
NBSN95_NBSN95A NB-IoT Sensor Node User Manual

last modified by Xiaoling

on 2023/01/07 08:56

Table of Contents

1. Introduction	6
1.1 What is NBSN95 NB-IoT Sensor Node	6
1.2 Features	6
1.3 Specification	6
1.4 Applications	7
1.5 Pin Definitions & Switch	7
1.5.1 Jumper JP2	8
1.5.2 BOOT MODE / SW1	9
1.5.3 Reset Button	9
1.5.4 LED	9
1.6 Hole Option	9
2. Use NBSN95 to communicate with IoT Server	11
2.1 How it works	11
2.2 Configure the NBSN95	12
2.2.1 Power On NBSN95	12
2.2.2 Test Requirement	13
2.2.3 Insert SIM card	13
2.2.4 Connect USB - TTL to NBSN95 to configure it	14
2.2.5 Use CoAP protocol to uplink data	15
2.2.6 Use UDP protocol to uplink data(Default protocol)	17
2.2.7 Use MQTT protocol to uplink data	19
2.2.8 Use TCP protocol to uplink data	21
2.2.9 Change Update Interval	22
2.3 Working Mode & Uplink Payload	22
2.3.1 CFGMOD=1 (Default Mode)	22
2.3.2 CFGMOD=2 (Distance Mode)	23
2.3.3 CFGMOD=3 (3 ADC + I2C)	25
2.3.4 CFGMOD=4 (3 x DS18B20)	27
2.3.5 CFGMOD=5 (Weight Measurement by HX711)	29
2.3.6 CFGMOD=6 (Counting mode)	30
2.4 Payload Explanation and Sensor Interface	32
2.4.1 Device ID	32
2.4.2 Version Info	32
2.4.3 Battery Info	32
2.4.4 Signal Strength	32
2.4.5 Temperature (DS18B20)	32
2.4.6 Digital Input	33
2.4.7 Analogue Digital Converter (ADC)	33
2.4.8 Digital Interrupt	35
2.4.9 I2C Interface (SHT20)	37
2.4.10 Distance Reading	39
2.4.11 Ultrasonic Sensor	39
2.4.12 +5V Output	40
2.4.13 Weigh Sensor HX711	41
2.4.14 Timestamp	41
2.5 Downlink Payload	41
2.6 Firmware Change Log	42
2.7 Use VDD or +5V to Power External Sensor	42
2.8 Battery Analysis	42
2.8.1 Battery Type	42
2.8.2 Power consumption Analyze	43
2.8.3 Battery Note	43
2.8.4 Replace the battery	43

3. Access NB-IoT Module	43
4. Using the AT Commands	44
4.1 Access AT Commands	44
5. Developer Guide	45
5.1 Get and compile Software	45
5.2 Get hardware source	45
6. FAQ	46
6.1 How to Upgrade Firmware	46
7. Trouble Shooting	46
7.1 Connection problem when uploading firmware	46
7.2 AT Command input doesn't work	46
8. Order Info	46
9. Packing Info	46
10. Support	47



Table of Contents:

- [1. Introduction](#)
 - [1.1 What is NBSN95 NB-IoT Sensor Node](#)
 - [1.2 Features](#)
 - [1.3 Specification](#)
 - [1.4 Applications](#)
 - [1.5 Pin Definitions & Switch](#)
 - [1.5.1 Jumper JP2](#)
 - [1.5.2 BOOT MODE / SW1](#)
 - [1.5.3 Reset Button](#)
 - [1.5.4 LED](#)
 - [1.6 Hole Option](#)

- [2. Use NBSN95 to communicate with IoT Server](#)
 - [2.1 How it works](#)
 - [2.2 Configure the NBSN95](#)
 - [2.2.1 Power On NBSN95](#)
 - [2.2.2 Test Requirement](#)
 - [2.2.3 Insert SIM card](#)
 - [2.2.4 Connect USB – TTL to NBSN95 to configure it](#)
 - [2.2.5 Use CoAP protocol to uplink data](#)
 - [2.2.6 Use UDP protocol to uplink data\(Default protocol\)](#)
 - [2.2.7 Use MQTT protocol to uplink data](#)
 - [2.2.8 Use TCP protocol to uplink data](#)
 - [2.2.9 Change Update Interval](#)
 - [2.3 Working Mode & Uplink Payload](#)
 - [2.3.1 CFGMOD=1 \(Default Mode\)](#)
 - [2.3.2 CFGMOD=2 \(Distance Mode\)](#)
 - [Connection of LIDAR-Lite V3HP:](#)
 - [2.3.3 CFGMOD=3 \(3 ADC + I2C\)](#)
 - [2.3.4 CFGMOD=4 \(3 x DS18B20\)](#)
 - [2.3.5 CFGMOD=5 \(Weight Measurement by HX711\)](#)
 - [2.3.6 CFGMOD=6 \(Counting mode\)](#)
 - [2.4 Payload Explanation and Sensor Interface](#)
 - [2.4.1 Device ID](#)
 - [2.4.2 Version Info](#)
 - [2.4.3 Battery Info](#)
 - [2.4.4 Signal Strength](#)
 - [2.4.5 Temperature \(DS18B20\)](#)
 - [2.4.6 Digital Input](#)
 - [2.4.7 Analogue Digital Converter \(ADC\)](#)
 - [2.4.8 Digital Interrupt](#)
 - [2.4.9 I2C Interface \(SHT20\)](#)
 - [2.4.10 Distance Reading](#)
 - [2.4.11 Ultrasonic Sensor](#)
 - [2.4.12 +5V Output](#)
 - [2.4.13 Weigh Sensor HX711](#)
 - [2.4.14 Timestamp](#)
 - [2.5 Downlink Payload](#)
 - [2.6 Firmware Change Log](#)
 - [2.7 Use VDD or +5V to Power External Sensor](#)
 - [2.8 Battery Analysis](#)
 - [2.8.1 Battery Type](#)
 - [2.8.2 Power consumption Analyze](#)
 - [2.8.3 Battery Note](#)
 - [2.8.4 Replace the battery](#)
- [3. Access NB-IoT Module](#)
- [4. Using the AT Commands](#)
 - [4.1 Access AT Commands](#)
- [5. Developer Guide](#)
 - [5.1 Get and compile Software](#)
 - [5.2 Get hardware source](#)
- [6. FAQ](#)
 - [6.1 How to Upgrade Firmware](#)
- [7. Trouble Shooting](#)
 - [7.1 Connection problem when uploading firmware](#)
 - [7.2 AT Command input doesn't work](#)
- [8. Order Info](#)
- [9. Packing Info](#)
- [10. Support](#)

1. Introduction

1.1 What is NBSN95 NB-IoT Sensor Node

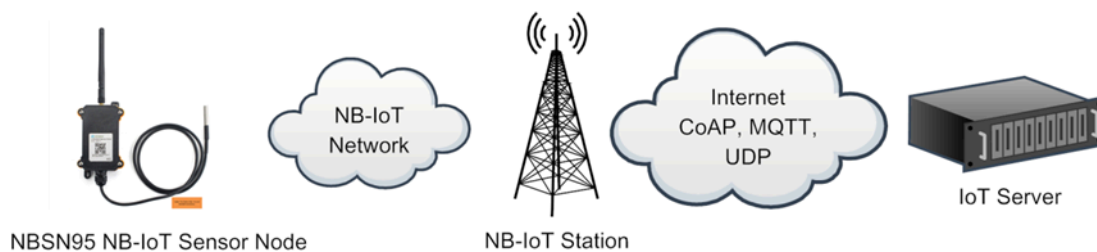
NBSN95 is a Long Range NB-IoT Sensor Node. It is designed for **outdoor data logging** and powered by **Li/SOCI2 battery** for long term use and secure data transmission. It is designed to facilitate developers to quickly deploy industrial level NB-IoT solutions. It helps users to turn the idea into a practical application and make the Internet of Things a reality. It is easy to program, create and connect your things everywhere.

NarrowBand-Internet of Things (NB-IoT) is a standards-based low power wide area (LPWA) technology developed to enable a wide range of new IoT devices and services. NB-IoT significantly improves the power consumption of user devices, system capacity and spectrum efficiency, especially in deep coverage.

NBSN95 uses STM32I0x chip from ST, STM32L0x is the **ultra-low-power** STM32L072xx microcontrollers incorporate the connectivity power of the universal serial bus (USB 2.0 crystal-less) with the high-performance ARM® Cortex®-M0+ 32-bit RISC core operating at a 32 MHz frequency, a memory protection unit (MPU), high-speed embedded memories (192 Kbytes of Flash program memory, 6 Kbytes of data EEPROM and 20 Kbytes of RAM) plus an extensive range of enhanced I/Os and peripherals.

NBSN95 is an **open source product**, it is based on the STM32Cube HAL drivers and lots of libraries can be found in ST site for rapid development.

NBSN95 in a NB-IoT Network



1.2 Features

- STM32L072xxxx MCU
- NB-IoT Bands: B1/B3/B8/B5/B20/B28 @H-FDD
- Pre-load bootloader on USART1/USART2
- MDK-ARM Version 5.24a IDE
- I2C, LPUSART1, USB, SPI2
- 3x12bit ADC, 1x12bit DAC
- 20xDigital I/O
- Open-source hardware / software
- IP66 Waterproof Enclosure
- Ultra-Low Power consumption
- AT Commands to change parameters
- Micro SIM card slot for NB-IoT SIM
- 8500mAh Battery for long term use

1.3 Specification

Micro Controller:

- MCU: STM32L072xxxx
- Flash: 128KB
- RAM: 20KB
- EEPROM: 6KB
- Clock Speed: 32Mhz

Common DC Characteristics:

- Supply Voltage: 2.1v ~ 3.6v
- Operating Temperature: -40 ~ 85°C
- I/O pins: Refer to STM32L072 datasheet

NB-IoT Spec:

- B1 @H-FDD: 2100MHz
- B3 @H-FDD: 1800MHz
- B8 @H-FDD: 900MHz
- B5 @H-FDD: 850MHz
- B20 @H-FDD: 800MHz
- B28 @H-FDD: 700MHz

Battery:

- Li/SOCl₂ un-chargeable battery
- Capacity: 8500mAh
- Self Discharge: <1% / Year @ 25°C
- Max continuously current: 130mA
- Max boost current: 2A, 1 second

Power Consumption

- STOP Mode: 10uA @ 3.3v
- Max transmit power: 350mA@3.3v

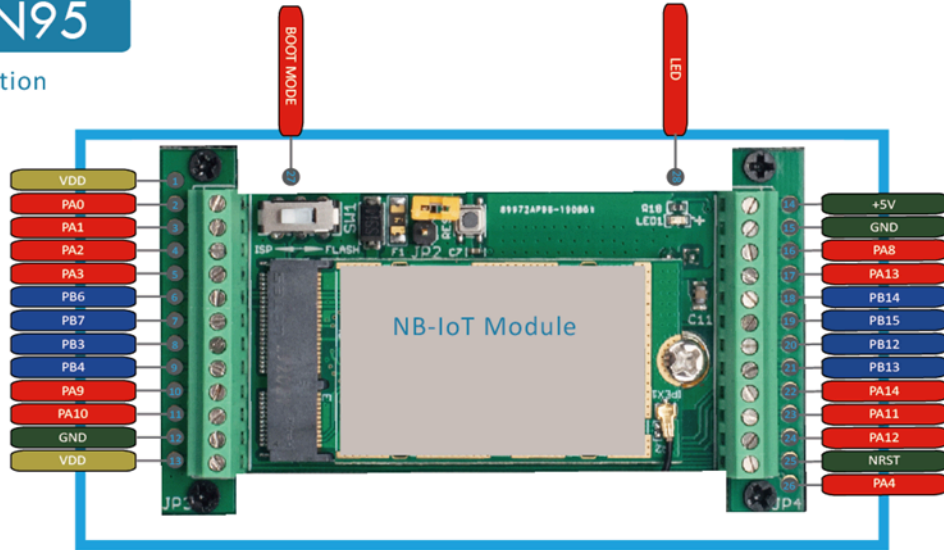
1.4 Applications

- Smart Buildings & Home Automation
- Logistics and Supply Chain Management
- Smart Metering
- Smart Agriculture
- Smart Cities
- Smart Factory

1.5 Pin Definitions & Switch

NBSN95

Pin Definition



No.	Signal	Direction	Function	Remark
1	VCC(2.9V)	OUTPUT	VCC	Directly connect to main power for board
2	PA0	In/Out	Directly from STM32 chip	Used as ADC in NBSN95 image
3	PA1	In/Out	Directly from STM32 chip	
4	PA2	In/Out	Directly from STM32 chip, 10k pull up to VCC	Used as UART_TXD in NBSN95 image
5	PA3	In/Out	Directly from STM32 chip, 10k pull up to VCC	Used as UART_RXD in NBSN95 image
6	PB6	In/Out	Directly from STM32 chip, 10k pull up to VCC	
7	PB7	In/Out	Directly from STM32 chip, 10k pull up to VCC	
8	PB3	In/Out	Directly from STM32 chip, 10k pull up to VCC	
9	PB4	In/Out	Directly from STM32 chip	
10	PA9	In/Out	Directly from STM32 chip, 10k pull up to VCC	
11	PA10	In/Out	Directly from STM32 chip, 10k pull up to VCC	
12	GND		Ground	
13	VCC(2.9V)	OUTPUT	VCC	Directly connect to main power for board
14	Jumper		Power on/off jumper	
15	PA4	In/Out	Directly from STM32 chip	
16	NRST	In	Reset MCU	
17	PA12	In/Out	Directly from STM32 chip	
18	PA11	In/Out	Directly from STM32 chip	
19	PA14	In/Out	Directly from STM32 chip	
20	PB13	In/Out	Directly from STM32 chip	
21	PB12	In/Out	Directly from STM32 chip	
22	PB15	In/Out	Directly from STM32 chip	
23	PB14	In/Out	Directly from STM32 chip	
24	PA13	In/Out	Directly from STM32 chip	
25	PA8	In/Out	Directly from STM32 chip	Default use to turn on/off LED1 in NBSN95 image
26	GND		Ground	
27	+5V	Out	5v output power	Controlled by PB5(Low to Enable, High to Disable)
28	LED1		Controlled by PA8	Blink on transmit
29	BOOT MODE		Configure device in working mode or ISP program mode	Flash: Normal Working mode and send AT Commands ISP: UART Program Mode
30	NRST	In	Reset MCU	

1.5.1 Jumper JP2

Power on Device when put this jumper.

1.5.2 BOOT MODE / SW1

1) ISP: upgrade mode, device won't have any signal in this mode. but ready for upgrade firmware. LED won't work. Firmware won't run.

2) Flash: work mode, device starts to work and send out console output for further debug

1.5.3 Reset Button

Press to reboot the device.

1.5.4 LED

It will flash:

1. When boot the device in flash mode
2. Send an uplink packet

1.6 Hole Option

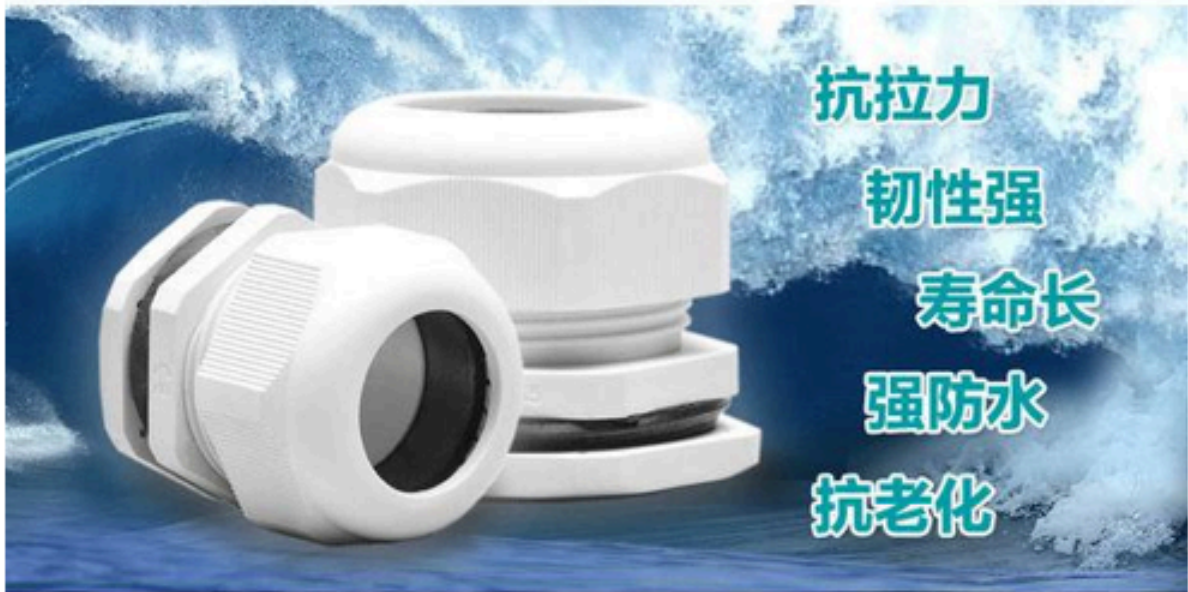
The NBSN95 provides different hole size options for different size sensor cable. The options provided are M12, M16. The definition is as below:



多项国际质量认证

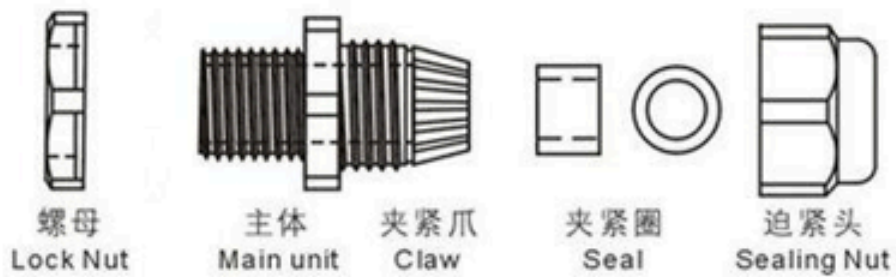


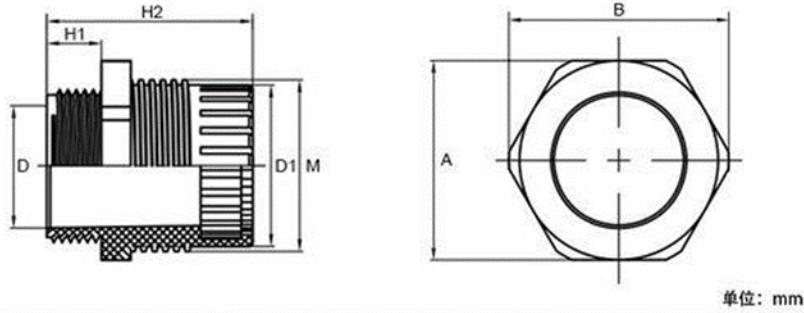
品质保证放心使用



产品结构

Structure





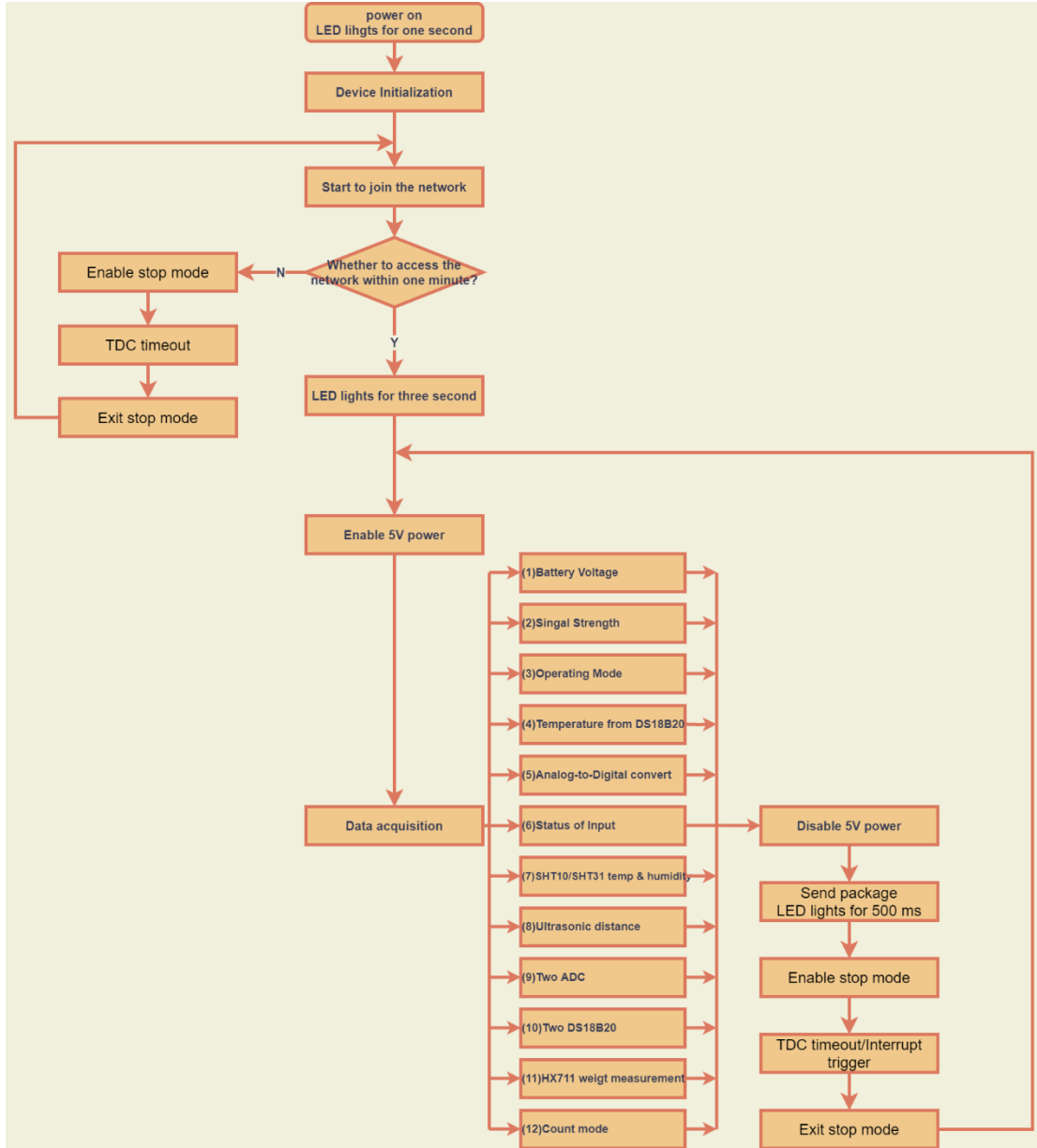
MODEL	H1	H2	M	M1	D	D1	A	B
M12*1.5	8	28.3	12.0	12.0	10.4	8.5 ± 0.2	16 ± 0.2	18 ± 0.2
M16*1.5	8	30.7	15.1	16.0	13.5	10.9 ± 0.2	18.8 ± 0.2	20.6 ± 0.2
M20*1.5	9	34.0	20.2	20.0	18.7	16.2 ± 0.2	22.8 ± 0.2	25.2 ± 0.2

2. Use NBSN95 to communicate with IoT Server

2.1 How it works

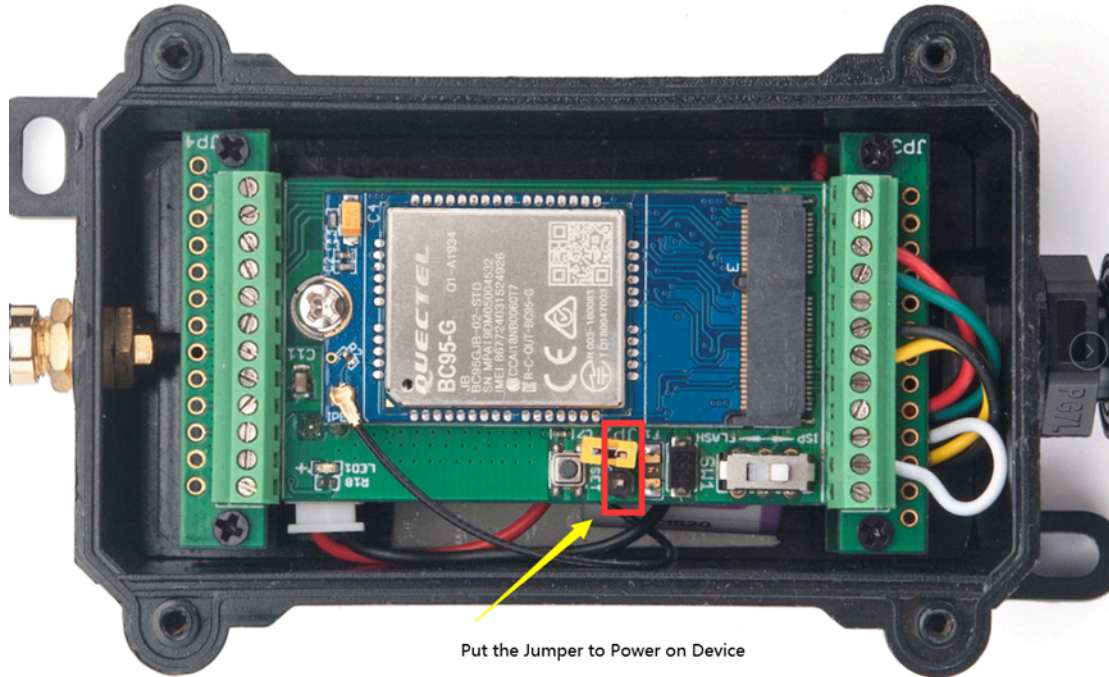
The NBSN95 is equipped with a NB-IoT module, the pre-loaded firmware in NBSN95 will get environment data from sensors and send the value to local NB-IoT network via the NB-IoT module. The NB-IoT network will forward this value to IoT server via the protocol defined by NBSN95.

The diagram below shows the working flow in default firmware of NBSN95:



2.2 Configure the NBSN95

2.2.1 Power On NBSN95



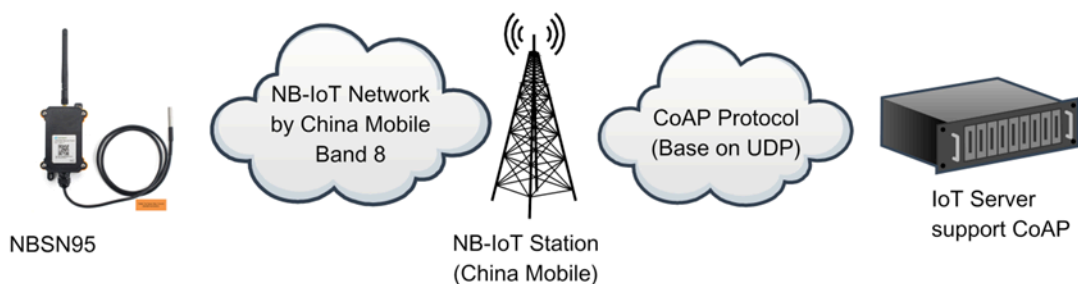
2.2.2 Test Requirement

To use NBSN95 in your city, make sure meet below requirements:

- Your local operator has already distributed a NB-IoT Network there.
- The local NB-IoT network used the band that NBSN95 supports.
- Your operator is able to distribute the data received in their NB-IoT network to your IoT server.

Below figure shows our testing structure. Here we have NB-IoT network coverage by China Mobile, the band they use is B8. The NBSN95 will use CoAP(120.24.4.116:5683) or raw UDP(120.24.4.116:5601) or MQTT(120.24.4.116:1883)or TCP(120.24.4.116:5600)protocol to send data to the test server

NBSN95 network example in Dragino office



2.2.3 Insert SIM card

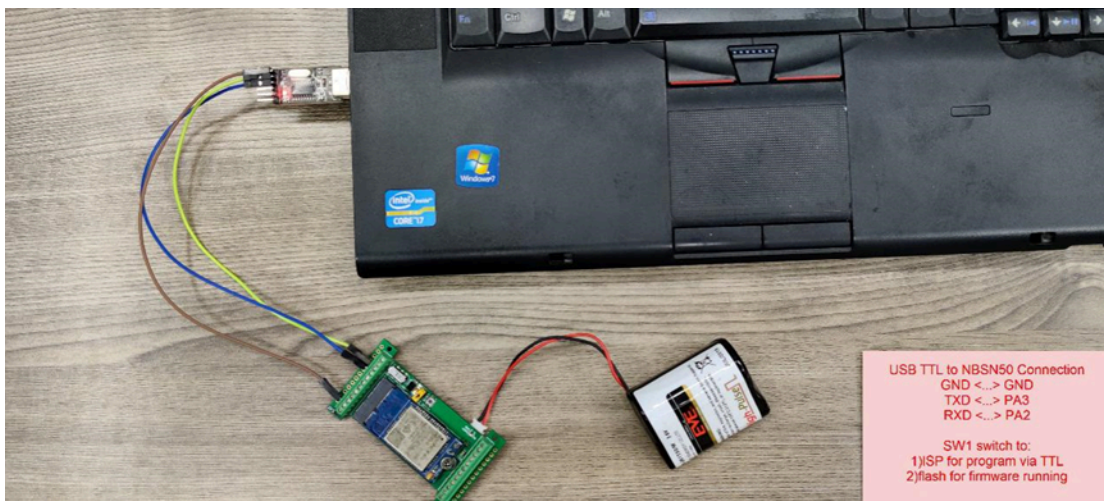
Insert the NB-IoT Card get from your provider.

User need to take out the NB-IoT module and insert the SIM card like below:



2.2.4 Connect USB – TTL to NBSN95 to configure it

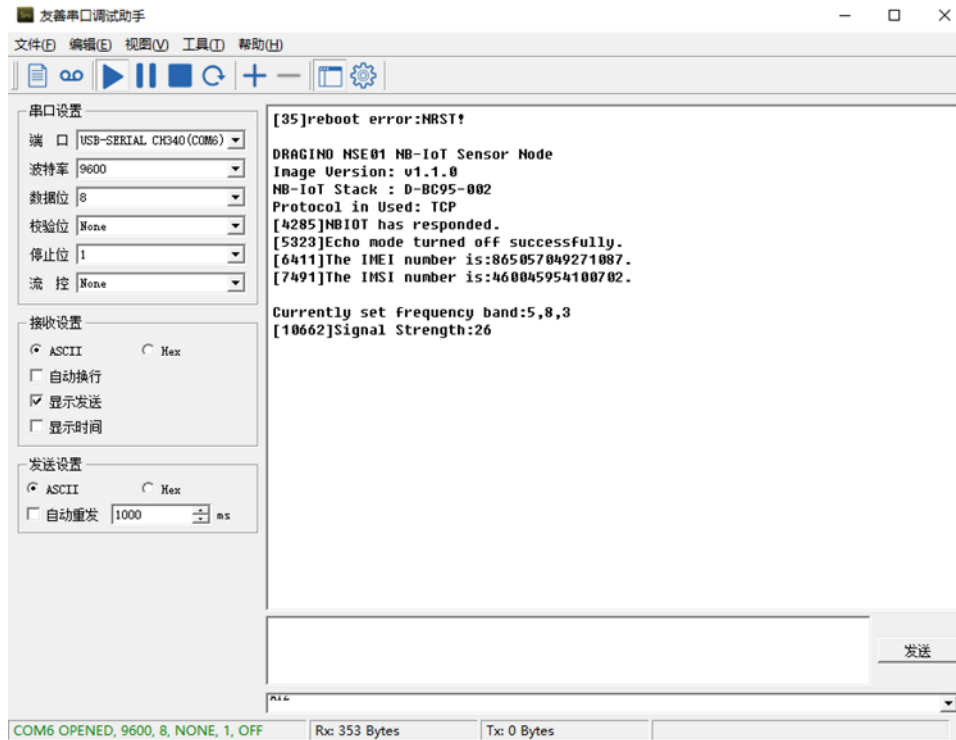
User need to configure NBSN95 via serial port to set the [Server Address / Uplink Topic](#) to define where and how-to uplink packets. NBSN95 support AT Commands, user can use a USB to TTL adapter to connect to NBSN95 and use AT Commands to configure it, as below.



In the PC, use below serial tool settings:

- Baud: **9600**
- Data bits: **8**
- Stop bits: **1**
- Parity: **None**
- Flow Control: **None**

Make sure the switch is in FLASH position, then power on device by connecting the jumper on NBSN95. NBSN95 will output system info once power on as below, we can enter the **password: 12345678** to access AT Command input.



Note: the valid AT Commands can be found at: <https://www.dropbox.com/sh/jao1xt9kw5r3yq4/AAAMpJkZzExF2JLbRWxGoQ9Na?dl=0>

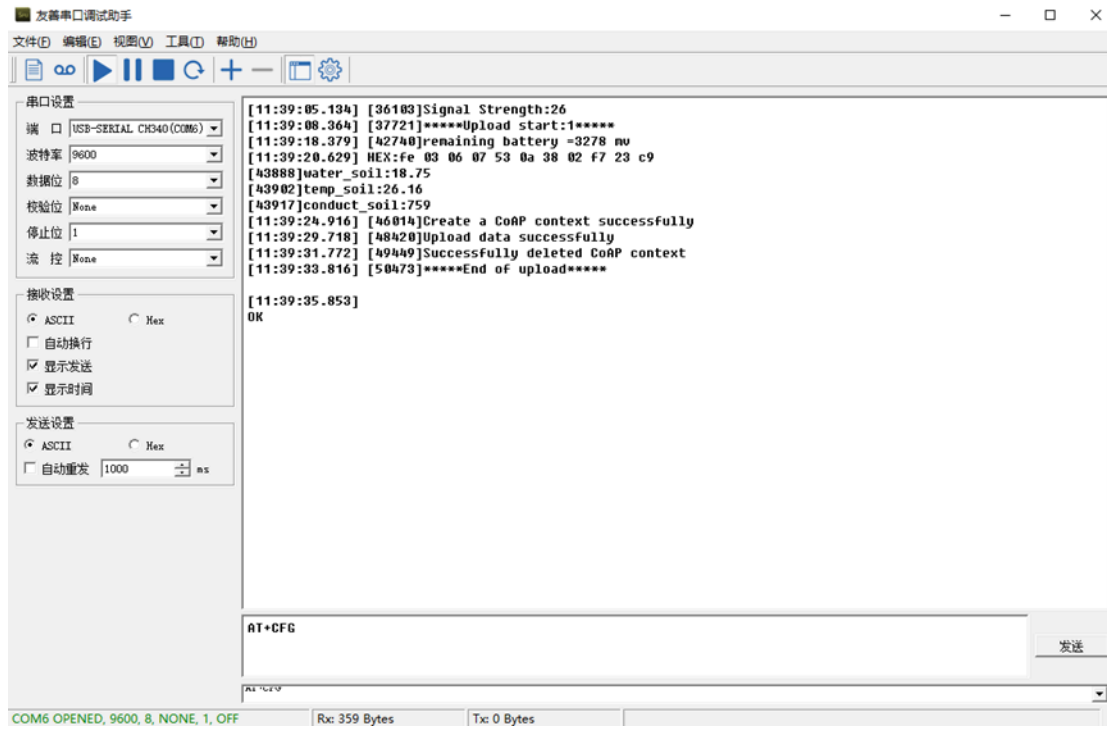
2.2.5 Use CoAP protocol to uplink data

Note: if you don't have CoAP server, you can refer this link to set up one: <http://wiki.dragino.com/xwiki/bin/view/Main/Set%20up%20CoAP%20Server/>

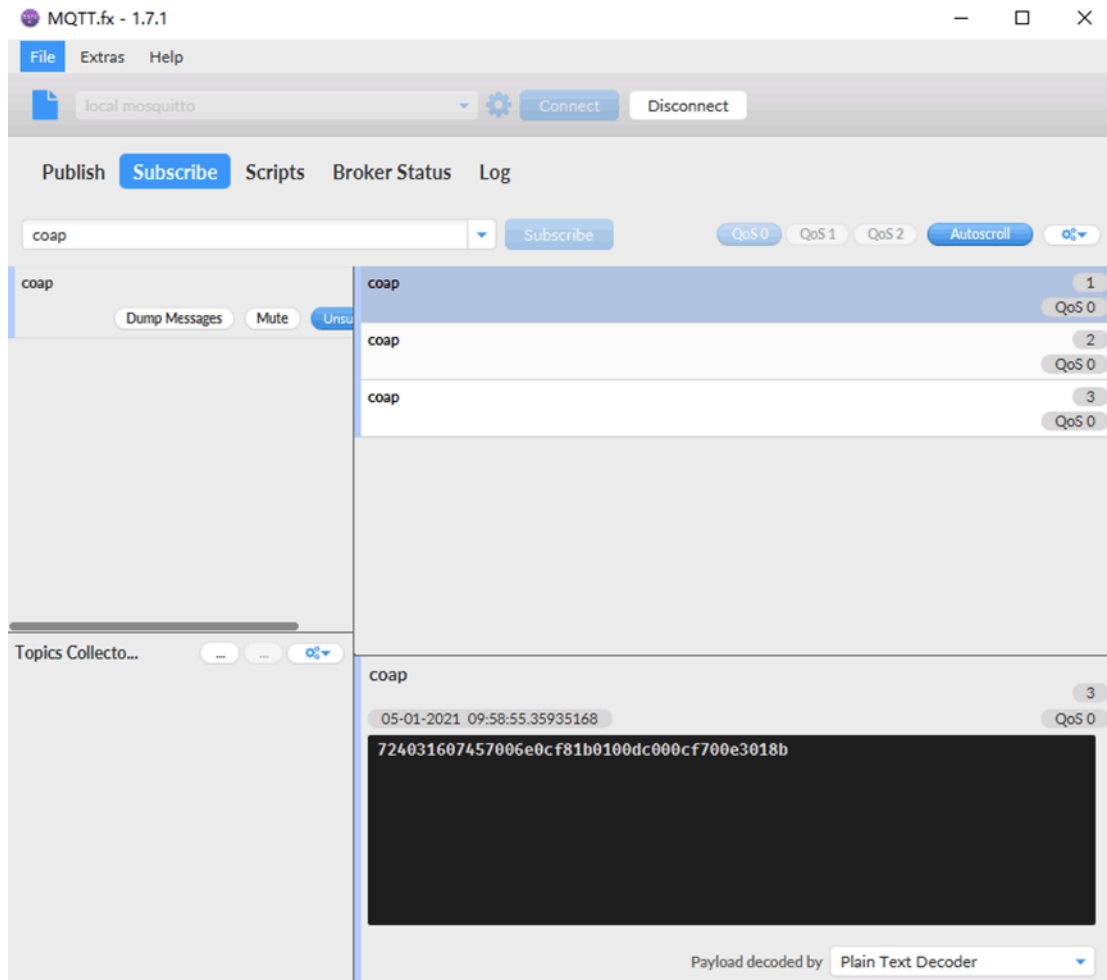
Use below commands:

- **AT+PRO=1** // Set to use CoAP protocol to uplink
- **AT+SERVADDR=120.24.4.116,5683** // to set CoAP server address and port
- **AT+URI=5,11,"mqtt",11,"coap",12,"0",15,"c=text1",23,"0"** //Set COAP resource path

For parameter description, please refer to AT command set



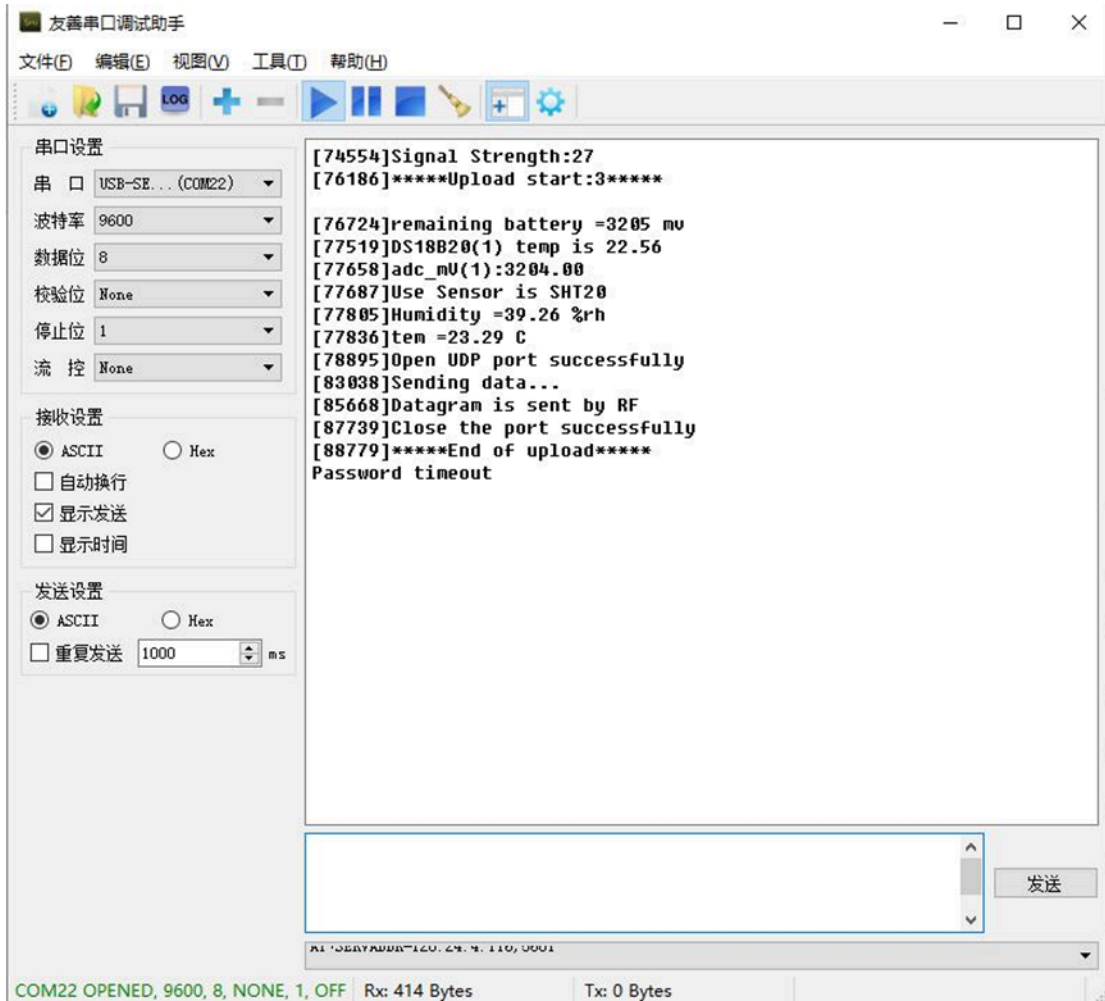
After configure the server address and **reset the device** (via AT+ATZ), NSE01 will start to uplink sensor values to CoAP server.

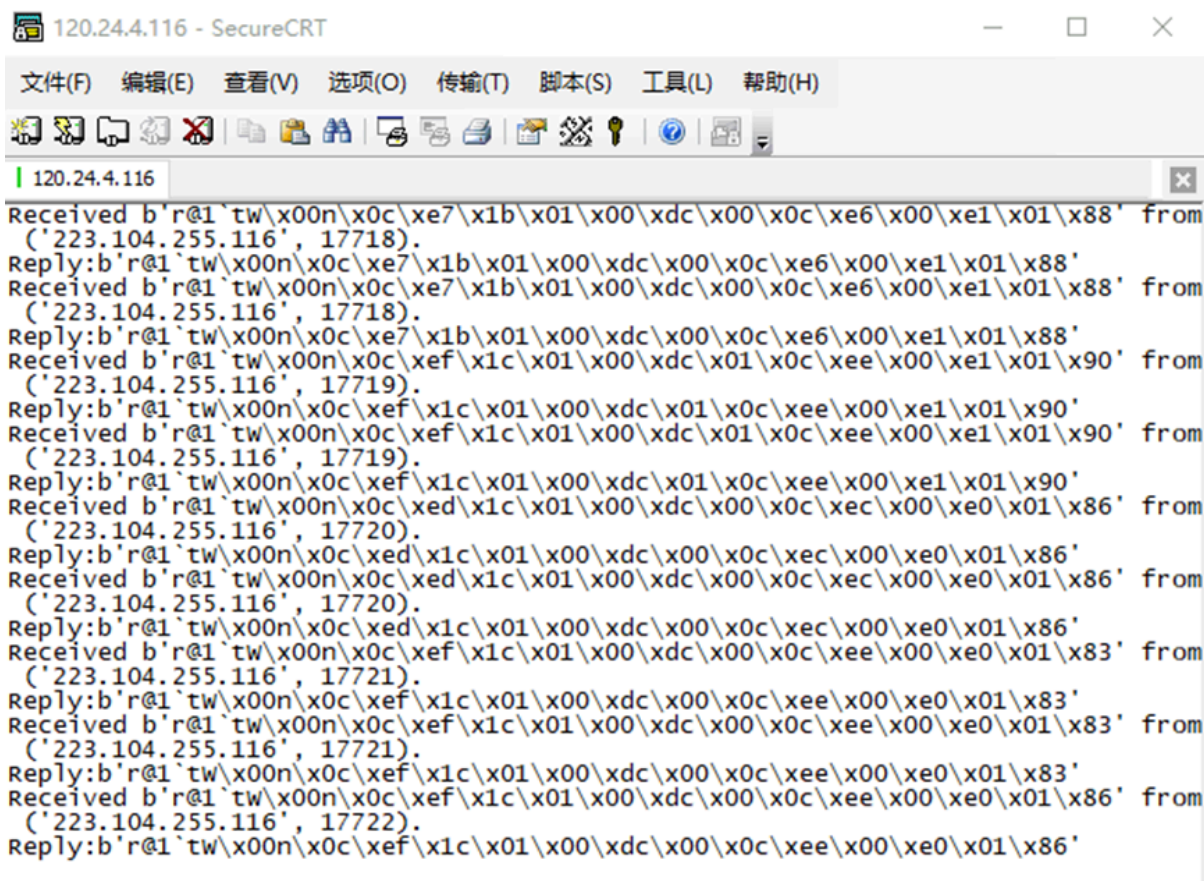


2.2.6 Use UDP protocol to uplink data(Default protocol)

This feature is supported since firmware version v1.0.1

- **AT+PRO=2** // Set to use UDP protocol to uplink
- **AT+SERVADDR=120.24.4.116,5601** // to set UDP server address and port
- **AT+CFM=1** // If the server does not respond, this command is unnecessary





```

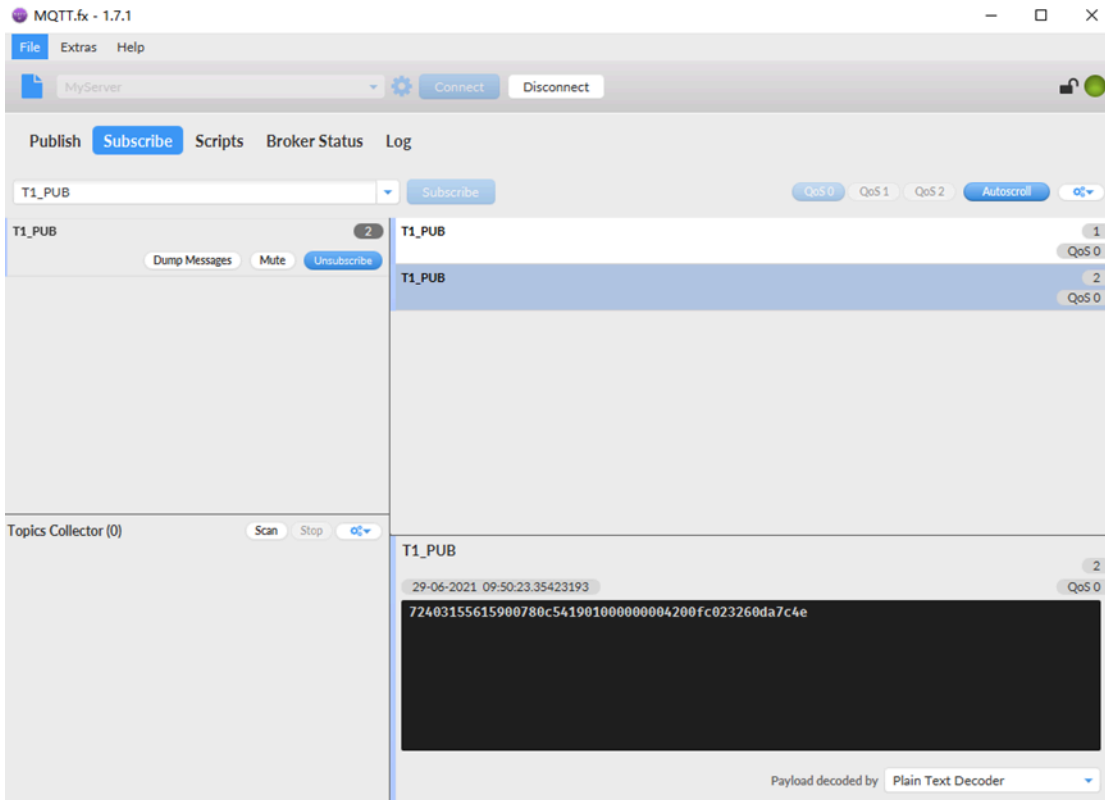
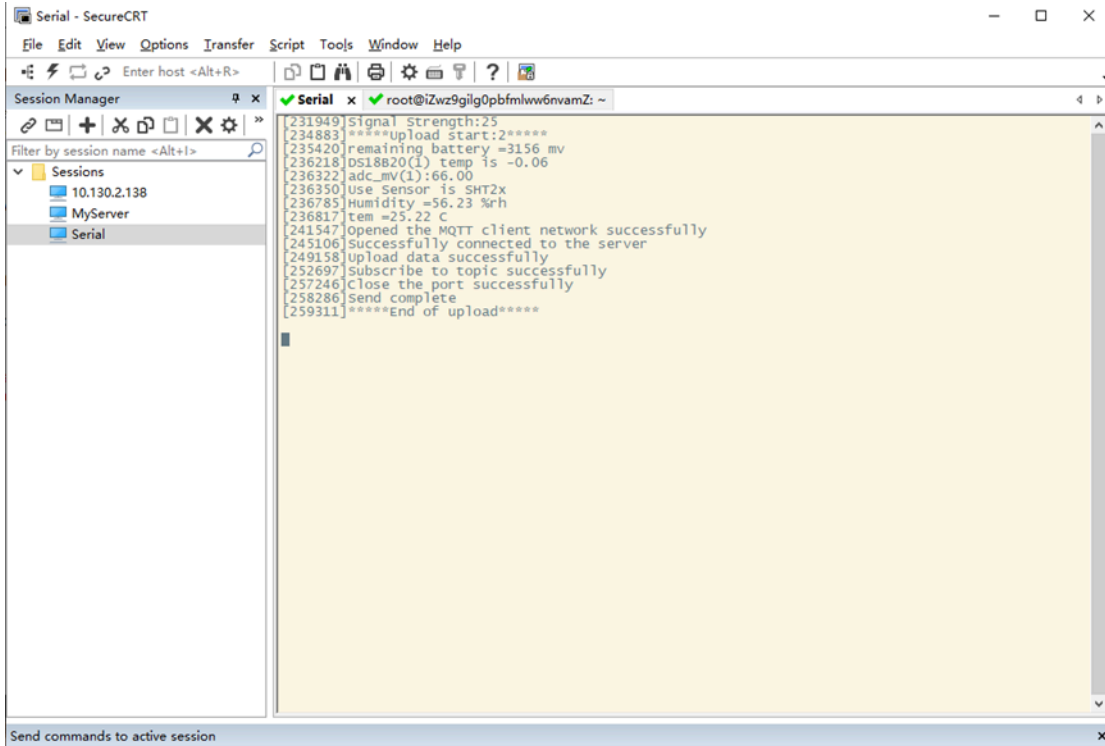
120.24.4.116 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
120.24.4.116
Received b'r@1'tw\x00n\x0c\xe7\x1b\x01\x00\xdc\x00\x0c\xe6\x00\xe1\x01\x88' from
('223.104.255.116', 17718).
Reply:b'r@1'tw\x00n\x0c\xe7\x1b\x01\x00\xdc\x00\x0c\xe6\x00\xe1\x01\x88'
Received b'r@1'tw\x00n\x0c\xe7\x1b\x01\x00\xdc\x00\x0c\xe6\x00\xe1\x01\x88' from
('223.104.255.116', 17718).
Reply:b'r@1'tw\x00n\x0c\xe7\x1b\x01\x00\xdc\x00\x0c\xe6\x00\xe1\x01\x88'
Received b'r@1'tw\x00n\x0c\xef\x1c\x01\x00\xdc\x01\x0c\xee\x00\xe1\x01\x90' from
('223.104.255.116', 17719).
Reply:b'r@1'tw\x00n\x0c\xef\x1c\x01\x00\xdc\x01\x0c\xee\x00\xe1\x01\x90'
Received b'r@1'tw\x00n\x0c\xef\x1c\x01\x00\xdc\x01\x0c\xee\x00\xe1\x01\x90' from
('223.104.255.116', 17719).
Reply:b'r@1'tw\x00n\x0c\xef\x1c\x01\x00\xdc\x01\x0c\xee\x00\xe1\x01\x90'
Received b'r@1'tw\x00n\x0c\xed\x1c\x01\x00\xdc\x00\x0c\xec\x00\xe0\x01\x86' from
('223.104.255.116', 17720).
Reply:b'r@1'tw\x00n\x0c\xed\x1c\x01\x00\xdc\x00\x0c\xec\x00\xe0\x01\x86'
Received b'r@1'tw\x00n\x0c\xed\x1c\x01\x00\xdc\x00\x0c\xec\x00\xe0\x01\x86' from
('223.104.255.116', 17720).
Reply:b'r@1'tw\x00n\x0c\xed\x1c\x01\x00\xdc\x00\x0c\xec\x00\xe0\x01\x86'
Received b'r@1'tw\x00n\x0c\xef\x1c\x01\x00\xdc\x00\x0c\xee\x00\xe0\x01\x83' from
('223.104.255.116', 17721).
Reply:b'r@1'tw\x00n\x0c\xef\x1c\x01\x00\xdc\x00\x0c\xee\x00\xe0\x01\x83'
Received b'r@1'tw\x00n\x0c\xef\x1c\x01\x00\xdc\x00\x0c\xee\x00\xe0\x01\x83' from
('223.104.255.116', 17721).
Reply:b'r@1'tw\x00n\x0c\xef\x1c\x01\x00\xdc\x00\x0c\xee\x00\xe0\x01\x83'
Received b'r@1'tw\x00n\x0c\xef\x1c\x01\x00\xdc\x00\x0c\xee\x00\xe0\x01\x86' from
('223.104.255.116', 17722).
Reply:b'r@1'tw\x00n\x0c\xef\x1c\x01\x00\xdc\x00\x0c\xee\x00\xe0\x01\x86'

```

2.2.7 Use MQTT protocol to uplink data

This feature is supported since firmware version v110, it supports only plain MQTT now it doesn't support TLS and other related encryption.

- **AT+PRO=3** // Set to use MQTT protocol to uplink
- **AT+SERVADDR=120.24.4.116,1883** // Set MQTT server address and port
- **AT+CLIENT=CLIENT** // Set up the CLIENT of MQTT
- **AT+UNAME=UNAME** // Set the username of MQTT
- **AT+PWD=PWD** // Set the password of MQTT
- **AT+PUBTOPIC=T1_PUB** // Set the sending topic of MQTT
- **AT+SUBTOPIC=T1_SUB** // Set the subscription topic of MQTT



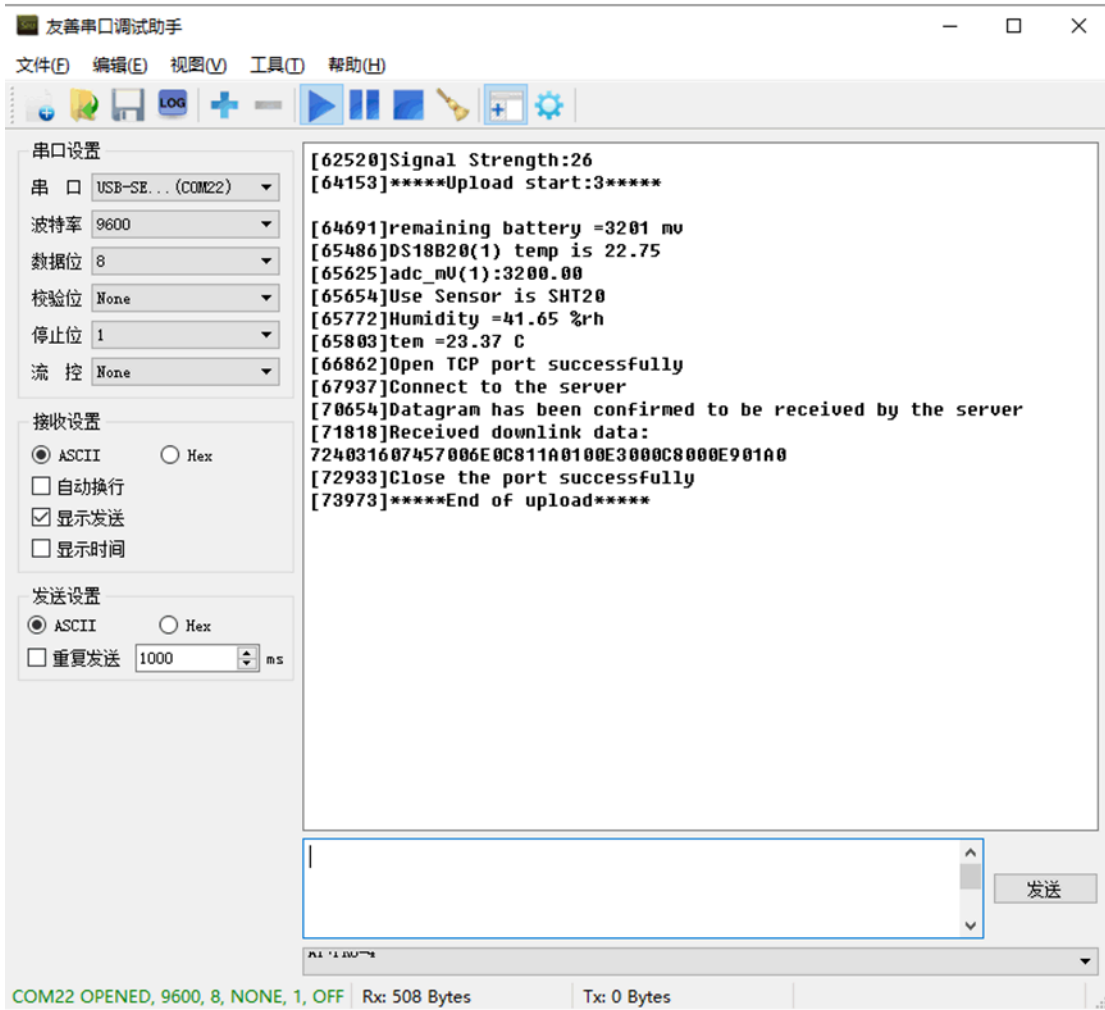
To save battery life, NBSN95 will establish a subscription before each uplink and close the subscription 3 seconds after uplink successful. Any downlink commands from server will only arrive during the subscription period.

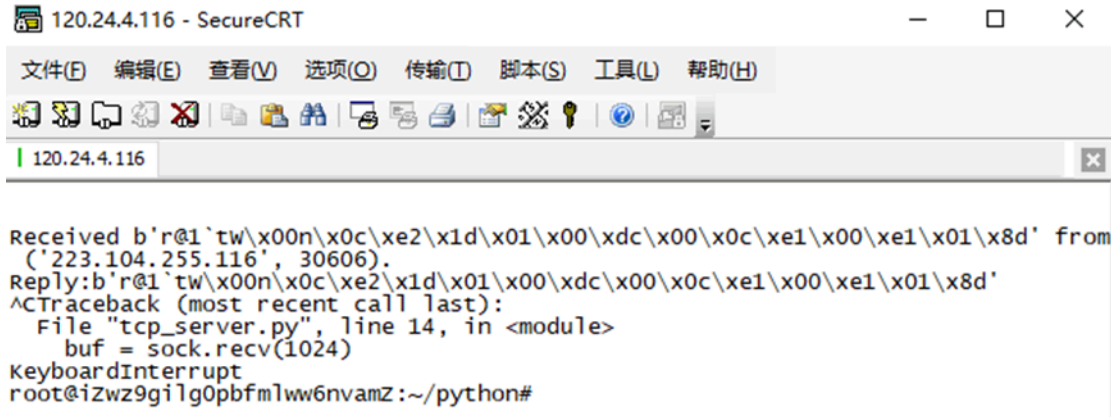
MQTT protocol has a much higher power consumption compare vs UDP / CoAP protocol. Please check the power analyze document and adjust the uplink period to a suitable interval.

2.2.8 Use TCP protocol to uplink data

This feature is supported since firmware version v110

- **AT+PRO=4** // Set to use TCP protocol to uplink
- **AT+SERVADDR=120.24.4.116,5600** // to set TCP server address and port





2.2.9 Change Update Interval

User can use below command to change the **uplink interval**.

- **AT+TDC=600** // Set Update Interval to 600s

2.3 Working Mode & Uplink Payload

NBSN95 has different working mode for the connections of different type of sensors. This section describes these modes. User can use the AT Command **AT+CFGMOD** to set NBSN95 to different working modes.

For example:

AT+CFGMOD=2 // will set the NBSN95 to work in MOD=2 distance mode which target to measure distance via [Ultrasonic Sensor](#).

The uplink payloads are composed in ASCII String. For example:

0a cd 00 ed 0a cc 00 00 ef 02 d2 1d (total 24 ASCII Chars). Representative the actually payload:

0x 0a cd 00 ed 0a cc 00 00 ef 02 d2 1d Total 12 bytes

NOTE:

1. All modes share the same Payload Explanation from [HERE](#).
2. By default, the device will send an uplink message every 1 hour.

2.3.1 CFGMOD=1 (Default Mode)

In this mode, the uplink payload usually contains 27 bytes. (Note: Time stamp field are added since firmware version v120)

Size(bytes)	8	2	2	1	1	2	1	2	2	2	4
Value	Device ID	Ver	BAT	Signal Strength	MOD 0x01	Temperature (DS18B20)	Digital in & Interrupt	ADC	Temperature by SHT20/SHT31	Humidity by SHT20/SHT31	Timestamp

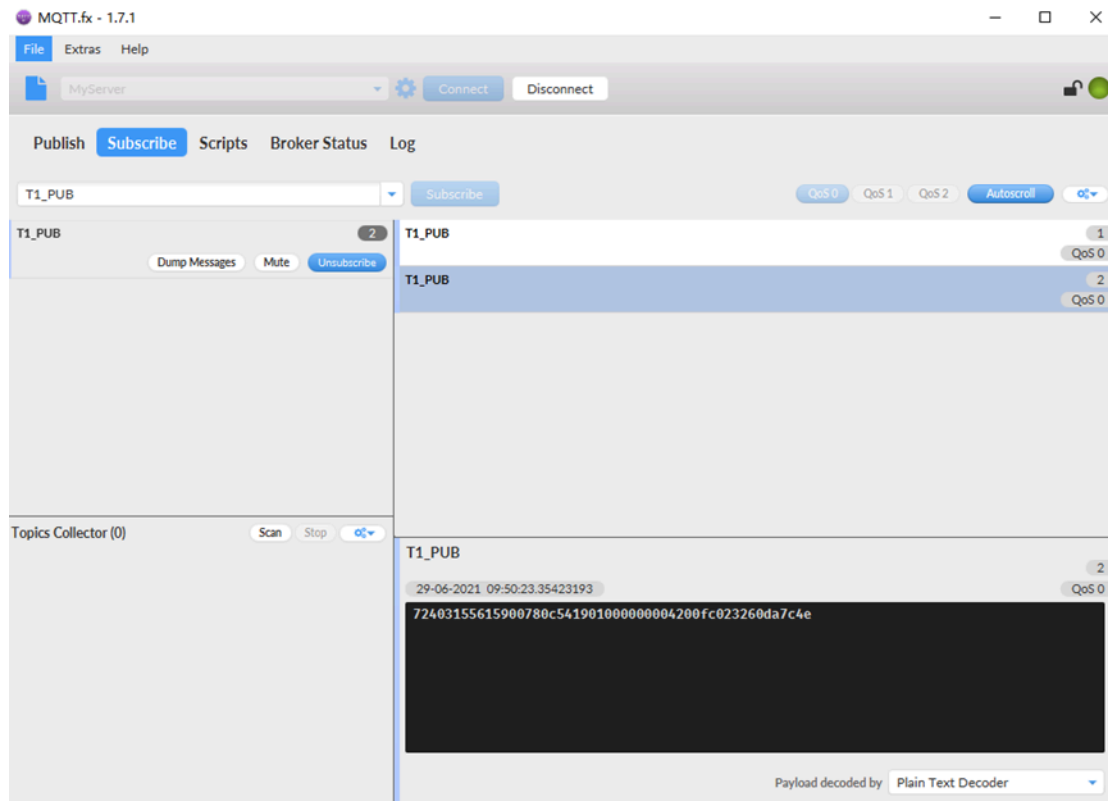
If the cache upload mechanism is turned on, you will receive the payload shown in the figure below.



NOTE:

1. Only up to 10 sets of latest data will be cached.
2. Theoretically, the maximum upload bytes are 215.

If we use the MQTT client to subscribe to this MQTT topic, we can see the following information when the NB sensor uplink data.



The payload is ASCII string, representative same HEX: **0x f868411056754138 0078 0c54 19 01 0000 00 0042 00fc 0232 60da7c4e**

where:

- **Device ID:** 0xf868411056754138 = 868411056754138
- **Version:** 0x0078=120=1.2.0
- **BAT:** 0x0c54 = 3156 mV = 3.156V
- **Singal:** 0x19 = 25
- **Model:** 0x01 = 1
- **Temperature by DS18B20:** 0x0000 = 0
- **Interrupt:** 0x00 = 0
- **ADC:** 0x0042 = 66 = 66mv
- **Temperature by SHT20/SHT31:** 0x00fc = 252 = 25.2 °C
- **Humidity by SHT20/SHT31:** 0x0232 = 562 = 56.2 %rh
- **Timestamp:** 0x60da7c4e = 1,624,931,406 = 2021-06-29 09:50:06

2.3.2 CFGMOD=2 (Distance Mode)

This mode is target to measure the distance. Total 25 bytes, (**Note: Time stamp field are added since firmware version v120**)

Size(bytes)	8	2	2	1	1	2	1	2	2	4
Value	Device ID	Ver	BAT	Signal Strength	MOD 0x02	Temperature (DS18B20)	Digital in & Interrupt	ADC	Distance measure by: 1) LIDAR-Lite V3HP Or 2) Ultrasonic Sensor	Timestamp

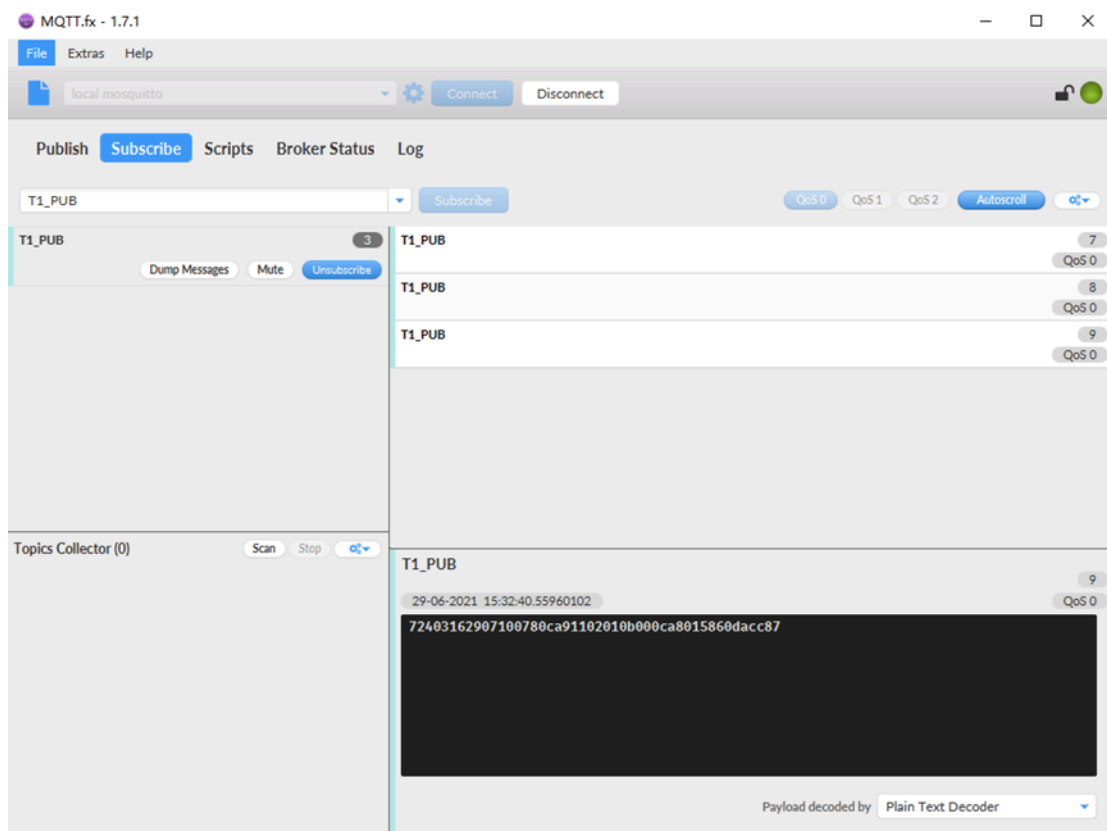
If the cache upload mechanism is turned on, you will receive the payload shown in the figure below.



NOTE:

1. Only up to 10 sets of latest data will be cached.
2. Theoretically, the maximum upload bytes are 193.

If we use the MQTT client to subscribe to this MQTT topic, we can see the following information when the NB sensor uplink data.

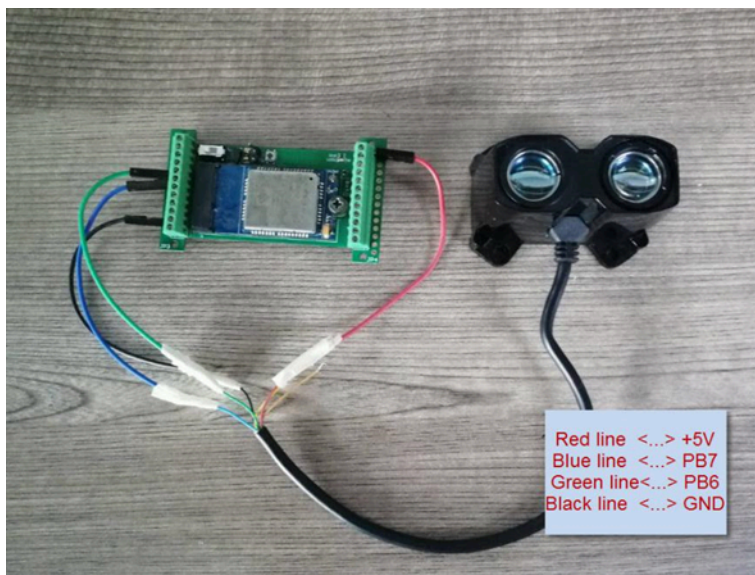


So the payload is **0xf868411056754138 0078 0ca9 11 02 010b 00 0ca8 0158 60dacc87**

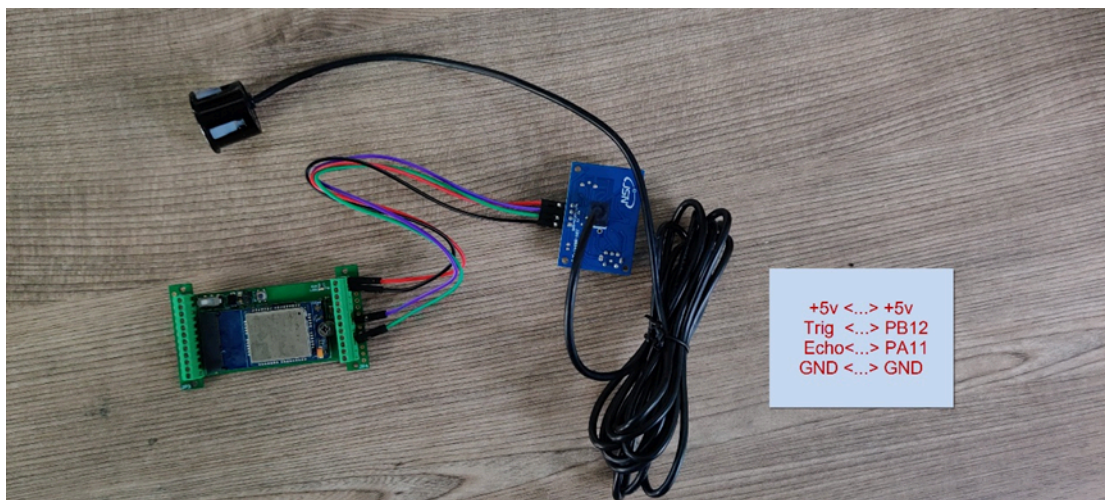
where:

- **Device ID:** 0xf868411056754138 = 868411056754138
- **Version:** 0x0078=120=1.2.0
- **BAT:** 0x0ca9 = 3241mV = 3.241 V
- **Singal:** 0x11 = 17
- **Model:** 0x02 = 2
- **Temperature by DS18B20:** 0x010b= 267 = 26.7 °C
- **Interrupt:** 0x00 = 0
- **ADC:** 0x0ca8 = 3240 mv
- **Distance by LIDAR-Lite V3HP/Ultrasonic Sensor:** 0x0158 = 344 cm
- **Timestamp:** 0x60dacc87 = 1,624,951,943 = 2021-06-29 15:32:23

Connection of LIDAR-Lite V3HP:



Connection to Ultrasonic Sensor:



2.3.3 CFGMOD=3 (3 ADC + I2C)

This mode has total 29 bytes. Include 3 x ADC + 1x I2C, (**Note: Time stamp field are added since firmware version v120**)

Size(bytes)	8	2	2	1	1	2	1	2	2	2	2	4
Value	Device ID	Ver	BAT	Signal Strength	MOD 0x03	ADC1	Digital in & Interrupt	ADC2	Temperature by SHT20/SHT31	Humidity by SHT20/SHT31	ADC3	Timestamp

ADC1 uses pin PA0 to measure

ADC2 uses pin PA1 to measure

ADC3 uses pin PA4 to measure

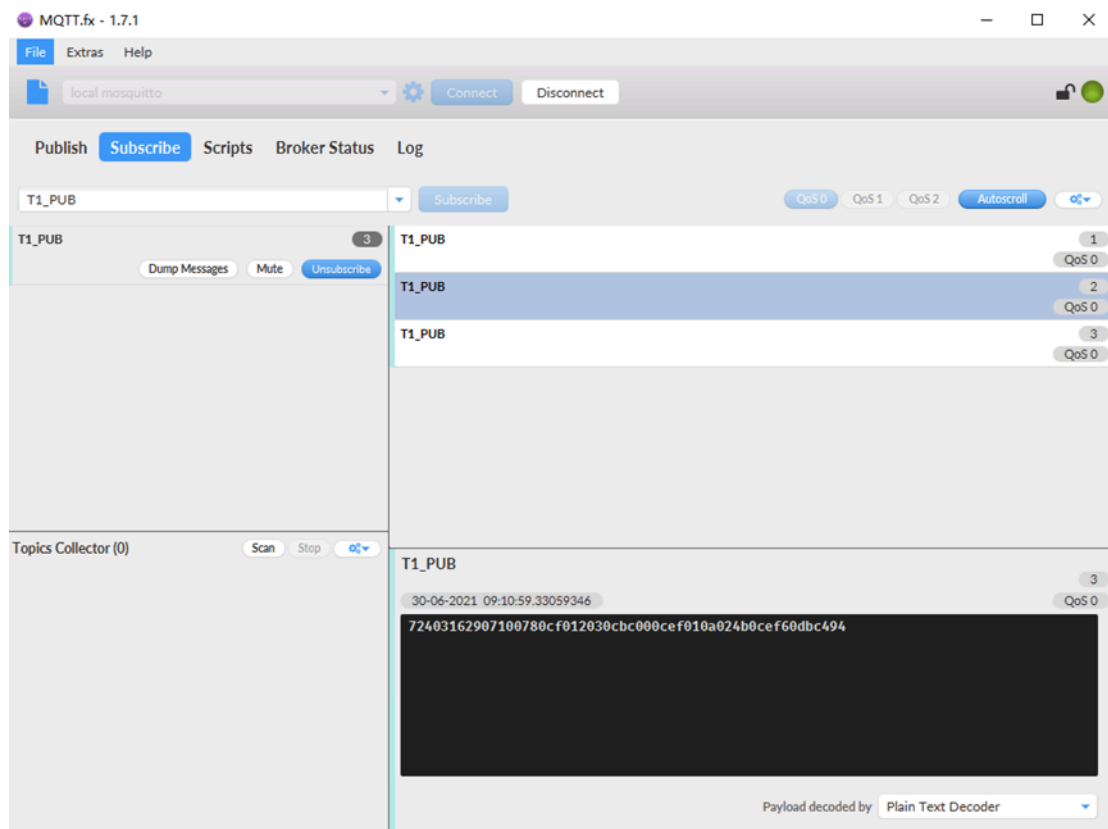
If the cache upload mechanism is turned on, you will receive the payload shown in the figure below.

Frame header	Frame data(1)	Frame data(2)	F...	Frame data(X)
--------------	---------------	---------------	------	---------------

NOTE:

1. Only up to 10 sets of latest data will be cached.
2. Theoretically, the maximum upload bytes are 226.

If we use the MQTT client to subscribe to this MQTT topic, we can see the following information when the NB sensor uplink data.



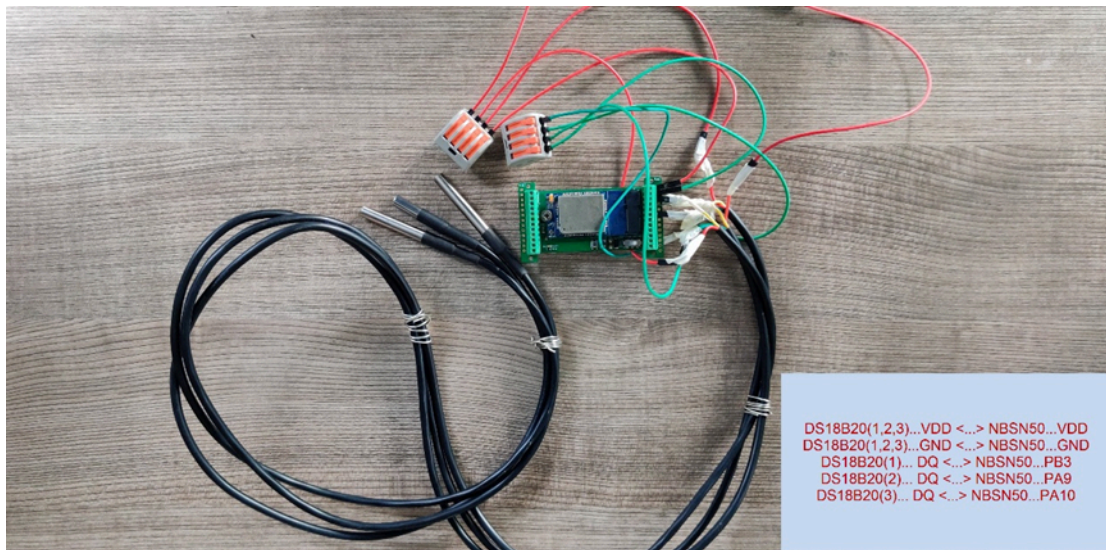
So the payload is **0x f868411056754138 78 0cf0 12 03 0cbc 00 0cef 010a 024b 0cef 60dbc494**

where:

- **Device ID:** 0xf868411056754138 = 868411056754138
- **Version:** 0x78=120=1.2.0
- **BAT:** 0x0cf0 = 3312 mV = 3.312 V
- **Singal:** 0x12 = 18
- **Model:** 0x03 = 3
- **ADC1:** 0x0cbc= 3260mV
- **Interrupt:** 0x00 = 0
- **ADC2:** 0x0cef =3311 mv
- **Temperature by SHT20/SHT31:** 0x010a = 266 = 26.6 °C
- **Humidity by SHT20/SHT31:** 0x024b =587 = 58.7 %rh
- **ADC3:** 0x0cef = 3311 mv
- **Timestamp:** 0x60dbc494 = 1,625,015,444= 2021-06-30 09:10:44

2.3.4 CFGMOD=4 (3 x DS18B20)

Hardware connection is as below, (**Note: R3 & R4 should change from 10k to 4.7k to support DS18B20, Software set to AT+CFGMOD=4**)



This mode has total 27 bytes. (**Note: Time stamp field are added since firmware version v120**) As shown below:

Size(bytes)	8	2	2	1	1	2	2	1	2	2	4
Value	Device ID	Ver	BAT	Signal Strength	MOD 0x04	Temperature1 (DS18B20) (PB3)	ADC	Digital in & Interrupt	Temperature2 (DS18B20) (PA9)	Temperature3 (DS18B20) (PA10)	Timestamp

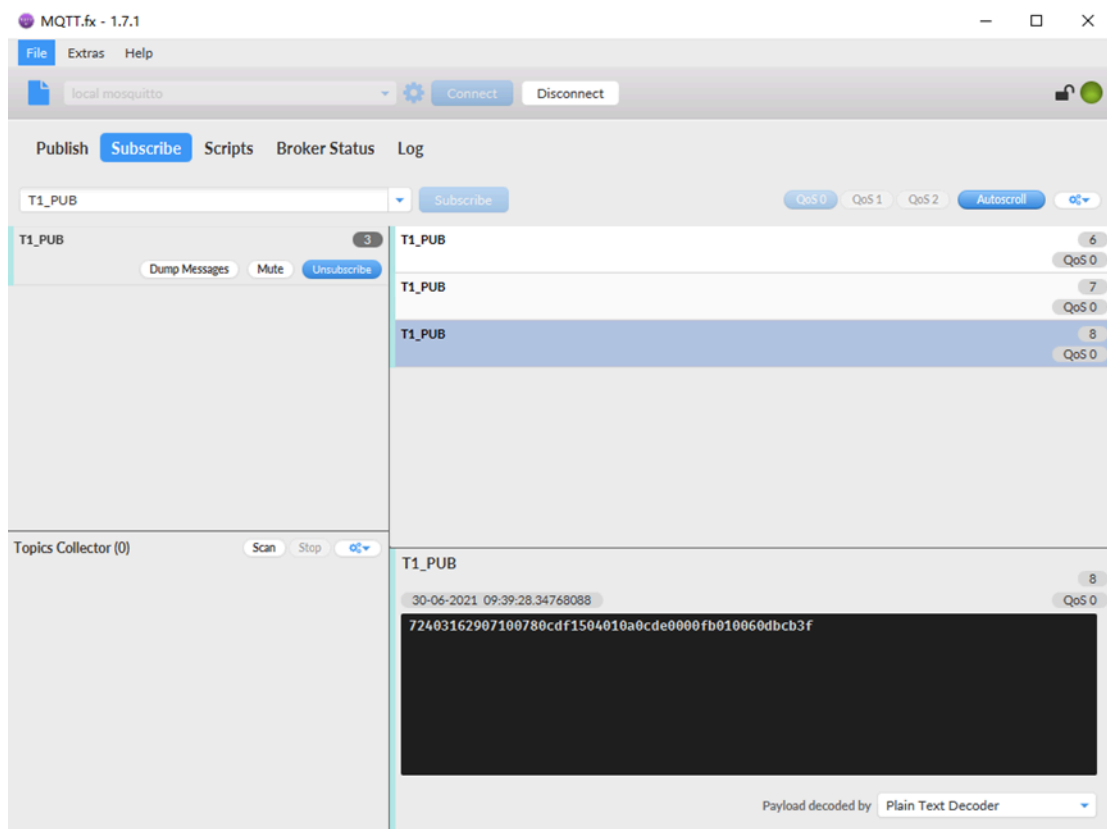
If the cache upload mechanism is turned on, you will receive the payload shown in the figure below.



NOTE:

1. Only up to 10 sets of latest data will be cached.
2. Theoretically, the maximum upload bytes is 215.

If we use the MQTT client to subscribe to this MQTT topic, we can see the following information when the NB sensor uplink data.

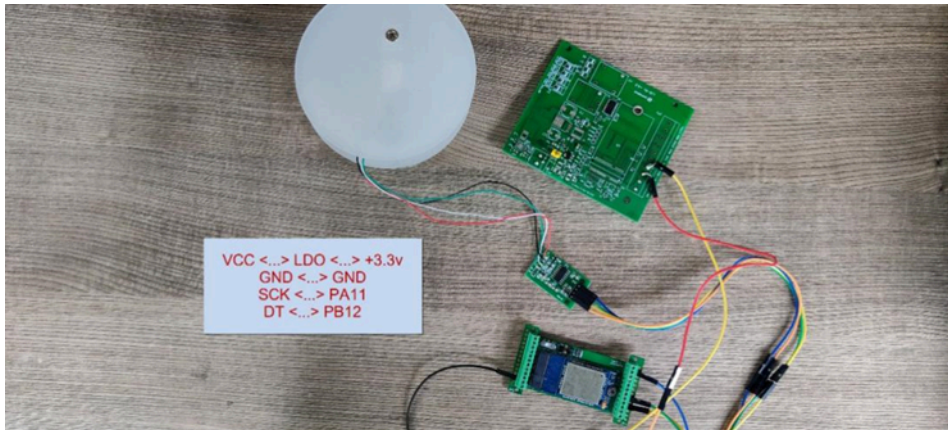


So the payload is **0x f868411056754138 0078 0cdf 15 04 010a 0cde 00 00fb 0100 60dbcb3f**

where:

- **Device ID:** 0xf868411056754138 = 868411056754138
- **Version:** 0x0078=120=1.2.0
- **BAT:** 0x0cdf = 3295 mV = 3.295 V
- **Singal:** 0x15 = 21
- **Model:** 0x04 = 4
- **Temperature1 by DS18B20:** 0x010a = 226 = 22.6 °C
- **ADC:** 0x0cde = 3294 mv
- **Interrupt:** 0x00 = 0
- **Temperature2 by DS18B20:** 0x00fb = 251 = 25.1 °C
- **Temperature3 by DS18B20:** 0x0100 = 256 = 25.6 °C
- **Timestamp:** 0x60dbcb3f = 1,625,017,151= 2021-06-30 09:39:11

2.3.5 CFGMOD=5 (Weight Measurement by HX711)



Notes about hardware connection:

1. Don't connect the HX711 module VCC to NBSN95 3.3v VCC, in this case, the NBSN95 will always power on HX711 and the battery will run out soon.
2. HX711 support 5v VCC, but while connect the NBSN95's +5V to HX711 VCC, the value from HX711 is not stable.
3. Connect NBSN95 +5V to HX711 VCC via a LDO module is stable.

Each HX711 need to be calibrated before used. User need to do below two steps:

1. Zero calibration. Don't put anything on load cell and run **AT+WEIGRE** to calibrate to Zero gram.
2. Adjust calibration factor (default value 400): Put a known weight thing on load cell and run **AT+WEIGAP** to adjust the Calibration Factor.

For example:

AT+WEIGAP =403.0

Response: Weight is 401 g

Check the response of this command and adjust the value to match the real value for thing.

This mode has total 25 bytes. (**Note: Time stamp field are added since firmware version v120**). As shown below:

Size(bytes)	8	2	2	1	1	2	2	1	2	4
Value	Device ID	Ver	BAT	Signal Strength	MOD 0x05	Temperature (DS18B20)	ADC	Digital in & Interrupt	Weight	Timestamp

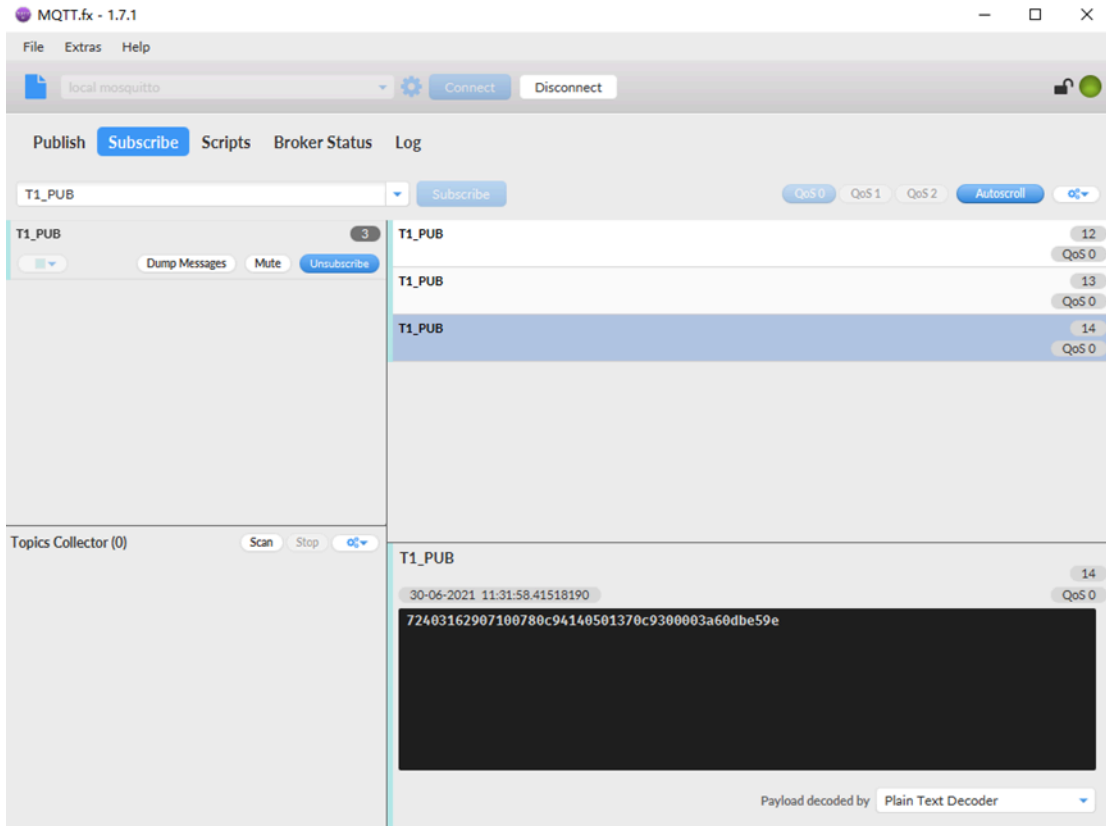
If the cache upload mechanism is turned on, you will receive the payload shown in the figure below.

Frame header	Frame data (1)	Frame data (2)	F...	Frame data(X)
---------------------	-----------------------	-----------------------	-------------	----------------------

NOTE:

1. **Only up to 10 sets of latest data will be cached.**
2. **Theoretically, the maximum upload bytes are 193.**

If we use the MQTT client to subscribe to this MQTT topic, we can see the following information when the NB sensor uplink data.



So the payload is **0x f868411056754138 0078 0c94 14 05 0137 0c93 00 003a 60dbe59e**

where:

- **Device ID:** 0xf868411056754138 = 868411056754138
- **Version:** 0x0078=120=1.2.0
- **BAT:** 0x0c94 = 3220 mV = 3.220 V
- **Singal:** 0x14 = 20
- **Model:** 0x05 = 5
- **Temperature by DS18b20:** 0x0137 = 311 = 31.1 °C
- **ADC:** 0x0c93 = 3219 mv
- **Interrupt:** 0x00 = 0
- **Weigt by HX711:** 0x003a = 58 g
- **Timestamp:** 0x60dbe59e = 1,625,023,902= 2021-06-30 11:31:42

2.3.6 CFGMOD=6 (Counting mode)

In this mode, uplink payload includes in total 22 bytes, (**Note: Time stamp field are added since firmware version v120**)

Size(bytes)	8	2	2	1	1	4	4
Value	Device ID	Ver	BAT	Signal Strength	MOD 0x06	Pulse count	Timestamp

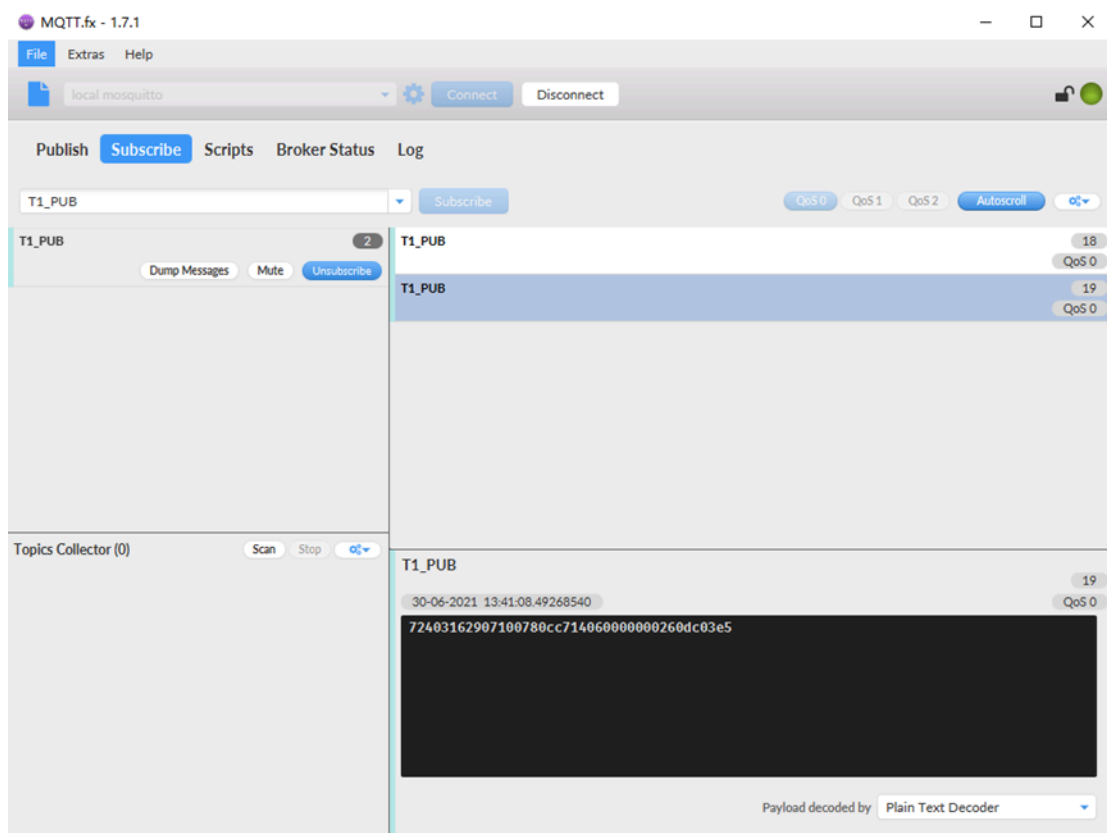
If the cache upload mechanism is turned on, you will receive the payload shown in the figure below.

Frame header	Frame data (1)	Frame data (2)	F...	Frame data(X)
--------------	----------------	----------------	------	---------------

NOTE:

1. Only up to 10 sets of latest data will be cached.
2. Theoretically, the maximum upload bytes are 160.

If we use the MQTT client to subscribe to this MQTT topic, we can see the following information when the NB sensor uplink data.



The payload is ASCII string, representative same HEX: **0x f868411056754138 0078 0cc7 14 06 00000002 60dc03e5**

where:

- **Device ID:** 0xf868411056754138 = 868411056754138
- **Version:** 0x0078=120=1.2.0
- **BAT:** 0x0cc7 =3271mV =3.271V
- **Singal:** 0x14 = 20
- **Model:** 0x06 = 6
- **Pulse count:** 0x00000002= 2
- **Timestamp:** 0x60dc03e5 = 1,625,031,653= 2021-06-30 13:40:53

2.4 Payload Explanation and Sensor Interface

2.4.1 Device ID

By default, the Device ID equal to the last 15 bits of IMEI.

User can use **AT+DEUI** to set Device ID

Example:

AT+DEUI=868411056754138

The Device ID is stored in a none-erase area, Upgrade the firmware or run AT+FDR won't erase Device ID.

2.4.2 Version Info

These bytes include the hardware and software version.

Higher byte: Specify hardware version: always 0x01 for NBSN95

Lower byte: Specify the software version: 0x6E=110, means firmware version 110

For example: 0x01 6E: this device is NBSN95 with firmware version 110.

2.4.3 Battery Info

Ex1: 0x0B45 = 2885mV

Ex2: 0x0B49 = 2889mV

2.4.4 Signal Strength

NB-IoT Network signal Strength.

Ex1: 0x1d = 29

0 -113dBm or less

1 -111dBm

2...30 -109dBm... -53dBm

31 -51dBm or greater

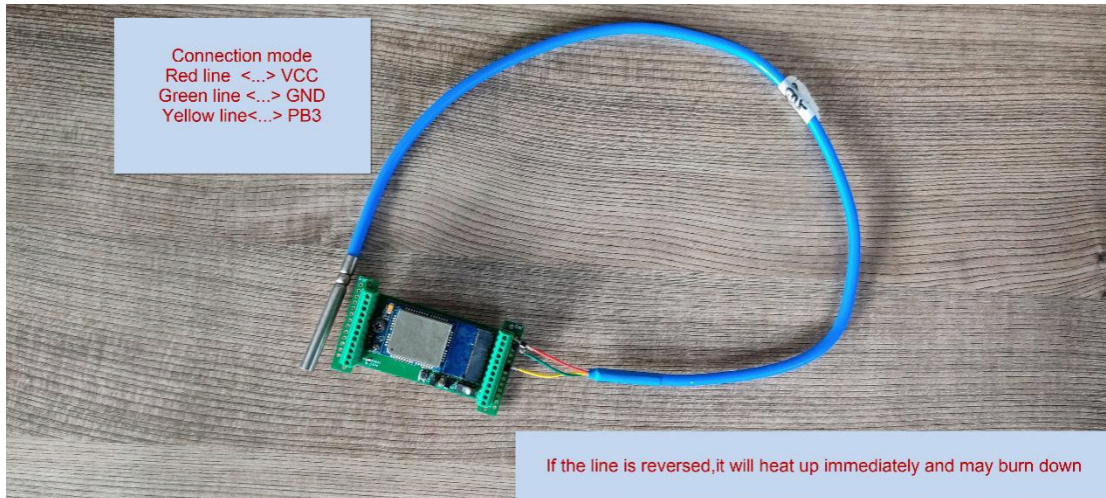
99 Not known or not detectable

2.4.5 Temperature (DS18B20)

If there is a DS18B20 connected to PB3 pin. The temperature will be uploaded in the payload.

More DS18B20 can check the [3 DS18B20 mode](#)

Connection for one DS18B20



Example:

If payload is: 0x0105: (0105 & FC00 == 0), temp = 0x0105 /10 = 26.1 degree

If payload is: 0xFF3F: (FF3F & FC00 == 1) , temp = (0xFF3F - 65536)/10 = -19.3 degree.

2.4.6 Digital Input

The digital input is for pin PA12,

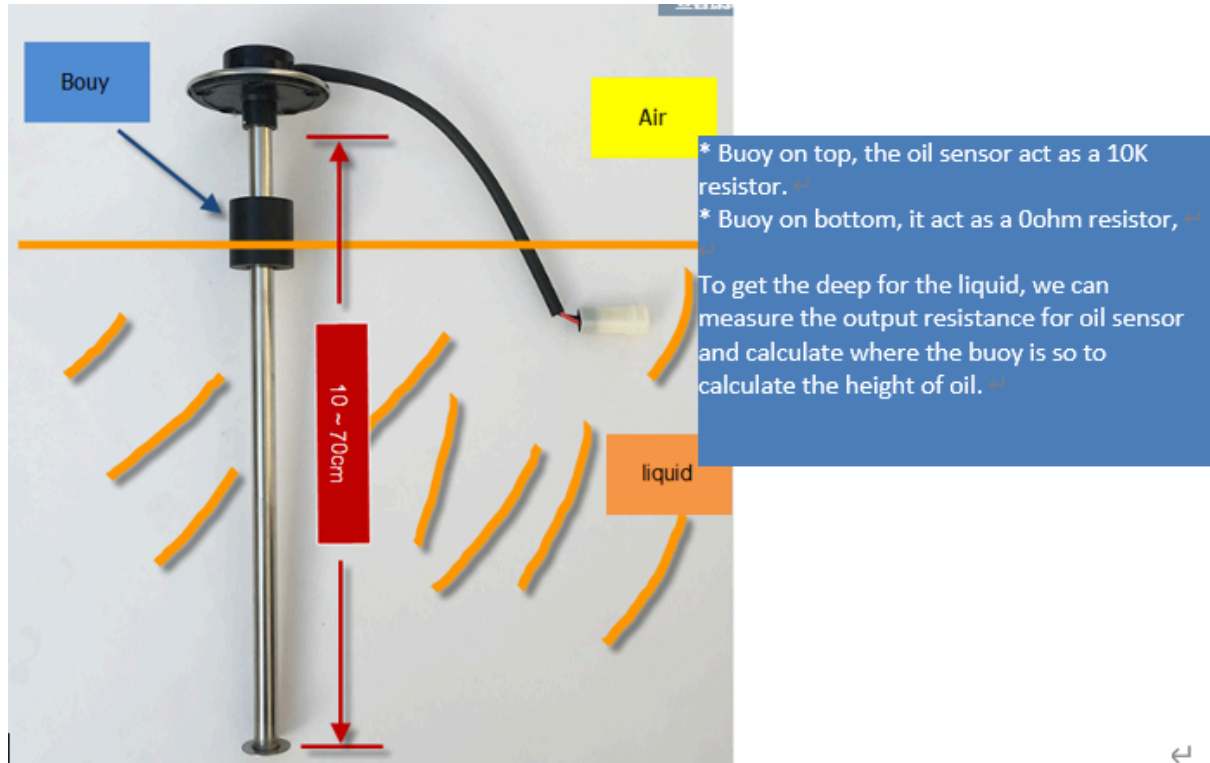
- When PA12 is high, the bit2 of this byte is 1.
- When PA12 is low, the bit2 of this byte is 0.

2.4.7 Analogue Digital Converter (ADC)

The ADC monitors the voltage on the **PA0** line, in mV. Max value is from 0v to BAT voltage

Ex: 0x021F = 543mv,

Example1: Reading a Liquid Level Sensor (Read a resistance value):

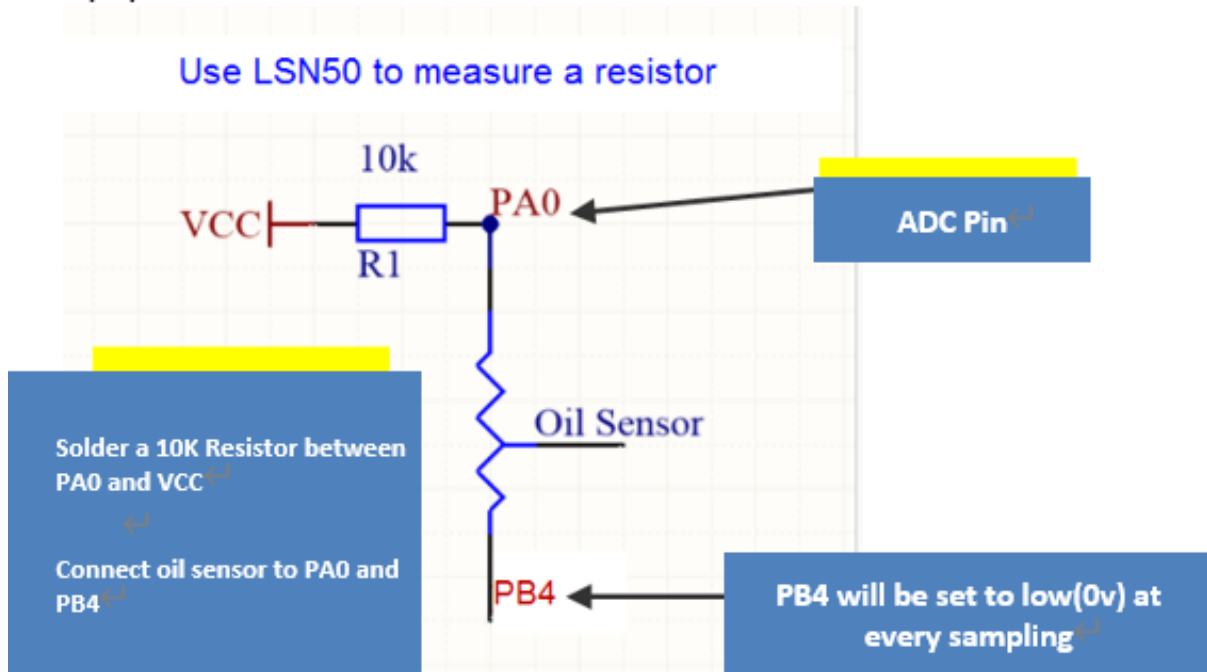


In the NBSN95, we can use PB4 and PA0 pin to calculate the resistance for the liquid level sensor. The bottom of this sensor equal to 0ohm and top position equals to 10kohm.

Steps:

1. Solder a 10K resistor between PA0 and VCC.
2. Screw liquid level sensor's two pins to PA0 and PB4.

The equipment circuit is as below:



According to above diagram:

$$\frac{VCC - V_{PA0}}{10k} = \frac{V_{PA0}}{R_{oil\ sensor}}$$

So

$$R_{oil\ sensor} = V_{PA0} \times 10 \frac{K}{(VCC - V_{PA0})}$$

V_{PA0} is the reading of ADC. So if ADC=0x05DC=0.9 v and VCC (BAT) is 2.9v

The $R_{oil\ sensor} = 0.9 \times \frac{10K}{2.9-0.9} = 4.5K$ ohm

Since the buoy is linear resistance from 10 ~ 70cm.

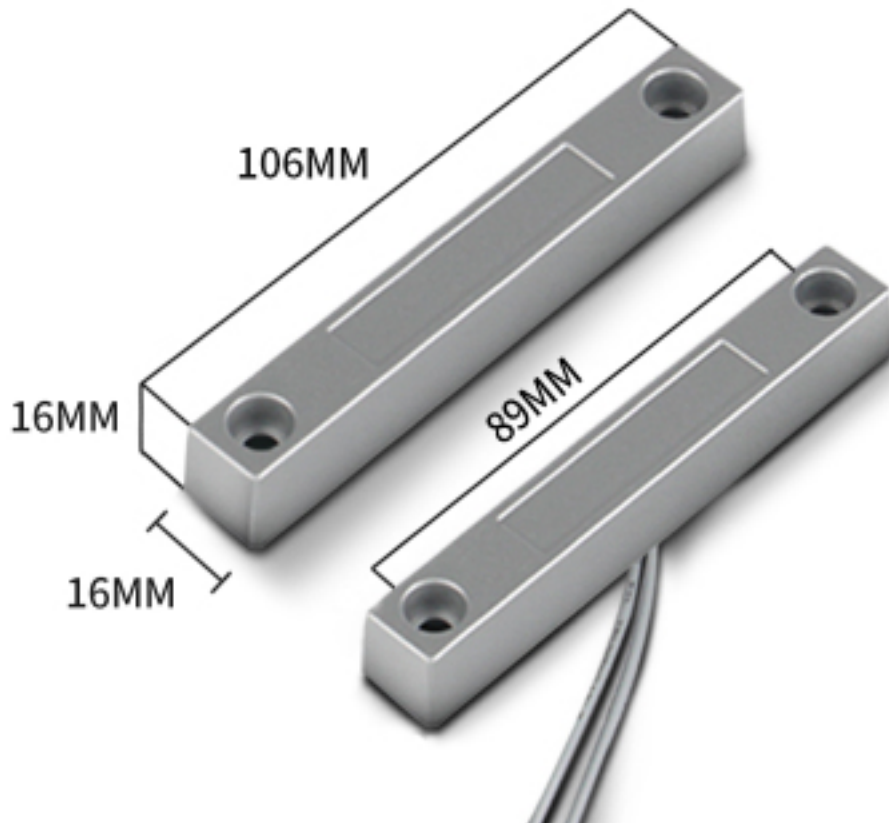
The position of buoy is $\frac{4.5K}{10K} \times (70cm - 10cm) + 10cm = 37cm$, from the bottom of buoy.

2.4.8 Digital Interrupt

Digital Interrupt refers to pin PB14, and there are different trigger methods. When there is a trigger, the NBSN95 will send a packet to the server.

Example to use with door sensor

The door sensor is shown at right. It is a two wire magnetic contact switch used for detecting the open/close status of doors or windows.



When the two pieces are close to each other, the 2 wire output will be short or open (depending on the type), while if the two pieces are away from each other, the 2 wire output will be the opposite status. So we can use NBSN95 interrupt interface to detect the status for the door or window.

Below is the installation example:

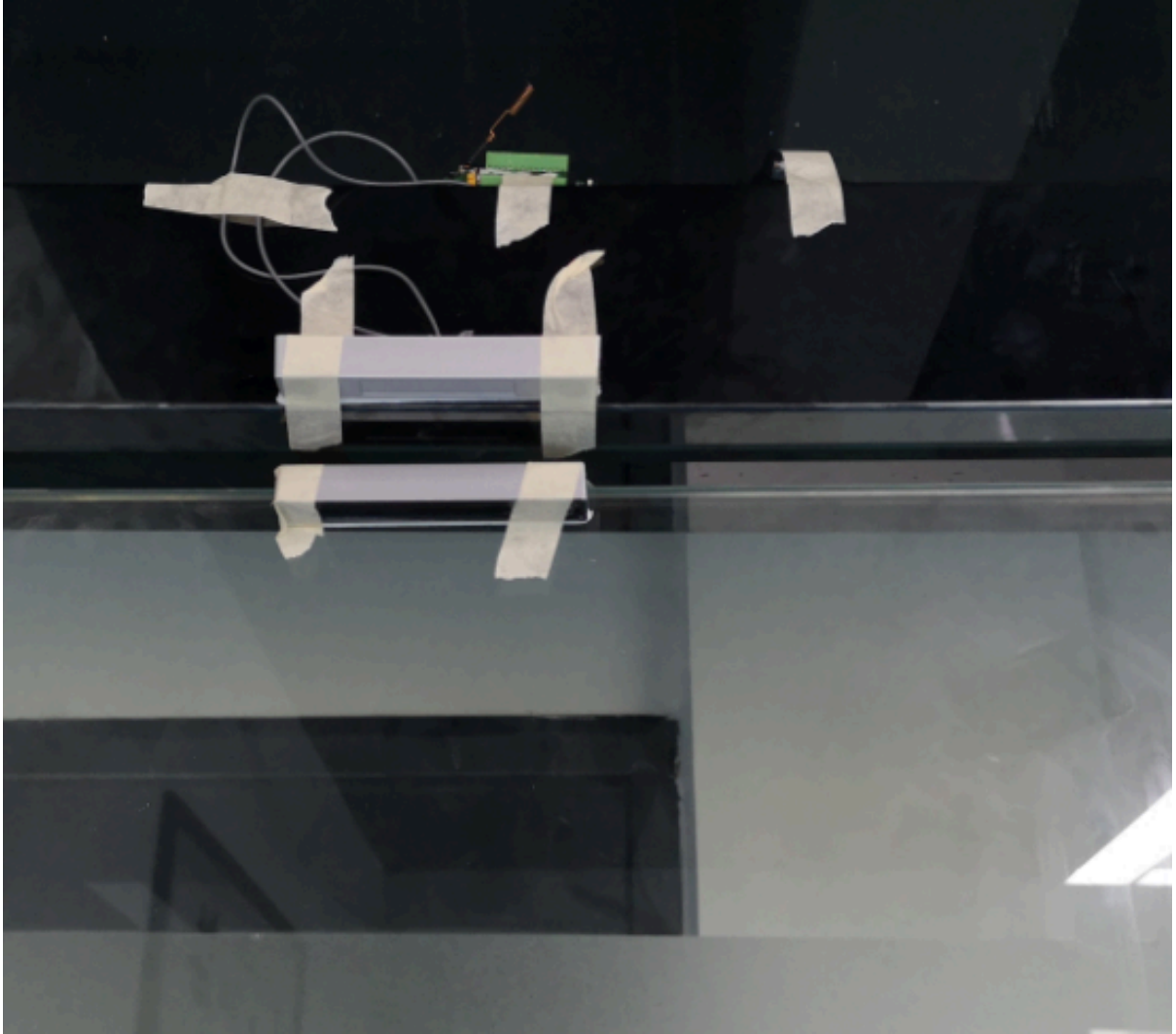
Fix one piece of the magnetic sensor to the door and connect the two pins to NBSN95 as follows:

- One pin to NBSN95's PB14 pin
- The other pin to NBSN95's VCC pin

Install the other piece to the door. Find a place where the two pieces will be close to each other when the door is closed. For this particular magnetic sensor, when the door is closed, the output will be short, and PB14 will be at the VCC voltage.

Door sensors have two types: NC (Normal close) and NO (Normal Open). The connection for both type sensors are the same. But the decoding for payload is reverse, user need to modify this in the IoT Server decoder.

When door sensor is shorted, there will extra power consumption in the circuit, the extra current is $3v3/R14 = 3v2/1Mohm = 0.3\mu A$ which can be ignored.



The above photos shows the two parts of the magnetic switch fitted to a door.

The software by default uses the falling edge on the signal line as an interrupt. We need to modify it to accept both the rising edge (0v --> VCC , door close) and the falling edge (VCC --> 0v , door open) as the interrupt.

The command is:

AT+INTMOD=1 // (more info about INMOD please refer [AT Command Manual](#)).

Below shows some screen captures in TTN:

In MOD=1, user can use the **Digital Input & Interrupt** byte to see the status for door open or close. The Decode is:

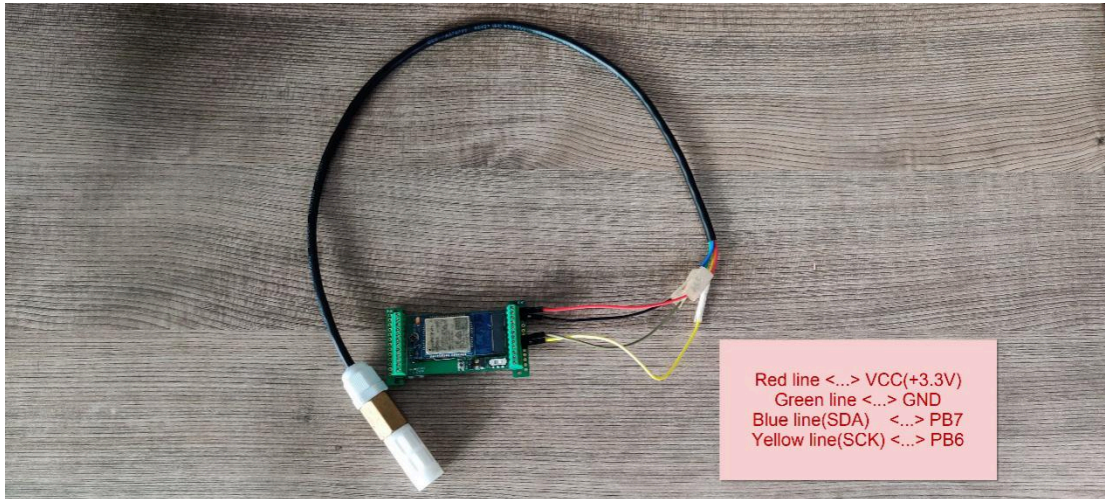
- When PB14 is high, the bit8 of this byte is 1.
- When PB14 is low, the bit8 of this byte is 0.

2.4.9 I2C Interface (SHT20)

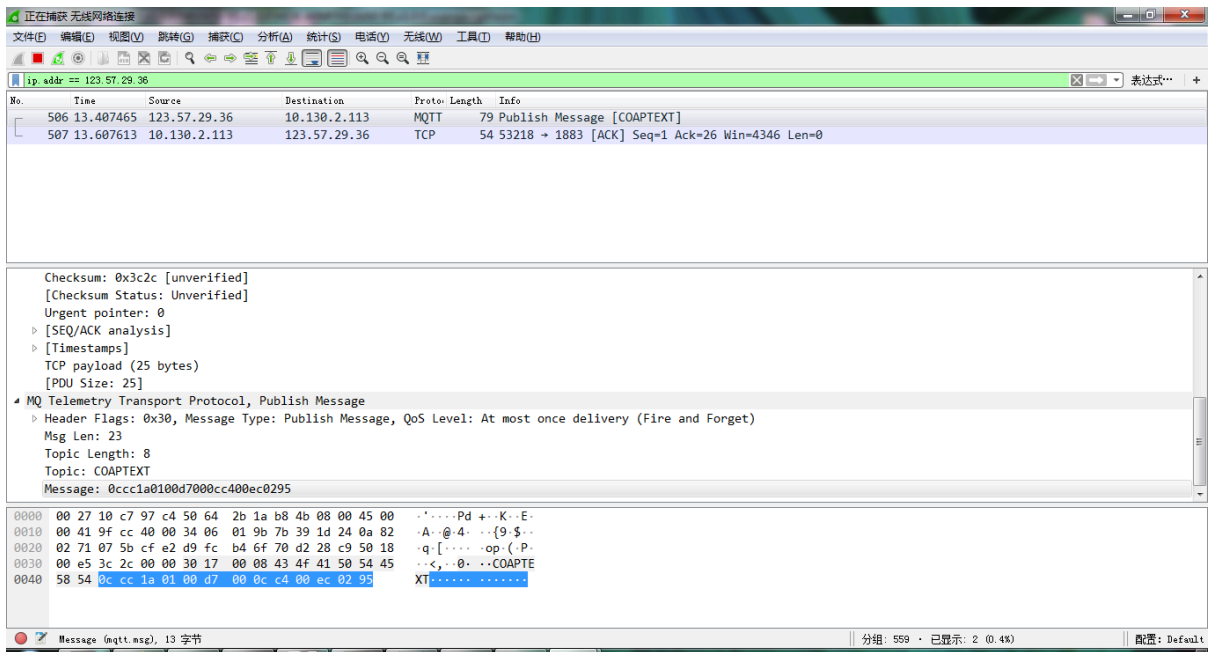
The **PB6(SDA)** and **PB7(SCK)** are I2C interface. User can use these pins to connect to an I2C device and get the sensor data.

There is an example to show how to use the I2C interface to connect to the SHT20 Temperature and Humidity Sensor. This is support in **AT+CFGMOD=1 (default value)**.

Hardware connection for SHT20 is as below:



The device will be able to get the I2C sensor data now and upload to IoT Server.



Convert the read byte to decimal and divide it by ten.

Example:

Temperature: Read:00ec (H) = 236(D) Value: 236 /10=23.6°C

Humidity: Read:0295(H)=661(D) Value: 661 / 10=66.1, So 66.1%

If you want to use other I2C device, please refer the SHT20 part source code as reference.

2.4.10 Distance Reading

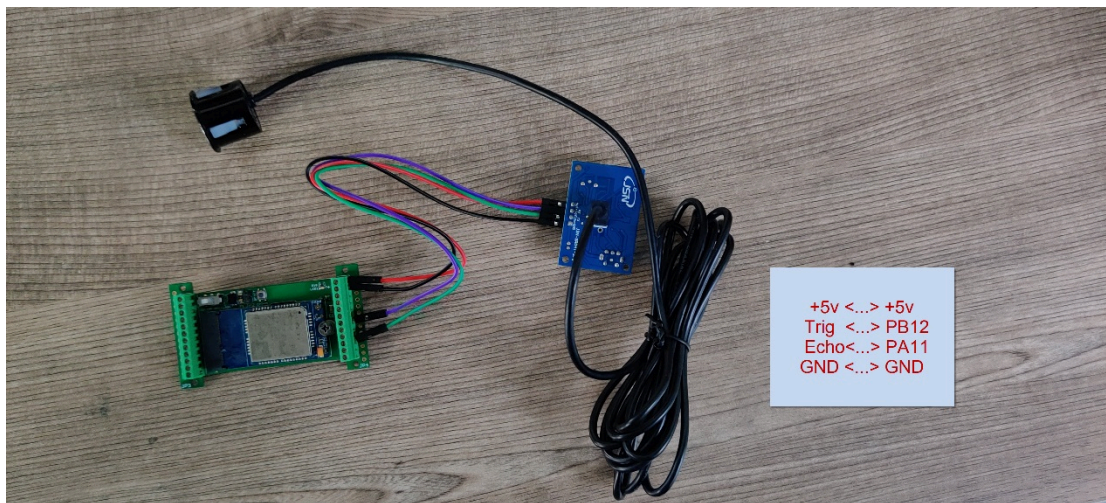
Refer [Ultrasonic Sensor section](#).

2.4.11 Ultrasonic Sensor

The NBSN95 firmware supports ultrasonic sensor (with AT+CFGMOD=2) such as SEN0208 from DF-Robot. This Fundamental Principles of this sensor can be found at this link: https://wiki.dfrobot.com/Weather_-_proof_Ultrasonic_Sensor_with_Separate_Probe_SKU_SEN0208

The NBSN95 detects the pulse width of the sensor and converts it to mm output. The accuracy will be within 1 centimeter. The usable range (the distance between the ultrasonic probe and the measured object) is between 24cm and 600cm.

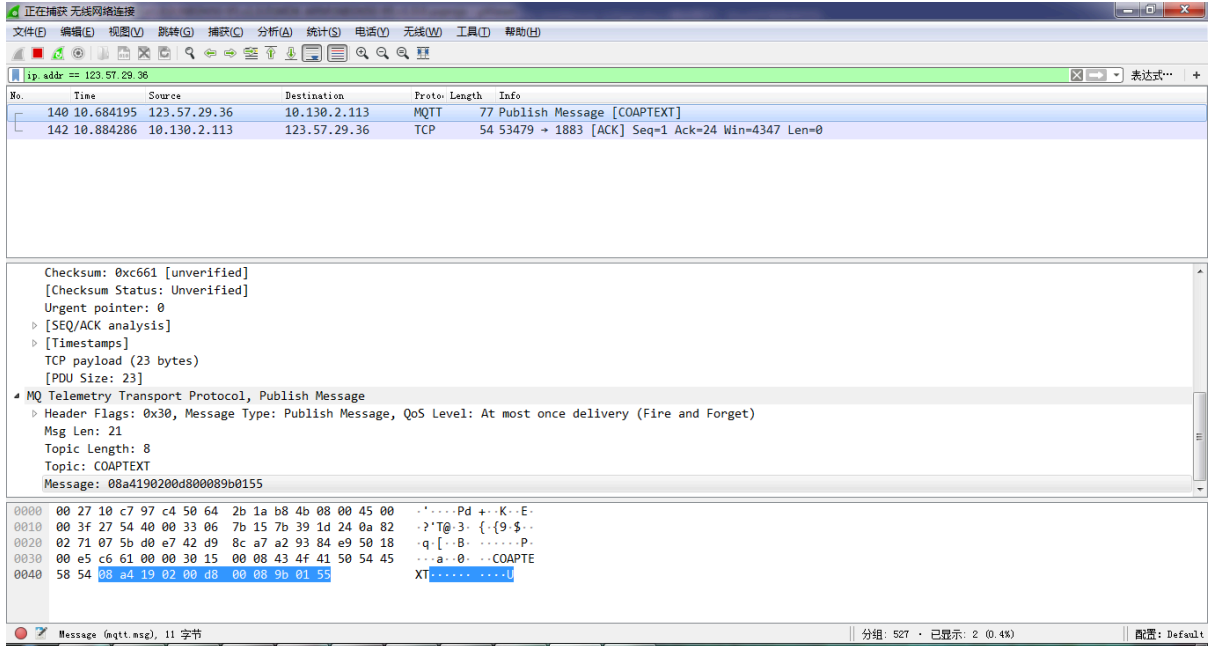
The picture below shows the connection:



Connect to the NBSN95 and run [AT+CFGMOD=2](#) to switch to ultrasonic mode (ULT).

Example:

Distance: Read:0155(Hex) = 3410(D) Value: 3410 mm=341.0 cm



You can see the serial output in ULT mode as below:

```
[20-03-28 09:59:49]*****Upload start:2*****
[20-03-28 09:59:49]remaining battery =2147 mv
[20-03-28 09:59:49]DS18B20(1) temp is 26.25 ° C
[20-03-28 09:59:50]adc_mV(1):2146.00

[20-03-28 09:59:50]Distance =338cm
[20-03-28 09:59:50]NB-IoT Signal Strength
[20-03-28 09:59:50]Protocol in Used: Coap
[20-03-28 09:59:50]Signal Strength:24

[20-03-28 09:59:50]Waiting to send get-data.
[20-03-28 09:59:52]Get-Data downlink successfully.
[20-03-28 09:59:52]Waiting to send put-data.
[20-03-28 09:59:55]Put-Data uploaded successfully.
[20-03-28 09:59:55]*****End of upload*****

[20-03-28 09:59:55]Enter low power mode
```

2.4.12 +5V Output

NBSN95 will enable +5V output before all sampling and disable the +5v after all sampling.

The 5V output time can be controlled by AT Command.

AT+5VT=1000

Means set 5V valid time to have 1000ms. So the real 5V output will actually have 1000ms + sampling time for other sensors.

2.4.13 Weigh Sensor HX711

NBSN95 supports Weigh Sensor HX711. See [this link](#) for instruction.

2.4.14 Timestamp

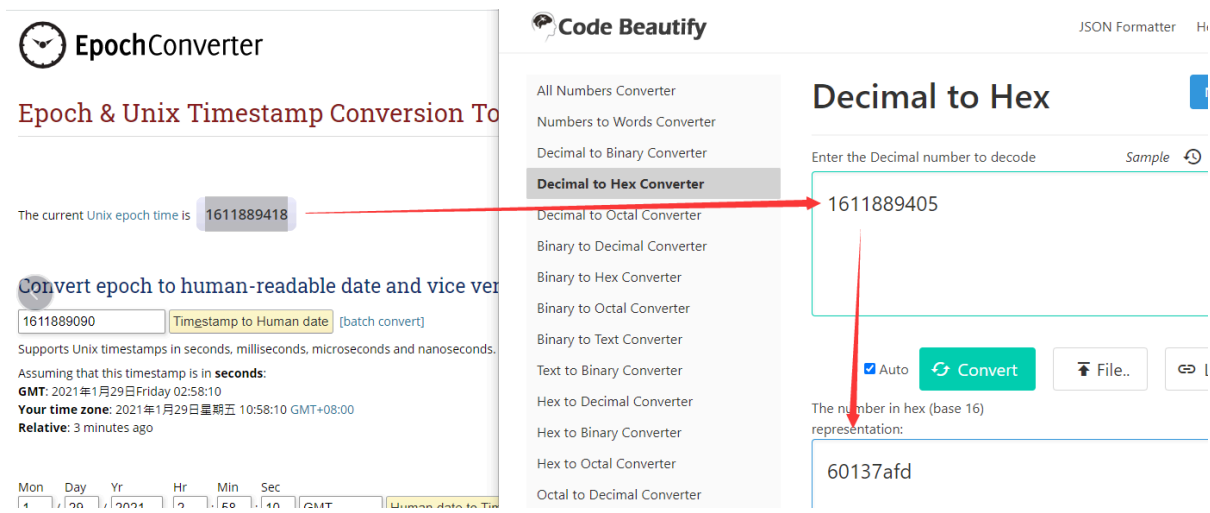
From the v1.2.0 version, the timestamp will be added after each upload link.

The time of the timestamp is based on the time in time zone 0.

Such as: 0x60da837e = 1624933246 = 2021-06-29 2:20:46+32(Beijing time)

User can get this time from link: <https://www.epochconverter.com/> :

Below is the converter example



2.5 Downlink Payload

By default, NBSN95 prints the downlink payload to console port.

Downlink Control Type	FPort	Type Code	Downlink payload size(bytes)
TDC (Transmit Time Interval)	Any	01	4
RESET	Any	04	2
INTMOD	Any	06	4

Examples:

- **Set TDC**

If the payload=0100003C, it means set the END Node's TDC to 0x00003C=60(S), while type code is 01.

Payload: 01 00 00 1E TDC=30S

Payload: 01 00 00 3C TDC=60S

- **Reset**

If payload = 0x04FF, it will reset the NBSN95

- **INTMOD**

Downlink Payload: 06000003, Set AT+INTMOD=3

2.6 Firmware Change Log

Download URL & Firmware Change log: <https://www.dropbox.com/sh/heyfnuod8qwwcm0/AACjmliw8Aeb9tYc74YoE9Qra?dl=0>

Upgrade Instruction: [Upgrade Firmware](#)

2.7 Use VDD or +5V to Power External Sensor

User can use VDD or +5V to power external sensor.

Note:

1. VDD is 2.5~3.3v from the battery + diode, the VDD is always on, so when use VDD to power external sensor, make sure the sensor has a low power consumption in sleep mode (less 50 uA) to get a long battery life.

2. +5V output is only ON when sampling. And MCU will turn off it after sampling. So if sensor can support 5v, +5V out is the best choice. [See here for more info.](#)

Note: Always test the actually current pass by the JP2 jumper when connect to a new type of sensor.

2.8 Battery Analysis

2.8.1 Battery Type

The NBSN95 battery is a combination of an 8500mAh Li/SOCI2 Battery and a Super Capacitor. The battery is none-rechargeable battery type with a low discharge rate (<2% per year). This type of battery is commonly used in IoT devices such as water meter.

The battery is designed to last for several years depends on the actually use environment and update interval.

The battery related documents as below:

- [Battery Dimension](#)
- [Lithium-Thionyl Chloride Battery datasheet](#)
- [Lithium-ion Battery-Capacitor datasheet](#)



2.8.2 Power consumption Analyze

The file **DRAGINO_NBSN95-Power-Analyzer.pdf** from https://www.dropbox.com/sh/zwex6i331j5oeq2/AACIMf9f_v2qsJ39CuMQ5Py_a?dl=0 describes a detail measurement to analyze the power consumption in different case. User can use it for design guideline for their project.

2.8.3 Battery Note

The Li-SiCO battery is designed for small current / long period application. It is not good to use a high current, short period transmit method. The recommended minimum period for use of this battery is 5 minutes. If you use a shorter period time to transmit LoRa, then the battery life may be decreased.

2.8.4 Replace the battery

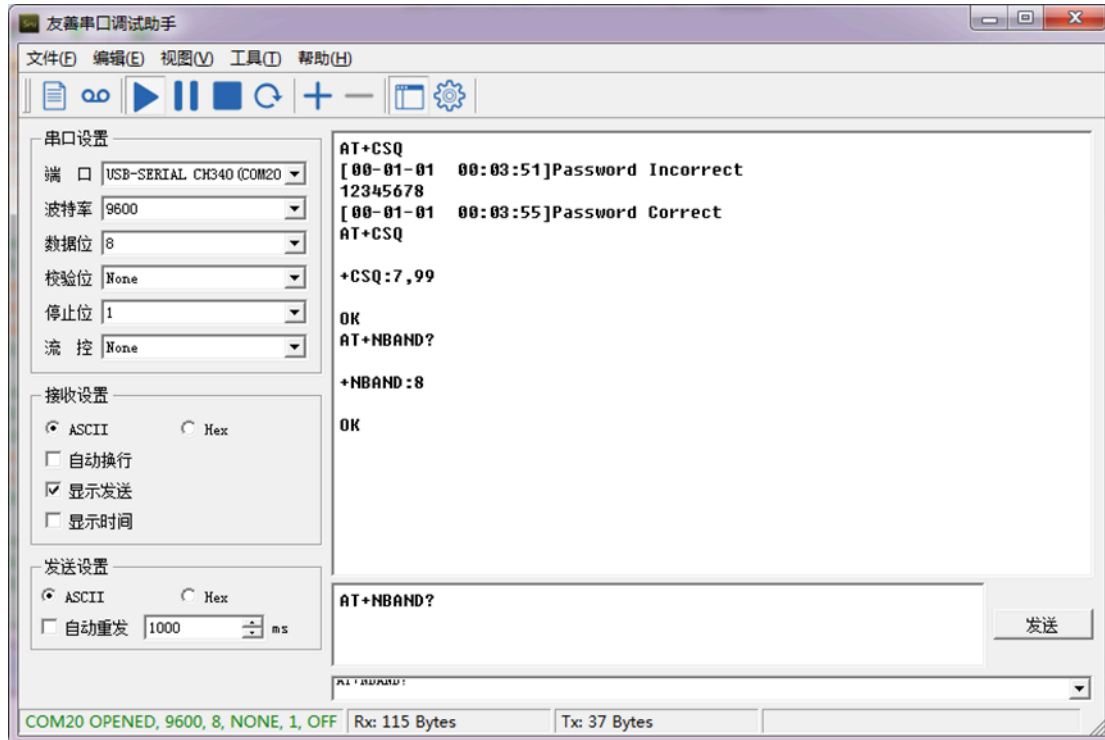
You can change the battery in the NBSN95. The type of battery is not limited as long as the output is between 3v to 3.6v. On the main board, there is a diode (D1) between the battery and the main circuit. If you need to use a battery with less than 3.3v, please remove the D1 and shortcut the two pads of it so there won't be voltage drop between battery and main board.

The default battery pack of NBSN95 includes a ER26500 plus super capacitor. If user can't find this pack locally, they can find ER26500 or equivalence without the SPC1520 capacitor, which will also work in most case. The SPC can enlarge the battery life for high frequency use (update period below 5 minutes).

3. Access NB-IoT Module

Users can directly access the AT command set of the NB-IoT module.

The AT Command set can refer the BC95-G AT Command: https://www.dragino.com/downloads/index.php?dir=datasheet/other_vendors/BC95-G/



4. Using the AT Commands

4.1 Access AT Commands

See this link for detail: <https://www.dropbox.com/sh/jao1xt9kw5r3yq4/AAAMpJkZzExF2JLbRWxGoQ9Na?dl=0>

AT+<CMD>? : Help on <CMD>
 AT+<CMD> : Run <CMD>
 AT+<CMD>=<value> : Set the value
 AT+<CMD>=? : Get the value

General Commands

AT : Attention
 AT? : Short Help
 ATZ : MCU Reset
 AT+TDC : Application Data Transmission Interval
 AT+CFG : Print all configurations
 AT+CFGMOD : Working mode selection
 AT+INTMOD : Set the trigger interrupt mode
 AT+5VT : Set extend the time of 5V power
 AT+PRO : Choose agreement

AT+WEIGRE : Get weight or set weight to 0
AT+WEIGAP : Get or Set the GapValue of weight
AT+RXDL : Extend the sending and receiving time
AT+CNTFAC : Get or set counting parameters
AT+SERVADDR : Server Address

COAP Management

AT+URI : Resource parameters

UDP Management

AT+CFM : Upload confirmation mode (only valid for UDP)

MQTT Management

AT+CLIENT : Get or Set MQTT client
AT+UNAME : Get or Set MQTT Username
AT+PWD : Get or Set MQTT password
AT+PUBTOPIC : Get or Set MQTT publish topic
AT+SUBTOPIC : Get or Set MQTT subscription topic

Information

AT+LDATA : Get the last upload data
AT+CDP : Read or Clear cached data
AT+FDR : Factory Data Reset
AT+PASSWORD : Serial Access Password

5. Developer Guide

5.1 Get and compile Software

NBSN95 is an open-source project, developer can compile their firmware for customized applications. User can get the source code from:

- Software Source Code: <https://github.com/dragino/NBSN95>
- Compile instruction: <http://wiki.dragino.com/xwiki/bin/view/Main/Firmware%20Compile%20Instruction%20--%20STM32/>
- Upgrade firmware instruction

See FAQ of this file.

The project file is in: **NBSN95-95.v1.0.0\NBSN95-95.v1.0.0\MDK-ARM\NBSN95-95.v1.0.0.uvprojx**

5.2 Get hardware source

NBSN95 are the compose of two PCB modules:

1. Mother board: [LoRa ST Sensor node mother board](#).
2. NB-IoT Module: <https://github.com/dragino/NB-IoT/tree/master/NB%20ST/BC95>

6. FAQ

6.1 How to Upgrade Firmware

User can upgrade the firmware for 1) bug fix, 2) new feature release.

Please see this link for how to upgrade: <http://wiki.dragino.com/xwiki/bin/view/Main/Firmware%20Upgrade%20Instruction%20for%20STM32%20base%20products/#H2.HardwareUpgradeMethodSupportList>

Notice, NBSN95 and LSN50v2 share the same mother board. They use the same connection and method to update.

7. Trouble Shooting

7.1 Connection problem when uploading firmware

Please see: <http://wiki.dragino.com/xwiki/bin/view/Main/Firmware%20Upgrade%20Instruction%20for%20STM32%20base%20products/#H3.3Troubleshooting>

7.2 AT Command input doesn't work

In the case if user can see the console output but can't type input to the device. Please check if you already include the **ENTER** while sending out the command. Some serial tool doesn't send **ENTER** while press the send key, user need to add ENTER in their string.

8. Order Info

Part Number:

NBSN95-YY (Base on BC95-G)

or

NBSN95A-YY(Base on BC35-G)

YY:

- **12:** With M12 waterproof cable hole
- **16:** With M16 waterproof cable hole
- **NH:** No Hole

9. Packing Info

Package Includes:

- NBSN95 NB-IoT Soil Moisture & EC Sensor x 1
- External antenna x 1

Dimension and weight:

- Device Size: 13.0 x 5 x 4.5 cm
- Device Weight: 150g
- Package Size / pcs : 14.0 x 8x 5 cm
- Weight / pcs : 180g

10. Support

- Support is provided Monday to Friday, from 09:00 to 18:00 GMT+8. Due to different timezones we cannot offer live support. However, your questions will be answered as soon as possible in the before-mentioned schedule.
- Provide as much information as possible regarding your enquiry (product models, accurately describe your problem and steps to replicate it etc) and send a mail to support@dragino.com