Pico-LoRa-SX1262

The Pico-LoRa-SX1262-868M is a LoRa node expansion module designed for Raspberry Pi Pico based on SX1262, which supports LoRaWAN protocol and EU868 band. It is allowed to connect the TTN, TTS servers through a LoRa gateway to use LoRa Cloud service fast and easily.

Features

- Onboard Raspberry Pi Pico interface for Raspberry Pi Pico series boards.
- Support LoRa and (G) FSK modulation, suitable for EU433, EU868, AS923, US915, etc.
- Onboard PH1.25 battery holder and charging IC, which can be connected to a lithium battery for charging and discharging.
- On-board 2 LED indicators for module operating status.
- Provide a complete supporting information manual (C sample program and user manual, etc.)

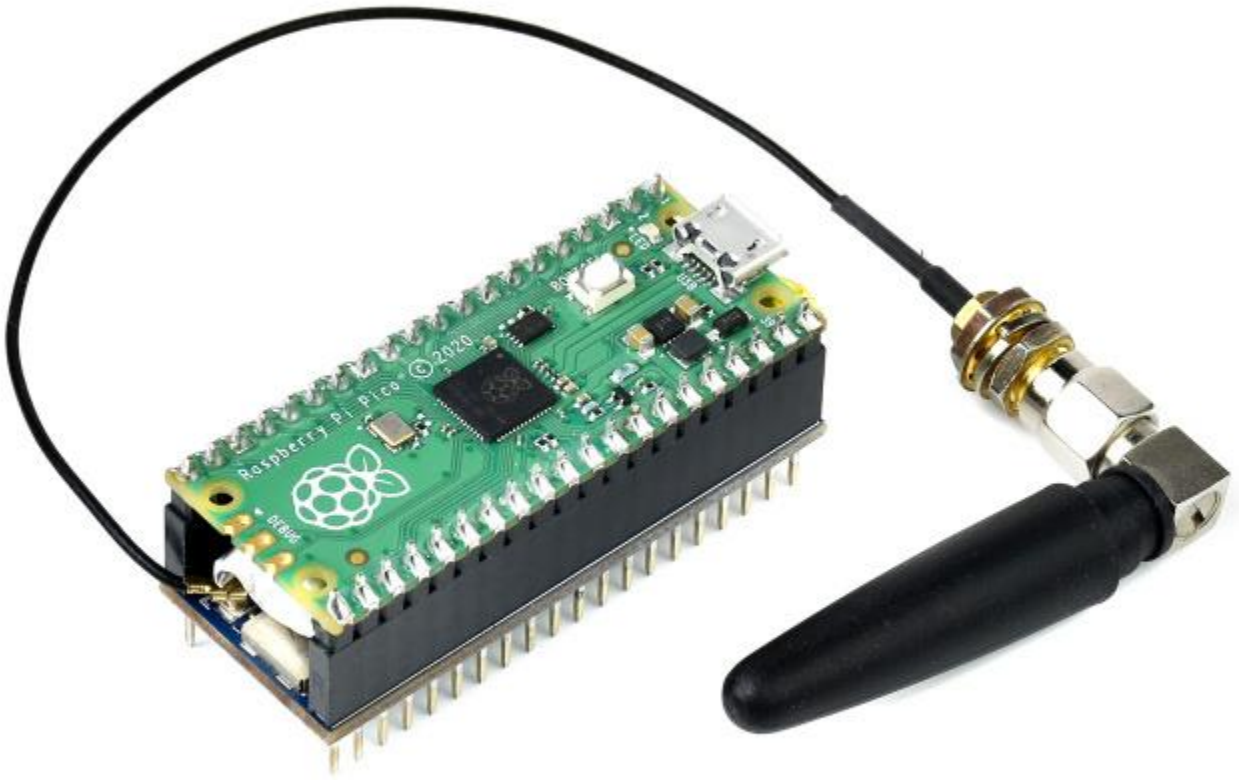
Specifications

Electrical	Parameters
RF chip	SX1262
Working Band	EU433 EU868 (863~870MHz) AS923 (915~928MHz) US915(902~928MHz)
Signal modulation	LoRa/(G)FSK
Transmit power	MAX@22dBm(adjustable)@3.3V
Working Voltage	5V
Module consumption	Transmit current: 45mA@14dBm Receive current: 5.3mA@125KHz
Communication interface	SPI
Working temperature	-40 ~ 85℃

Size	21.00 × 52.00mm

Hardware Description

- Pico-LoRa-SX1262-868M x 1 (included)
- Raspberry Pi Pico x 1 (not included)
- Micro USB cable x 1 (not included)



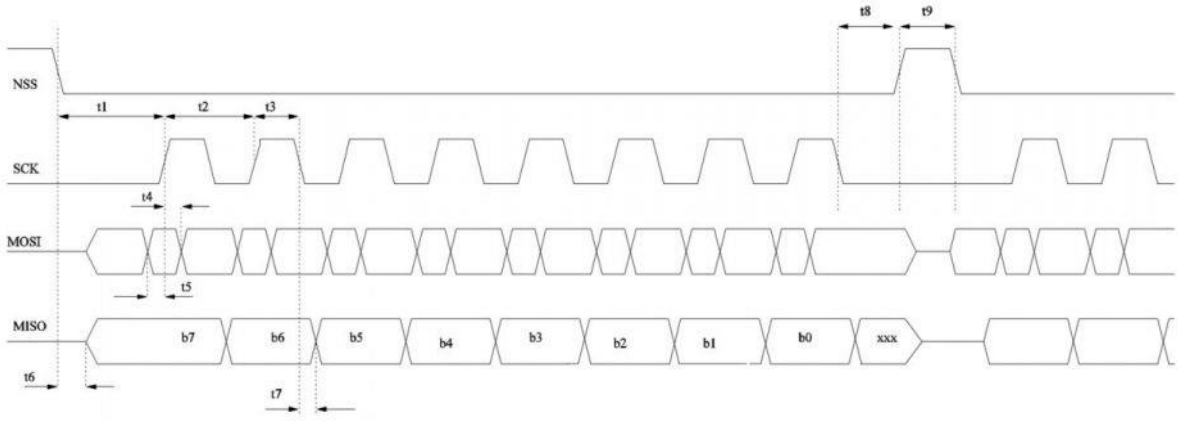
LoRa	Pico	Features
VCC	VSYS	Power Input
GND	GND	Ground
BAT_AD	GP26	Onboard battery ADC pin
LoRa_DIO1	GP20	SX1262 interrupt output pin or special function pin

LoRa_RESET	GP15	Reset pin (low active)
LoRa_MISO	GP12	SPI communication MISO pin, data output from the device
LoRa_MOSI	GP11	SPI communication MOSI pin, data input from the device
LoRa_CLK	GP10	SPI communication SCK pin, slave device clock input
LoRa_CS	GP3	SPI chip selection pin (low active)
LoRa_BUSY	GP2	Busy status output pin

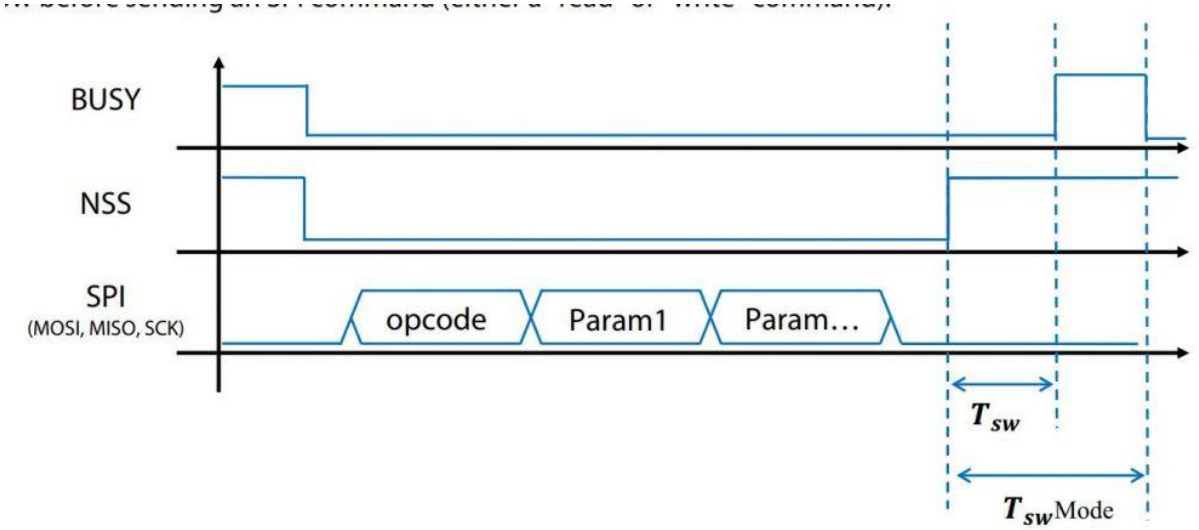
Pinout

Communication Timing

- The Raspberry Pi Pico uses the SPI bus to read and write the 36 registers of the SX1262 to complete LoRa wireless data transmission. The SPI bus frequency should be less than 18MHz. For specific SPI Timing Requires, see Chapter 8 Digital Interface and Control of the datasheet.



- The SPI bus writing a command to the LoRa chip will trigger the switching of the internal state machine mode. It should be noted that the low level of the BUSY pin indicates that the internal idle is allowed to receive commands, and the high level indicates that the internal occupied cannot accept the SPI command. The SPI bus needs to wait for the BUSY pin Pull low again to continue reading and writing operations, TswMode is up to 3.5ms.



Dimension

LoRa & LoRaWAN

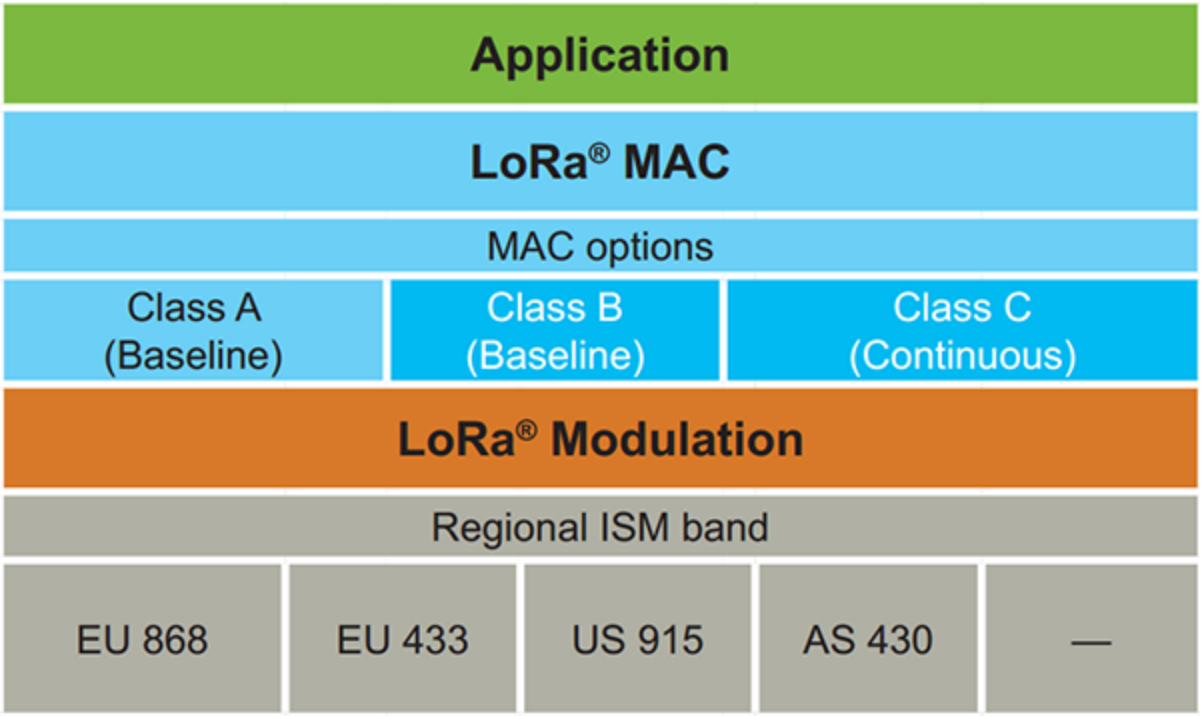
- What is LoRa ?

[Semtech]'s LoRa is a long-distance, low-power wireless platform for the Internet of Things (IoT), which generally refers to radio frequency chips using LoRa technology. The main features are as follows:

- The spread spectrum modulation technology used by LoRa (abbreviation of long range) is derived from chirp spread spectrum (CSS) technology, which is one of long-distance wireless transmission technology and LPWAN communication technology. At present, LoRa mainly operates in the ISM frequency band, mainly including 433 , 868, 915 MHz, etc.
- LoRa technology integrates technologies such as digital spread spectrum, digital signal processing and forward error correction coding, which greatly improves the performance of long-distance communication. LoRa's link budget is better than any other standardized communication technology. The main factors that determine distance in a given environment
- LoRa RF chips mainly include SX127X series, SX126X series, SX130X series, of which SX127X, SX126X series are used for LoRa nodes, and SX130X is used for LoRa gateways. For details, please refer to [Semtech]'s product list.

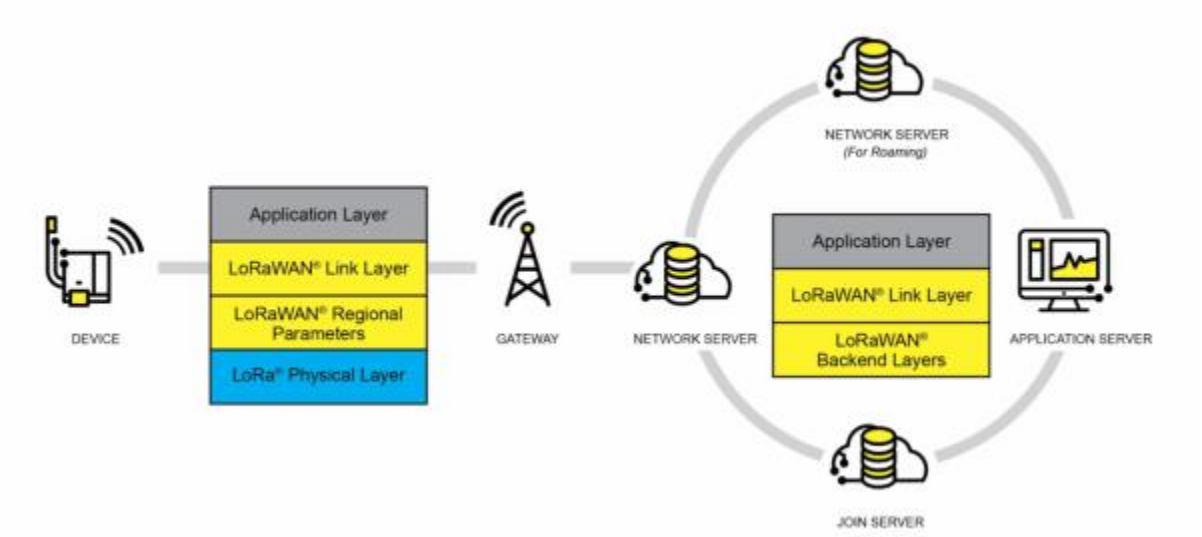
What is LoRaWAN ?

- [LoRaWAN] is a low power wide area network open protocol based on LoRa radio modulation technology. Aims to wirelessly connect battery-powered "things" to the Internet in a regional, national or global network and targets key Internet of Things (IoT) requirements such as bi-directional communications, end-to-end security, mobility and localized services. Node wireless connection to the Internet has access authentication, which is equivalent to the establishment of encrypted communication channel between node and server. Access details refer to [document] and [source code]. LoRaWAN protocol level is shown in the figure below.
- The Class A/B/C node devices in the MAC layer basically cover all the application scenarios of the Internet of Things. The differences among the three nodes lie in the different time slots for receiving and receiving nodes
- EU868 and AS430 in the Modulation layer show that frequency band parameters are different in different countries. Please click the reference [link] for regional parameters



To achieve LoRaWAN network coverage in cities or other areas, it needs to be composed of four parts: node (LoRa node radio frequency chip), gateway (or base station, LoRa gateway radio frequency chip), server and cloud, as shown in the following figure

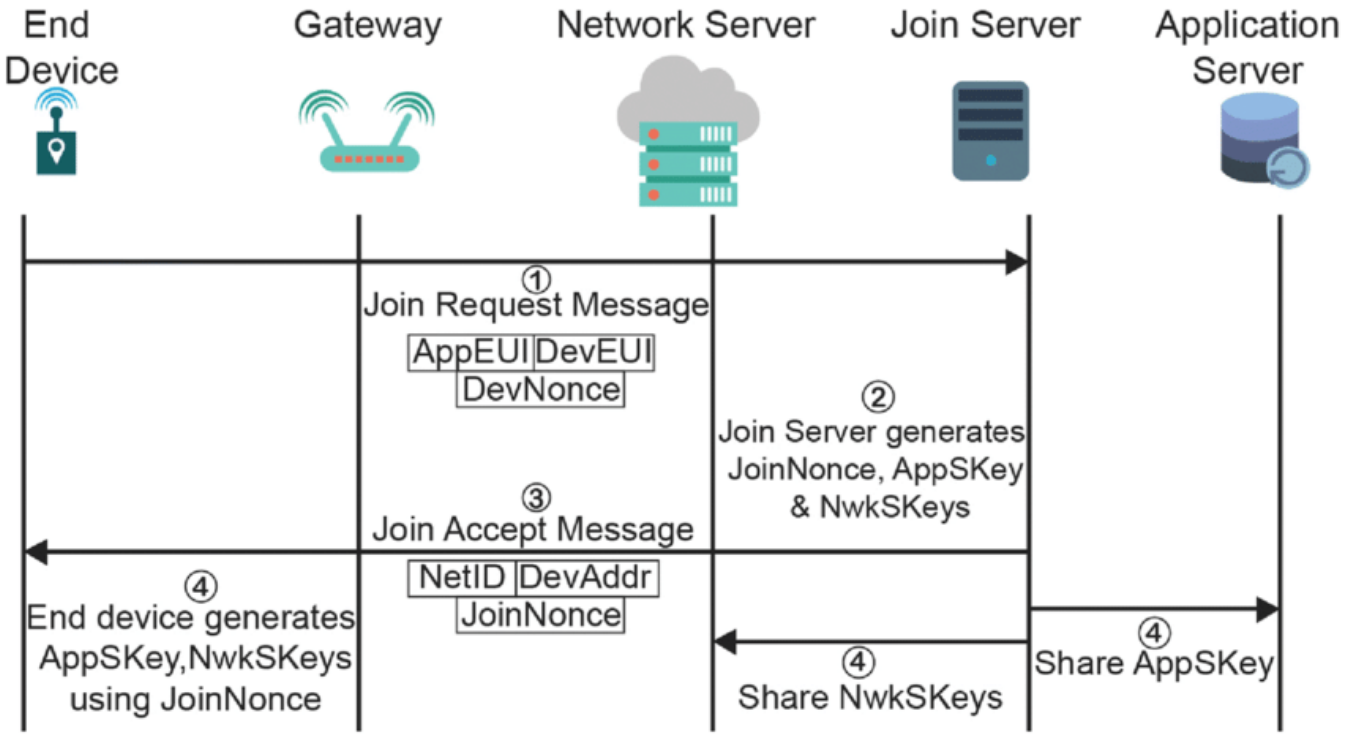
- The DEVICE (node device) needs to initiate a network access request packet to the GATEWAY (gateway) and then to the server. After the authentication is passed, it can send and receive application data with the server normally.
- GATEWAY (gateway) can communicate with the server through a wired network, 3/4/5G wireless network
- The main operators on the server side are [TTN], etc. For building cloud services by yourself, please refer to [lorawan-stack], [chirpstack]



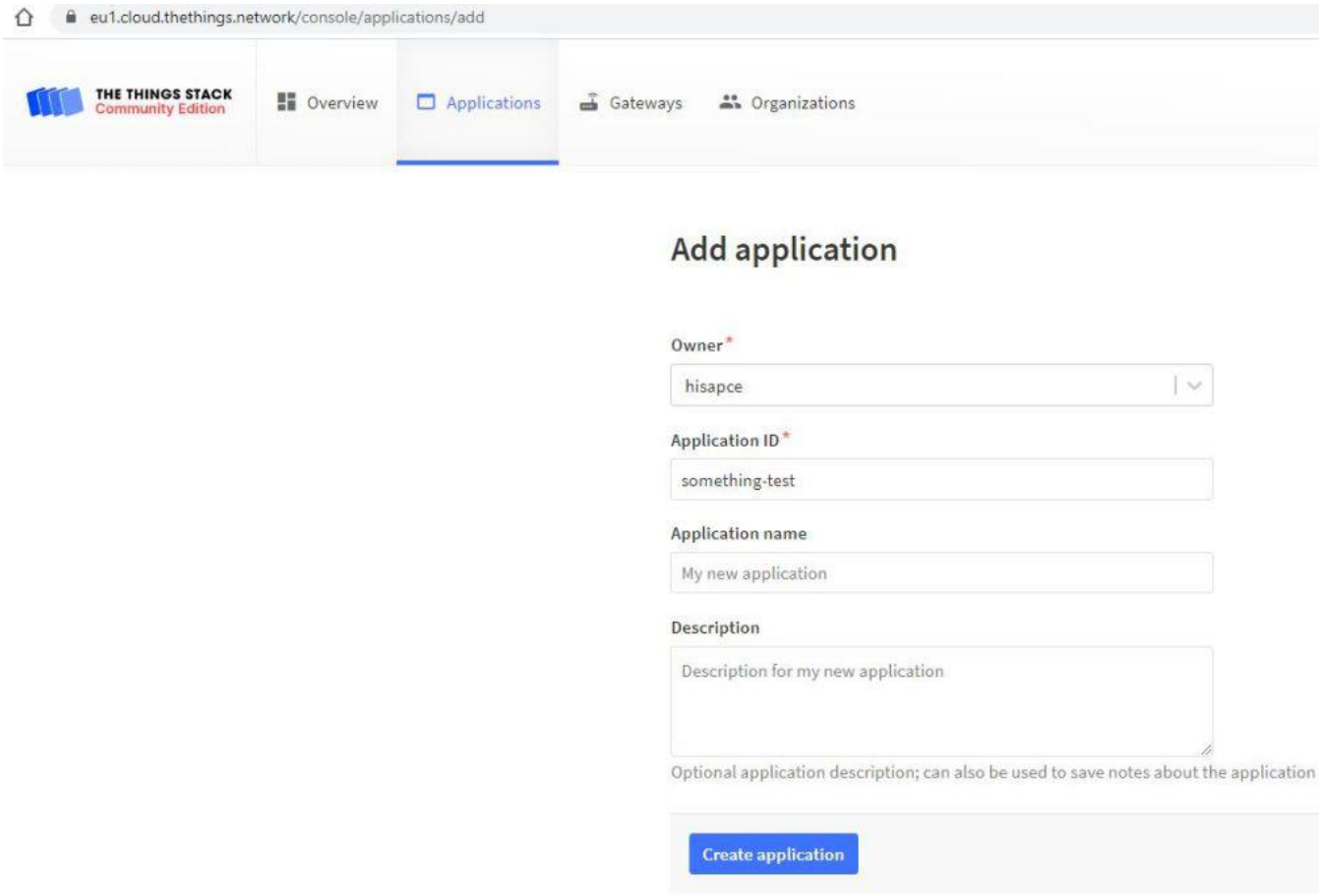
- There are two methods for Raspberry Pico and Pico-LoRa-SX1262 to connect to the network via LoRaWAN: OTAA (Over-The-Air-Activation) and ABP (Activation By Personalization). Here we use OTAA, as shown below. Also you can refer to link 1, link 2 and source code.
- Step 1: Please send the "Join-Request" message to the network to be added, and note that the adding process is always initiated by the end device. The join-Request message can be transmitted at any rate and in one of the region-specific join channels.For example: in Europe, the end device can send the join-Request message at 868.10 MHz, 868.30MHz or 838.50MHz. Also, the message can send to the network server by one or more gateway. In addition, you need to pay attention to choosing the applicable frequency band according to the local radio management regulations. You can refer to link and LoRa Alliance for the frequency distribution table. The Join-Request message is combined by the following field, AppEUI, and DevEUI are generated by the server end.

    - AppEUI: A 64-bit globally unique application identifier in the IEEE EUI64 address space that uniquely identifies an entity capable of processing Join-Request frames.
    - DevEUI: A 64-bit globally unique device identifier that uniquely identifies an end device in the IEEE EUI64 address space.
    - DevNonce: A unique random 2-byte value generated by the end device. The web server uses each end device's DevNonce to keep track of their join requests. If the end device sends a join request with the previously used DevNonce (this case is called a replay attack), the network server will reject the join request and will not allow the end device to register with the network.
- Step 2: The web server handles the Join-Request-Message. If the end device is allowed to join the network, the web server will generate two session keys (NwkSKey and AppSKey) and the Join-accept message. The Join-accept message itself is then encrypted using AppKey. The web server uses AES decryption operation in ECB mode to encrypt Join-accept messages.
- Step 3: The network server sends the encrypted join to accept the message back to the end device as a normal downlink.

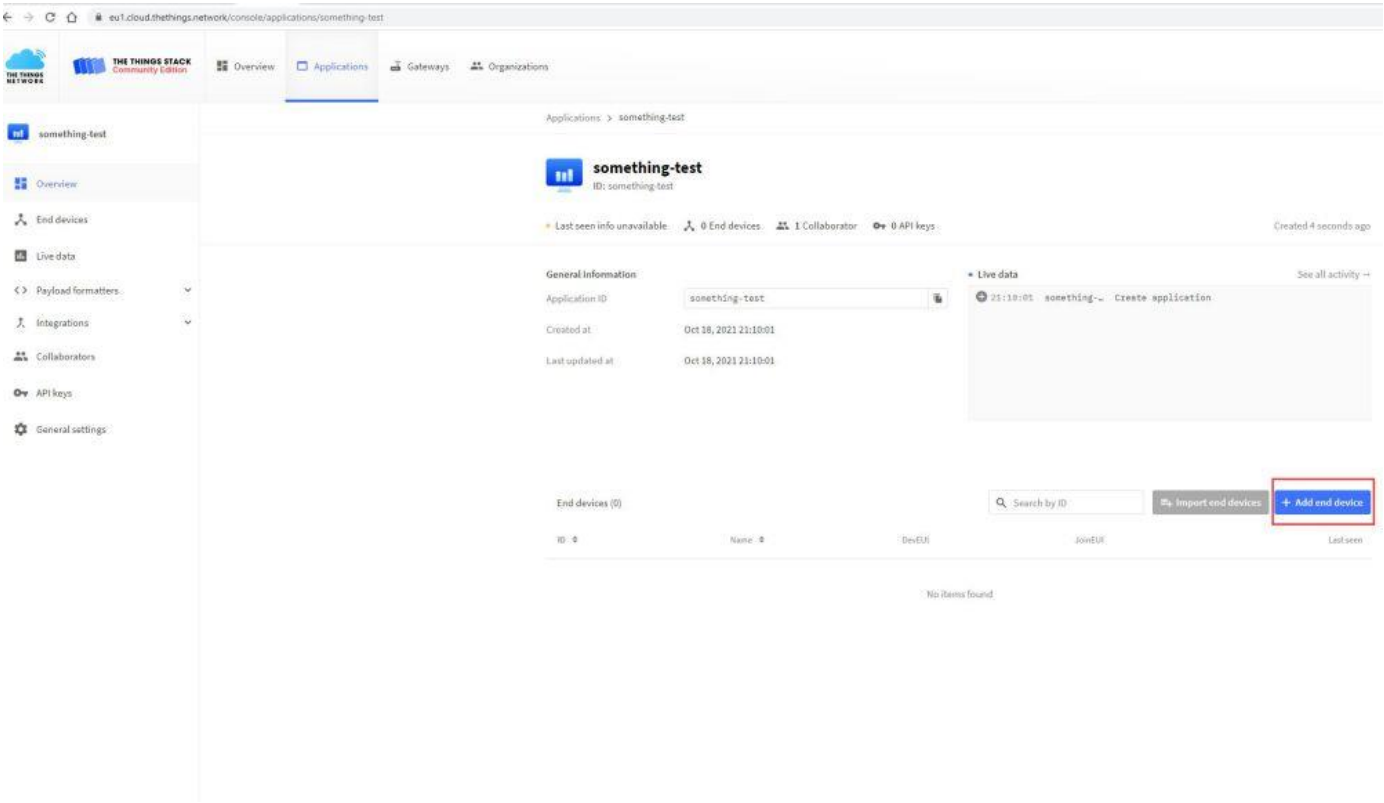
- Step 4: The end device uses AES to decrypt Join-Accept. And uses AppKey and AppNonce to generate two session keys NwkSKey and AppSKey for subsequent communication with the Networking server. Network Server also saves kSKey, Join server distributes AppSKey to Application Server.



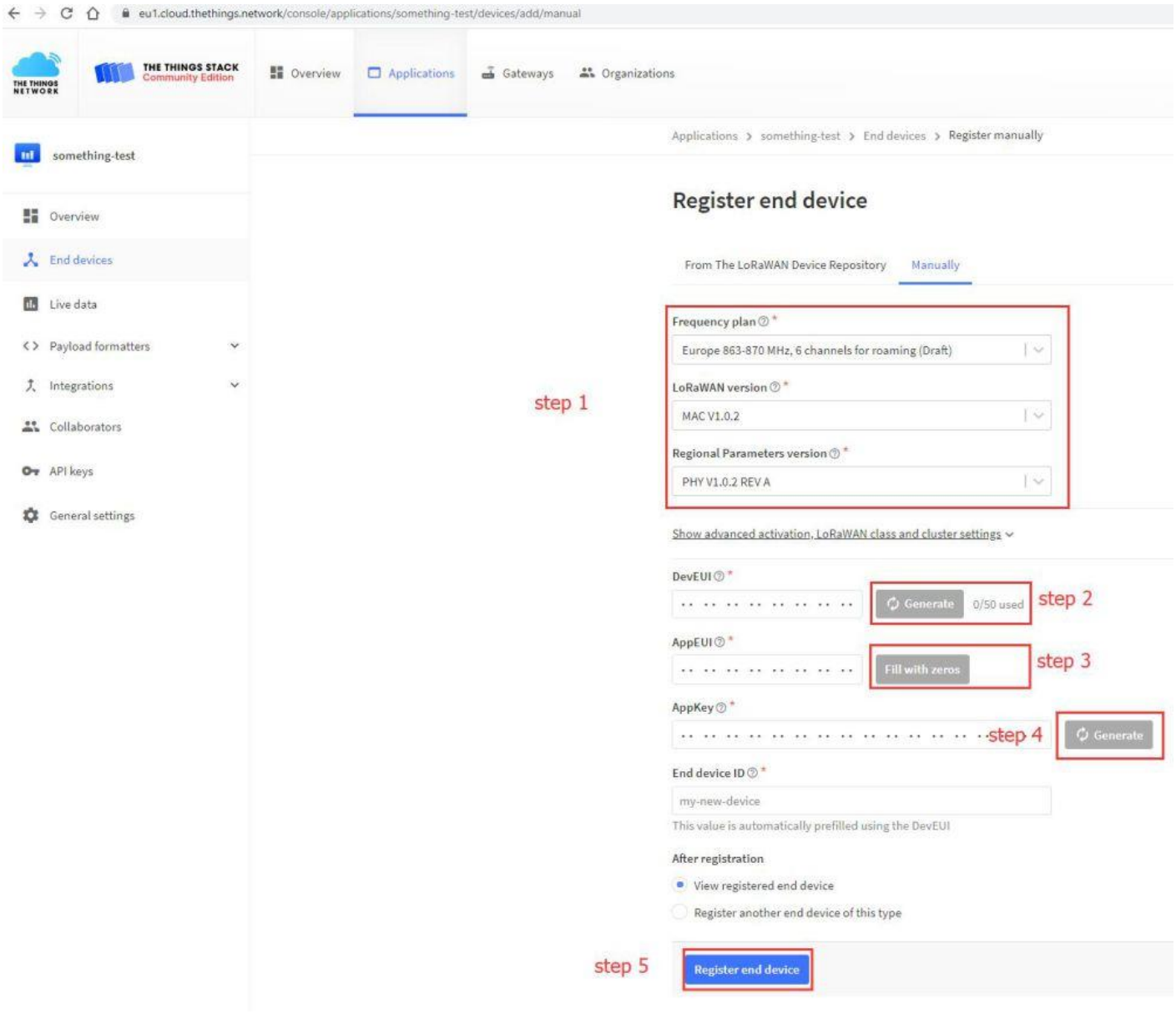
- The DevEUI and AppEUI parameters used as terminal devices to access the Internet need to be registered and generated by the server. The specific process is as follows:
    - Register an account in TTS and log in. Create an Application.



- Add an End Device



- Configure the End Device as in the picture. Please save the DevEUI and the AppKey for further use.

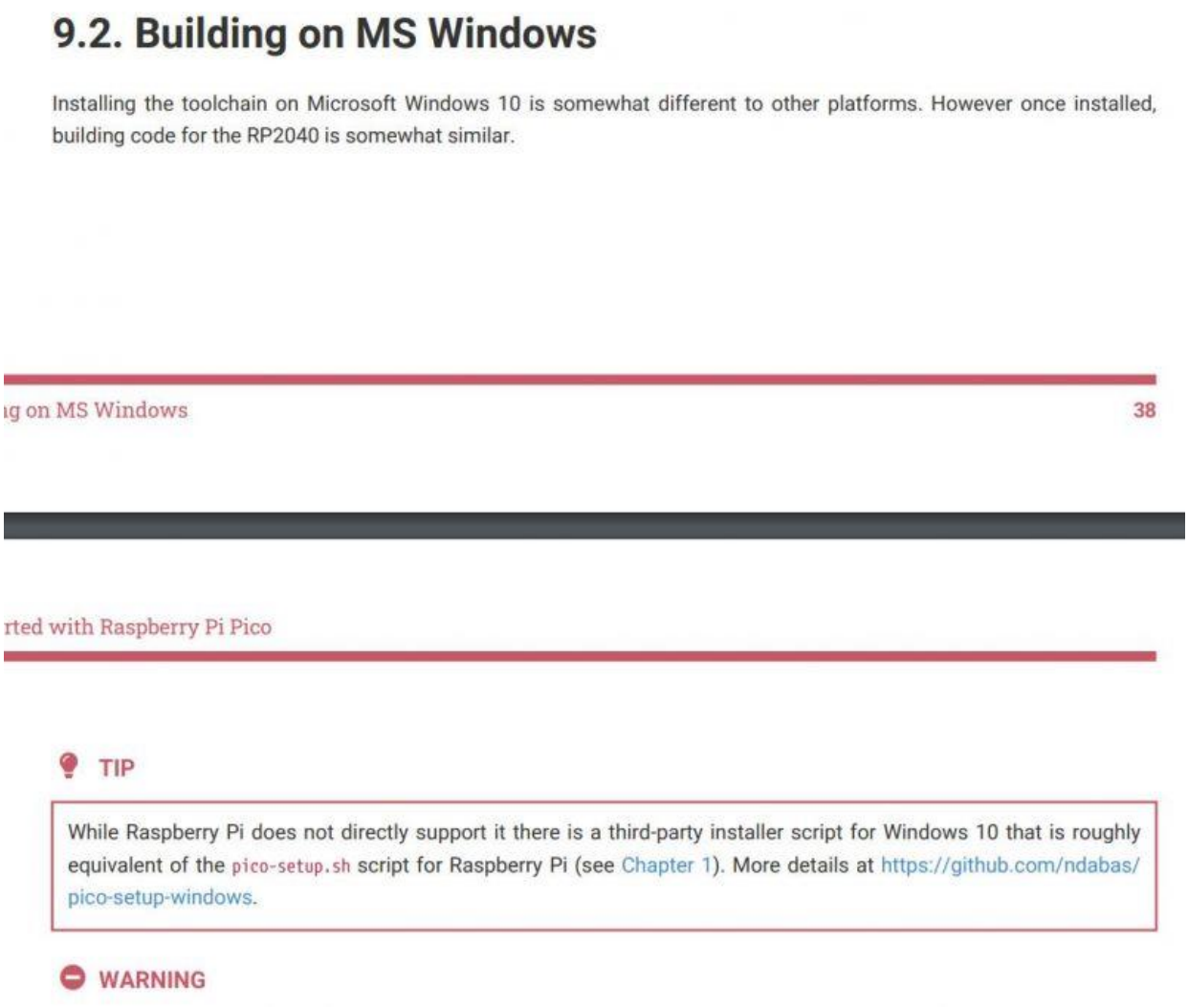


- Application

LoRa devices and networks such as LoRaWAN enable smart IoT applications to help solve the planet's formidable challenges in energy management, natural resource reduction, pollution control, infrastructure efficiency, disaster prevention, and more. Semtech's LoRa devices have achieved hundreds of successful use cases in smart cities, homes and buildings, communities, metrology, supply chain and logistics, agriculture, and more. LoRa networks have covered hundreds of millions of devices in more than 100 countries and are committed to a smarter planet.

Setup Environment

- This tutorial uses VScode (Cmake) to develop in the Windows 10 compilation environment, click to download the relevant IDE and install it and open it.
- Please refer to The C/C++ SDK, 9.2. Building on MS Windows in GET-START document for the compilation environment of VScode(Cmake).



9.2. Building on MS Windows

Installing the toolchain on Microsoft Windows 10 is somewhat different to other platforms. However once installed, building code for the RP2040 is somewhat similar.

ıg on MS Windows 38

rted with Raspberry Pi Pico

TIP

While Raspberry Pi does not directly support it there is a third-party installer script for Windows 10 that is roughly equivalent of the `pico-setup.sh` script for Raspberry Pi (see Chapter 1). More details at https://github.com/ndabas/pico-setup-windows.

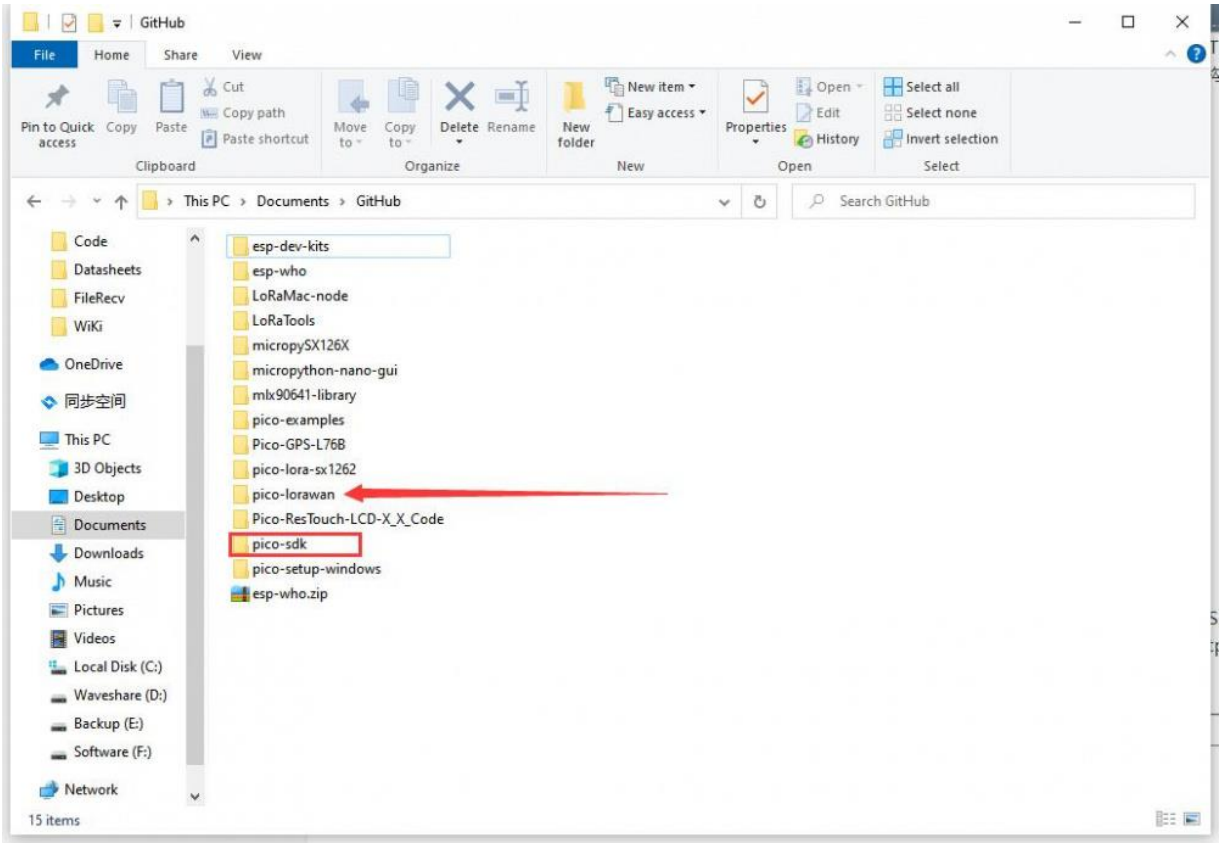
WARNING

Demo Download

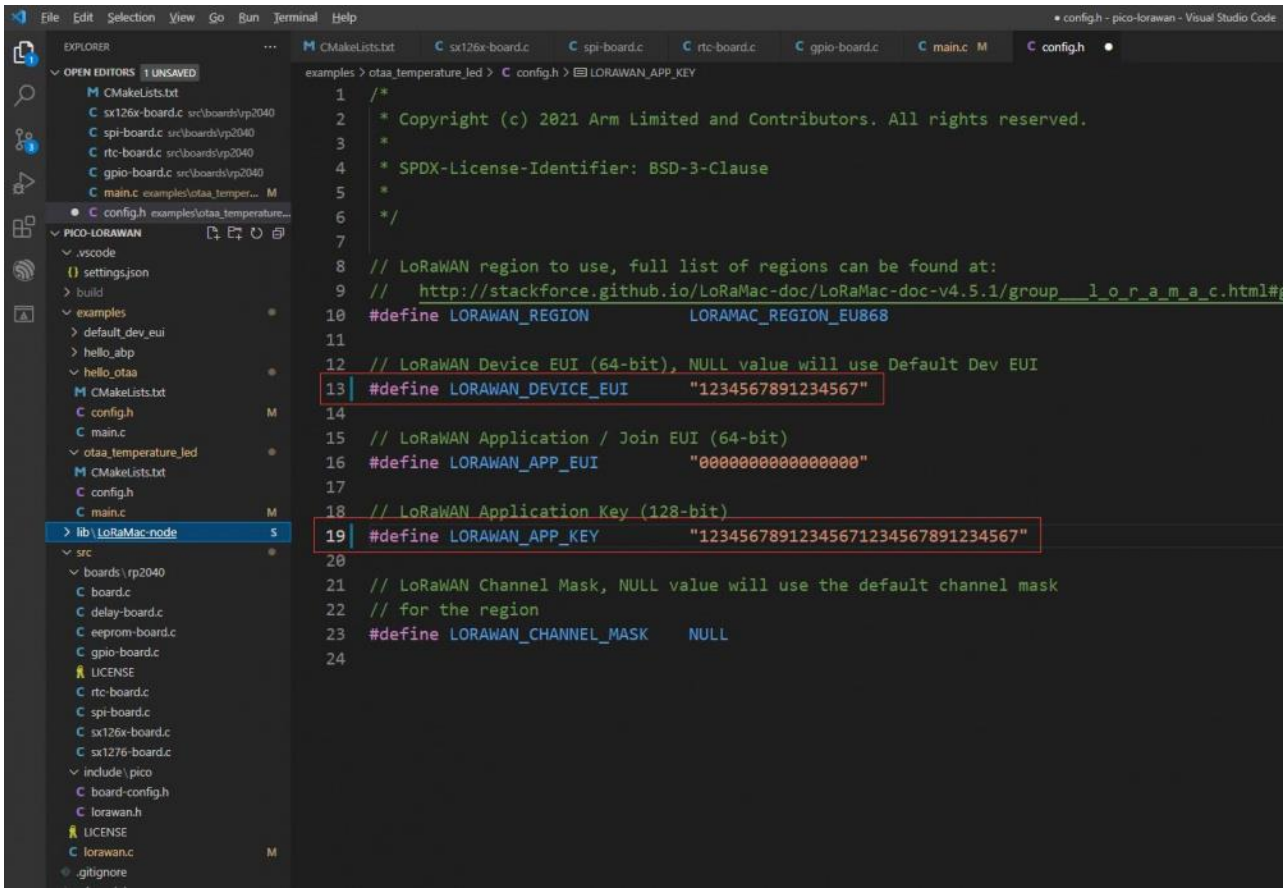
1. Please click to download link.
2. Download from github.

Run the Demo

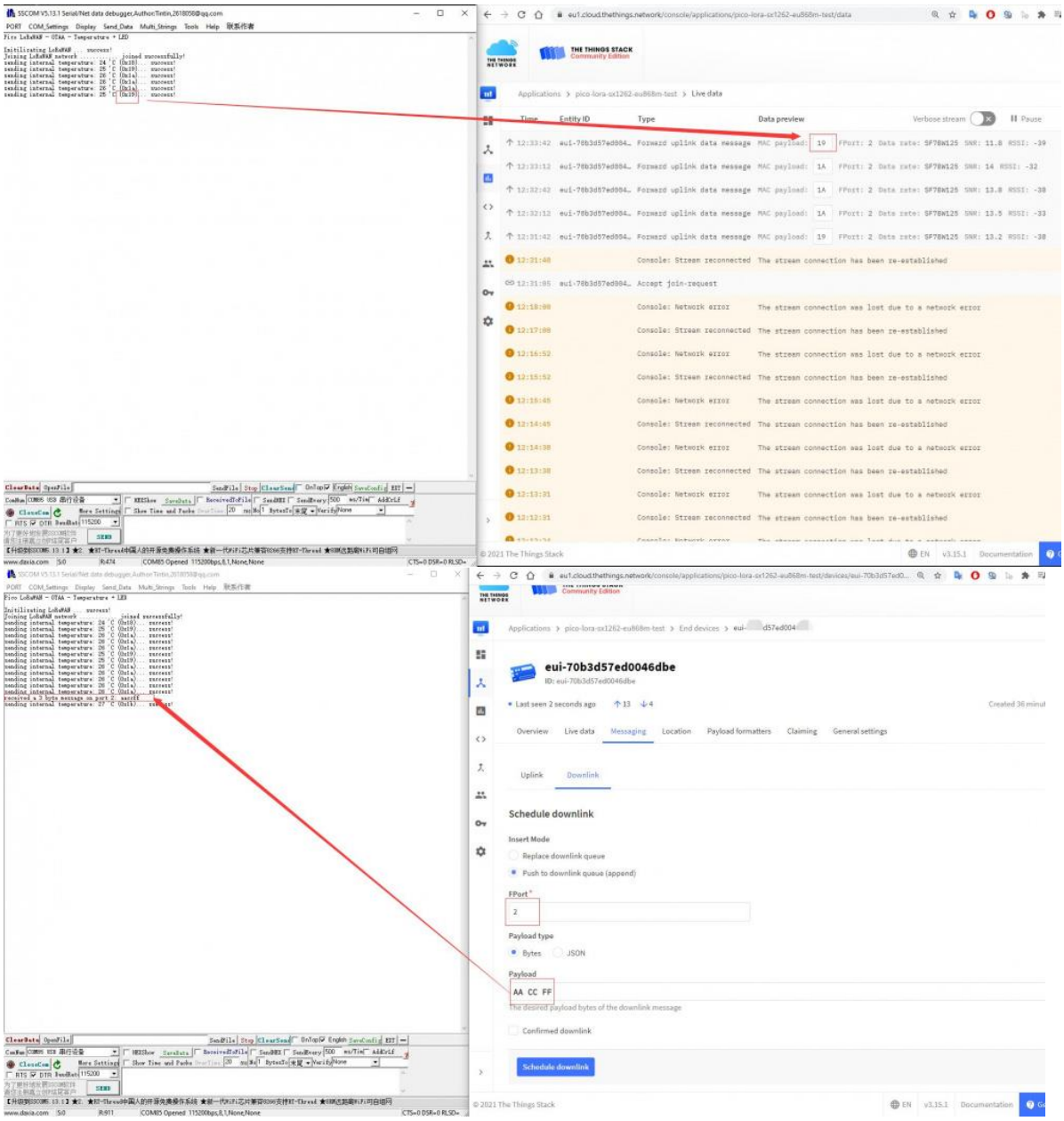
- Unzip the program or use git to download the program to the same level directory as pico-sdk. For the compilation environment installation, refer to the Pico environment setting chapter.



- Open VScode, choose to open the pico-lorawan folder in VScode, and fill in the DevEUI and AppKey values just saved in the example\otaa_temperature_led\config.h document.



- If the environment settings are correct, click the Build button of VScode to wait for the compilation to end, download the compiled file to the RaspberryPi Pico that enters the Boot mode, and open the serial port to view the log information.



About the Demo codes

- The demo codes are based on the pico-lorawan project, and updated to compatible the Pico-LoRa-SX1262-868M module
- For more details about the LoRaWAN protocol, pleaser refer to LoRa Allicance
- If you want to create your own cloud server, pleaser refer to lorawan-stack,chrpstack

Resources

- Schematic
- Demo Code
- github link
- SX1262 Datasheet
- link of TTS Document
- link of LoRaWAN document