CHAPTER T0

Prepare to Develop

한동대학교-

테크노니아

마이크로프로
세서응용

Contents

Agenda

- nRF Connect SDK
- Develop Zephyr application
- Check Tools and Board
- Build and Flash Application

nRF Connect SDK

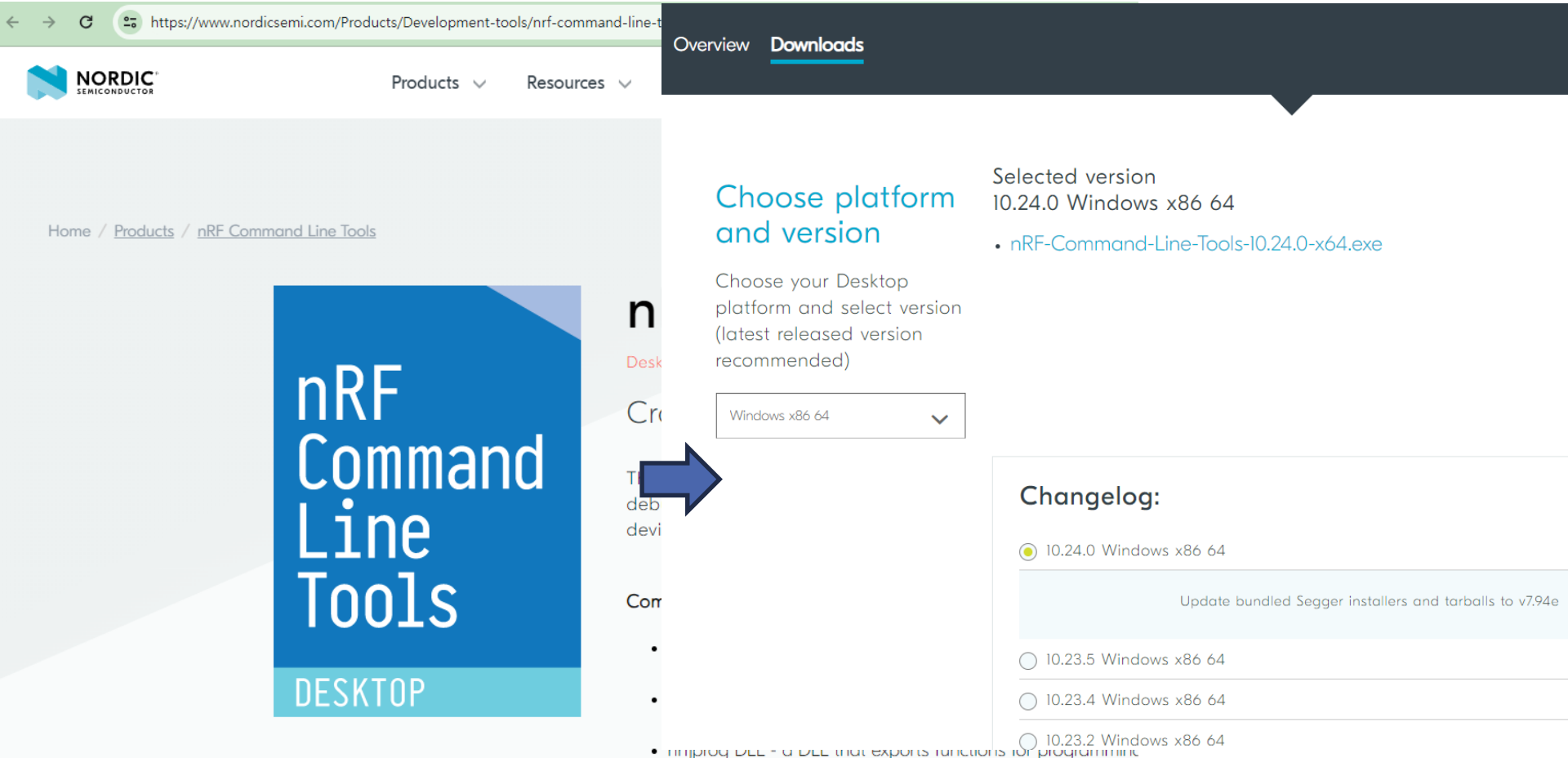
- nRF Connect SDK
  - For Nordic Semicoductor nRF52, nRF53, nRF70 and nRF91.
  - Supports Microsoft Windows, Linux, macOS
- nRF Connect SDK features
  - Based on Zephyr RTOS and open source
    - ✔ https://developer.nordicsemi.com/nRF_Connect_SDK/doc/2.5.2/zephyr/introduction/index.html
    - ✔ https://developer.nordicsemi.com/nRF_Connect_SDK/doc/2.5.2/nrf/releases_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
    - ✔ **Recommneded : we go this way!!**
  - With command line and nRF Util
- Install Prerequisities
  - 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

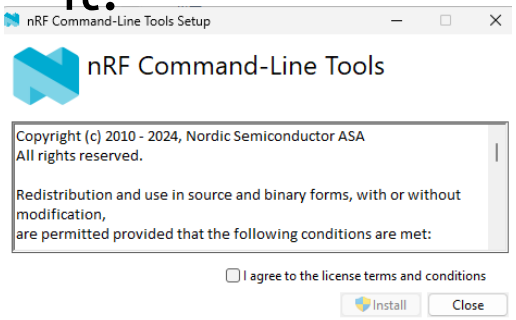
Installing nRF Connect SDK

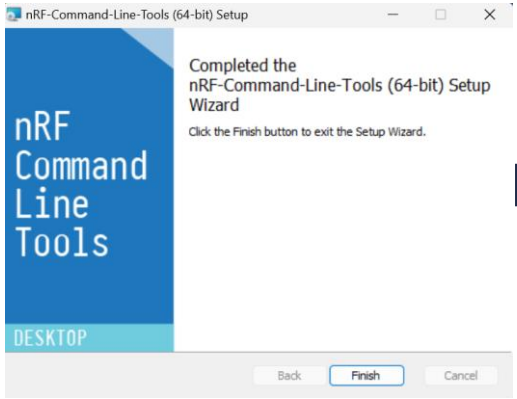
- nRF Command Line Tools
  - nrfjprog : tool for programming and debug

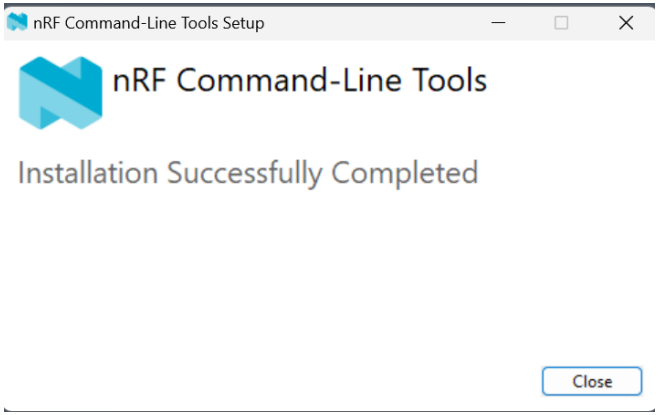


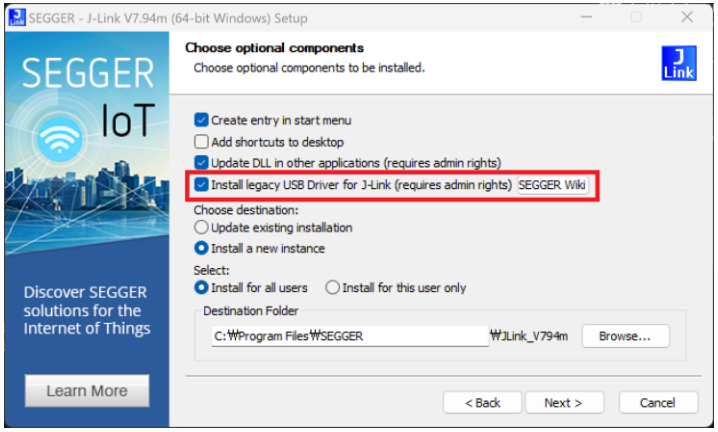
Installing nRF Connect SDK

- Download install file and run it.



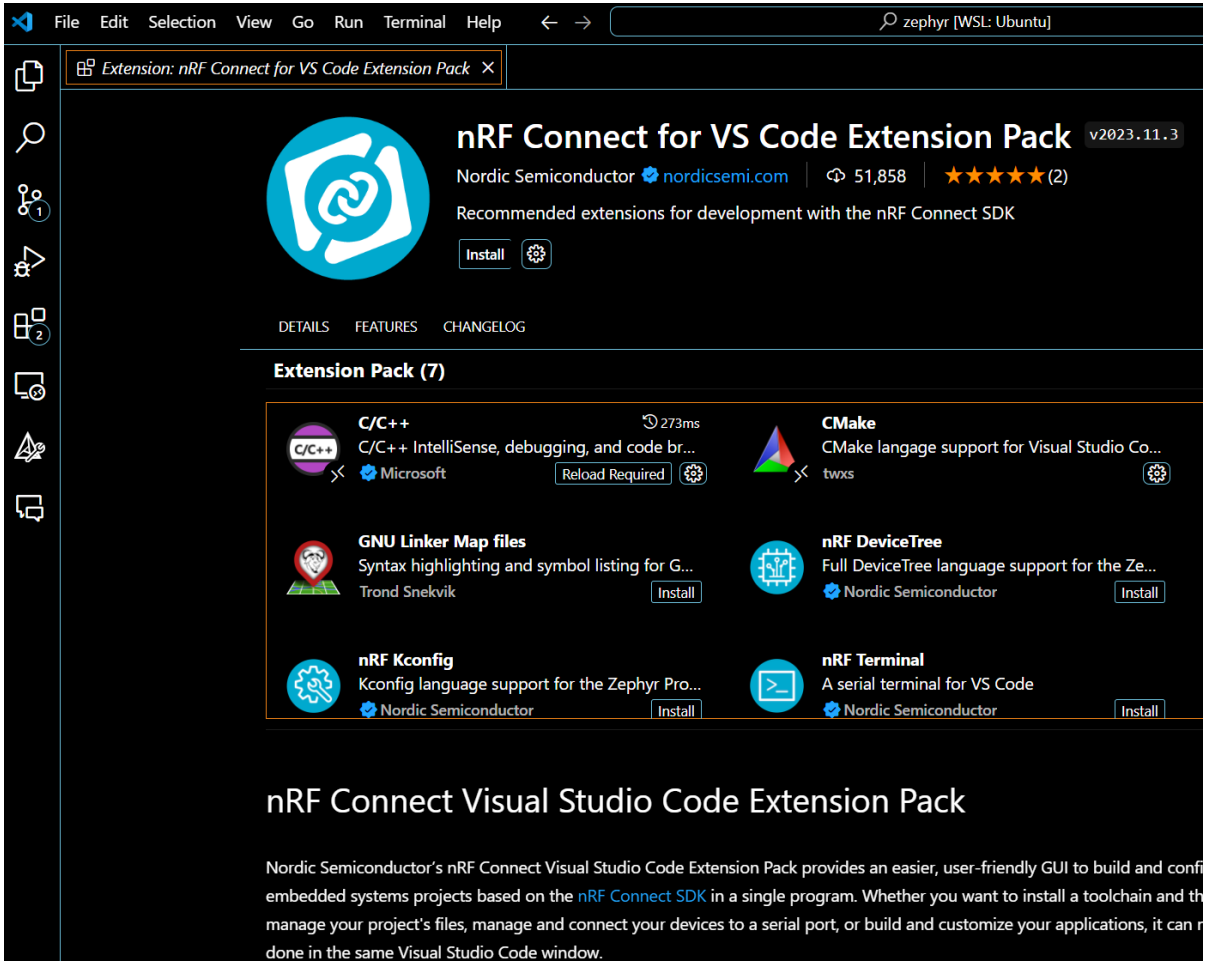






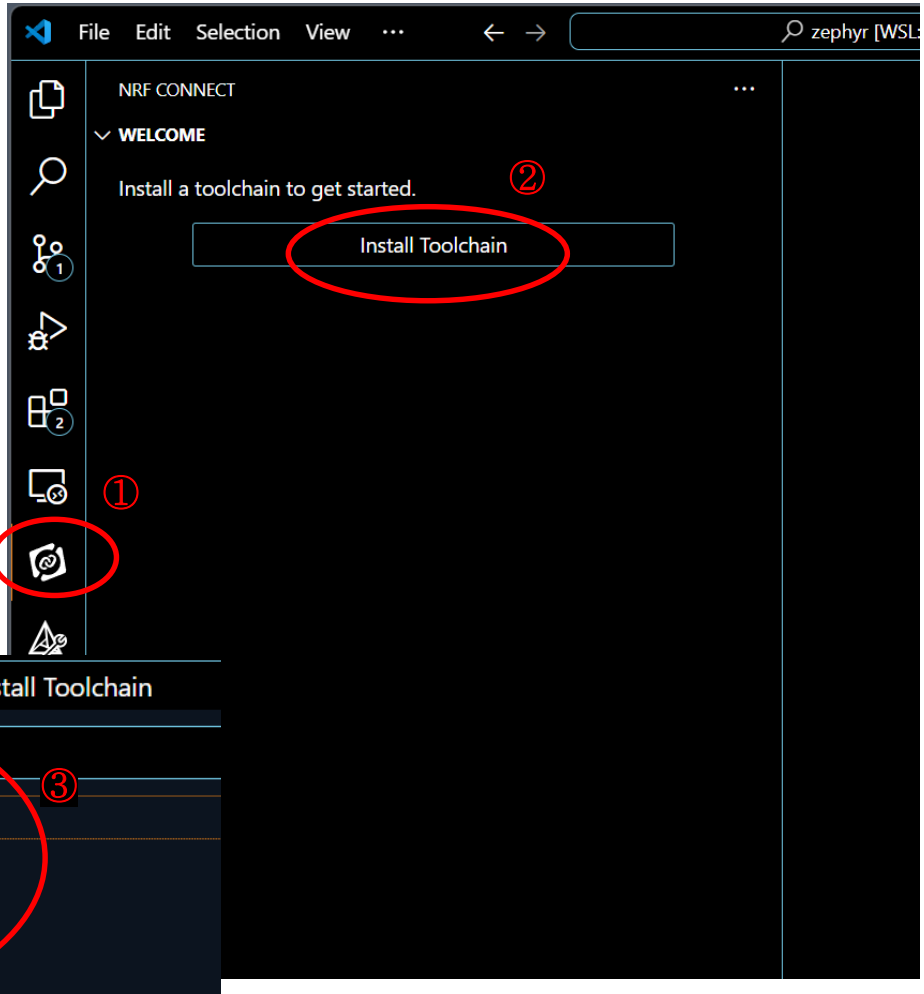
Installing nRF Connect SDK

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

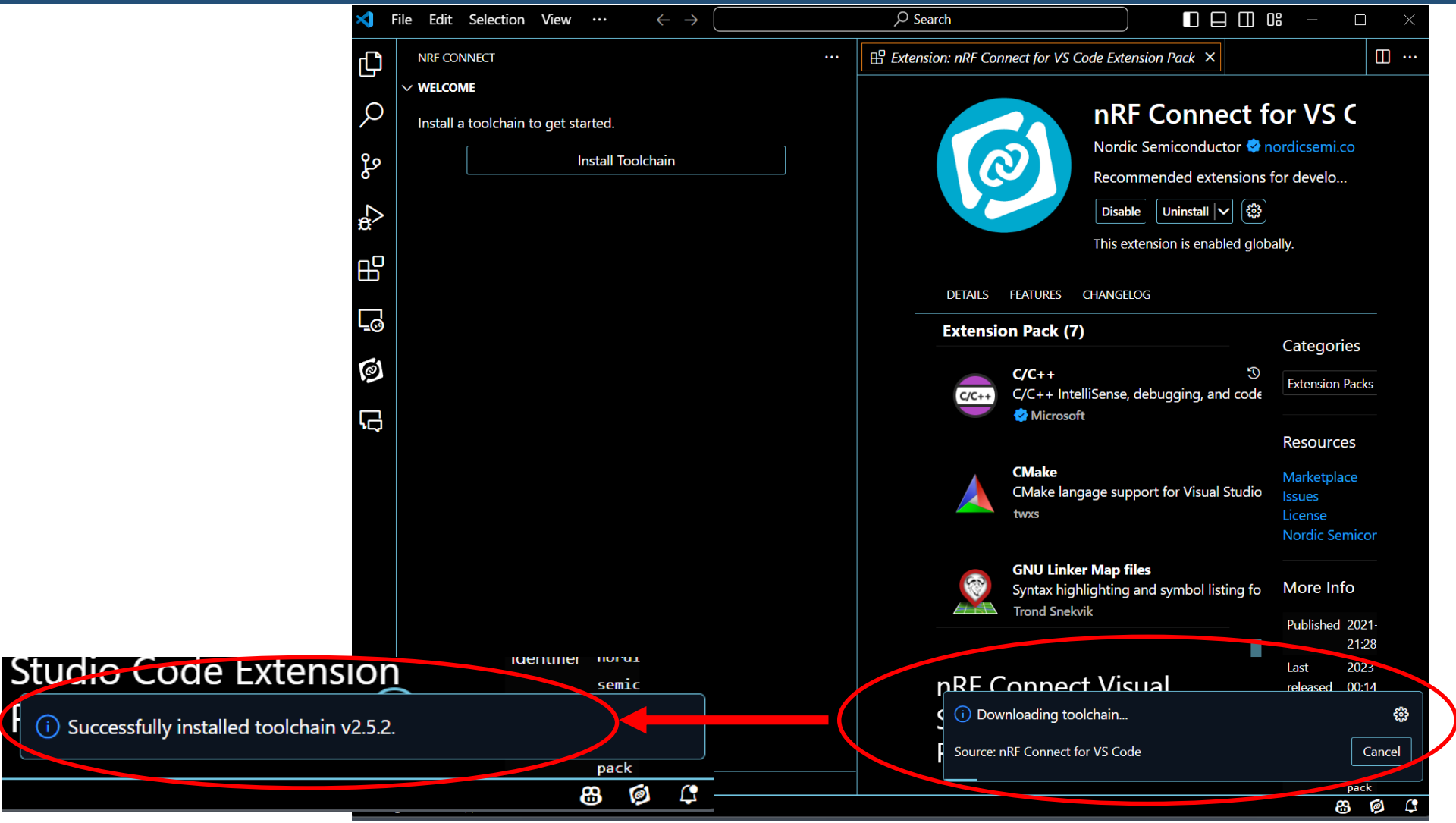


Install nRF Connect SDK toolchain

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

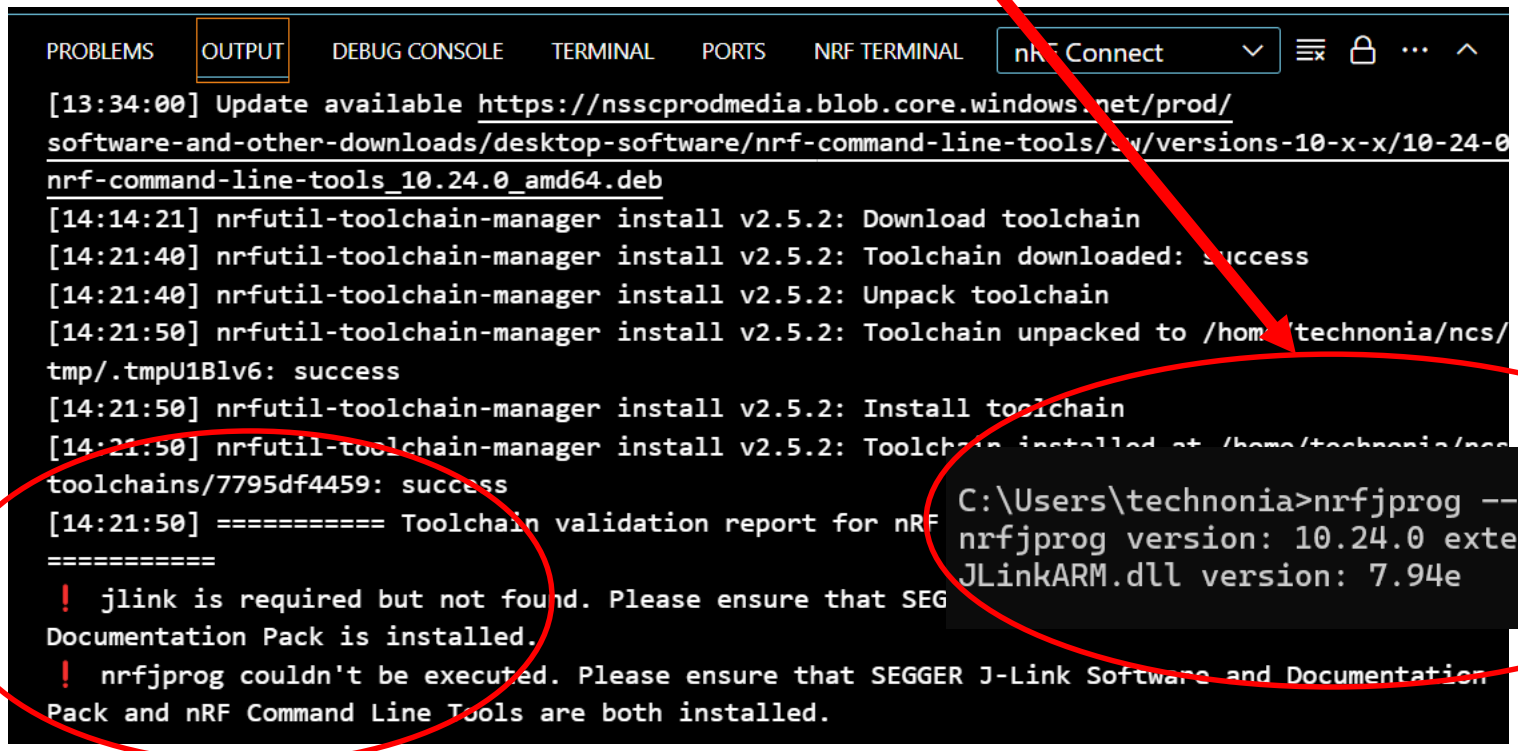


Install nRF Connect SDK toolchain



Tips

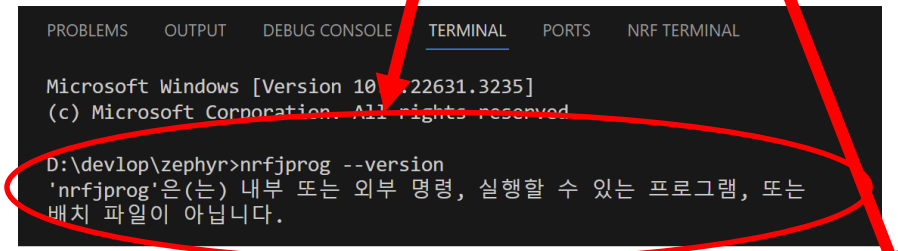
- If you meet errors as below picture while install toolchain
  - System can’t find command line tools like jlink and nrfjprog
  - Make sure “command line tools” are correctly installed
  - Check environment like PATH

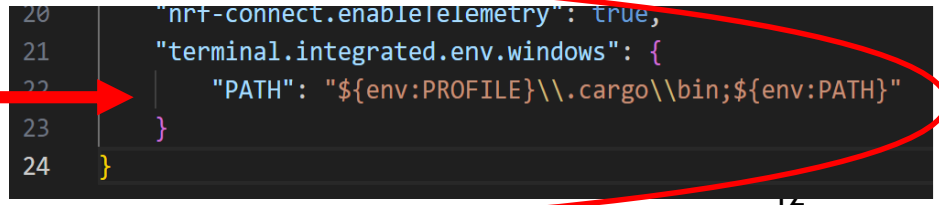


Tips

- 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

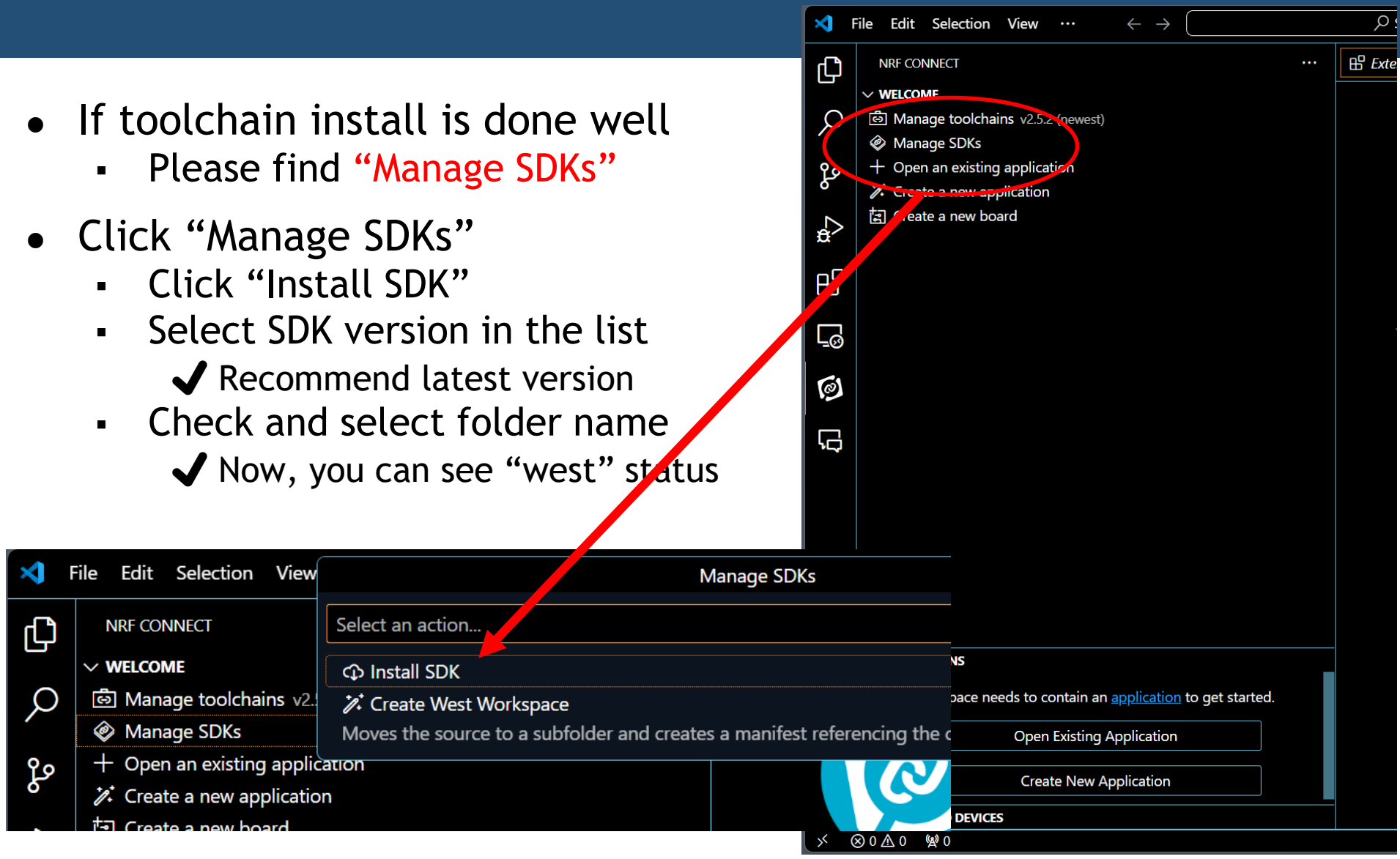
("terminal.integrated.env.windows": { "PATH": "${env:PROFILE}\\.cargo\\bin;${env:PATH}" })





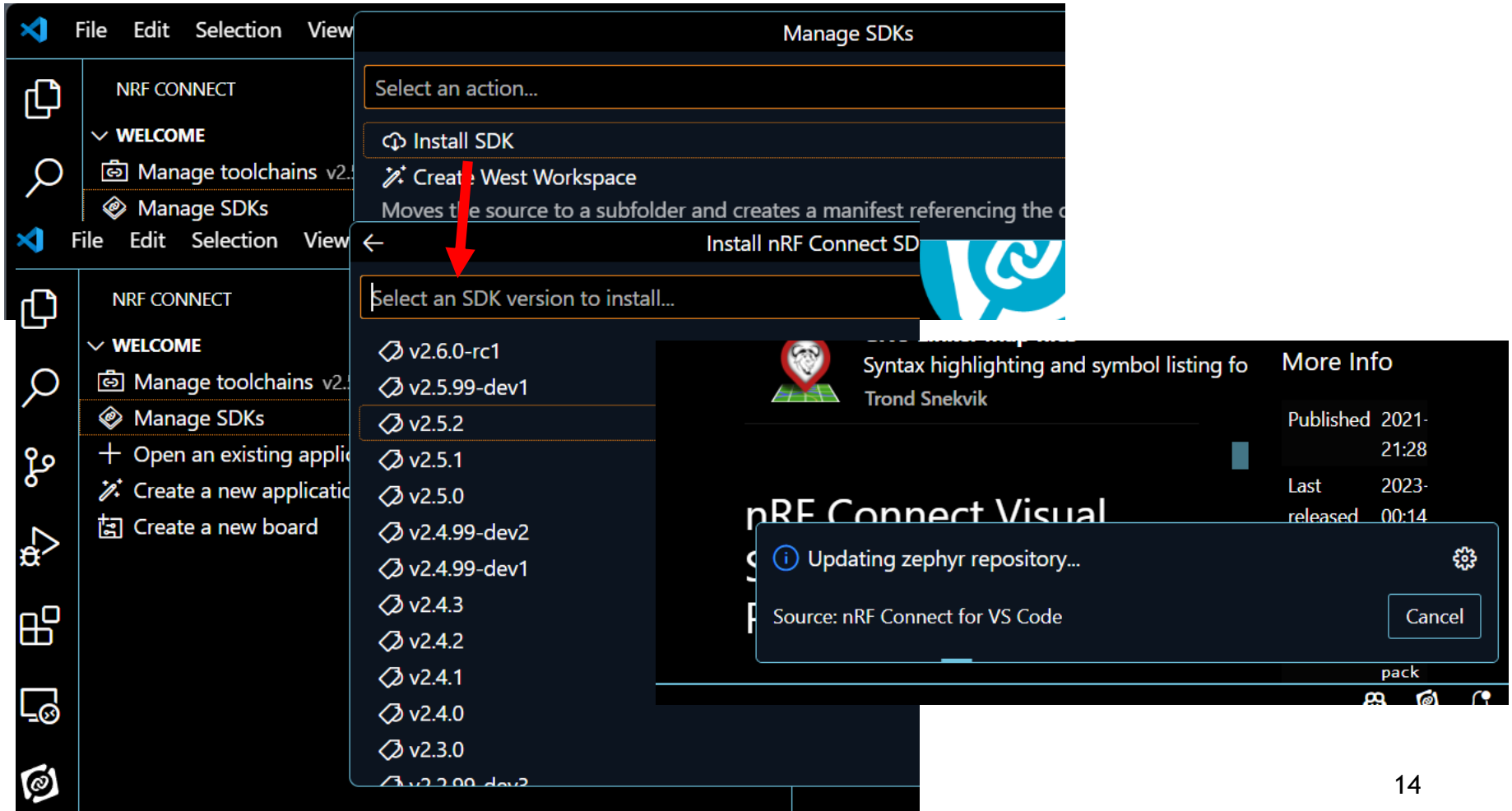
Get the nRF Connect SDK code

- 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



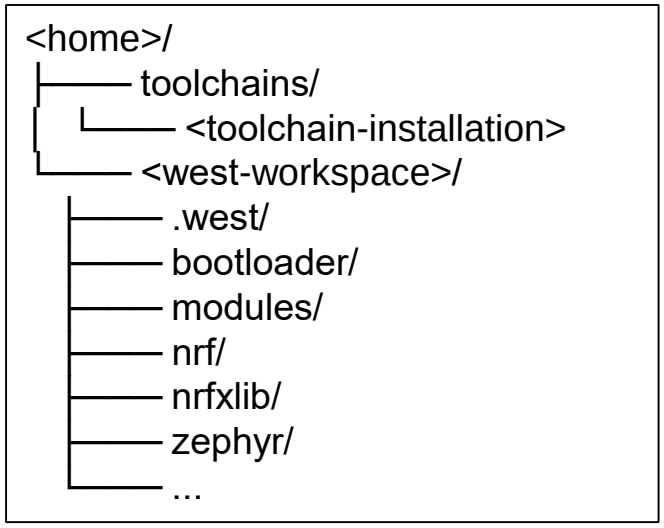
Get the nRF Connect SDK code

- 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
    - ✔ Prerequisities : command line toolchain



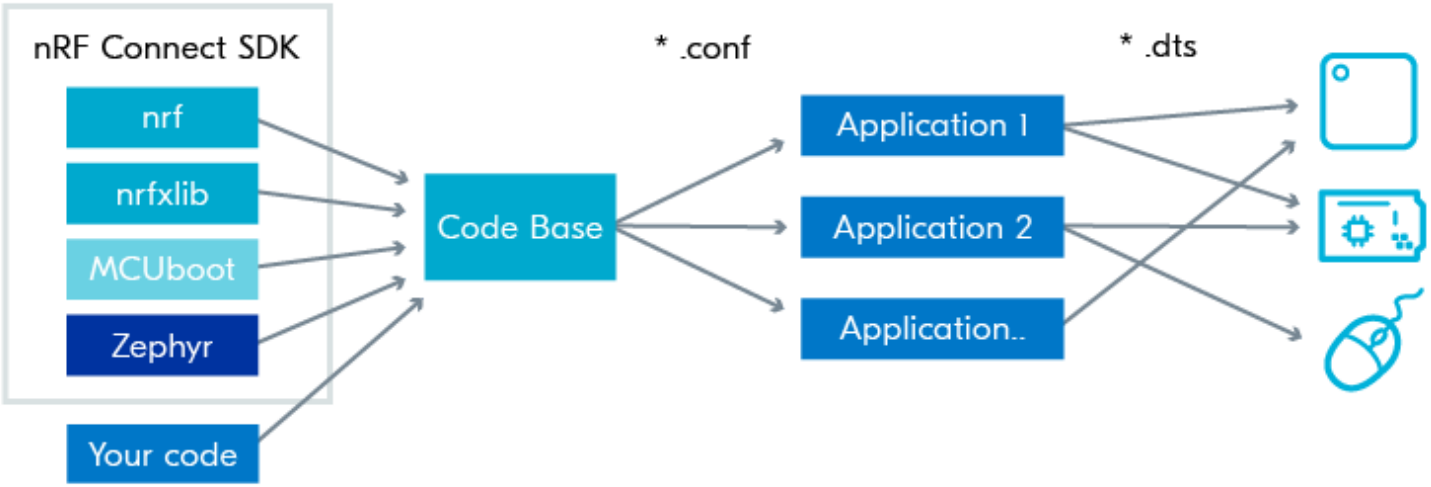
<Your directory structure looks like this>

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>

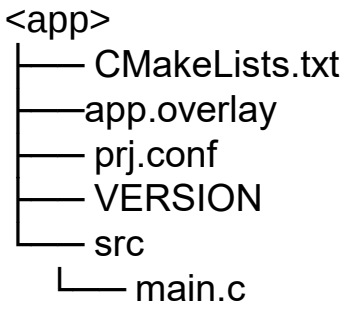
Toolchain in nRF Connect SDK

- 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

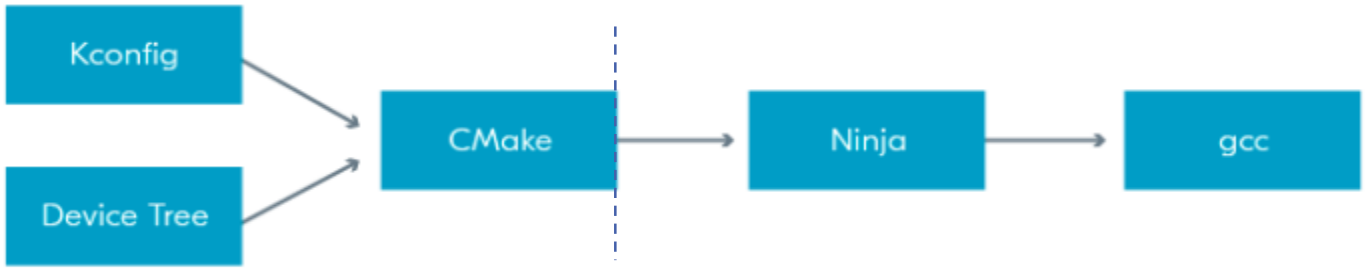
- Simplest application directory



  - 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

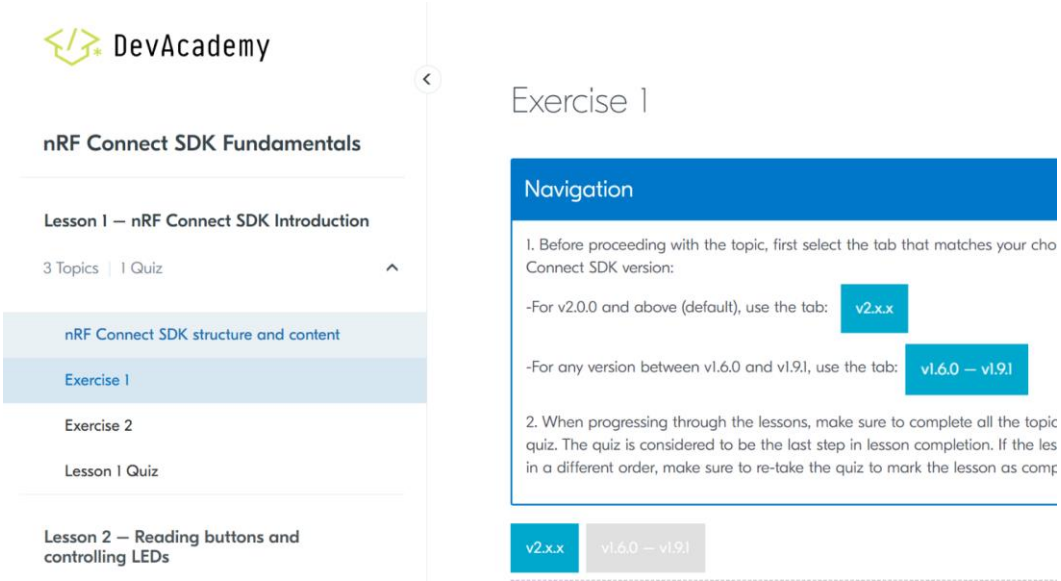
Zephyr Application

- 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

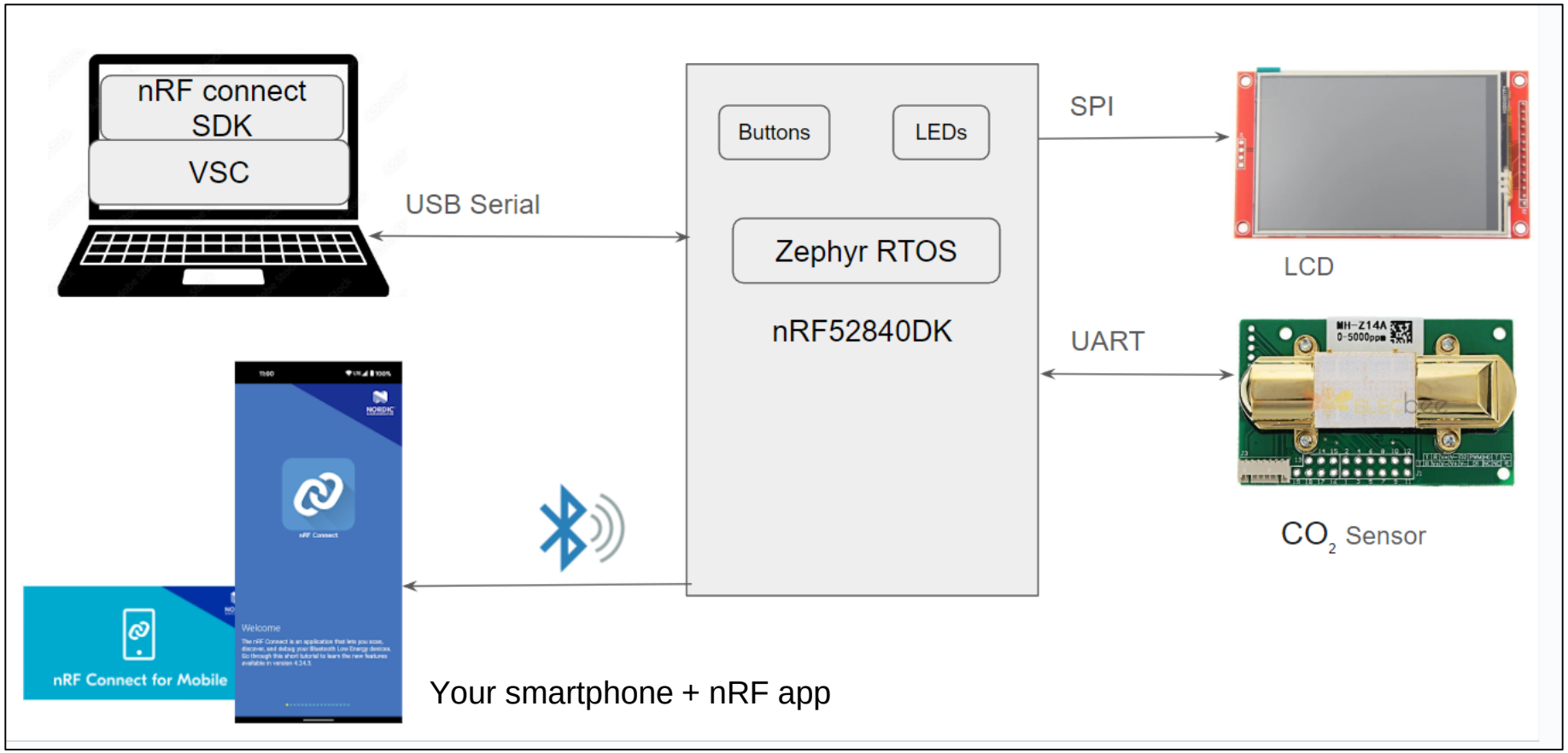


Before going to Lab

- 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/>

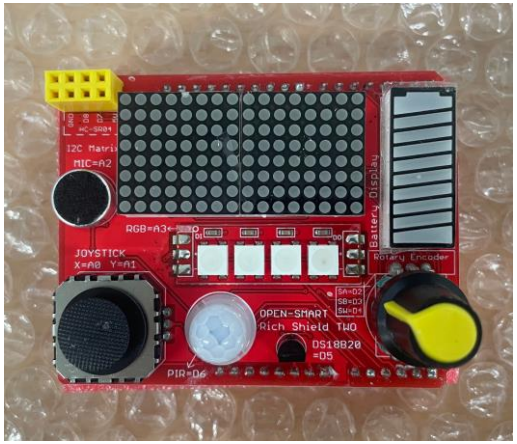


Lab Environment



Hardware Checklist

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

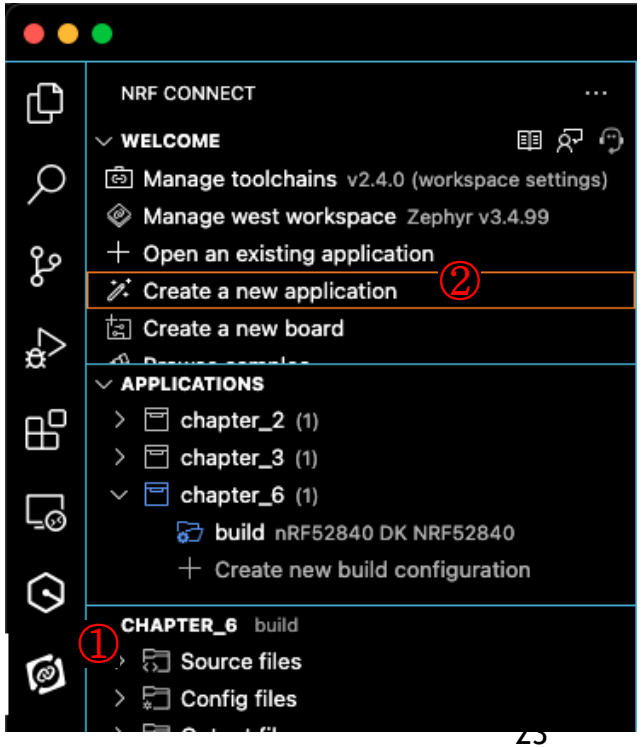


Run Example

- 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”



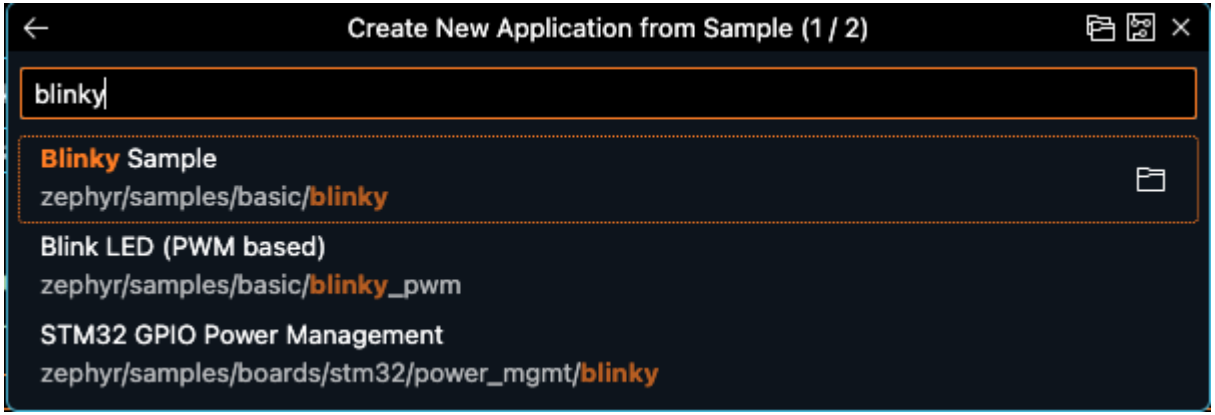
Run Example

- 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

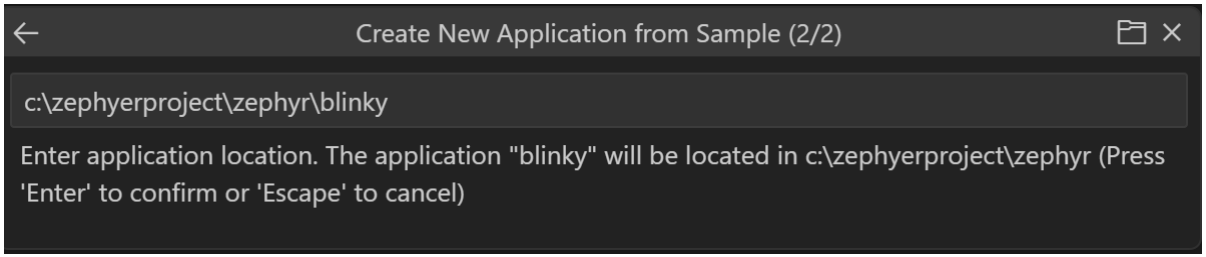
Run Example

- 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

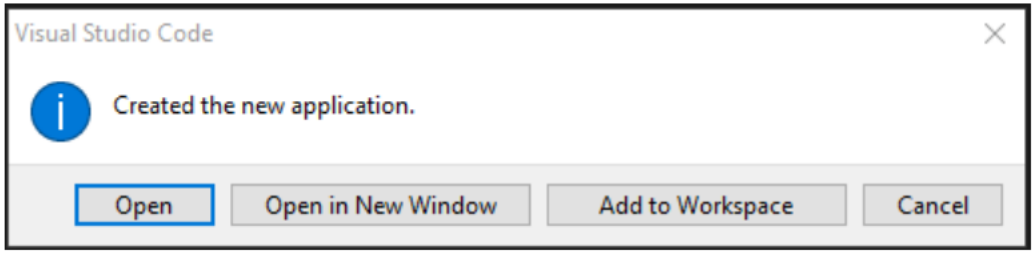


Run Example

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

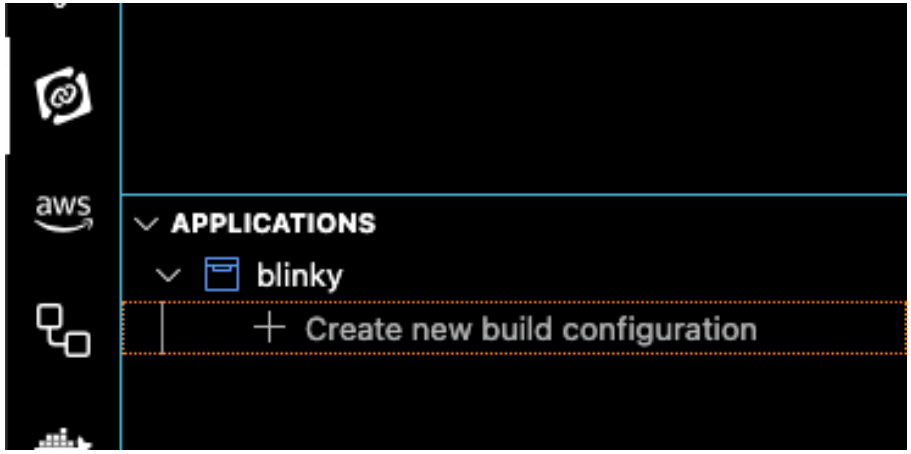


- VSCode will ask you how to open the application
  - Select “Open” to open it in the same VSCode instance.



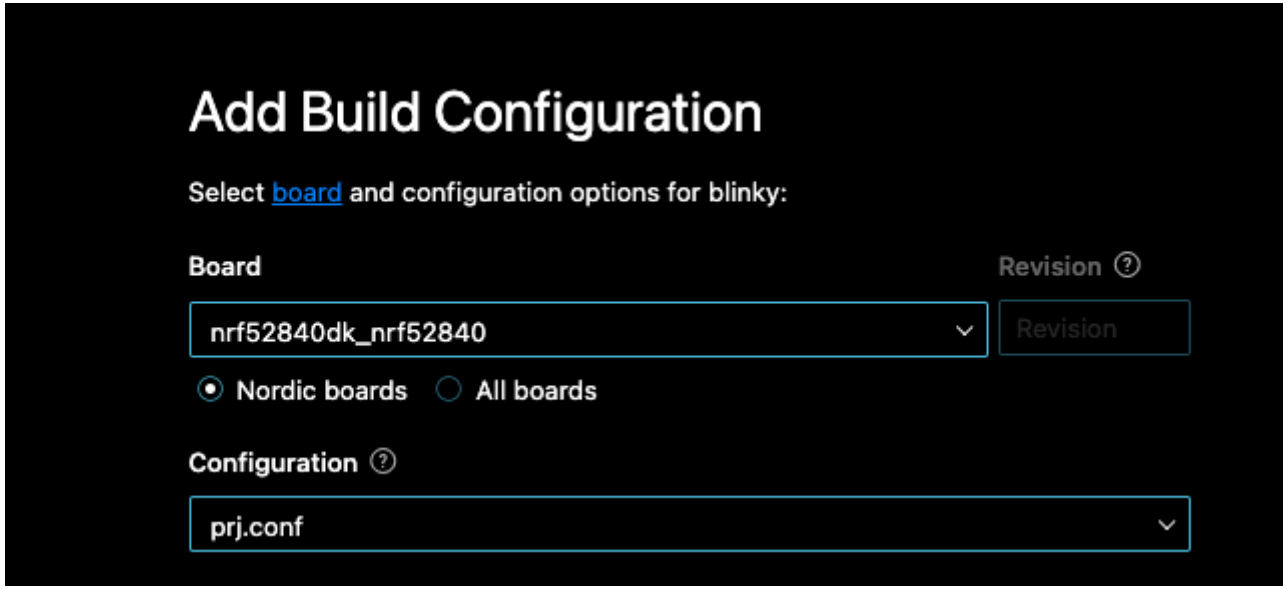
Run Example

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



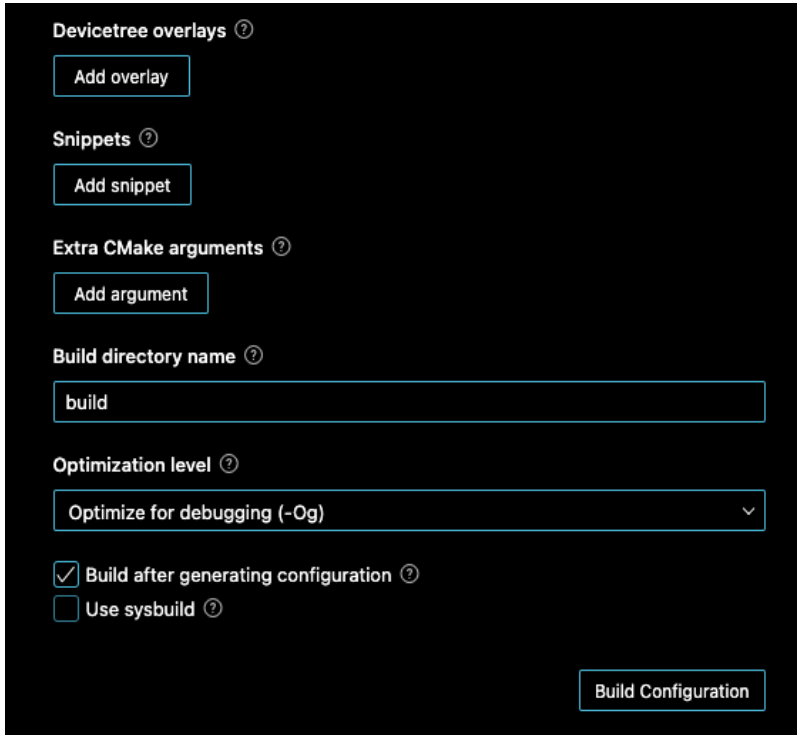
Run Example

- 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

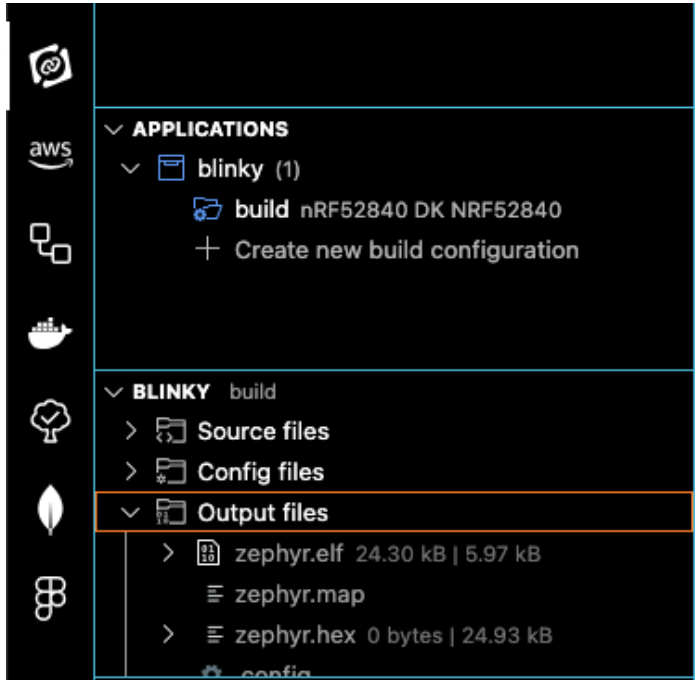
```
[153/164] Linking C static library zephyr\kernel\libkernel.a
[154/164] Linking C executable zephyr\zephyr_pre0.elf

[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.
```

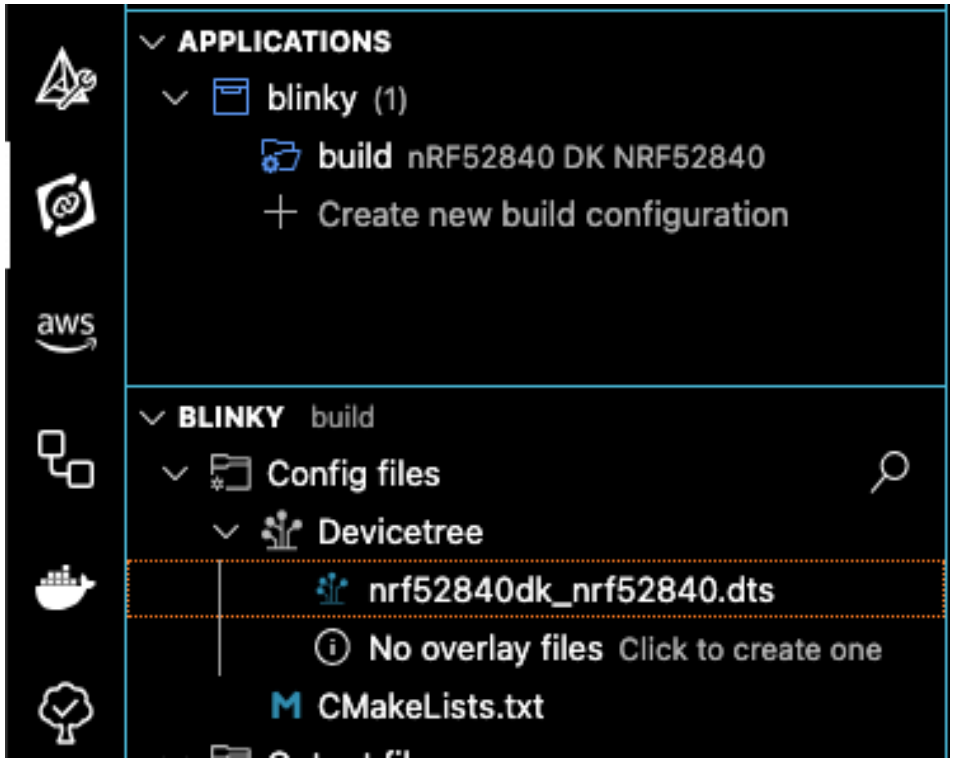
Run Example

- 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

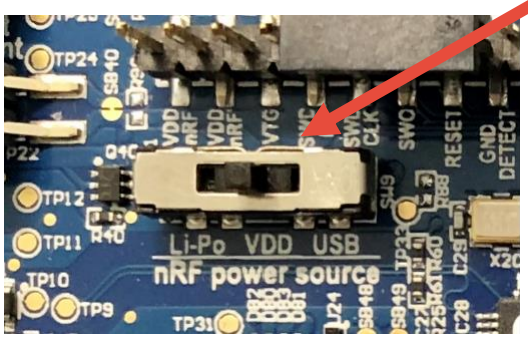


Run Example

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

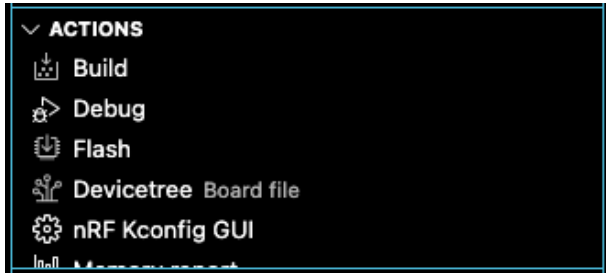
Run Example

- 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.

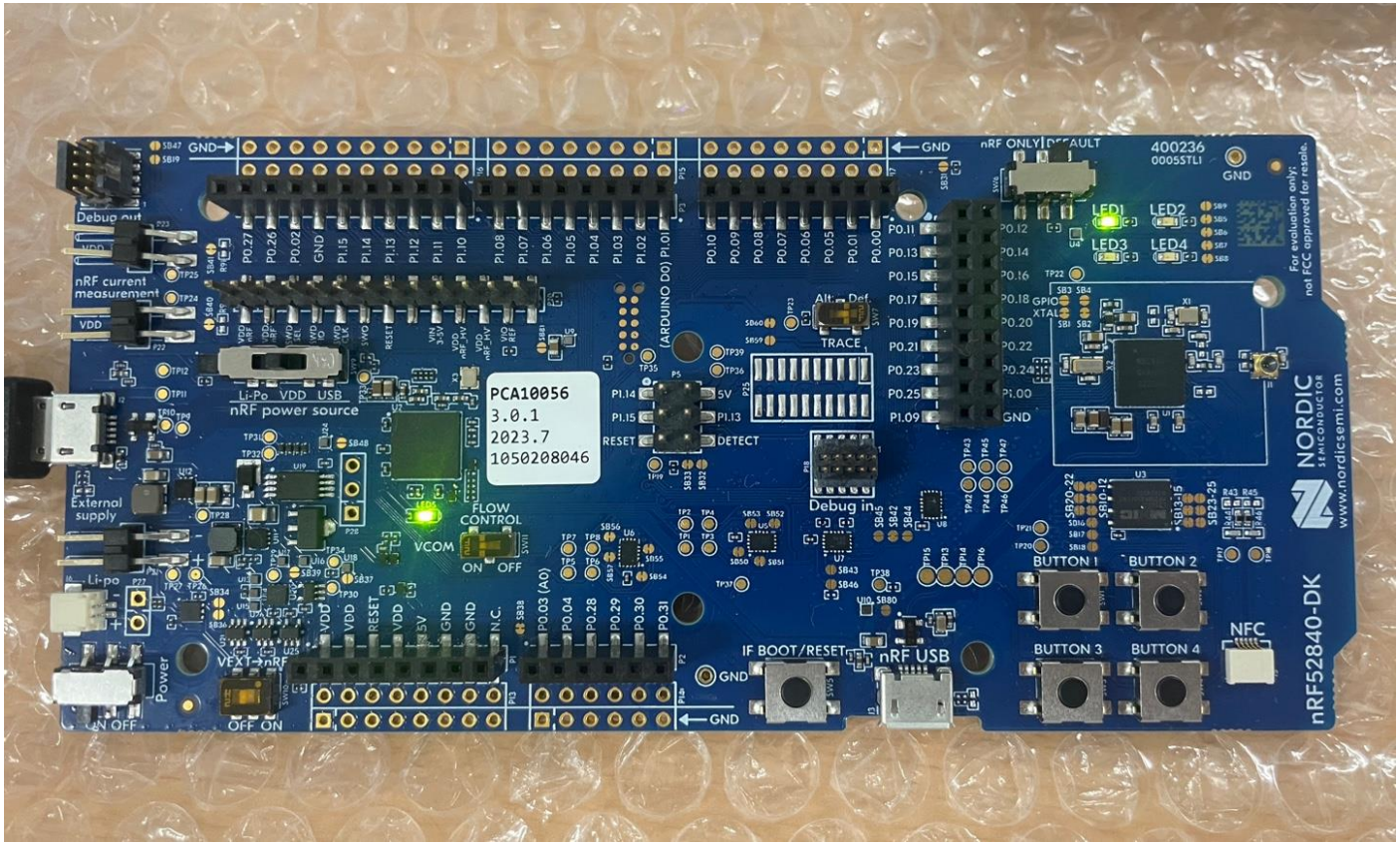



```
west flash -d /Users/yeonwoosung/Desktop/blinky/build --skip-rebuild --dev-id 1050286265

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