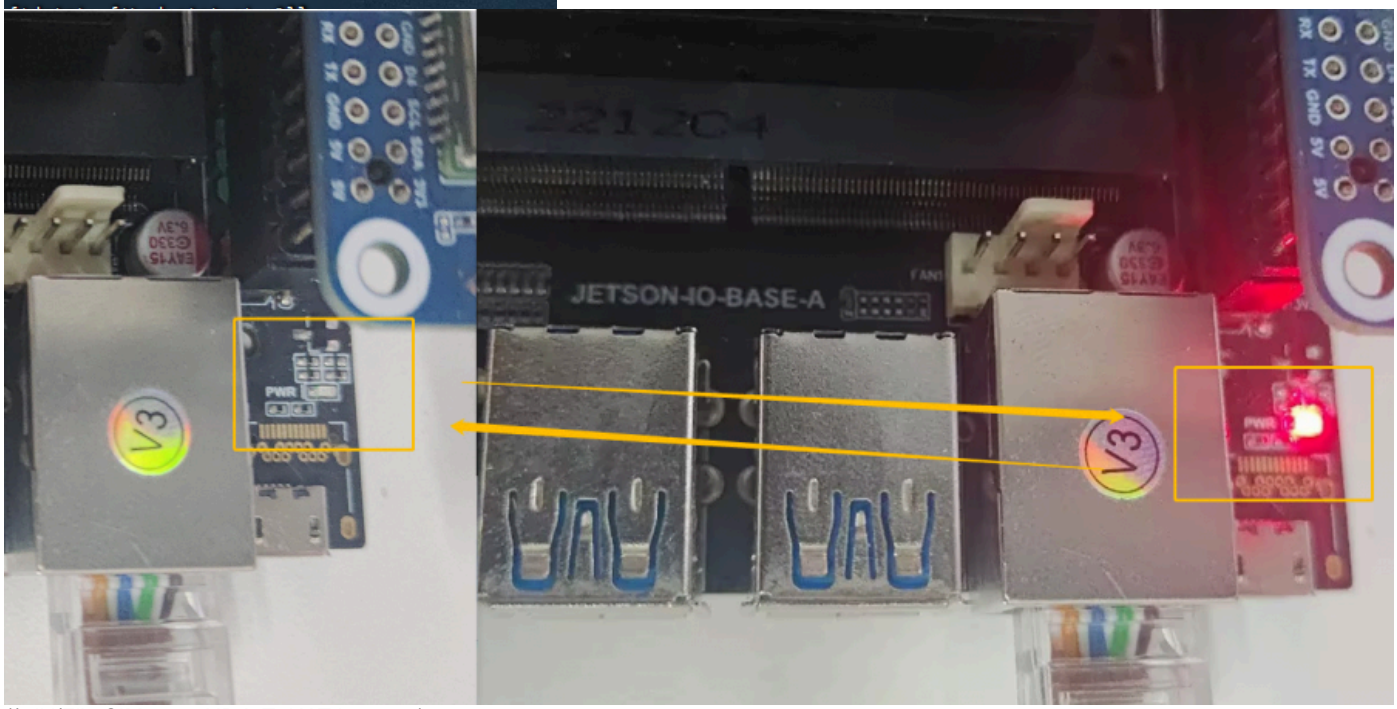
```
jetson@tegra-ubuntu:~$ sudo python3 AT_mqtt_s.py
Got Sim7208!

NetWork Connected

Mqtt is already Connected

SubTopic Sub
{'data': {'Led_status': 1}}
{'data': {'Led_status': 1}}
{'data': {'Led_status': 1}}
{'data': {'Led_status': 1}}
{'data': {'Led_status': 1}}
{'data': {'Led_status': 1}}
LED is LOW
{'data': {'Led_status': 0}}
{'data': {'Led_status': 0}}
{'data': {'Led_status': 0}}
{'data': {'Led_status': 0}}
{'data': {'Led_status': 0}}
{'data': {'Led_status': 0}}
LED is HIGH
{'data': {'Led_status': 1}}
{'data': {'Led_status': 1}}
{'data': {'Led_status': 1}}
```

(/wiki/File:SIM7028_NB-IoT_HAT-http04.png)



(/wiki/File:SIM7028_NB-IoT_HAT-http.png)

Connect to Arduino/ESP32

Hardware Connection

Arduino can be connected with a software serial port, and ESP32 can be connected with a hardware serial port, Serial 2.

Esp32	Sim7028 NB-IoT HAT	LED
5V	5V (Raspberry Pi 40PIN GPIO No. 2)	
GND	GND	-
Pin16	TX	
Pin17	RX	
Pin4	RST	

Pin5		+

Software Configuration

Open Arduino -> File -> Preferences -> Additional Board Manager URLs, add: https://github.com/espressif/arduino-esp32 (https://github.com/espressif/arduino-esp32)
Then open the development board management and search for esp32 to download and install. You can also install arduino-esp32 through the installer, you can find the relevant installation instructions in the Arduino community: Link (https://arduino.me/a/esp32).
Go to Tools -> Board -> Board Manager to add the esp32 library.
Go to Tools -> Manage Libraries to add "pubsubclient" library and "ArduinoJson" library.

Access Waveshare Cloud

SIM7028 MQTT Request Introduction

MQTT (Message Queuing Telemetry Transport) is a lightweight, publish/subscribe model-based messaging protocol designed to provide reliable and efficient communication for IoT devices.

MQTT Request Components

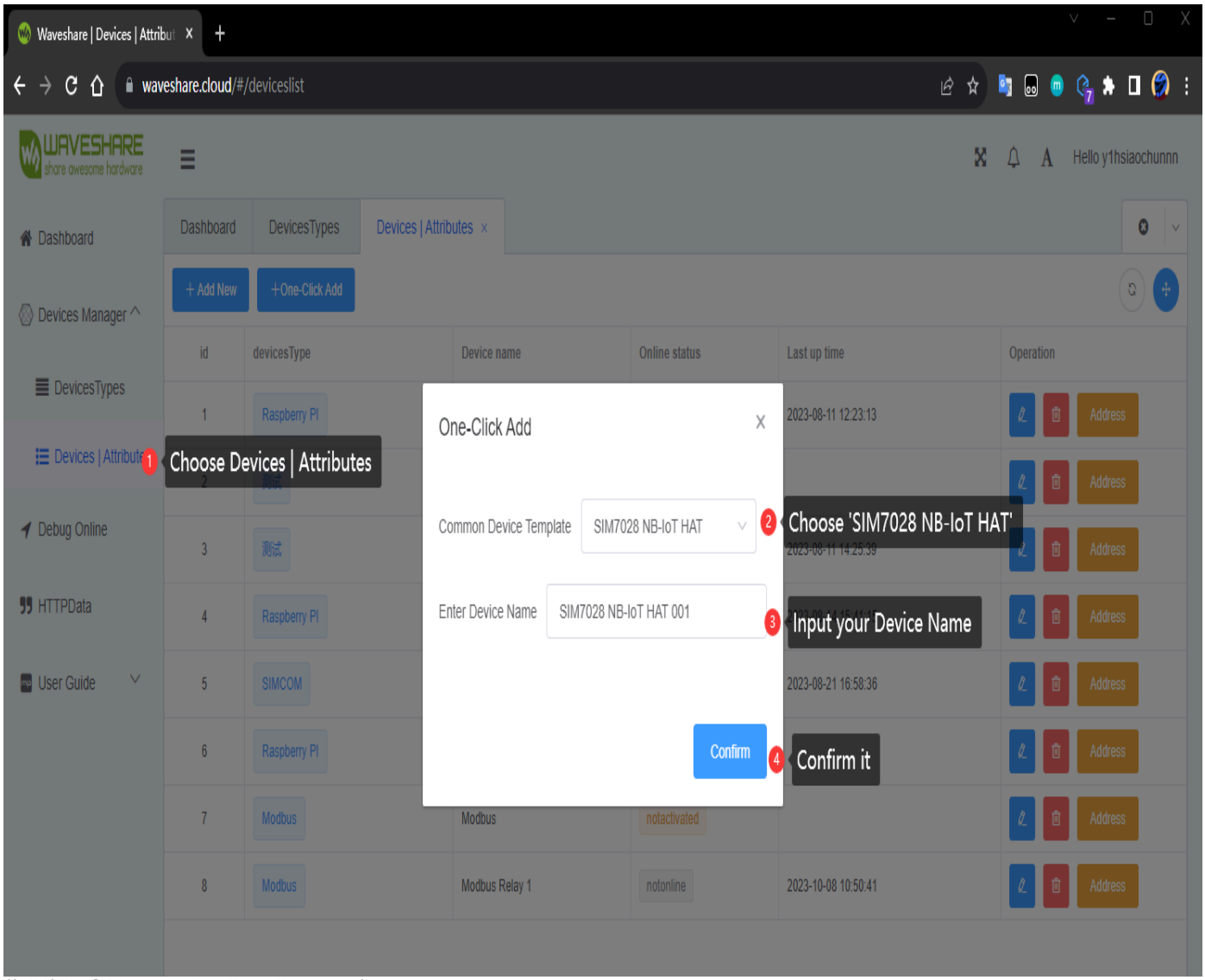
- Fixed Header: The fixed header of an MQTT message is a mandatory component in every message. It contains essential control information for message transmission, such as message type, Quality of Service (QoS) level, whether the topic should be retained, and more. The fixed header has a fixed length of one byte.
- Variable Header: The variable header of an MQTT message is an optional part, and its length depends on the message type and specific operations. It contains control information related to the message type, such as connection flags and the QoS level of subscribed topics.
- Topic Name: The topic name is a crucial part that identifies the message. When publishing a message, you need to specify a topic name, and subscribers use this topic name to receive messages of interest. Topic names are typically encoded using UTF-8.
- Payload: The payload is the actual data content being transmitted. For published messages, the payload is the data you want to send, and for subscribed messages, the payload is the data received from the publisher. The length of the payload can range from zero bytes up to the maximum message size limit.

MQTT Request Process

- Establish connection:
  - The client establishes a connection with the MQTT broker using the TCP/IP protocol. MQTT defaults to using port 1883 for unencrypted connections or port 8883 for encrypted connections (via TLS/SSL).
  - Clients have the option to maintain a session, allowing them to recover previous subscriptions and publishing states after disconnecting.
- Publishing Messages:
  - As a publisher, the client sends a PUBLISH message to publish a message on a specific topic.
  - The PUBLISH message consists of a fixed header, variable header, topic name and payload (message content).
  - After publishing a message, the MQTT broker delivers the message to subscribed clients based on their subscription information.
- Subscribing to Topics:
  - As a subscriber, the client sends a SUBSCRIBE message to subscribe to specific topics.
  - The SUBSCRIBE message includes a fixed header, variable header, one or more topic filters and their corresponding Quality of Service (QoS) levels.
  - Subscribers can simultaneously subscribe to multiple topics.
- Broker Confirmation of Subscriptions:
  - When the broker receives a subscription request, it will acknowledge the subscription and store the corresponding subscription information.
  - Subscribers can now receive messages published on the topics they subscribed to.
- Receiving Messages:
  - When a new message is published to a topic subscribed by a client, the MQTT broker will send the message to the subscriber.
  - Upon receiving the message, the subscriber can process the payload and take appropriate actions.
- Disconnecting:
  - After completing communication, clients (publishers or subscribers) can choose to disconnect from the MQTT broker.
  - When disconnecting, clients have the option to retain the session, allowing them to preserve previous subscription and publishing states when reconnecting.

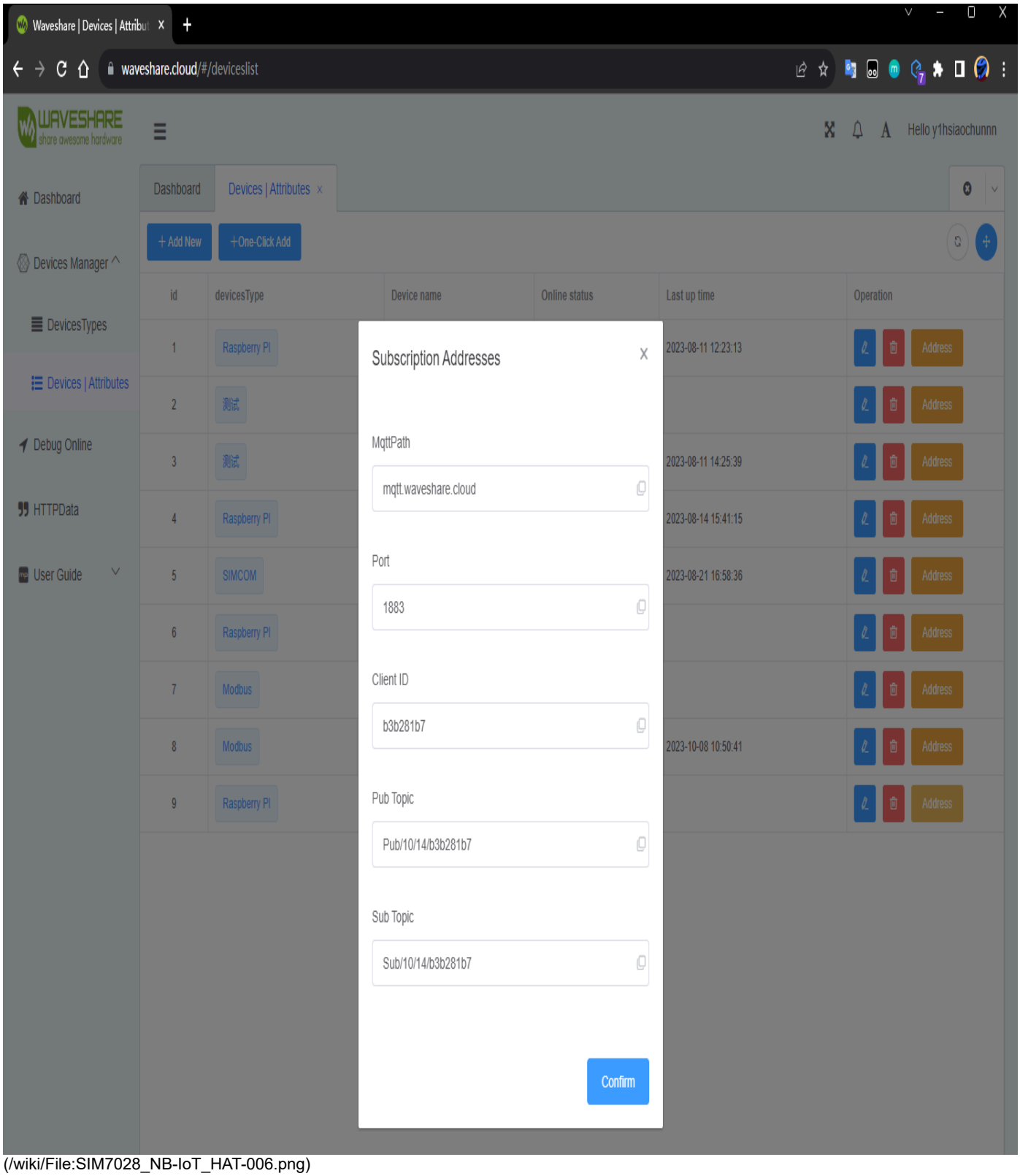
Log in to the platform, create a new device

- To create a specific device, click Add to Device List and fill in the relevant data.



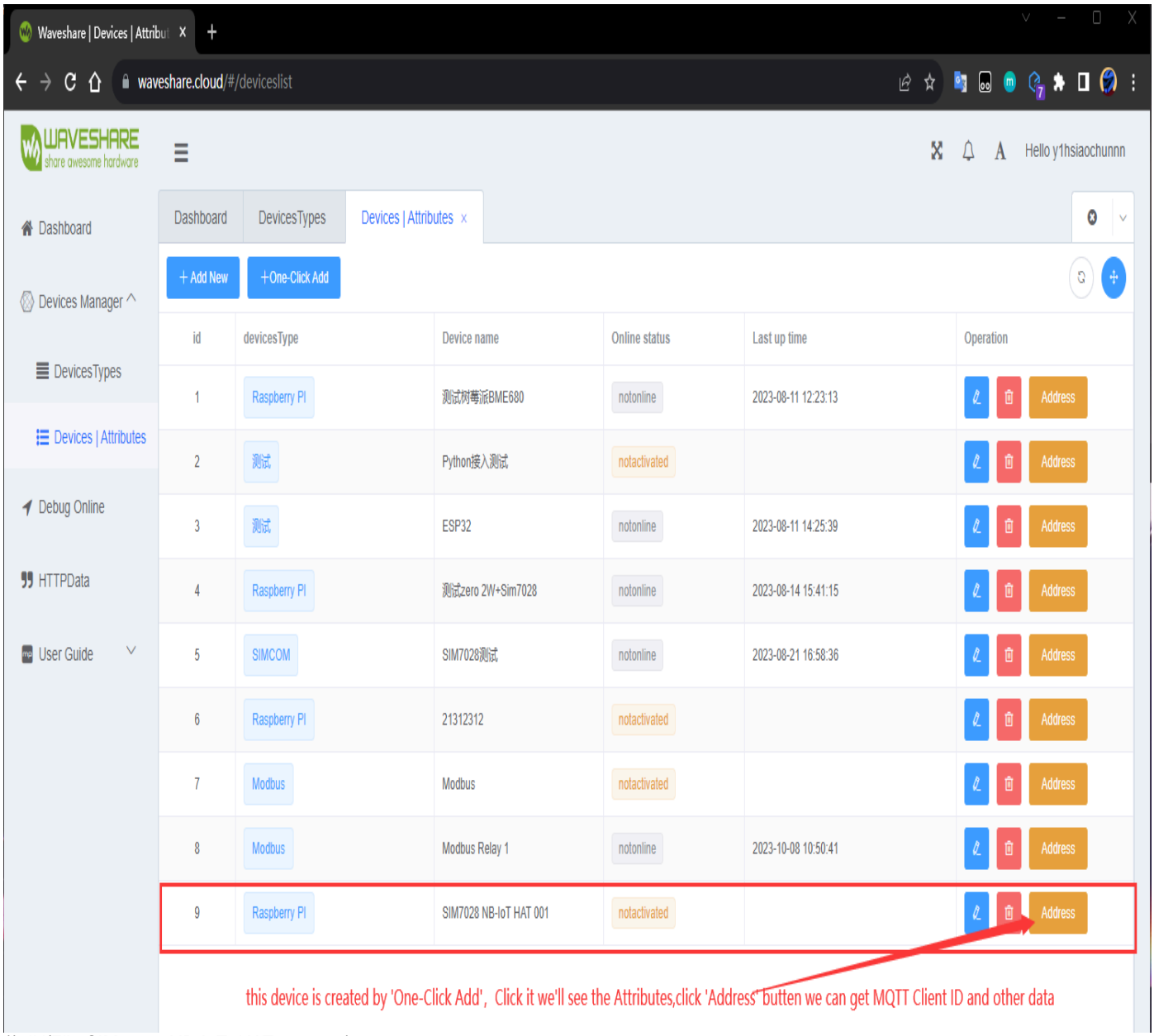
(/wiki/File:SIM7028_NB-IoT_HAT-005.png)

- After the creation, the device is inactive, you can view the contents of the MQTT connection properties through the right address details.

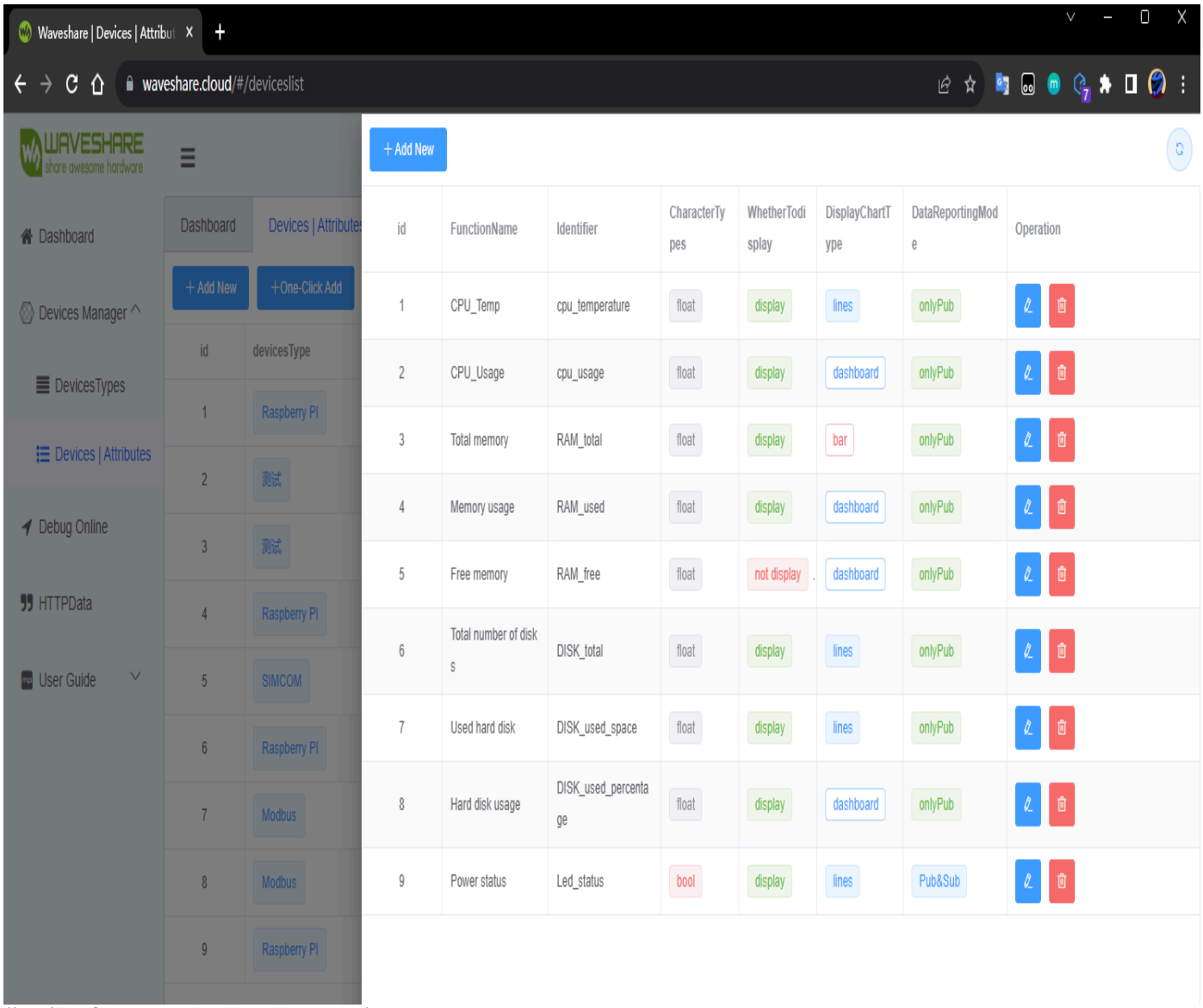


(/wiki/File:SIM7028_NB-IoT_HAT-006.png)

- The device attributes can be uploaded through the need to upload the data content to define the upload, here through a key to add the device that already has LED attributes.



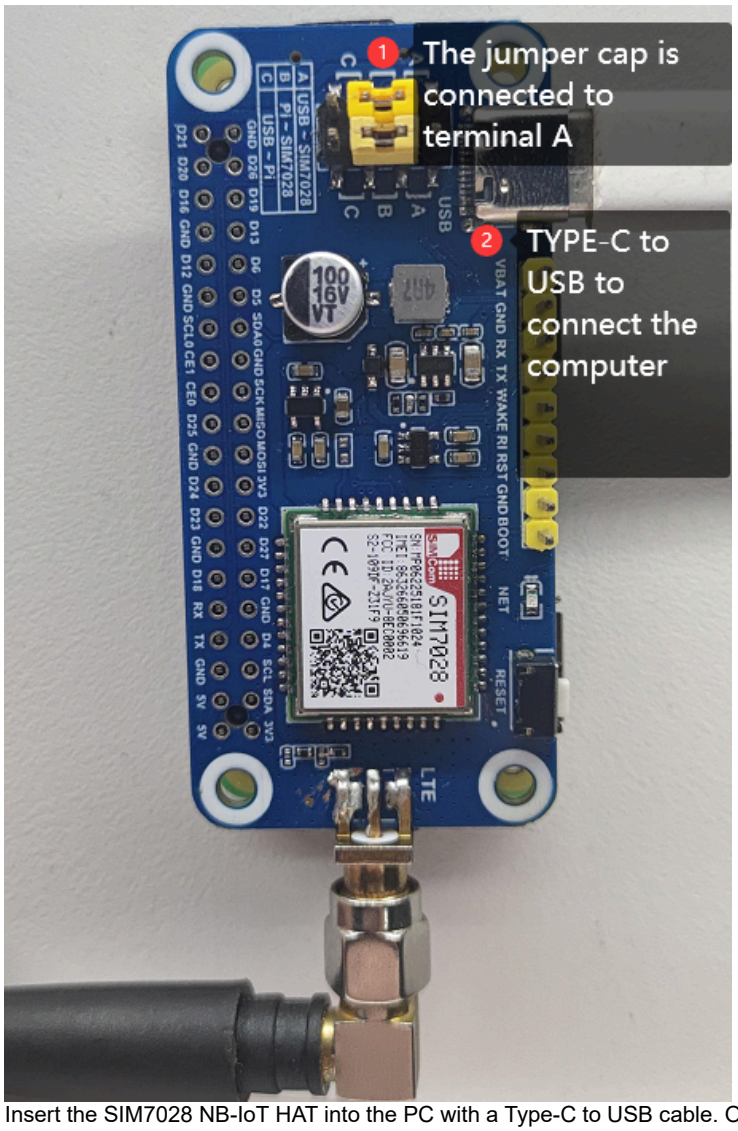
(/wiki/File:SIM7028_NB-IoT_HAT-007.png)



(/wiki/File:SIM7028_NB-IoT_HAT0071.png)

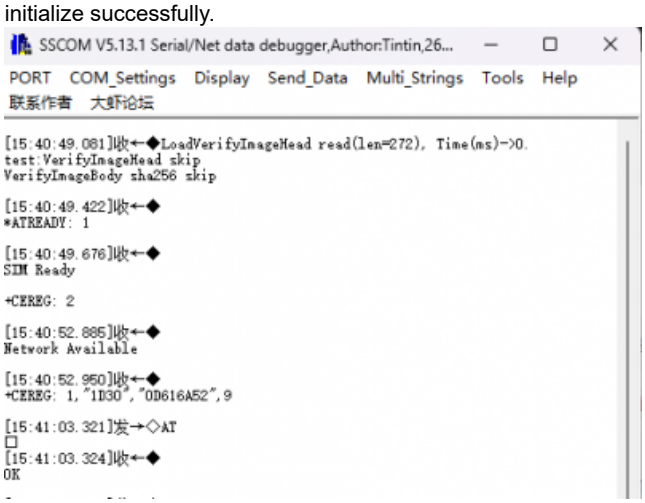
Use MQTT to Connect Waveshare Cloud

First, use a jumper cap to connect A, that is, the Type-C connects to Sim7028.



(/wiki/File:SIM7028_NB-IoT_HAT-mqtt.png)

Insert the SIM7028 NB-IoT HAT into the PC with a Type-C to USB cable. Open the serial port using sscom and wait for the network to initialize successfully.



(/wiki/File:SIM7028_NB-IoT_HAT-mqtt02.png)

Initialize MQTT Connection

```
AT+CMQTTSTART
```

```
AT+CMQTTACCQ=0,"{Type the client id assigned by the platform}",0
```

```
AT+CMQTTCONNECT=0,tcp://mqtt.waveshare.cloud:1883,20,1
```

Subscribe & Post Topic

```
AT+CMQTTTOPIC=0,18 #Type Pub Subject and 18 is the character length
```

```
AT+CMQTTSUB=0,18,0 #Type Sub Subject and 18 is the character length
```

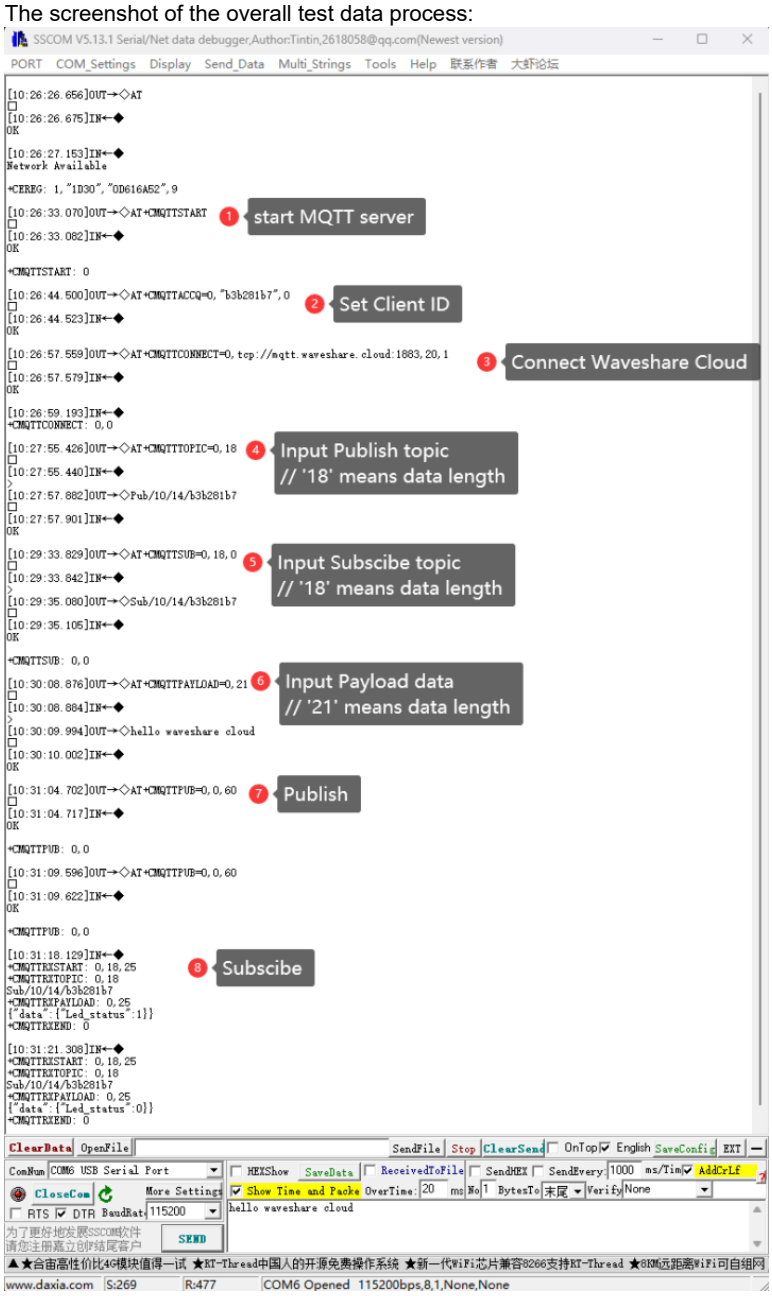
Fill in the content of the data to be uploaded:

```
AT+CMQTTPAYLOAD=0,21 #Character length is less than 10240, here type hello waveshare cloud 21-bit test
```

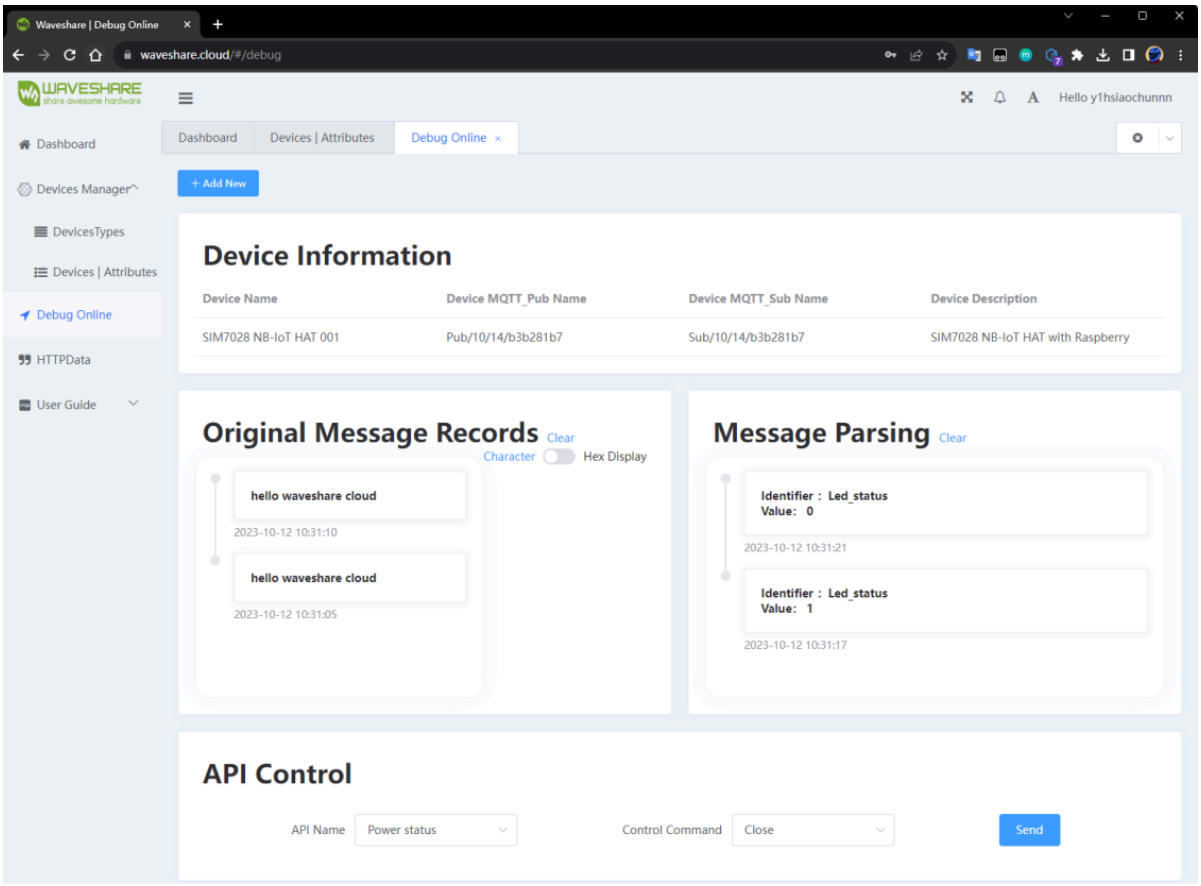
Publish:

```
AT+CMQTTPUB=0,0,60
```

The screenshot of the overall test data process:



(/wiki/File:WaveshareCloud_Images7.png)



(/wiki/File:WaveshareCloud_Images8.png)

Others

Low-power Mode Details

NB-IoT technology supports three power-saving modes: PSM (Power Saving Mode), DRX (Discontinuous Reception Mode) and eDRX (Extended DRX). In NB-IoT, PSM (Power Saving Mode) and eDRX (Extended Discontinuous Reception) are used to save power. In PSM mode, the terminal doesn't need to actively listen for paging messages to check for downstream data. On the other hand, eDRX mode has longer paging cycles compared to DRX mode, which can result in longer latency and potentially affect the real-time nature of data transmission. The choice between PSM and eDRX depends on the capabilities and configurations of both the terminal device and the network.

Regarding capabilities, it's important not to configure network features that the terminal does not support. Additionally, the terminal's supported capabilities may vary depending on the specific network conditions.

Low-power mode differences

No	Mode	Description
1	PSM	Extremely save power, and do not receive any data.
2	DRX	Regularly turn off network services to find the device at any time.
3	eDRX	eDRX can further reduce the power consumption of NB-IoT devices and extend battery life compared to the traditional DRX mode. However, at the same time, the device may delay when receiving data due to the longer cycle and shorter reception duration of eDRX.

PSM

In PSM (Power Saving Mode), the terminal does not actively check for paging data in the downlink. PSM mode remains active until either a Tracking Area Update (TAU) or uplink data transmission is required. T3412 indicates the tracking area update time, and T3324 indicates the timer that enters the PSM in IDLE mode.

Test Description

AT command	Description	Return value
AT+QCPMUCFG=1,4	Deep sleep mode	OK

AT+QCPSMR=1	Enable low-power URC report	OK
AT+CPSMS=1,,,"01011111","00000001"	Set the PSM mode of the timer	OK
AT+CPSMS=0	Exit PSM mode	OK

DRX & eDRX

DRX (Discontinuous Reception Mode) can be considered a mode where downstream data can reach the terminal device at any time. Within each DRX cycle (typically set to 1.28 seconds, 2.56 seconds, 5.12 seconds, 10.24 seconds, etc.), the terminal checks for the arrival of downstream data. This mode is suitable for applications with relatively high latency requirements. Terminal devices in this mode are typically powered, such as those used in services like street lighting. Compared to DRX, eDRX (Extended DRX) has longer paging cycles, making the terminal more power-efficient. However, it also comes with longer downstream data latency (e.g., DRX values are 1.28 seconds, and 2.56 seconds, while eDRX values can go up to 20.48 seconds or even 2.9 hours). This makes it ideal for application scenarios where time constraints are not very high.

Test Description

AT Command	Description	Return Value
AT+CEDRXS=1,5,"0010"	Enable eDRX mode	OK
AT+CEDRXS?	Query eDRX status	+CEDRXS: 5,"0010" OK
AT+CEDRXRDP	Query whether to support	+CEDRXRDP: 5,"0000","0010","0100"
AT+CEDRXS=0	Exit eDRX mode	OK

Resource

Schematic

- Schematic (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/SIM7028-NB-IoT-HAT-Schematic.pdf)

Demo

- Demo (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/SIM7028-NB-IoT-HAT-Demo-Code.zip)

Software

- SSCOM (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/SIM7020-SSCOM.7z)
- MQTT Test (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/MQTT-TEST.7z)
- Tool (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/firmware/FlashTools_V3.0.1.zip)
- Firmware (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/2110B08V03SIM7028.rar)

Documents

- SIM7022 Series AT Command Manual V1.05 (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/SIM7028%20NB-IoT%20HAT-Doc/SIM7022_Series_AT_Command_Manual_V1.05.pdf)
- SIM7028 Series COAP Application Note V1.00 (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/SIM7028%20NB-IoT%20HAT-Doc/SIM7028_Series_COAP_Application_Note_V1.00.pdf)
- SIM7028 Series FOTA Application Note V1.00 (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/SIM7028%20NB-IoT%20HAT-Doc/SIM7028_Series_FOTA_Application_Note_V1.00.pdf)
- SIM7028 Series HTTP(S) Application Note V1.04 (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/SIM7028%20NB-IoT%20HAT-Doc/SIM7028_Series_HTTP(S)_Application_Note_V1.04.pdf)
- SIM7028 Series LWM2M Application Note V1.03 (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/SIM7028%20NB-IoT%20HAT-Doc/SIM7028_Series_LWM2M_Application_Note_V1.03.pdf)
- SIM7028 Series Low Power Mode Application Note V1.01 (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/SIM7028%20NB-IoT%20HAT-Doc/SIM7028_Series_Low_Power_Mode_Application_Note_V1.01.pdf)
- SIM7028 Series MQTT(S) Application_Note V1.03 (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/SIM7028%20NB-IoT%20HAT-Doc/SIM7028_Series_MQTT(S)_Application_Note_V1.03.pdf)
- SIM7028 Series SSL Application Note V1.00 (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/SIM7028%20NB-IoT%20HAT-Doc/SIM7028_Series_SSL_Application_Note_V1.00.pdf)

- SIM7028 Series TCPIP Application Note V1.04 (https://files.waveshare.com/wiki/SIM7028-NB-IoT-HAT/SIM7028%20NB-IoT%20HAT-Doc/SIM7028_Series_TCPIP_Application_Note_V1.04.pdf)

Related Tutorials

- Raspberry Pi Documentation (https://www.waveshare.com/wiki/Raspberry_Pi_Documentation)

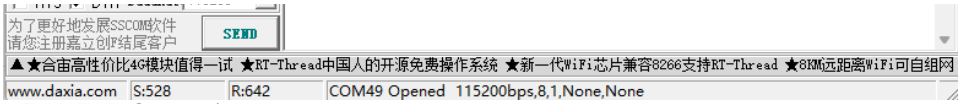
FAQ

Question:How does SIM7028 connect to AWS IoT?

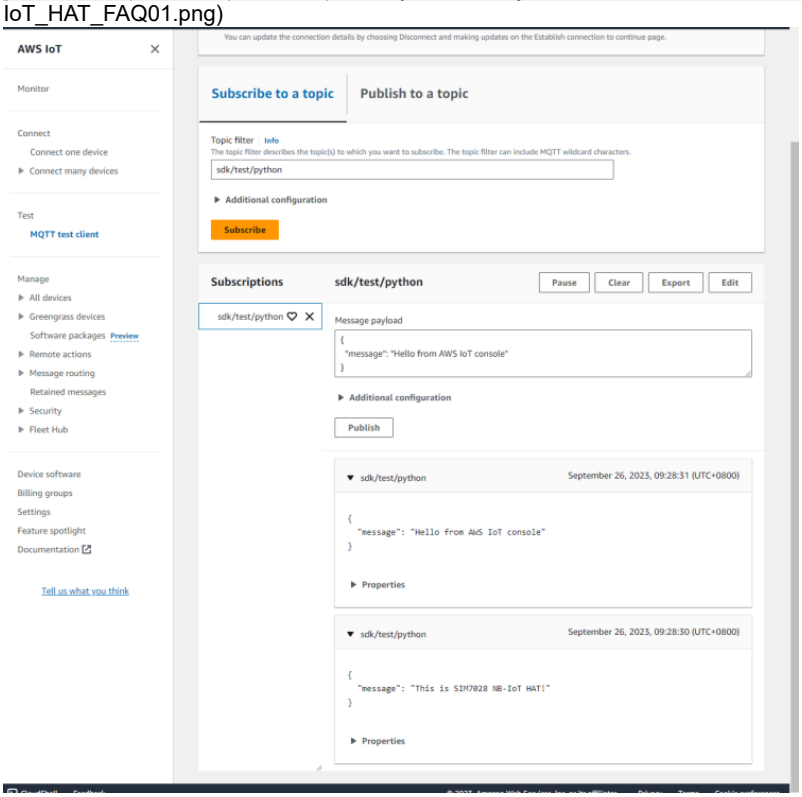
Answer:

1. First check the firmware version of SIM7028, the firmware version is required to be at least **2110B08SIM7028**.
2. AWS IoT Core requires certificate-based two-way authentication. Of course, you can achieve self-authenticated access using AWS IoT Lambda functions. You can refer to the following steps for certificate-based access:

(/wiki/File:SIM7028_NB-



IoT_HAT_FAQ01.png)



(/wiki/File:SIM7028_NB-IoT_HAT_FAQ02.png)

Question:How does SIM7028 access AWS IoT without using a certificate?

Answer:

AWS IoT Core typically involves two-way authentication. AWS IoT provides a feature called Lambda functions to handle certificate-based authentication and access, similar to the concept of "one device, one secret" in the Alibaba Cloud IoT platform. However, Alibaba Cloud IoT uses cryptographic methods to address authentication, while AWS IoT Core primarily uses certificates for authentication, and the custom code through the function of lambda function returns the message sending logic judgment to the corresponding certificate of AWS IoT Core to confirm the connection.
Here's how it works: <username\password> Authentification -> <Function Evaluation> -> <AWS IoT Core Certificate Authentication Connection> -> <Confirmation of Connection Authorization Granted>.

Question:Why do some of the codes not work after updating the firmware on SIM7028?

Answer:

Firmware version **2110B08SIM7028** and above has modified the underlying logic, including the fact that after MQTT sends a message, it will clear the contents cached in the message body, so at this point, we should add another cache command to input the message body before requesting it.

Question:How can I upgrade the firmware?

Answer:

Please refer to the following steps to upgrade the firmware：
Guide manual:
https://files.waveshare.com/wiki/SIM7070G/firmware/sim7070_sim7080_qdl%20only%20for%20update.pdf
(https://files.waveshare.com/wiki/SIM7070G/firmware/sim7070_sim7080_qdl%20only%20for%20update.pdf)

Tool:
https://files.waveshare.com/wiki/SIM7080G/firmware/SIM7070_SIM7080%20QDL%20V1.55%20(BackupQMBN)Only%20for%20Update.zip (https://files.waveshare.com/wiki/SIM7080G/firmware/SIM7070_SIM7080%20QDL%20V1.55%20(BackupQMBN)Only%20for%20Update.zip)
Firmware:
https://files.waveshare.com/wiki/SIM7070G/firmware/1951B07SIM7070.zip (https://files.waveshare.com/wiki/SIM7070G/firmware/1951B07SIM7070.zip)
Tips:
1. Pay attention to the device manager. During the upgrade process, new device will be prompted to insert.
2. Pay attention to the USB cable. During the upgrade process, the USB cable has a high rate, so you need to choose a USB cable of better quality to avoid bad contact.
4. Uninstall and reinstall the upgrade tool.
5. More details please refer to the operation of the video:
http://www.waveshare.net/wiki/SIM7600-Firmware-upgrade-Video (http://www.waveshare.net/wiki/SIM7600-Firmware-upgrade-Video)
6. When the programming bar is scrolled, the whole process cannot be interrupted, otherwise the module system will be stuck.

Support

Technical Support

If you need technical support or have any feedback/review, please click the **Submit Now** button to submit a ticket, Our support team will check and reply to you within 1 to 2 working days. Please be patient as we make every effort to help you to resolve the issue.
Working Time: 9 AM - 6 PM GMT+8 (Monday to Friday)

Submit Now (https://service.waveshare.com/)