KCM0A5S Open(SDK) Quick Start Guide

Short-Range Module Series

Version: 1.0

Date: 2025-07-29

Status: Released

At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local offices. For more information, please visit:
http://www.quectel.com/support/sales.htm.

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm.
Or email us at: support@quectel.com.

Legal Notices

We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an “as available” basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions

License Agreements

Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright

Our and third-party products hereunder may contain copyrighted material. Such copyrighted material shall not be copied, reproduced, distributed, merged, published, translated, or modified without prior written consent. We and the third party have exclusive rights over copyrighted material. No license shall be granted or conveyed under any patents, copyrights, trademarks, or service mark rights. To avoid ambiguities, purchasing in any form cannot be deemed as granting a license other than the normal non-exclusive, royalty-free license to use the material. We reserve the right to take legal action for noncompliance with abovementioned requirements, unauthorized use, or other illegal or malicious use of the material.

Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel's or third-party's servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.
b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.
c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.
d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

About the Document

Revision History

Version	Date	Author	Description
-	2025-04-30	Tam TANG/Vic CHENG	Creation of the document
1.0	2025-07-29	Tam TANG/Vic CHENG	First official release

Contents

Table Index

Figure Index

1 Introduction

Quectel KCM0A5S module supports open solution, which can simplify the software design and development process for IoT applications. For detailed information about the aforementioned open solution, please visit https://github.com/SiliconLabs/simplicity_sdk.

This document is applicable to open solution based on SDK build environment. It is the KCM0A5S module quick start guide. It outlines application development using the Quectel KCM0A5S module in a Wi-SUN network, including compilation environment setup, application development process, and general application debugging methods.

Wi-SUN is an IPv6 Sub-GHz mesh technology. It brings Smart Ubiquitous Networks to service providers, utilities, municipalities/local government, and other enterprises, by enabling interoperable, multi-service, and secure wireless mesh networks. Wi-SUN can be used for large-scale outdoor IoT wireless communication networks.

NOTE

For more details on Wi-SUN, visit https://docs.silabs.com/wisun/latest/wisun-getting-started-overview/.

2 Set up Compilation Environment

2.1. Hardware Environment

Table 1: Hardware Environment

Hardware	Quantity
KCM0A5S-TE-B	2
KCM0A5S module	2
J-Link debugger/ SI-DBG1015A Simplicity Link debugger	1
Raspberry Pi 4 Model B	1
Micro SD Card (64 GB or larger)	1
Card Reader	1
Router	1
Antenna	1
USB cable (for connecting the development board to PC)	1
Cable for connecting the J-Link debugger or SI-DBG1015A Simplicity Link debugger to the module development board	1
PC	1

NOTE

1. This document uses SI-DBG1015A Simplicity Link debugger as an example.
2. For instructions on how to connect KCM0A5S-TE-B to J-Link debugger or SI-DBG1015A Simplicity Link debugger, please see ***document [1]***.

3. Setting up a Wi-SUN local network requires a border router and a network node device. One KCM0A5S-TE-B (referred to as "Development Board A") and the Raspberry Pi 4 Model B form the border router, while another KCM0A5S-TE-B (referred to as "Development Board B") serves as the network node device.
4. Establishing a Wi-SUN local area network must include a border router and a network node device. Specifically:
   - Border Router: Consists of a Raspberry Pi 4B development board and one set of KCM0A5S-TE-B development boards (hereinafter referred to as "Development Board A").
   - Network Node Device: Served by another set of KCM0A5S-TE-B development boards (hereinafter referred to as "Development Board B").

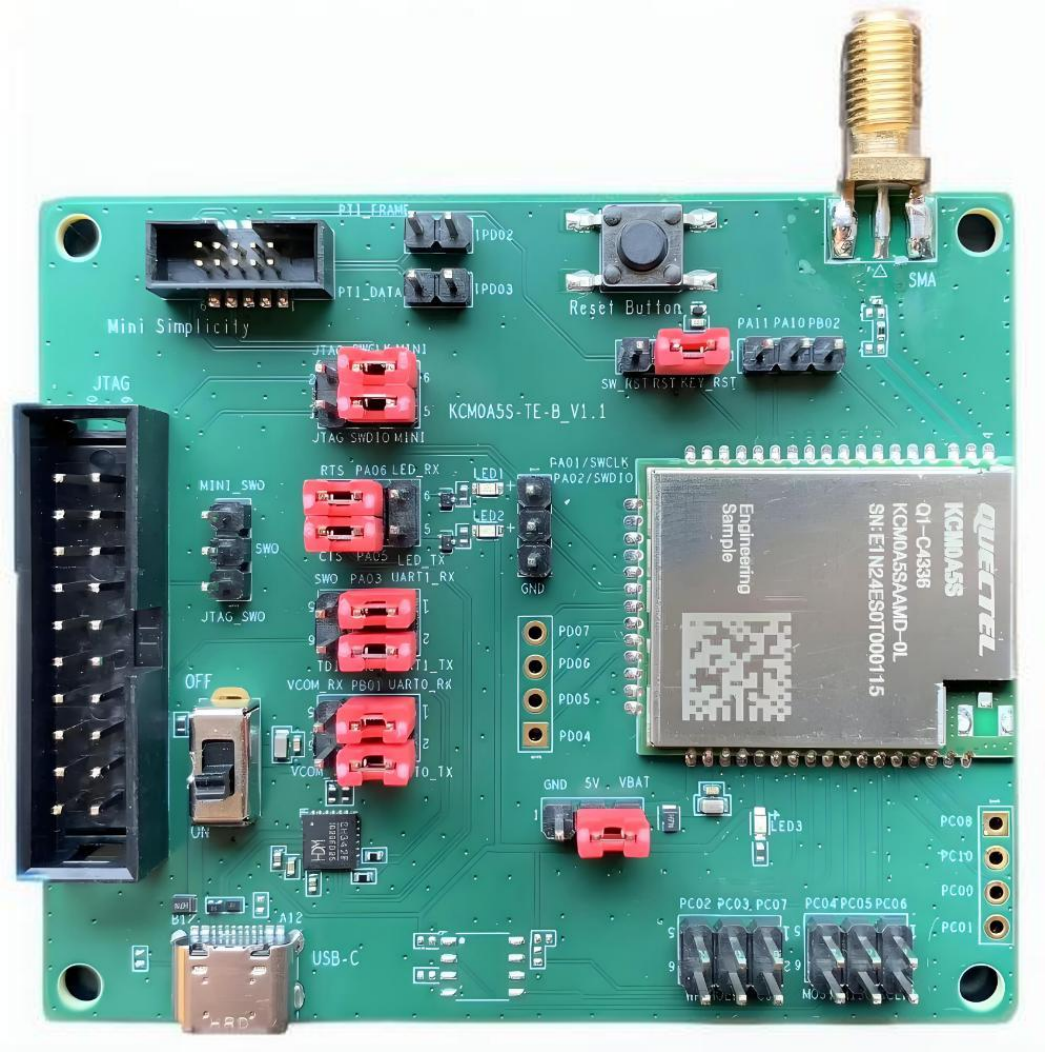


Figure 1: KCM0A5S-TE-B

NOTE

For the function definitions of KCM0A5S-TE-B interfaces, please see ***document [1]***.

Table 2: Pin Mapping for KCM0A5S

Module Pin No.	Module Pin Name	Main Chip Pin Name
1	GND	GND
2	ANT_RF	/
3	GND	GND
4	RESERVED	/
5	RESERVED	/
6	GPIO7	PB02
7	EUSART_RXD	PB01
8	EUSART_TXD	PB00
9	GPIO1	PA03
10	GPIO2	PA04
11	GPIO3	PA05
12	GPIO4	PA06
13	GPIO5	PA10
14	GPIO6	PA11
15	RESET_N	RESET_N
16	RESERVED	/
17	RESERVED	/
18	RESERVED	/
19	RESERVED	/
20	RESERVED	/

21	RESERVED	/
22	RESERVED	/
23	SWCLK	PA01
24	SWDIO	PA02
25	GPIO23	PD07
26	GPIO22	PD06
27	GPIO21	PD05
28	GPIO20	PD04
29	GPIO19	PD03
30	GPIO18	PD02
31	RESERVED	/
32	VBAT	VBAT
33	VBAT	VBAT
34	GND	GND
35	GND	GND
36	3V3_OUT	3V3_OUT
37	GPIO9	PC01
38	GPIO10	PC02
39	GPIO11	PC03
40	GPIO15	PC07
41	GPIO12	PC04
42	GPIO13	PC05
43	GPIO14	PC06
44	GPIO8	PC00
45	GPIO17	PC10

46	GPIO16	PC08
47	GND	GND
48	GND	GND

NOTE

See ***document [2]*** for details of pin information and the main chip.

2.2. Software Environment

Table 3: Software Environment

Type	Description
Tool	Install Simplicity Studio (Simplicity Studio SV5.10.0 is used in this document) on the PC
SSDK	Simplicity SDK V2024.6.2

NOTE

1. For instructions on how to use Simplicity Studio SV5.10.0, please refer to https://docs.silabs.com/simplicity-studio-5-users-guide/latest/ss-5-users-guide-overview.
2. This document uses Simplicity SDK V2024.6.2 as an example. You can use the corresponding SDK according to your actual needs.

2.2.1. Install Simplicity Studio

Step 1: Download Simplicity Studio toolkit from https://www.silabs.com/developers/simplicity-studio.

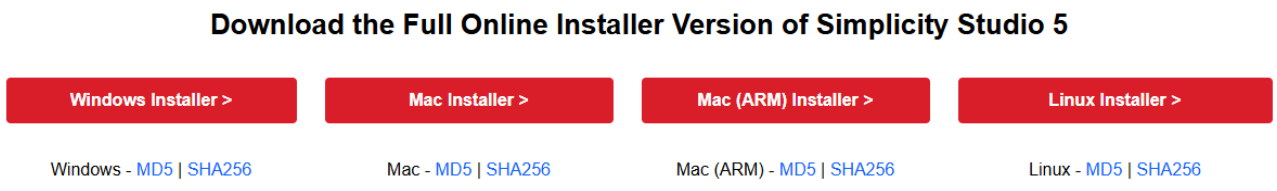


Figure 2: Download Simplicity Studio Toolkit

Step 2: After decompressing the *SimplicityStudio-5.iso* toolkit, double-click *setup.exe* in the toolkit and follow the prompts to complete the installation.

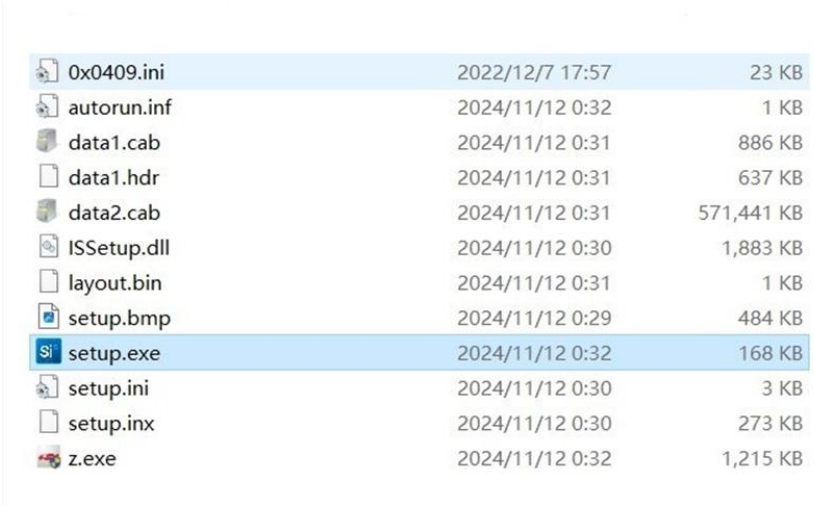


Figure 3: Install setup.exe

Step 3: Connect Development Board A to SI-DBG1015A Simplicity Link debugger, and then connect the debugger to your PC.

Step 4: Open Simplicity Studio. The software automatically detects the debugger and displays it (e.g., "(69602814) (ID:69602814)") in the "Debug Adapters" section of the main interface. Click "**Install**" on the Simplicity Studio toolbar to open the "Installation Manager", and then select "**Install by connecting device(s)**" to proceed to the basic component installation interface.

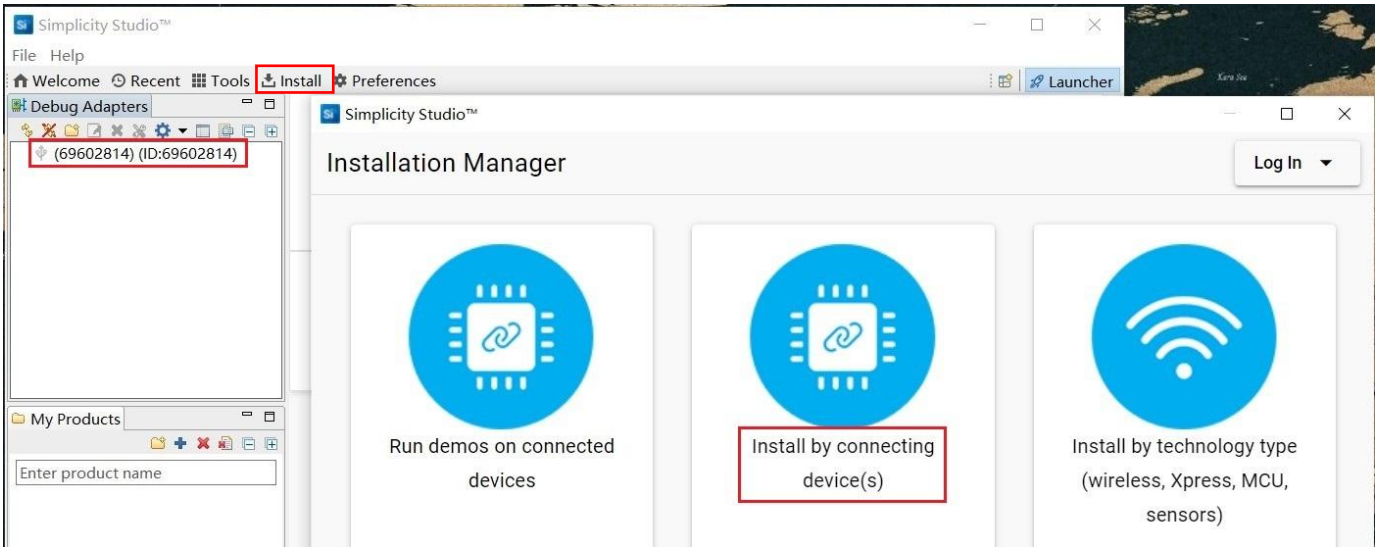


Figure 4: Select Basic Component Installation Method

1) Once the "Installation Manager" interface is displayed, click "**Update All**" on the "**Product Updates**" tab to update all basic components.

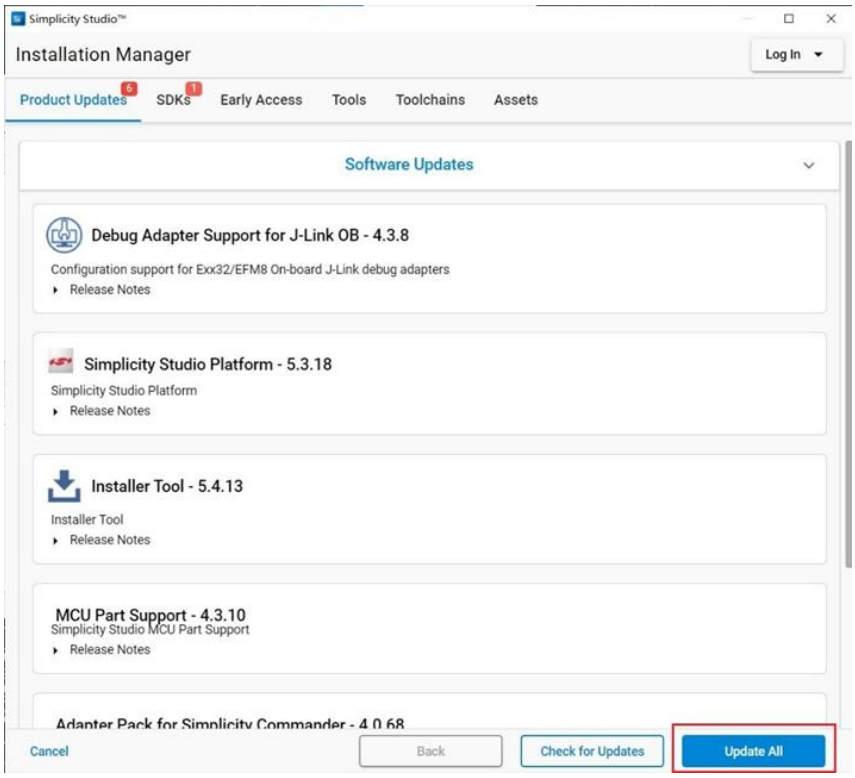


Figure 5: Update all Basic Components

2) After the update is completed, the following window opens. Click "**Restart**" to restart the software and finalize the process.

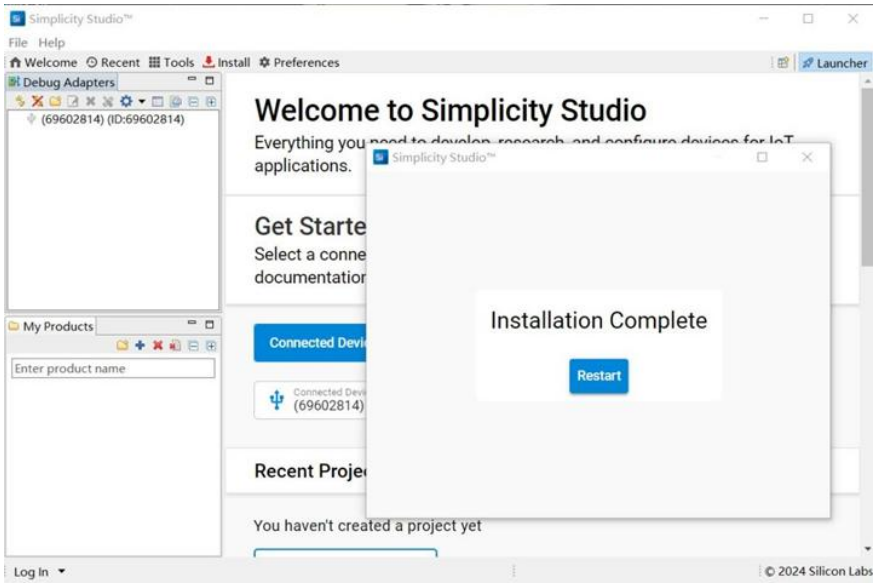


Figure 6: Update Completed

2.2.2. Install Components

Step 1: Connect Development Board B to SI-DBG1015A Simplicity Link debugger, and then connect the debugger to your PC.

Step 2: In the "Debug Adapters" section of the Simplicity Studio main interface, find and right-click the displayed debugger (440301390) (ID: 440301390), and select "**Device configuration**".

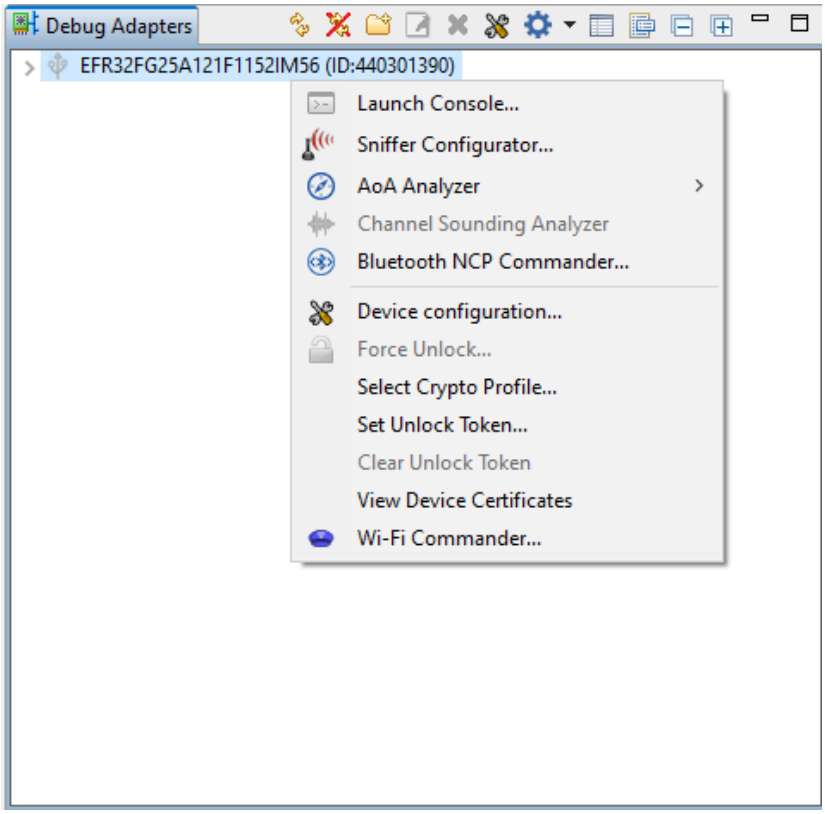


Figure 7: Select "Device configuration"

Step 3: The "(440301390)" configuration interface opens on the "**Device hardware**" tab. The "**Target Interface**" is set to "SWD" by default. Click "**Detect Part**", and the tool automatically detects and displays the chip model EFR32FG25A121F1152IM56. Then, click "**OK**". The debugger is displayed as EFR32FG25A121F1152IM56 (ID: 440301390) in the "Debug Adapters" section.

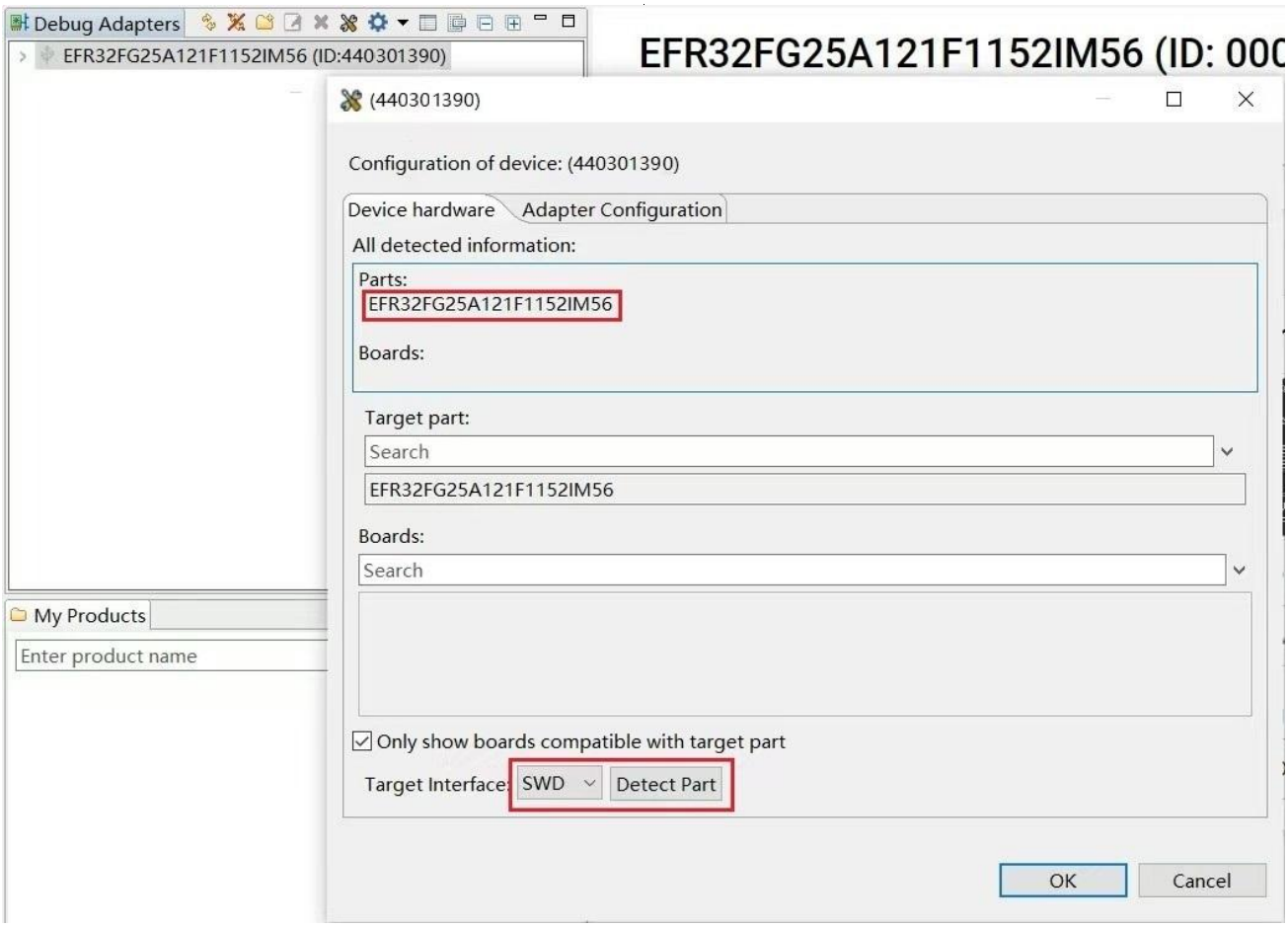


Figure 8: Detect Chip Model

Step 4: Click "**Install**" on the Simplicity Studio toolbar to open the Installation Manager.

Figure 9: Open Installation Manager

Step 5: Select "**Install by connecting device(s)**" to install the components needed for development.

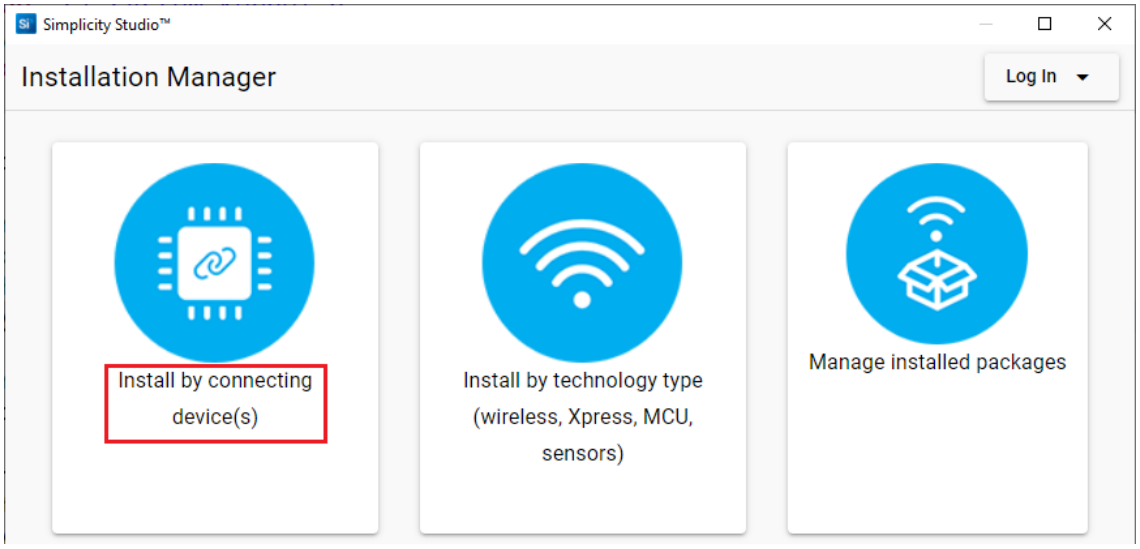


Figure 10: Select Component Installation Method

Step 6: Select "**EFR32FG25A121F1152IM56 (ID: 000440301390)**", and then click "**Next**".

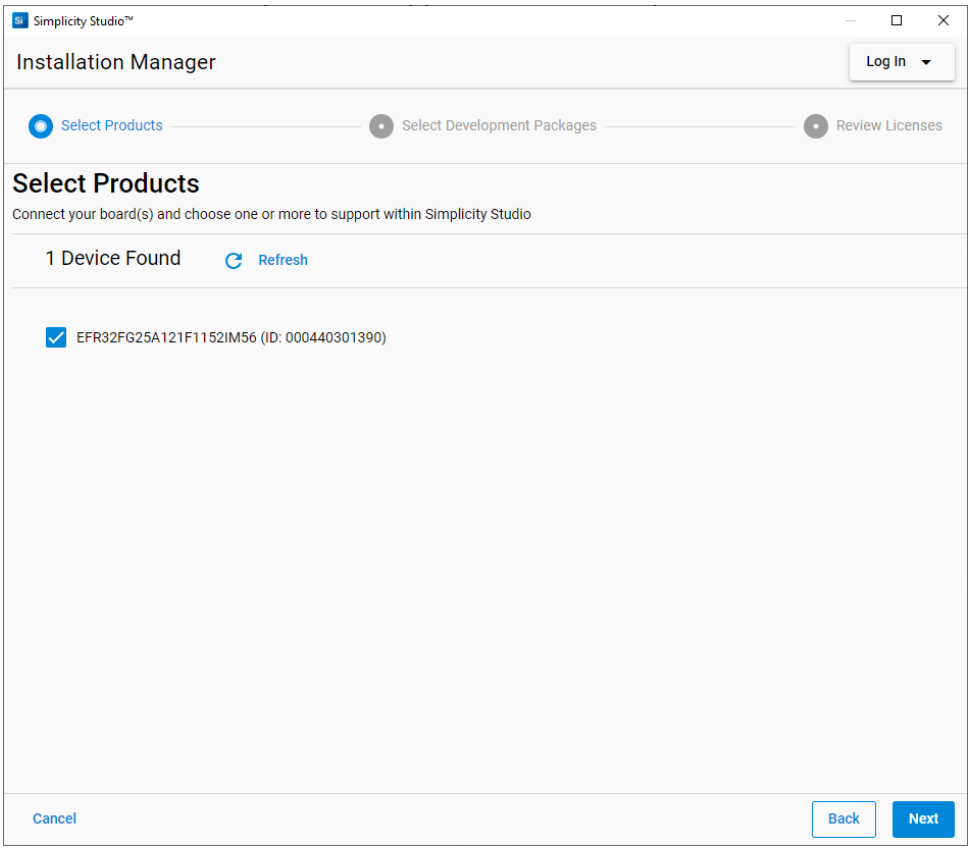


Figure 11: Select Target Device

Step 7: In the "Package Installation Options" section, select "**Auto**" to automatically install the corresponding components, then click "**Next**".

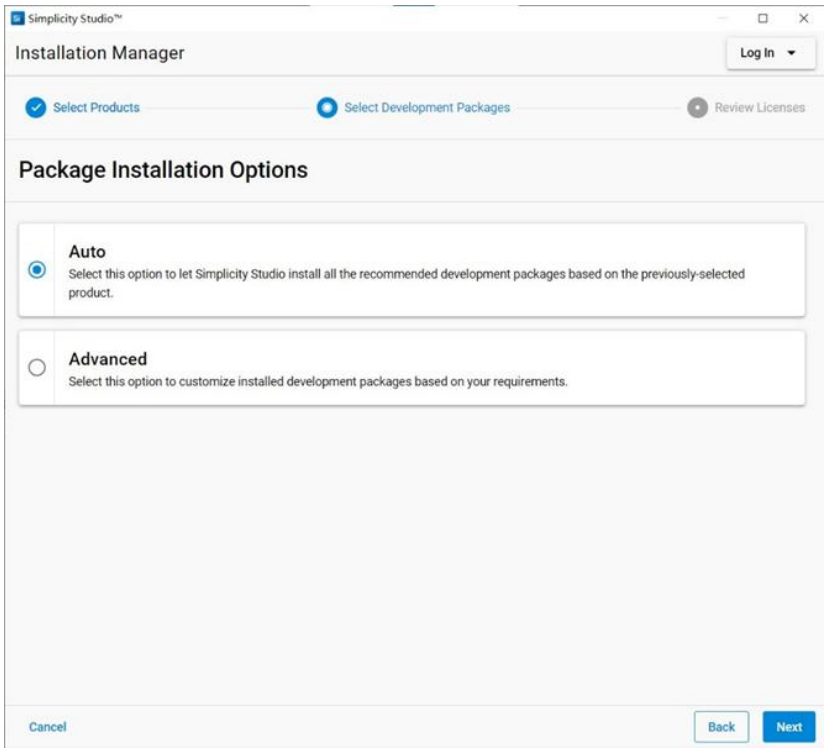


Figure 12: Configure Component Installation Option

Step 8: Once the installation is completed, click "**Restart**" in the pop-up window to restart the software.

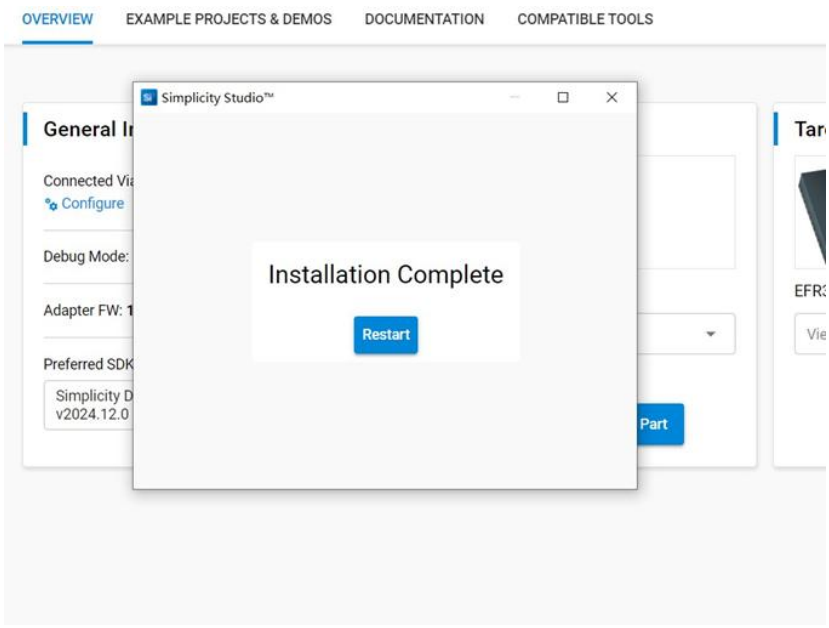


Figure 13: Installation Completed

When installing the Simplicity Studio components, an installation failure message may appear. If this occurs, it is recommended to reinstall the components. If the issue persists, try uninstalling and reinstalling Simplicity Studio.

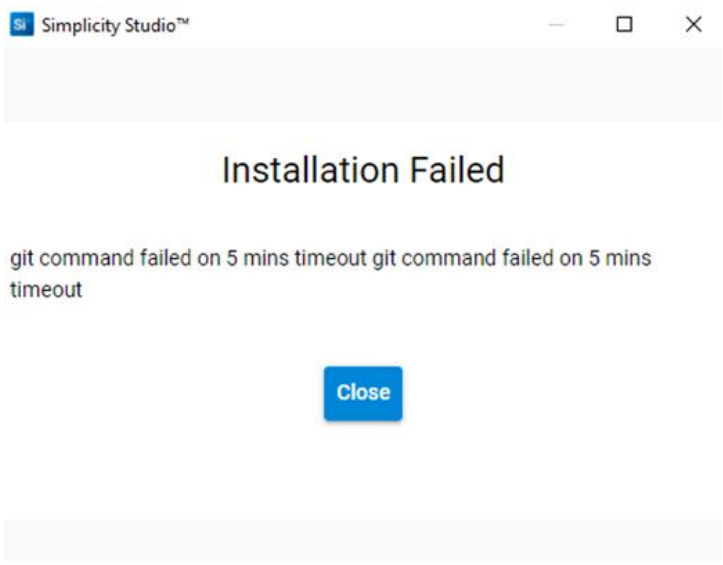


Figure 14: Installation Failure Message

2.2.3. Install SDK

Step 1: Click "**Install**" on the Simplicity Studio toolbar to open the Installation Manager, then select "**Manage installed packages**".

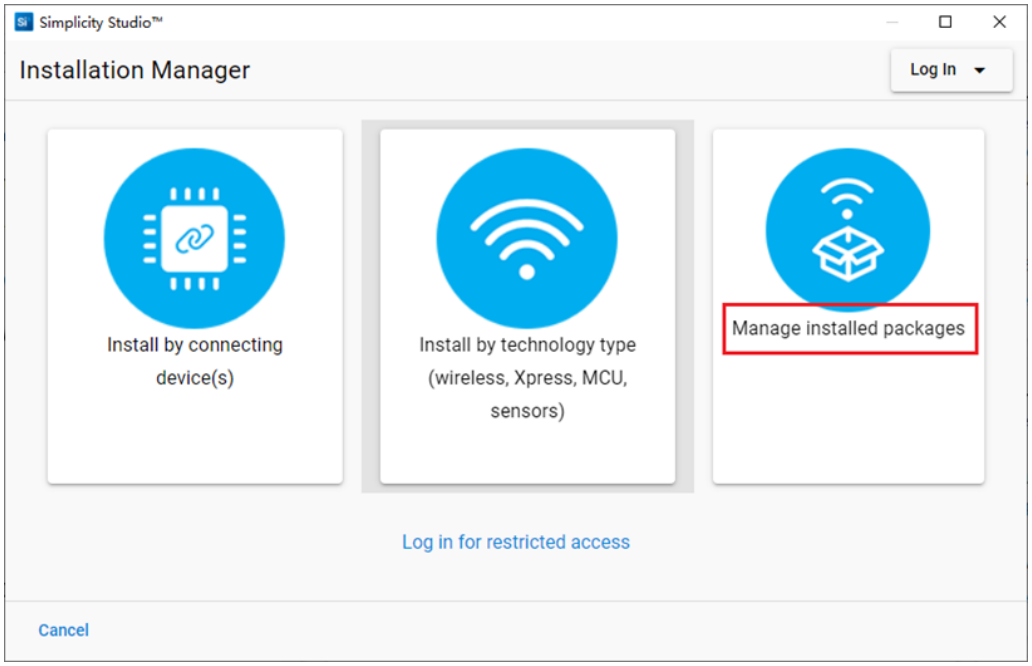
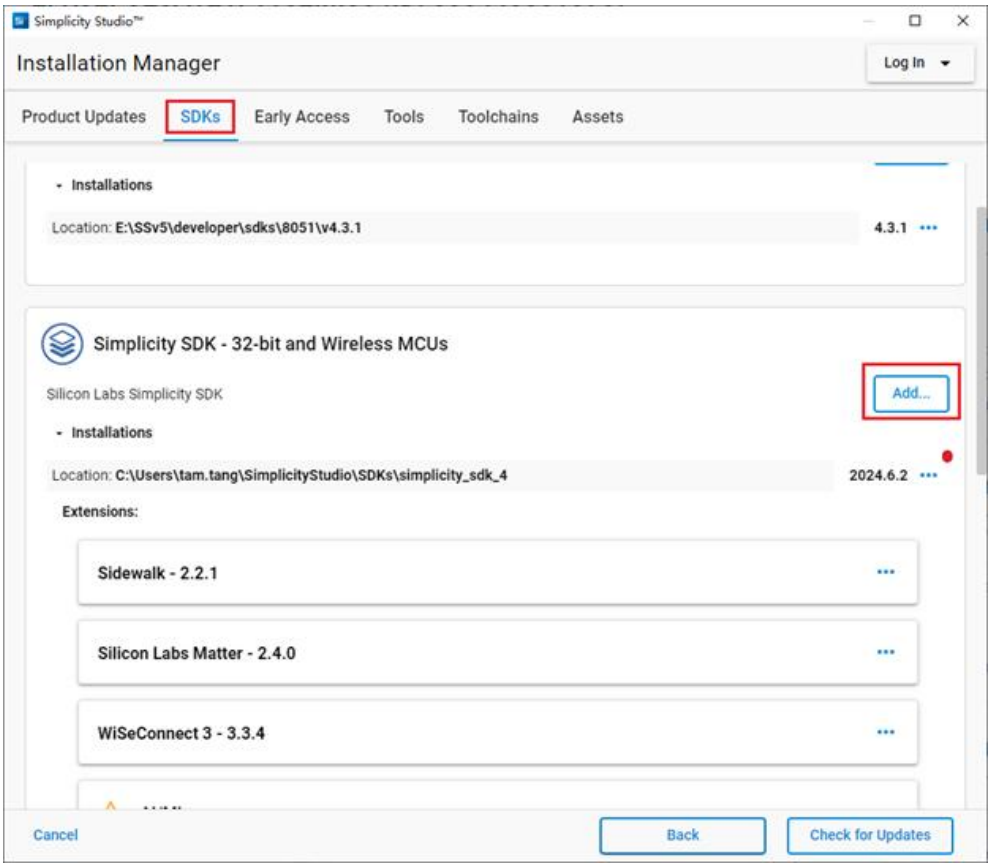


Figure 15: Manage Packages

Step 2: Select the "**SDKs**" tab, and click "**Add…**". In the "Add new Simplicity SDK" pop-up window, select the required SDK version (e.g., 2024.6.2) and set the SDK storage location. Then, click "**FINISH**" to proceed with the installation.



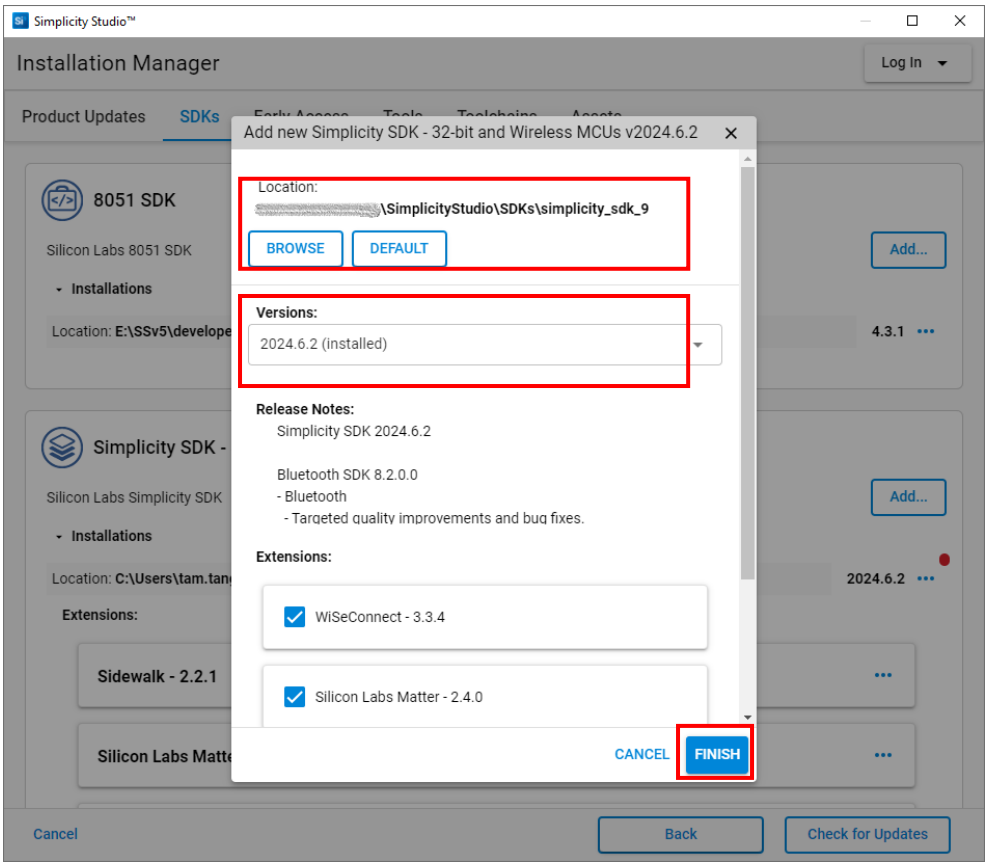


Figure 16: Install SDK

2.3. Install Raspberry Pi Environment

The Border Router serves as the core component of a Wi-SUN network. It handles network initialization, configuration, and management, enabling end devices to join the network while facilitating communication both between devices and between devices and the Border Router. This document uses the Raspberry Pi 4 Model B with the Development Board A as an example to explain how to install the Raspberry Pi environment.

2.3.1. Install Raspberry Pi System

Step 1: Download the Ubuntu image *ubuntu-24.04.1-preinstalled-server-arm64+raspi.img.xz* from https://cdimage.ubuntu.com/releases/24.04/release/ubuntu-24.04.1-preinstalled-server-arm64+raspi.img.xz.

ubuntu-24.04.1-preinstalled-desktop-arm64+raspi.img.xz.zsync	2024-08-29 15:59	5.2M	Preinstalled desktop image for Raspberry Pi Generic (64-bit ARM) computers (preinstalled SD Card image)
ubuntu-24.04.1-preinstalled-desktop-arm64+raspi.manifest	2024-08-27 14:46	49K	Preinstalled desktop image for Raspberry Pi Generic (64-bit ARM) computers (contents of live filesystem)
ubuntu-24.04.1-preinstalled-server-arm64+raspi.img.xz	2024-08-27 14:33	1.1G	Preinstalled server image for Raspberry Pi Generic (64-bit ARM) computers (preinstalled SD Card image)
ubuntu-24.04.1-preinstalled-server-arm64+raspi.img.xz.zsync	2024-08-29 16:04	2.1M	Preinstalled server image for Raspberry Pi Generic (64-bit ARM) computers (preinstalled SD Card image)

Figure 17: Ubuntu Image

Step 2: Download the Raspberry Pi Imager *imager_1.8.5.exe* from https://downloads.raspberrypi.org/imager/imager_latest.exe. Double-click *imager_1.8.5.exe* to open it, and follow the prompts to install Raspberry Pi Imager v1.8.5.

Step 3: Insert the Micro SD card into the PC via the card reader.

Step 4: Open the Raspberry Pi Imager. The main interface of the tool is shown below. Click "**CHOOSE DEVICE**".

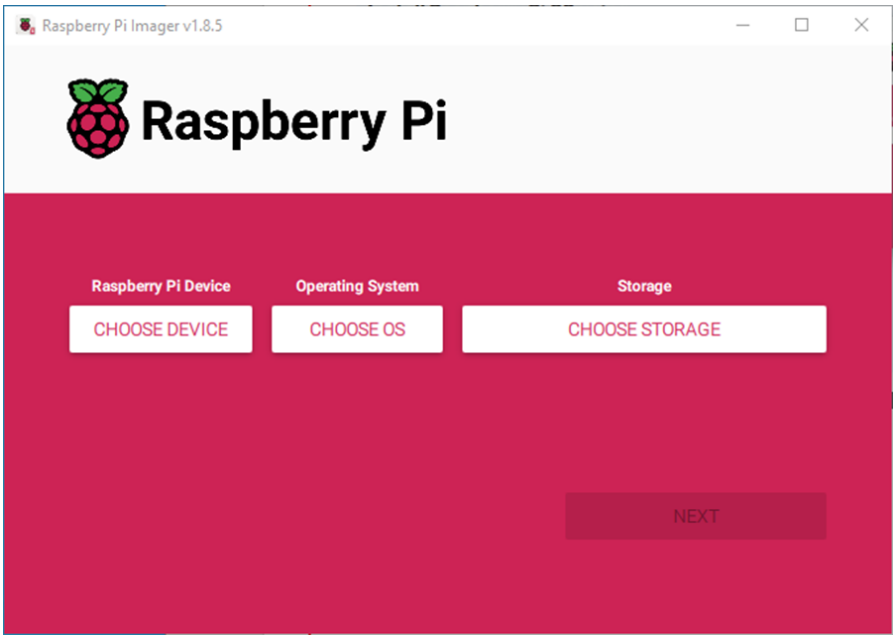


Figure 18: Main Interface of Raspberry Pi Imager

Step 5: In the "Raspberry Pi Device" pop-up window, select the Raspberry Pi model (e.g., Raspberry Pi 4).

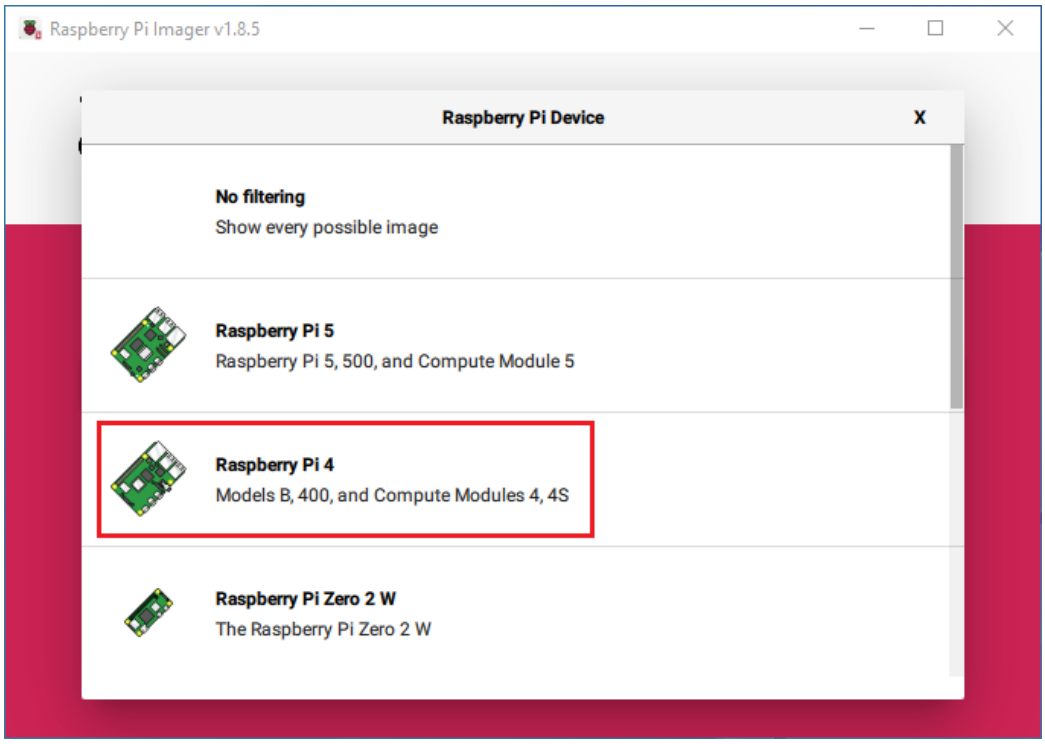


Figure 19: Select Raspberry Pi Model

Step 6: Click "**CHOOSE OS**" on the main interface of the tool. In the "Operating System" pop-up window, click "**Use custom**" to open the image selection window and select the custom image.

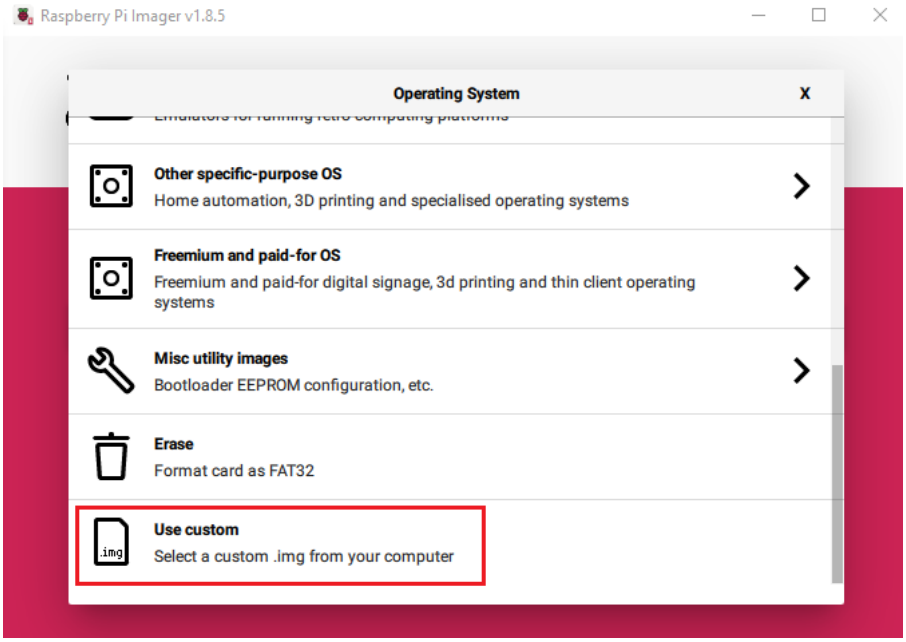


Figure 20: Choose Custom Image

Step 7: In the "Select image" pop-up window, select the Ubuntu image downloaded in ***Step 1***, then click "**Open**".

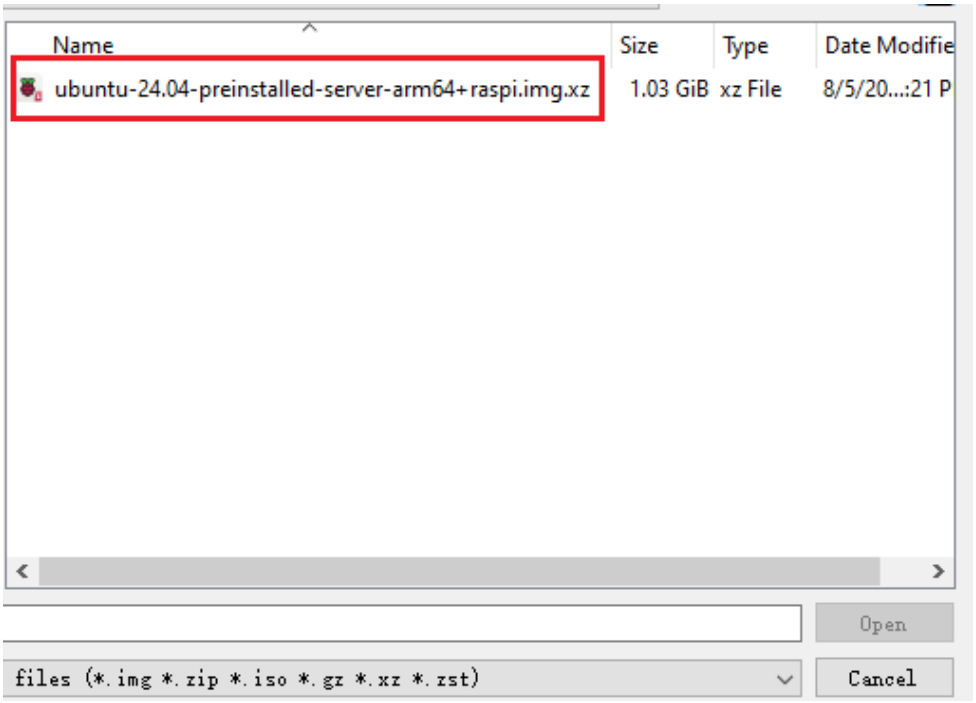


Figure 21: Select Ubuntu Image

Step 8: Click "**CHOOSE STORAGE**" on the main interface of the tool. Select the storage card in the "Storage" pop-up window.

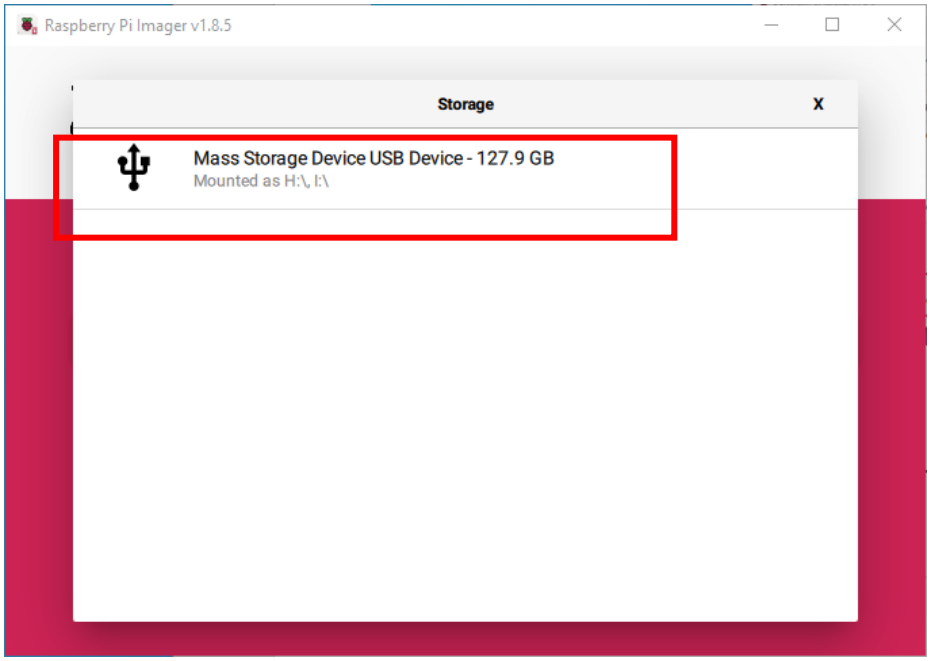


Figure 22: Select Storage Card

Step 9: Click "**NEXT**" on the main interface of the tool. In the "Use OS customisation?" pop-up window, select "**EDIT SETTINGS**" to configure the system parameters.

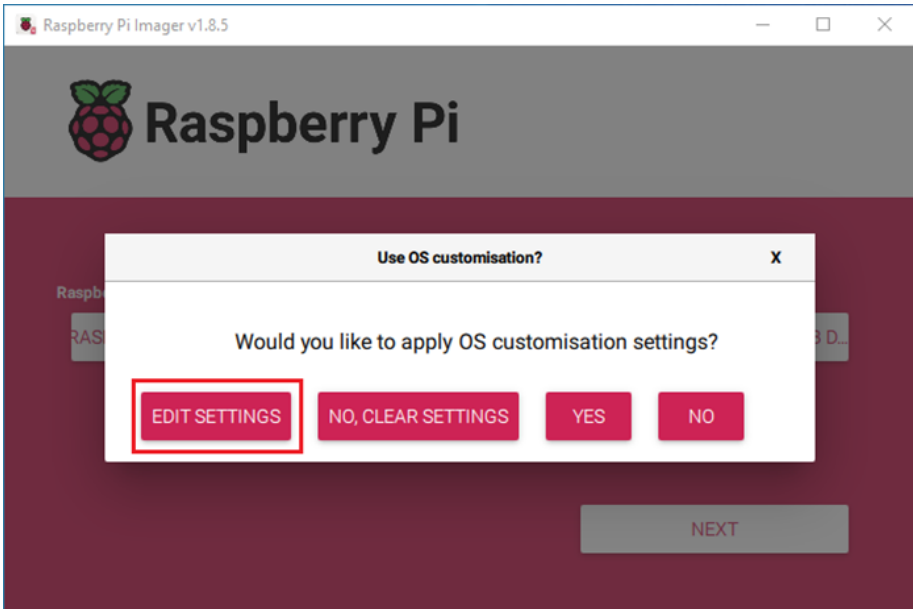


Figure 23: Configure System Parameters

Step 10: The "OS Customisation" window opens on the "**GENERAL**" tab. Set the hostname, username, and password. Select the "**SERVICES**" tab, select "**Enable SSH**" and select "**Use password authentication**". Then, click "**SAVE**" to save the configuration and close the "OS Customisation" window.

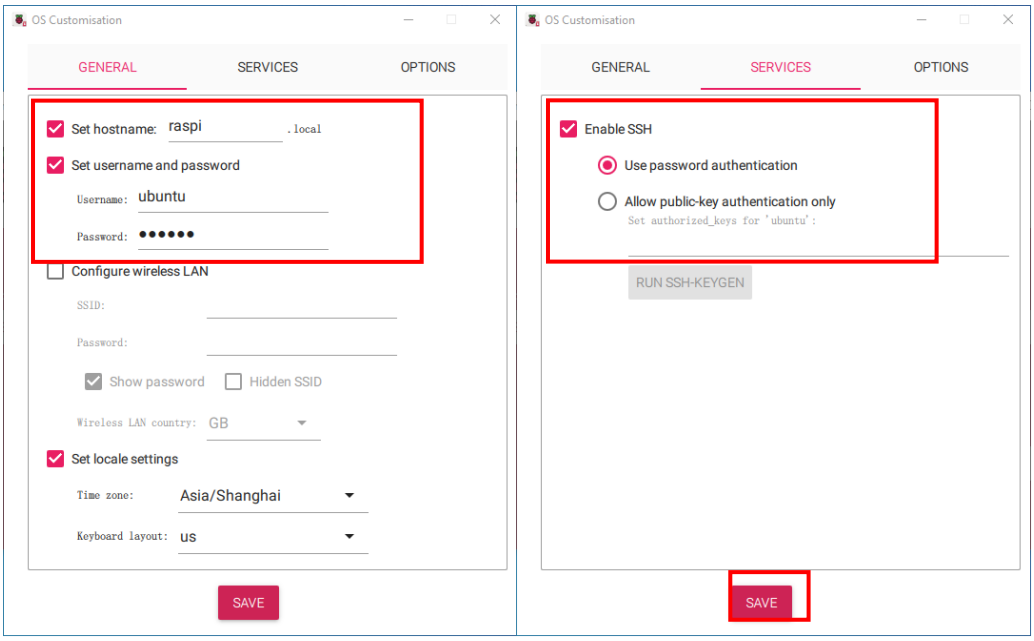


Figure 24: Set Login Parameters

Step 11: In the "Use OS customisation?" pop-up window, click "**Yes**" to apply the settings.

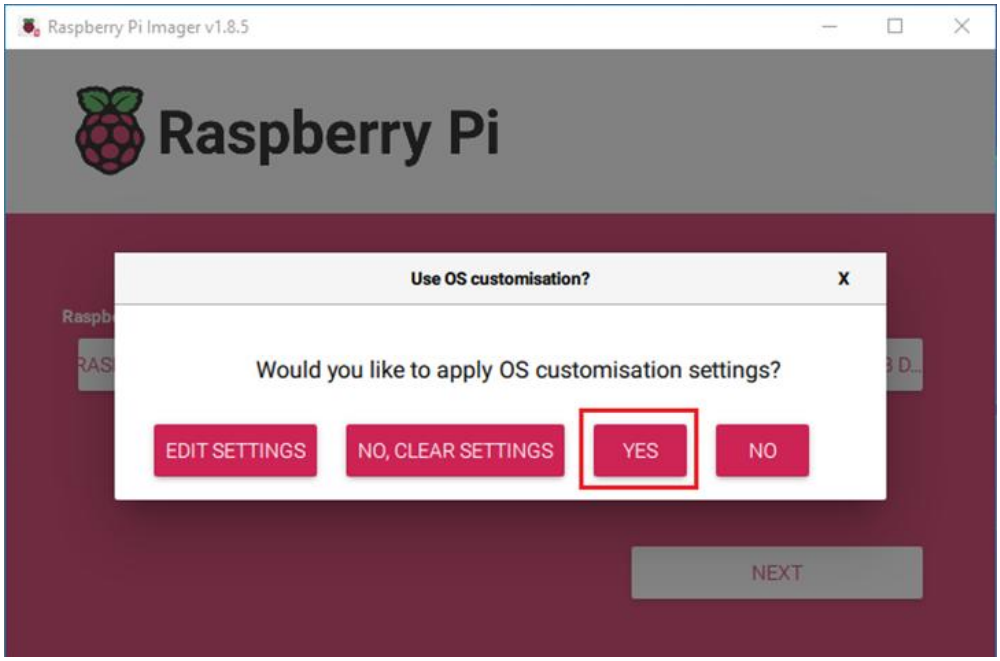


Figure 25: Apply Settings

Step 12: In the "Warning" pop-up window, click "**YES**" to erase the data and begin flashing the image.

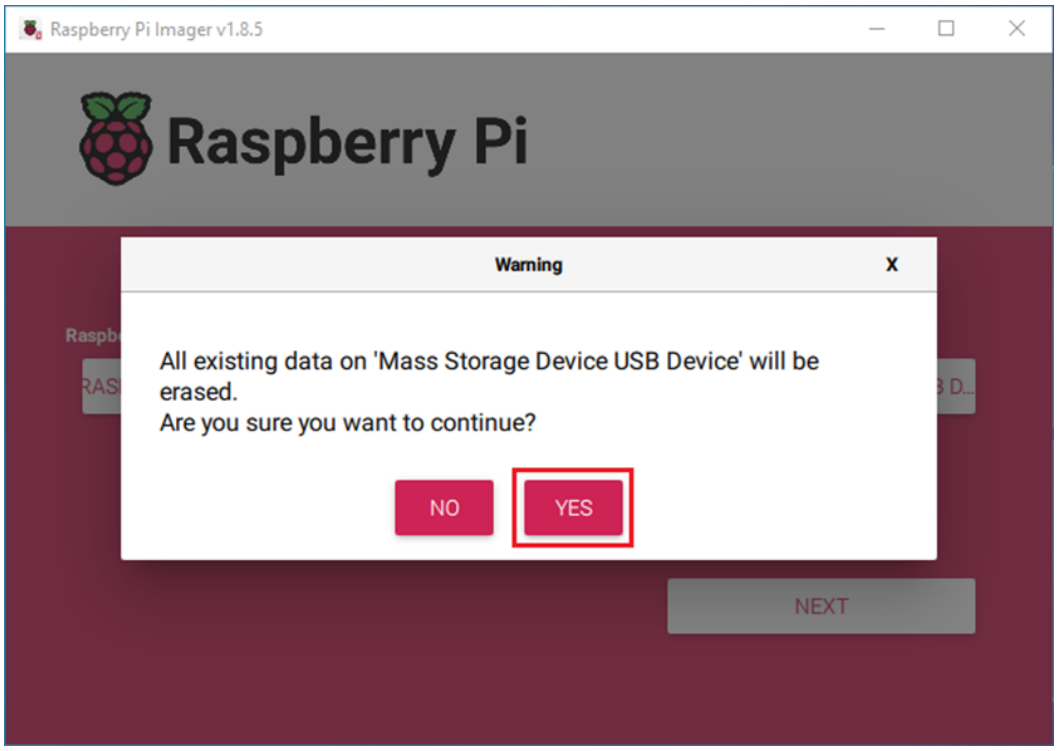


Figure 26: Erase Data

Flashing the image may take some time. Please wait until the process is complete.

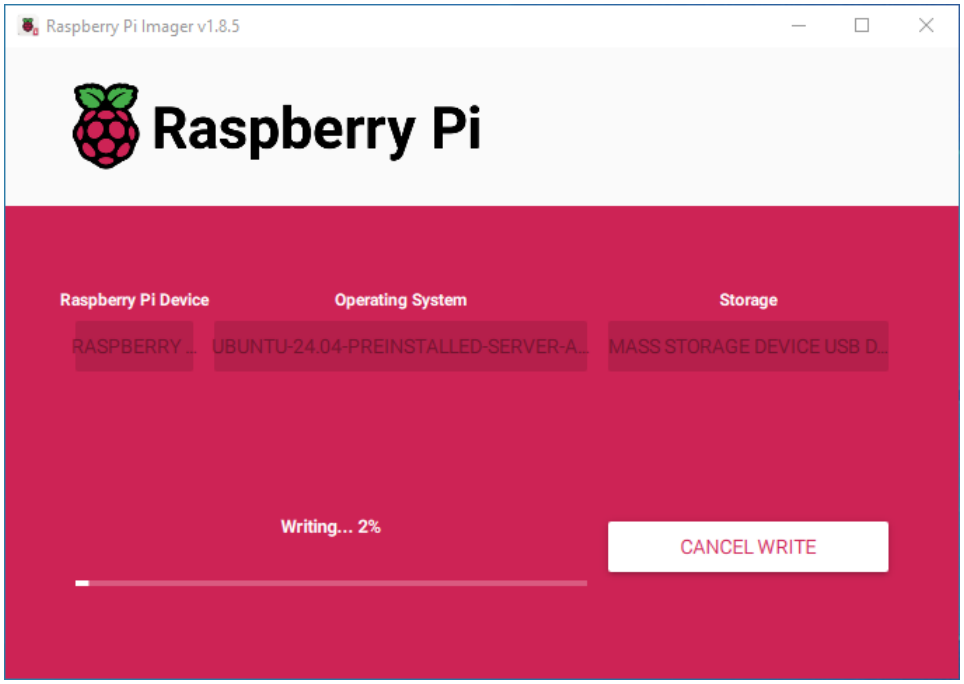


Figure 27: Flash Image

Step 13: During the flashing process, the following error message may appear to indicate that the storage card cannot be found. Click "**OK**" to ignore this message.

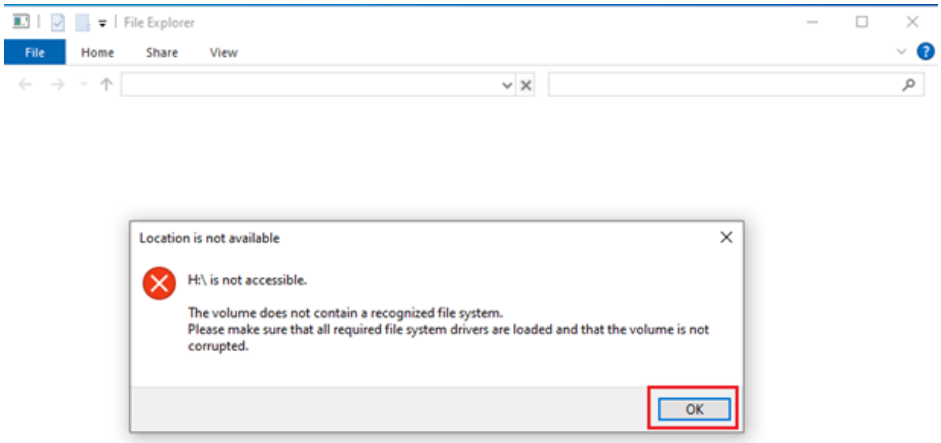


Figure 28: Error Message

After the flashing is completed, the following window appears.

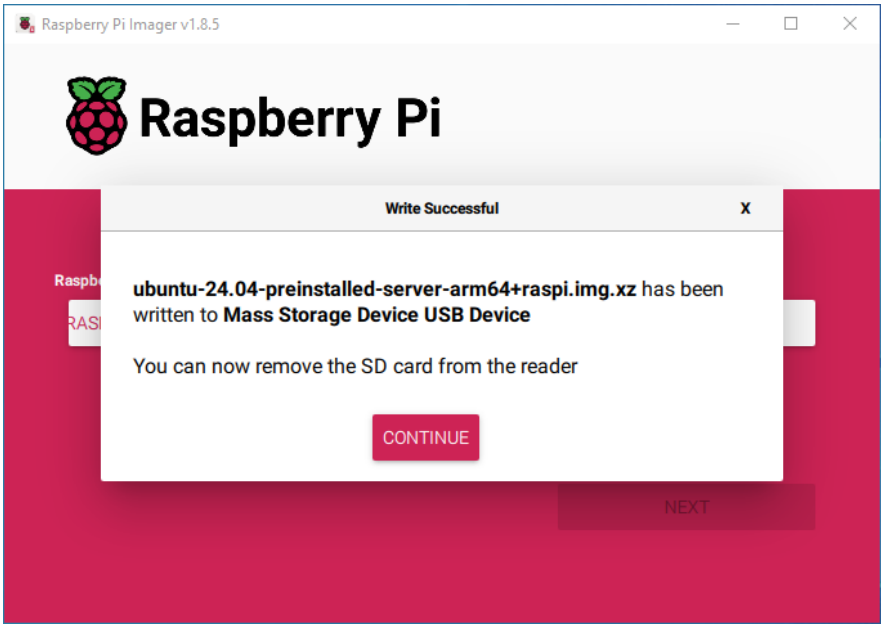


Figure 29: Flashing Completed

2.3.2. Log into Raspberry Pi 4 Model B

Step 1: Insert the Micro SD card into the Raspberry Pi 4 Model B. Connect the Raspberry Pi 4 Model B to a power source and use an Ethernet cable to connect it to the router's LAN port.

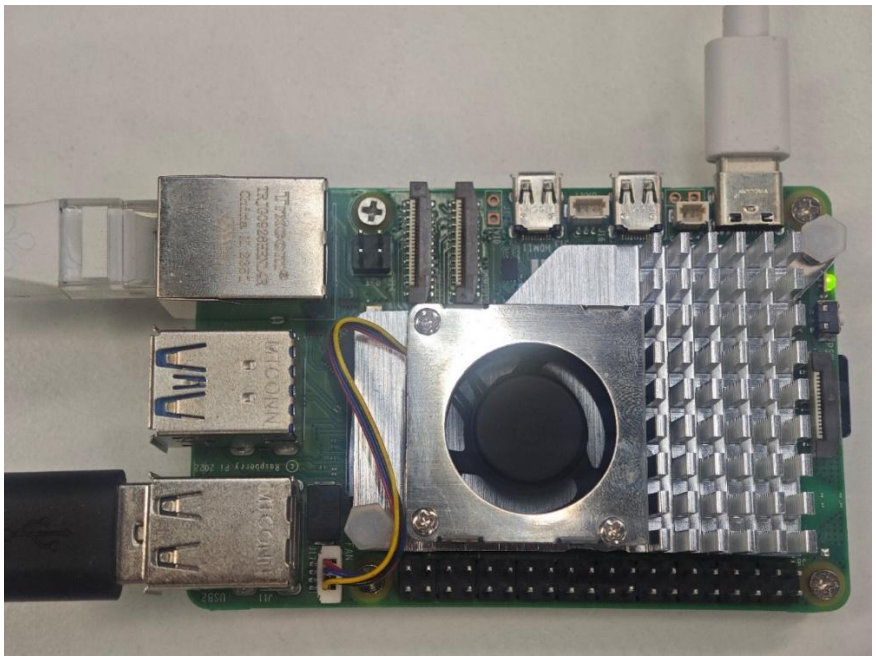


Figure 30: Raspberry Pi 4 Model B Connections

Step 2: Check the IP address of the Raspberry Pi 4 Model B through the router's management interface.

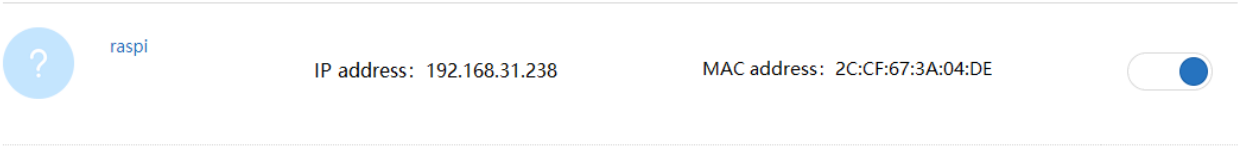


Figure 31: Router's Management Interface

Step 3: Log in to the Raspberry Pi 4 Model B using the terminal tool MobaXterm.

1) Download MobaXterm from https://mobaxterm.mobatek.net. Decompress the downloaded MobaXterm installation package, double-click the .msi file, and follow the prompts to complete the installation.
2) After the installation is completed, open the tool, click "**Session**" on the toolbar of the main interface to open the "Session settings" window.

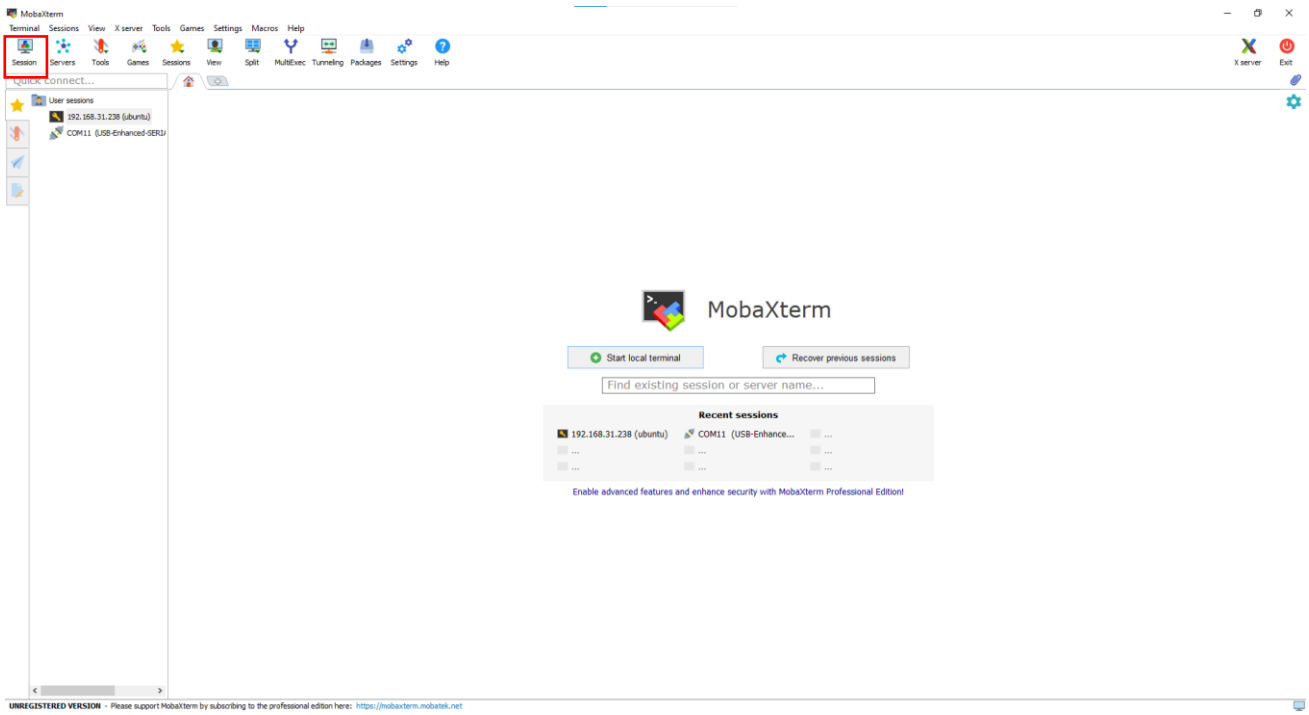


Figure 32: MobaXterm Main Interface

3) In the "Session settings" window, select "**SSH**". In the "Basic SSH settings" section, enter the Raspberry Pi 4 Model B's IP address and username ("**Port**" is set to 22 by default, and there is no need to modify it). Then click "**OK**".

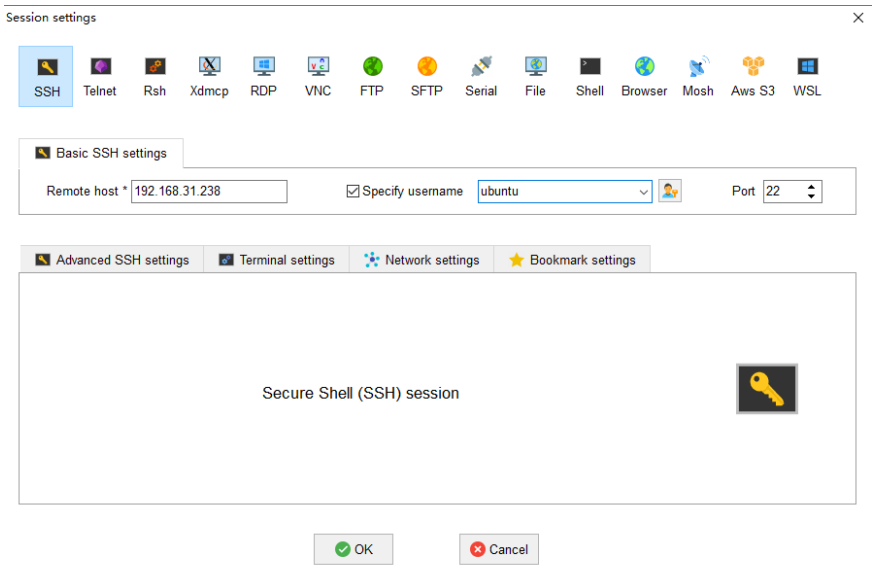


Figure 33: "Session settings" Window

4) In the pop-up terminal window, enter the password as prompted, then press the "**Enter**" key to log in to the Raspberry Pi 4 Model B. Once you are logged in successfully, the terminal window displays the following information:

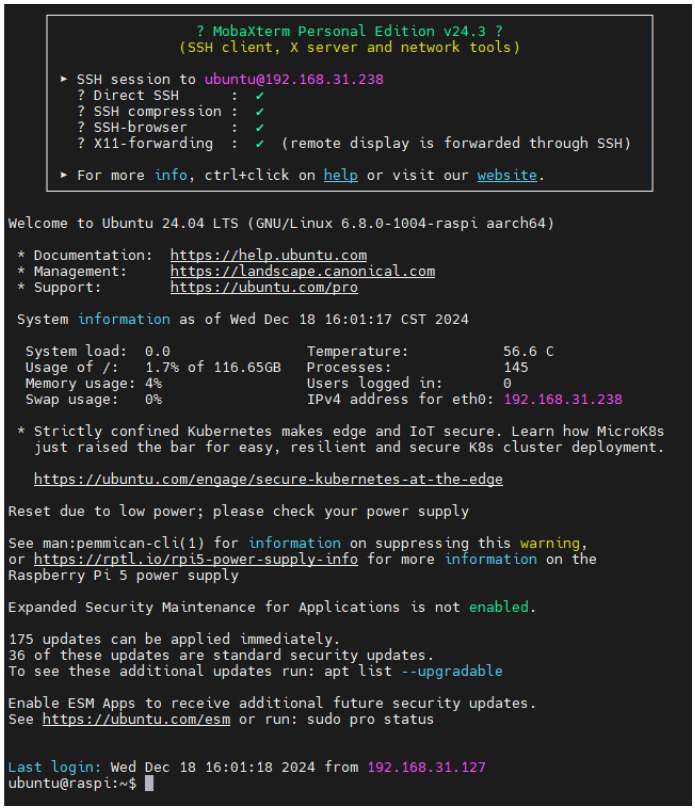


Figure 34: Log in to Raspberry Pi 4 Model B

2.3.3. Install wisun-br-linux

To implement the Wi-SUN protocol on a Linux device, you need to install the wisun-br-linux project. The wisun-br-linux project allows you to use Linux devices as border routers for Wi-SUN networks, implementing the Wi-SUN protocol by porting the Wi-SUN protocol stack to the Linux devices. You can download wisun-br-linux project from https://github.com/SiliconLabs/wisun-br-linux.

Step 1: Execute the following commands in the MobaXterm tool terminal window to install the necessary environment dependencies.

```
//Install dependencies
sudo apt-get install libnl-3-dev libnl-route-3-dev libcap-dev
sudo apt-get install libsystemd-dev cmake ninja-build pkg-config lrzsz g++
//Install MbedTLS
cd ..
git clone --branch=v3.6.2 --recurse-submodules https://github.com/ARMmbed/mbedtls
cd mbedtls
cmake -G Ninja .
ninja
sudo ninja install
```

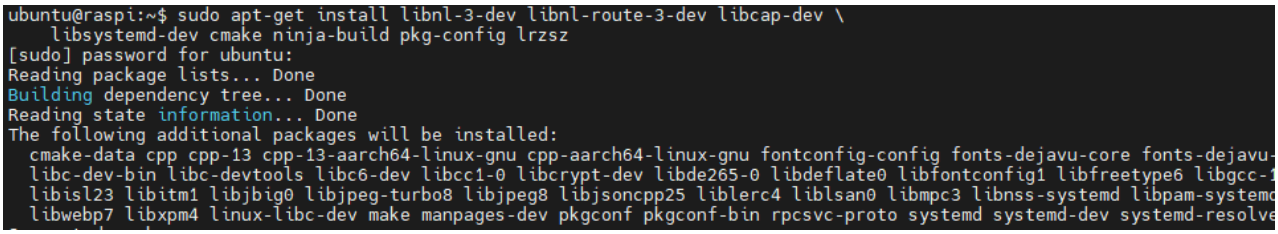


Figure 35: Dependency Installation Process

```
ubuntu@raspi:~$ ls
wisun-br-linux
ubuntu@raspi:~$ cd wisun-br-linux/
ubuntu@raspi:~/wisun-br-linux$ ls
CHANGES.md  CMakeLists.txt  DBUS.md  HIF.md  LICENSE  LICENSES  README.md  app_wsbrd  app_wsrd  cmake
ubuntu@raspi:~/wisun-br-linux$ cargo fetch --manifest-path=tools/wsbrd_cli/Cargo.toml
    Updating crates.io index
  Downloaded atty v0.2.14
  Downloaded hermit-abi v0.1.19
  Downloaded bitflags v0.9.1
  Downloaded vec_map v0.8.2
  Downloaded unicode-width v0.1.11
  Downloaded libdbus-sys v0.2.2
  Downloaded pkg-config v0.3.27
  Downloaded dbus v0.9.6
  Downloaded libc v0.2.149
  Downloaded ansi_term v0.9.0
  Downloaded strsim v0.6.0
  Downloaded textwrap v0.9.0
  Downloaded clap v2.27.0
  Downloaded winapi v0.3.9
  Downloaded winapi-i686-pc-windows-gnu v0.4.0
  Downloaded winapi-x86_64-pc-windows-gnu v0.4.0
  Downloaded 16 crates (8.2 MB) in 3.07s (largest was `winapi-x86_64-pc-windows-gnu` at 2.9 MB)
ubuntu@raspi:~/wisun-br-linux$
```

Figure 36: Install wsbrd_cli Dependency

```
ubuntu@raspi:~/wisun-br-linux$ cd ..
ubuntu@raspi:~$ git clone --branch=v3.6.2 --recurse-submodules https://github.com/ARMmbed/mbedtls
Cloning into 'mbedtls'...
remote: Enumerating objects: 265719, done.
remote: Counting objects: 100% (362/362), done.
remote: Compressing objects: 100% (190/190), done.
remote: Total 265719 (delta 270), reused 172 (delta 172), pack-reused 265357 (from 3)
Receiving objects: 100% (265719/265719), 127.46 MiB | 3.34 MiB/s, done.
Resolving deltas: 100% (207385/207385), done.
Note: switching to '107ea89daaefb9867ea9121002fbbdf926780e98'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

Submodule 'framework' (https://github.com/Mbed-TLS/mbedtls-framework) registered for path 'framework'
Cloning into '/home/ubuntu/mbedtls/framework'...
remote: Enumerating objects: 204805, done.
remote: Counting objects: 100% (1193/1193), done.
remote: Compressing objects: 100% (535/535), done.
remote: Total 204805 (delta 778), reused 664 (delta 658), pack-reused 203612 (from 4)
Receiving objects: 100% (204805/204805), 95.30 MiB | 3.32 MiB/s, done.
Resolving deltas: 100% (158271/158271), done.
Submodule path 'framework': checked out '94599c0e3b5036e086446a51a3f79640f70f22f6'
```

Figure 37: Download mbedtls Project

```
ubuntu@raspi:~$ cd mbedtls
ubuntu@raspi:~/mbedtls$ cmake -G Ninja .
-- The C compiler identification is GNU 13.3.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Found Python3: /usr/bin/python3 (found version "3.12.3") found components: Interpreter
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Success
-- Found Threads: TRUE
-- Performing Test C_COMPILER_SUPPORTS_WFORMAT_SIGNEDNESS
-- Performing Test C_COMPILER_SUPPORTS_WFORMAT_SIGNEDNESS - Success
-- Configuring done (1.3s)
-- Generating done (0.2s)
-- Build files have been written to: /home/ubuntu/mbedtls
ubuntu@raspi:~/mbedtls$ ninja
[690/690] Linking C executable tests/test_suite_x509parse
ubuntu@raspi:~/mbedtls$ sudo ninja install
[sudo] password for ubuntu:
[0/1] Install the project...
-- Install configuration: ""
-- Installing: /usr/local/lib/cmake/MbedTLS/MbedTLSConfig.cmake
-- Installing: /usr/local/lib/cmake/MbedTLS/MbedTLSConfigVersion.cmake
-- Installing: /usr/local/lib/cmake/MbedTLS/MbedTLSTargets.cmake
-- Installing: /usr/local/lib/cmake/MbedTLS/MbedTLSTargets-noconfig.cmake
-- Installing: /usr/local/include/mbedtls/aes.h
-- Installing: /usr/local/include/mbedtls/aria.h
-- Installing: /usr/local/include/mbedtls/asn1.h
-- Installing: /usr/local/include/mbedtls/asn1write.h
-- Installing: /usr/local/include/mbedtls/base64.h
```

Figure 38: Install mbedtls Project

Step 2: Execute the following commands to download the wisun-br-linux project source code, compile, and install the wisun-br-linux project.

```
//Download source code
git clone https://github.com/SiliconLabs/wisun-br-linux.git
//Enter directory
cd wisun-br-linux/
//Install dependencies
sudo apt-get install cargo libdbus-1-dev
cargo fetch --manifest-path=tools/wsbrd_cli/Cargo.toml
// Generate the build script
cmake -G Ninja .
//Compile project
Ninja
//Install project
sudo ninja install
```

Step 3: Execute the following command to copy the example configuration file from the */usr/local/share/doc/wsbrd/examples* directory of the wisun-br-linux project source code to */wisun-br-linux/* directory.

```
//Copy the example configuration file to the directory and replace the existing default configuration.
cp -r /usr/local/share/doc/wsbrd/examples ~/wisun-br-linux/
```

Step 4: Execute the following command to check if the wisun-br-linux project has been successfully installed.

```
//Query the help information
wsbrd -h
```

If the wisun-br-linux project is installed successfully, executing **wsbrd -h** returns the following information in the MobaXterm tool terminal window:

```
ubuntu@raspi:~/wisun-br-linux$ wsbrd -h
Silicon Labs Wi-SUN border router v2.2

Start Wi-SUN border router

Usage:
  wsbrd [OPTIONS]
  wsbrd [OPTIONS] --list-rf-configs

Common options:
  -u UART_DEVICE        Use UART bus
  -t TUN                Map a specific TUN device (eg. allocated with 'ip tuntap add tun0')
  -T, --trace=TAG[,TAG] Enable traces marked with TAG. Valid tags: bus, cpc, hif, hif-extra
                          tun, timers, trickle, 15.4-mngt, 15.4, eap, icmp, dhcp, rpl, neigh,
                          drop, queue
  -F, --config=FILE     Read parameters from FILE. Command line options always have priority
                          on config file
  -o, --opt=PARM=VAL    Assign VAL to the parameter PARM. PARM can be any parameter accepted
                          in the config file
  -D, --delete-storage  Delete storage upon start, which deauhenticates any previously
                          connected nodes. Useful for testing.
                          Setting this option twice (-DD) deletes the storage then exits.
  -v, --version         Print version and exit

Wi-SUN related options:
  -l, --list-rf-configs Retrieve the possible RF configurations from the RCP then exit. Most
                          of parameters are ignored in this mode
  -n, --network=NAME    Set Wi-SUN network name
  -d, --domain=COUNTRY  Set Wi-SUN regulatory domain. Valid values: WW, EU, NA, JP...
  -m, --mode=VAL        Set operating mode. Valid values: 1a, 1b (default), 2a, 2b, 3, 4a,
                          4b and 5
  -c, --class=VAL       Set operating class. Valid values: 1 (default), 2, 3 or 4
  -S, --size=SIZE       Optimize network timings considering the number of expected nodes on
                          the network. Valid values: S (< 100, default), M (100-1000),
                          L (> 1000)

Wi-SUN network authentication:
  The following option are mandatory. Every option has to specify a file in PEM od DER
  format.
  -K, --key=FILE         Private key (keep it secret)
  -C, --certificate=FILE Certificate for the key
  -A, --authority=FILE   Certificate of the authority (CA) (shared with all devices of the
                           network)

Debug:
  --capture=FILE        Record raw data received on UART and network interfaces, and save it
                          to FILE. Also record timer ticks, and use a predicable RNG for
                          replay using wsbrd-fuzz.

Examples:
  wsbrd -u /dev/ttyUSB0 -n Wi-SUN -d EU -C cert.pem -A ca.pem -K key.pem
```

Figure 39: wsbrd -h Output

2.4. Calibrate Frequency

Tolerances in the electronic components and PCB materials of the KCM0A5S module can lead to an RF signal offset in the module. Therefore, it is necessary to calibrate the HFXO output frequency of the module. The HFXO clock integrated in the chip is the source of the output frequency. By adjusting the CTune (crystal oscillator load capacitor tuning) value, you can adjust the HFXO output frequency, which in turn adjusts the output frequency at the antenna interface.

Since this example uses Development Board A and Development Board B, each equipped with a KCM0A5S module, both modules need to be calibrated, and the calibration values must be written into the modules. For detailed calibration steps, please refer to https://community.silabs.com/s/article/hfxo-capacitor-bank-ctune-calibration-on-efr32.

After the calibration is completed, connect the KCM0A5S-TE-B to the SI-DBG1015A Simplicity Link debugger, then connect the debugger to the PC, and follow the steps below to write the calibration values:

Step 1: Launch Simplicity Studio, click "**Tools**" on the Simplicity Studio toolbar, select the "**Simplicity Commander**" tool, and then click "**OK**" to open the selected tool.

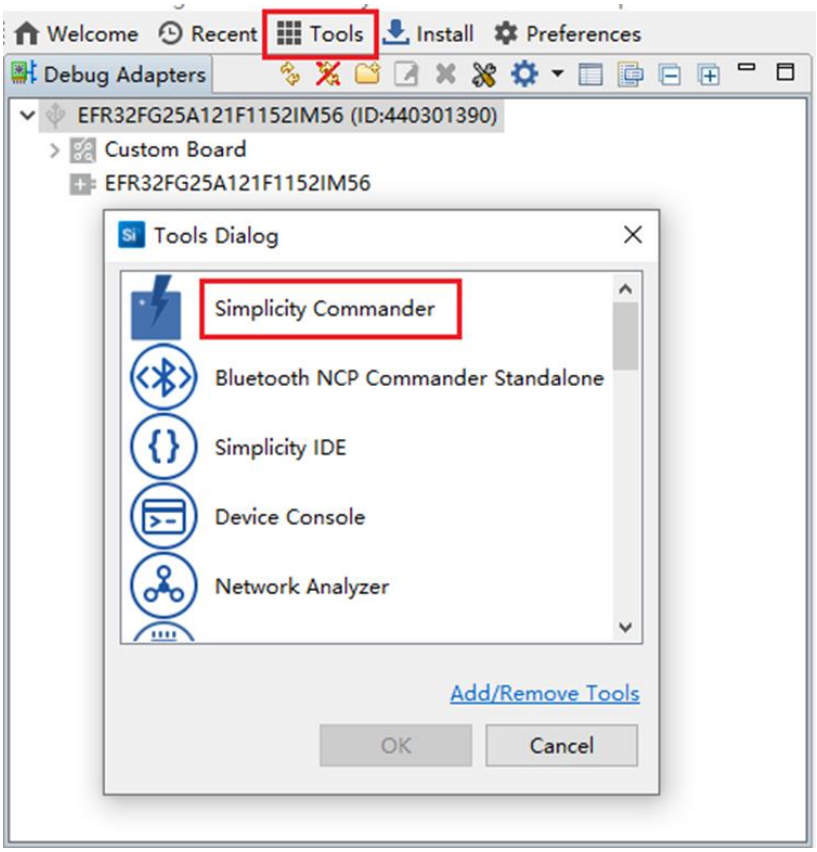


Figure 40: Select Simplicity Commander Tool

Step 2: Click "**Open Shell**" to open the shell console.

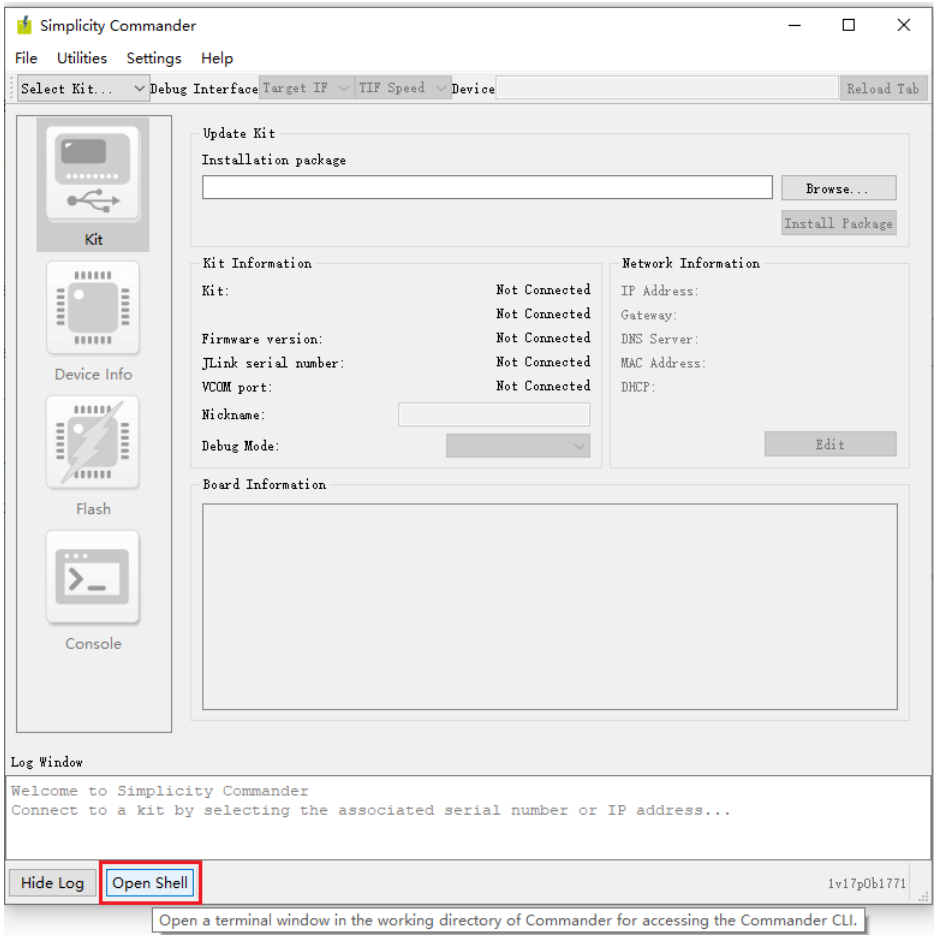


Figure 41: Open Shell Console

Step 3: In the Shell console, enter the following commands to read and write the CTune value from/to the chip.

```
//Read CTune value
commander ctune get --device EFR32FG25A121F1152IM56
//Set the CTune value. After measurement, it is recommended to set CTune value as 100.
commander ctune set --value 100 --device EFR32FG25A121F1152IM56
```

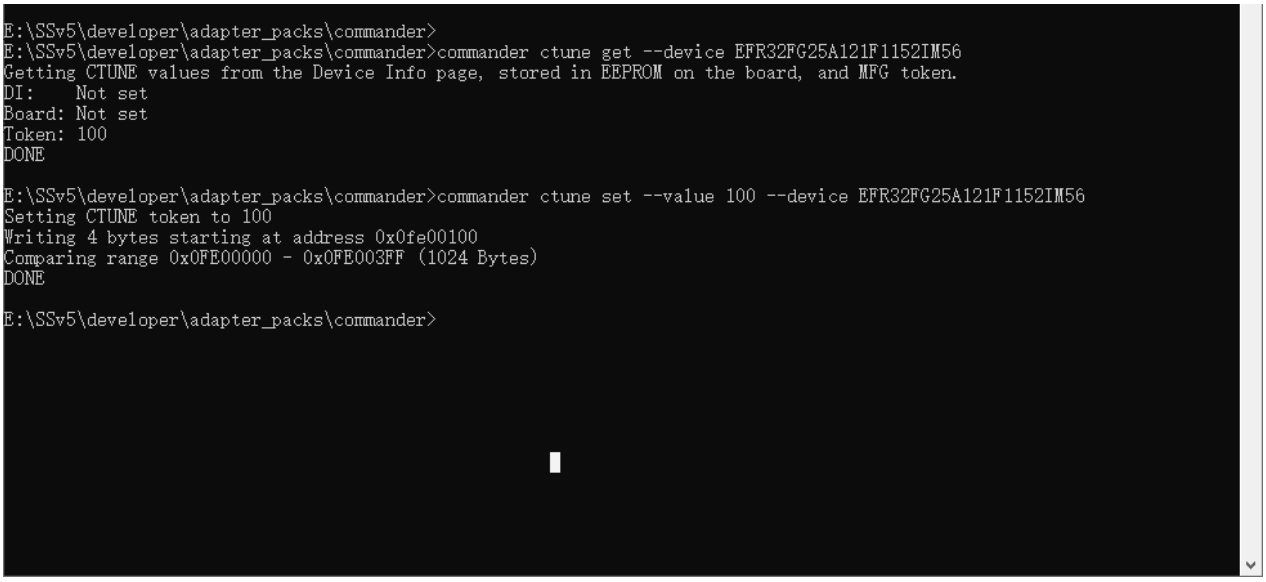


Figure 42: Command Execution Result

2.5. View Relevant Information

Open the Simplicity Studio tool on your PC, and then select "**Launcher**" in the upper-right corner of the main interface to switch to the Launcher perspective. In the "Debug Adapters" section, select the connected SI-DBG1015A Simplicity Link debugger. The device information will be displayed on the right pane. Select the "**EXAMPLE PROJECTS & DEMOS**" tab, check the "**Wi-SUN**" checkbox under the "**Wireless Technology**" category to view the Wi-SUN demos.

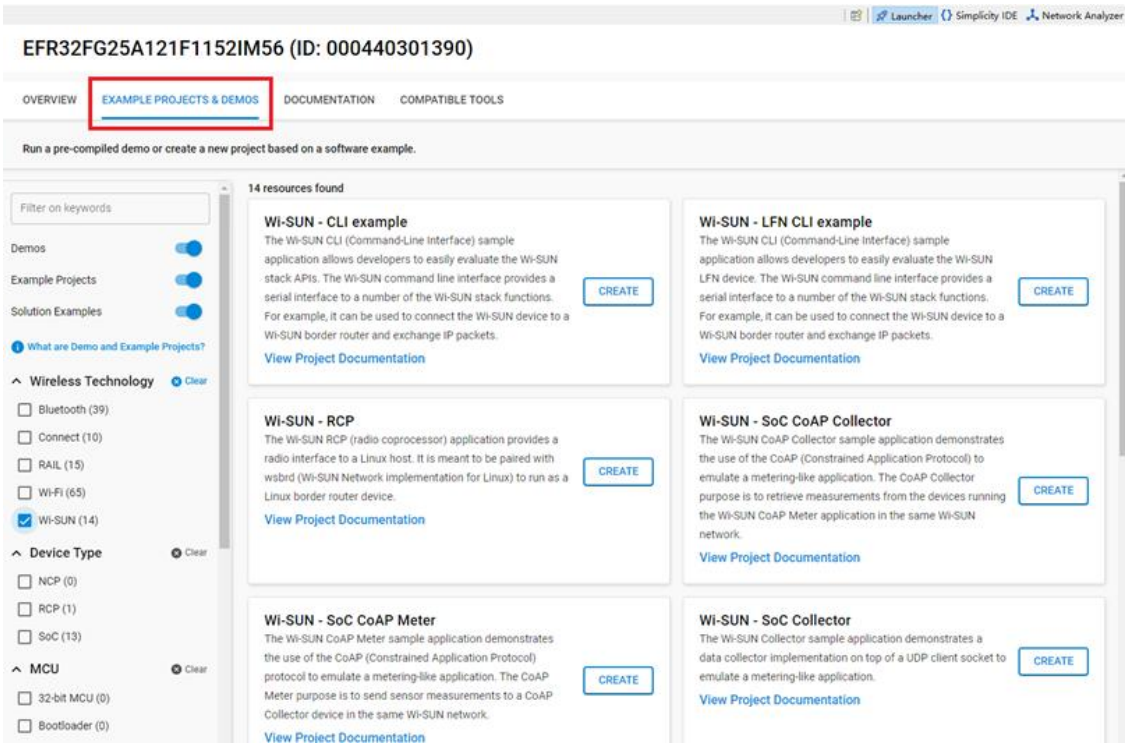


Figure 43: Wi-SUN Demo

On the "**Documentation**" tab, you can view the relevant documents. Additionally, you can visit https://docs.silabs.com/wisun/2.2.0/wisun-start to view detailed information on Wi-SUN development.

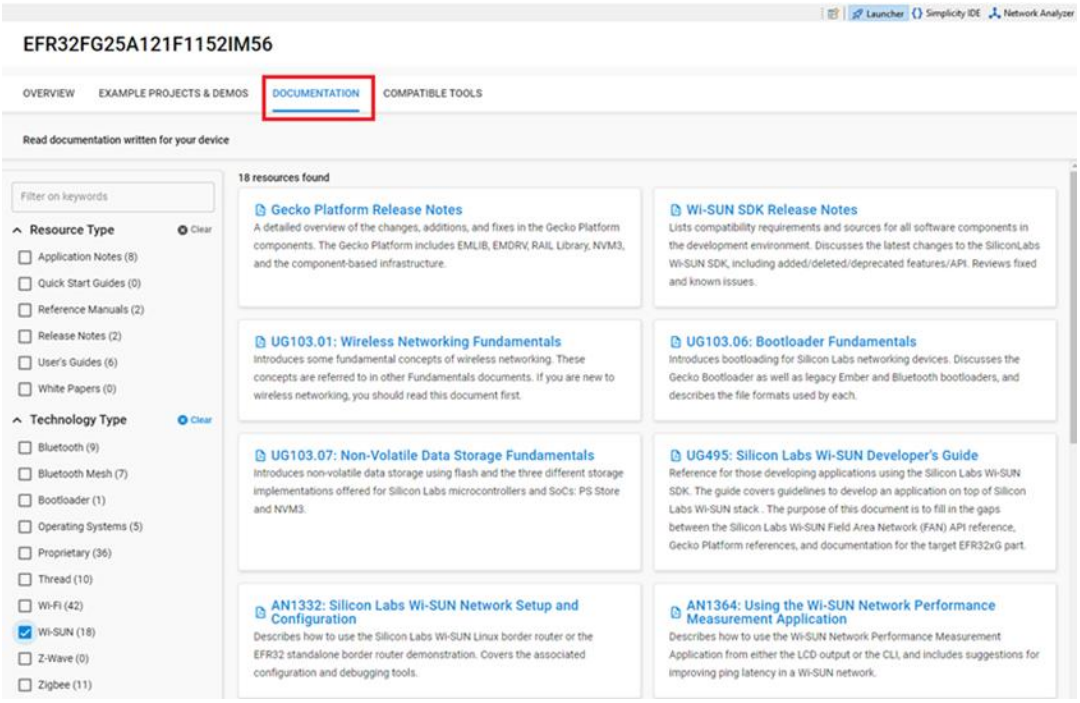


Figure 44: Documentation

3 Develop Application

This chapter introduces the application development process, covering the compilation and flashing of the border router RCP firmware and node CLI firmware, as well as the networking and debugging of both the border router and node.

3.1. RCP Application

The entire border router solution consists of the Raspberry Pi 4 Model B and Development Board A. The wisun-br-linux project is installed on the Raspberry Pi 4 Model B (see ***Chapter 2.3.3***), while the RCP application runs on Development Board A. This chapter describes the RCP application development process, including the creation, modification, compilation, flashing, and debugging of the RCP project.

3.1.1. Create a Project

3.1.1.1. Project Creation Process

Step 1: Connect the Development Board A to the SI-DBG1015A Simplicity Link debugger, then connect the debugger to the PC.

Step 2: In the "Debug Adapters" section of the Simplicity Studio tool's main interface, find and right-click the debugger and select "**Device configuration**". The debugger configuration window opens on the "**Device hardware**" tab. Enter and select the chip model "**EFR32FG25A121F1152IM56**" corresponding to the module in the "**Target part**" text box. Enter and select the development boards. Then click "**OK**".

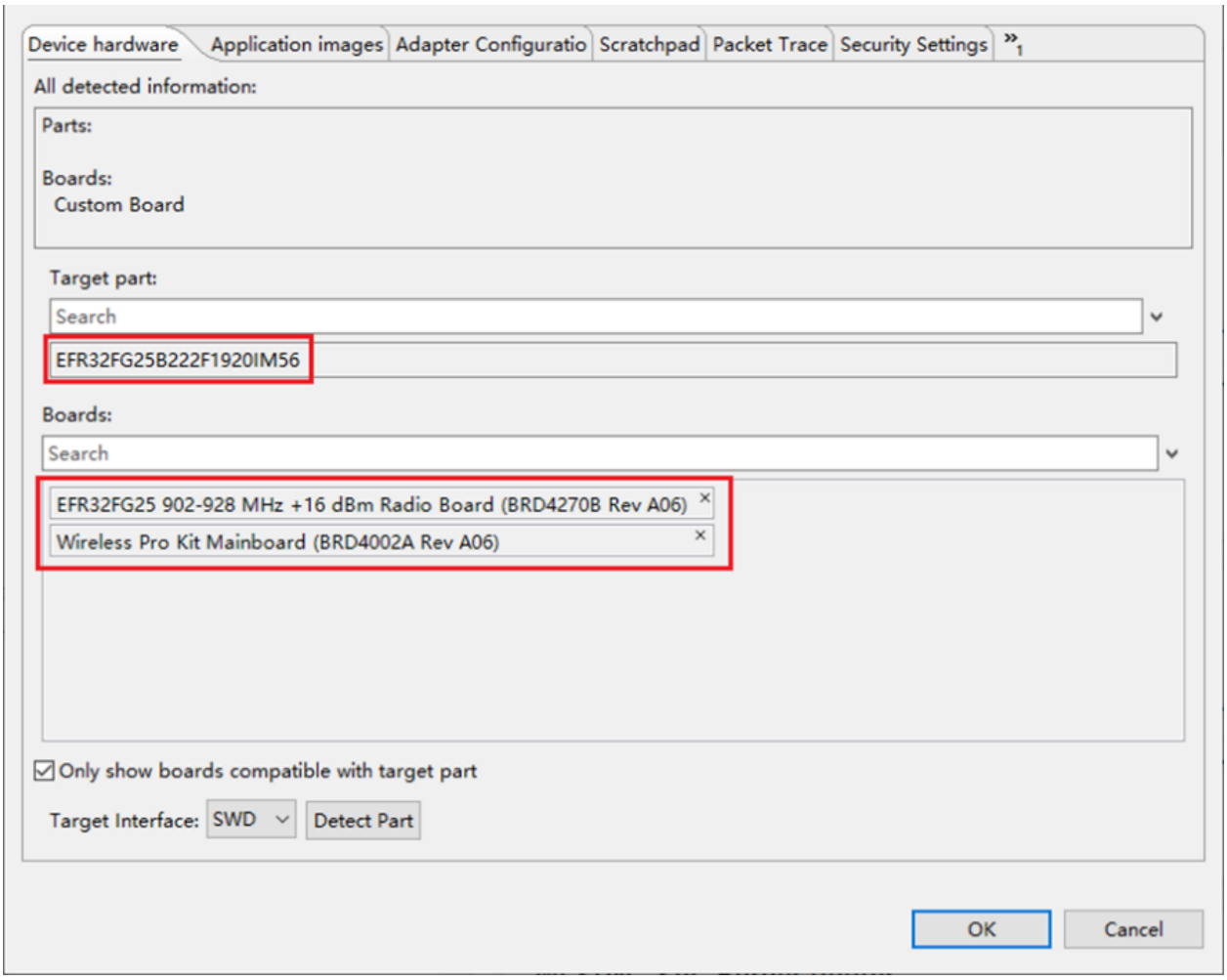


Figure 45: Set Chip Model and Development Board Model

Step 3: Click "**Launcher**" in the upper-right corner of the Simplicity Studio tool's main interface to switch to the Launcher perspective. In the "Debug Adapters" section, select the debugger. Select the "**EXAMPLE PROJECTS & DEMOS**" tab, check the "**Wi-SUN**" checkbox under the "Wireless Technology " category, select the "Wi-SUN – RCP" demo template from the results, and click "**CREATE**" to open the "Project Configuration" interface.

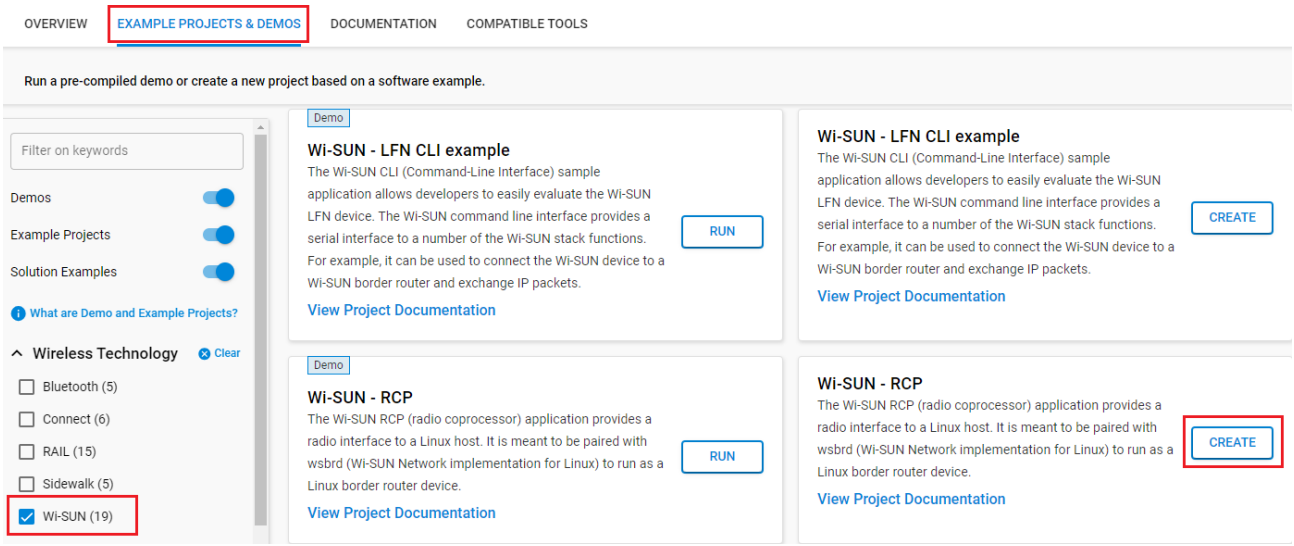


Figure 46: Create RCP Project

Step 4: Set the project name (e.g., wisun_rcp) and storage location. In the "With project files" section, select the project linking method (it is recommended to choose "**Copy contents**", which copies the files from the SDK directly to the project directory without relying on file links). Then click "**FINISH**" to complete the project creation.

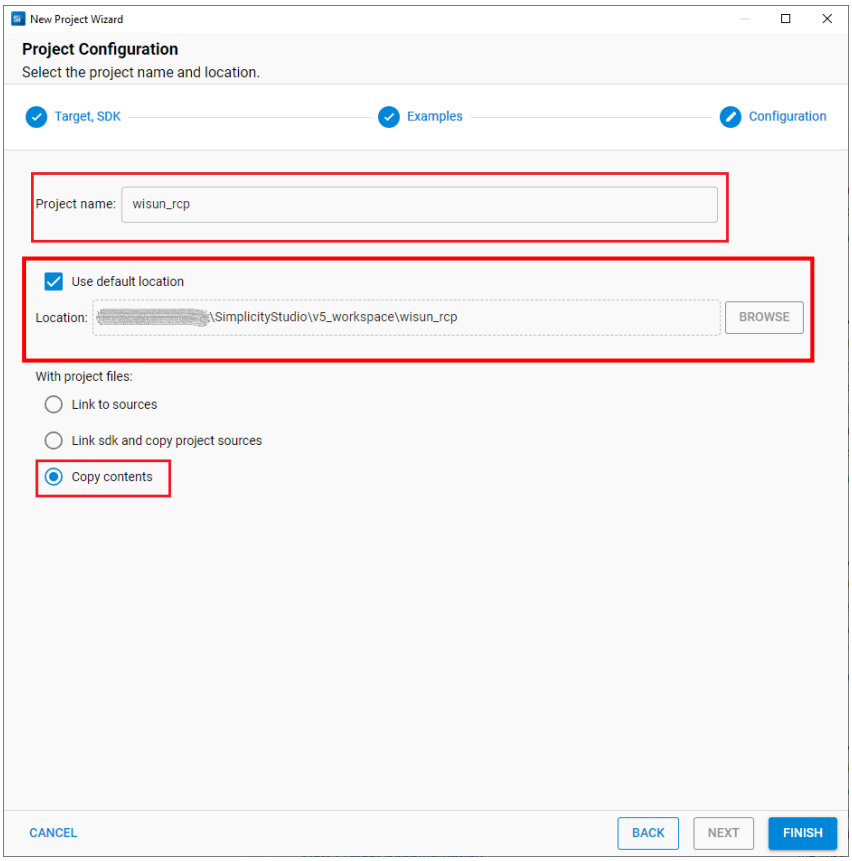


Figure 47: Set Project Name and Location

The created wisun_rcp project directory is displayed in the "Project Explorer" section of the tool interface, and it contains the following files and folders:

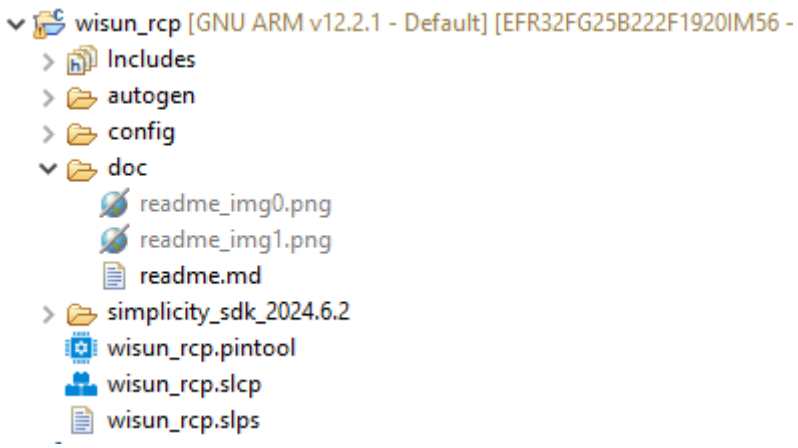


Figure 48: wisun_rcp Project Directory

3.1.1.2. Project Directory Structure

A software project typically contains the following files and folders:

Table 4: Project Directory Structure

Name	Description
autogen	Stores the files generated automatically in Simplicity Studio.
config	Stores project configuration files.
simplicity_sdk_xxx	Stores the SDK of chip manufacturer (for example, *simplicity_sdk_2024.6.2*).
GNU ARM v12.2.1 - Default	Stores the files, such as *.hex* file, generated after compilation (This folder is automatically generated during the project compilation).
xxx.pintool	Used for configuring pins.
xxx.slcp	Used for installing and configuring components.
readme.md	Documentation for the examples.

NOTE

Project directory contents may vary according to software projects. This table lists only the files and folders that are included in all software project directories.

3.1.2. Modify Project Configuration

Step 1: Modify the development board model and target chip model.

1) In the wisun_rcp project directory, find and double-click the *wisun_rcp.slcp* file to open the Project Configurator. From the "**OVERVIEW**" tab, click "**Change Target/SDK/Generators**".
2) Modify the development board model to "**Custom Board**".
3) Modify the target chip to "EFR32FG25A121F1152IM56".
4) Then click "**Save**" to save the configurations.

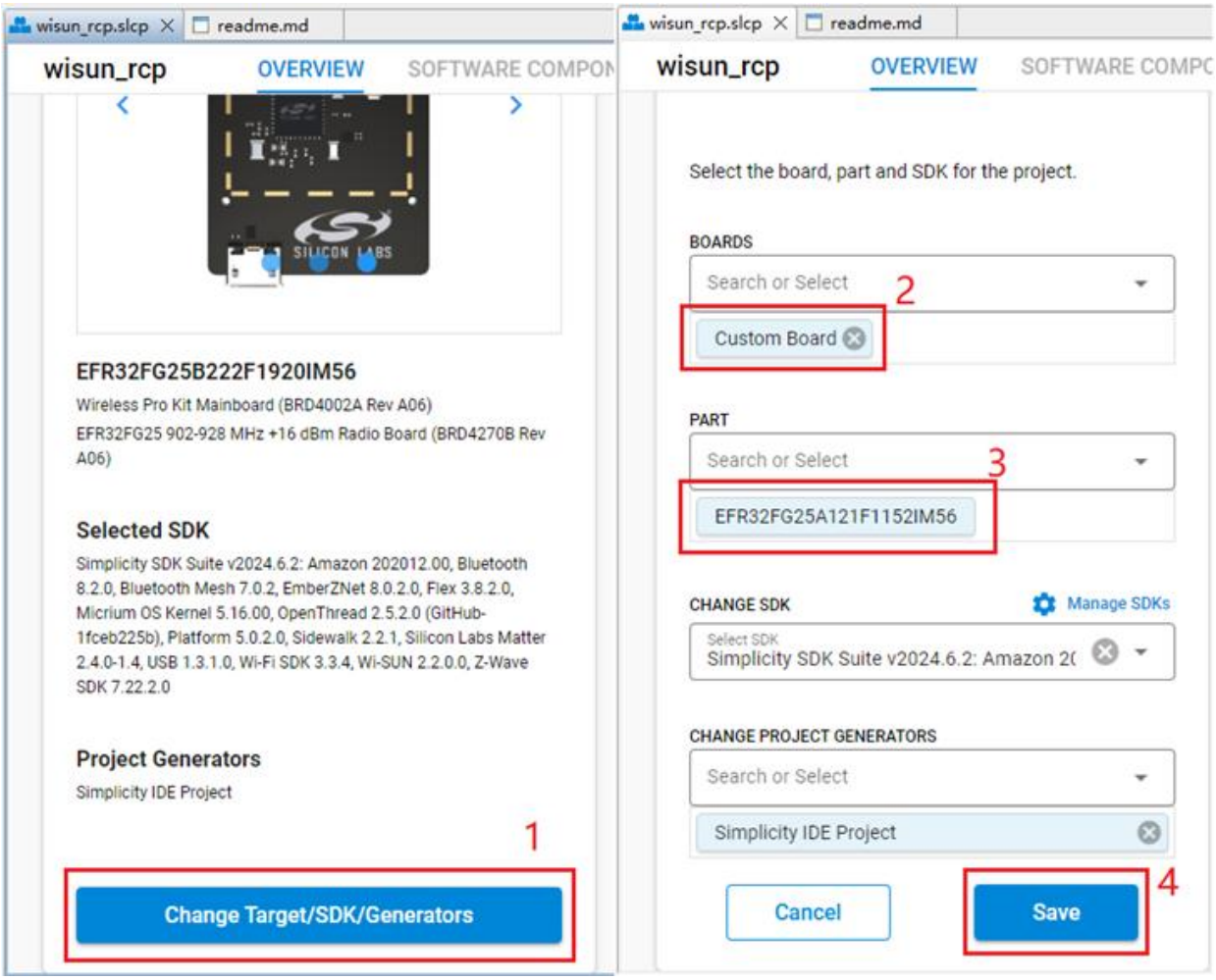


Figure 49: Modify Development Board Model and Chip Model

Step 2: Modify the UART configuration.

1) Select the "**SOFTWARE COMPONENTS**" tab, and check the "**Installed**" checkbox.
2) Expand "**Wi-SUN**" in the menu on the left, select "**WI-SUN Border Router RCP-UART interface**", then click "**Configure**".
3) In the "WI-SUN Border Router RCP-UART interface" configuration interface, click "**</> View Source**" in the upper-right corner.

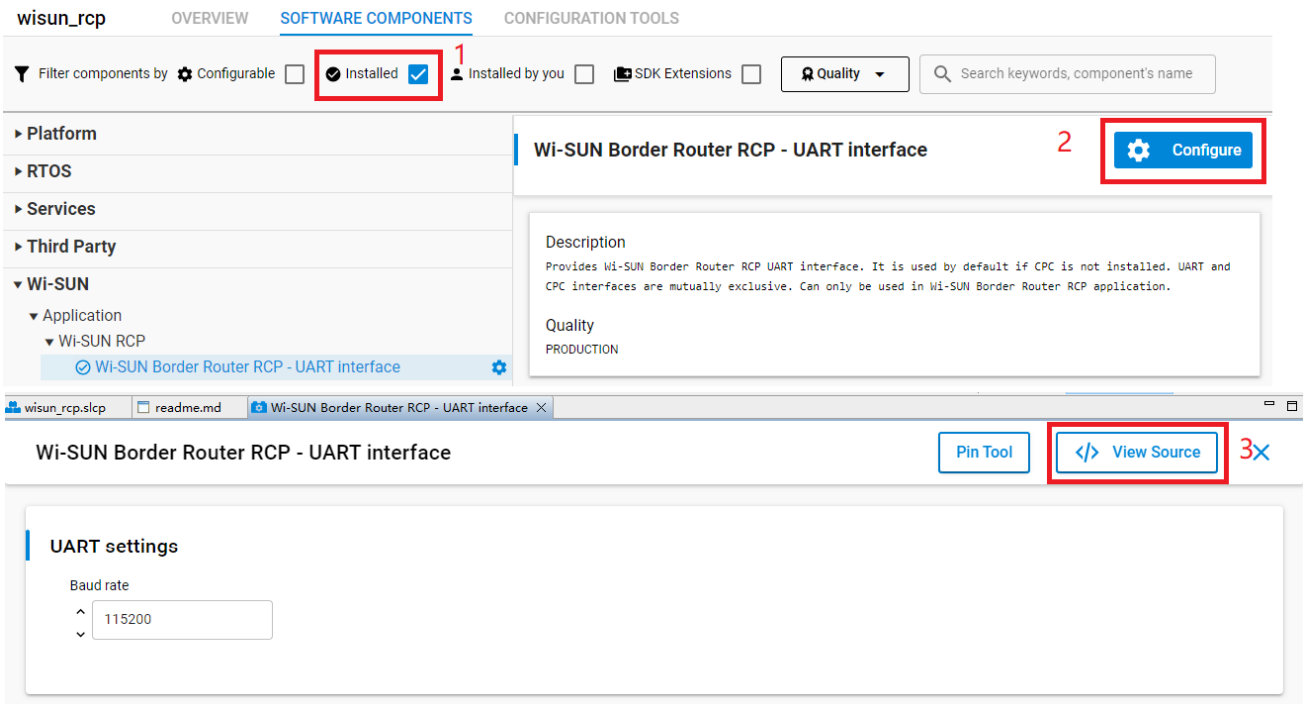


Figure 50: UART Configuration

4) Configure the UART TX and RX pins as follows:

```
// <i> Default: 115200
#define UART_BAUDRATE           115200
// </h>

// <<< end of configuration section >>>
#include <em_eusart.h>
#define UART_PERIPHERAL_VAL     &sl_peripheral_val_eusart0
#define UART_PERIPHERAL         EUSART0
#define UART_RX_IRQ             EUSART0_RX_IRQn
#define UART_LDMA_SIGNAL_RX     ldmaPeripheralSignal_EUSART0_RXFL
#define UART_LDMA_SIGNAL_TX     ldmaPeripheralSignal_EUSART0_TXFL

#define UART_PORT_TX            gpioPortB
#define UART_PIN_TX             0

#define UART_PORT_RX            gpioPortB
#define UART_PIN_RX             1
```

Figure 51: RCP UART PIN Configuration

Step 3: Configure FEM.

The module integrates an FEM (Front-End Module) chip to amplify Tx and Rx signals. The FEM chip needs to be configured accordingly. The CSD (control shutdown), CTX (transmit mode select input), and CPS (bypass mode select input) pins control the operating mode of the FEM chip, as shown in the table below:

Table 5: FEM Control Logic

CSD (PB03)	CTX (PB05)	CPS (PB04)	Operating Mode
1	1	1	Transmit mode
1	0	X	Receive LNA mode
1	1	0	Receive bypass mode
0	X	X	Shutdown mode

NOTE

1. "1" indicates that the control pin is at a high level; "0" indicates that the control pin is at a low level; "X" indicates that it can be 0 or 1.
2. PB03, PB04, and PB05 are the main chip pins. For details about these pins, please consult Quectel Technical Support.

1) Cancel the PB05 pin configuration.

Since this project uses the PB05 pin to control the on/off function, which conflicts with the FEM control pin, the PB5 pin configuration in the project needs to be canceled. In the wisun_rcp project directory, find and double-click the *wisun_rcp.pintool* file to open the pin configuration tool and see the project's pin configuration. In the table on the right side of the interface, find PB05, and click "**GPIO mode**". In the "Edit Pin: PB05 (#23)" window, click "**x**" to clear the GPIO mode configuration, and then click "**APPLY AND CLOSE**".

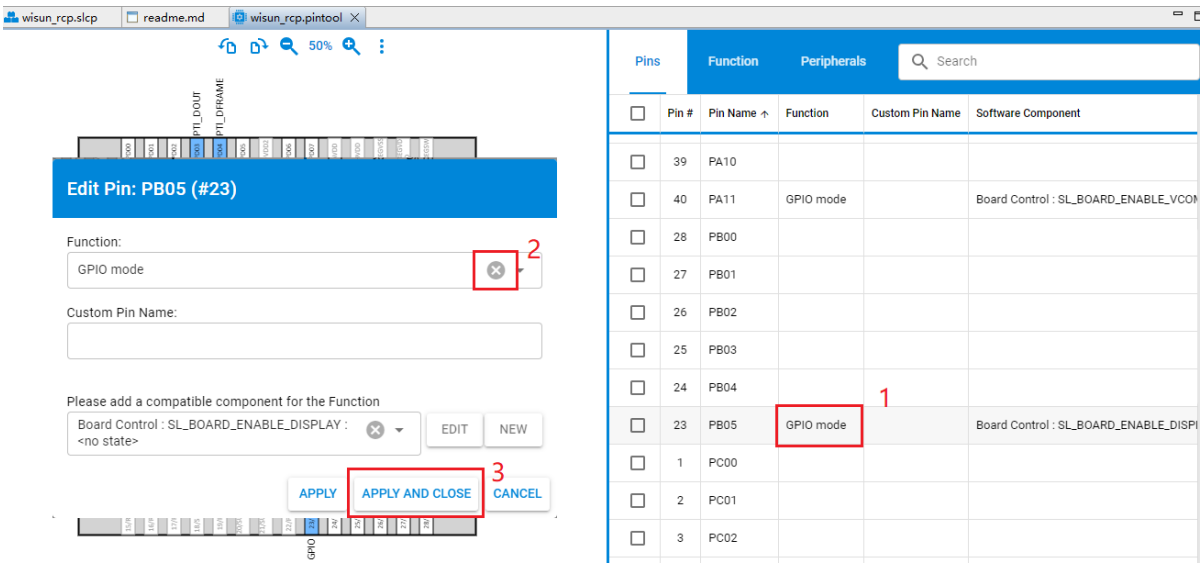


Figure 52: Cancel PB05 Configuration

2) Close the pin configuration tool and click "**Save**" to save the configuration.

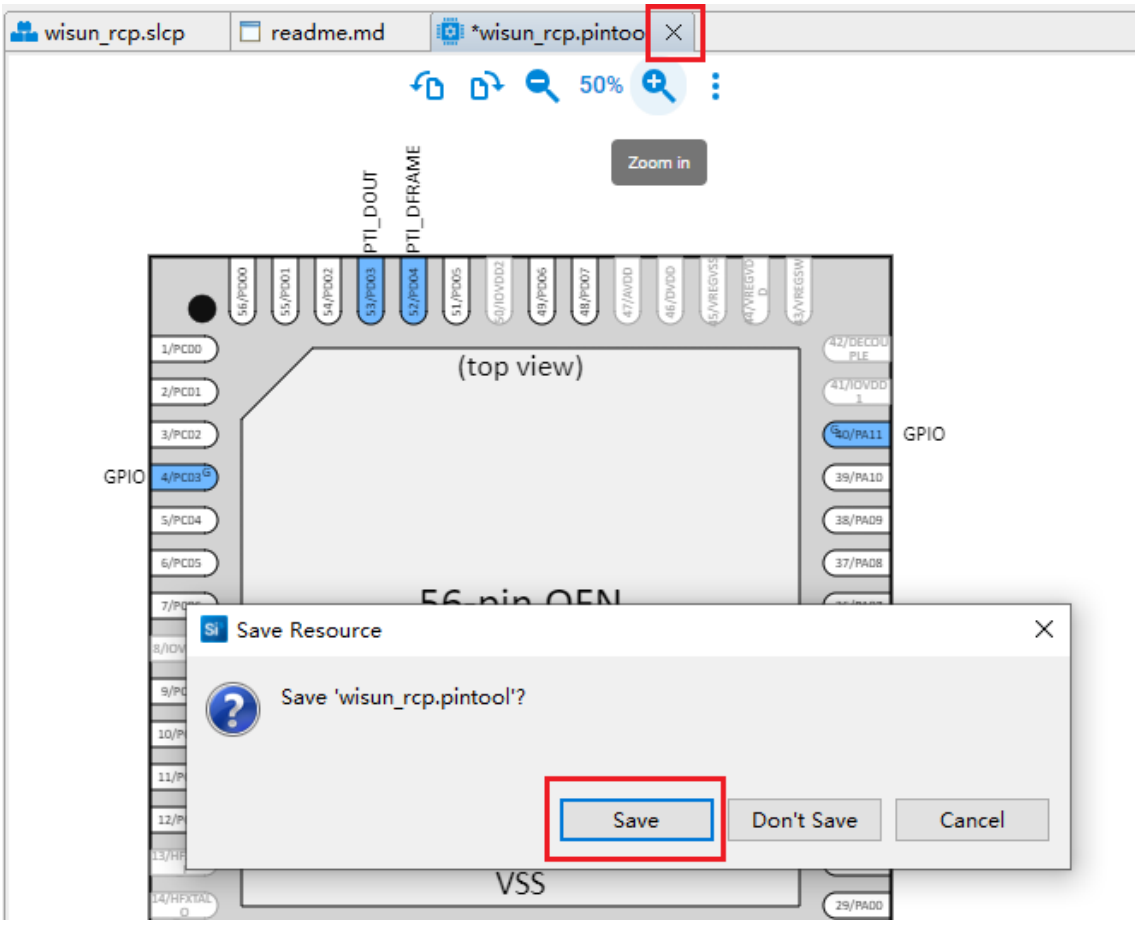


Figure 53: Save Pin Configuration

3) Install FEM component.

Select the "**SOFTWARE COMPONENTS**" tab, enter "fem" in the search box, select "**Radio Utility, FEM**" from the search results, and click "**Install**" to install the FEM component.

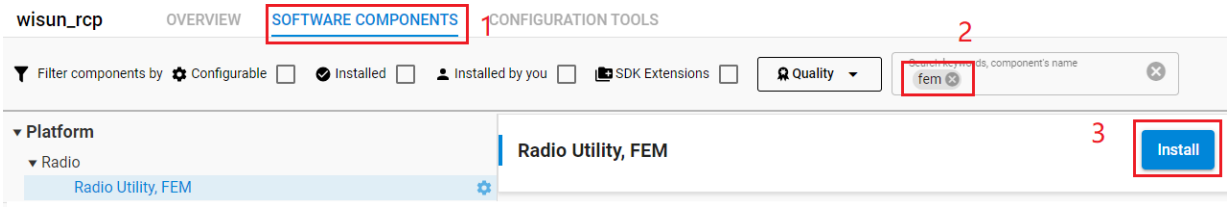
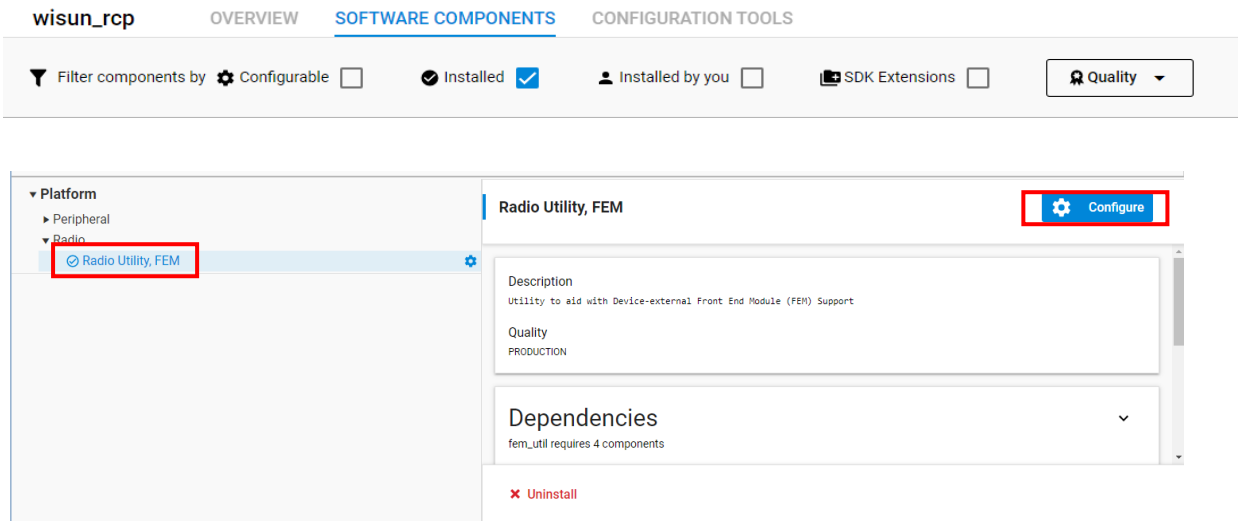


Figure 54: Install FEM Component

4) Configure FEM component.

On the "**SOFTWARE COMPONENTS**" tab, check the "**Installed**" checkbox, find and select "**Radio Utility, FEM**" and click "**Configure**" to open the FEM component configuration interface. Select "**Enable TX Mode**" and configure the TX control pins in the "SL_FEM_UTIL_TX" section as shown below.



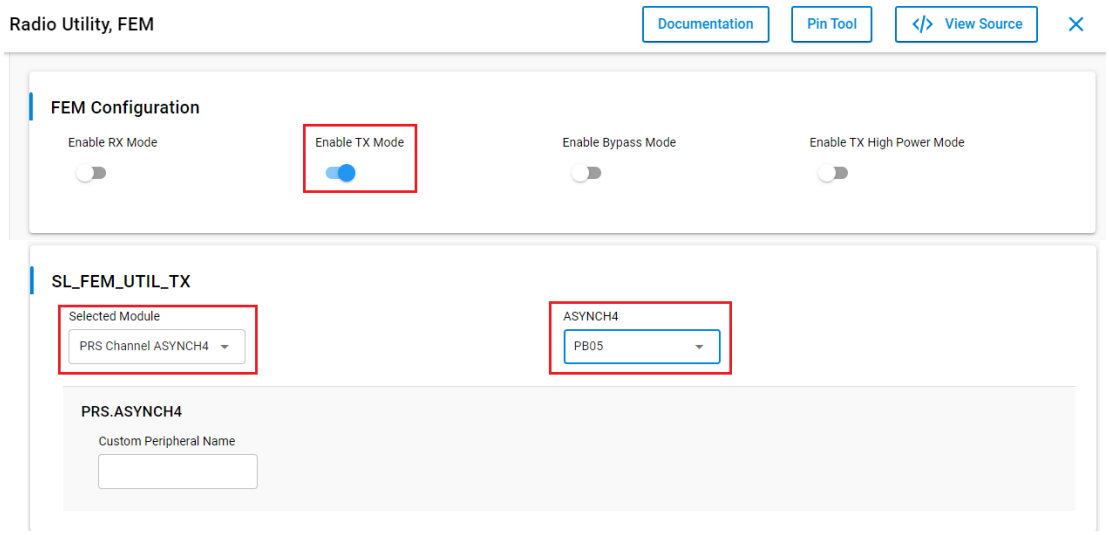


Figure 55: Configure FEM Component

5) Install GPIO initialization component.

On the "**SOFTWARE COMPONENTS**" tab, enter "gpio" in the search box, select "**GPIO Init**" from the search results, and then click "**Install**" to install the GPIO component.

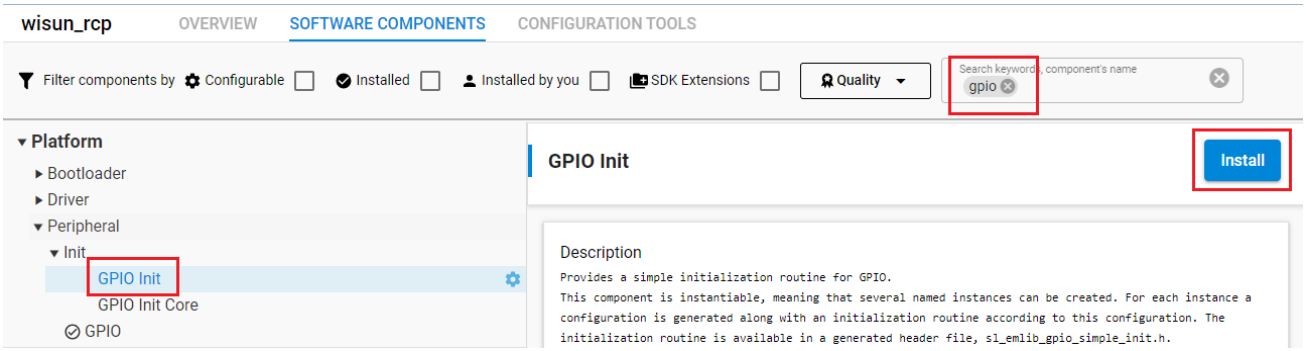


Figure 56: Install GPIO Component

In the "Create A Component Instance" window, enter the instance name (e.g., "fem_csd"), then click "**Done**" to complete the installation.

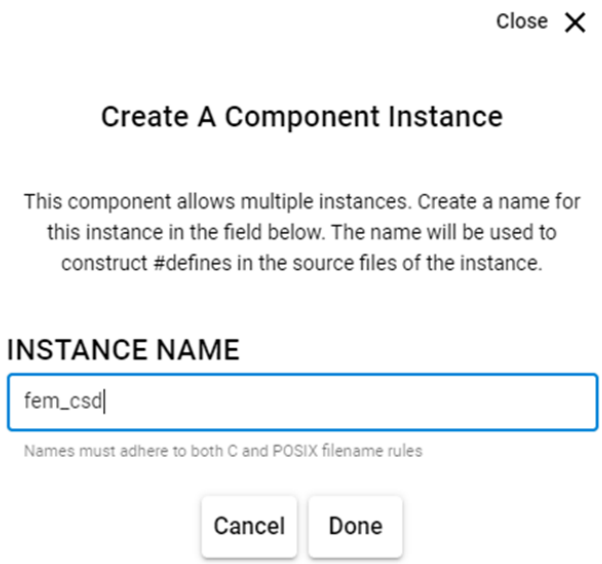


Figure 57: Create CSD Pin Instance

On the "**SOFTWARE COMPONENTS**" tab, check "**Installed**", find and select "**GPIO Init**", then click "**Configure**" to open the "GPIO Init (fem_csd)" configuration interface. Configure the CSD pin as shown below.

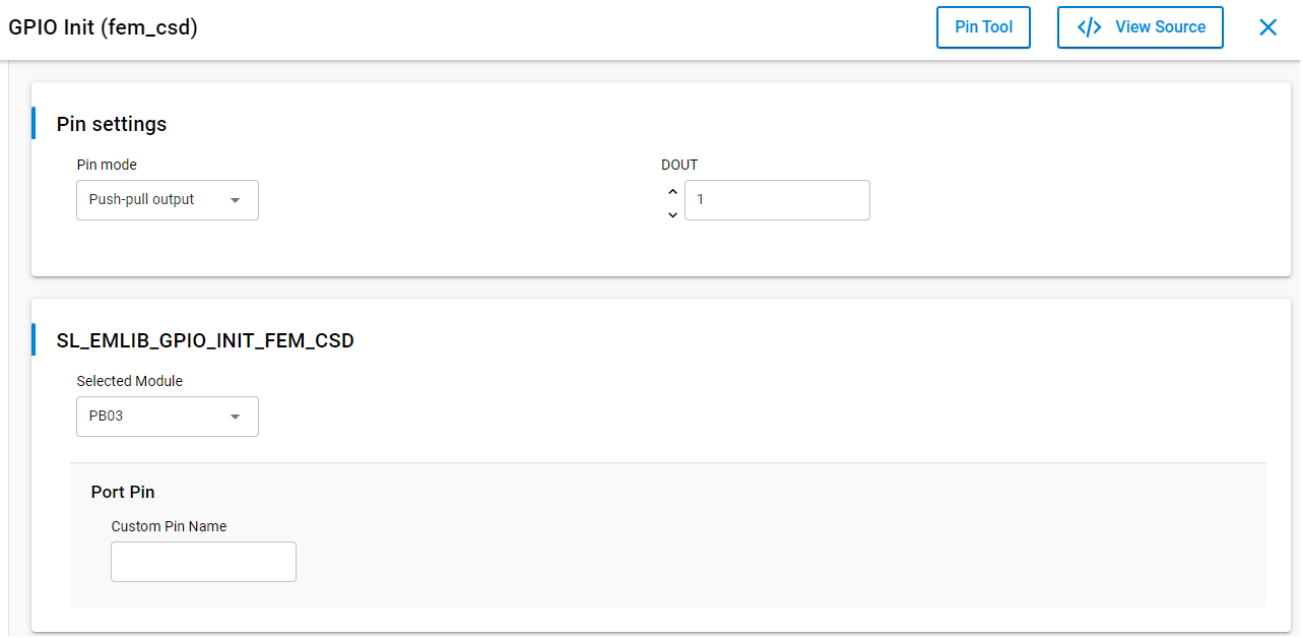


Figure 58: Configure CSD Pin

Follow the above step to install another GPIO component and configure the instance name (e.g., "fem_cps"). Then, configure the CPS pin as shown in ***Figure 61***.

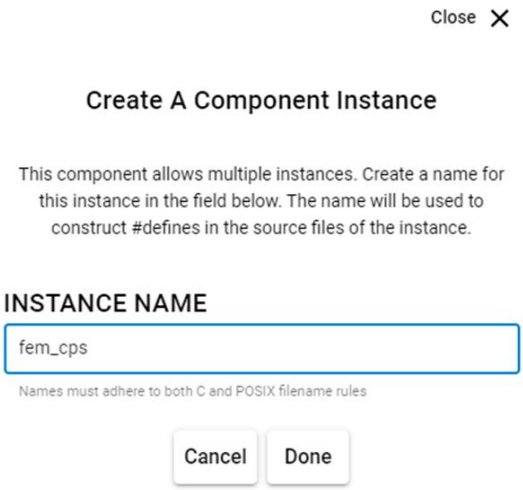


Figure 59: Create CPS Pin Instance

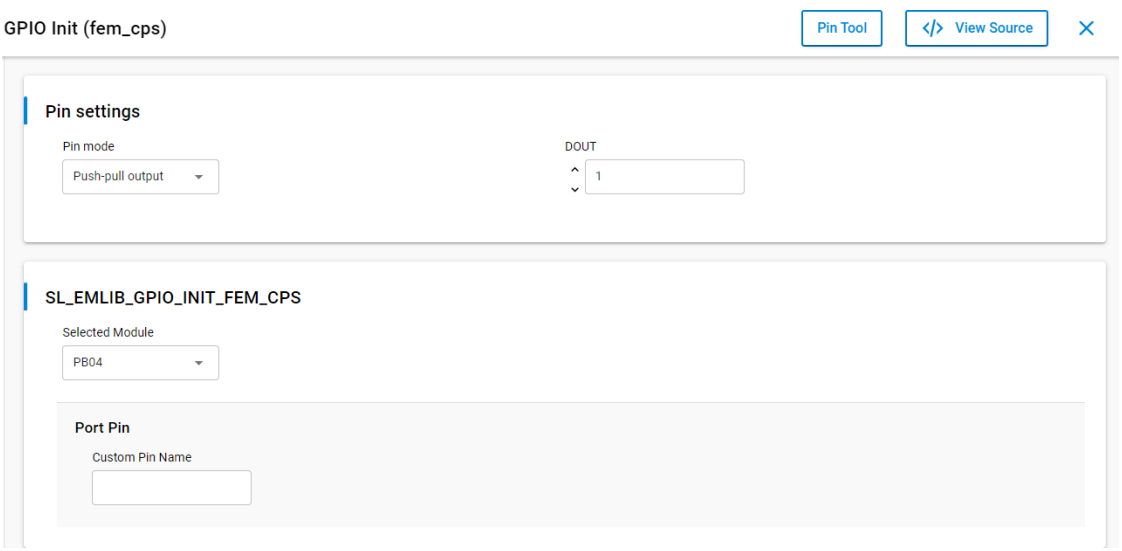


Figure 60: Configure CPS Pin

Step 4: Configure RSSI offset.

1) On the "**SOFTWARE COMPONENTS**" tab, enter "rssi" in the search box, select "**RAIL Utility, RSSI**" from the search results, and click "**Install**" to install the RSSI component.

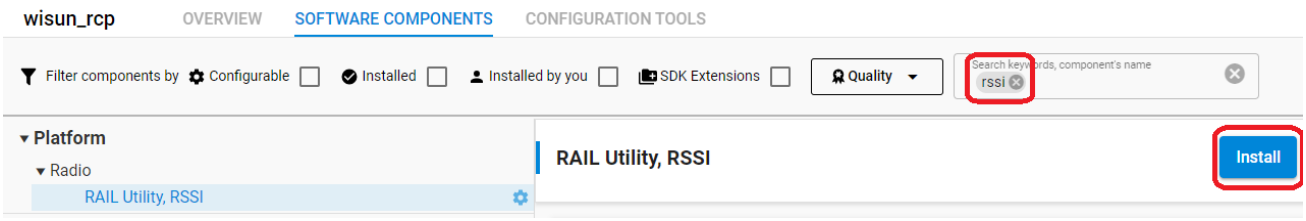


Figure 61: Install RSSI Component

2) On the "**SOFTWARE COMPONENTS**" tab, check the "**Installed**" checkbox, find and select "**RAIL Utility, RSSI**", then click "**Configure**" to open the "RAIL Utility, RSSI" configuration interface to set the offset value. After measurement, it is recommended to set the offset value to -22.

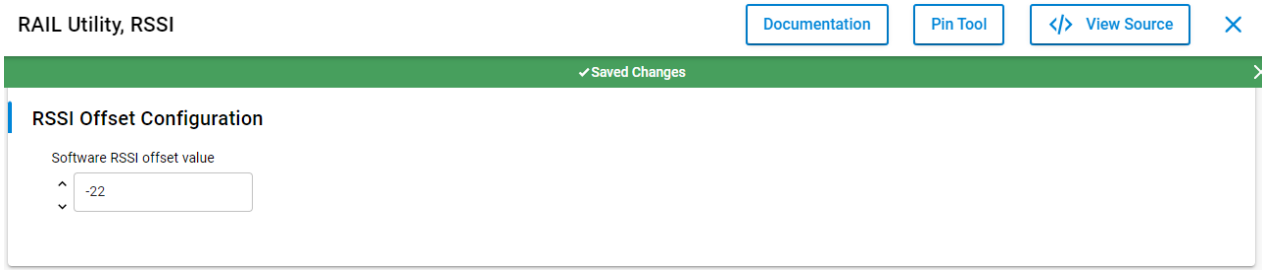


Figure 62: Set RSSI Offset Value

3.1.3. Compile Project

Step 1: Right-click the debugger in the "Debug Adapters" area of the tool's main interface and select "**Device configuration**" to open the debugger configuration interface. From the "**Device hardware**" tab, click "**Detect Part**", and then the tool automatically detects and displays the chip model as EFR32FG25A121F1152IM56. Then, click "**OK**".

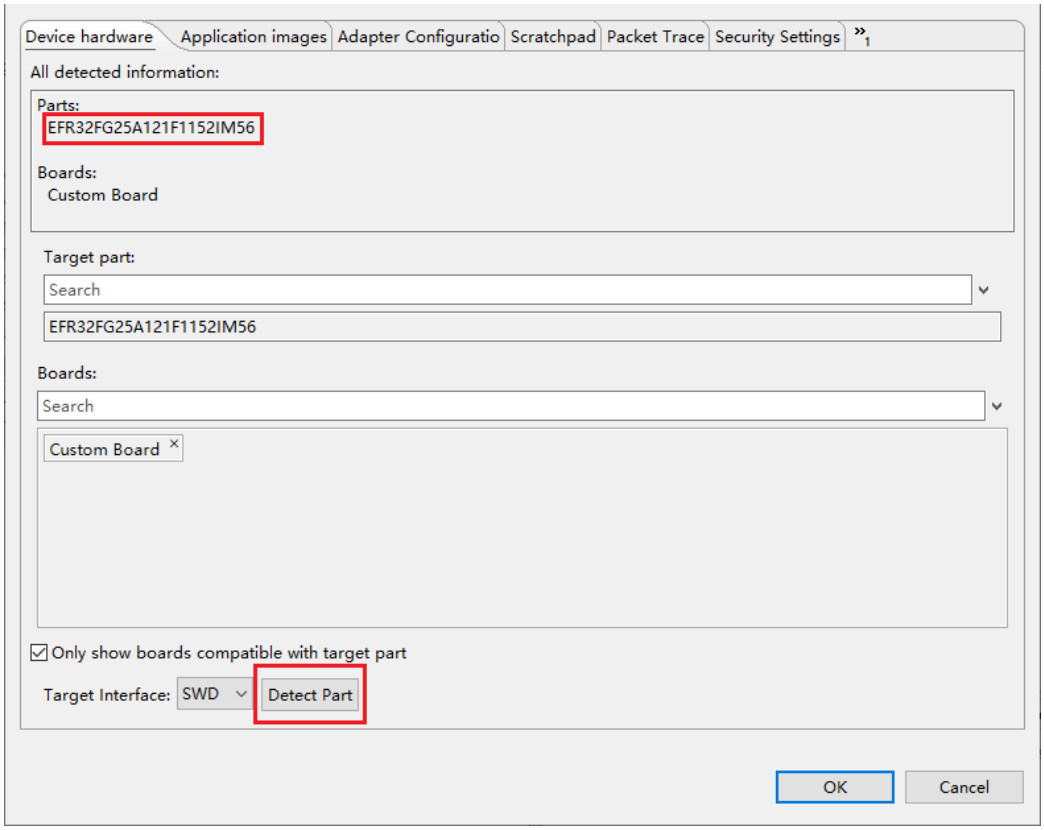


Figure 63: Detect Chip Model

Step 2: Right-click the project directory and click "**Build Project**" to compile the project.

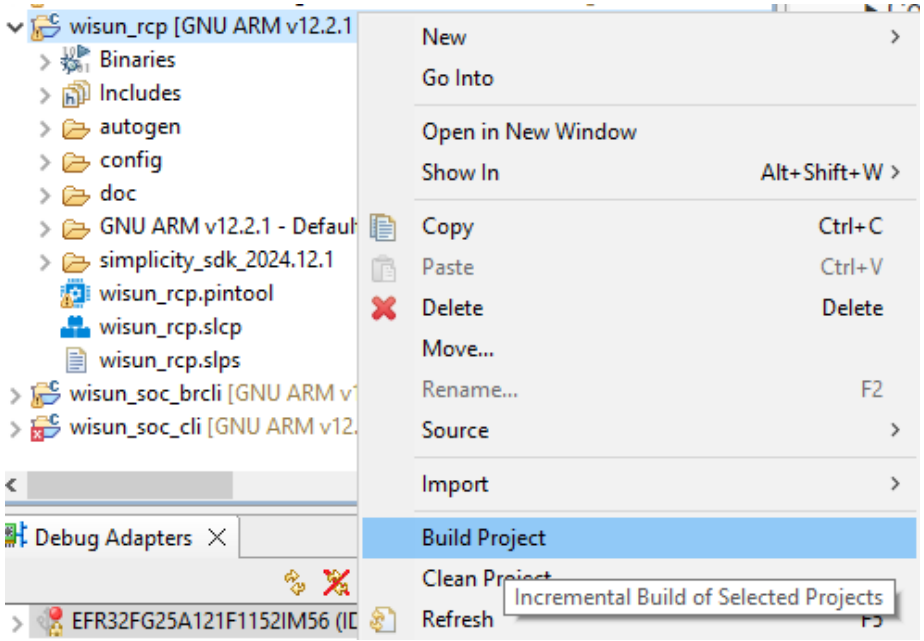


Figure 64: Compile Project

After the compilation is completed, the compilation output is displayed in the "Console" section at the bottom of the window.

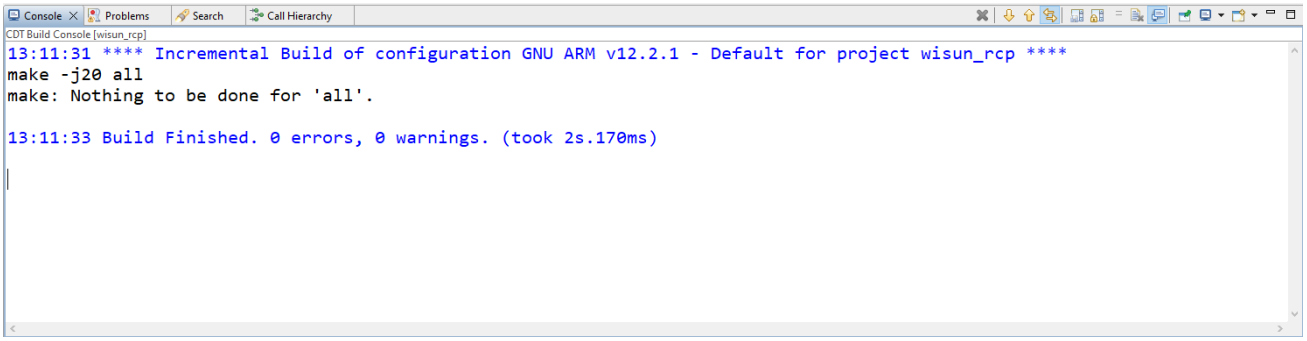


Figure 65: Compilation Output

A *GNU ARM v12.2.1 – Default* folder is automatically generated in the project directory and stores the following firmware files generated after compilation:

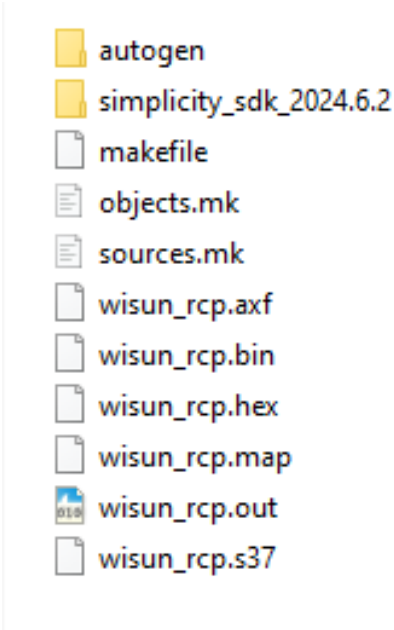


Figure 66: Firmware Storage Folder

3.1.4. Flash Firmware

This chapter explains how to flash the firmware. Before flashing the firmware, check if the SI-DBG1015A Simplicity Link debugger is connected correctly. If it is connected correctly, press the reset button on the Development Board A before flashing the firmware. The firmware is stored in the *GNU ARM v12.2.1 – Default* project folder, and can also be found in the *Binaries* project folder.

Step 1: Follow the ***Step 1*** in ***Chapter 3.1.3*** to open the debugger configuration interface and detect the chip model.

Step 2: Right-click any file in the *Binaries* folder and select "**Flash to Device…**" to open the "Flash Programmer" interface.

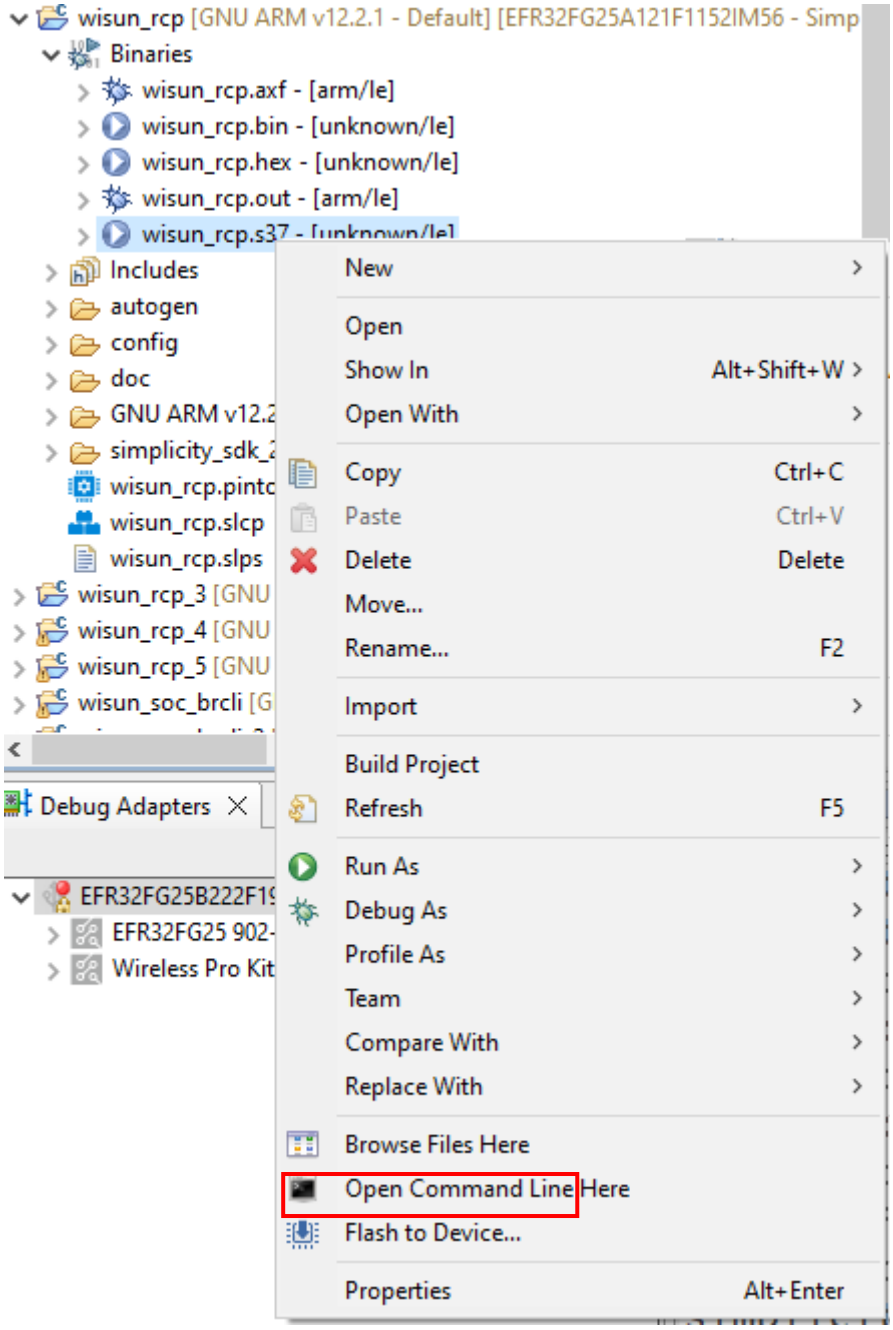


Figure 67: Open Flash Programmer

Step 3: Check if the development board model and chip model displayed in the "Device" section are correct. Click "**Browse...**" to select the file to be flashed (axf, bin, hex or s37 file in the *GNU ARM v12.2.1 – Default* folder), then click "**Program**" to start the flashing process.

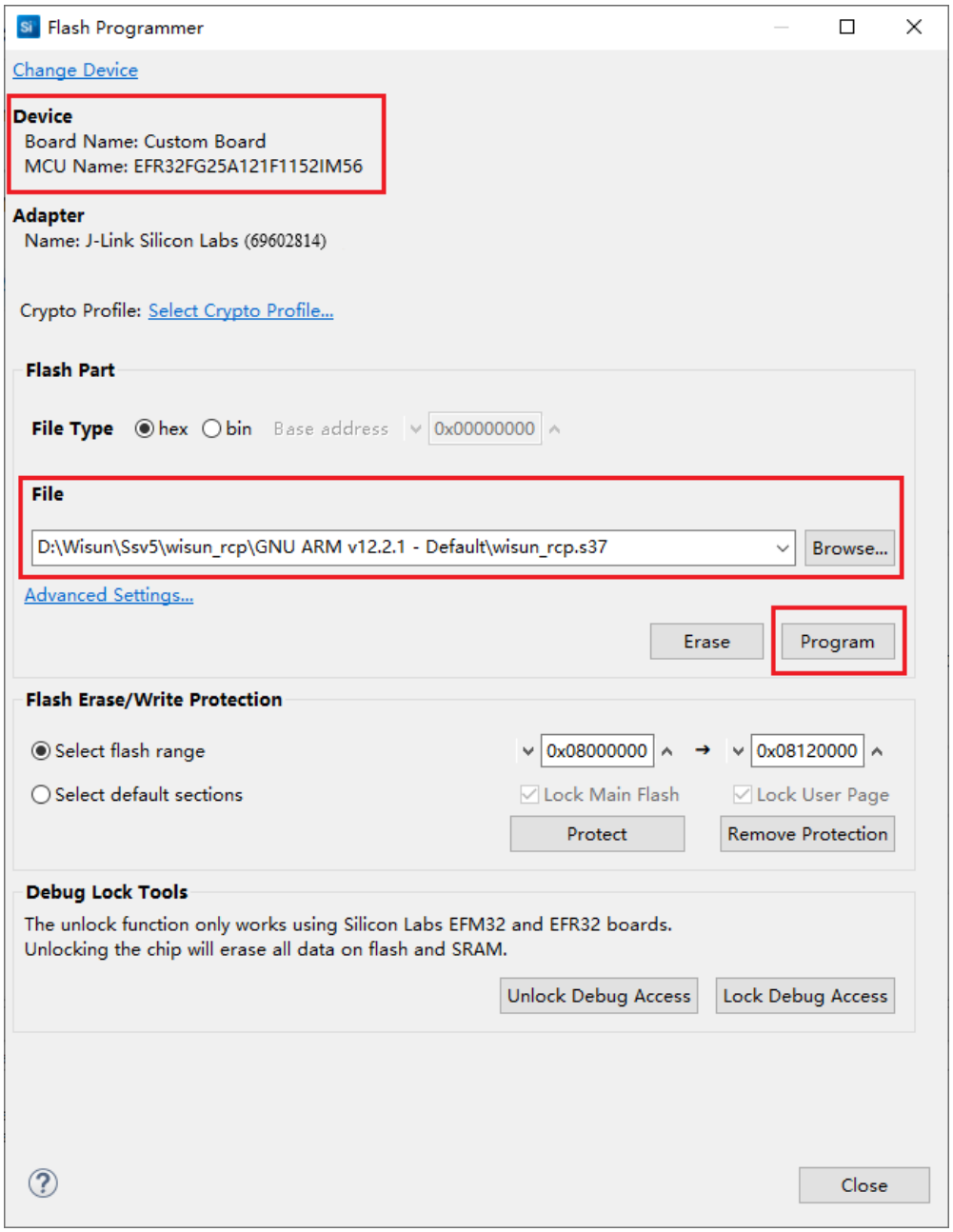


Figure 68: Flash Firmware

3.1.5. Debug Application

Step 1: Connect the Development Board A and the Raspberry Pi 4 Model B using a USB cable.

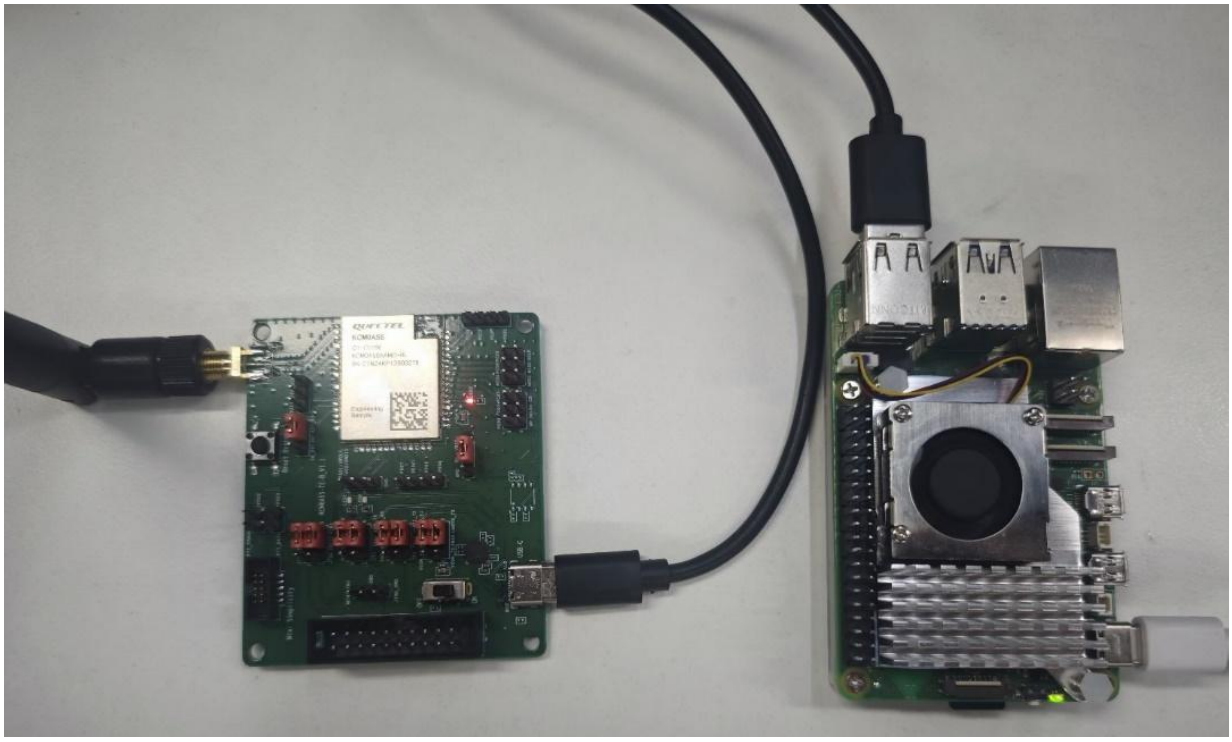


Figure 69: Connect Development Board A to Raspberry Pi 4 Model B

Step 2: Log in to the Raspberry Pi 4 Model B and modify the *~/wisun-br-linux/example/wsbrd.conf* configuration file.

The important parameters in the configuration file are shown in the table below:

Table 6: Important Parameters in Configuration File

Parameter	Description
uart_device	RCP UART device number
uart_baudrate	RCP UART baud rate. Default value: 115200
network_name	Network name
size	Network capacity
domain	Network domain number
chan_plan_id	Channel ID

phy_mode_id	PHY mode ID
tx_power	Transmission power. It is recommended to set this parameter to max. -10 dBm, especially when *chan_plan_id* is 5 and *phy_mode_id* is 38, as EVM may exceed the range and cause communication exceptions.
key	Device private key address
certificate	Device certificate address
authority	CA certificate address

1) Execute **cat examples/wsbrd.conf** in the MobaXterm terminal window to view the contents of the configuration file:

```
ubuntu@raspi:~/wisun-br-linux$ cat examples/wsbrd.conf
# Wi-SUN border router configuration example

# Parsing rules:
#   - Everything after # is ignored
#   - Spaces are ignored
#   - Escape sequences \xXX (for example, \x20 for space, \x0A for new line) are
#     accepted
#   - These characters are not accepted (you have to use escaped sequences):
#     SPACE, '#', '\\', '\n' and '\r'
#
# Unless specified in the comment, commented settings are optional and the value
# shown is the default value.
#
# Unless specified in the comment, if an option appears multiple times in the
# file, only the last one is taken into account.

################################################################################
# RCP serial configuration
################################################################################

# Path of the serial port connected to the RCP.
uart_device = /dev/ttyACM0

# Serial baudrate in bps.
#uart_baudrate = 115200

# Enable serial hardflow control.
#uart_rtscts = false
```

Figure 70: View Contents of wsbrd.conf Configuration File

2) Use the vi or vim editor to modify the transmission power value to -10. For detailed usage of the vi or Vim editor, please visit https://vimhelp.org/.

```
# Maximum TX power (dBm). Probably no hardware can exceed 18dBm.
# The device may utilize a lower value based on internal decision making or
# hardware limitations but will never exceed the given value.
tx_power = -10
```

Figure 71: Edit wsbrd.conf

Step 3: Execute the following commands in the MobaXterm terminal window to start the *wsbrd* program.

```
//Query RCP UART
ls /dev/ttyACM*
//Start wsbrd program
sudo wsbrd -F examples/wsbrd.conf -u /dev/ttyACM0
//Cache needs to be cleared after the configuration file is modified
sudo rm -rf /var/lib/wsbrd/*
```

```
ubuntu@raspi:~/wisun-br-linux$ ls /dev/ttyACM*
/dev/ttyACM0  /dev/ttyACM1
ubuntu@raspi:~/wisun-br-linux$ sudo wsbrd -F examples/wsbrd.conf -u /dev/ttyACM0
Silicon Labs Wi-SUN border router v2.2
Connected to RCP "2.0.0" (2.0.0), API 2.2.1
[INFO][addr]: Address added to IF 1: fe80::96b2:16ff:fe00:4c8c
[DBG ][addr]: LL64 Register OK!
[INFO][rout]: Added route:
[DBG ][rout]:                fe80::/64 if:1 src:'Static' id:0 lifetime:infinite
[DBG ][rout]:      On-link (met 128)
[INFO][rout]: Added route:
[DBG ][rout]:                 ff00::/8 if:1 src:'Static' id:0 lifetime:infinite
[DBG ][rout]:      On-link (met 192)
[INFO][addr]: Address added to IF 1: fd12:3456::96b2:16ff:fe00:4c8c
[INFO][rout]: Added route:
[DBG ][rout]:  fd12:3456::96b2:16ff:fe00:4c8c/128 if:1 src:'Loopback' id:0 lifetime:infinite
[DBG ][rout]:      On-link (met 128)
Successfully registered to system DBus
[INFO][wsbs]: Wi-SUN packet congestion minTh 85, maxTh 170, drop probability 10 weight 32, Packet/Seconds 2
[INFO][wspc]: Read GTK frame counter: index 0 value 1000000
[INFO][wspc]: Read LGTK frame counter: index 0 value 1000000
PAN version number update
  domain: NA
  channel plan id: 1
  phy mode id: 0x02
  phy operating modes: 3, 5, 6, 8, 18, 19, 21, 22, 24, 34, 35, 36, 37, 38, 51
  RCP configuration index: 0
  channel 0 frequency: 902.2MHz
  channel spacing: 200kHz
  channel count: 129
  channel masks:
              advertised                                          effective

    unicast   --                                                  00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:
00:fe
    broadcast --                                                  00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:
00:fe
    async     --                                                  00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:
00:fe

Nodes join ability:
  rank    FFN1.0    FFN1.1    LFN
    1     no        yes       yes
   >1     no        yes       can[1]
  [1]: neighboring routers must use a channel plan 2 (reg. domain & ChanPlanId)

[INFO][wspc]: GTK hash set 63:98:00:a7:4f:de:47:29 00:00:00:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:00:00:0
0:00:00:00
[INFO][wspc]: NW key set: 0, hash: 63:98:00:a7:4f:de:47:29
[INFO][wspc]: LGTK hash set a2:34:f3:b4:14:8d:80:19 00:00:00:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:00:00:
00:00:00:00
[INFO][wspc]: NW key set: 4, hash: a2:34:f3:b4:14:8d:80:19
[INFO][wspc]: NW send key index set: 0
[INFO][wspc]: NW send key index set: 4
Wi-SUN Border Router is ready
```

Figure 72: Start wsbrd Program

Step 4: Check the network status.

1) Right-click on the open MobaXterm terminal window and select "**Duplicate tab**" to open a new terminal window.

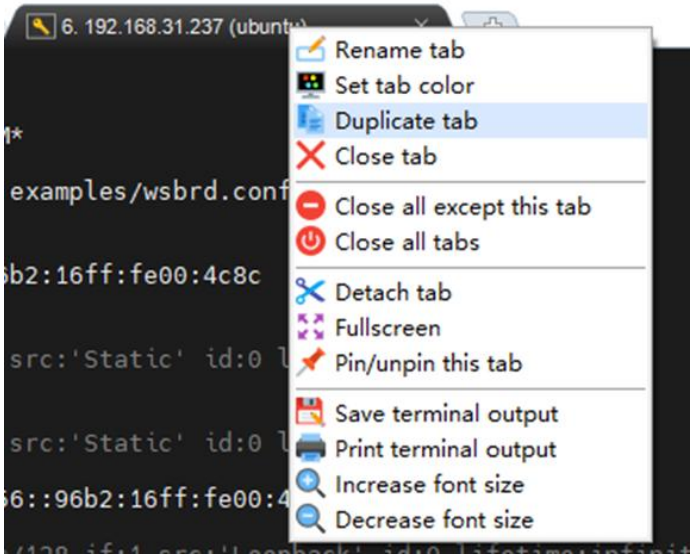


Figure 73: Open a New Terminal Window

2) Execute **wsbrd_cli status** in the new terminal window to check the network status.

```
ubuntu@raspi:~$ wsbrd_cli status
network_name: Wi-SUN Network
fan_version: FAN 1.1
domain: NA
phy_mode_id: 2
chan_plan_id: 1
panid: 0x8578
size: SMALL
GAK[0]: 31:b5:59:93:28:cf:57:54:f8:3b:dd:22:f2:d9:53:9e
GAK[1]: 26:1b:ac:fe:97:e5:2e:f1:3a:e5:d9:af:38:c2:0f:f3
GAK[2]: 26:1b:ac:fe:97:e5:2e:f1:3a:e5:d9:af:38:c2:0f:f3
GAK[3]: 26:1b:ac:fe:97:e5:2e:f1:3a:e5:d9:af:38:c2:0f:f3
GTK[0]: 48:e5:5b:4b:e6:7c:29:45:0f:75:0c:10:f2:2a:5a:20
GTK[1]: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
GTK[2]: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
GTK[3]: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
LGAK[0]: 39:cd:2f:66:3c:29:cc:c9:bc:77:27:3c:7b:a1:76:3a
LGAK[1]: 26:1b:ac:fe:97:e5:2e:f1:3a:e5:d9:af:38:c2:0f:f3
LGAK[2]: 26:1b:ac:fe:97:e5:2e:f1:3a:e5:d9:af:38:c2:0f:f3
LGTK[0]: 0c:2c:a3:62:61:81:2a:25:77:5b:4d:71:80:7e:76:78
LGTK[1]: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
LGTK[2]: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
fd12:3456::96b2:16ff:fe00:4c8c
ubuntu@raspi:~$
```

Figure 74: Check Network Status

3.2. CLI Application

The CLI application is a node demo program that provides a series of command-line interfaces to facilitate testing the connection and communication between the node and the router. This chapter uses the Development Board B as the test node to describe the CLI application development process, including the creation, modification, compilation, flashing, and debugging of the CLI project.

3.2.1. Create a Project

Step 1: Connect the Development Board B to the SI-DBG1015A Simplicity Link debugger, then connect the debugger to the PC.

Step 2: Follow ***Step 2*** in ***Chapter 3.1.1.1*** to set the chip model and development board model.

Step 3: Click "**Launcher**" in the upper-right corner of the Simplicity Studio tool's main interface to switch to the Launcher perspective. In the "Debug Adapters" section, select the debugger. Select the "**EXAMPLE PROJECTS & DEMOS**" tab, check "**Wi-SUN**" checkbox under the "Wireless Technology " category, select the "Wi-SUN – CLI example" demo template from the results, and click "**CREATE**" to open the "Project Configuration" interface.

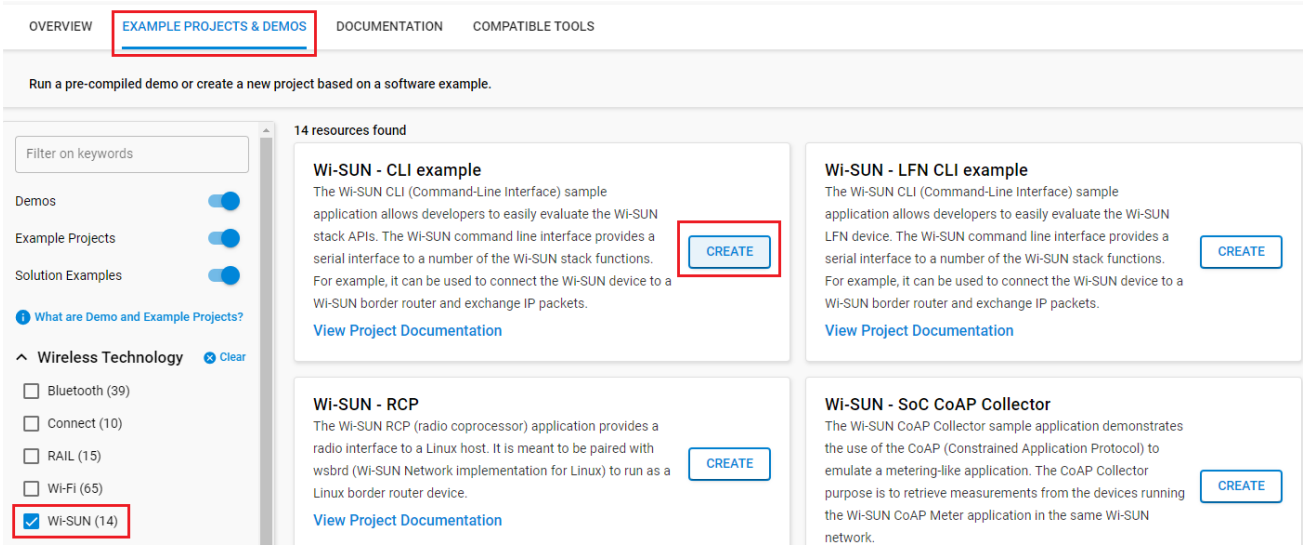


Figure 75: Create CLI Project

Step 4: Follow the ***Step 4*** in ***Chapter 3.1.1.1*** to set the project name (for example, "wisun_soc_cli") and project location. For details on the project directory structure, see ***Chapter 3.1.1.2***.

3.2.2. Modify Project Configuration

Step 1: Find the *.sclp* file in the wisun_soc_cli project directory and follow ***Step 1*** in ***Chapter 3.1.2*** to modify the development board model and target chip model.

Step 2: Modify CLI UART configuration.

1) Select the "**SOFTWARE COMPONENTS**" tab, check the "**Installed**" checkbox, find and select "**IO Stream: EUSART**", and click "**Configure**".
2) In the "IO Stream: EUSART" configuration interface, configure the settings in the "EUART settings" and "SL_IOSTREAM_EUSART_VCOM" sections.

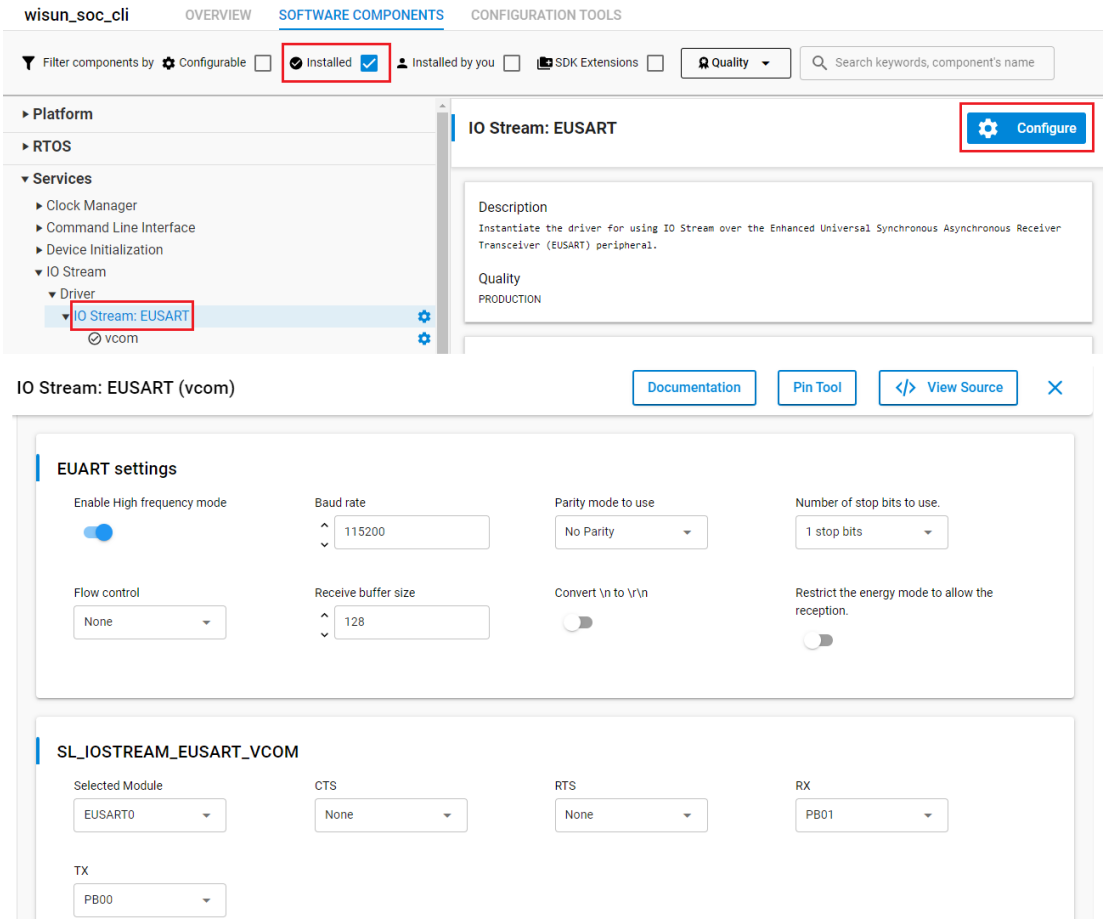


Figure 76: Modify UART Configuration

Step 3: Follow ***Step 3*** in ***Chapter 3.1.2*** to configure FEM.
Step 4: Follow ***Step 4*** in ***Chapter 3.1.2*** to configure RSSI offset.
Step 5: Modify the default RF configuration. Only default RF parameters for some specific development boards are configured in this demo, so you must modify the relevant codes. Find the *sl_wisun_default_phy.h* file in the *config* folder in the project directory. Double-click the file to open it, and modify the content as outlined in red below.

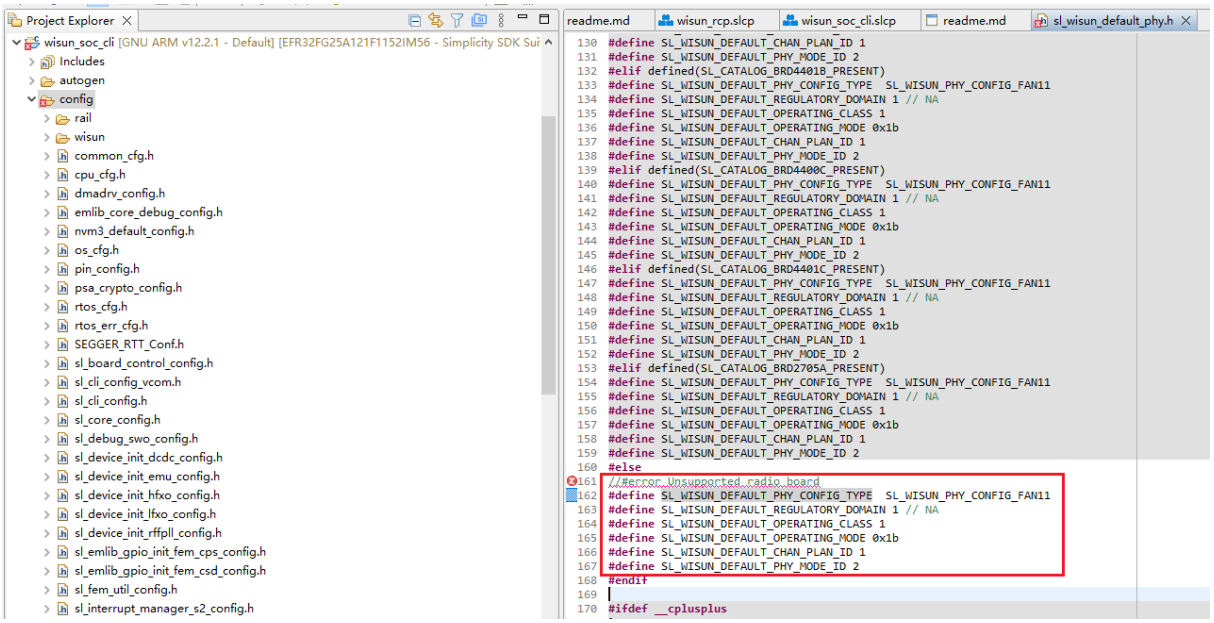


Figure 77: Modify Default RF Parameters

3.2.3. Compile Project

For details about project compilation, see ***Chapter 3.1.3***.

3.2.4. Flash Firmware

For details about firmware flashing, see ***Chapter 3.1.4***.

3.2.5. Debug Application

Step 1: Connect the Development Board B to the PC using a Type-C USB cable. The "Device Manager" on the PC displays the following two serial ports.

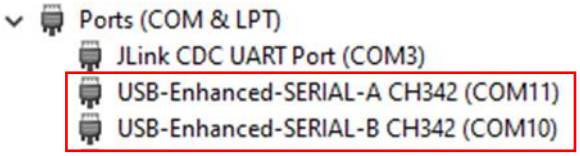


Figure 78: Device Manager

Step 2: Use any serial port debugging tool to connect to the SERIAL-A port. The baud rate and other configurations should match the UART component configuration in ***Step 2*** of ***Chapter 3.2.2***.

Step 3: Press the reset button on the Development Board B. The serial port tool prints "Wi-SUN CLI Application". Execute **wisun help** through the serial port tool to view the list of supported

commands and their descriptions. For detailed descriptions, refer to the *readme.md* file in the project directory.

```
Wi-SUN CLI Application
> wisun help
  set                       Set a variable
                            [*] empty | help | [string] Key [string] Value
  get                       Get a variable
                            [*] empty | help | [string] Key
  save                      Save variables to non-volatile storage
  reset                     Reset variables to default settings
  join_fan10                Connect to a Wi-SUN network using FAN1.0 settings: w j10
  join_fan11                Connect to a Wi-SUN network using FAN1.1 settings: w j11
  join_explicit             Connect to a Wi-SUN network using explicit PHY settings: w je
  join_ids                  Connect to a Wi-SUN network using radio conf ids: w ji
  join_custom_fsk           Connect to a Wi-SUN network using custom FSK PHY settings: w jcf
  join_custom_ofdm          Connect to a Wi-SUN network using custom OFDM PHY settings: w jcof
  join_custom_oqpsk         Connect to a Wi-SUN network using custom OQPSK PHY settings: w jcoq
  disconnect                Disconnect from the Wi-SUN network: w d
  ping                      Ping a remote host: w p
                            [string] Remote address
                            [uint16opt] packet size in byte
  tcp_client                Open a TCP client socket: w tc
                            [string] Remote address
                            [uint16] Remote port
  tcp_server                Open a TCP server socket: w ts
                            [uint16] Local port
  udp_client                Open a UDP client socket: w uc
                            [string] Remote address
                            [uint16] Remote port
  udp_server                Open a UDP server socket: w us
                            [uint16] Local port
  socket_close              Close a socket: w sc
                            [uint32] Socket Id
  socket_read               Read from a socket: w sr
                            [uint32] Socket Id
                            [uint16] Amount of bytes to read
  socket_write              Write to a socket: w sw
                            [uint32] Socket Id
                            [string] Data to write
  socket_writeto            Write to a socket to a specific host: w swt
                            [uint32] Socket Id
                            [string] Remote address
                            [uint16] Remote port
                            [string] Data to write
  socket_list               List open sockets: w sl
```

Figure 79: wisun help Output

3.3. Network Debugging

Setting up a Wi-SUN LAN requires at least one border router and one node. The border router consists of the Raspberry Pi 4 Model B and the Development Board A that runs the RCP application. The Development Board B serves as the node, which runs the CLI application, establishes network connections, and performs PING, TCP, and UDP communication tests via serial command line.

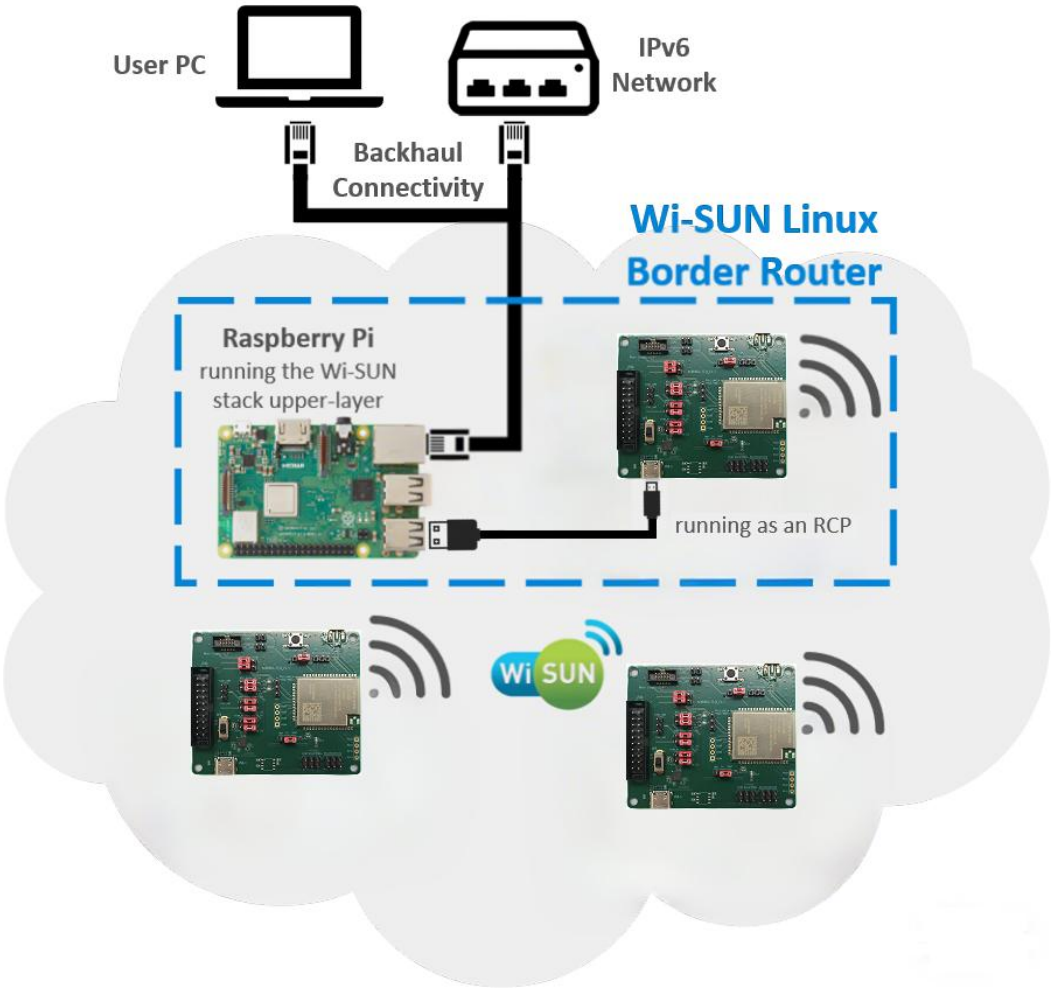


Figure 80: Network Connection

3.3.1. Establish Connection

Step 1: Log in to the Raspberry Pi 4 Model B (See ***Chapter 2.3.2***) and start the *wsbrd* program.

Step 2: Connect the node (Development Board B) to the PC and open a serial port debugging tool.

Step 3: Execute the following commands to configure the node network parameters, ensuring they match the border router's settings. Since the default node network parameters in this example already match the default configuration of the border router, only **wisun set wisun.tx_power -10** needs to be executed to modify the transmit power to -10 dBm (as configured in ***Step 2*** of ***Chapter 3.1.5***).

```
//Get the Wi-SUN network configuration
wisun get wisun
//Set the network name for the node to join
wisun set wisun.network_name "NAME"
//Set the transmit power
wisun set wisun.tx_power -10
```

```
//Set the channel number
wisun set wisun.chan_plan_id 1
//Set PHY mode
wisun set wisun.phy_mode_id 2
//Set the regulatory domain
wisun set wisun.regulatory_domain NA
//Join the network
wisun join_fan11
//Disconnect from the network
wisun disconnect
```

```
> wisun get wisun
wisun.join_state = Disconnected (0)
wisun.network_name = "Wi-SUN Network"
wisun.phy_config_type = FAN 1.1 (1)
wisun.regulatory_domain = NA (1)
wisun.operating_class = 1 (unused)
wisun.operating_mode = 0x1b (unused)
wisun.fec = 0 (unused)
wisun.chan_plan_id = 1
wisun.phy_mode_id = 2
wisun.ch0_frequency = 863100 (unused)
wisun.number_of_channels = 69 (unused)
wisun.channel_spacing = 100 (unused)
wisun.crc_type = 4-bytes (2) (unused)
wisun.preamble_length = 56 (unused)
wisun.stf_length = 4 (unused)
wisun.protocol_id = 0 (unused)
wisun.channel_id = 0 (unused)
wisun.network_size = small (1)
wisun.tx_power = -10
wisun.unicast_dwell_interval = 255
wisun.mac_address = 94:b2:16:ff:fe:00:4c:87
wisun.async_channel_mask = --
wisun.unicast_channel_mask = --
wisun.broadcast_channel_mask = --
wisun.allowed_channels = "0-255"
wisun.trace_filter.000-031 = 0xffffffff
wisun.trace_filter.032-063 = 0xffffffff
wisun.regulation = none (0)
wisun.regulation_warning_threshold = -1
wisun.regulation_alert_threshold = -1
wisun.device_type = FFN (0)
wisun.rx_mdr_capable = 0
wisun.rx_phy_mode_ids = no PhyModeId set
wisun.lfn_profile = test (0)
wisun.neighbor_table_size = 22
wisun.keychain = automatic (0)
wisun.keychain_index = 0
> wisun join_fan11
[Using built-in trusted CA #0]
[Using built-in device credentials]
[Connecting to "Wi-SUN Network"]
[Join state 1: Select PAN]
> [Join state 2: Authenticate]
[Join state 3: Acquire PAN Config]
[Join state 4: Configure Routing - parent selection]
[Join state 4: Configure Routing - DHCP]
[Join state 4: Configure Routing - address registration]
[Join state 4: Configure Routing - DAO registration]
[Join state 5: Operational]
[Connected: 122 s]
```

Figure 81: Node Connection Process

An example of the connection log of the border router in MobaXterm is as follows:

```
Nodes join ability:
  rank    FFN1.0    FFN1.1    LFN
    1     no        yes       yes
   >1     no        yes       can[1]
  [1]: neighboring routers must use a channel plan 2 (reg. domain & ChanPlanId)

[INFO][wspc]: GTK hash set 18:28:e5:4c:08:cc:b2:4d 00:00:00:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:00:00:00:00:00:00
[INFO][wspc]: NW key set: 0, hash: 18:28:e5:4c:08:cc:b2:4d
[INFO][wspc]: LGTK hash set 63:d9:58:a4:b3:d0:0c:7a 00:00:00:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:00:00:00:00:00:00
[INFO][wspc]: NW key set: 4, hash: 63:d9:58:a4:b3:d0:0c:7a
[INFO][wspc]: NW send key index set: 0
[INFO][wspc]: NW send key index set: 4
Wi-SUN Border Router is ready
[INFO][wsbs]: Congestion check, summed averageQ: 0 adapt averageQ: 0 LLC averageQ: 0 LLC EAPOL averageQ: 0 drop: F
[DBG ][wspa]: PAE: to active, eui-64: 94:b2:16:ff:fe:00:4c:87
[DBG ][ksep]: Initial-key init
[DBG ][ksep]: PMK not live PTK not live NR 1 GTKL 0 LGTKL 0
[INFO][wspa]: PAE: start EAP-TLS, eui-64: 94:b2:16:ff:fe:00:4c:87
[DBG ][eapa]: EAP-TLS init
[DBG ][tlsp]: TLS: init
[DBG ][eapa]: EAP-TLS start, eui-64: 94:b2:16:ff:fe:00:4c:87
[DBG ][ksep]: Initial-key finished, eui-64: 94:b2:16:ff:fe:00:4c:87
[DBG ][tlsp]: TLS: start, eui-64: 94:b2:16:ff:fe:00:4c:87
[DBG ][tlsp]: TLS: finish, eui-64: 94:b2:16:ff:fe:00:4c:87
[INFO][eapa]: EAP-TLS: handshake success
[DBG ][eapa]: EAP-TLS finish, eui-64: 94:b2:16:ff:fe:00:4c:87
[INFO][wspa]: PAE: start 4WH, eui-64: 94:b2:16:ff:fe:00:4c:87
[INFO][wspa]: PAE: update (L)GTK index: 0, eui-64: 94:b2:16:ff:fe:00:4c:87
[DBG ][afwh]: 4WH: init
[DBG ][afwh]: 4WH: start, eui-64: 94:b2:16:ff:fe:00:4c:87
[DBG ][secl]: PMKID 79:4c:c3:f0:d9:8f:56:95:c2:1f:3f:92:59:4d:69:3c EUI-64 94:b2:16:ff:fe:00:4c:8c 94:b2:16:ff:fe:00:4c:87
[DBG ][afwh]: 4WH: finish, eui-64: 94:b2:16:ff:fe:00:4c:87
[INFO][wspa]: PAE: start GKH for LGTK index 0, eui-64: 94:b2:16:ff:fe:00:4c:87
[INFO][wspa]: PAE: update (L)GTK index: 0, eui-64: 94:b2:16:ff:fe:00:4c:87
[DBG ][agkh]: GKH init
[DBG ][agkh]: GKH start, eui-64: 94:b2:16:ff:fe:00:4c:87
[DBG ][agkh]: GKH finish, eui-64: 94:b2:16:ff:fe:00:4c:87
[INFO][wspa]: PAE: authenticated, eui-64: 94:b2:16:ff:fe:00:4c:87
[DBG ][tlsp]: TLS: finished, eui-64: 94:b2:16:ff:fe:00:4c:87 free F
[DBG ][eapa]: EAP-TLS finished, eui-64: 94:b2:16:ff:fe:00:4c:87
[DBG ][afwh]: 4WH: finished, eui-64: 94:b2:16:ff:fe:00:4c:87
[DBG ][agkh]: GKH finished, eui-64: 94:b2:16:ff:fe:00:4c:87
[DBG ][wspl]: PAE: to inactive, eui-64: 94:b2:16:ff:fe:00:4c:87
[INFO][wspa]: Supplicant deleted
[INFO][wsbs]: Congestion check, summed averageQ: 0 adapt averageQ: 0 LLC averageQ: 0 LLC EAPOL averageQ: 0 drop: F
[INFO][rout]: Added route:
[DBG ][rout]:  fd12:3456::96b2:16ff:fe00:4c87/128 if:1 src:'ARO' id:0 lifetime:2218
[DBG ][rout]:      next-hop fd12:3456::96b2:16ff:fe00:4c87 (met 32)
[DBG ][icmp]: Omit NA ARO success
[INFO][rout]: Added route:
[DBG ][rout]:  fd12:3456::96b2:16ff:fe00:4c87/128 if:1 src:'RPL DAO SR' id:0 lifetime:infinite
[DBG ][rout]:      next-hop :: (met 128)
```

Figure 82: Border Router Connection Log

Step 4: Execute **wsbrd_cli status** in the MobaXterm terminal window to query the network status of the border router.

```
ubuntu@raspi:~$ wsbrd_cli status
network_name: Wi-SUN Network
fan_version: FAN 1.1
domain: NA
phy_mode_id: 2
chan_plan_id: 1
panid: 0x6033
size: SMALL
GAK[0]: 81:8f:e6:15:14:15:26:a7:2a:b2:c4:aa:02:50:90:bf
GAK[1]: 26:1b:ac:fe:97:e5:2e:f1:3a:e5:d9:af:38:c2:0f:f3
GAK[2]: 26:1b:ac:fe:97:e5:2e:f1:3a:e5:d9:af:38:c2:0f:f3
GAK[3]: 26:1b:ac:fe:97:e5:2e:f1:3a:e5:d9:af:38:c2:0f:f3
GTK[0]: df:2f:43:ed:32:3a:57:bf:96:8e:81:c6:5c:da:c5:f4
GTK[1]: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
GTK[2]: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
GTK[3]: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
LGAK[0]: 74:cf:cf:1b:10:be:40:ce:55:20:28:66:7a:91:88:08
LGAK[1]: 26:1b:ac:fe:97:e5:2e:f1:3a:e5:d9:af:38:c2:0f:f3
LGAK[2]: 26:1b:ac:fe:97:e5:2e:f1:3a:e5:d9:af:38:c2:0f:f3
LGTK[0]: 56:5d:ab:11:fe:27:97:81:3b:19:60:05:07:da:1e:06
LGTK[1]: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
LGTK[2]: 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
fd12:3456::96b2:16ff:fe00:4c8c
  `- fd12:3456::96b2:16ff:fe00:4c87
ubuntu@raspi:~$
```

Figure 83: Query Network Status of Border Router

3.3.2. PING Test

Step 1: Execute **wisun get wisun.parents** to obtain the IP address of the parent node of Development Board B in the network.

Step 2: Execute **wisun ping [ipv6]** to ping the parent node. If the following information is returned, it indicates that the parent node was pinged successfully.

```
wisun get wisun.parents
wisun.parents = [fd12:3456::96b2:16ff:fe00:4c87]
> wisun ping fd12:3456::96b2:16ff:fe00:4c87
PING fd12:3456::96b2:16ff:fe00:4c87: 40 data bytes
> 40 bytes from fd12:3456::96b2:16ff:fe00:4c87: icmp_seq=1
time=148 ms
wisun ping fd12:3456::96b2:16ff:fe00:4c87
PING fd12:3456::96b2:16ff:fe00:4c87: 40 data bytes
> 40 bytes from fd12:3456::96b2:16ff:fe00:4c87: icmp_seq=1
time=86 ms
wisun ping fd12:3456::96b2:16ff:fe00:4c87
PING fd12:3456::96b2:16ff:fe00:4c87: 40 data bytes
> 40 bytes from fd12:3456::96b2:16ff:fe00:4c87: icmp_seq=1
time=78 ms
```

Figure 84: PING Test

3.3.3. TCP Communication Test

Step 1: Open another MobaXterm terminal window on the border router side, and use the Linux system's built-in Netcat tool to execute the following command to start a TCP server.

```
//Start a TCP server with port 12345
nc -6 -l -p 12345
```

Step 2: Execute the following commands on Development Board B to perform a TCP communication test.

```
//Start a TCP client
wisun tcp_client [SERVER IP] [PORT]
//Send data to the server
wisun socket_write [Socket_id] [MESSAGE]
//Close the client
wisun socket_close [Socket_id]
```

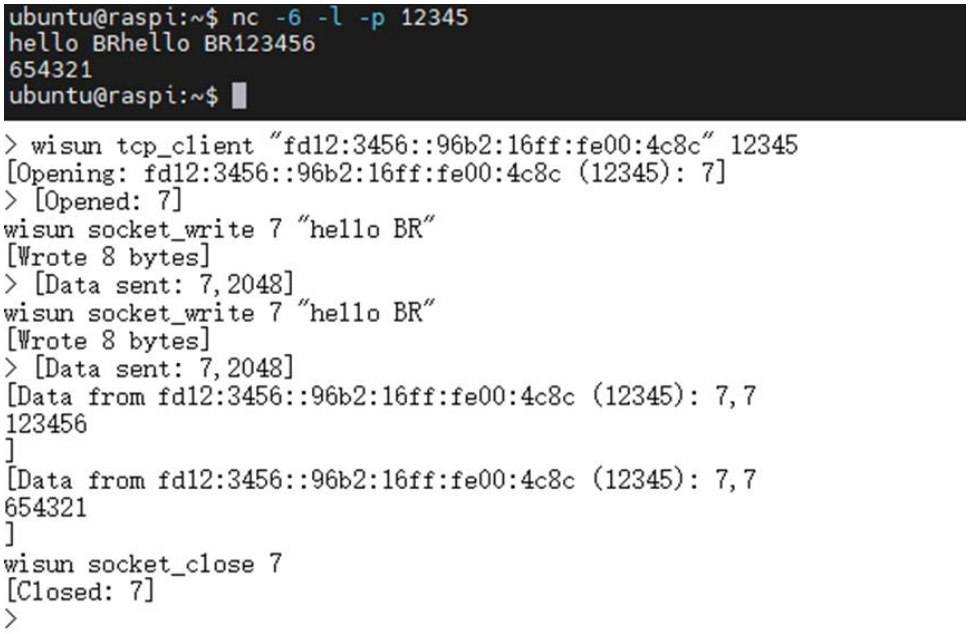


Figure 85: TCP Communication Test

3.3.4. UDP Communication Test

Step 1: Open another MobaXterm terminal window on the border router side, and use the Linux system's built-in Netcat tool to execute the following command to start a UDP server.

```
//Start a UDP server with port 12345.
nc -6 -lu 12345
```

Step 2: Execute the following commands on Development Board B to perform a UDP communication test.

```
//Start a UDP client
wisun udp_client [SERVER IP] [PORT]
//Send data to the server
wisun socket_write [Socket_id] [MESSAGE]
//Close the client
wisun socket_close [Socket_id]
```

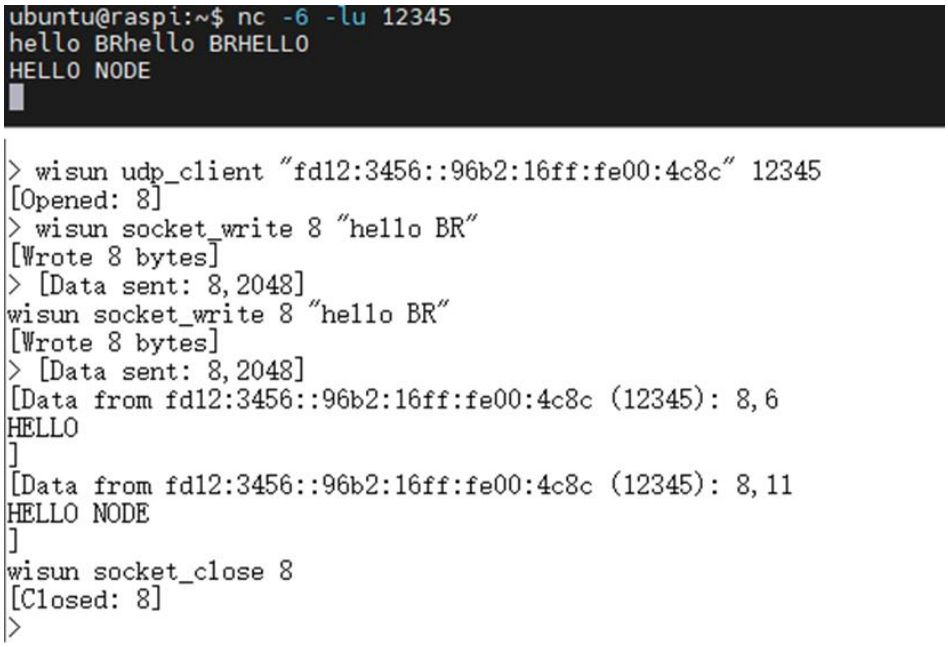


Figure 86: UDP Communication Test

4 General Application Debugging Methods

This chapter introduces general debugging methods applicable to both the border routers and node devices. The strings printed by the Wi-SUN Stack can be viewed through J-Link RTT Viewer. Packet information can be captured using the Network Analyzer tool in Simplicity Studio and the SI-DBG1015A Simplicity Link debugger.

4.1. Log Viewing

Step 1: Download J-Link RTT Viewer from https://www.segger.com/downloads/jlink (it is recommended to download version V8.10k or later) and follow the prompts to install it.

Step 2: After installation, open J-Link RTT Viewer, set the chip model, and click "**OK**" to save the configuration.

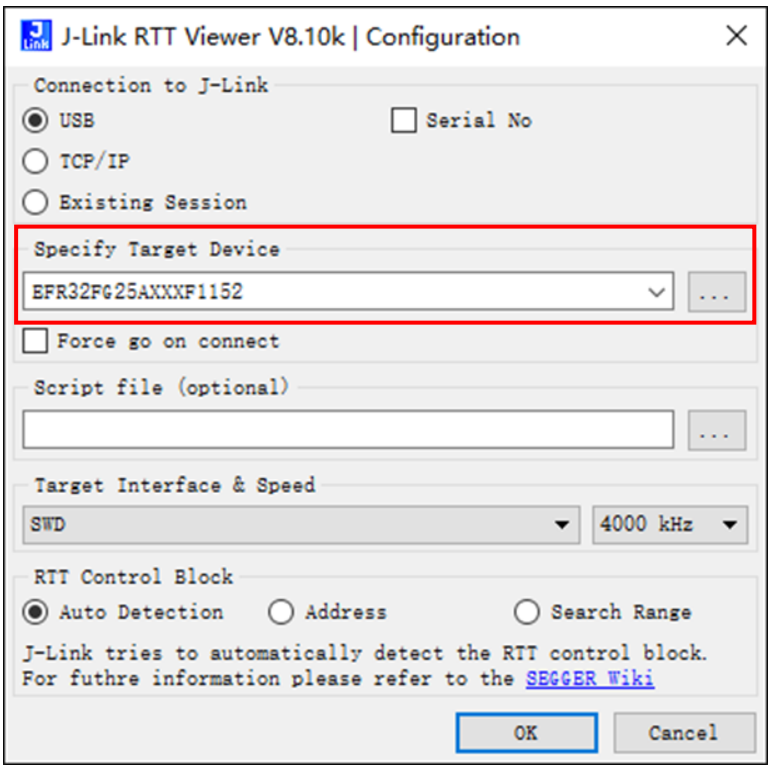


Figure 87: J-Link RTT Viewer Settings

Step 3: After the above configuration is completed, the tool automatically connects to the device. You can also press the **F2** key to reconnect. Once the connection is successful, the tool automatically starts capturing the logs.

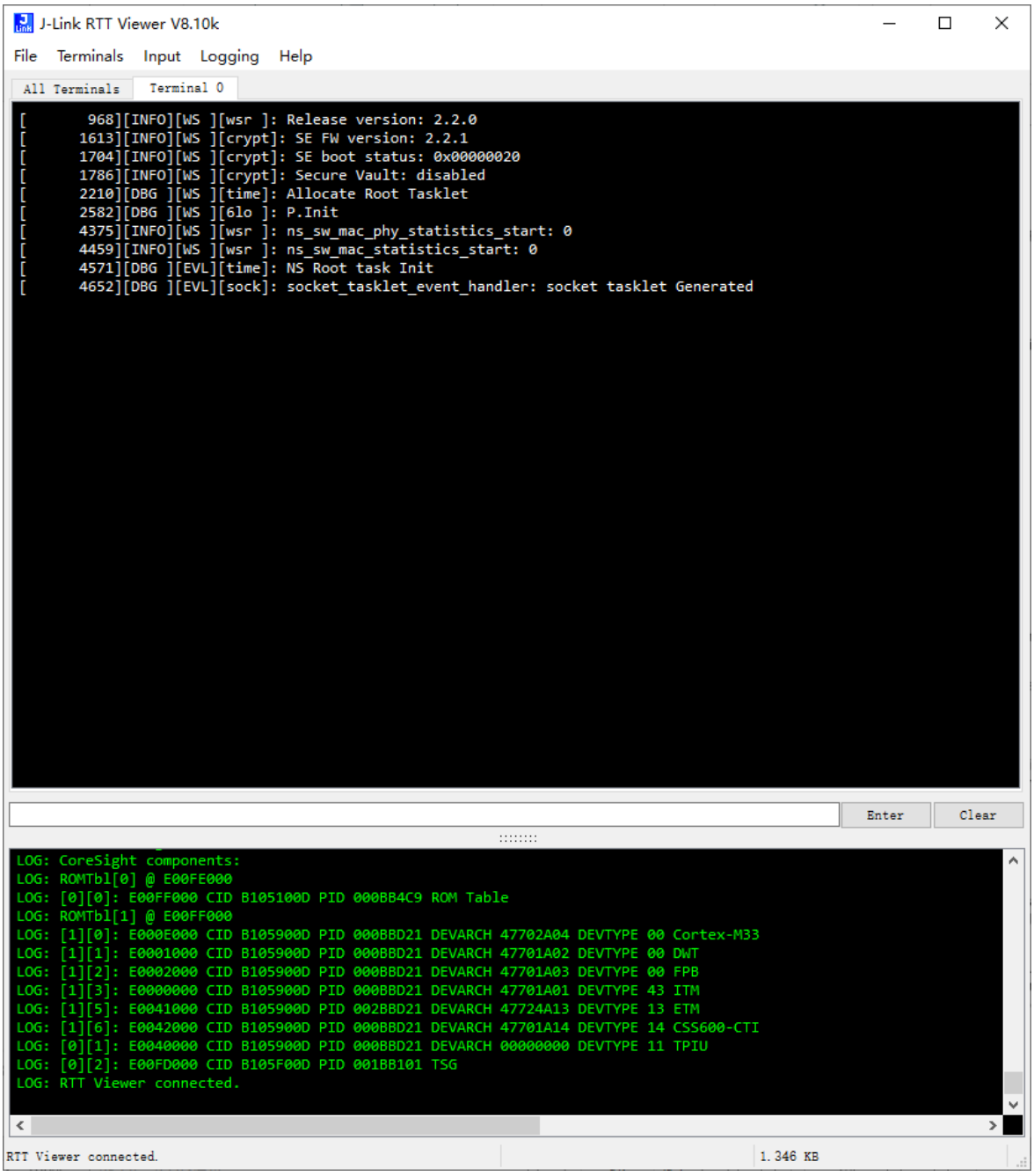


Figure 88: Log Capture via J-Link RTT Viewer

4.2. Packet Capture

Packet capture requires the use of the PTI interface on the SI-DBG1015A Simplicity Link debugger and the Network Analyzer tool in Simplicity Studio. The software PTI interface component needs to be added, and the hardware PTI interface pins must be used.

Step 1: Use jumper caps to connect the PTI_FRAME and PTI_DATA pins from the SI-DBG1015A Simplicity Link debugger's interface to the PD2 and PD3 pins on the KCM0A5S-TE-B.

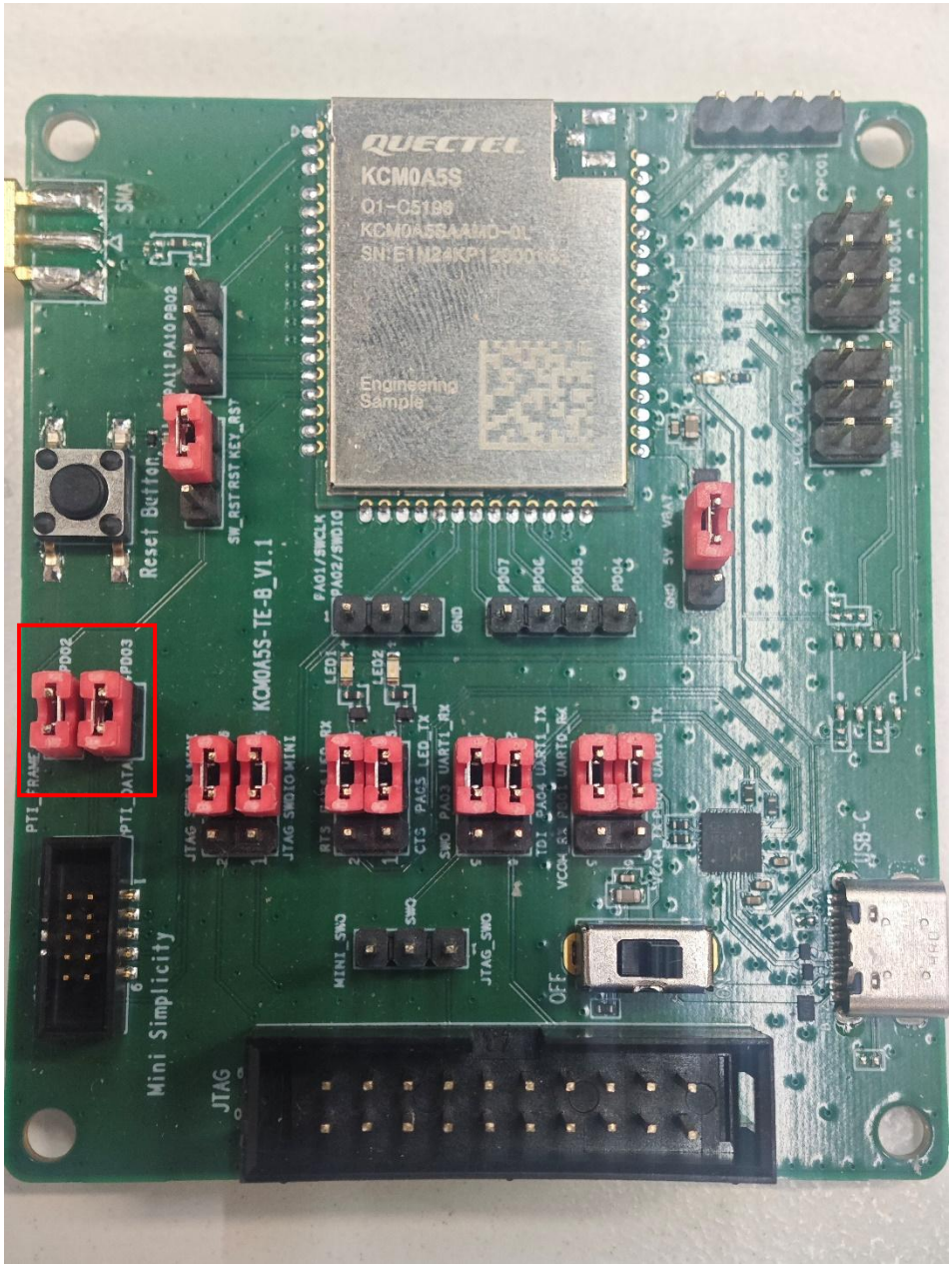


Figure 89: Jumper Placement on PTI Interface

Step 2: In the Simplicity Studio tool's "Project Explorer" section, find and expand the relevant project directory. In the project directory, find and double-click the .sclp file to open it. Select the "**SOFTWARE COMPONENTS**" tab, check the "**Installed**" checkbox, then enter "pti" in the search box. Select "**RAIL Utility, PTI**" from the search results and click "**Configure**" to open the PTI component configuration interface.

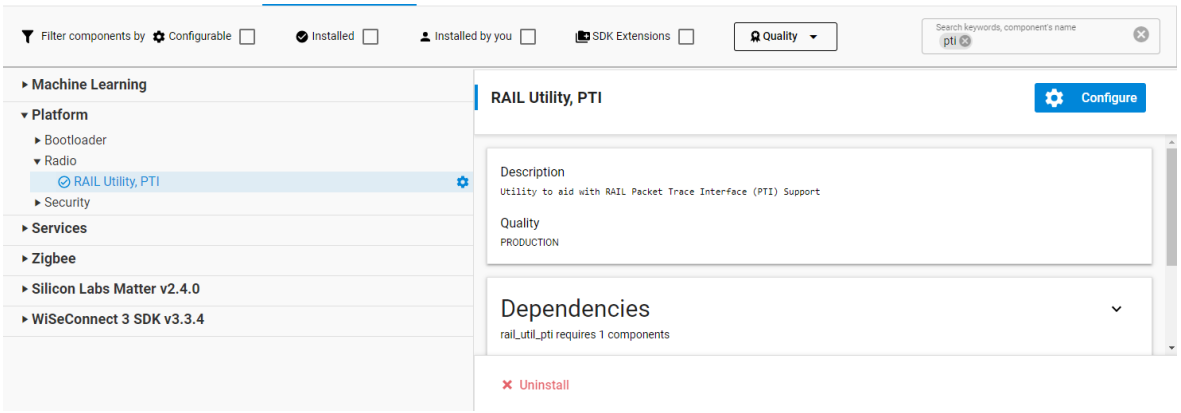


Figure 90: Locate PTI Component

Step 3: Configure the PTI control pin as follows:

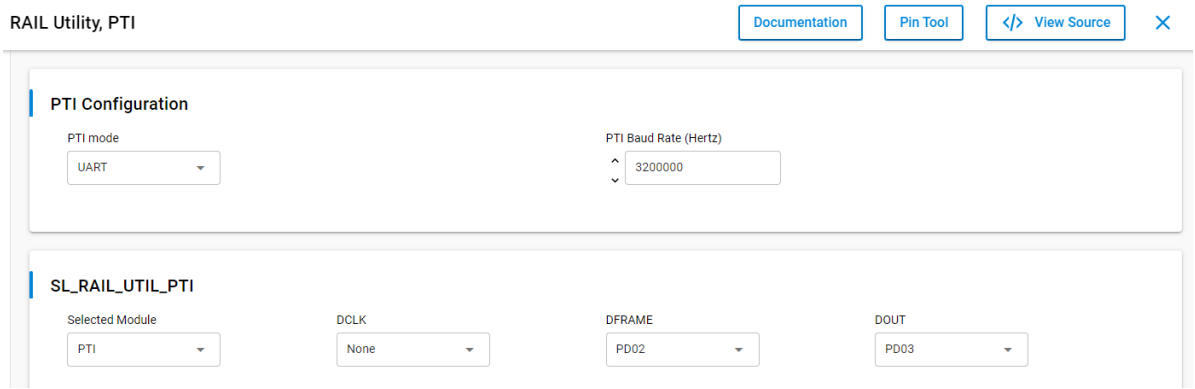


Figure 91: Configure PTI Control Pin

Step 4: Click "**Tools**" on the Simplicity Studio toolbar, select the "**Device Console**" tool, and then click "**OK**" to open the tool.

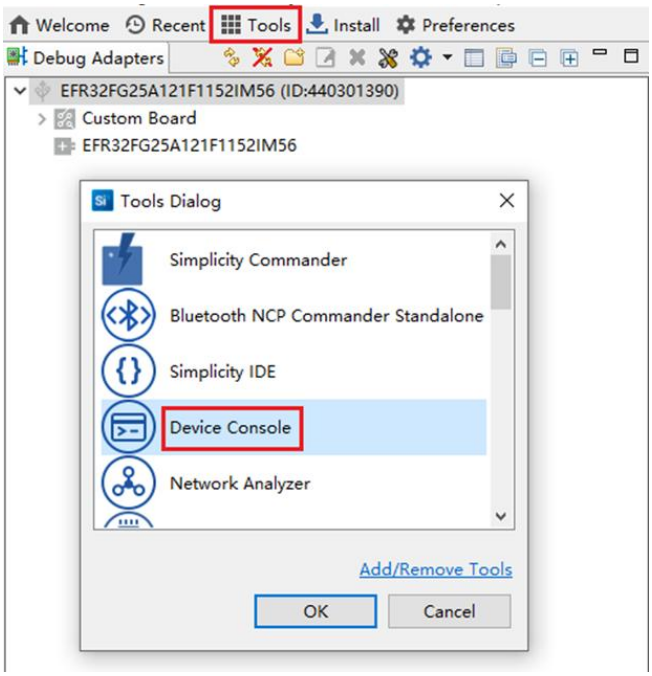


Figure 92: Open Device Console

Step 5: Select "**Admin**" and execute **pti config** to query the baud rate. To modify the baud rate, execute **pti config <interface> efruart <bitrate>**, for example, **pti config 0 efruart 3200000**.

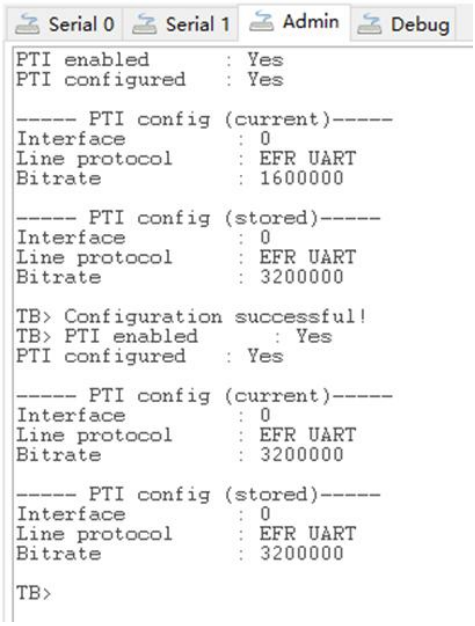


Figure 93: Set PTI Baud Rate

Step 6: Connect the SI-DBG1015A Simplicity Link Debugger, which is connected to the KCM0A5S-TE-B, to the PC. In the "Debug Adapters" section of Simplicity Studio, find and right-click the debugger (e.g., J-Link Silicon Labs (440301390)), and select "**Device configuration**". The configuration window opens on the "**Device hardware**" tab. In the "**Boards**" text box, enter and select the corresponding development board model, then click "**OK**" to save the configuration.

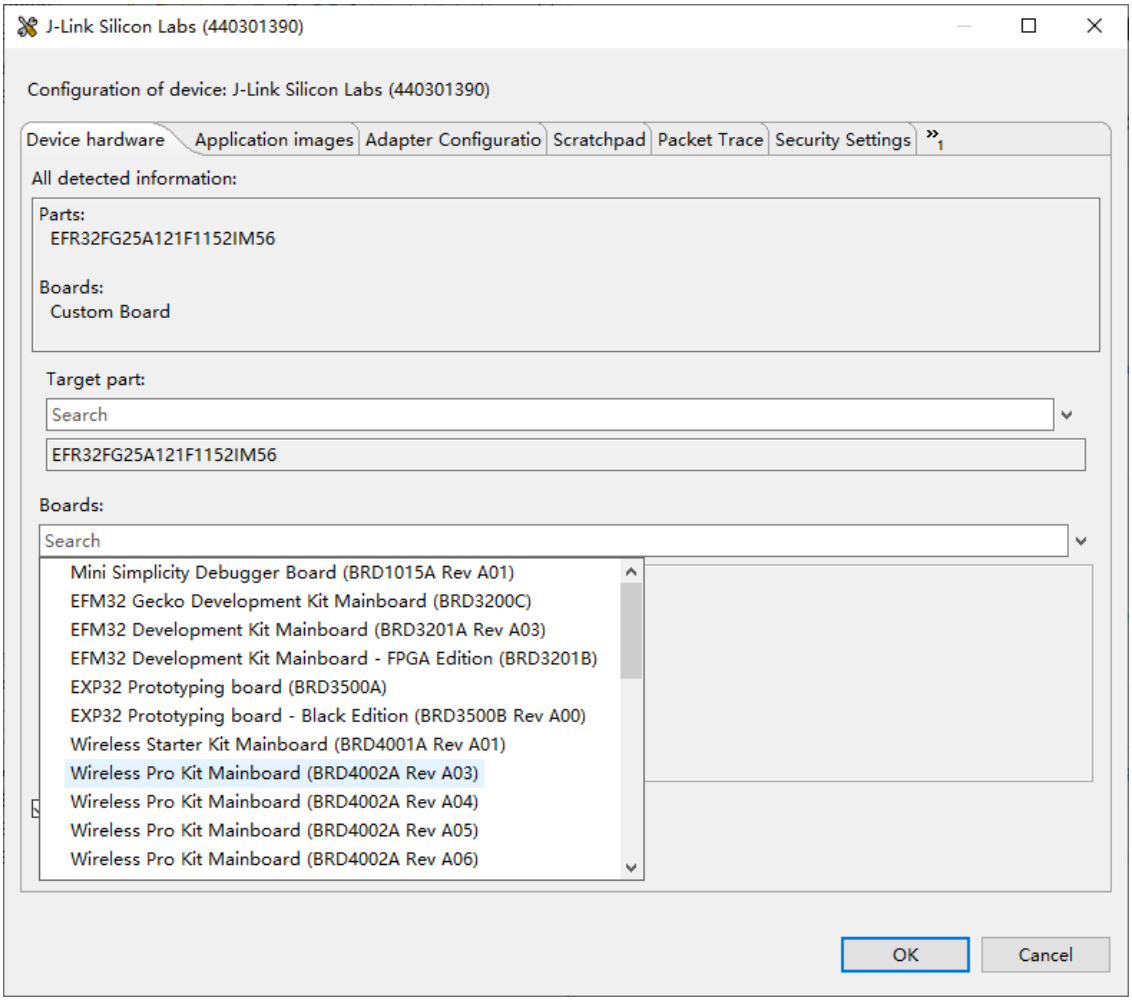


Figure 94: Select Development Board Model

Step 7: Right-click the debugger in the "Debug Adapters" section of Simplicity Studio and select "**Connect**" to connect the device.

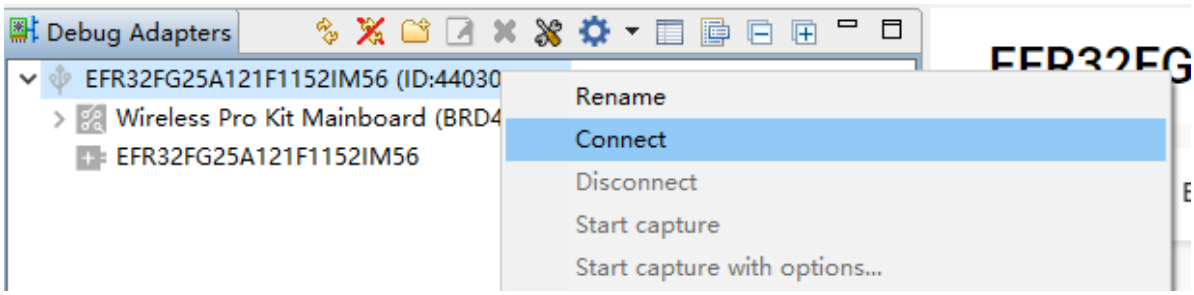


Figure 95: Connect Device

Step 8: After a successful connection, right-click the debugger, click "**Start capture**" to open the network analysis tool page to begin capturing packets.

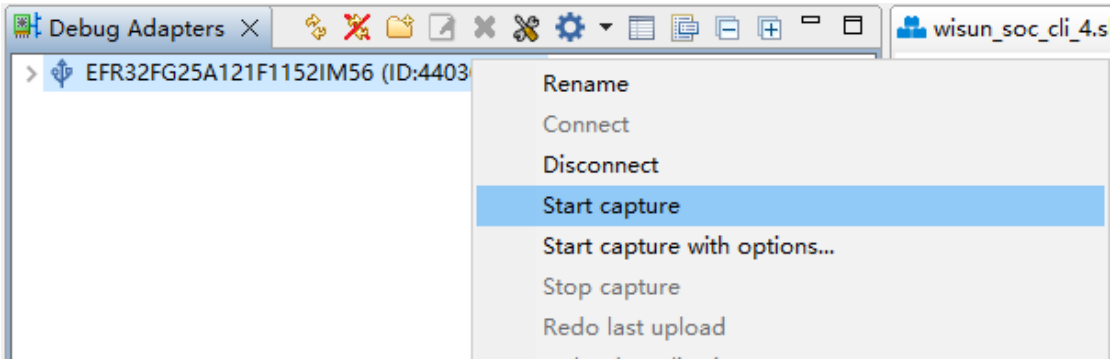


Figure 96: Begin Capturing Packets

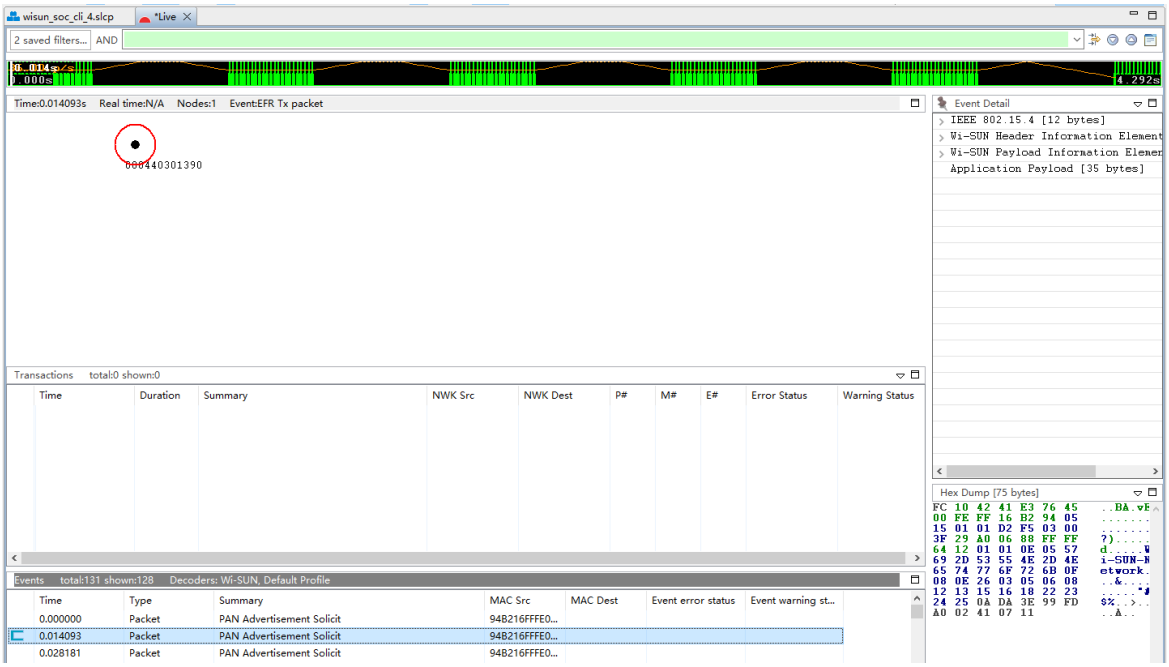


Figure 97: Network Analysis Tool

Step 9: Click the file export icon on the toolbar. In the "Network analyzer data export" pop-up window, configure the export options and click "**OK**". The tool then begins exporting packets.

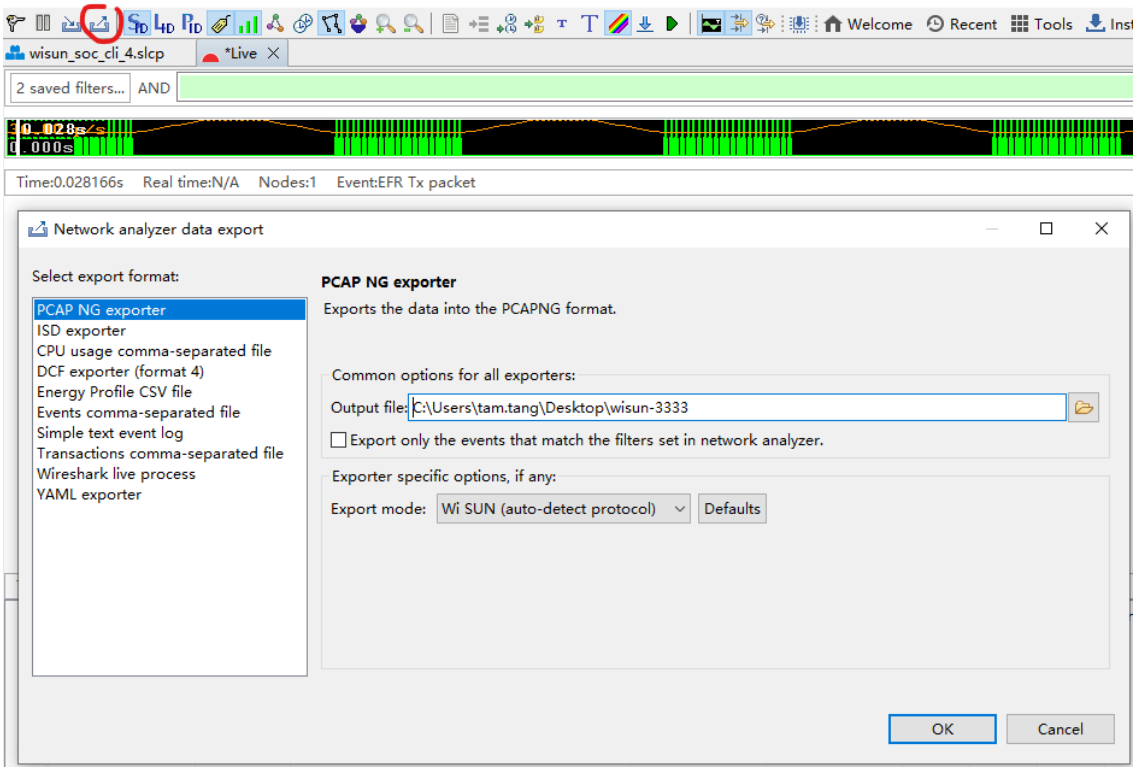


Figure 98: Export Packet Data

5 Appendix References

Table 7: Related Documents

Document Name
[1] Quectel_KCM0A5S_TE-B_User_Guide
[2] Quectel_KCM0A5S_Hardware_Design

Table 8: Terms and Abbreviations

Abbreviation	Description
BR	Border Router
CA	Certificate Authority
CDC	Communication Device Class
CLI	Command-line Interface
CSD	Control Shutdown
EVM	Error Vector Magnitude
FEM	Front-end Modules
GPIO	General-Purpose Input/Output
HFXO	High-Frequency Crystal Oscillator
ID	Identifier
IDE	Integrated Development Environment
IoT	Internet of Things
IP	Internet Protocol
LAN	Local Area Network
LNA	Low-Noise Amplifier

MAC	Medium Access Control
MCU	Microprogrammed Control Unit
OTA	Over-the-air programming
OS	Operating System
PC	Personal Computer
PCB	Printed Circuit Board
PHY	Physical
PTI	Packet Trace Interface
RAIL	Radio Abstraction Interface Layer
RCP	Radio Co-Processor
RSSI	Received Signal Strength Indicator
RTOS	Real-Time Operating System
RTT	Real-Time Transfer
RX	Receive
SD	Secure Digital Card
SDK	Software Development Kit
SoC	System on Chip
SSH	Secure Shell
SWD	Serial Wire Debug
TCP	Transmission Control Protocol
TX	Transmit
UDP	User Datagram Protocol
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
Wi-SUN	Wireless Smart Ubiquitous Network