Arad Connectivity Co.,Ltd
TEL: 886-2-2680-0180 FAX: 886-2-2680-0360
OFFICIAL WEBSITE: www.aradconn.com
9F., No. 162, Sec. 3, Jiayuan Rd., Shulin Dist., New Taipei City 238671, Taiwan (R.O.C.) (Business Address)

nRF Connect SDK 安裝與入門

Jayant Tang, Nordic Semiconductor

nRF Connect SDK 是跨平臺的（Windows/Linux/OSX），其基礎是 Zephyr 系統。 Zephyr 系統是 Linux 基金會維護的一個項目，除了一個基本的 Zephyr RTOS 之外，還 有很多方便的協力廠商軟體庫，像是 MCUBoot、TCP/UDP/MQTT/TLS 等網路庫等等。 更多資訊可參考： https://docs.nordicsemi.com/bundle/ncs-latest/page/nrf/installation.html

簡介

本節將會詳細介紹如何在一台 Windows 11 的電腦上安裝 nRF Connect SDK 開發環境 （Windows 10 也適用），包含 nRF Connect SDK、編譯器等工具。如果你已經安裝 nRF Connect SDK 安裝與入門- Nordic Semiconductor 中文官網 nRF Connect SDK 開發環境，可以跳過本節。 需要安裝的內容清單：

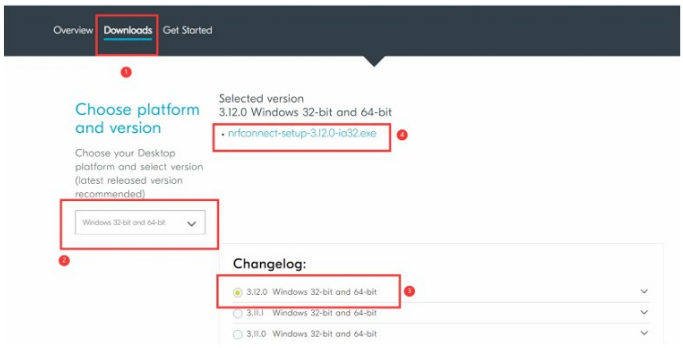
軟體	分類	用途
Visual Studio Code	編輯器	程式碼文字編輯器，並且透過安裝外掛程式的方式為其他開發 調試工具提供可視介面
nRF Command Line Tools	工具鏈	J-link 驅動、nrfjprog 等命令列燒錄工具
nRF Connect for Desktop	桌面工具	桌面工具集合，含 SDK 安裝器、功耗評估、射頻抓封包等工具
Git	工具鏈	版本管理工具
Python 3.9	工具鏈	為編譯套件的 python 腳本提供運行環境
Ninja	工具鏈	快速建置工具（類似 make）
CMake	工具鏈	工程管理器，根據配置產生.ninja 或 Makefile 檔案的工具
Zephyr SDK	工具鏈	交叉編譯工具鏈（編譯器、連結器等）
nRF Connect SDK	SDK	nRF 產品驅動套件、核心、第三源碼包 方函式庫等程式碼及編 譯設定檔等
VS Code: nRF Connect 外掛程式包	編輯器	在 VS Code 中為 nRF Connect SDK 功能提供視覺化

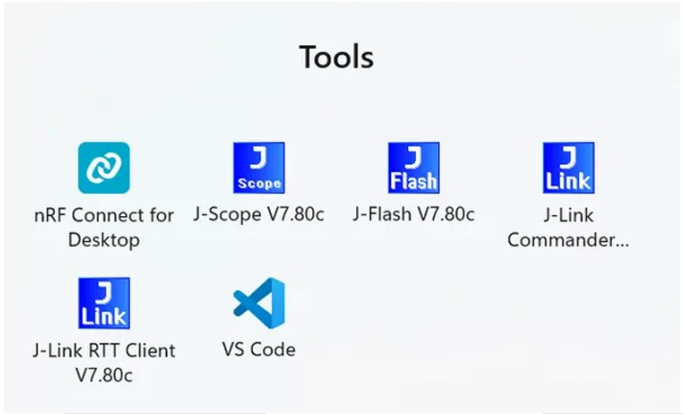
下載安裝包並進行安裝

上述列表中 1~3 號軟體都是使用安裝包的形式進行安裝，點選表格中的連結進入官網，下載安裝後可以直接雙擊安裝。

安裝路徑需要無中文、空格或“-”，“_”以外的特殊字元。

VS Code 的安裝，這裡不做介紹，一步一步安裝即可。剩餘的兩個 Nordic 的工具，安裝方式是類似的。下圖僅範例 nRF Connect for Desktop 的下載方式：





「開始」功能表中安裝好的工具（未完全顯示）

3. 安裝 SDK 和工具鏈

3.1. 安裝 SDK

清單中 4~10 號軟體有 2 種安裝方式，自動安裝和手動安裝。

自動安裝：自動下載 SDK 和工具鏈，所有工具鏈都放在一個資料夾內，編譯時， 只使用此資料夾內的工具鏈。工具鏈與電腦上本身安裝的軟體不衝突。這種方式 最簡單，只不過很多操作都隱含在後台了。

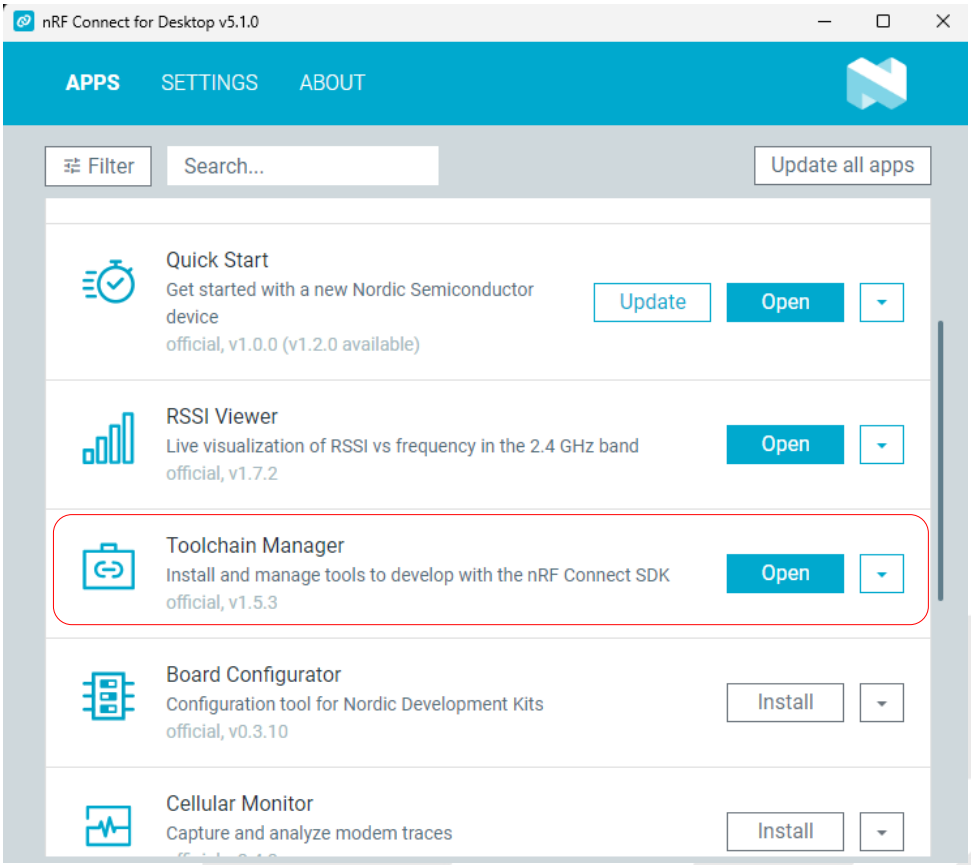
手動安裝（簡易版）：分別下載 SDK 和工具鏈，所有工具鏈都放在一個資料夾內，編譯時，只使用此資料夾內的工具鏈。工具鏈與電腦上本身安裝的軟體不衝突。這種方式能讓你更瞭解 nRF Connect SDK 開發環境各文件的存放位置。

手動安裝：只拉取 SDK，工具鏈單獨安裝到電腦裡，並加入到 PATH 環境變量，編 譯時直接用 PATH 環境變數。

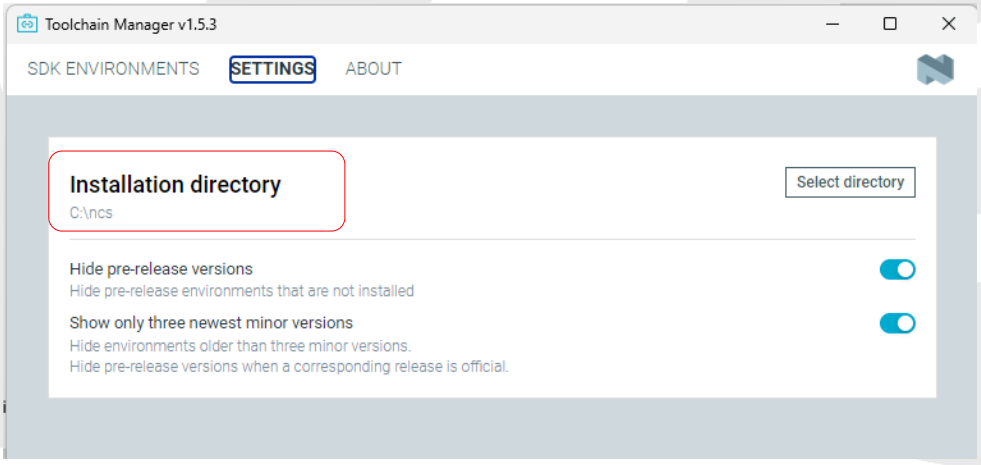
3.1.1 自動安裝［點擊］

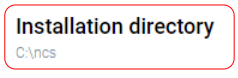
自動安裝

開啟剛安裝的 nRF Connect for Desktop 軟體，找到 Toolchain Manager，Install 然 後 Open。

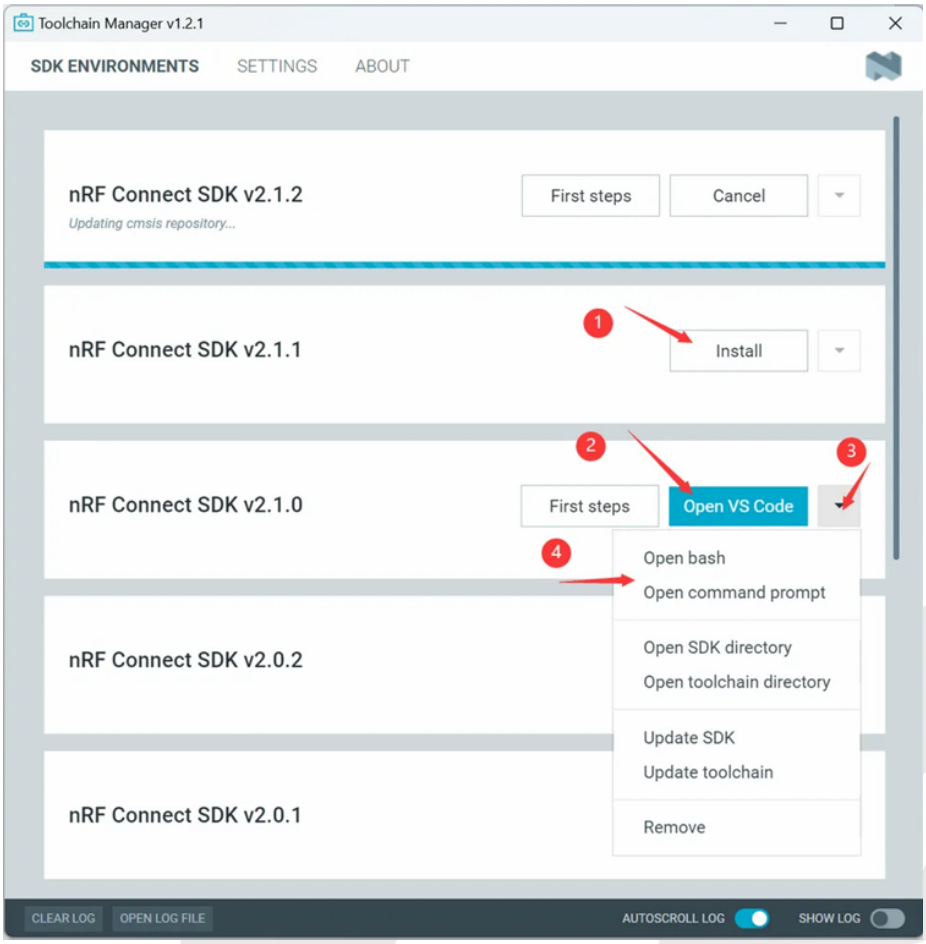


開啟後，先在 SETTINGS 中選擇自己想要存放 nRF Connect SDK 和 toolchains 的位置：

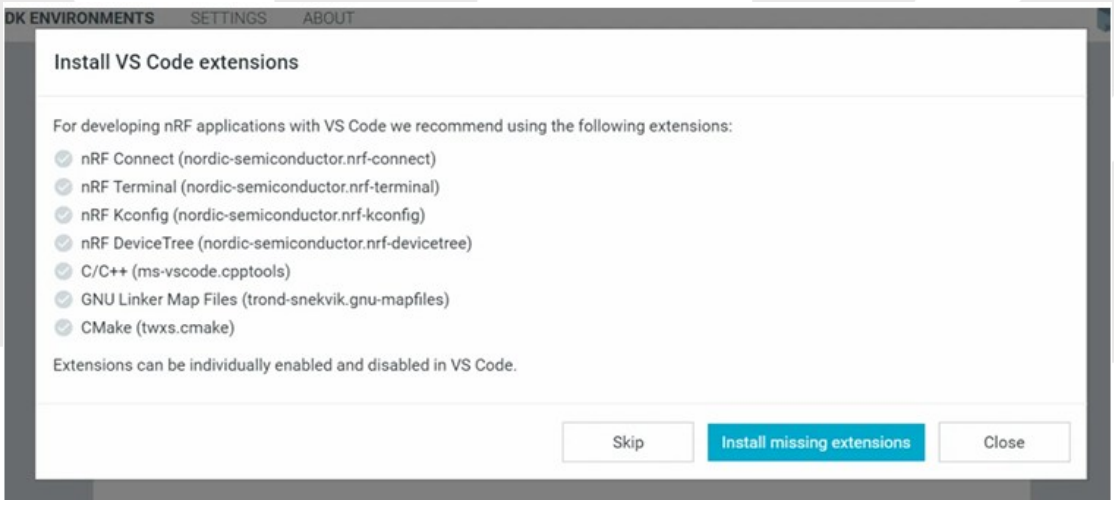




然後，在 SDK ENVIRONMENTS 中選擇想安裝的 SDK 版本進行安裝（如①）。由於是 從 Github 拉取，可能會要很久。安裝完畢後，可開啟 VS Code（如②）。也可以在對 應目錄下開啟終端機（如③④）。



點擊“開啟 VS Code”，它會彈跳視窗提示，幫你自動安裝 VS Code 的 nRF Connect 外掛 包，如下圖：



【重要】檢查安裝是否完整

即使安裝過程中沒有任何報錯，SDK 很有可能安裝是不完整的。不過不必擔心 toolchain 不完整，因為 toolchain 不完整的時候會報錯。

這時，需要在 SDK 的目錄下開啟命令列。但不是直接打開，因為直接打開命令列，環 境變數都是 PATH。而是要從上圖

的「Open command prompt」來開啟命令列，這時，開啟的命令列中的環境變數是指向 Toolchain 資料夾的。

開啟後，輸入 west update。 nRF Connect SDK 是由多個 Github 上的倉庫組成的，此命令會依序 pull 這些倉庫，如果有檔案缺失，就能從命令列的輸出看出來。 如果某些倉庫 pull 失敗，就繼續不斷重複執行 west update ，因為它是可以斷點繼續下載的。直到指令不報錯為止，則表示倉庫全部拉取完畢。

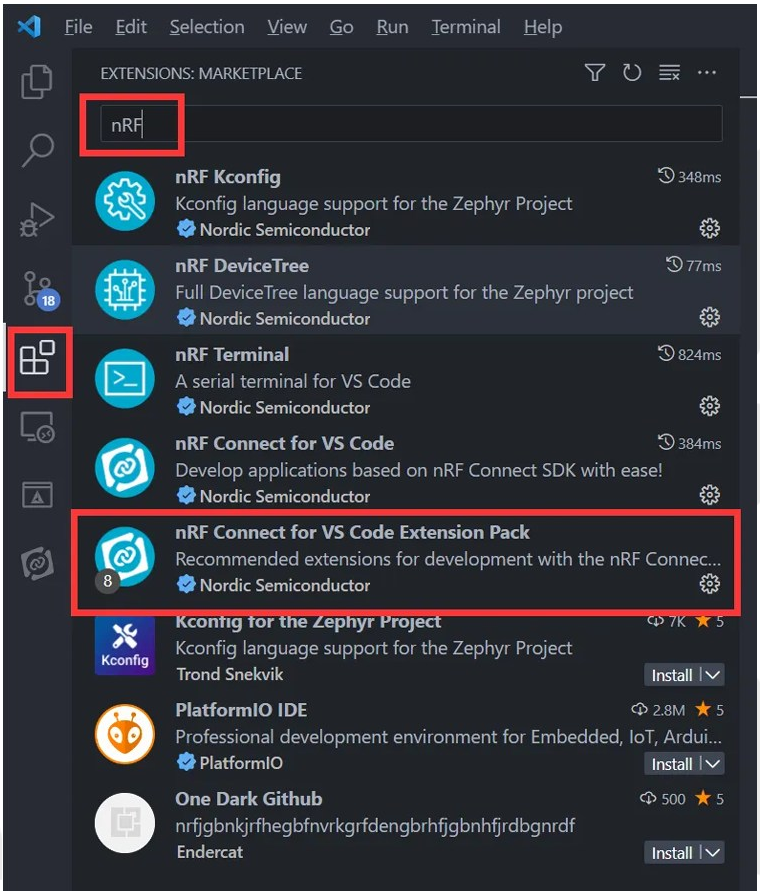
最後執行 west zephyr-export，此指令會讓工具鏈中的 CMake 記住 SDK 的位置。

3.1.2 手動安裝（簡易版）[點擊]

手動安裝（簡易版）

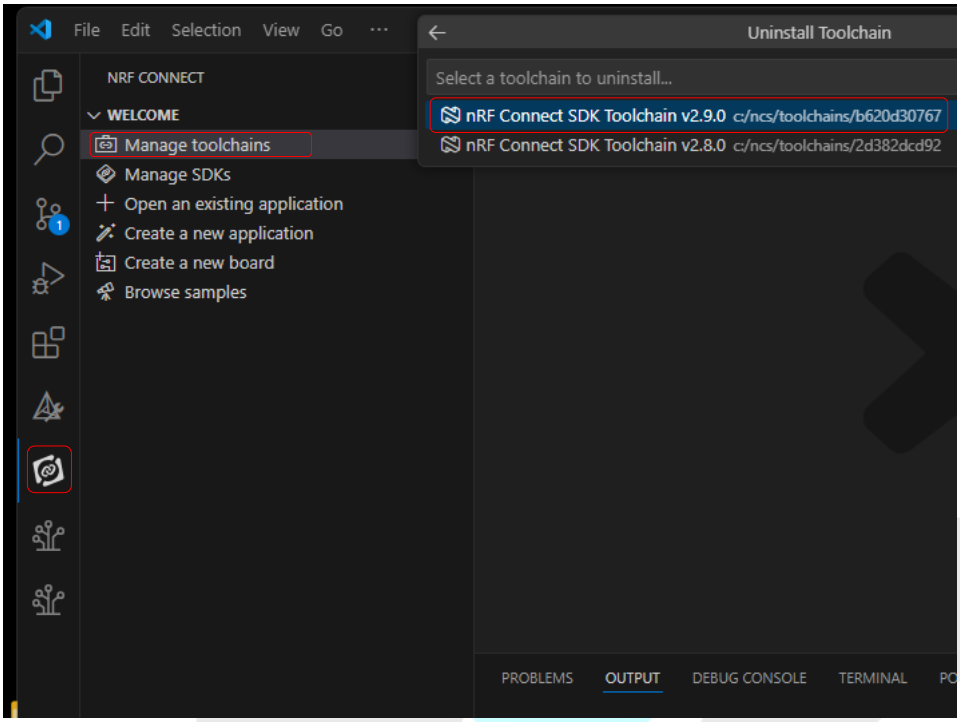
安裝 VS Code 外掛程式

開啟 VS Code，在外掛程式中心搜尋 nRF Connect for VS Code Extension Pack，這個外掛程式包會自動安裝其他 nRF Connect 所需的 VS Code 外掛程式。



安裝 Toolchain

在外掛程式的 Welcome 介面，選擇安裝新的 Toolchain



Toolchain 的版本號碼與你想安裝的 SDK 的版本號碼保持一致即可。 這種方法下載的各個版本的 toolchain 壓縮包會放在${HOME}/ncs/toolchains 目錄下 （Windows 上是 C:\\ncs\toolchains\）。 然後，我們在 VS Code 中，可以設定預設的 Toolchain：

Toolchain 資料夾中已經安裝了我們需要的各種工具，而這些工具不會被加入到你的電 腦上 PATH 環境變數。這樣就可以防止工具鏈中的軟體和你電腦上已經安裝的同名軟體 產生衝突（如 Python）。

假如 tool chain 下載失敗想重新下載，需要把創建的 ncs 資料夾下的所有內容刪除乾 淨，尤其是.west 隱藏資料夾。然後再次執行 west init 即可；1. 更新倉庫 / west update

匯出 Zephyr CMake package，

方便 CMake 自動辨識 SDK 的路徑 後續以 SDK 中的程式為範本建立新工程時，就是用這個來找 SDK 的位置。

west zephyr-export west zephyr-export

3.1.3 手動安裝［點擊］

手動安裝

[Windows 專用】安裝 choco

choco 是一個套件管理工具，類似 Ubuntu 中的 apt-get，可以透過命令列安裝軟體工具，並自動加入到全域 PATH 環境變數。 首先右鍵單擊開始選單，然後打開管理員終端，輸入以下命令：

```
# 設置腳本執行許可權
Set-ExecutionPolicy AllSigned Set-ExecutionPolicy AllSigned
   # 從網絡端執行安裝腳本
      Set-ExecutionPolicy Bypass -Scope Process -Force;  [System.Net.ServicePointManager]::Securi-
tyProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object Sys-
tem.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))#
```

測試是否安裝成功：

```
choco
# 有版本號輸出即為成功
```

利用 choco 安裝其他工具

部分工具也可自行去官網下載，並用安裝包安裝，注意安裝時要勾選「新增至PATH 環 境變數」。

```
# 設置 choco
  choco feature enable enable  -n allowGlobalConfirmation
# 安裝 cmake（也可以把這步替換成從官網下載 cmake 安裝包，注意安裝時要勾選添加進 PATH 環境變量）
 choco install cmake   --installargs  'ADD_CMAKE_TO_PATH=System'
# 安裝 git（也可以把這步替換成從官網下載 git 安裝包，注意安裝時要勾選添加到環境變數）
choco install git
# 安裝 python（建議把這步換成從官網下載 python3.9 安裝包，並勾選添加到環境變數,通過 choco 安裝容易出錯）
choco install python --version = =3.9.13
# 安裝其他工具
choco install ninja gperf dtc-msys2 wget unzip
```

Git 也建議從官網安裝，同時勾選安裝 Git bash。可以讓你在 windows 上使用 bash 終端，而不是 powershell。

安裝 GN 工具（選購）

GN 工具是開發 Matter 所需的工具。

從 GN 網站下載編譯好的 Windows 壓縮包（Getting a binary），建議在使用者目錄 （C:\Users）下解壓縮。並且加入 PATH 環境變數即可。 如果你的 Windows 使用者名稱是中文，那還是換個無中文的目錄。

安裝 west

west 是一個多倉庫管理工具（類似 Android 的 repo），支援添加自訂外掛程式。在 nRF Connect SDK 中，除了可以管理 nRF Connect SDK 倉庫外，還透過外掛程式實現了闆卡選擇、觸發編譯動作、觸發 flash 燒寫的功能。

利用 Python 的 pip 進行安裝

pip3 install west
#若 python 版本不對，這一步可能會報錯

安裝 nRF Connect SDK
nRF Connect SDK 前面已經介紹過，包含驅動、核心以及協力廠商函式庫的來源碼。
在一個無中文、無空格的合適路徑下開啟終端（powershell 或 bash）：

```
#創建並進入安裝目錄
 mkdir ncs cd
 cd ncs
# 初始化倉庫（從 github 拉取）
west init -m https://github.com/nrfconnect/sdk-nrf --mr v2.4.2
# 也可選擇其他分支或 tag，如：
west init -m https://github.com/nrfconnect/sdk-nrf --mr main
# 但通常不建議用戶使用 main 分支
```

這一步如果下載失敗想重新下載，需要把創建的 ncs 資料夾下的所有內容刪除乾淨，尤其是.west 隱藏資料夾。然後再次執行 west init 即可；
更新倉庫

```
#更新倉庫 west update
```

如果失敗了，就重複 west update 就行了。不需要像 west init 失敗一樣刪除乾 淨重新下載。
匯出 Zephyr CMake package，方便 CMake 自動辨識 SDK 的路徑，後續生成工程

```
west zephyr-export
```

安裝額外的 python 依賴
安裝 python 依賴之前，還需要安裝"Microsoft Visual C++ Build Tools 14.0"或更高版本：
Microsoft C++ Build Tools - Visual Studio，
用來編譯這些 python 工具。在上述 微軟連結下載，會獲得一個 VS 安裝工具。只在 Workloads 欄裡選擇"Desktop Development with C++"，然後安裝即可。

安裝 python 依賴套件：

```
pip3 install -r zephyr/scripts/requirements.txt
pip3 install -r nrf/scripts/requirements.txt
pip3 install -r bootloader/mcuboot/requirements.txt
```

安裝 Zephyr SDK 工具鏈
Zephyr SDK 是編譯器、連結器等工具。建議放在使用者目錄下（同樣的，如果你的使 用者名稱是中文，還是換目錄吧）。
下方展示了透過 Poweshell 指令下載 Zephyr SDK 的方式。其中 Zephyr 工具鏈的版本是 我安裝時所使用的版本。你需要取得最新的版本，最新版本的下載位址可從

Zephyr SDK — Zephyr Project Documentation 取得。

```
# 進入到用戶目錄（c:\Users\[用戶名]）
cd $HOME
# 如果是你的終端是 bash 而非 powershell，則命令為
# cd ~
# 下載（最新版本的下載位址可從上述鏈接獲取）
wget https://github.com/zephyrproject-rtos/sdk-ng/releases/download/v0.15.1/zephyr-sdk-0.15.1
# 解壓
unzip zephyr-sdk-0.15.1_windows-x86_64.zip
# 配置
cd zephyr-sdk-0.15.1
setup.cmd
```

注意：

setup.cmd 只需執行一次。如果改變了安裝位置，需要再次執行。

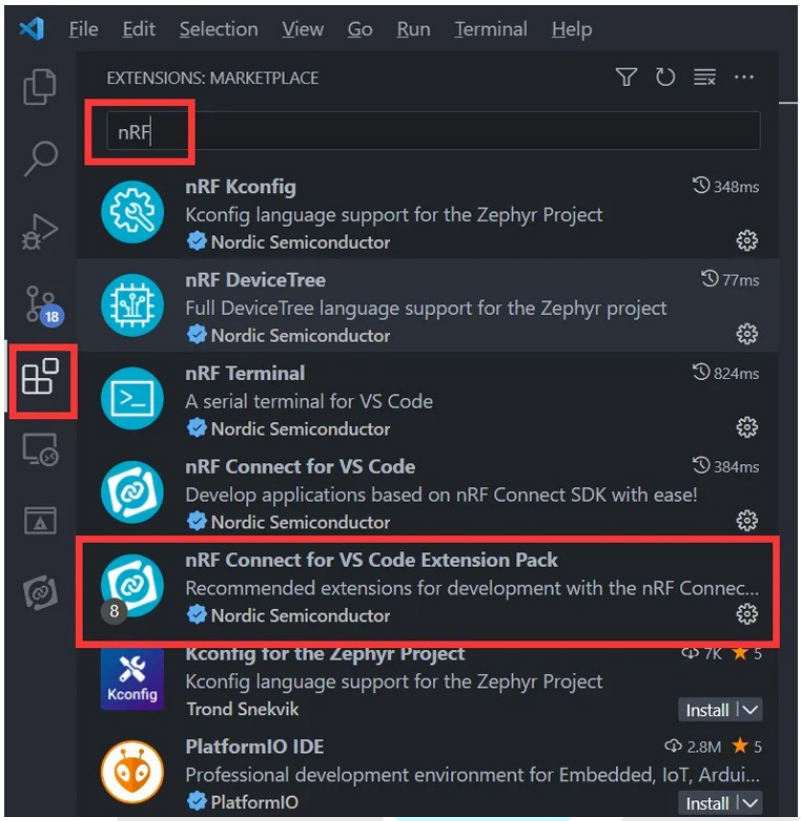
Zephyr SDK 安裝完成後，會發現：如果在前面安裝了 ncs 的目錄下執行 west - help 會比其他目錄下執行 west --help 多出一些擴充指令，如 build , board 等等。 這是因為 nRF Connect SDK 中的.west 資料夾的配置了 Zephyer 的 base 路徑，提 供了這個倉庫獨有的外掛程式。這些擴充的指令就是呼叫了外掛程式進行編譯、調試、燒 寫等工作。

為了讓其他目錄下也能使用 Zephyr 工具，需要設定全域環境變數： 在 Windows 環境變數中新建 ZEPHYR_BASE 環境變量，並把其值設定為 ncs 安裝目錄下的 zephyr 目錄的路徑即可。

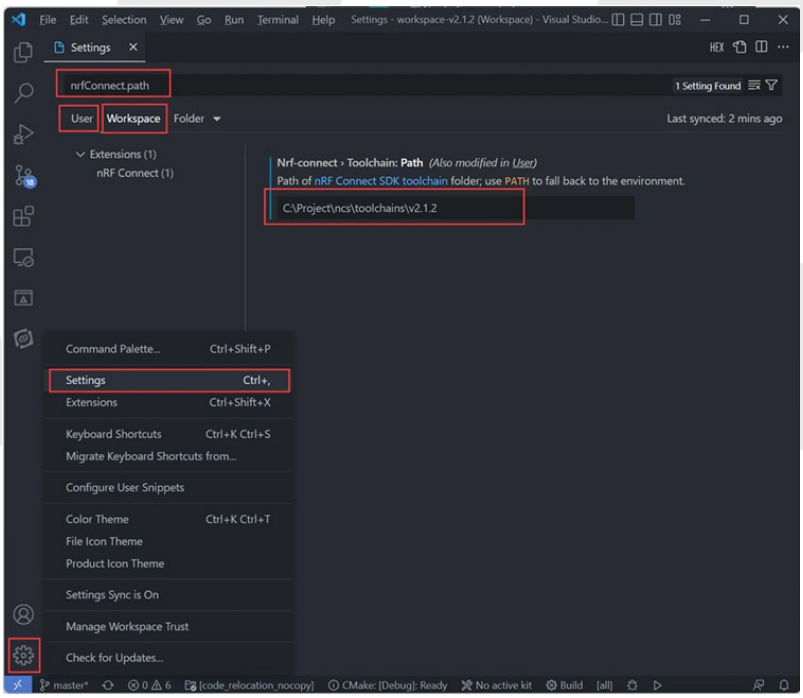
（這個效果和執行一次 ncs 目錄下的 zephyr/zephyr-env.cmd 腳本是一樣的，但這個腳本設定環境變數只是暫時生效，關閉終端機再另開一個終端就不起作用了。而設定 全域環境變數是永久生效。）

安裝 VS Code 外掛程式

開啟 VS Code，在外掛程式中心搜尋 nRF Connect for VS Code Extension Pack，這個外掛程式包會自動安裝其他 nRF Connect 所需的 VS Code 外掛程式。



可以在 VS Code 的設定中，對外掛程式進行單獨的設置，例如可以設定使用工具鏈的 路徑。可以對全域進行設定（USER），也可以單獨對某個工作區進行設定 （WORKSPACE）。



由於我們是手動安裝的，已經設定了 PATH 環境變數。所以把外掛程式設定的工具鏈路徑設 為 PATH 即可。

3.2. SDK 更新

手動安裝和自動安裝的 SDK 是一樣的。如果有新版本 SDK 出來，想切換到新版本，都可以依照以下步驟操作： 確保 SDK

中的 git 倉庫狀態均為 Clean 這意味著，客戶平時不要隨便改 SDK 中的任何程式碼。也不要往裡面添加任何內容。 但是編譯常式是沒問題的，因為常式編譯目錄 build *是被.gitignore 忽略掉的。 # 此命令可查看當前 git 倉庫的狀態
git status #
但是 nRF Connect SDK 中的倉庫很多。也可以用 VS Code 開啟整個 nRF Connect SDK，用 git 介面圖形化來查看是否每個倉庫都是 clean。

檢查 manifest 有無新版本

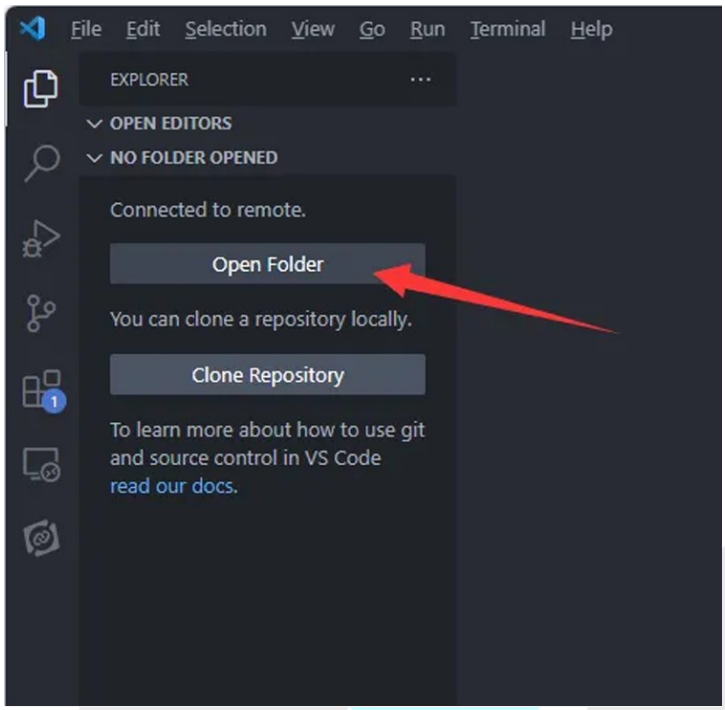
nRF Connect SDK 中，nrf 為主倉庫，nrf 的版本即為整個 SDK 的版本
查看 nfr 倉庫下有多少版本
```
cd nrf
git pull
git tag   # 按鍵盤上下鍵翻閱，按 q 退出 # 按鍵盤上下鍵翻閱，按 q 退出
```

切換到自己想要的版本
檢出想要的版本
git checkout v2.4.2

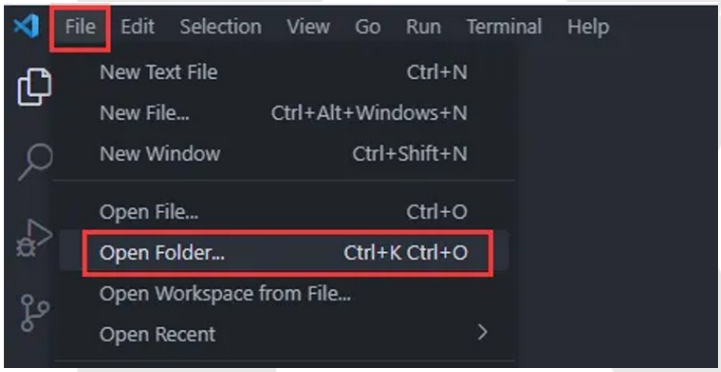
更新整個倉庫
west update

4. 打開一個常式

從 VS Code 的全新窗口，選擇開啟資料夾：



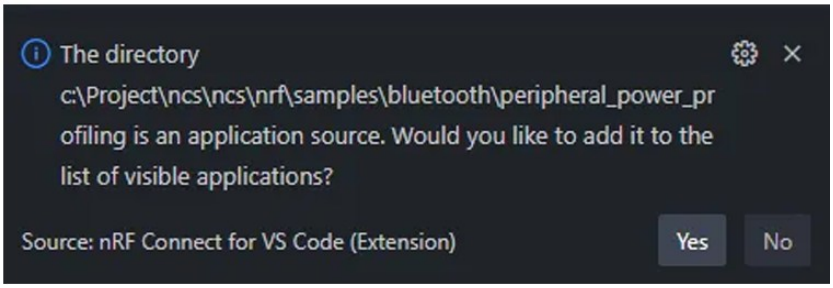
或者：



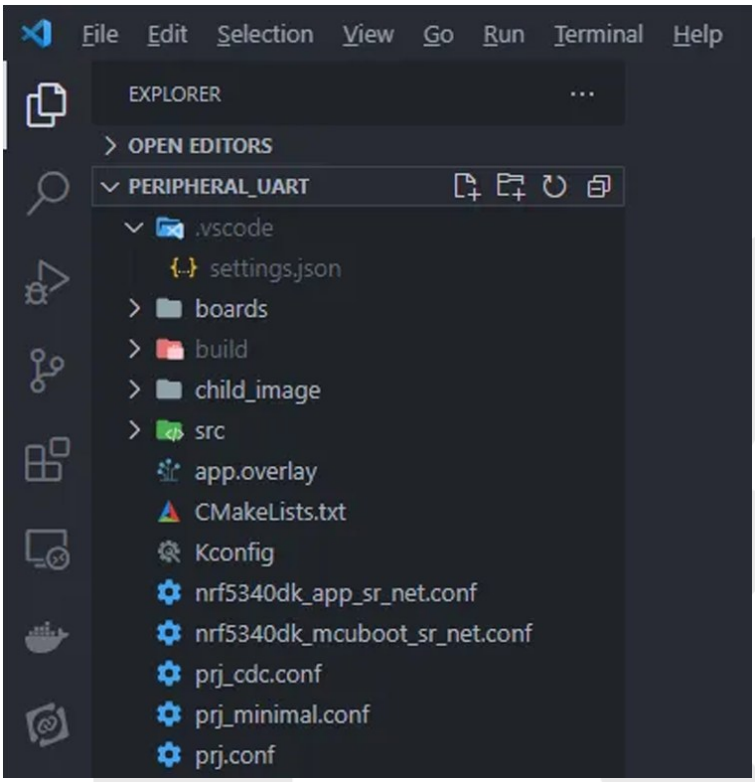
zephyr/samples/中有 RTOS 的組件常式、Zephyr 支援的各類廠商的闊卡常式、各類 感測器的常式等，其中也有藍牙常式。 zephyr/tests/中有全部的 API 測試常式。

nrf 倉庫的目錄結構仿造 zephyr 倉庫，也有 samples/和 tests/ 目錄。 samples/ 中有 Nordic 提供的軟體庫常式、Zephyr 未收錄的常式（如 nRF9160 的 LTE）等。

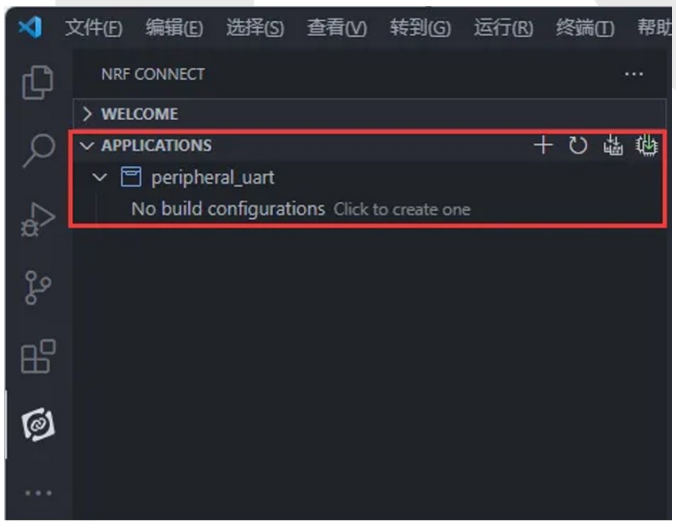
我們選擇一個藍牙常式，nrf/samples/bluetooth/peripheral_uart，並開啟資料 夾。這只是在 VS Code 中打開，打開後，nRF Connect 外掛程式會檢測到這是一個 Application 資料夾，詢問你是否要把它加入到 nRF Connect 外掛程式的 Application 中， 點擊 Yes 即可：



開啟後，在 VS Code 資源管理器中可以看到資料夾檢視：



另外，也可以在 Application 中看到這個常式：



5. 以常式為範本建立新工程

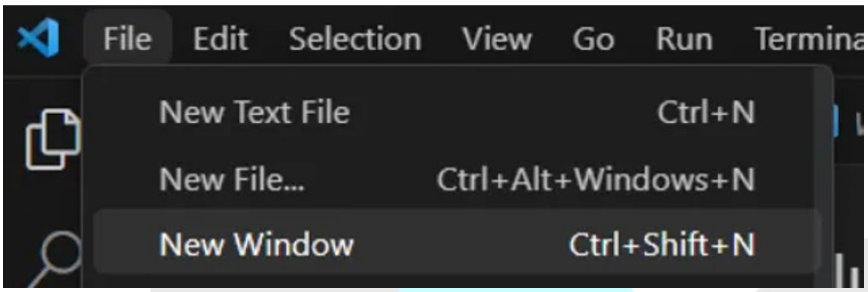
上一節講解如何開啟一個常式。

如果我們只是開啟常式，常式的資料夾還是在 ncs 倉庫內部，受到 ncs 的 git 倉庫的管理。如果想自己開發項目，並用 git 管理版本，就需要建立新工程。 nRF Connect SDK 支援把常式當作範本，複製到 nRF Connect SDK 外部，並建立新工程。

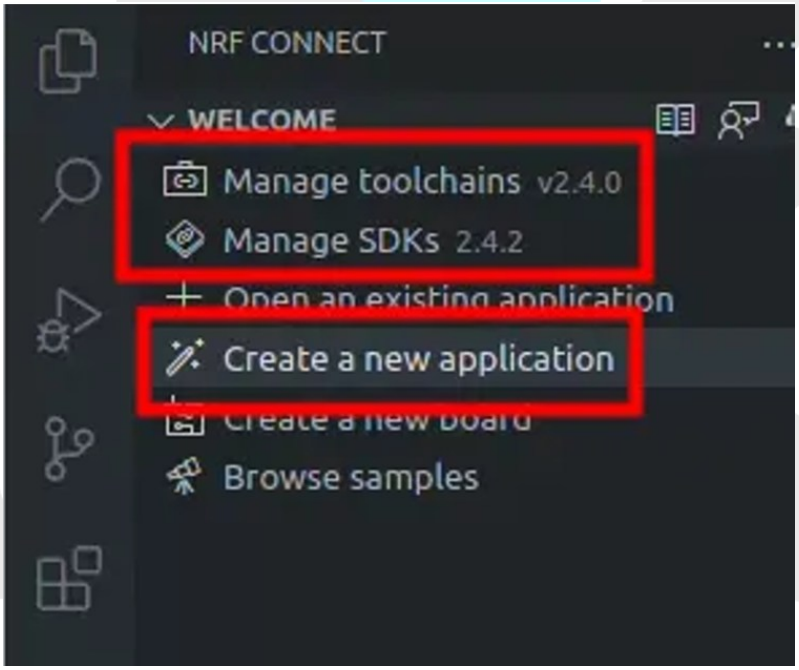
5.1. 建立新工程

nRF Connect SDK 支援以常式作為範本，複製並建立新的工程。這也是 Nordic 非常推薦的方式。

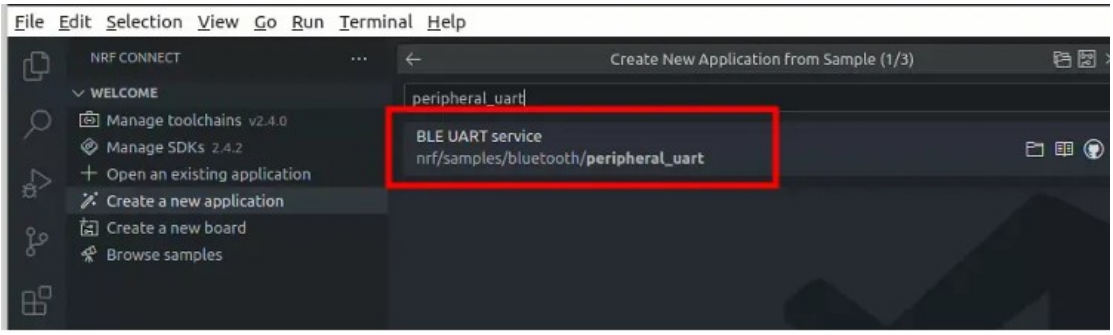
首先在 VS Code 中開啟一個新視窗:



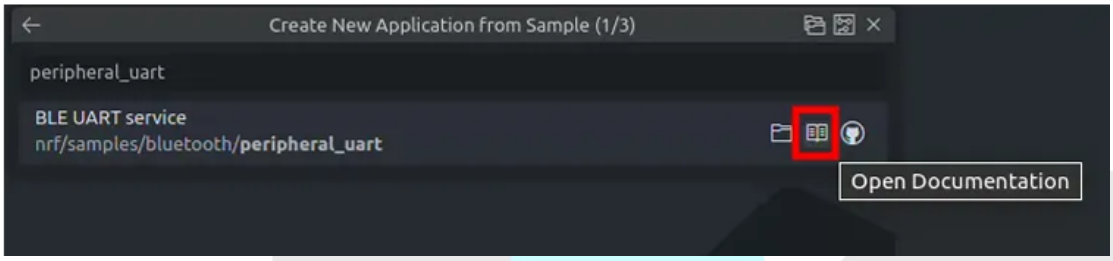
在 VS Code 中，選擇左側 nRF Connect for VS Code 外掛程式，進入 Welcome 頁面，先檢查 toolchain 和 SDK 是否已經偵測到。 然後點選 Create a new application 建立新工程。



選擇自己想要拷貝的常式，支援文字搜尋：



這裡的常式清單，和第 4 節中提到的目錄結構是一致的。同時也和 SDK 官網的常式說明是保持一致的，每個常式都有其對應的說明文件：



Nordic 商業級應用

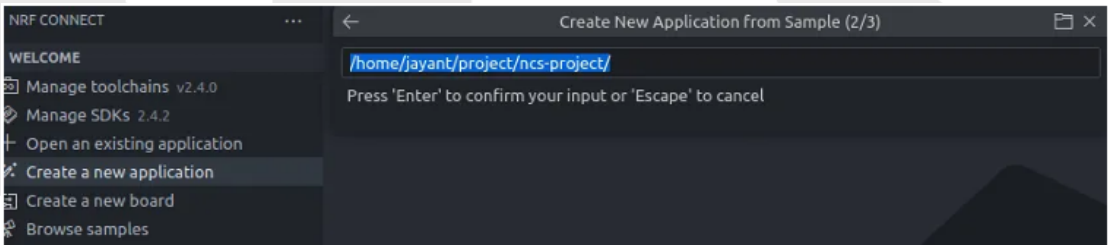
Nordic 常式

Zephyr 常式

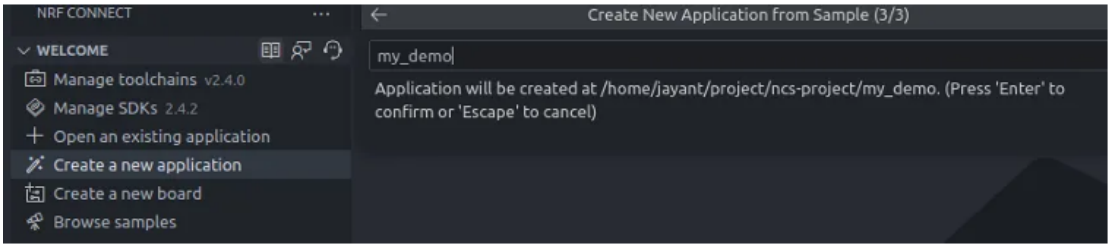
此外，還有一些隱藏常式不會出現在這個介面，但可供參考： ${NCS}/modules/hal/nordic/nrfx/samples/src/：NRFX 週邊驅動程式庫常式。如果使用者不想用、或 Zephyr 沒有提供某些週邊裝置的標準驅動，則可以使用 NRFX 驅動，其用法和舊的 nRF5 SDK 基本上一致。

${NCS}/zephyr/tests：zephyr 所有的 API 的測試案例。如果你不知道某個 Zephyr API 怎麼用，可以從這裡面找。

選擇自己新常式的父目錄（我這裡是 Linux 電腦的路徑）：



選擇自己新常式的資料夾名稱：



然後就可以開啟新的工程。

5.2. 使用 git 追蹤你的程式碼修改

如果你從來沒用過 git，需要先設定使用者名稱和信箱。這個使用者名稱和郵箱不是登錄什麼網站用的，而是一個簽名，在提交程式碼時用來標記這段程式碼是誰提交的。 這個設定存在你電腦的本地，而且是全域的，對所有 git 倉庫都有效。

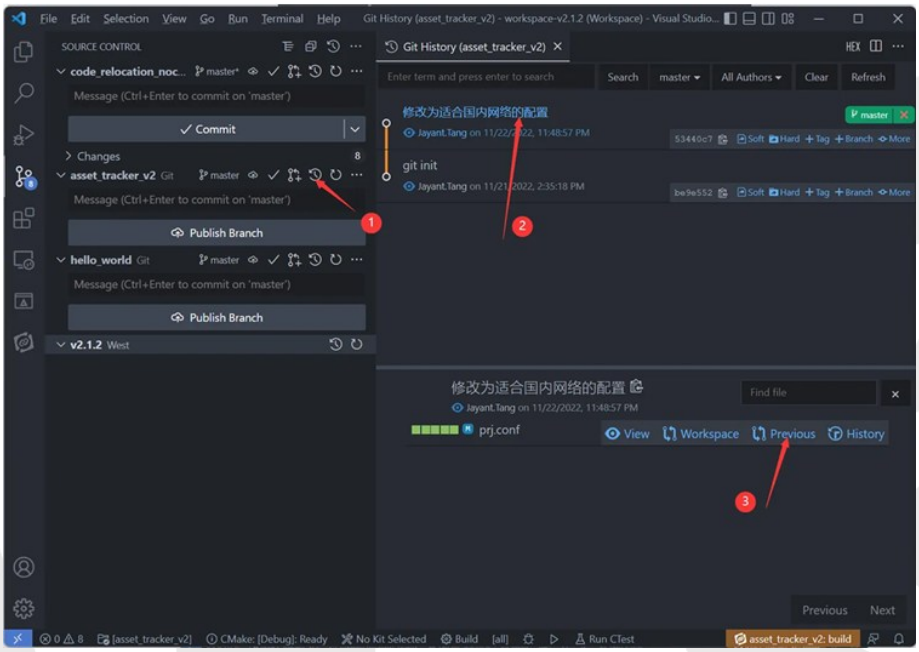
```
git config --global user.name "Jayant.Tang"
git config --global user. "[email.protected]"
```

新建的工程都會自動初始化 git 倉庫，我們可以看到.gitignore 檔案：

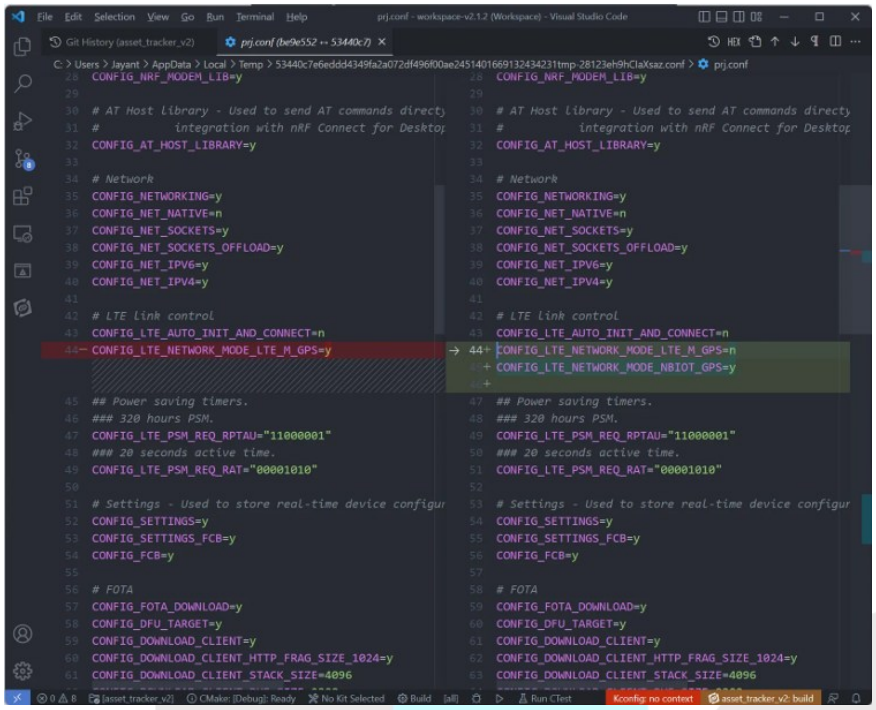
你可以把.vscode/加到其中

如果你不熟悉 Git 以及 Git 在 VS Code 中的使用，強烈建議去學習一下，它極大的方便了程式碼的管理。

例如：如果安裝了 git history 外掛程式，就可以查看提交歷史記錄：



Git History 提供了一個很方便的視圖，可以看到每次 commit 都改動了哪些程式碼和配置（左側是舊的，右側是新的）：



更多 Git 的使用，可以去網路上瞭解其他教學。本文不再贅述。

6. 編譯工程

6.1. 建立一個編譯目標（Build Target）

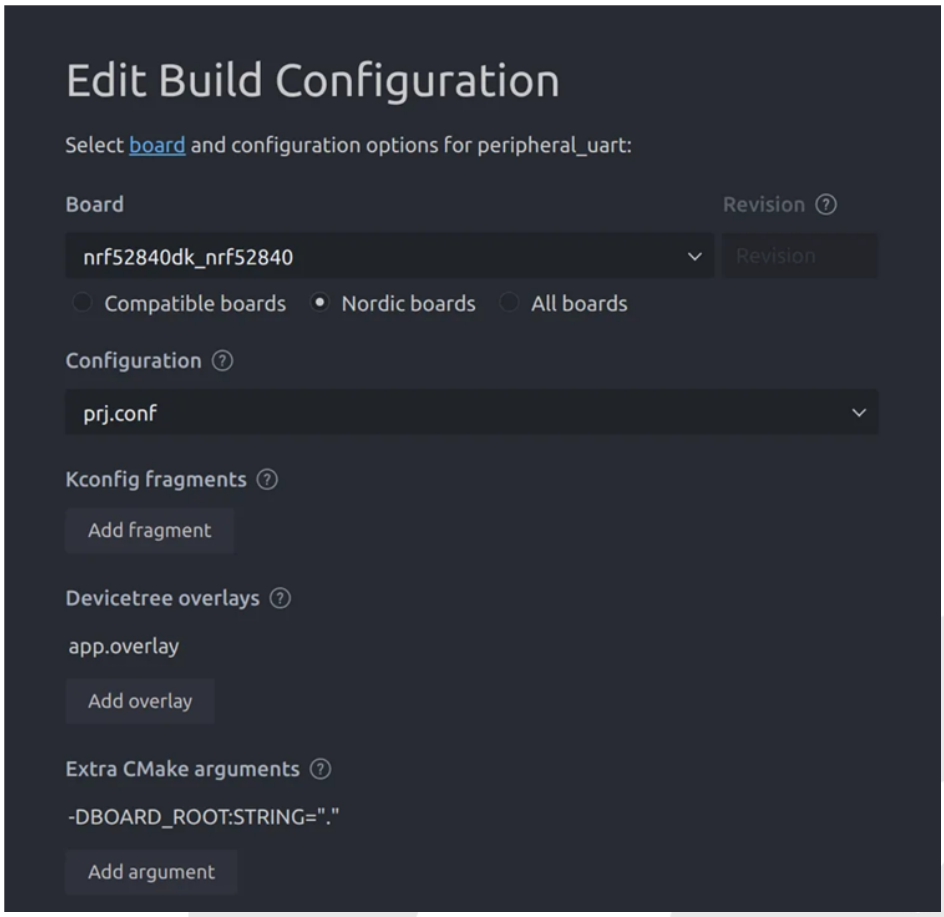
所謂編譯目標就是在同一套程式碼下，可能有不同的設定項（Debug/Release，不同 的最佳化等級等等），編譯出不同的執行檔。一個專案下可以建立多個編譯目標。

Board

建立 Build，需要選擇自己使用的板子，Zephyr 自備各廠商的開發板配置。 下圖中，Board 下拉清單是用來選板子的，下方還有三個過濾器，來過濾選購的板子： Compatible boards：本程式適配的板子，如果選擇這些板子，不需要任何修改 就可以燒錄進去使用

Nordic boards：Nordic 出品的官方開發板，以及一些 Nordic 的 demo 板

All boards：Zephyr 支援的所有開發板



Nordic 開發板 board 配置的命名規則：
例如：nrf52840dk_nrf52840，是說這個板子的配置是為 nrf52840dk 這塊開發板上的 nrf52840 這顆 MCU 創建的，會記錄這個 MCU 的外設位址，以及此 MCU 連接的外部硬體的資訊（如 SPI Flash）。

例如：nrf9160dk_nrf9160 和 nrf9160dk_nrf52840，都是 nrf9160dk 這塊開發板的設定。但這塊開發板上有兩顆 MCU/SoC，一顆是 9160 SiP，另一顆是 52840。所以有兩個配置可選，分別為這兩顆 MCU/SoC 編譯韌體。

例如：nrf5340dk_nrf5340_cpuapp 和 nrf5340dk_nrf5340_cpunet，都是 nrf5340dk 這塊板子的配置，而這塊板子上只有 nRF5340 這一顆主控。但是 nRF5340 是一顆雙核心 MCU，所以，可以有兩種配置來區分兩個核心。這兩個核 的韌體是分開運行的，因此編譯時也是分別編譯的。 例如：nrf5340dk_nrf5340_cpuapp 和 nrf5340dk_nrf5340_cpuapp_ns，都是 nrf5340dk 開發板上，nrf5340 晶片的應用核的配置。但是，這顆應用核使用的 CPU 是 Cortex-M33，基於 Arm V8 架構，提供了 TrustZone 的安全保護技術，同樣的一個週邊寄存器，可以有安全（Secure）和非安全（Non-Secure）兩個位址，這樣可以把安全應用和非安全應用隔離開來。因此，這兩個 board 配置的不同之處，就是從安全位址或非安全位址去存取晶片上的周邊資源。
例如：nrf52833dk_nrf52820。這塊開發板上只有 nrf52833 這一塊主控。但由於 nRF52833 和 nRF52820 同屬 nRF52 系列，52820 上的資源是 52833 的子集，且 Nordic 並未單獨為 52820 製作開發板，因此可以用 52833 來模擬 52820。此設定 檔限制了 52833 上的硬體資源，使其表現和 52820 相同。 更詳細的資訊牽扯到 DeviceTree，可參考：【詳解 Zephyr

設備數與設備驅動模型】

Configuation

選用的 Kconfig 配置。 Zephyr 的 Kconfig 選單中，很多設定項都是有預設值的。專案內的 prj.conf 設定檔的作用是，將原始的預設配置進行覆寫。通常只選 prj.conf 即可， 如果有不一樣的，可以參考常式的說明檔。

Kconfig fragments

其實和 prj.conf 差不多，相當於 prj.conf 的補充。通常，在同一個常式支援許多不同的板子的情況下，prj.conf 中記錄的是常式通用的配置。

而 boards/xxx.conf 中記錄的是某種開發板單獨需要的配置。boards/下的通常不用選，編譯時會自動依照板子選擇。

另外還可能有其他的設定檔可以選，說明這個常式支援不同的功能，具體需要看那個常式的檔。

Devicetree overlays

系統選擇板子時，板子都會有一個 device tree。這裡的 overlay 就是當前常式對板子 device tree 的增、刪、改。所以叫做覆蓋（overlay）。預設名稱是 app.overlay。 boards/目錄下也可能有不同開發板對應的 overlay 檔。通常也不用選擇，編譯時會自動使用 app.overlay，或自動依板子選擇。

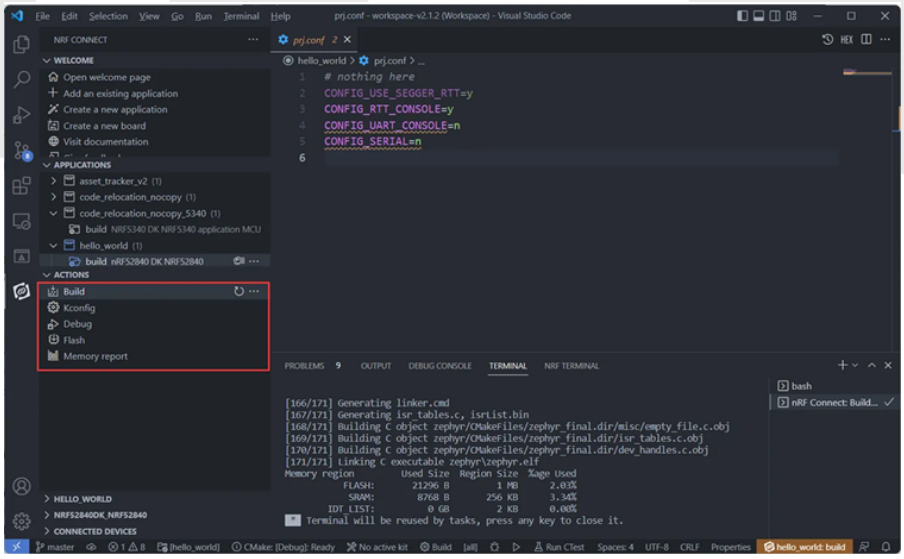
Extra CMake arguments

跟直接在 CMakeLists.txt 裡面寫 set(xxx yyy)定義一個巨集的值，效果是一樣的。格式 是-Dxxx=yyy，也就是在命令列中執行 CMake 時透過-D 進行傳參。

6.2. 進行編譯

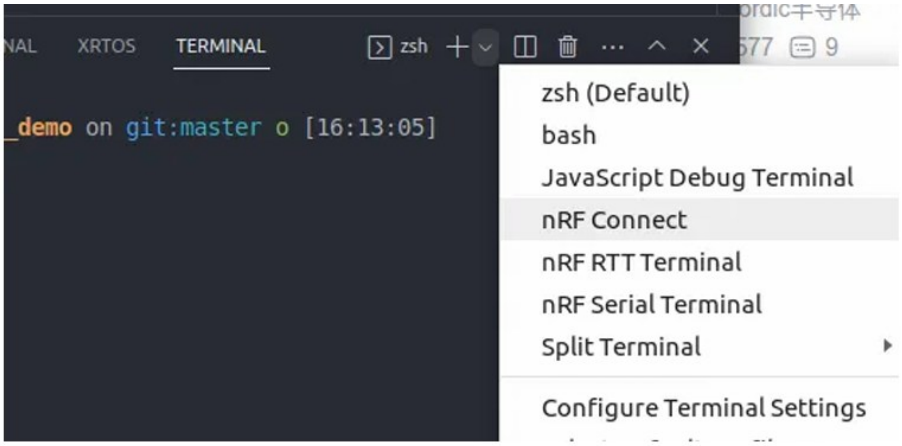
新建完 build target 後，點選 Build Configuration 進行編譯。 如果後續要再編譯這個 target，可以在 APPLICATIONS 欄位選取自己要建置的工程和 target。然後在 ACTIONS 欄中透過 build 按鈕進行專案的建構。

按下 Build 旁的圓圈箭頭按鈕，可以全部重新編譯。



補充：命令列編譯

按" CTRL + ** "，可呼出終端。點選「+」號右邊的下拉箭頭，選擇 nRF Connect：



這樣打開的終端，其環境變數指向前面安裝的 toolchain

利用當前已經創建的 build target 進行編譯

west build

更多用法：

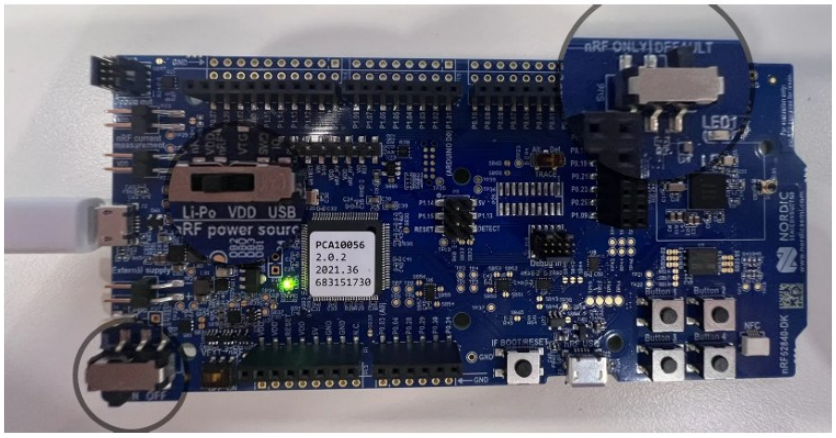
west build -h

編譯時可以指定專案根目錄、build 目錄、板子名稱、設定檔、overlay 檔等。你可以先用上面的圖形化的方式在 VS Code 中進行編譯，然後在 VS Code 終端機中查看這次編譯的命令是什麼。

7.連接設備

nrf-connect 外掛程式，底層呼叫的是 nrfjprog 或 nrfutil 指令來連接開發板上的 JLink。因 此，需要透過 USB 線連接到 JLink 口。

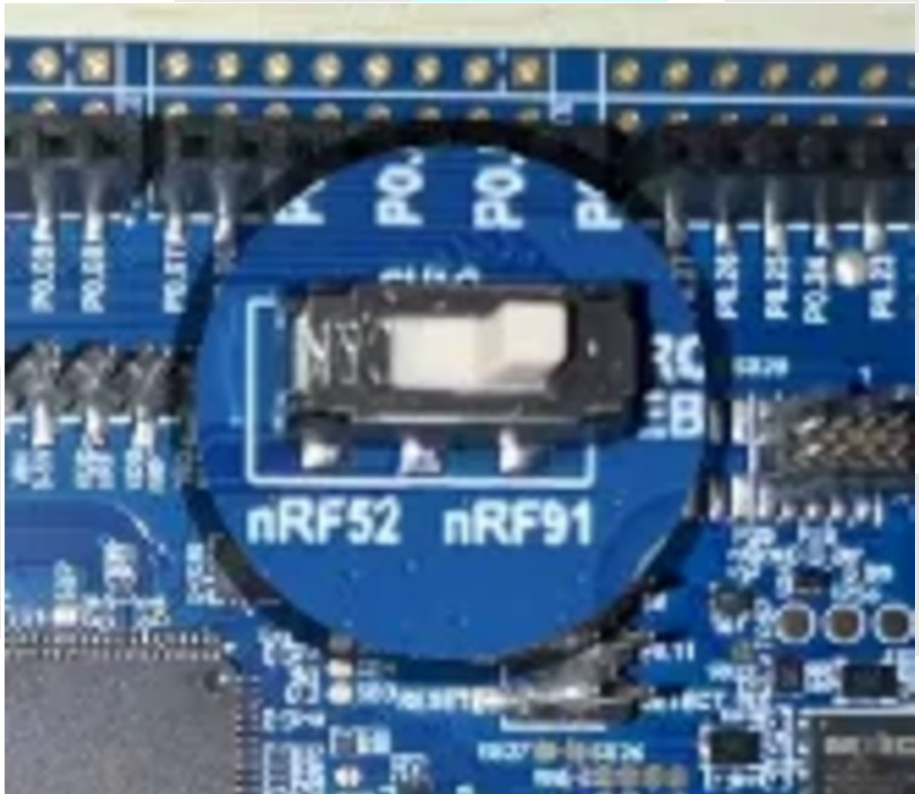
Arad Connectivity Co.,Ltd
TEL: 886-2-2680-0180 FAX: 886-2-2680-0360
OFFICIAL WEBSITE: www.aradconn.com
9F., No. 162, Sec. 3, Jiayuan Rd., Shulin Dist., New Taipei City 238671, Taiwan (R.O.C.) (Business Address)



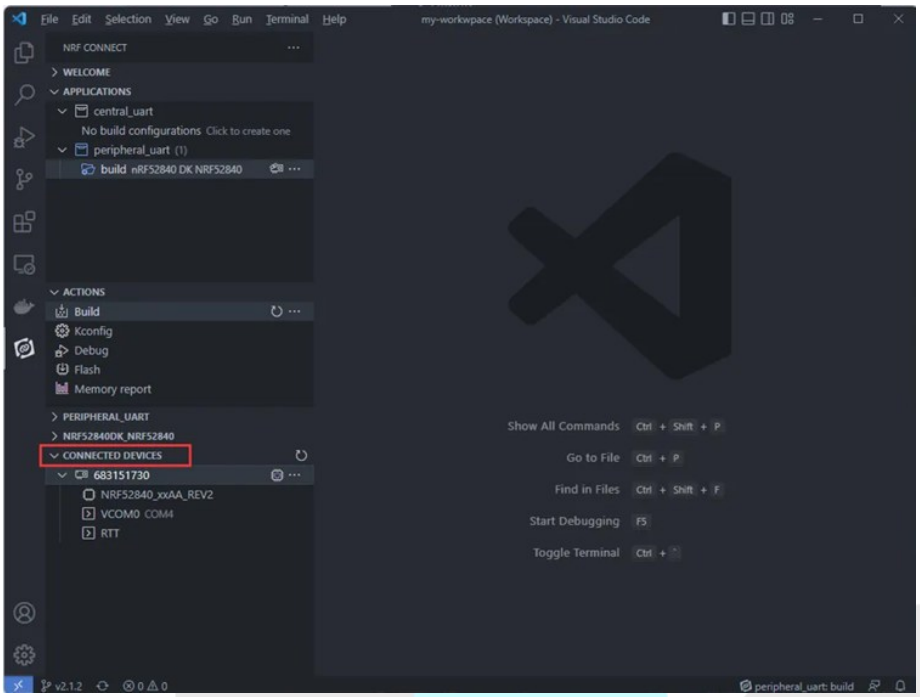
以 nRF52840DK 為例，中間最大的貼有貼紙的晶片為 JLink 主控（官方稱為 Interface MCU），左側為 JLink USB 口，此介面可用來給整塊板供電。

需確保左下角電源開關打開。左側中間位置的開關置於 VDD 擋位，右上角開關置於 DEFAULT 擋位（如上圖）。

對於一些有多顆 MCU 的開發板，請注意使用撥碼開關選擇自己要調試的 MCU，例如 nRF9160DK 可選擇 9160 和 52840：

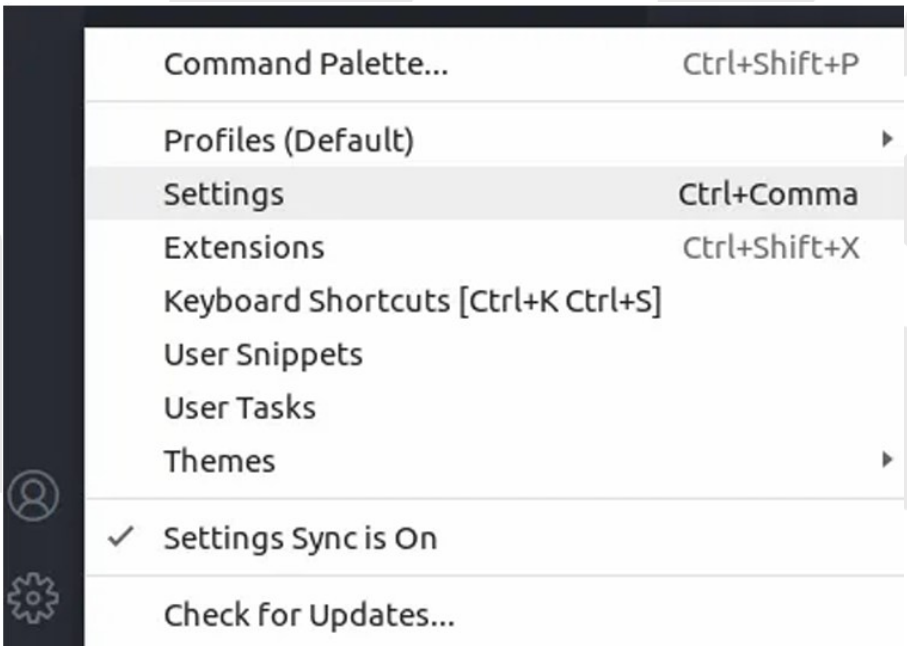


然後就可以在 VS Code 中辨識到設備了：

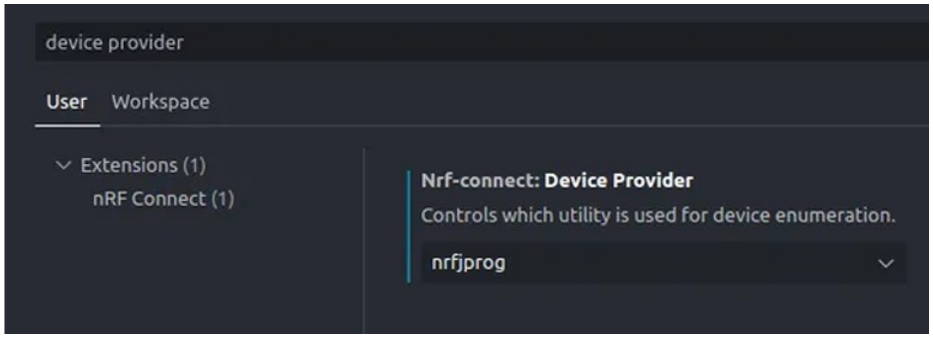


如果不能自動辨識到，或者很不穩定。可能是最新的底層 Python 工具 nrfutil 在 Windows 上不太穩定。可以把它改回之前的 nrfjprog：

開啟 VS Code Settings:

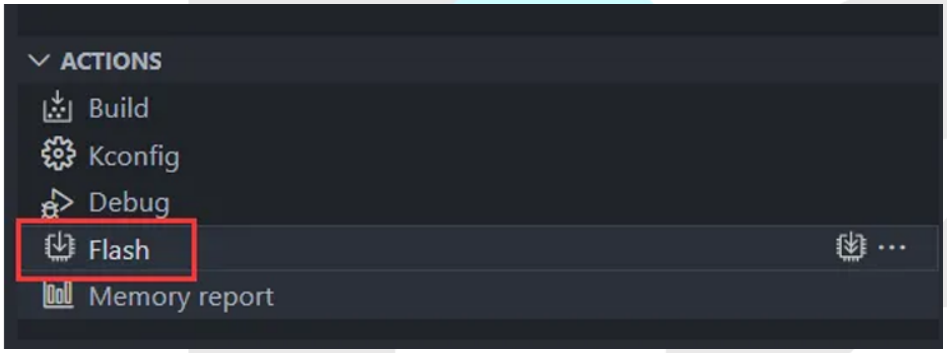


搜尋“Device Provider”，並改為 nrfjprog：



8. 燒錄固件

連線並成功辨識 Jlink 後，可以透過 ACTIONS 欄中的 Flash 按鈕觸發燒錄動作：

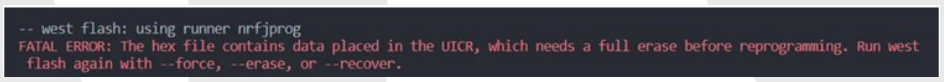


也可以透過命令列進行燒錄:

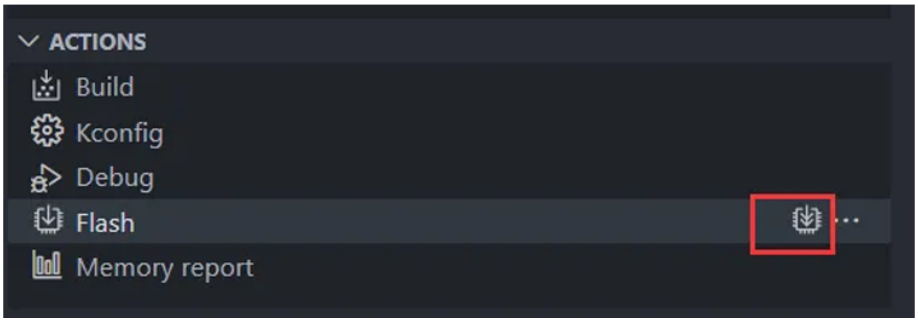
west flash

備註：

這樣直接燒錄，有一部分項目可能會燒寫失敗，顯示：



這是因為，Nordic 的 MCU 中通常都有一個用於儲存使用者資訊的暫存器（UICR)， 可以認為是一塊特殊的 flash 區域，儲存了客戶自己的加密金鑰、引腳配置等產品資訊。由於資訊安全的原因，是不允許在保持 UICR 不變的情況下燒寫新的韌體的。相關 資料，可參考 Nordic 晶片數據手冊的 UICR 章節。 這種情況下只能全片擦除然後再燒錄，點選 Flash 右邊的按鈕：



或使用命令列方式：

west flash --force --erase 此外，還有一種可能是，調試介面啟用了保護，需要 recover 這顆晶片來解除保護。通常，右下角會有彈跳窗問你要不要 recover，就選 Yes 就好。 如果沒有效果，也可以用命令列來 recover

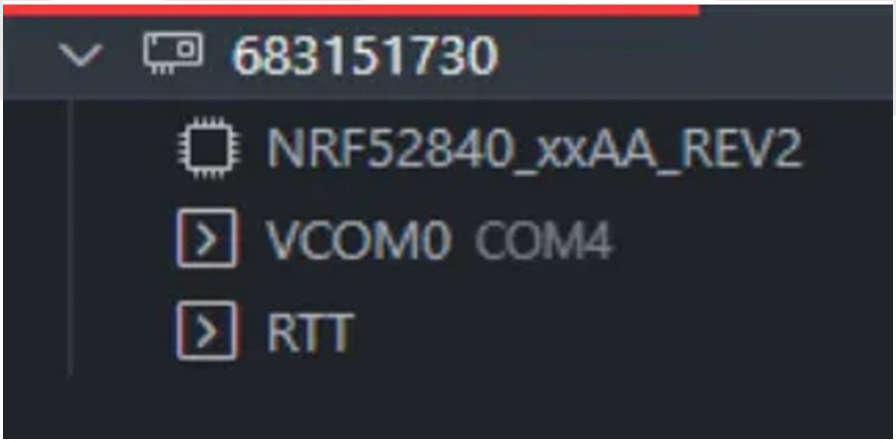
nrfjprog --recover ‘‘ ‘‘‘ ‘shell shell 如果是 nRF5340 這種雙核晶片，那麼網絡核也要 recover 如果是 nRF5340 這種雙核晶片，那麼網絡核也要 recover

‘ ‘‘‘shell

nrfjprog -recover -coprocessor CP_NETWORK

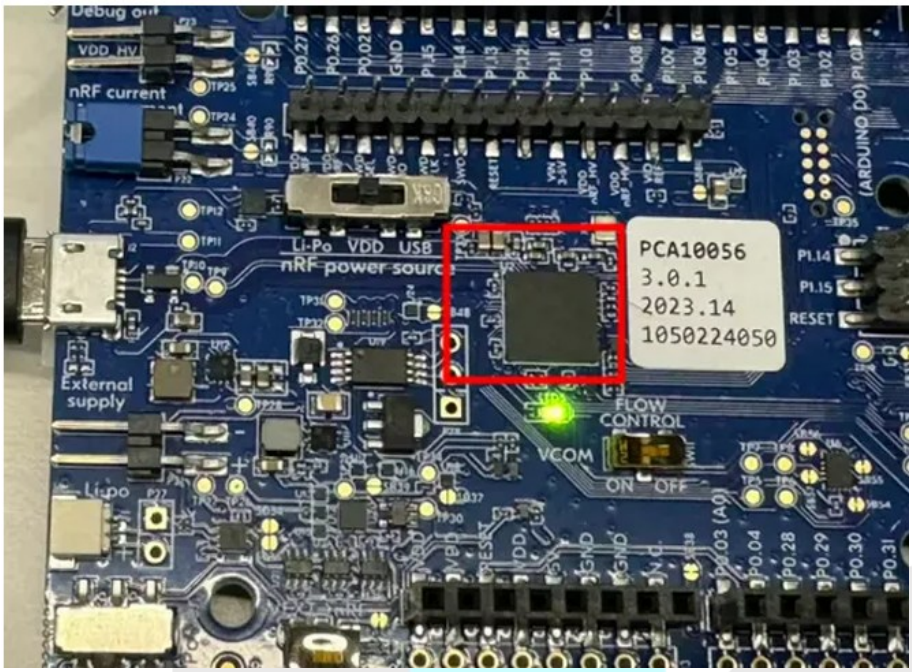
9. 運行並測試

連接的設備，可以看到 Jlink 上的主控晶片、串列埠以及 RTT。



這裡的串列埠是 MCU 上真實的實體串口，在開發板上透過 PCB走線連接到 Jlink，然後 Jlink 把這個串口轉換成 USB 虛

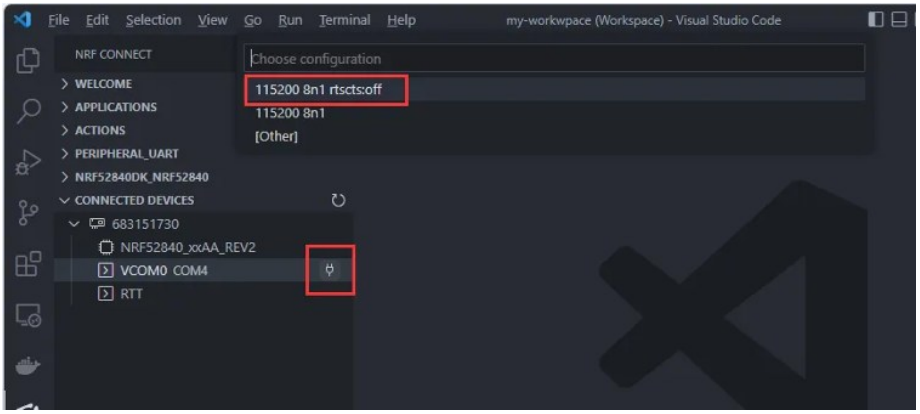
擬串口。 新款開發板，板載的 Jlink 是拿 5340 做的，這種新開發板有兩個 USB 虛擬串列埠：



但對於 52840DK 來說，開發板上只連了一個串口，另一個是空的。具體是哪個？要試 一下，因為可能 USB 枚舉的順序不一樣。

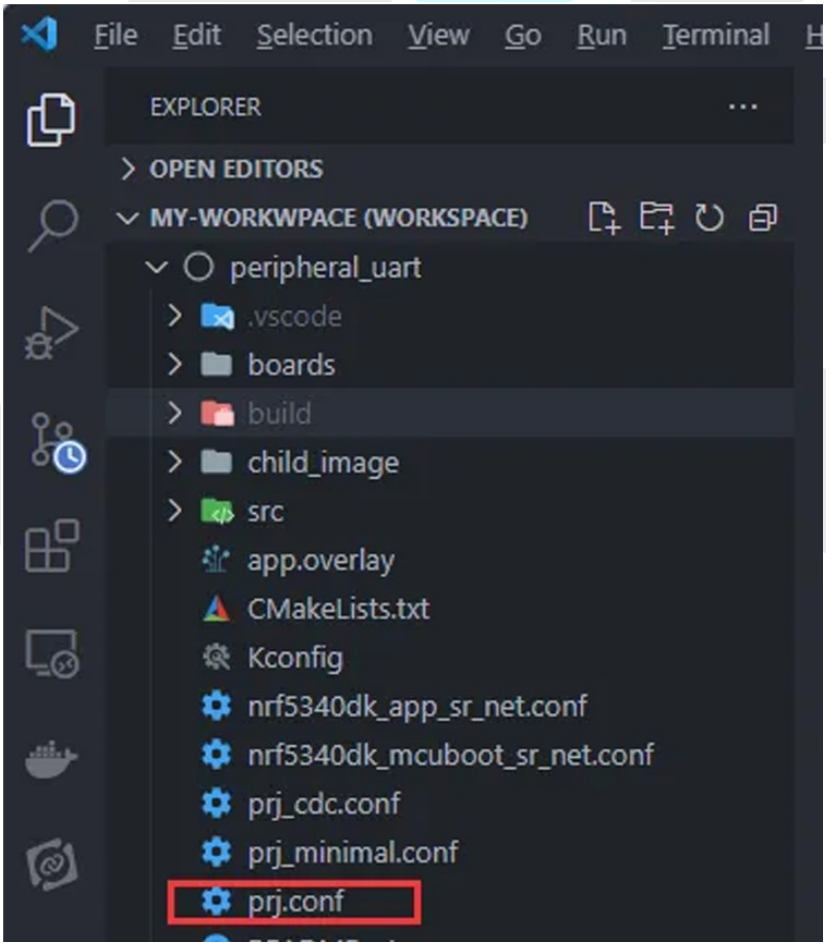
對於 5340DK, 7002DK 來說，兩個串列埠分別對應 Application Core 和 Network Core 的日誌輸出。

9.1. 連接串口



點擊串口，選擇串列傳輸速率，即可開啟串口。串列埠接收的訊息在 Terminal 展示： 這個串列工具類似於 Putty，按下鍵盤的按鍵就立即發送出去一個字元，不會顯示自己 發出了什麼。便於在這個串列埠上執行命令列終端機之類的，這也是 Zephyr 所支援 的。 9.2. 連接 RTT

RTT 是 Segger 提供的日誌調試手段，全名為 Real Time Transmit。 MCU 將日誌印到內部快取中，然後利用 Jlink 的高速通道，把日誌印到電腦上。這個方法不需要佔用串口 外設，而且速度極快，對 CPU 運作影響小。 大多數常式的預設日誌輸出口是串列埠。但本常式是藍牙串列口透傳，串列埠需要傳 輸使用者數據，因此日誌的預設輸出就是 RTT。 查看 RTT 日誌輸出的相關配置： 開啟工程根目錄下的.prj 檔：



可以看到：

```
CONFIG_USE_SEGGER_RTT= y        # 啟用 RTT
CONFIG_LOG_BACKEND_RTT = y      # 日志後端選用 RTT
CONFIG_LOG_BACKEND_UART= n     # 日志後端不選用串口
CONFIG_LOG_PRINTK= n             # 不啟用 PRINTK 日志
```

9.3. 測試 peripheral_uart 常式

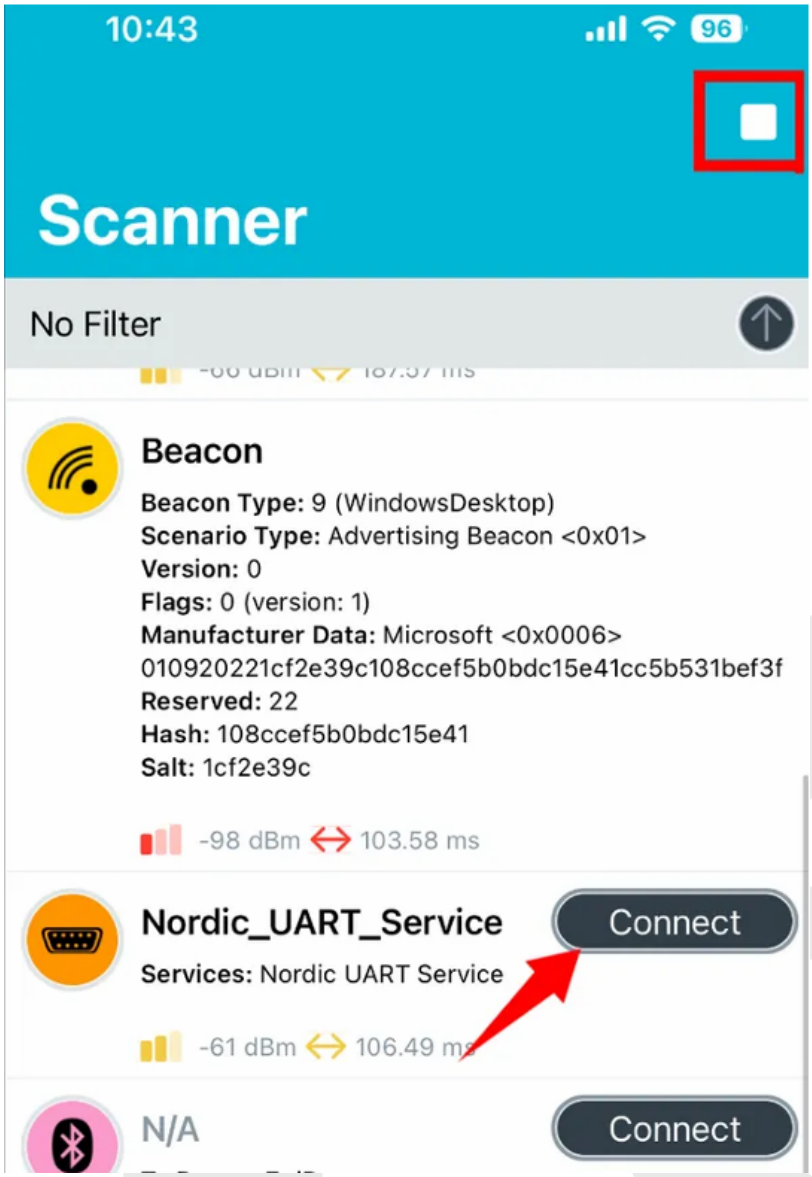
一般來說，需要兩塊開發板，一塊燒 peripheral_uart，一塊燒 central_uart。兩塊開 發板上電後會自動連接。從一個開發板串口輸入的數據，會從另一個開發板輸出。

但這裡我們只有一塊開發板，那 BLE central 我們就用手機。 iOS 應用程式商店可以下 載 nRF Connect，安卓可以在 Google 商店下載，或是直接去 Github 下載 APK。

透過 BLE 連接設備

在 nRF Connect APP 中，先掃描，掃到設備後，再連接：

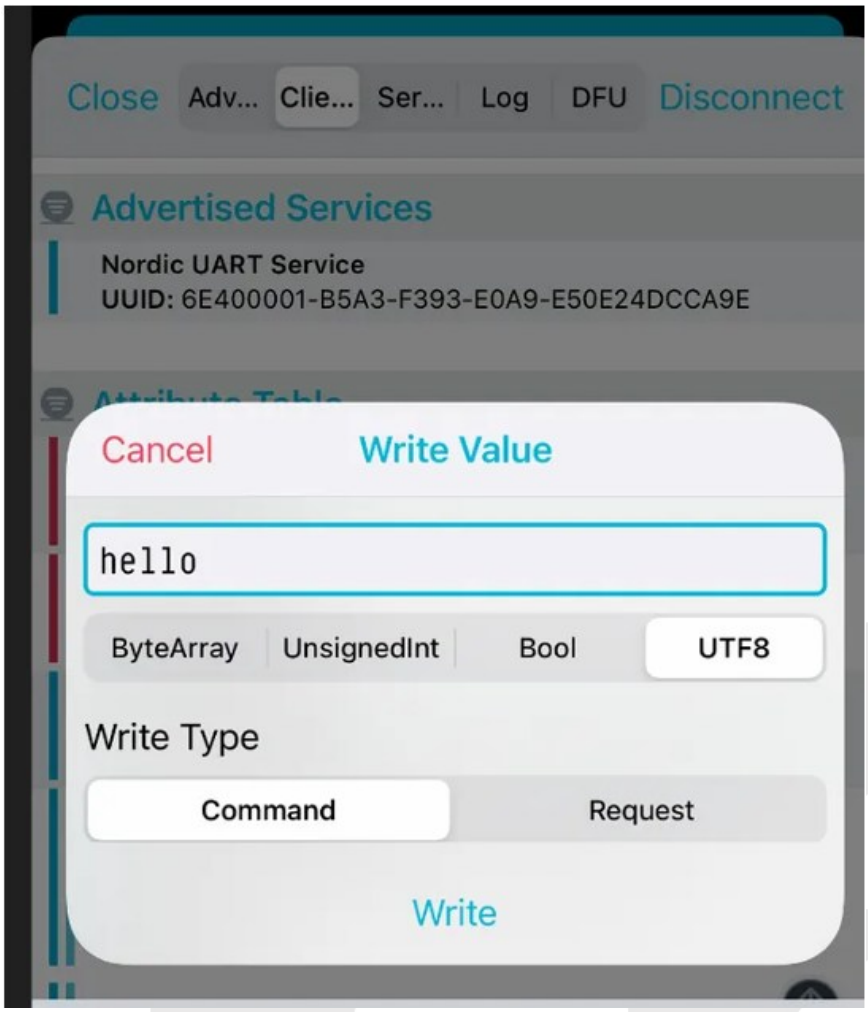
Arad Connectivity Co.,Ltd
TEL: 886-2-2680-0180 FAX: 886-2-2680-0360
OFFICIAL WEBSITE: www.aradconn.com
9F., No. 162, Sec. 3, Jiayuan Rd., Shulin Dist., New Taipei City 238671, Taiwan (R.O.C.) (Business Address)



開發板接收數據

Arad Connectivity Co.,Ltd
TEL: 886-2-2680-0180 FAX: 886-2-2680-0360
OFFICIAL WEBSITE: www.aradconn.com
9F., No. 162, Sec. 3, Jiayuan Rd., Shulin Dist., New Taipei City 238671, Taiwan (R.O.C.) (Business Address)

Arad Connectivity Co.,Ltd
TEL: 886-2-2680-0180 FAX: 886-2-2680-0360
OFFICIAL WEBSITE: www.aradconn.com
9F., No. 162, Sec. 3, Jiayuan Rd., Shulin Dist., New Taipei City 238671, Taiwan (R.O.C.) (Business Address)



可以在串口看到資料：

開發板發送數據

BLE 協定是 Client-Server 架構。 BLE 協定規定，從機作為 Server，只能被 Client 讀取、 寫上面的屬性。預設情況下不能主動發送訊息到 Client。除非 Client 使能了 Notify 的功 能，Server 才能 Nortify 到 Client。更多資訊，大家可以搜尋 CCCD(Client Characteristic Configuration Descriptor)。這裡，就需要點亮 TX 屬性的 CCCD：

然後在串口中透過鍵盤輸入內容：hello+ abc。螢幕上不會顯示東西，但是按鍵確實 會送出。 這個串列工具類似於 Putty，按下鍵盤的按鍵就立即發送出去一個字元，不會在螢幕上 顯示自己發出了什麼。

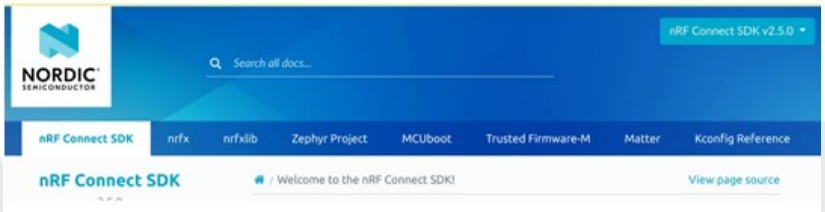
這裡之所以要加回車，是因為常式程式碼就是這麼寫的。在串口回呼函數內，偵測到 回車，才會把串列埠資料打包從藍牙發出。

至此，我們完成了在 nRF52840DK 上的 peripheral_uart 常式的編譯、燒錄與運行測試。

10. 官方資料

nRF Connect SDK 官網

https://docs.nordicsemi.com/bundle/ncs-latest/page/nrf/index.html



進入官網，首先看到右上角可以選擇檔的版本，需要與 SDK 的版本對應。 然後可以看到中間一排的標籤頁：

Zephyr Project：是 Zephyr 官方檔的鏡像，包含 Zephyr RTOS 核心服務、作業 系統 API、各種驅動、協定支援以及它們的常式檔。一些比較通用的功能的如日誌、Flash 儲存、線程間通訊等功能的檔都在這裡面。它對應的是 nRF Connect SDK 中的 zephyr 資料夾。

nRF Connect SDK:是 Nordic 在 Zephyr 系統上擴充的各種 Nordic 獨有的函式庫、 驅動程式和常式的檔。裡面大多數是一些 Nordic 獨有的技術，對應的是 nRF Connect SDK 中的 nrf 資料夾。

nrfx 與 nrfxlib：Nordic 的周邊驅動程式庫，是最接近暫存器操作的一層，和目前 已經停止維護的的 nRF5 SDK 中的 nrfx 幾乎是一樣的。在 Zephyr 中，通常應用層 只需呼叫 Zephyr 的標準 API，Nordic 提供的底層驅動會把 nrfxlib 和

一些暫存器操作封裝成 Zephyr 的標準 API。通常，只有客戶在對 MCU 週邊功能進行較為深入的開發時，會參考到這一塊的文件。

MCUboot：MCUboot 是一個開源的協力廠商安全 bootloader，支援許多系統和平臺，Zephyr 只是其中之一。許多支援 OTA 的常式基本上都是使用 MCUboot

Trust Firmware-M：ARM 提出了平臺安全架構（Platform Security Architecture, PSA），意思是說，客戶自己開發軟體容易有安全性漏洞，因此運作環境應分為安全環境（SPE）和非安全環境（NSPE）。客戶開發的程式，屬於非安全環境。安全環境的程式，由廠商提供，主要提供一些安全儲存、安全啟動之類的 API 給客戶的非安全環境來呼叫。 Trust Firmware-M(TFM)是安全環境的一個樣板韌體。 如果你使用了 nRF5340 或 nRF9160 這種有 ARM v8 架構的主控平臺，則在編譯選板子時，都可以看到_s 或_ns 後綴。_s 的意思是說，客戶直接在安全環境開發程式，安全性全由客戶自己掌控。_ns 的意思是說，客戶在非安全環境開發程式，編譯時，Zephyr 會自動把 TFM 一起編譯進去，和客戶的應用程式一起工作。對於 9160 來說，由於要和蜂窩 modem 進行交互，因此，牽扯到蜂窩網絡操作的常式，都必須選擇 nrf9160dk_nrf9160_ns。

Matter：Matter 是智慧家庭的新標準，目的是打破廠商之間的壁壘，實現生態融合。從連接方式上講，Matter 是基於區域網路 IPv6 的，因此，Wi-Fi 和 Thread 都是可以作為 Matter 的底層的。從配網方式上講，Matter 透過 BLE 來傳輸認證訊息，此外可以透過 NFC 或二維碼的方式，讓手機快速的找到要配網的這個裝置的 BLE 廣播。此頁面主要是 Matter SDK 的文檔，並不限於在 Nordic MCU 上進行開發。如果要找 Matter 在 Nordic 產品上運行的常式，還是要去 nRF Connect SDK 頁面的 Samples 目錄下去找。

Kconfig：Zephyr 系統中有大量的 Kconfig 配置，Nordic 擴充的函式庫、驅動程式中也有大量 Kconfig 配置。如果你不知道一個 Kconfig 設定是做什麼的，可以在這個頁面搜尋。

總之，nRF Connect SDK 官網裡面有大量的技術細節，在運行一個常式之前，一定要參考網站中該常式的說明進行操作。

Nordic 資料中心 https://docs.nordicsemi.com/

目前最新的資料中心，可以透過技術或產品系列進行分類，尋找想要的資料。晶片數據手冊（Specification）、開發板說明都可以在這裡查看。也會導向到 nRF Connect SDK 官網。

Nordic 官網 https://www.nordicsemi.com/

一些商業新聞和產品介紹。但最重要的是一些工具軟體、開發板原理 圖/PCB/BOM 之類，需要在這裡下載。 例如：nRF52840DK 開發板預設程式、Jlink 韌體、原理圖等： https://www.nordicsemi.com/Products/Development-hardware/nRF52840 DK/Download?lang=en#infotabs

DevZone 開發者論壇 https://devzone.nordicsemi.com/

有問題可以在上面搜索，也可以用英文提問。每天都有原廠 support team 查看問 題並回覆。 Nordic 註冊客戶，也可以提交 private ticket，解決一些與程式碼、板子有關的問題，也可以簡單審核 PCB。