

CHAPTER TO

Prepare to
Develop



한동대학교-
테크노니아
마이크로프로
세서응용

Contents

| Chapter | Contents |
|---------|---|
| T0 | Prepare to Develop. Build Environment (nRF Connect SDK, Zephyr) |
| T1 | Devicetree. Development in Zephyr. Control Button and LED. |
| T2 | Booting. Power Management. RAM Retention. |
| T3 | UART communication. Read CO2 sensor using UART. |
| T4 | Bluetooth Low Energy (BLE). BLE in Zephyr. Build Broadcaster and Peripheral device. |
| T5 | Various Interfaces with shield board. |
| T6 | LCD control with SPI interface |

Agenda

nRF Connect
SDK

Develop Zephyr
application

Check
Tools and Board

Build and Flash
Application

nRF Connect SDK

- nRF Connect SDK
 - For Nordic Semiconductor [nRF52](#), nRF53, nRF70 and nRF91.
 - Supports Microsoft Windows, Linux, macOS
- nRF Connect SDK features
 - Based on [Zephyr RTOS](#) and open source
 - ✓ https://developer.nordicsemi.com/nRF_Connect_SDK/doc/2.5.2/zephyr/introduction/index.html
 - ✓ https://developer.nordicsemi.com/nRF_Connect_SDK/doc/2.5.2/nrf/releases_and_maturity/developing/code_base.html#nrf-connect-sdk-code-base
 - Middleware and security
 - ✓ MQTT, Trusted Firmware-M, MbedTLS
 - Connectivity
 - ✓ [BLE](#), IPv6, TCP/IP, UDP, LoRa, WiFi, Matter

Installing nRF Connect SDK

- Two ways to install
 - With Visual Studio Code
 - ✓ nRF Connect for VS Code extension
 - ✓ Recommended : we go this way!!
 - With command line and nRF Util
- Install Prerequisites
 - nRF Command Line Tools package
 - ✓ Download it from nRF Command Line Tools page
 - ✓ <https://www.nordicsemi.com/Products/Development-tools/nrf-command-line-tools>
 - Visual Studio Code
 - nRF Connect for VS Code Extension Pack
- Use Version
 - nRF Connect SDK : v2.5.2
 - J-Link : v7.94e or v7.94m

Installing nRF Connect SDK

- nRF Command Line Tools
 - nrfjprog : tool for programming and debug

The screenshot shows a web browser displaying the [nRF Command Line Tools](https://www.nordicsemi.com/Products/Development-tools/nrf-command-line-tools) page on the Nordic Semiconductor website. The URL in the address bar is `https://www.nordicsemi.com/Products/Development-tools/nrf-command-line-tools`. The page has a dark header with the Nordic Semiconductor logo, a navigation bar with 'Overview' and 'Downloads' (which is underlined), and a 'Products' and 'Resources' dropdown. Below the header, there's a large image of the nRF Command Line Tools software interface with the text 'nRF Command Line Tools DESKTOP'. The main content area is titled 'Choose platform and version' and includes a dropdown menu set to 'Windows x86 64'. A blue arrow points from the text 'Selected version' to this dropdown. To the right, a 'Changelog' section lists several versions of the tool, with the latest one being '10.24.0 Windows x86 64'.

Selected version
10.24.0 Windows x86 64

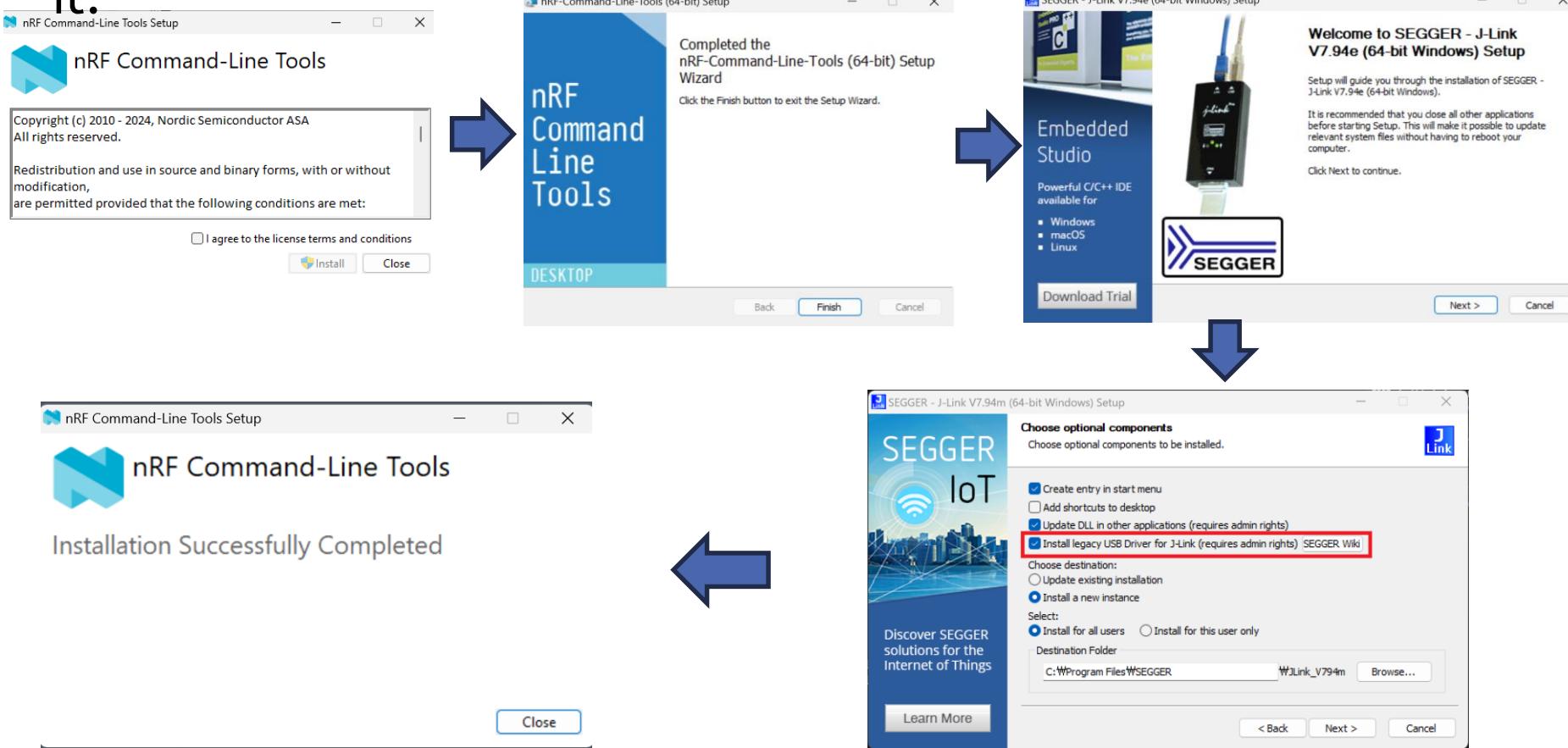
- nRF-Command-Line-Tools-10.24.0-x64.exe

Changelog:

- 10.24.0 Windows x86 64
 - Update bundled Segger installers and tarballs to v7.94e
- 10.23.5 Windows x86 64
- 10.23.4 Windows x86 64
- 10.23.2 Windows x86 64

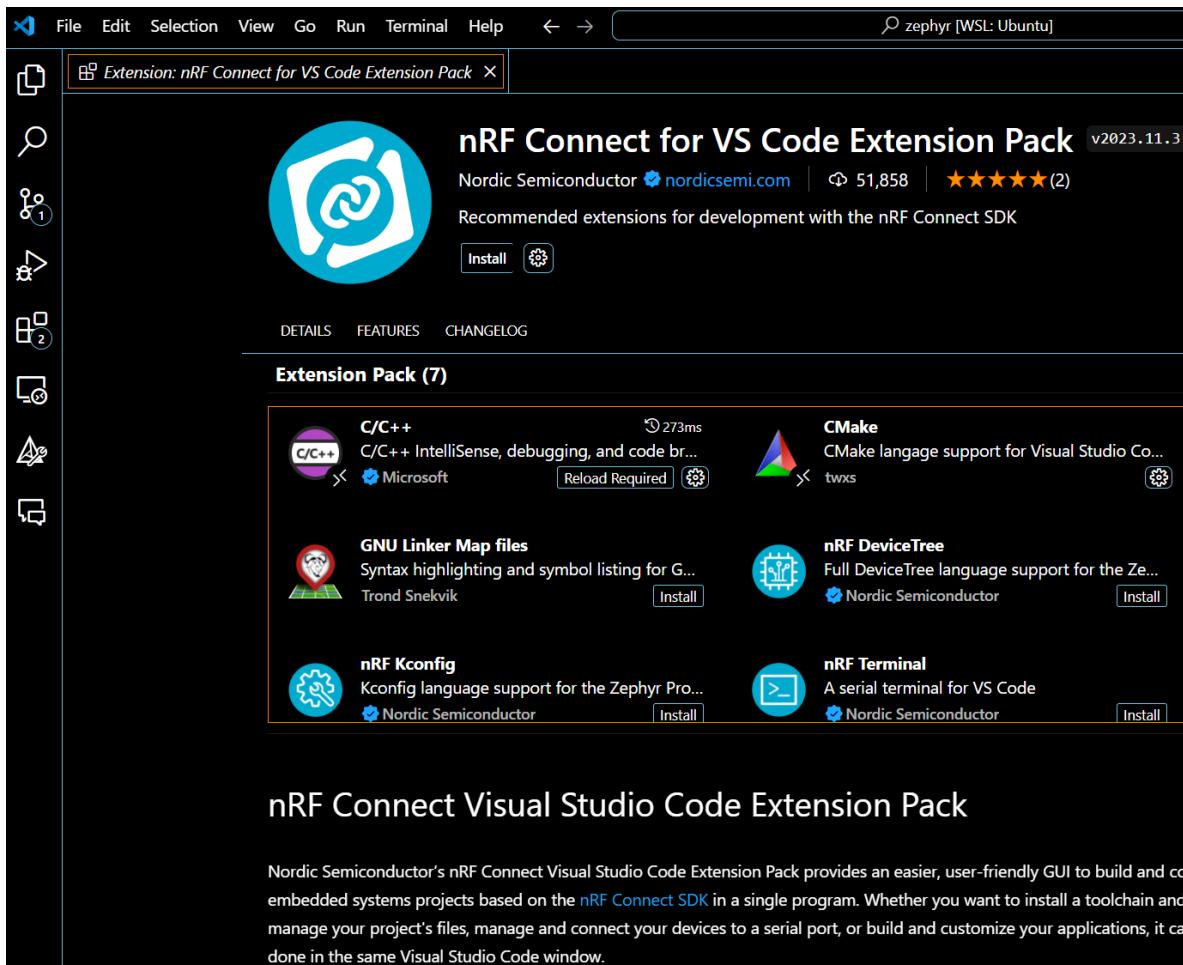
Installing nRF Connect SDK

- Download install file and run it.



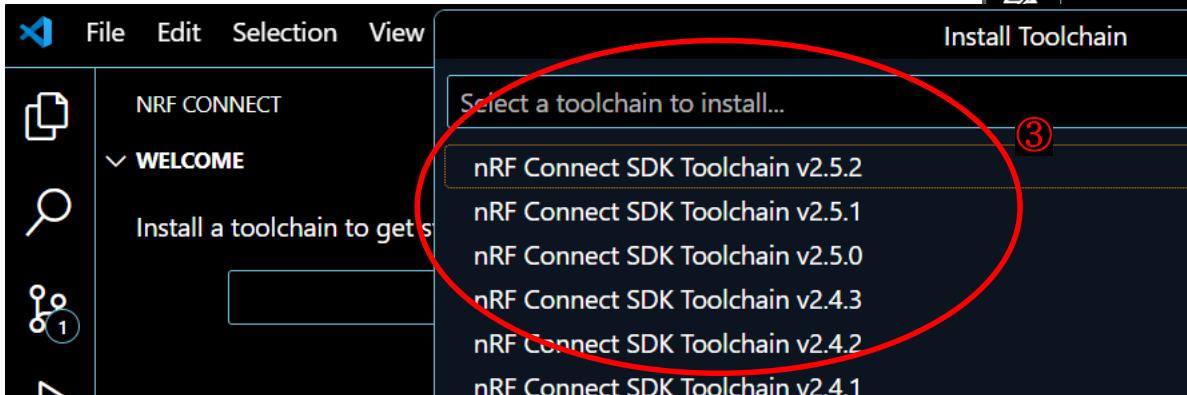
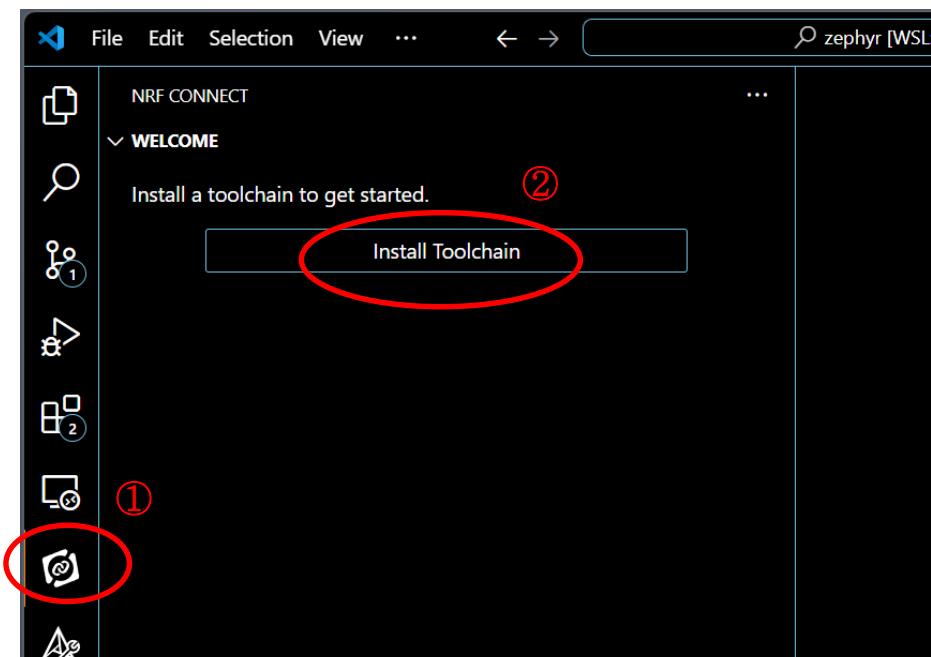
Installing nRF Connect SDK

- nRF Connect for VS Code Extension Pack
 - from Extensions menu in VSCode

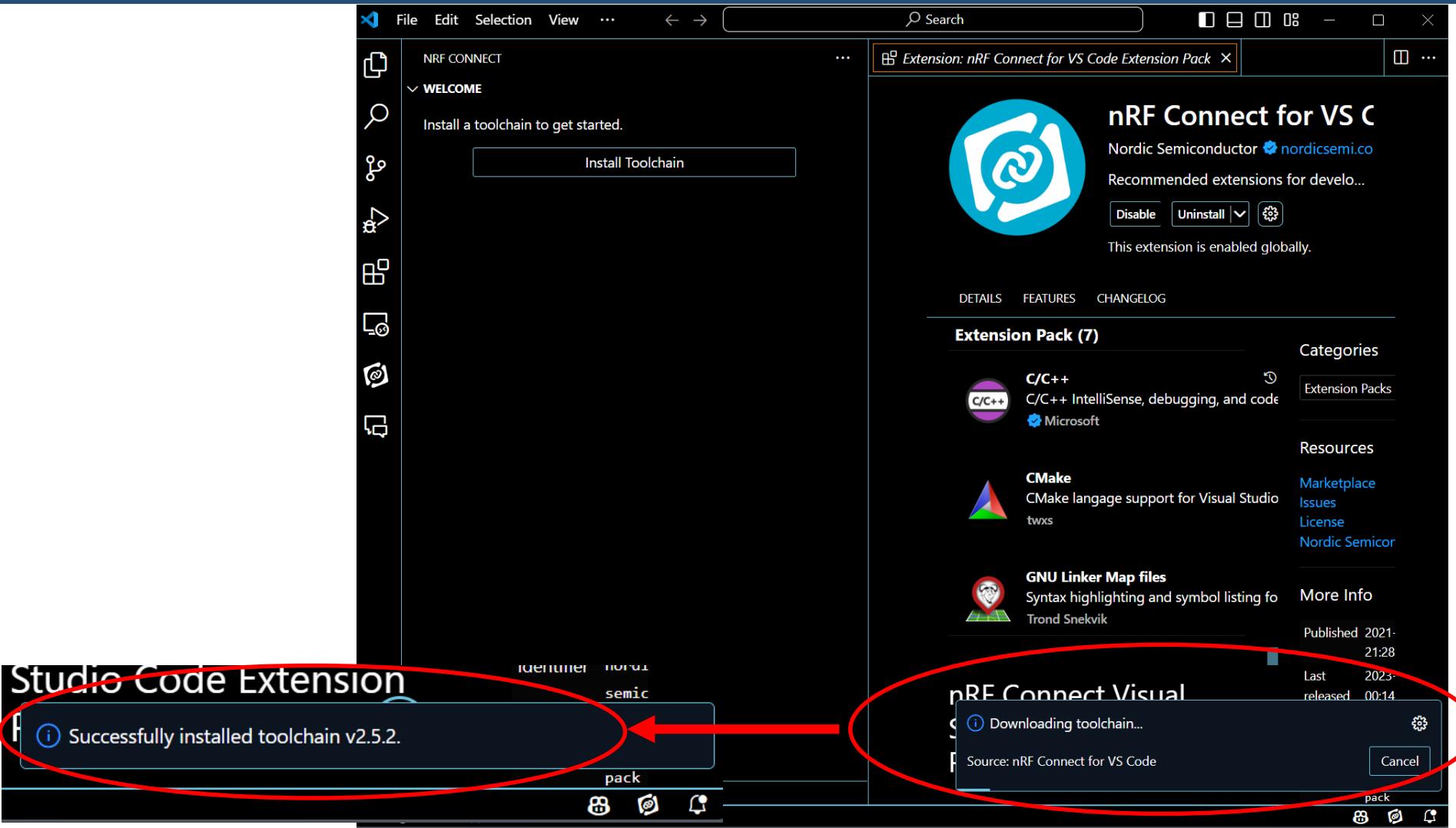


Install nRF Connect SDK toolchain

- Click nRF Connect SDK extension
 - ① in the Activity Bar
- Extension open Welcome View
 - ② Click “Install Toolchain”
- ③ Select toolchain version
 - recommend latest version



Install nRF Connect SDK toolchain



Tips

- If you meet errors as below picture while install toolchain
 - System can't find command line tools like jlink and nrfjprog
 - Make sure “command line tools” are correctly installed
 - Check environment like PATH

The screenshot shows a terminal window with the following log output:

```
[13:34:00] Update available https://nsscpromedia.blob.core.windows.net/prod/software-and-other-downloads/desktop-software/nrf-command-line-tools/sw/versions-10-x-x/10-24-0/nrf-command-line-tools_10.24.0_amd64.deb
[14:14:21] nrfutil-toolchain-manager install v2.5.2: Download toolchain
[14:21:40] nrfutil-toolchain-manager install v2.5.2: Toolchain downloaded: success
[14:21:40] nrfutil-toolchain-manager install v2.5.2: Unpack toolchain
[14:21:50] nrfutil-toolchain-manager install v2.5.2: Toolchain unpacked to /home/technonia/ncs/tmp/.tmpu1Blv6: success
[14:21:50] nrfutil-toolchain-manager install v2.5.2: Install toolchain
[14:21:50] nrfutil-toolchain-manager install v2.5.2: Toolchain installed at /home/technonia/ncs/toolchains/7795df4459: success
[14:21:50] ===== Toolchain validation report for nRF =====
! jlink is required but not found. Please ensure that SEGGER J-Link Software and Documentation Pack is installed.
! nrfjprog couldn't be executed. Please ensure that SEGGER J-Link Software and Documentation Pack and nRF Command Line Tools are both installed.
```

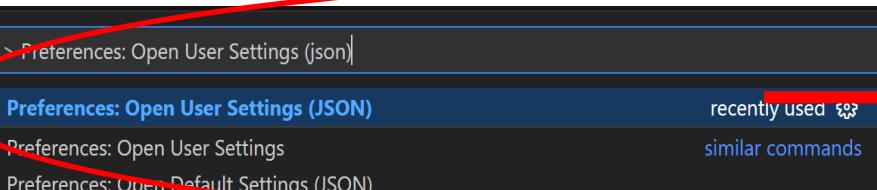
A red arrow points from the error message "jlink is required but not found" to the "jlink" entry in the list of circled errors. A red circle encloses the entire error section at the bottom of the log.

Tips

- If you meet errors as below picture while check nrfjprog version
 - VScode can't find nrfjprog command
 - Vscode setting.json input environment like PATH
 - How to open setting.json in vscode
 - ✓ Ctrl + shift + P with > Preferences: Open User Settings (json)
 - ✓ Add env variable PATH at the bottom of setting.json
- ("terminal.integrated.env.windows": { "PATH": "\${env:PROFILE}\\\\.cargo\\bin;\${env:PATH}" })

```
Microsoft Windows [Version 10.0.22631.3235]
(c) Microsoft Corporation. All rights reserved.

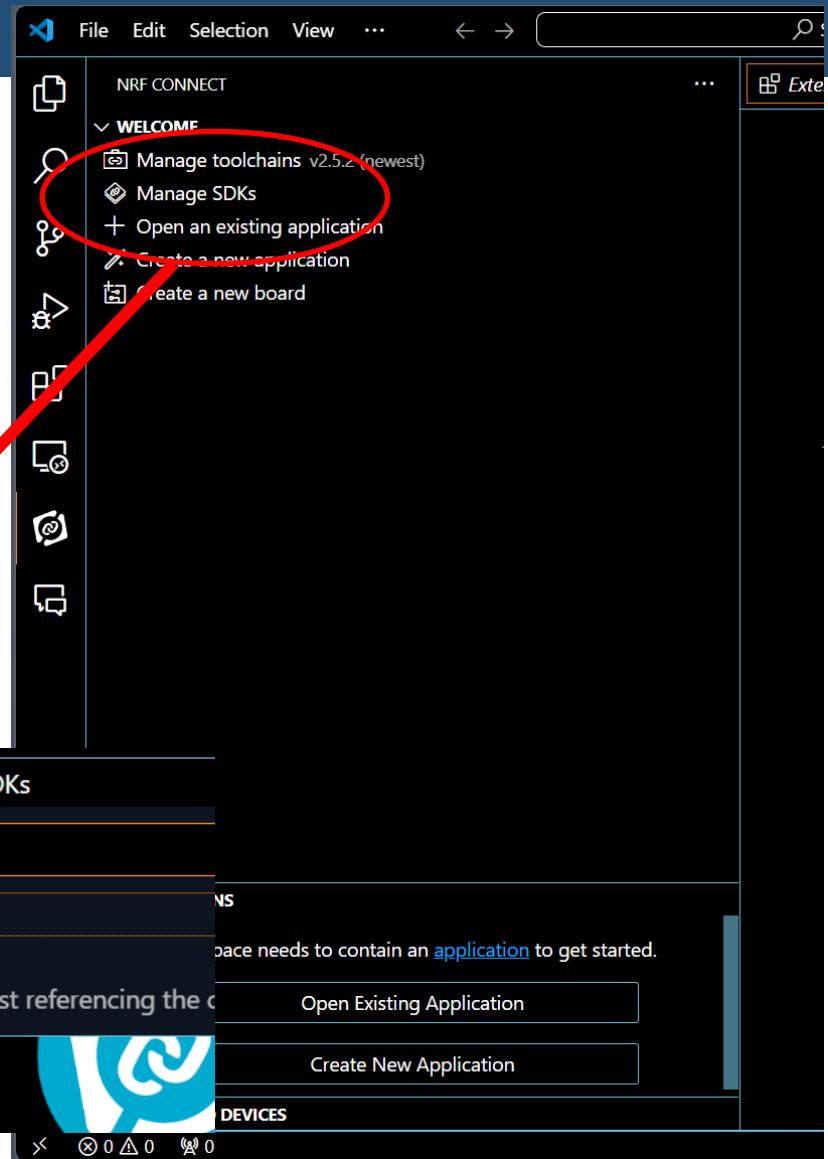
D:\devlop\zephyr>nrfjprog --version
'nrfjprog'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.
```



```
20
21
22
23
24 }      "nrf-connect.enableTelemetry": true,
           "terminal.integrated.env.windows": {
               "PATH": "${env:PROFILE}\\\\.cargo\\bin;${env:PATH}"
           }
```

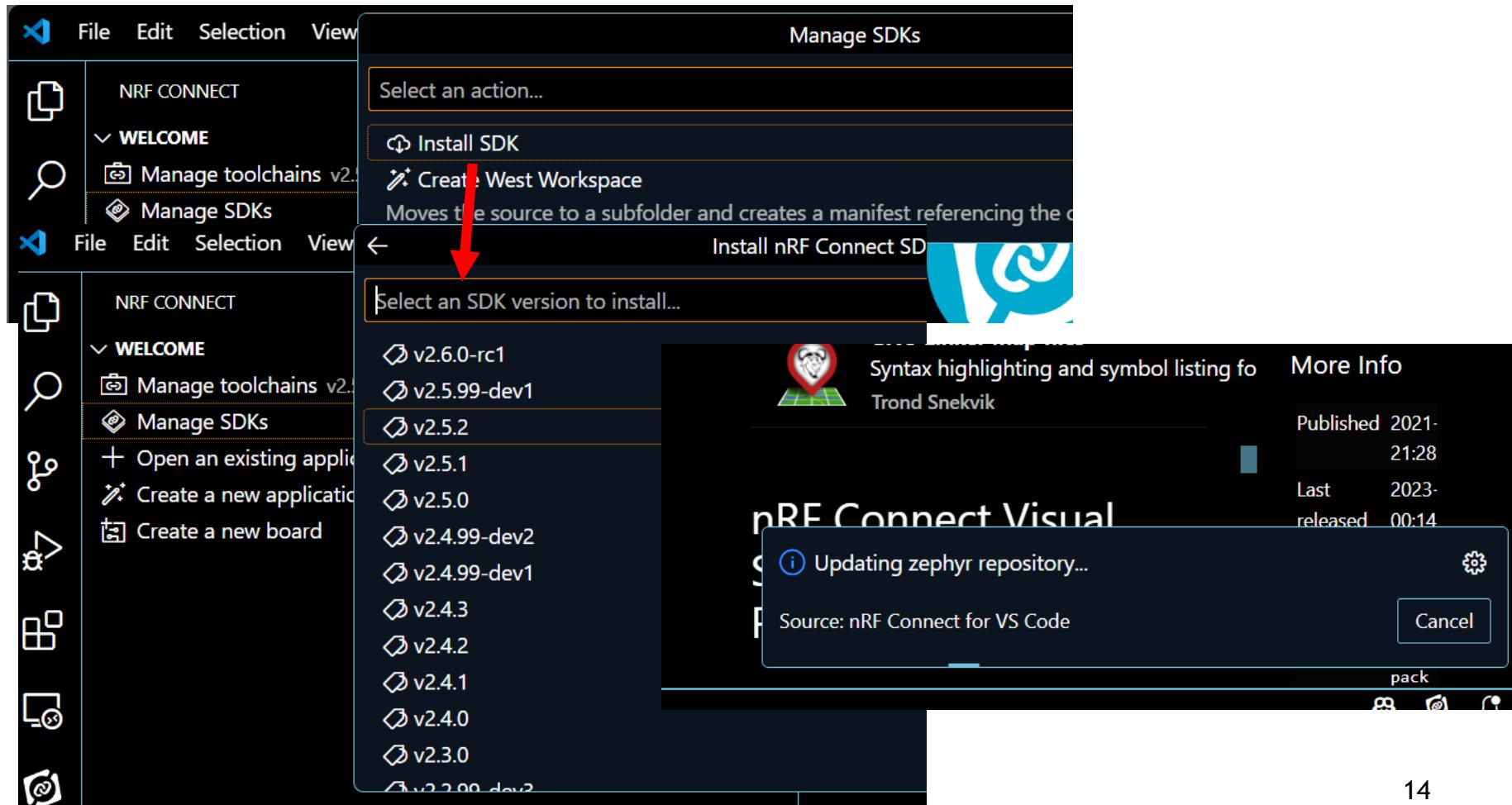
Get the nRF Connect SDK code

- If toolchain install is done well
 - Please find “Manage SDKs”
- Click “Manage SDKs”
 - Click “Install SDK”
 - Select SDK version in the list
 - ✓ Recommend latest version
 - Check and select folder name
 - ✓ Now, you can see “west” status



Get the nRF Connect SDK code

- Actually, it's a download from git repositories with "west"



Wrapup : nRF Connect SDK install

- VSCode with nRF Connect extension
 - Use VSCode as IDE
 - nRF Connect extension for building firmware
 - ✓ Prerequisites : command line toolchain

```
<home>/  
  └── toolchains/  
      └── <toolchain-installation>  
          └── <west-workspace>/  
              ├── .west/  
              ├── bootloader/  
              ├── modules/  
              ├── nrf/  
              ├── nrfxlib/  
              ├── zephyr/  
              └── ...
```

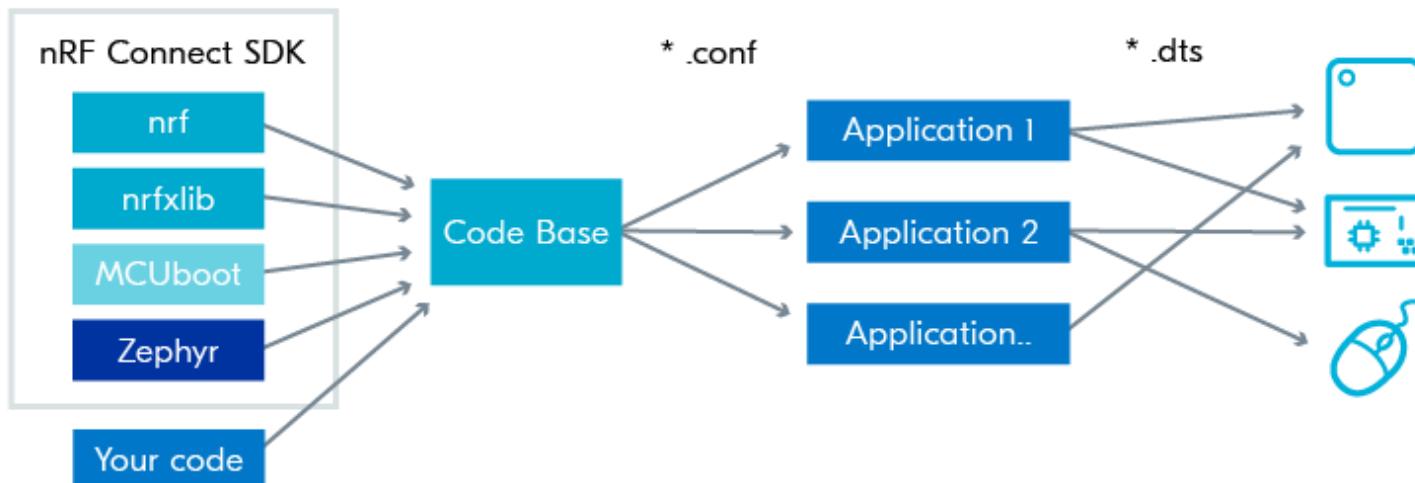
| | |
|---------|--|
| nrf | Applications, samples, connectivity protocols (Nordic) |
| nrfxlib | Common libraries and stacks (Nordic) |
| zephyr | RTOS & Board configurations (open source) |
| MCUBoot | Secure Bootloader (open source) |

<nRF Connect SDK code repositories>

<Your directory structure looks like this>

Toolchain in nRF Connect SDK

- Portability and maintainability
 - The high decoupling of source (*.c), configuration (*.conf) and hardware description (devicetree, *.dts).
 - Same application source code on different hardware and with different configurations with minimal changes.



Zephyr Application

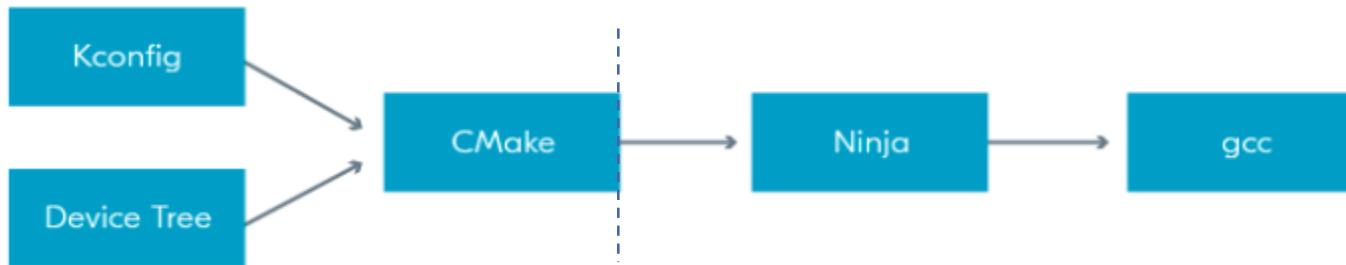
- Simplest application directory

```
<app>
  └── CMakeLists.txt
  └── app.overlay
  └── prj.conf
  └── VERSION
    └── src
      └── main.c
```

- **CMakeLists.txt**
 - Names and location of application files
- Kernel configuration (**prj.conf**)
 - Application specific kernel options
- “src” folder contains application codes
- More details
 - https://developer.nordicsemi.com/nRF_Connect_SDK/doc/2.5.2/zephyr/develop/application/index.html

Zephyr Application

- Zephyr Build Process has two phases
 - Configuration Phase
 - ✓ Begins when user invokes Cmake (or Click build button in VSCode)
 - ✓ Input : Source Location (CMakeLists.txt) and Board Target (*.dts) Kconfig file (prj.conf)
 - ✓ Output : build files, header files (devicetree.h from devicetree), config file (autoconf.h from prj.conf)
 - Build Phase
 - ✓ Use arm-gcc as cross compiler
 - ✓ Output : zephyr.elf, zephyr.hex, zephyr.bin
 - You can find files in <application_folder>/build/zephyr
 - Input to Flash tools



Before going to Lab

- Github for Lab
 - Document, sources and others
 - <https://github.com/UmileVX/IoT-Development-with-Nordic-Zephyr>
- Access more instructions in Nordic's official tutorial
 - <<https://academy.nordicsemi.com/courses/nrf-connect-sdk-fundamentals/lessons/lesson-1-nrf-connect-sdk-introduction/topic/exercise-1-1/>>

The screenshot shows a course interface for 'nRF Connect SDK Fundamentals'. On the left, there's a sidebar with 'DevAcademy' logo, 'nRF Connect SDK Fundamentals' title, 'Lesson 1 – nRF Connect SDK Introduction' (with 3 Topics and 1 Quiz), and a list of exercises: 'Exercise 1' (selected), 'Exercise 2', and 'Lesson 1 Quiz'. The main content area is titled 'Exercise 1' and contains a 'Navigation' section. It provides instructions for selecting the correct tab based on the chosen Connect SDK version: 'v2.x.x' for v2.0.0 and above (default), and 'v1.6.0 – v1.9.1' for versions between v1.6.0 and v1.9.1. It also advises completing all topics and quizzes for lesson completion.

DevAcademy

nRF Connect SDK Fundamentals

Lesson 1 – nRF Connect SDK Introduction

3 Topics | 1 Quiz

nRF Connect SDK structure and content

Exercise 1

Exercise 2

Lesson 1 Quiz

Navigation

I. Before proceeding with the topic, first select the tab that matches your chosen Connect SDK version:

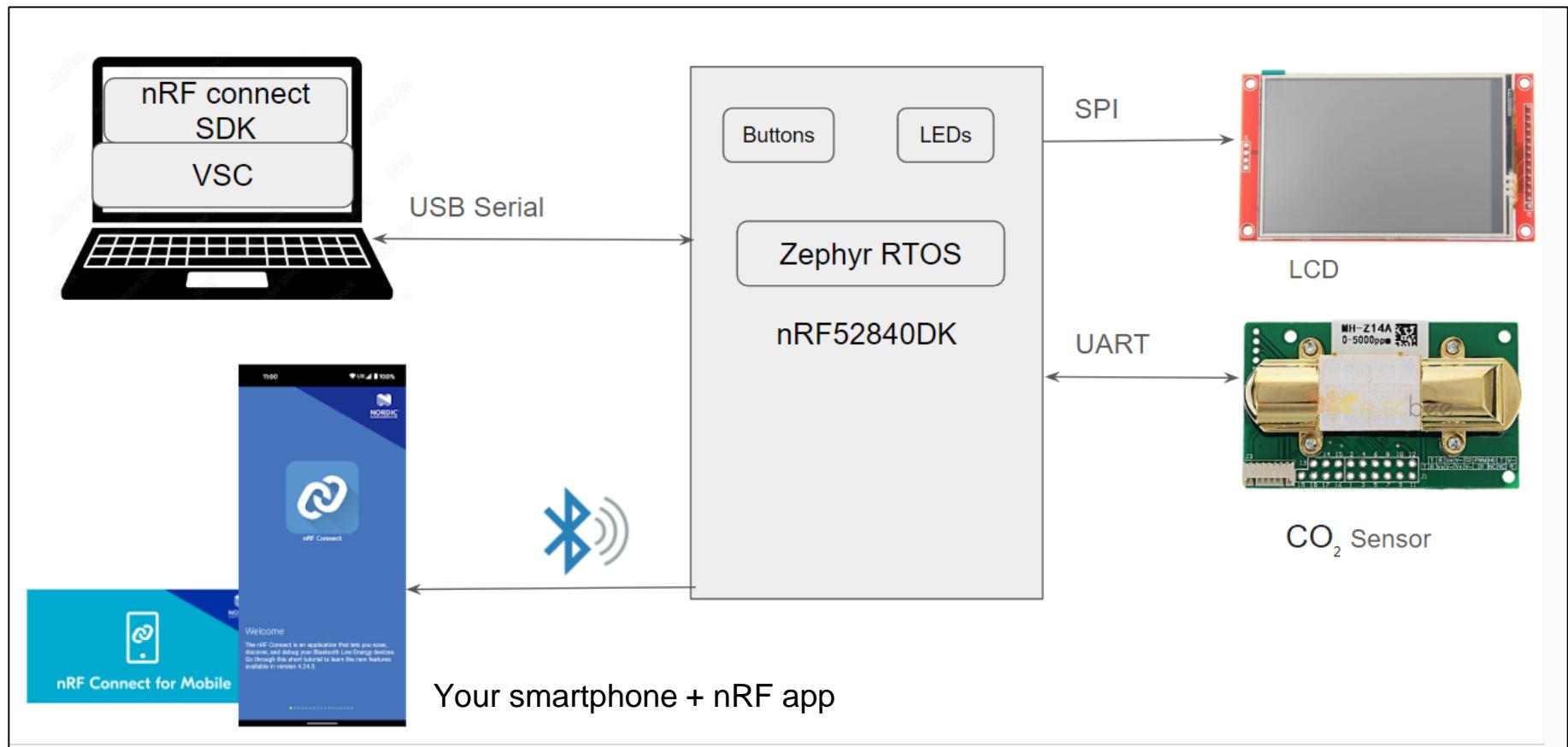
- For v2.0.0 and above (default), use the tab: **v2.x.x**
- For any version between v1.6.0 and v1.9.1, use the tab: **v1.6.0 – v1.9.1**

2. When progressing through the lessons, make sure to complete all the topic quizzes. The quiz is considered to be the last step in lesson completion. If the lessons are in a different order, make sure to re-take the quiz to mark the lesson as completed.

v2.x.x v1.6.0 – v1.9.1

Lesson 2 – Reading buttons and controlling LEDs

Lab Environment



Hardware Checklist

- Dev Kit
 - nRF52840 DK board
 - MH-Z14A sensor (CO₂ sensor)
 - USB-to-serial cable x 2
 - Jump wire bundle
 - Sensor shield

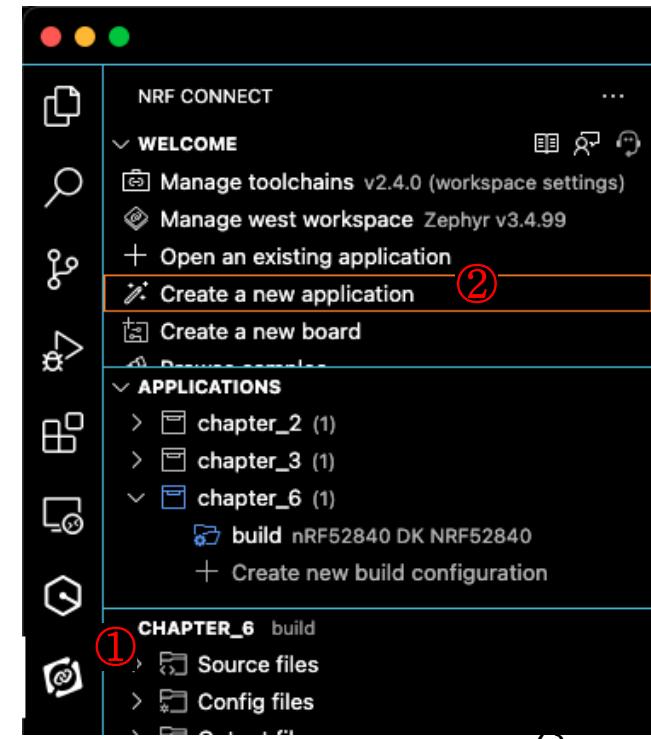


Run Example

- Build and Flash your first nRF Connect SDK application
- Blinky example
 - Toggle an LED on our board
- By running this example, you will be able to:
 - Create a new application based on a template(sample)
 - Build an application
 - Flash an application to a board

Run Example

- Create a folder that will hold all the exercises
 - For example c:\mywork
 - Please avoid path is too long or includes “Hangul”
 - The build system might fail
- Select VSCode nRF Connect extension
 - Click the “Create a new application”



Run Example

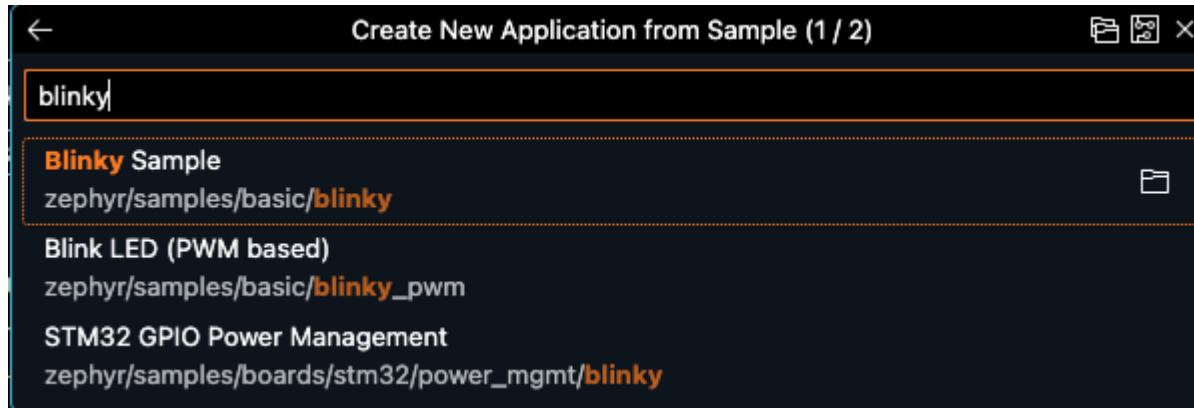
- Select “Copy a sample”



- Create a blank application : create with empty main()
- Copy a sample : show you all the template “samples”
 - From different modules in nRF Connect SDK

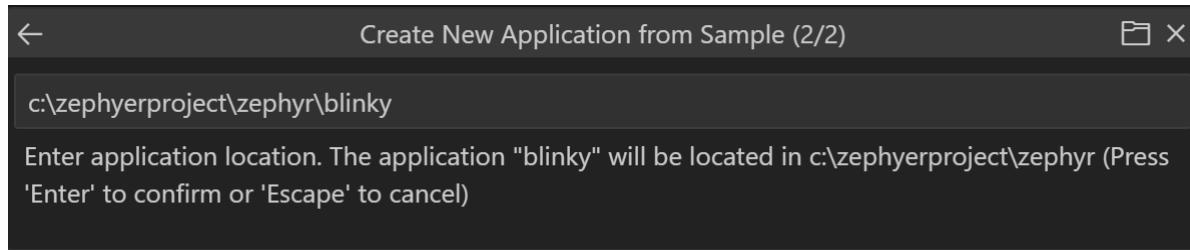
Run Example

- Enter the name of the sample project you want to copy
 - In this time, we are going to use “Blinky Sample”
 - Blink samples comes from zephyr module
 - zephyr/samples/basic/blinky

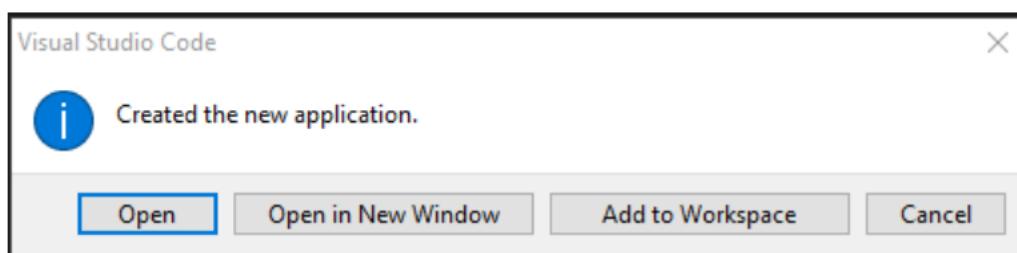


Run Example

- Configure the path of the new sample project
 - Select where you want to store your application

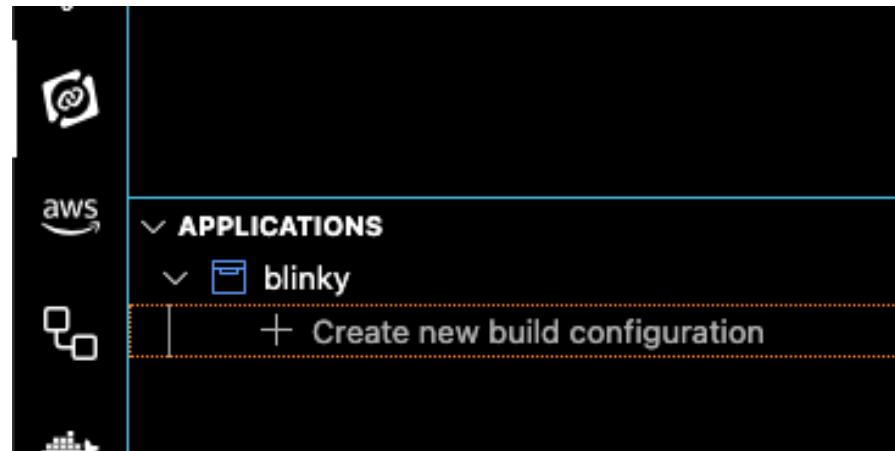


- VSCode will ask you how to open the application
 - Select “Open” to open it in the same VSCode instance.



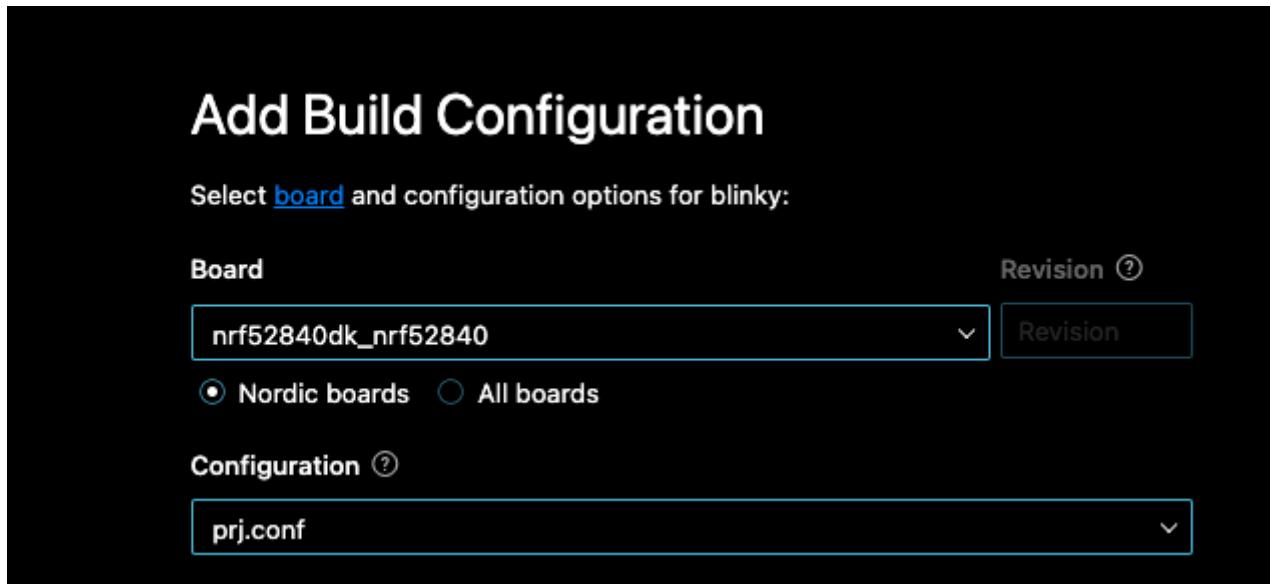
Run Example

- Create the build configuration
 - by clicking “Create new build configuration”
- This step will specify which board we use.
- Also will select the software configuration



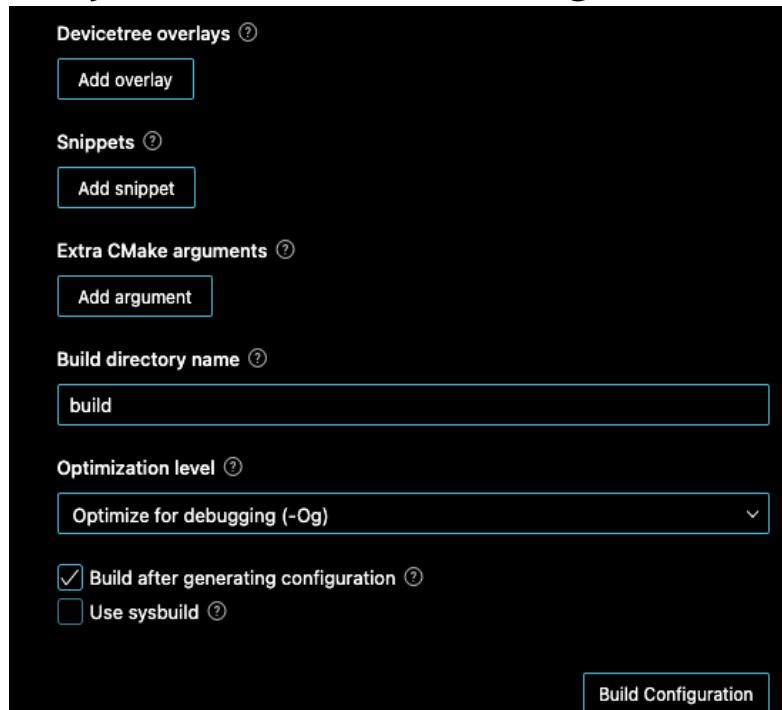
Run Example

- Select the target board (nrf52840dk_nrf52840)
- Select the target configuration file (prj.conf)



Run Example

- The default name of the build directory is “build”
- Select the Optimization level that you want to use
- “Build after generating configuration” enables to trigger build process.
- Once you are ready, click “Build Configuration”



Run Example

- Open the nRF Connect Terminal (View->Terminal)
 - To see the progress of the build
 - Successful build displays application's memory usage

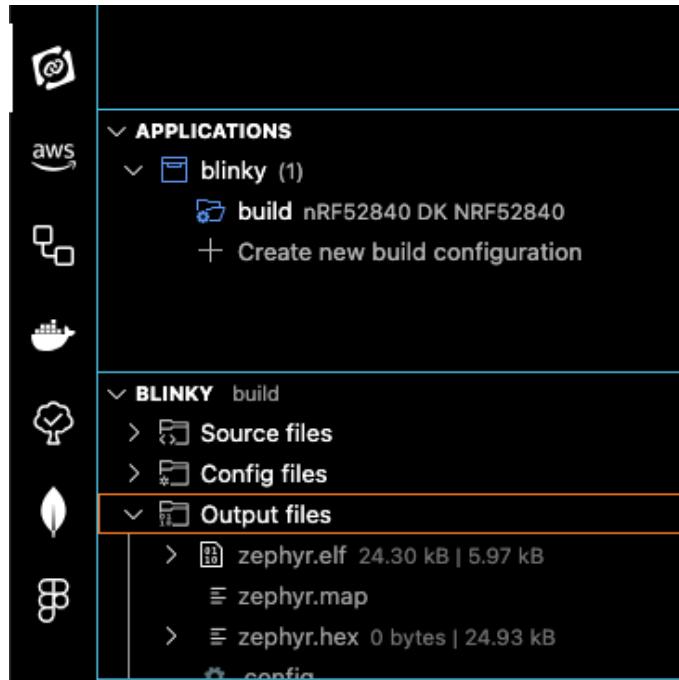
```
[153/164] Linking C static library zephyr\kernel\libkernel.a
[154/164] Linking C executable zephyr\zephyr_pre0.elf

[155/164] Generating dev_handles.c
[156/164] Building C object zephyr\CMakeFiles/zephyr_pre1.dir/misc/empty_file.c.obj
[157/164] Building C object zephyr\CMakeFiles/zephyr_pre1.dir/dev_handles.c.obj
[158/164] Linking C executable zephyr\zephyr_pre1.elf

[159/164] Generating linker.cmd
[160/164] Generating isr_tables.c, isrList.bin
[161/164] Building C object zephyr\CMakeFiles/zephyr_final.dir/misc/empty_file.c.obj
[162/164] Building C object zephyr\CMakeFiles/zephyr_final.dir/dev_handles.c.obj
[163/164] Building C object zephyr\CMakeFiles/zephyr_final.dir/isr_tables.c.obj
[164/164] Linking C executable zephyr\zephyr.elf
Memory region      Used Size  Region Size %age Used
          FLASH:    19652 B     512 KB   3.75%
          RAM:      5568 B     128 KB   4.25%
        IDT_LIST:       0 GB      2 KB   0.00%
* Terminal will be reused by tasks, press any key to close it.
```

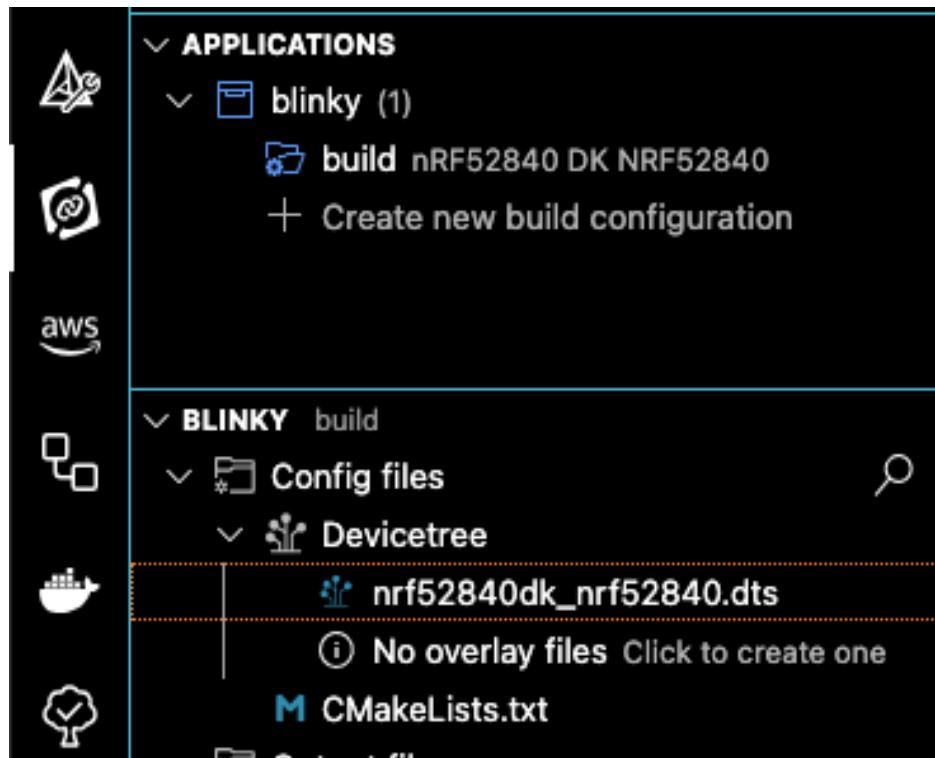
Run Example

- Once you successfully build the project, you will be able to see the generated files
- You could look up all generated outputs easily by using nRF Connect extension



Run Example

- You could also check the device tree file via nRF Connect extension



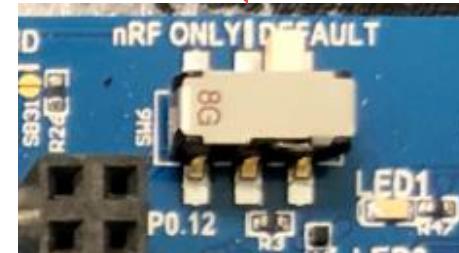
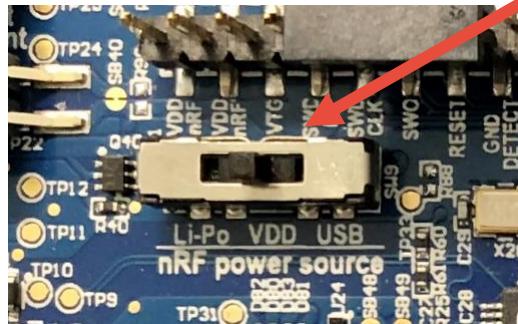
Run Example

- Connect the device to the laptop via usb-serial cable
- Check if the connected device is visible on the “connected devices” list



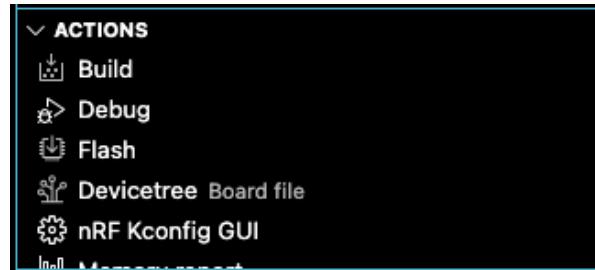
Run Example

- If the device fail to connect
- Check nRF power source switch is on the VDD
- Check nRF ONLY|DEFAULT switch is on DEFAULT



Run Example

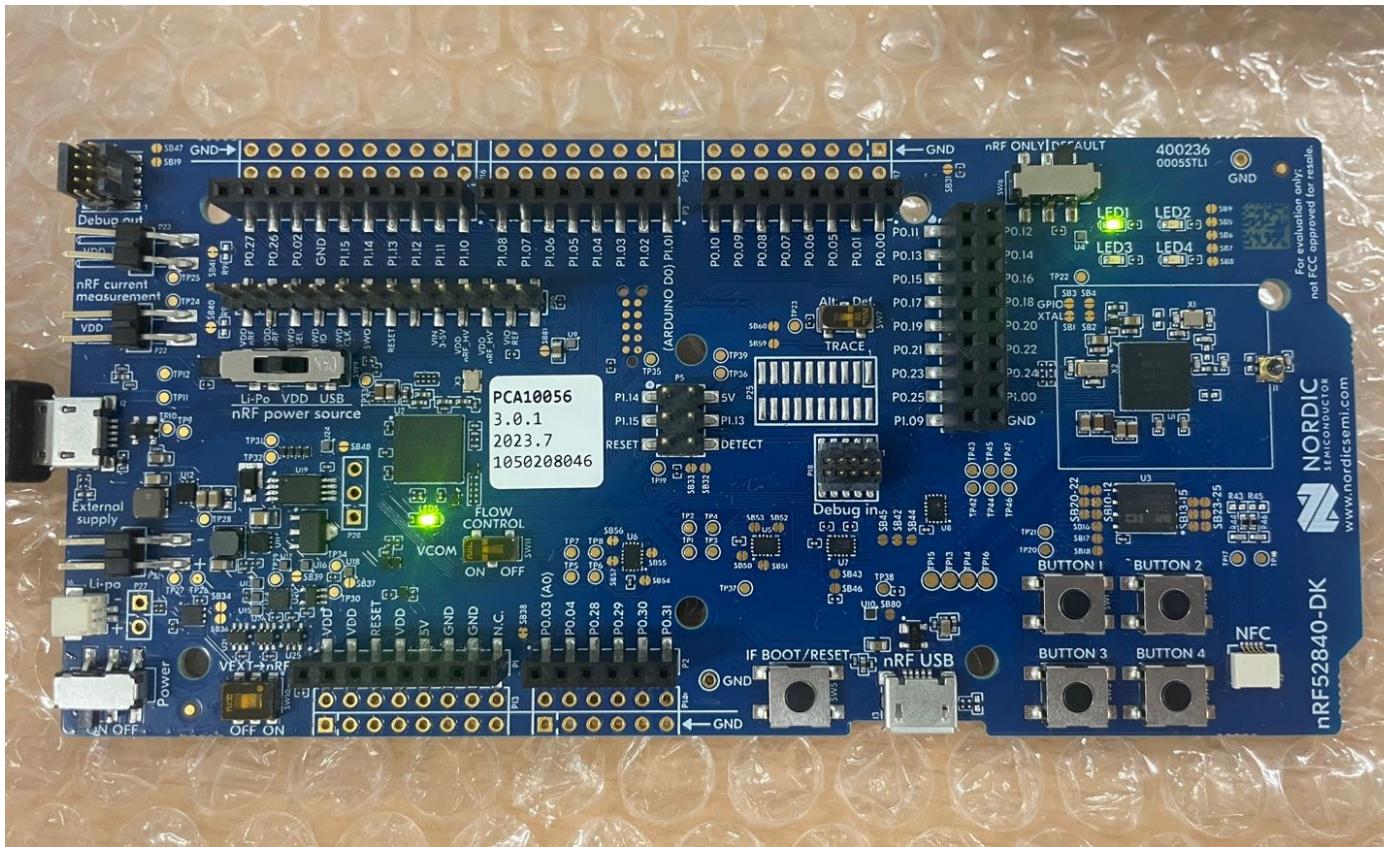
- Once the device is connected successfully, flash the hex file
 - use GUI button rather using CLI for convenience
- You can see a LED is blinking on SDK board.



```
west flash -d /Users/yeonwoosung/Desktop/blinky/build --skip-rebuild --dev-id 1050286265
-- west flash: using runner nrfjprog
-- runners.nrfjprog: Flashing file: /Users/yeonwoosung/Desktop/blinky/build/zephyr/zephyr.hex
[ ##### ] 0.792s | Erase file - Done erasing
[ ##### ] 0.171s | Program file - Done programming
[ ##### ] 0.171s | Verify file - Done verifying
Enabling pin reset.
Applying pin reset.
-- runners.nrfjprog: Board with serial number 1050286265 flashed successfully.
```

Run Example

- You can see a LED is blinking on SDK board.



LAB : use Command Line

- Build the Blinky Sample
 - cd c:\zephyrproject\zephyr
 - west build --build-dir c:/zephyrproject/zephyr/blinky/build c:/zephyrproject/zephyr/blinky
- Flash the sample
 - west flash -d c:\zephyrproject\zephyr\blinky\build --skip-rebuild --dev-id <your-board-name>
 - Quiz : what should be written at <your-board-name>?

```
[ #####          ] 0.000s | Verify file - Check image
[ #####          ] 0.000s | Check image validity - Initialize device info
[ #####          ] 0.000s | Check image validity - Check region 0 settings
[ #####          ] 0.427s | Check image validity - block 1 of 2
[ #####          ] 0.007s | Check image validity - Finished
[ #####          ] 0.000s | Verify file - Verifying
[ #####          ] 0.000s | Verifying image - block 1 of 1
[ #####          ] 0.000s | Verifying image - Verify successful
[ #####          ] 0.186s | Verify file - Done verifying

Enabling pin reset.
Applying pin reset.
-- runners.nrfjprog: Board with serial number 1050208046 flashed successfully.
```