這份安裝說明完成後可實現的功能有以下兩點

1. 在 Win64 系統中執行 Server 端及兩個 Client 端程式，進而完成聯邦學習式監督離散雜湊模型的訓練(只能進行模型好壞評估指標及速度的測試)。
2. 在 Win64 系統中執行 Server 端同時在 Nano 及 Rasp 執行 Client 端程式，並利用 Win64 系統中的 WSL2 (Windows Subsystem for Linux)執行 rsync 指令互相傳遞權重達到聯邦式雜湊模型的訓練(進行模型好壞評估、速度、功耗和占用記憶體量的測試)。

Win64 系統安裝步驟

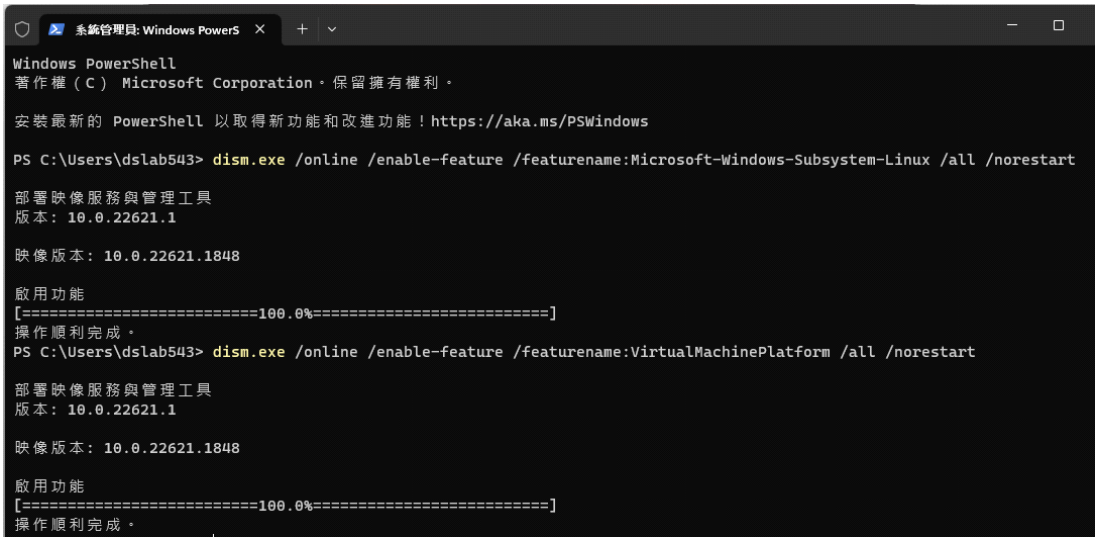
1 設定 Server 端的環境

1.1 在 Windows 上安裝 WSL

- 按 Win+X，啟動 終端機(系統管理員)

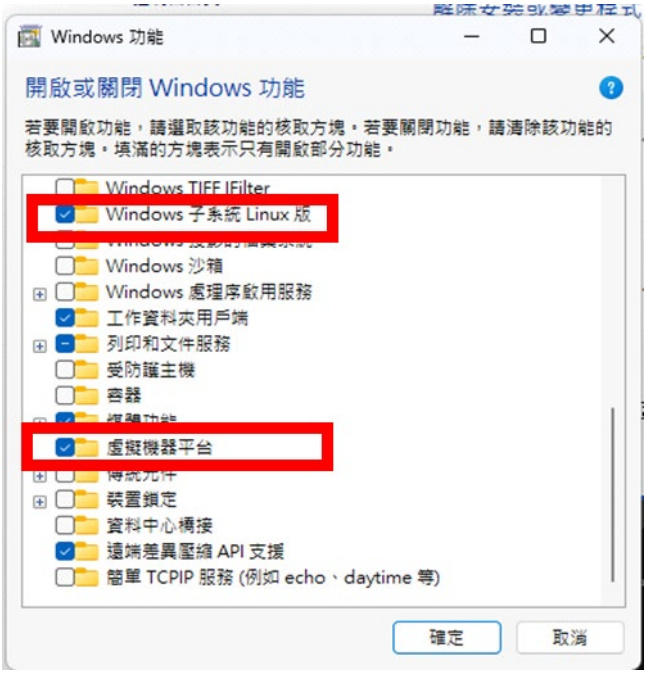
- 在 windows terminal 視窗中，輸入如下指令來啟用 WSL 服務：

```
# 開啟 linux 子系統

dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart

#開啟虛擬機器平臺

dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```



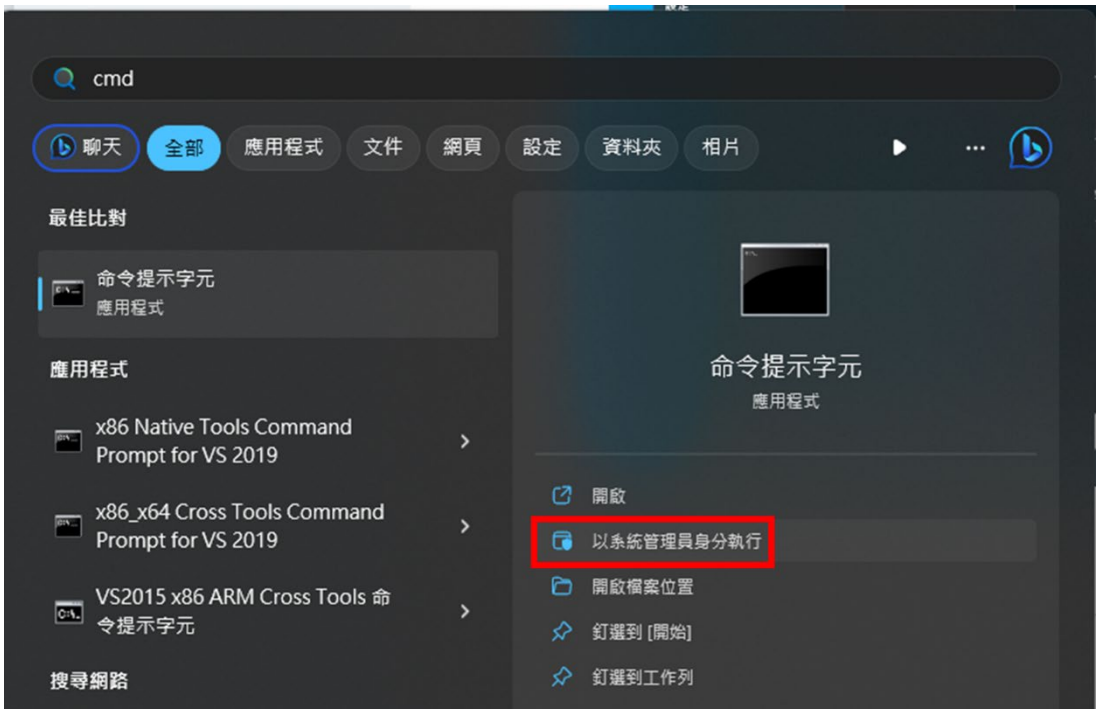
- 按下 Win + R，調出命令輸入視窗。輸入指令 appwiz.cpl。
- 點選左側的【啟動或關閉 Windows 功能】:

- 彈出下面這個視窗：需要勾選【適用於 Linux 的 Windows 子系統】和【虛擬機器平臺】這兩項。



1.2 安裝 Ubuntu 作業系統，並設定 WSL (Linux 的 Windows 子系統)的版本為 WSL2

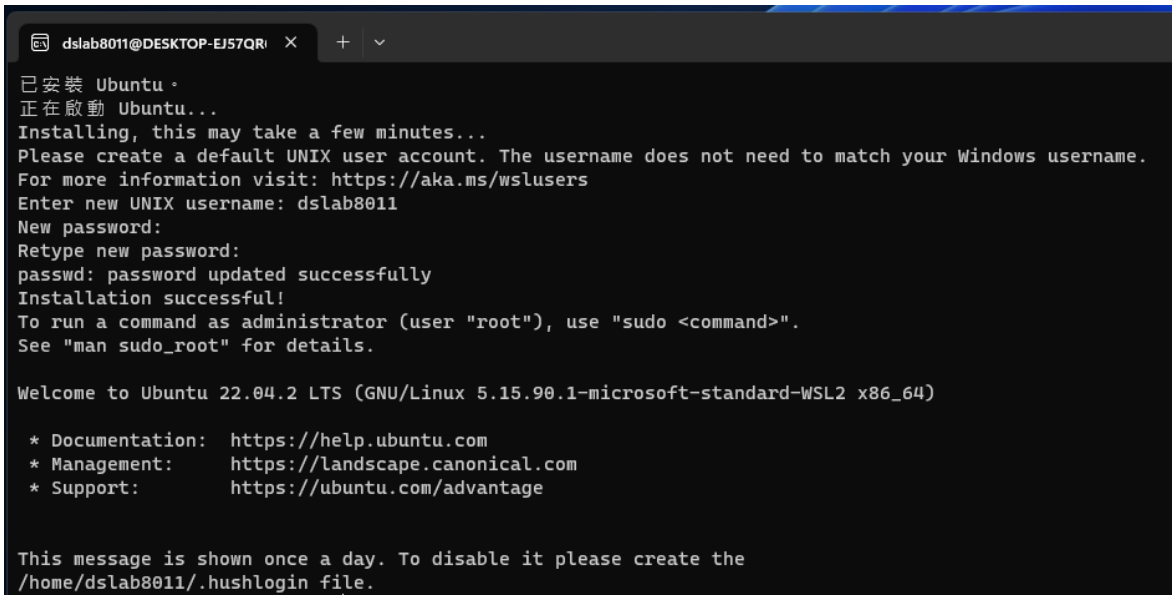
- 首先以**系統管理員**身份執行 Windows 的命令提示字元(cmd)




```
# 安裝 WSL(Windows Subsystem for Linux)系統

wsl --install
```

- 重新開機後會自動開啟命令提示字元來完成後續的安裝。安裝好後要設定 Linux 一般使用者的名稱和使用者的密碼(這邊設定的**密碼**需跟 C:\程式包路徑\程式包名\Server\SendOrReturnWeight.sh)中的 rsync 裡的

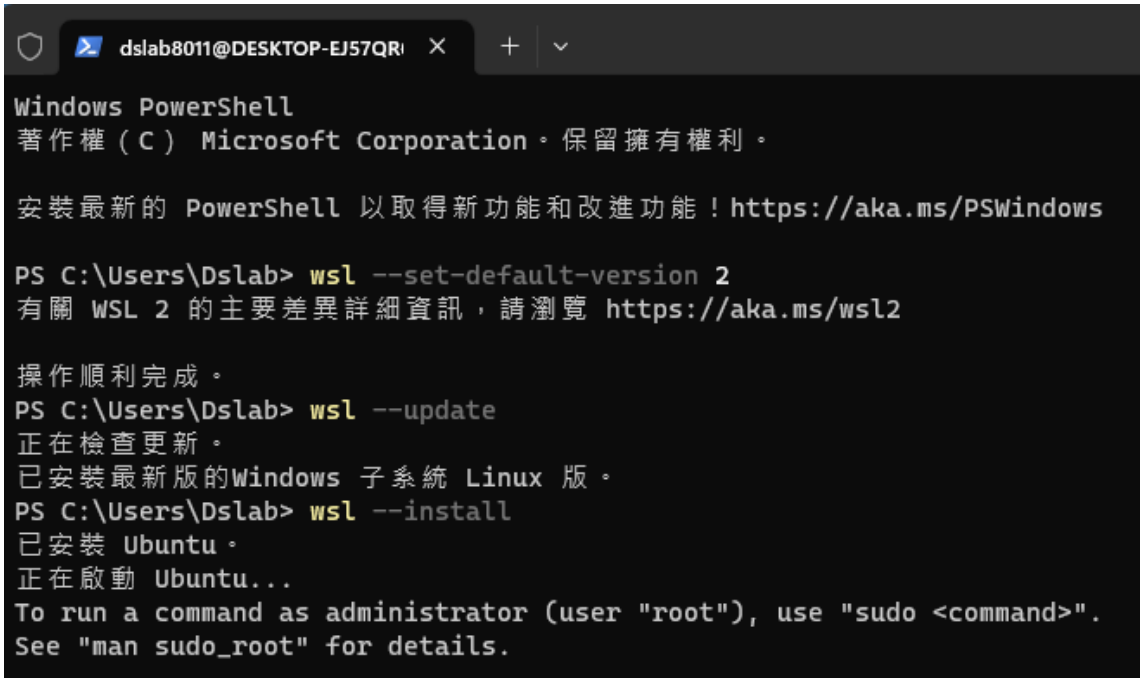
sshpass -p **密碼** 一致。



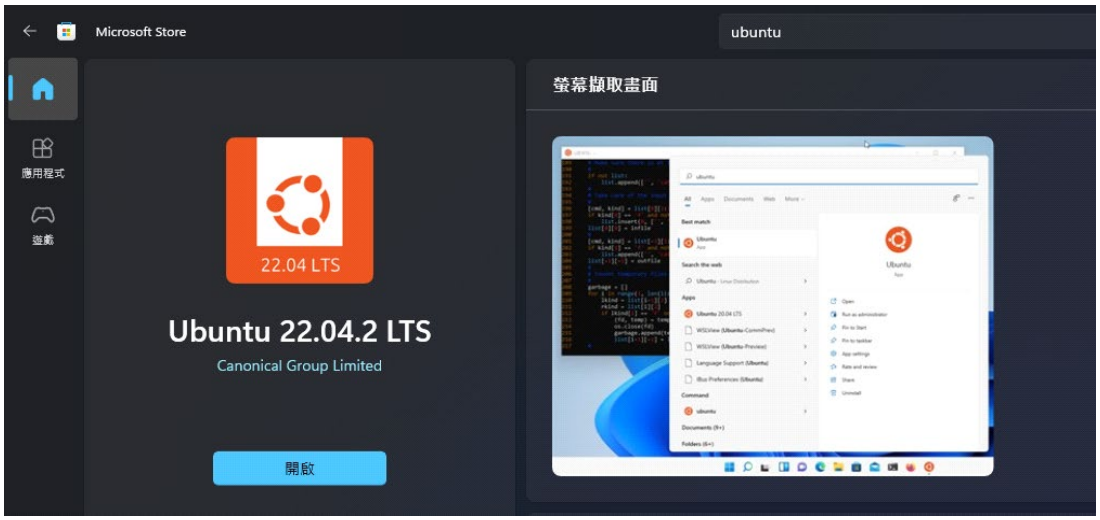
- 按 Win+X，**啟動** 終端機(系統管理員)


```
# 確認 WSL 是否為最新版

wsl --set-default-version 2

# 執行更新 wsl 命令

wsl --update
```

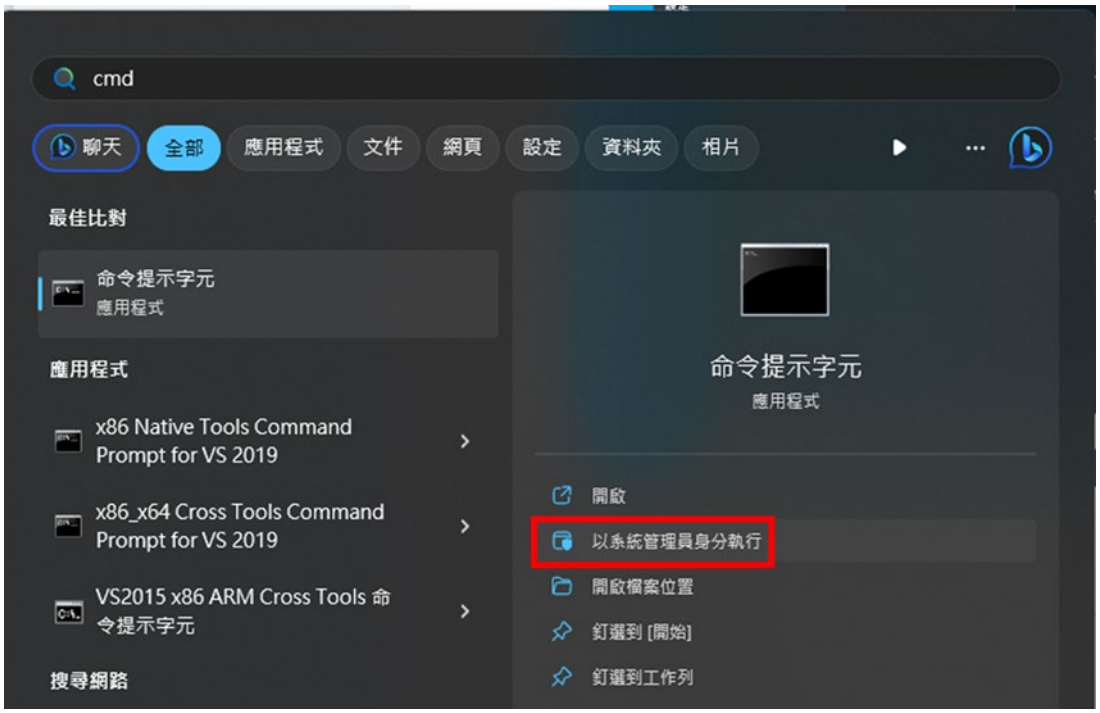
```
wsl --install
```



- 開啟【Windows Store】並搜尋 Ubuntu，然後安裝 Ubuntu 22.04.2 LTS 的系統

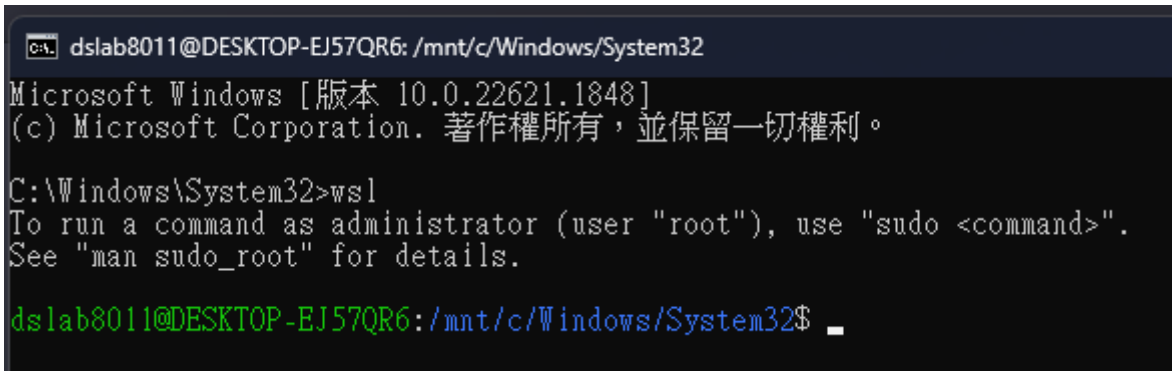


- 以系統管理員身份執行 Windows 的命令提示字元(cmd)



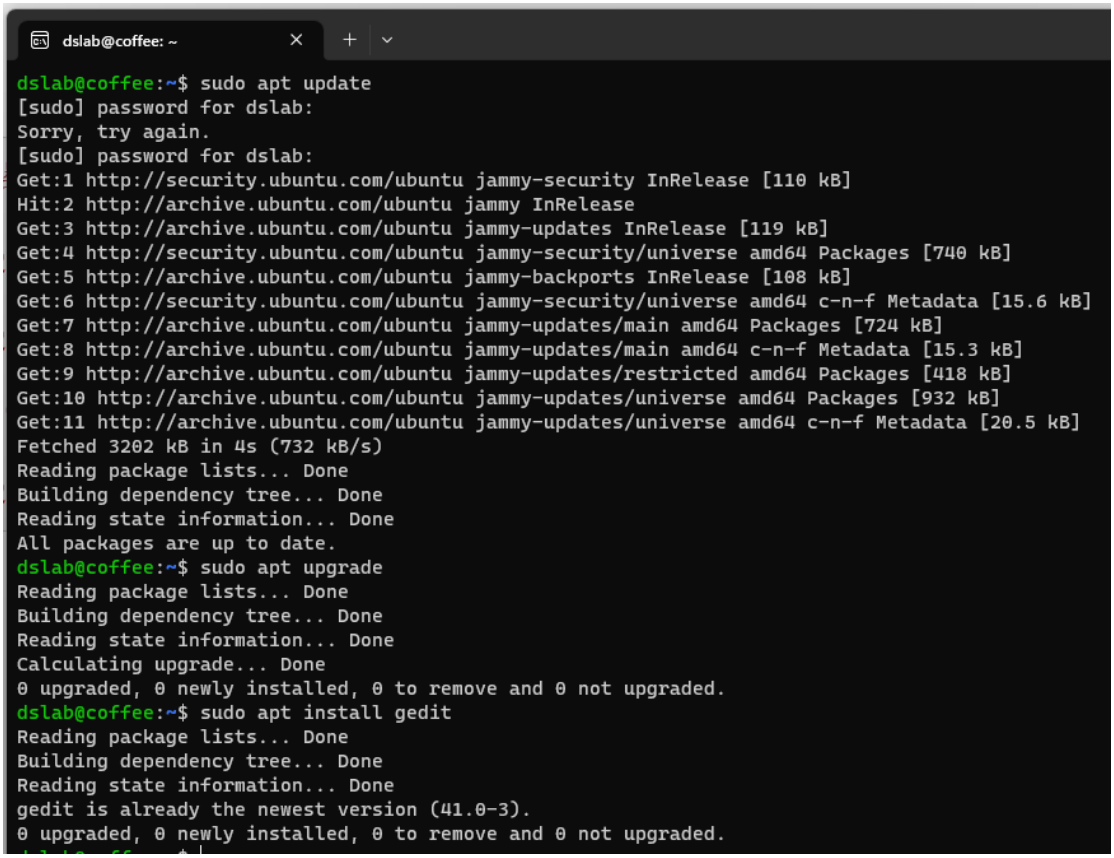

```
# 於 cmd 上開啟 WSL 系統

wsl
```

- 開啟後可以看到在 Linux 子系統中，Windows 的磁碟機會被掛載到 Linux 作業系統的/mnt 目錄下



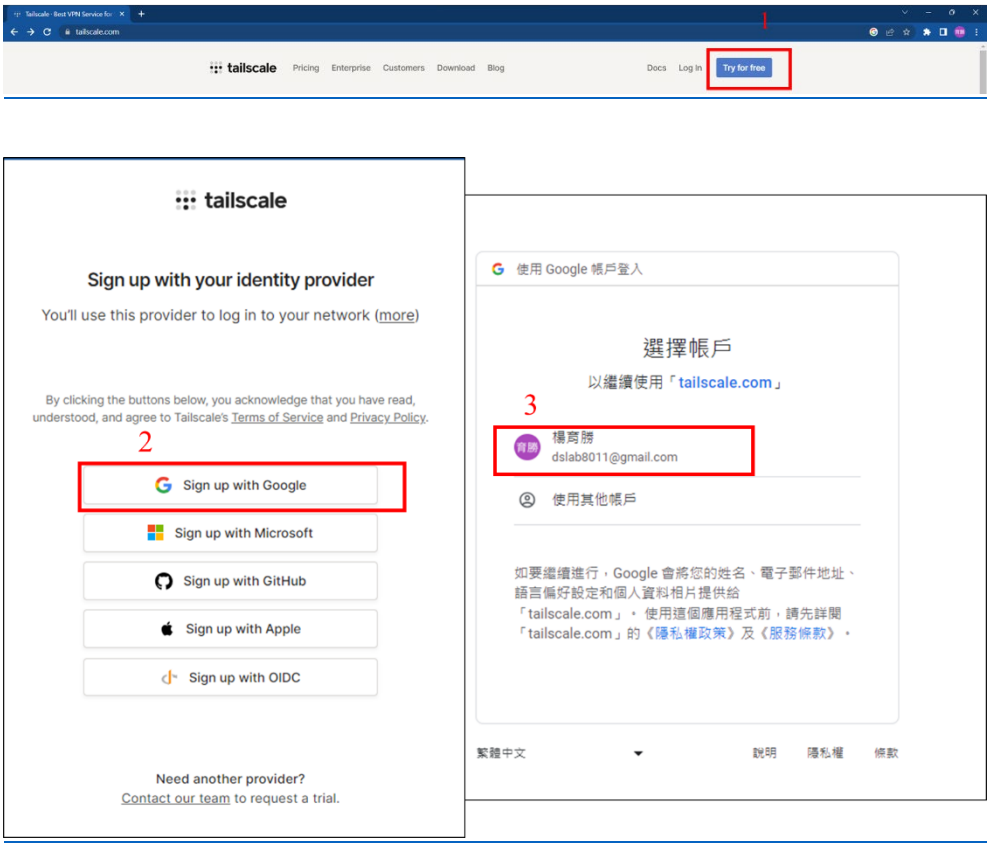
- /mnt/c/Windows/System32 => 這一串很重要當使用嵌入式版訓練時需更改 initpath.h 中的變數將它改為這一串

```
# 安裝 GUI 程式並測試

sudo apt update

sudo apt upgrade

# gnome 桌面下的編輯器

sudo apt install gedit
```



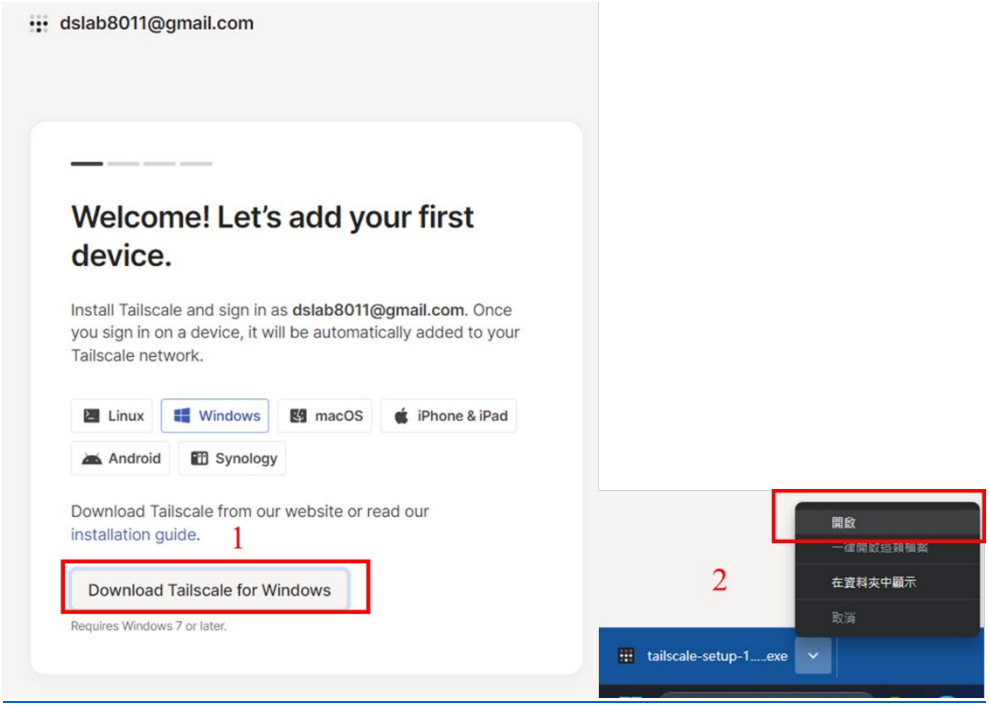
1.3 安裝 tailscale(VPN)

- 申請帳號➢ 紅框 1：Try for free ➜ 紅框 2：選擇 google 帳號登入即可 ➜ 紅框 3：點擊自己帳號



- tailscale 1.44.0 (Win10 下載連結如下) ➢ 紅框 1：Download Tailscale for Windows➔ 紅框 2：下載完後選擇開啟，照指示安裝即可

  https://pkgs.tailscale.com/stable/tailscale-setup-latest.exe



1.4 安裝 Visual Studio 2019

- Visual Studio 2019 16.11 版版本資訊(下載連結)

https://download.visualstudio.microsoft.com/download/pr/7c09e2e8-2b3e-4213-93ab-5646874f8a2b/0ac797413a56c6b2772f48a567a32cdddd3b739f5b2af649fcf90be4245762ff/vs_Community.exe

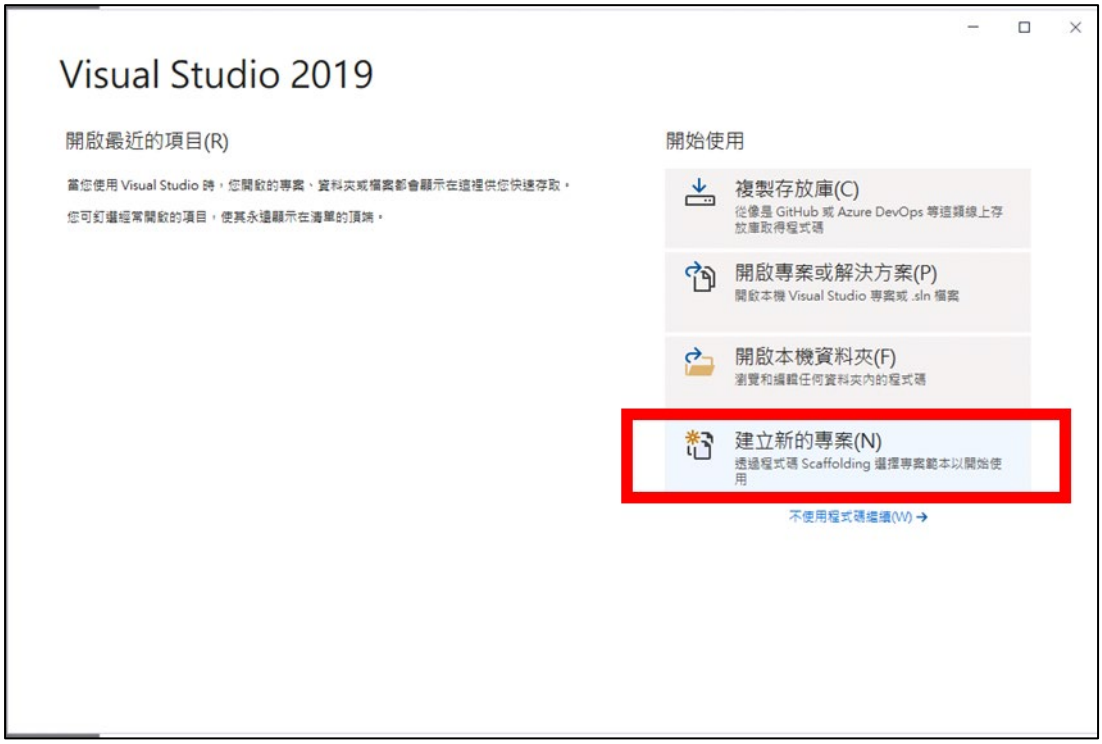
- 安裝套件：使用 c++ 的桌面開發 =>紅框的開發套件包全部打勾 => 按下修改


修改中 — Visual Studio Community 2019 — 16.11.27
工作負載
個別元件
語言套件
安裝位置
Web 與雲端 (4)
ASP.NET 與網頁程式開發
使用 ASP.NET Core、ASP.NET、HTML/JavaScript 及容器 (包括 Docker 支援) 建立 Web 應用程式。
Azure 開發
用於使用 .NET 和 .NET Framework 開發雲端應用程式及建立資源的 Azure SDK、工具及專案。同時包含用於將應用...
Python 開發
對 Python 進行編輯、偵錯、互動式開發及原始檔控制。
Node.js 開發
使用非同步的事件驅動 JavaScript 執行階段 Node.js 建置可調整的網路應用程式。
傳統型與行動裝置 (5)
.NET 桌面開發
使用 C#、Visual Basic 及 F#，利用 .NET 和 .NET Framework 建置 WPF、Windows Forms 與主控台應用程式。
使用 C++ 的桌面開發
使用您選擇的工具 (包括 MSVC、Clang、CMake 或 MSBuild)，建置適用於 Windows 的新式 C++ 應用程式。
通用 Windows 平台開發
使用 C#、VB 或選用 C++，來建立適用於通用 Windows 平
Mobile development with .NET (out of support)
使用 Xamarin 建置適用於 iOS、Android 或 Windows 的跨
安裝詳細資料
Visual Studio 核心編輯器
使用 C++ 進行行動開發
使用 C++ 的桌面開發
已包含
C++ 核心桌面功能
選擇性
MSVC v142 - VS 2019 C++ x64/x86 建置工...
Windows 10 SDK (10.0.19041.0)
Just-in-Time 偵錯工具
C++ 分析工具
適用於 Windows 的 C++ CMake 工具
適用於最新 v142 建置工具的 C++ ALT (x8...
適用於 Boost.Test 的測試配接器
適用於 Google Test 的測試配接器
Live Share
IntelliCode
C++ AddressSanitizer
MSVC v142 - VS 2019 C++ ARM64 建置工...
位置
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community
繼續進行即表示您同意所選 Visual Studio 版本的授權。我們也可讓您使用 Visual Studio 下載其他軟體。此軟體為分開授權，如同協力廠商聲明或其隨附的授權中所述。繼續進行即表示您也同意該授權。
總共
需空間 8.09 GB
在下載時安裝
修改(M)

1.5 設定 Server 端上的程式

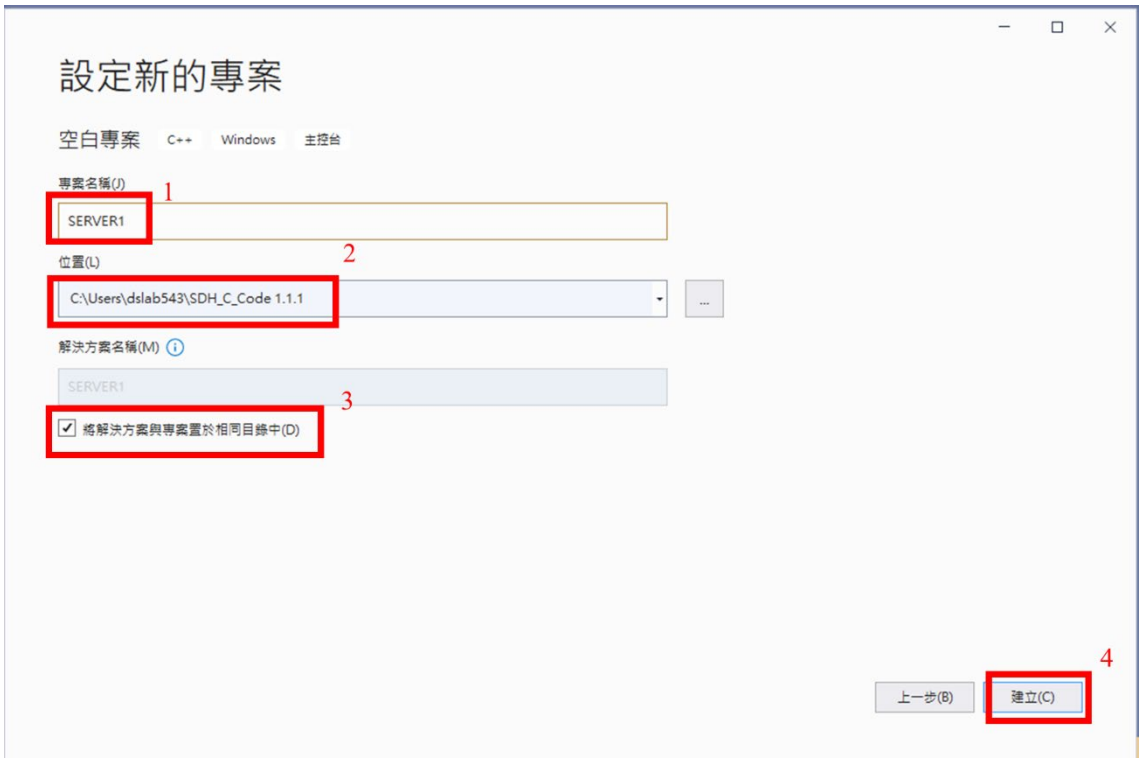
- 打開 Visual Studio 2019

步驟 1. 建立新的專案



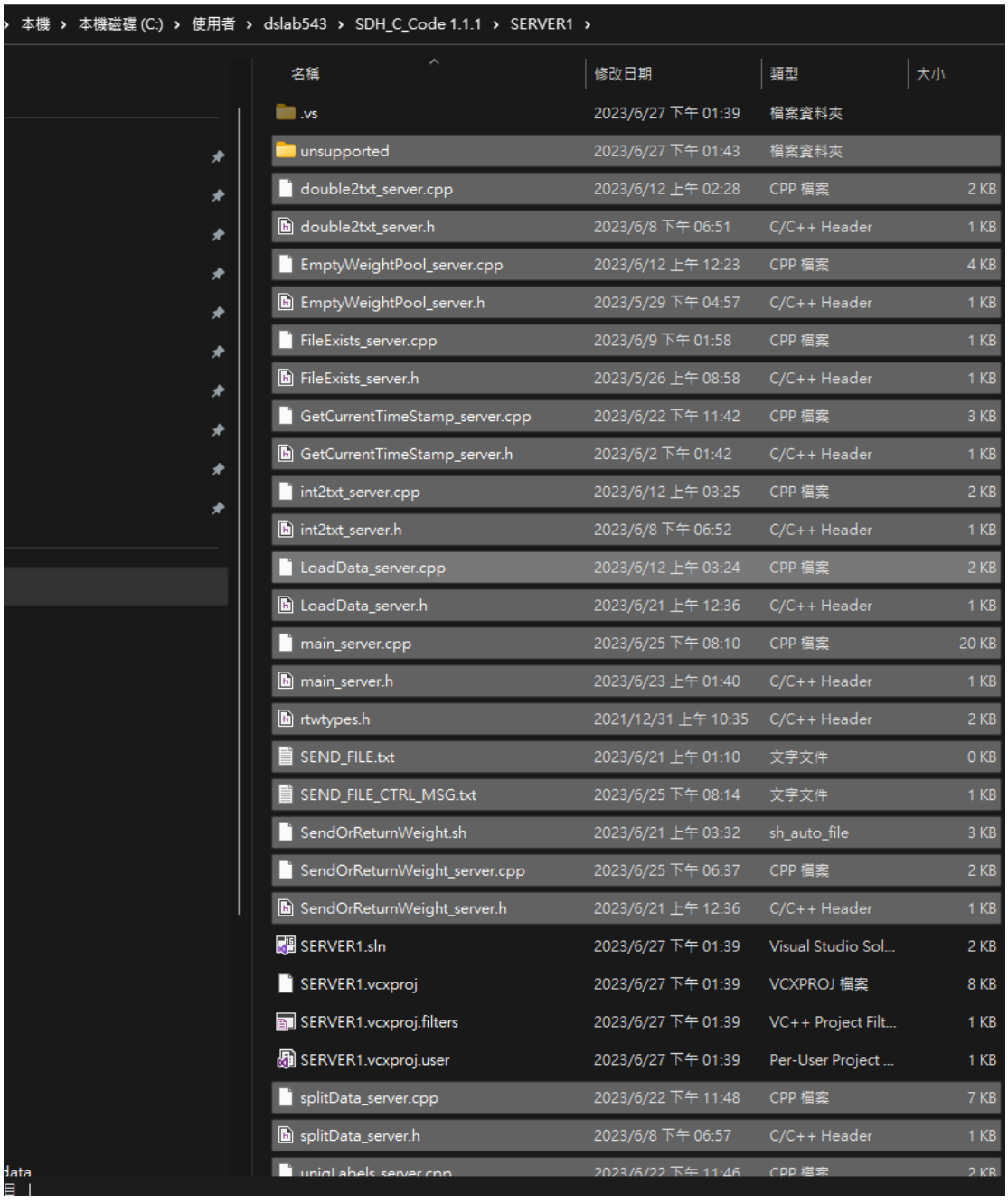
步驟 2. 空白專案 ➜ 下一步

步驟 3.　設定專案路徑➢紅框 1：設定專案名稱 **SERVER1**➜紅框 2：設定專案路徑至程式包下一層目錄➜紅框 3：勾選解決方案與專案置於相同目錄中➜紅框 4：建立

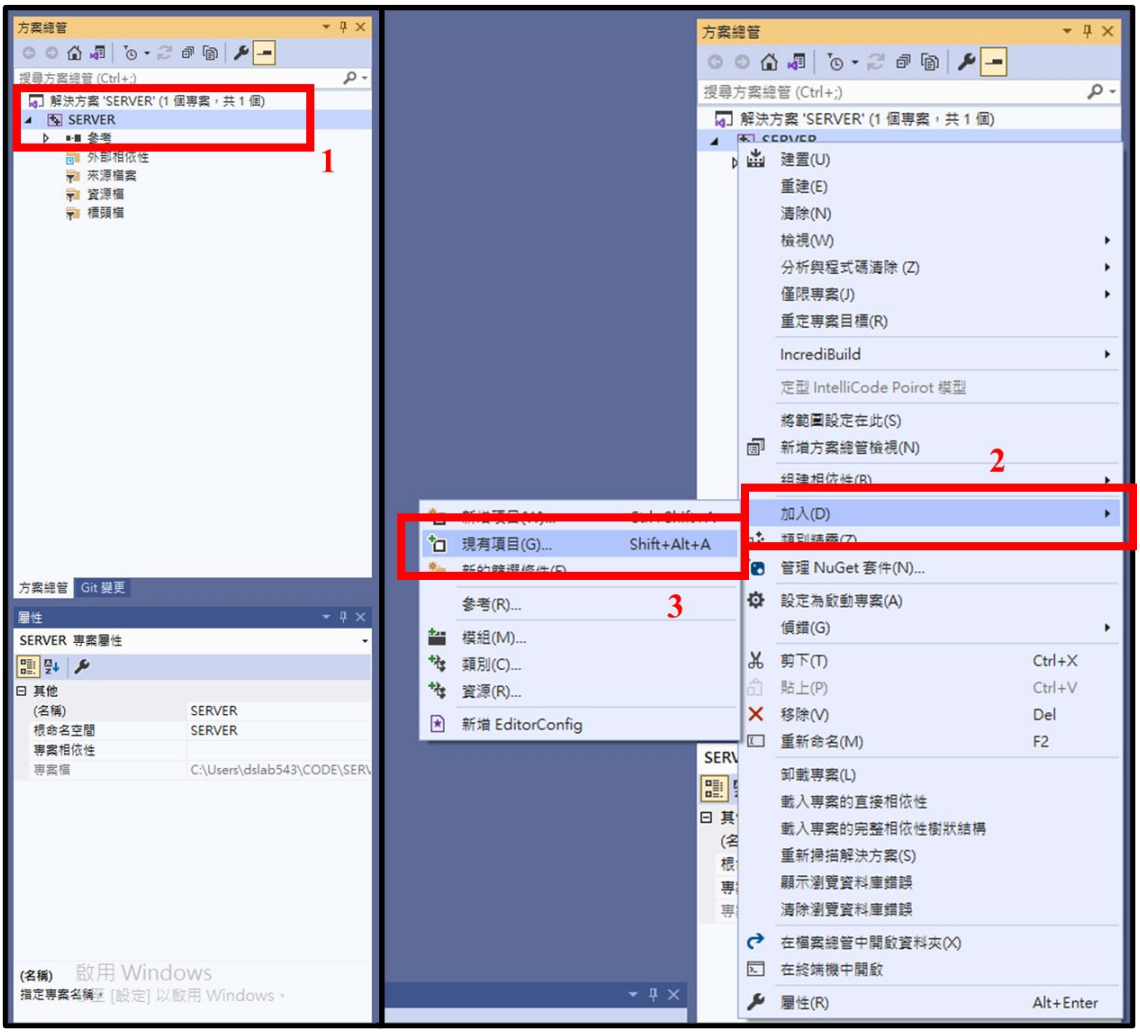


步驟 4.　將 server 端程式複製到剛創建專案 SERVER1

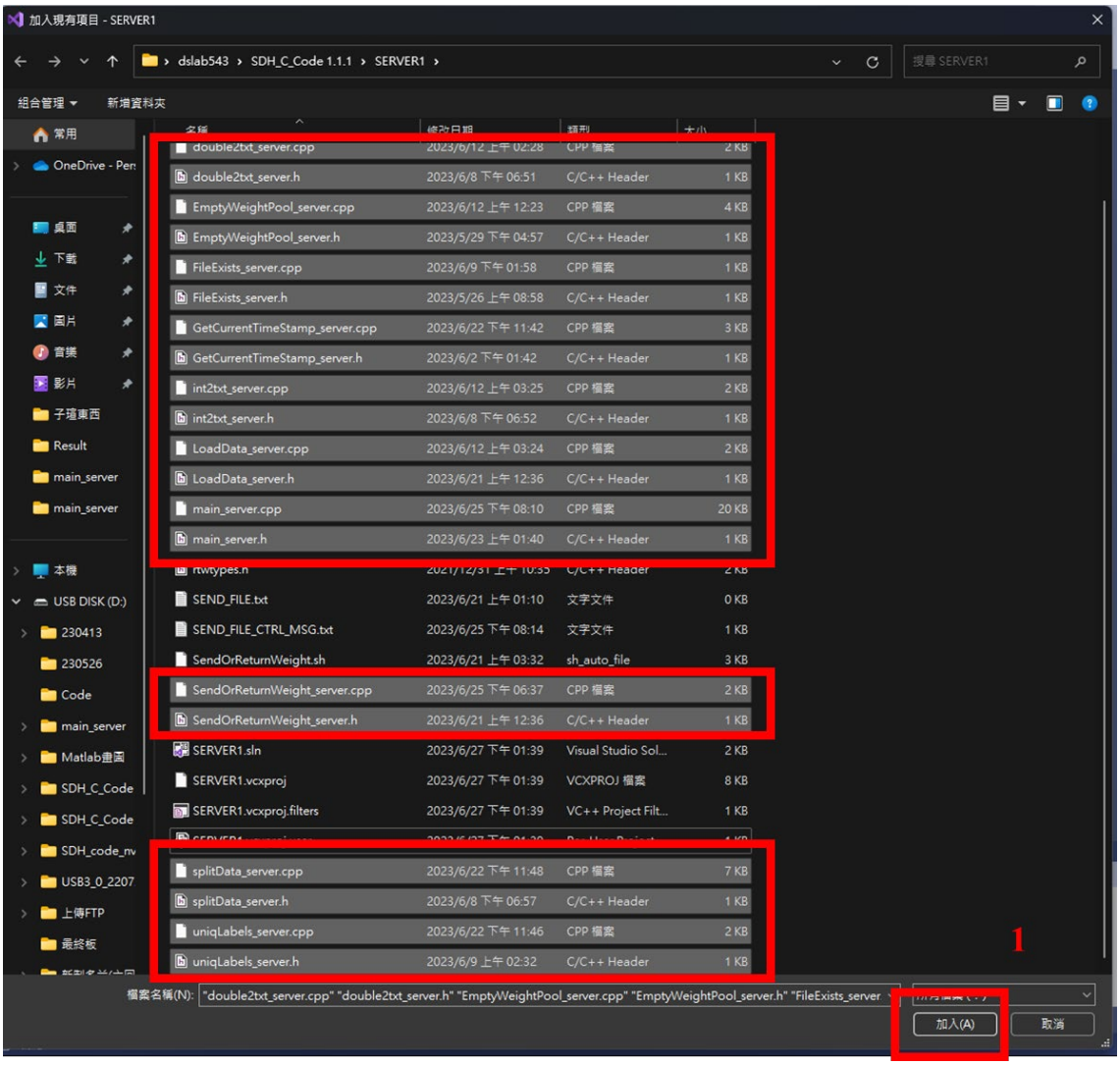
(C:\Users\dslab543\SDH_C_Code 1.1.1\SERVER1)底下



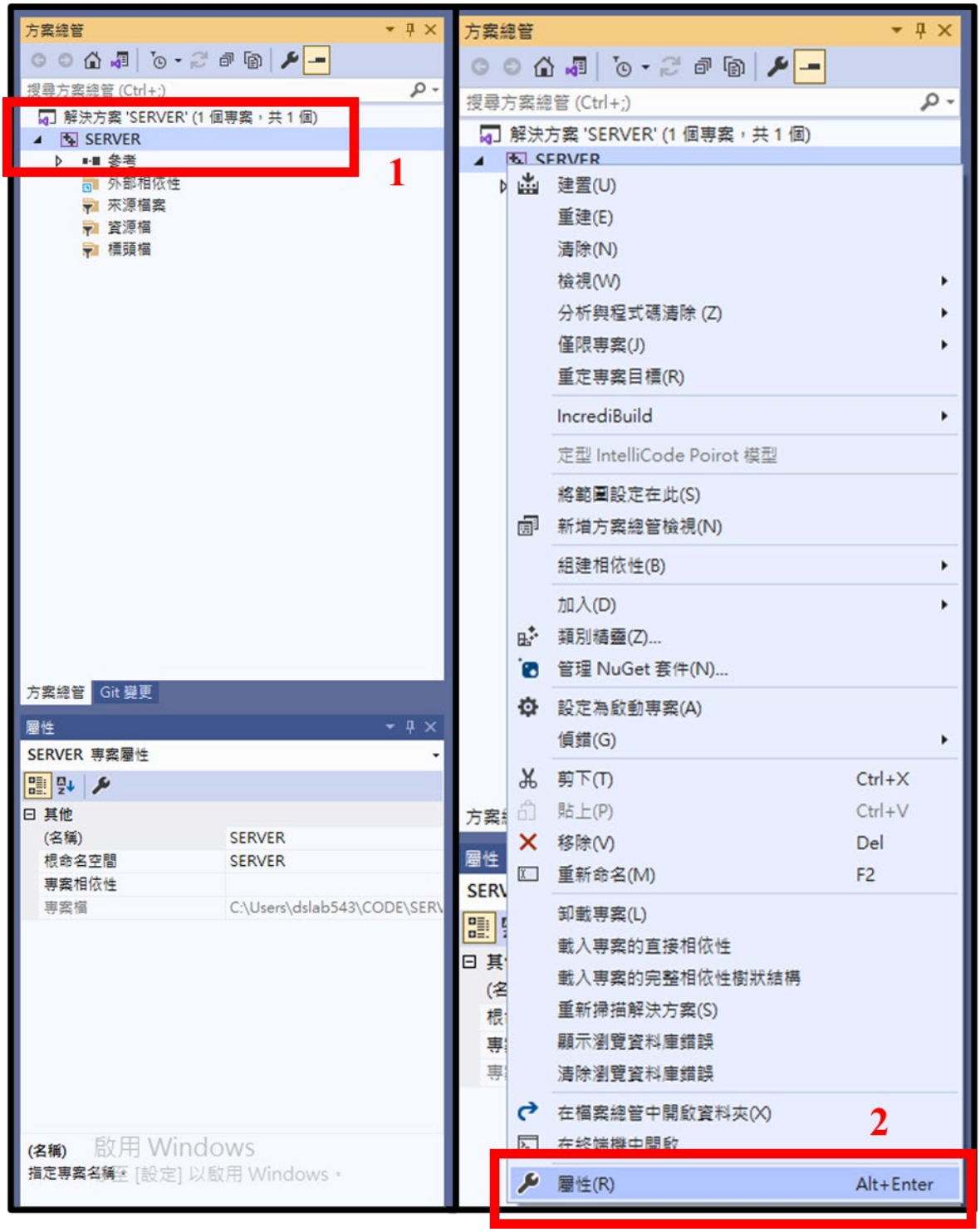
步驟 5.　加入 server 端程式加入專案中，讓專案讀取 ➢ 紅框 1：右鍵點擊旁邊專案檔 ➜紅框 2：加入 ➜紅框 3：現有項目，將 SERVER 端程式加入專案中，讓專案讀取



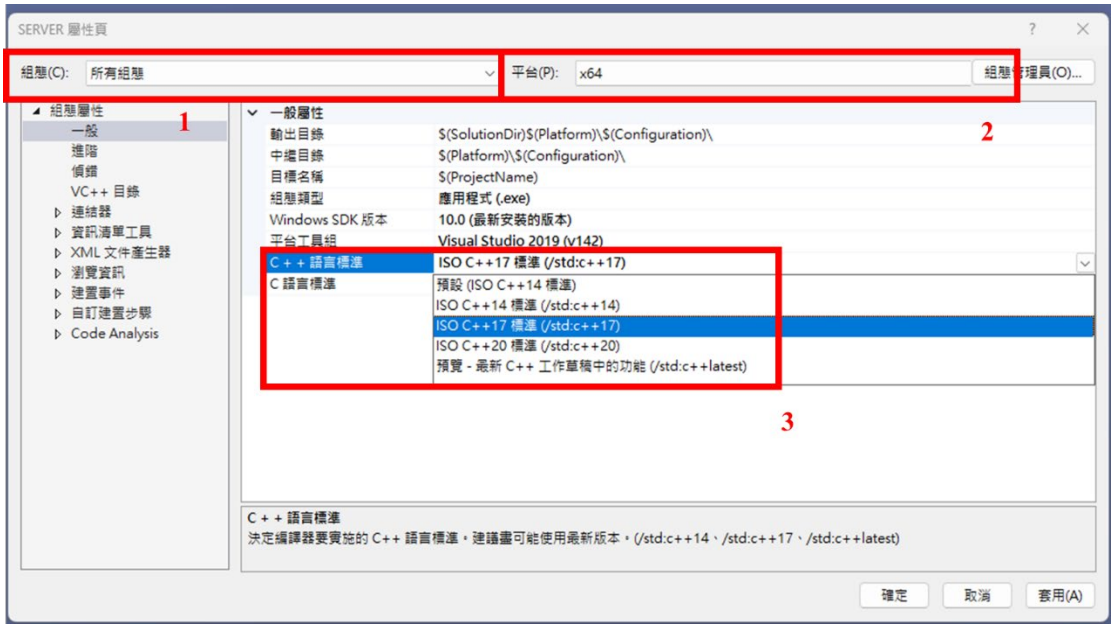
步驟 6.　加入 server 端程式至專案中，讓專案讀取➢ 只選取 CPP 檔案及 Header 檔案➜紅框 1：加入



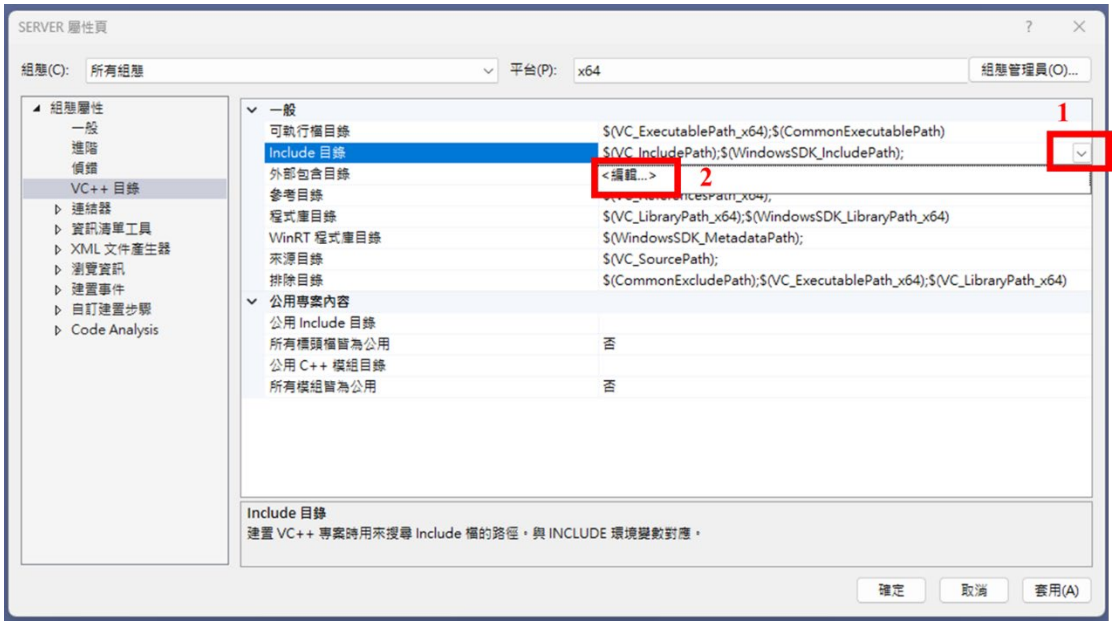
步驟 7.　將必需函式庫及套件包加入➢ 紅框 1：右鍵點擊旁邊專案檔 ➜ 紅框 2：屬性



步驟 8.　將必需函式庫及套件包加入➢ 紅框 1：將組態選為專案檔 ➜紅框 2：平台選為 x64 ➜紅框 3：一般 ➪ C++語言標準 ➪ ISO C++ 標準(/std c++17

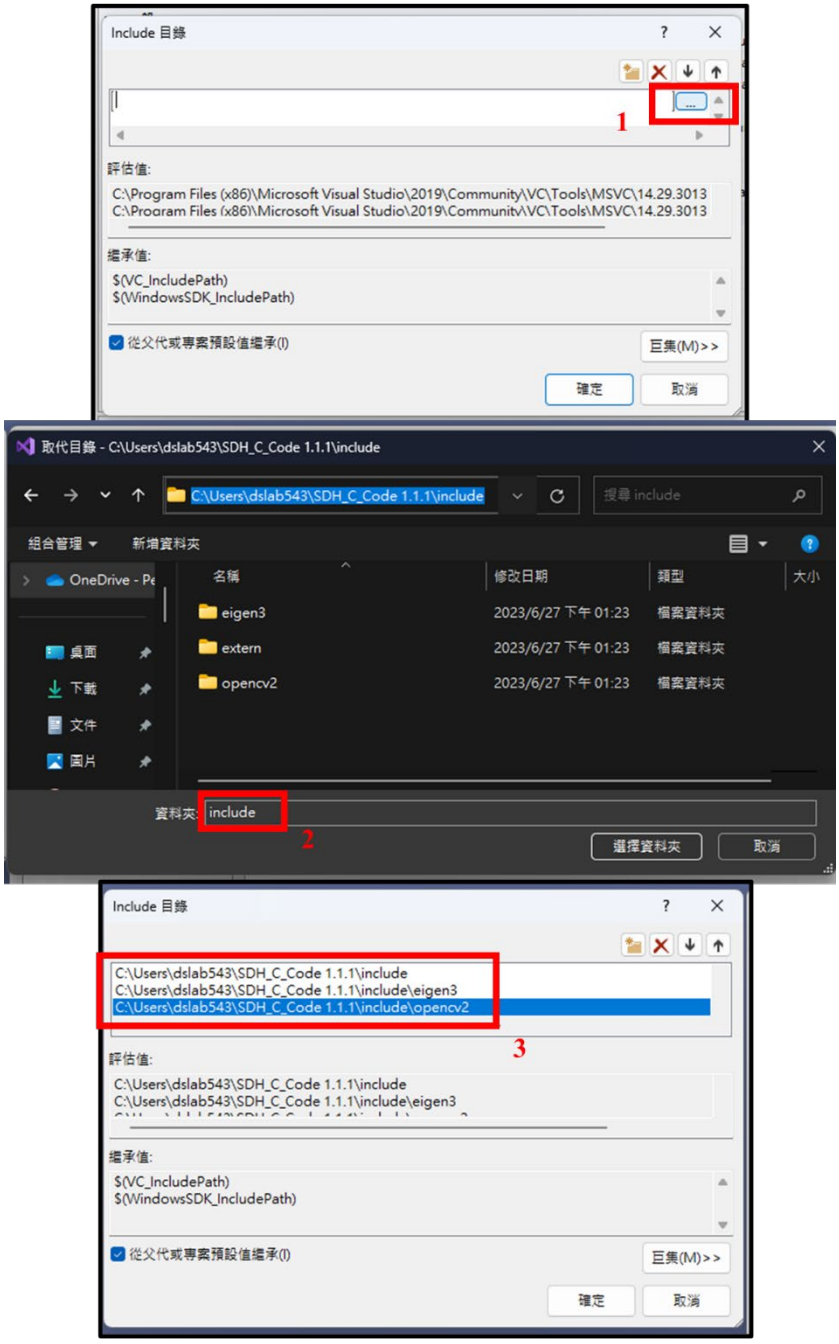


步驟 9.　將必需函式庫及套件包加入➢ 紅框 1：VC++ ➪ include 目錄 ，點擊下拉按鈕➜紅框 2：點擊編輯

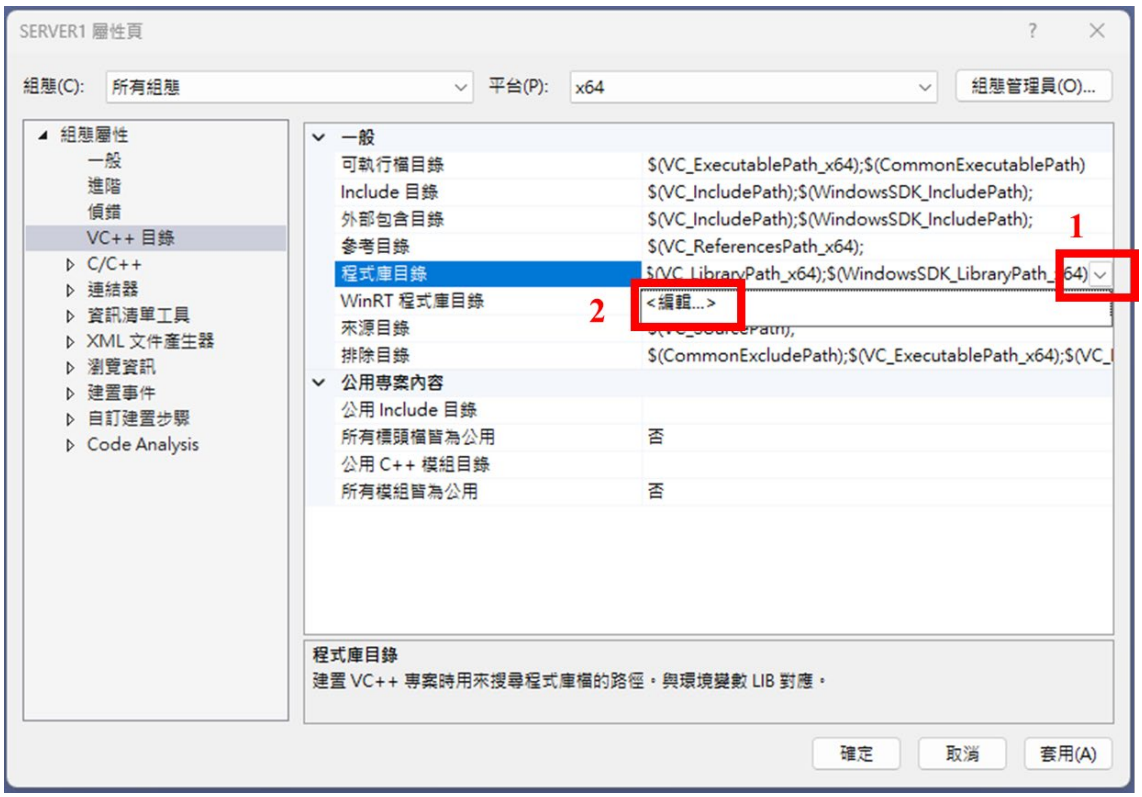


步驟 10. 將必需函式庫及套件包加入➢ 紅框 1：路徑按鈕 ➜紅框 2：選取下面指定的三個路徑➜紅框 3：指定完三個路徑的結果畫面➜點擊確認

```
# 指定的三個路徑

C:\程式包路徑\程式包名\include

C:\程式包路徑\程式包名\include\opencv2

C:\程式包路徑\程式包名\include\eigen3
```

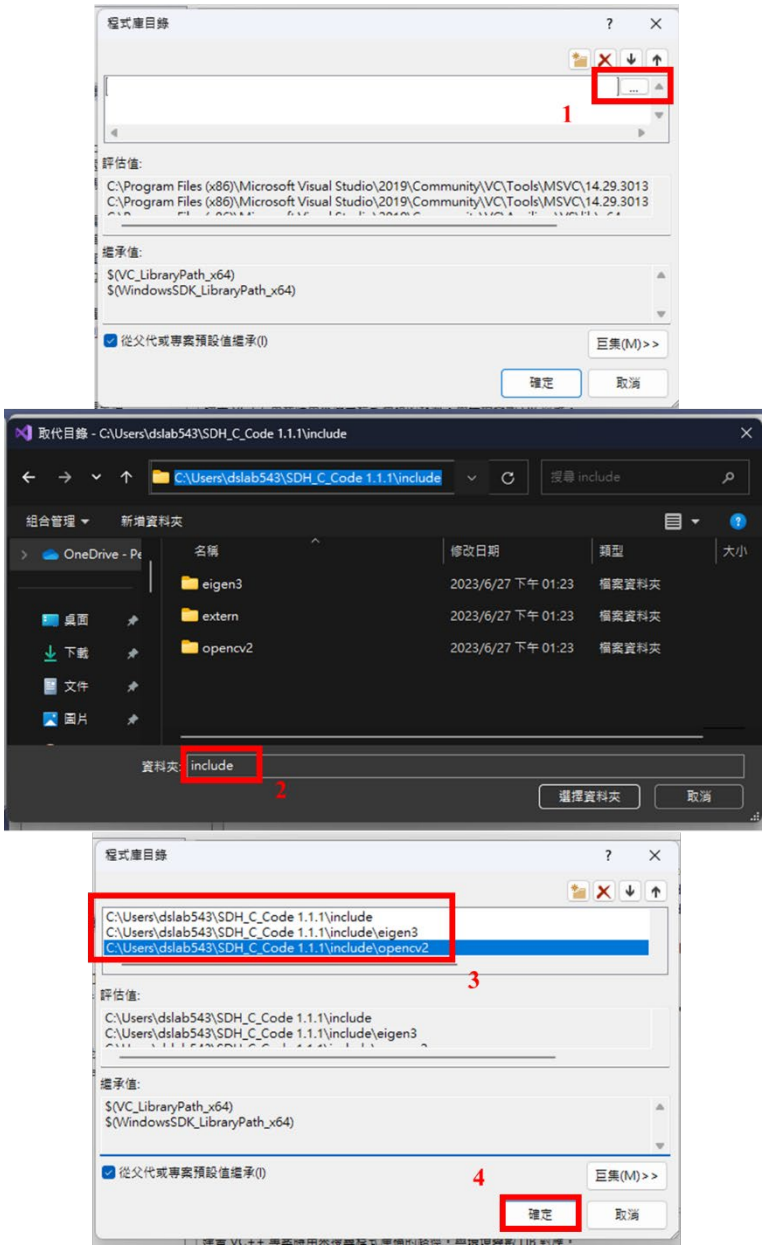


步驟 11. 將必需函式庫及套件包加入➢ 紅框 1： VC++ ➪ 程式庫目錄 ，點擊下拉按鈕➜紅框 2：點擊編輯

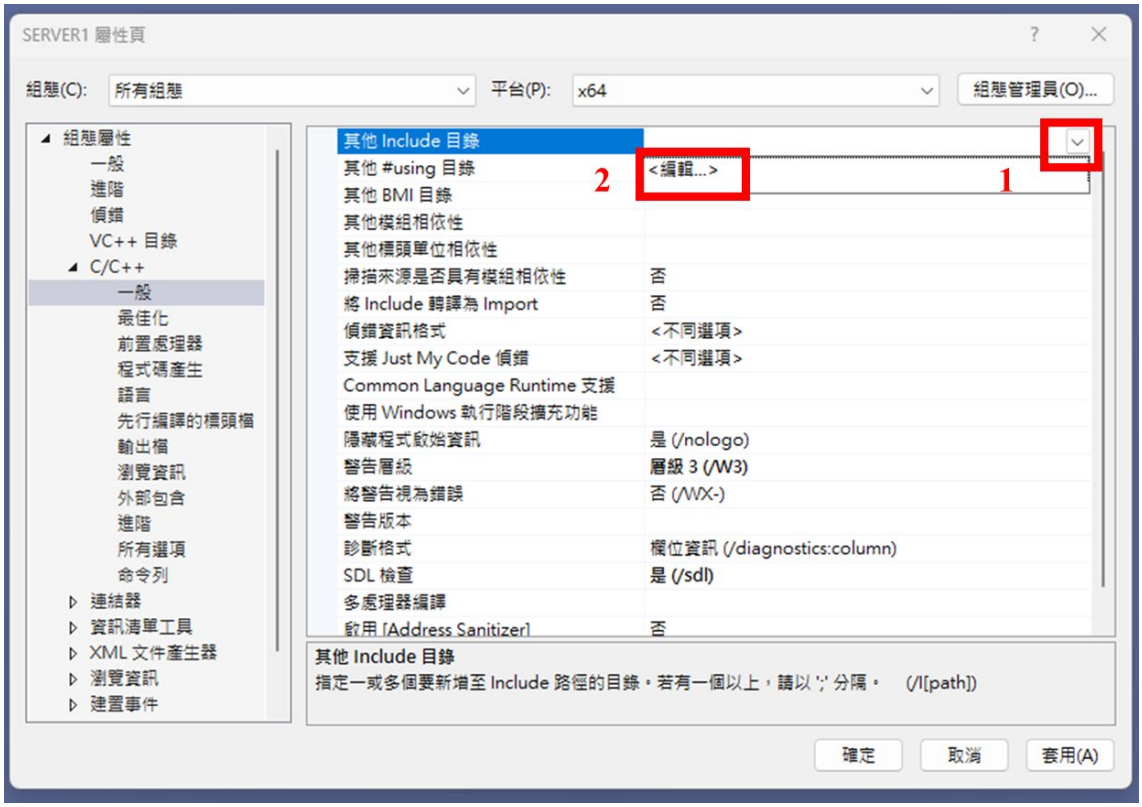


步驟 12. 重複步驟 10 的動作，將必需函式庫及套件包加入➢紅框 1：路徑按鈕➜紅框 2：選取下面指定的三個路徑➜紅框 3：指定完三個路徑的結果畫面➜點擊確定

```
# 指定的三個路徑

C:\程式包路徑\程式包名\include

C:\程式包路徑\程式包名\include\opencv2

C:\程式包路徑\程式包名\include\eigen3
```

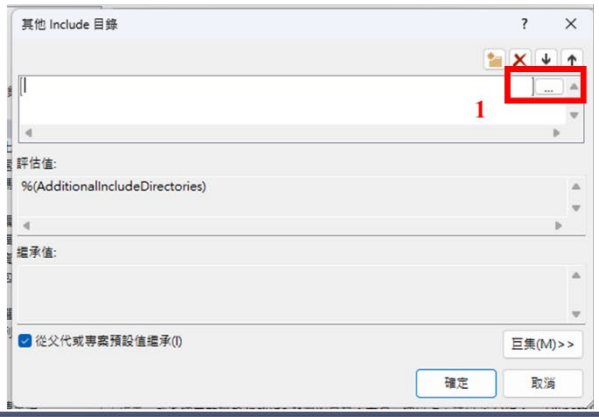


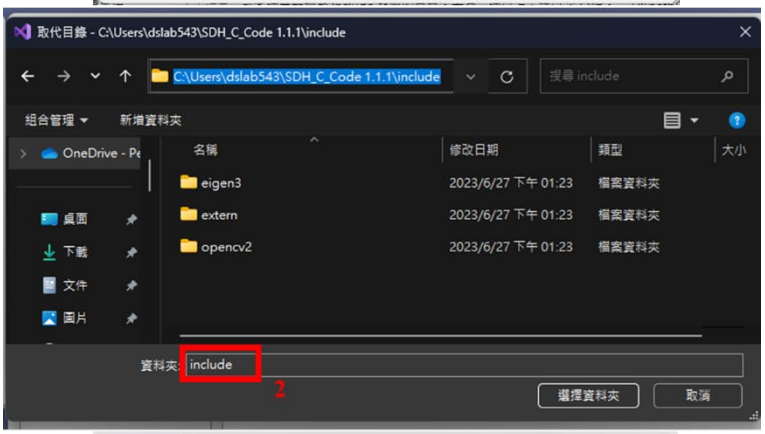
步驟 13. 將必需函式庫及套件包加入➢ 紅框 1： C++ ➪ 一般 ➪ 其他 include 目錄，點擊下拉按鈕➜紅框 2：點擊編輯

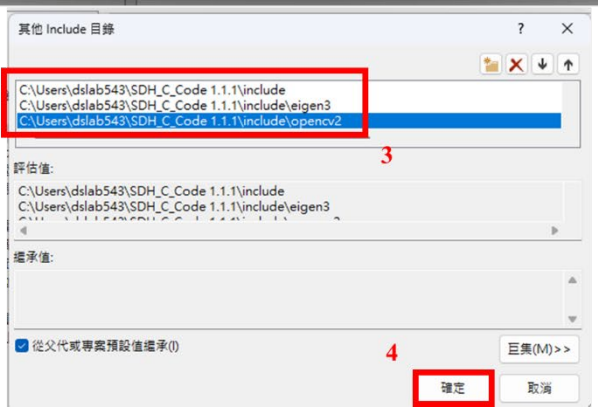


步驟 14. 重複步驟 10 的動作，將必需函式庫及套件包加入➢紅框 1：路徑按鈕➜紅框 2：選取下面指定的三個路徑➜紅框 3：指定完三個路徑的結果畫面➜點擊確定

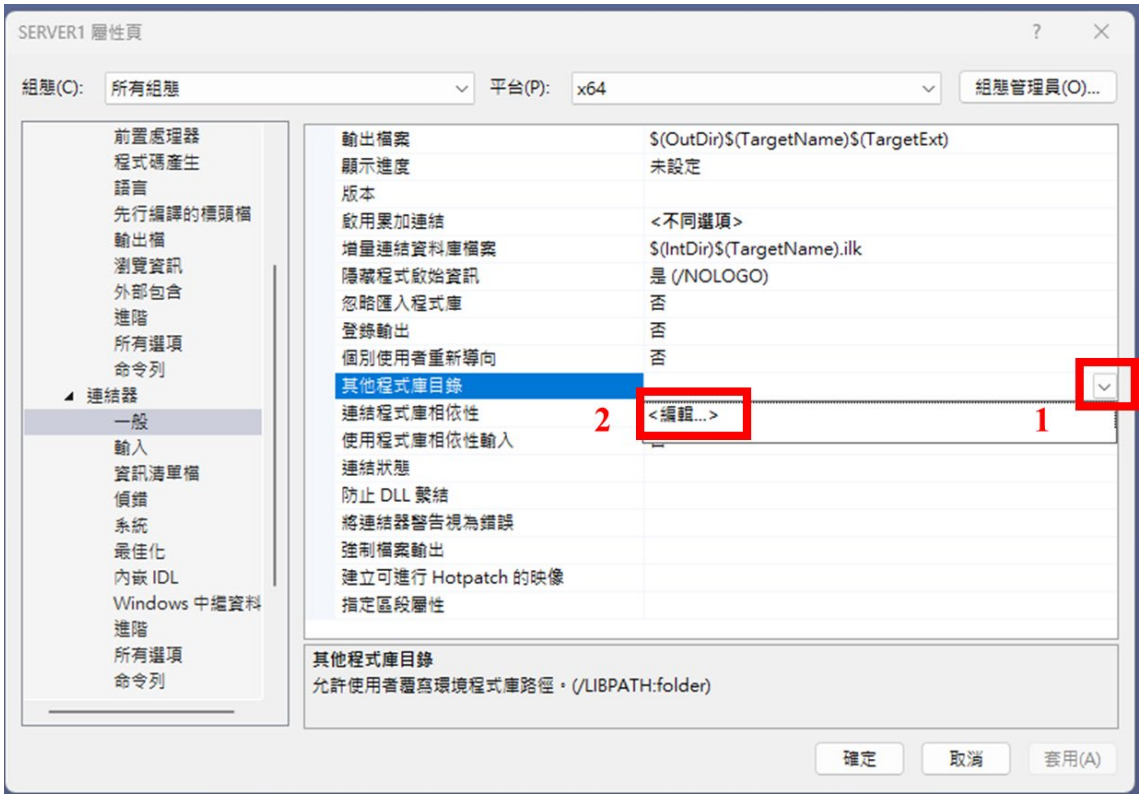
```
# 指定的三個路徑
C:\程式包路徑\程式包名\include
C:\程式包路徑\程式包名\include\opencv2
C:\程式包路徑\程式包名\include\eigen3
```





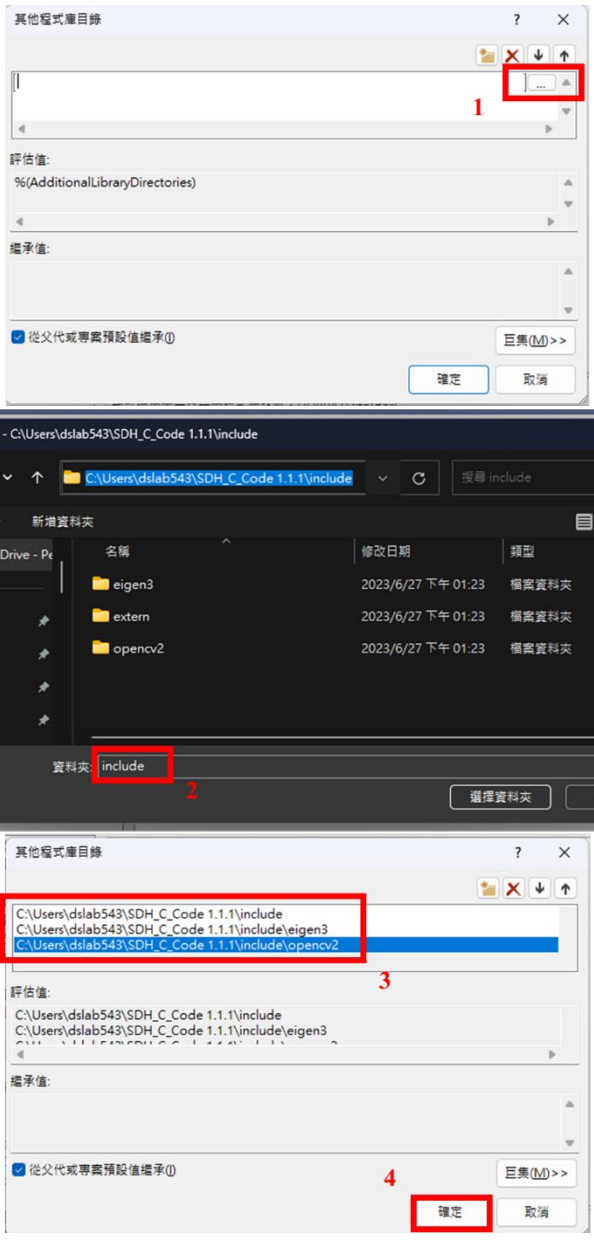


步驟 15. 將必需函式庫及套件包加入➢ 紅框 1： 連結器➪一般➪其他程式庫目錄，點擊下拉按鈕➜紅框 2：點擊編輯

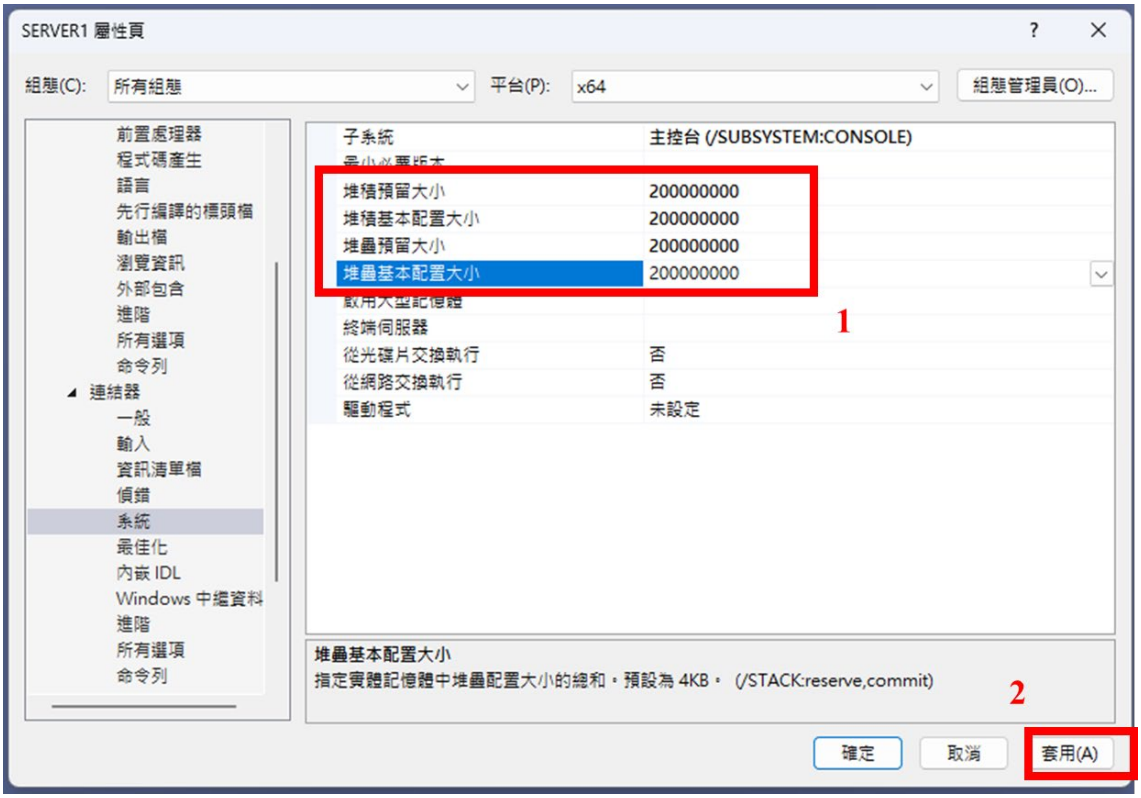


步驟 16. 重複步驟 10 的動作，將必需函式庫及套件包加入➢紅框 1：路徑按鈕➜紅框 2：選取下面指定的三個路徑➜紅框 3：指定完三個路徑的結果畫面➜點擊確定

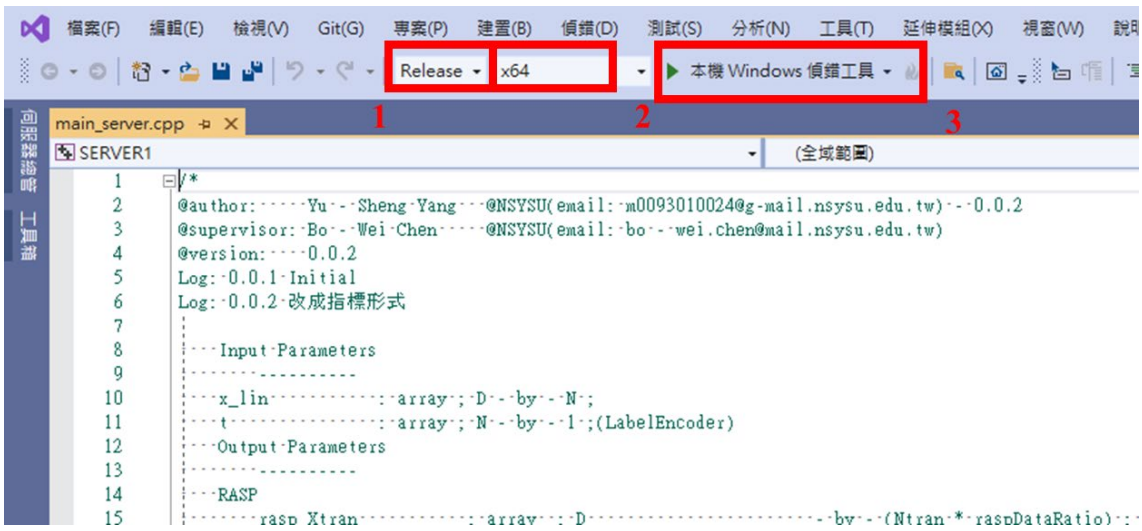
```
# 指定的三個路徑

C:\程式包路徑\程式包名\include

C:\程式包路徑\程式包名\include\opencv2

C:\程式包路徑\程式包名\include\eigen3
```



步驟 17. 將必需函式庫及套件包加入➢ 紅框 1： 堆積預留大小、堆積基本配置大小、堆積預留大小和堆疊基本配置大小 ，這四個欄位改為 200000000 ➜ 紅框 2：套用



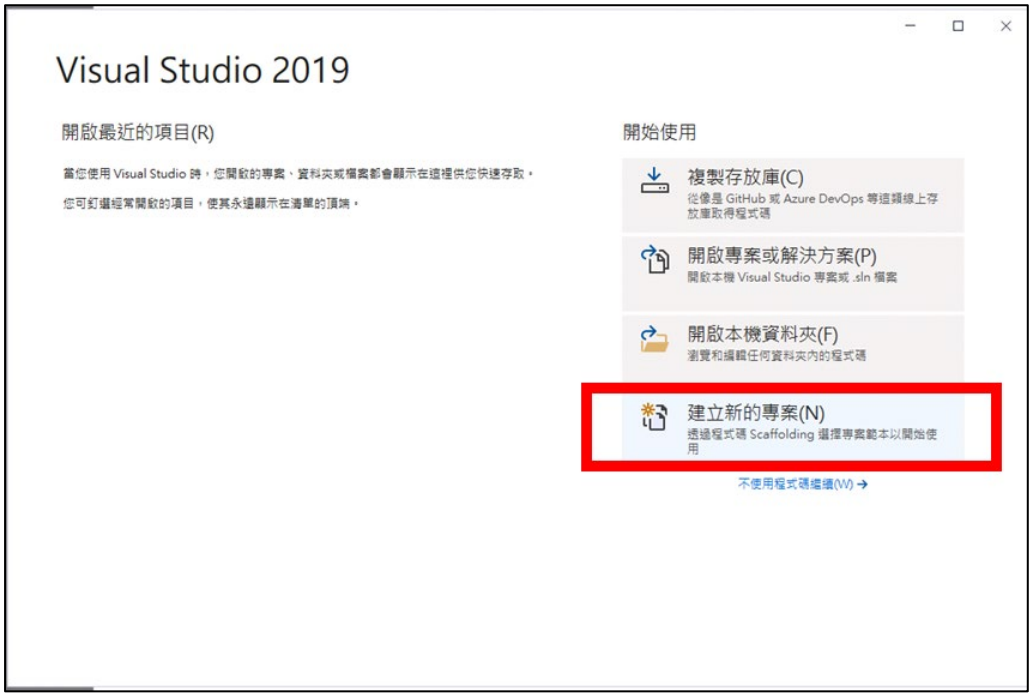
步驟 18. 轉換組態和平台➢ 紅框 1：組態選取 Release ➜紅框 2：平台選取 x64 ➜紅框 3：進行程式編譯點擊 本機 Windows 偵錯工具



1.6 設定 Client 端上的程式

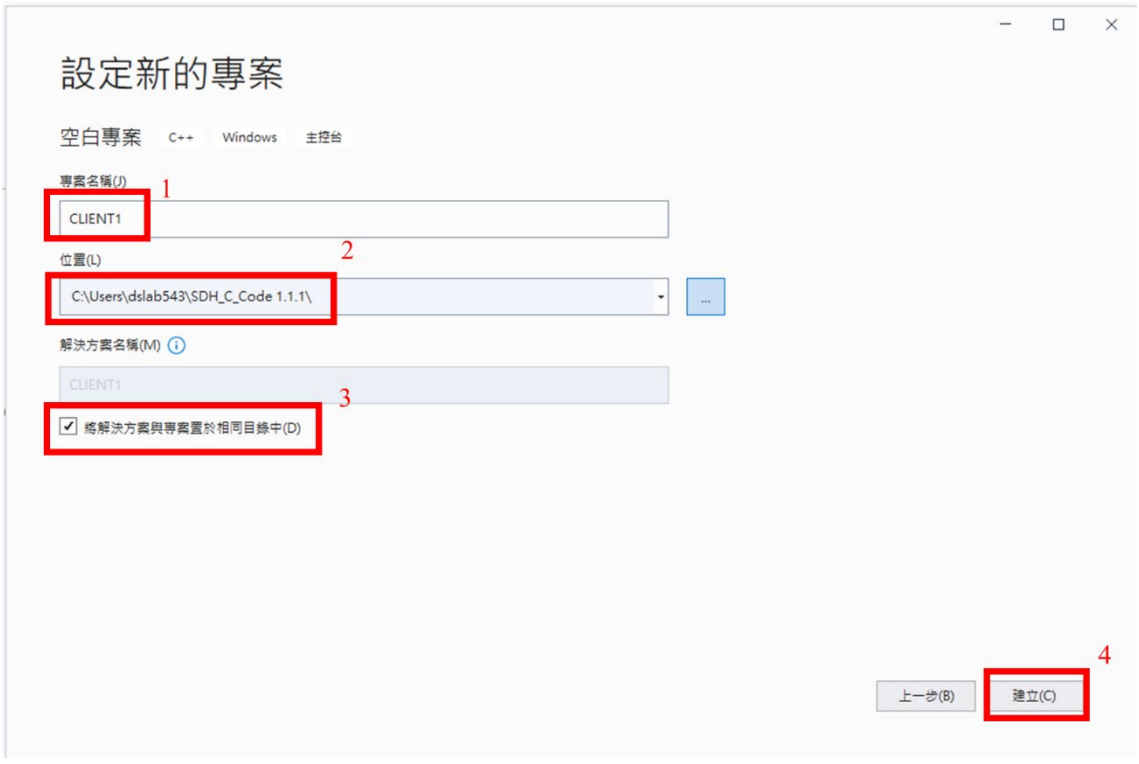
- 步驟大多與 **Server** 端一致，但須建立兩個 **Client** 端以完成聯邦式雜湊模型的訓練
- 設定第一個 **Client** 端

步驟 1. 建立新的專案(**CLIENT1**)



步驟 2.　空白專案 ➜ 下一步

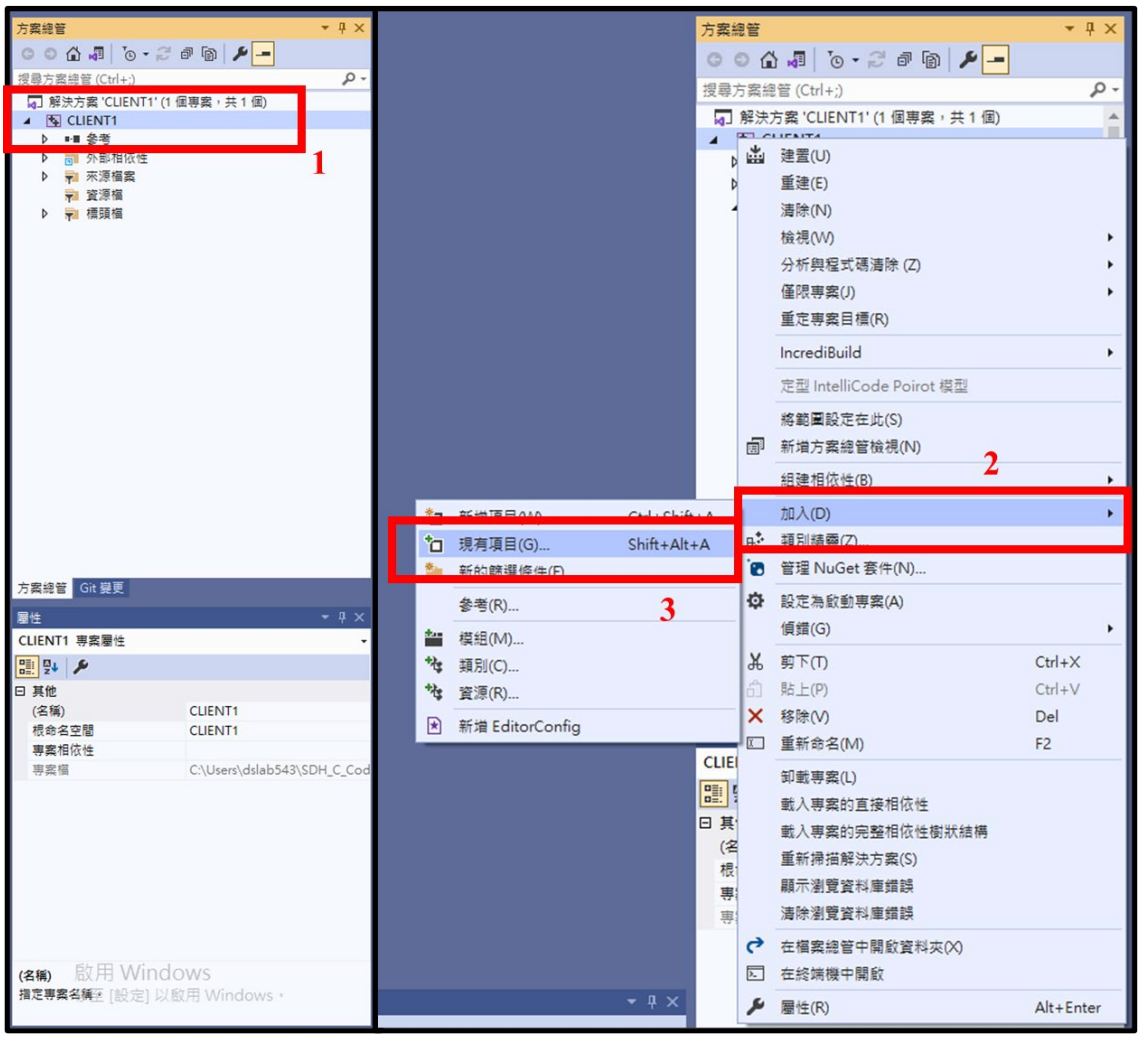
步驟 3.　設定專案路徑 ➢ 紅框 1：設定專案名稱 **CLIENT1** ➜紅框 2：設定專案路徑至程式包下一層目錄 ➜紅框 3：勾選解決方案與專案置於相同目錄中 ➜紅框 4：建立



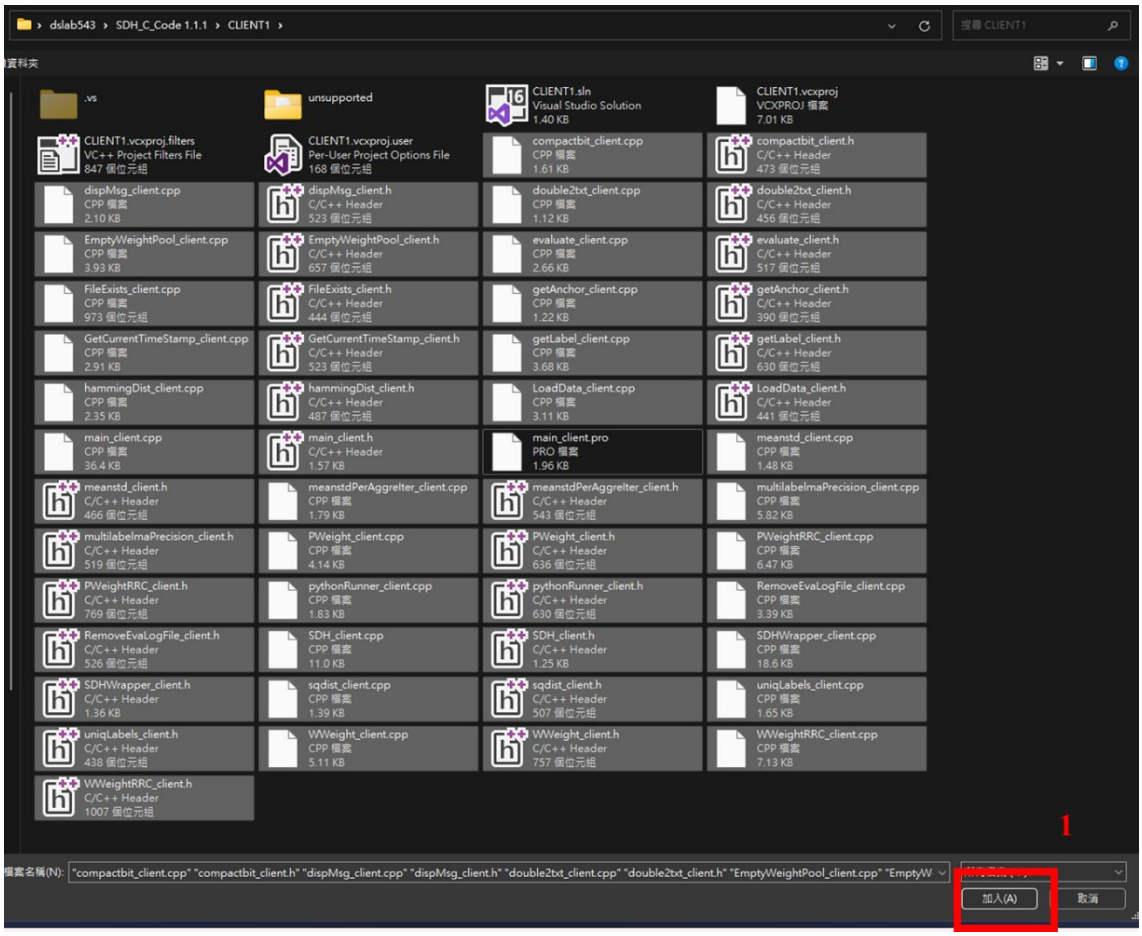
步驟 4.　將 client 端程式複製到剛創建專案 **CLIENT1**

(C:\Users\dslab543\SDH_C_Code 1.1.1\ CLIENT1)底下

步驟 5.　加入 server 端程式加入專案 **CLIENT1** 中，讓專案讀取 ➢ 紅框 1：右鍵點擊旁邊專案檔 ➜紅框 2：加入 ➜紅框 3：現有項目，將 client 端程式加入專案中，讓專案讀取



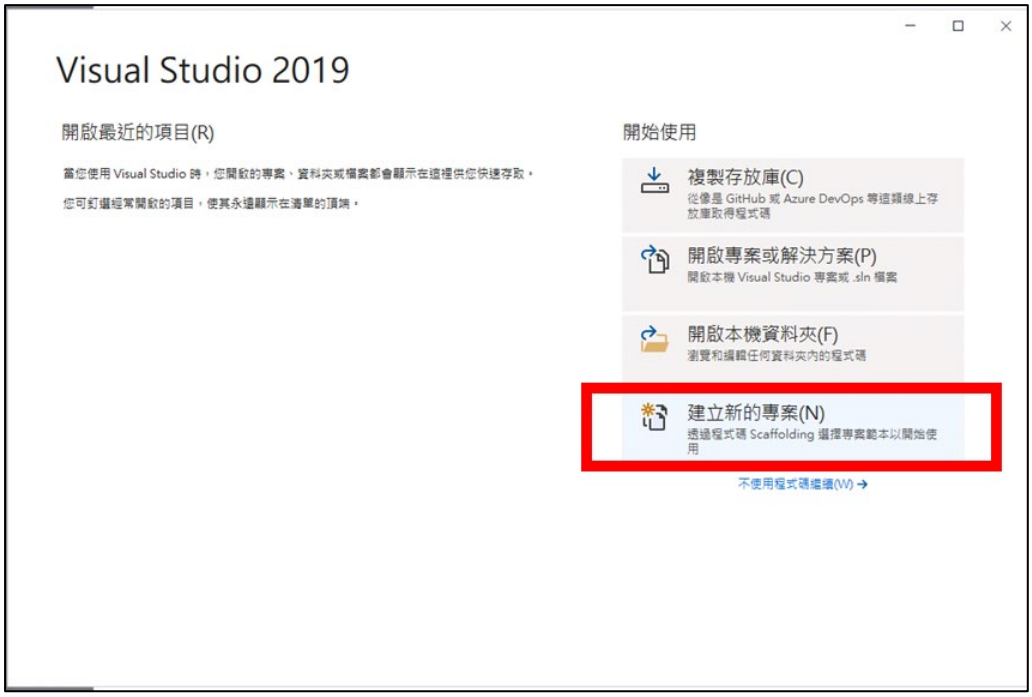
步驟 6.　加入 client 端程式至專案 **CLIENT1** 中，讓專案讀取➢只選取 CPP 檔案及 Header 檔案➜紅框 1：加入



重複設定 server 端程式的**步驟 7 ～ 步驟 18**

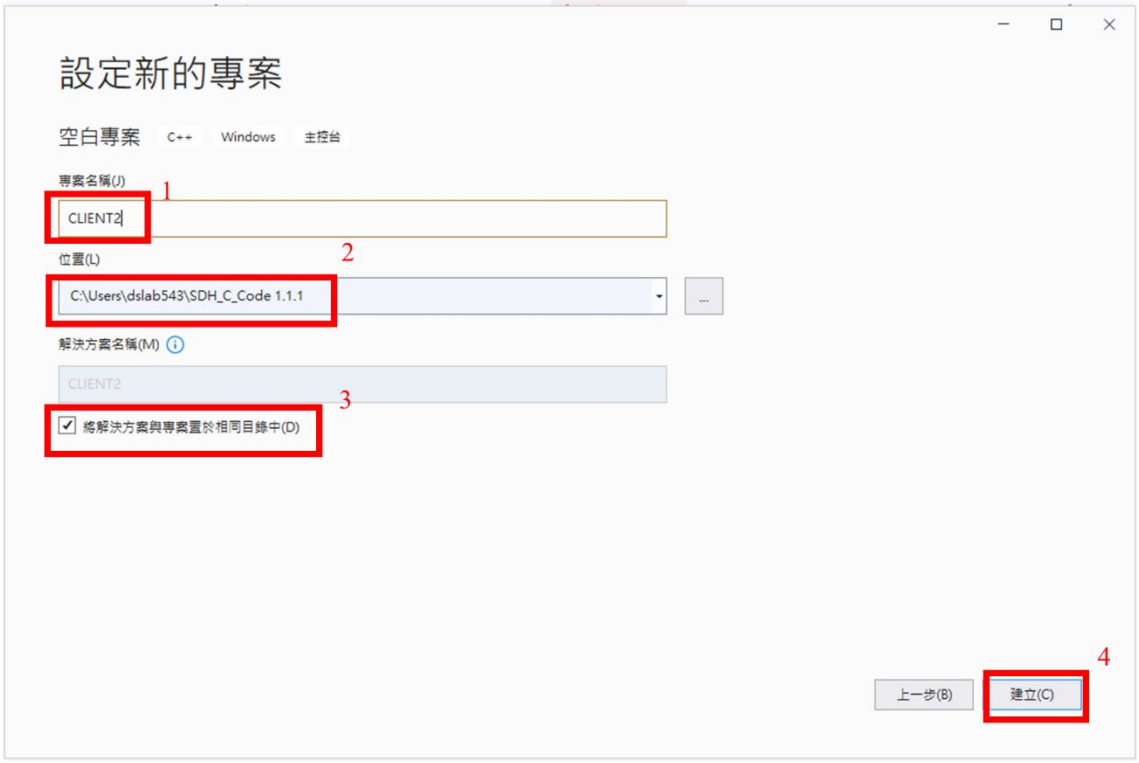
- 設定第二個 **Client** 端(重複第一個 **Client** 端做的操作)

步驟 1.　建立新的專案(**CLIENT2**)



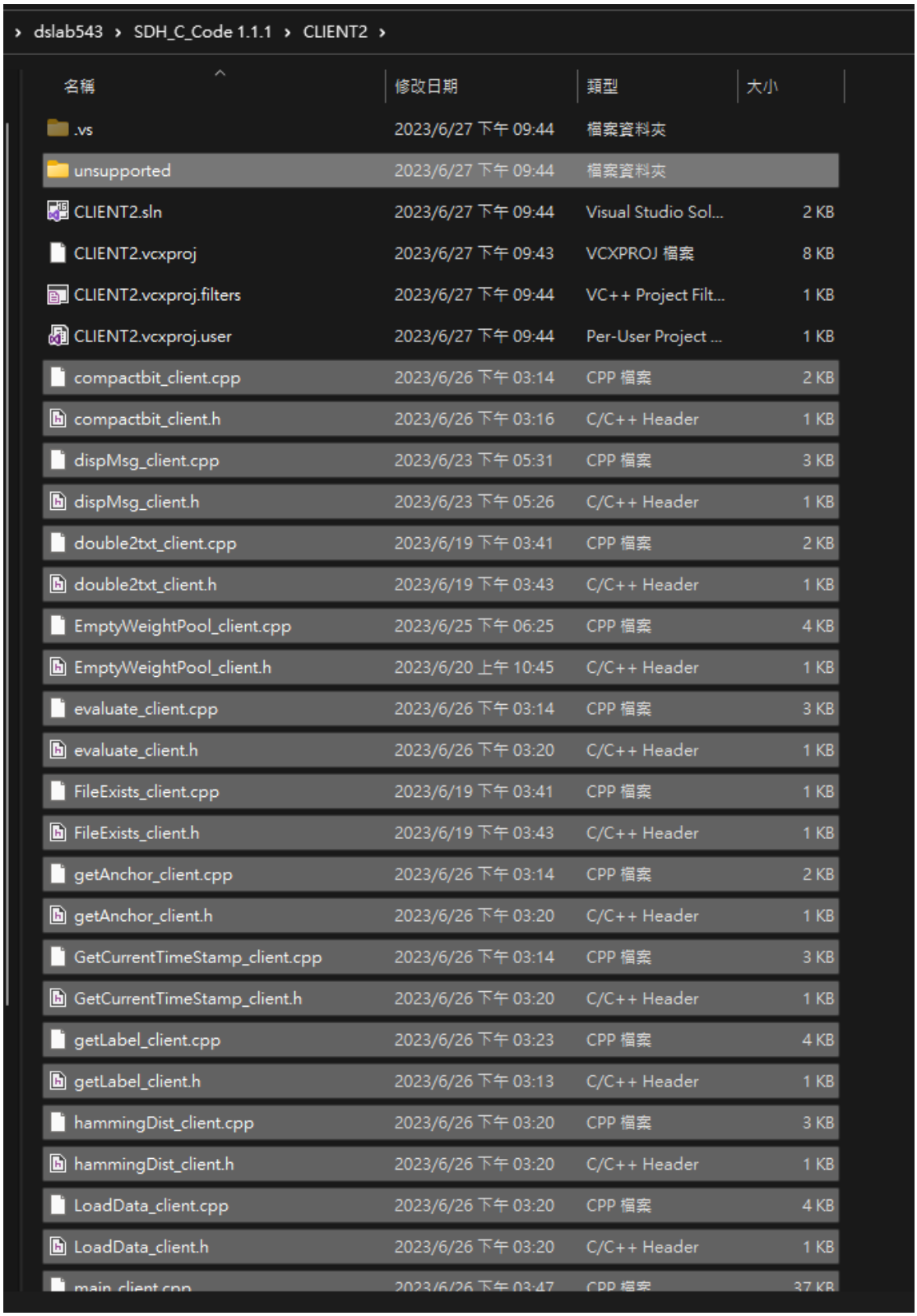
步驟 2.　空白專案 ➜ 下一步

步驟 3.　設定專案路徑 ➢ 紅框 1：設定專案名稱 **CLIENT2** ➜紅框 2：設定專案路徑至程式包的下一層目錄 ➜紅框 3：勾選解決方案與專案置於相同目錄中 ➜紅框 4：建立

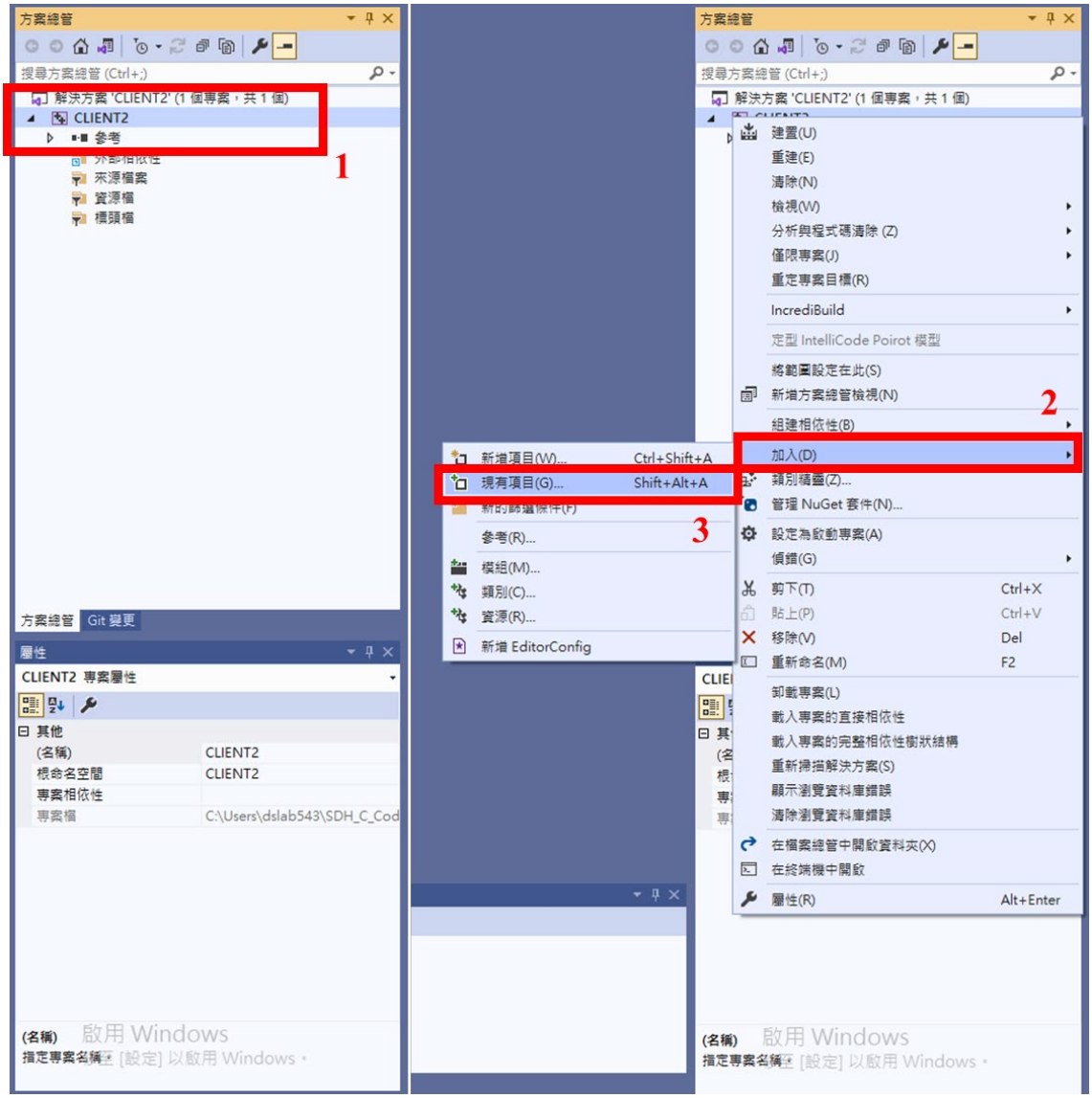


步驟 4.　將 client 端程式複製到剛創建專案 **CLIENT2**

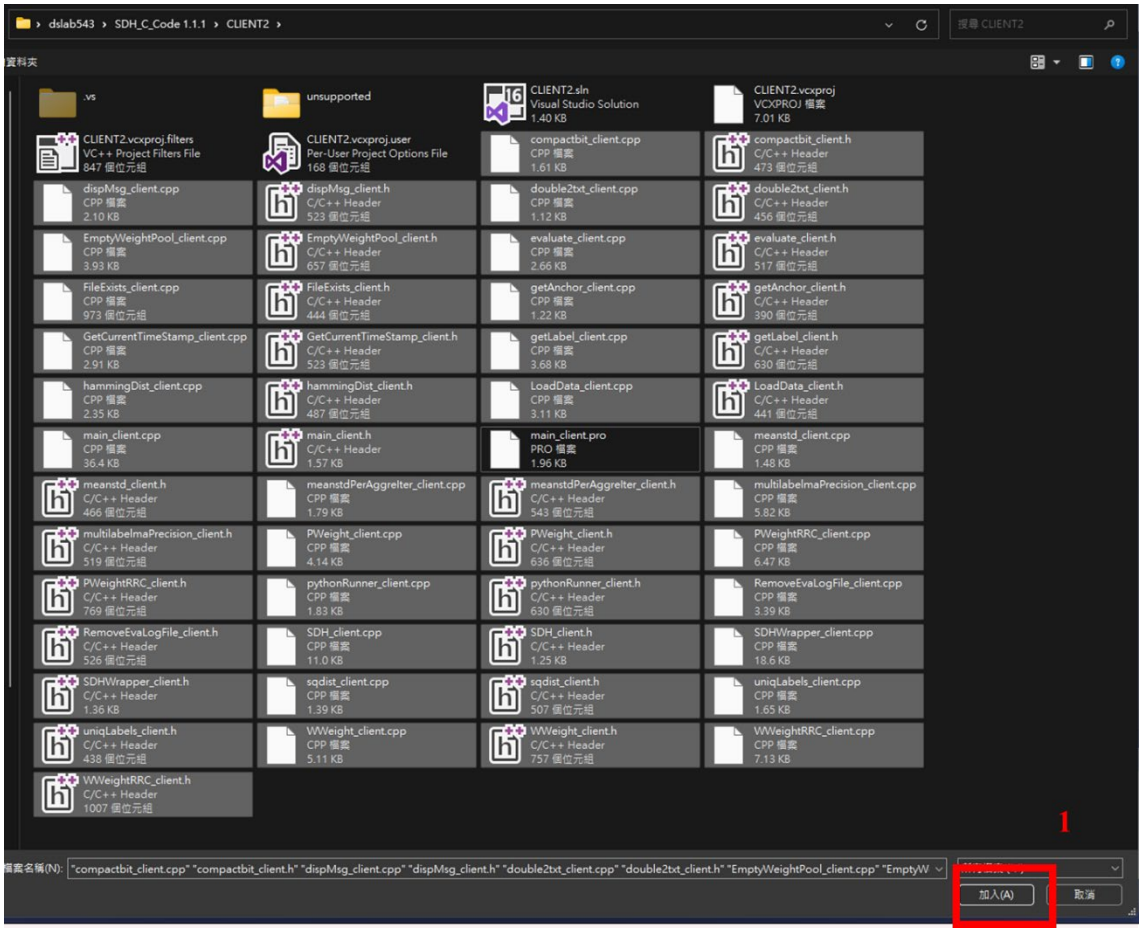
(C:\Users\dslab543\SDH_C_Code 1.1.1\ CLIENT2)底下



步驟 5.　加入 client 端程式至專案 **CLIENT2** 中，讓專案讀取 ➢ 紅框 1：右鍵點擊旁邊專案檔 ➜紅框 2：加入 ➜紅框 3：現有項目，將 client 端程式加入專案中，讓專案讀取

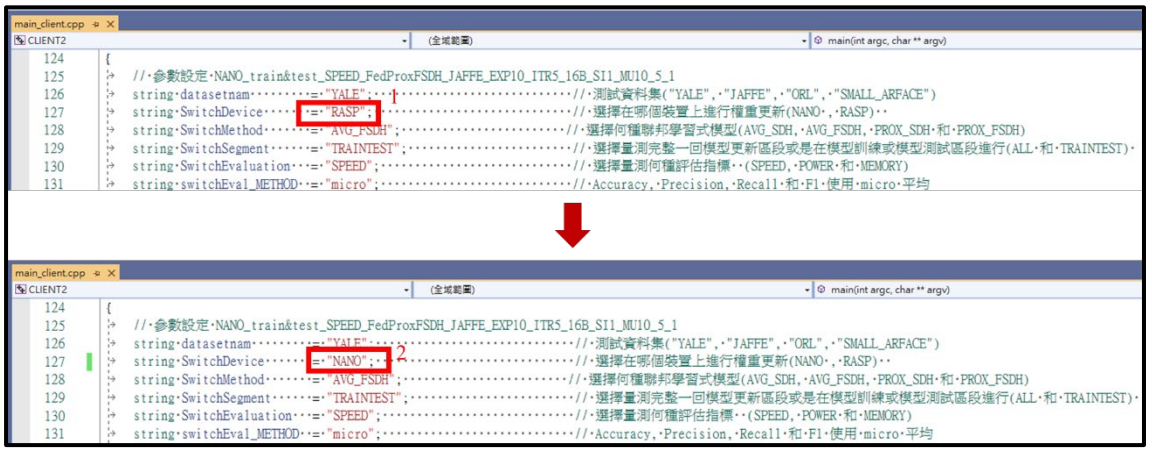


步驟 6.　加入 client 端程式至專案 **CLIENT2** 中，讓專案讀取➢只選取 CPP 檔案及 Header 檔案➜紅框 1：加入

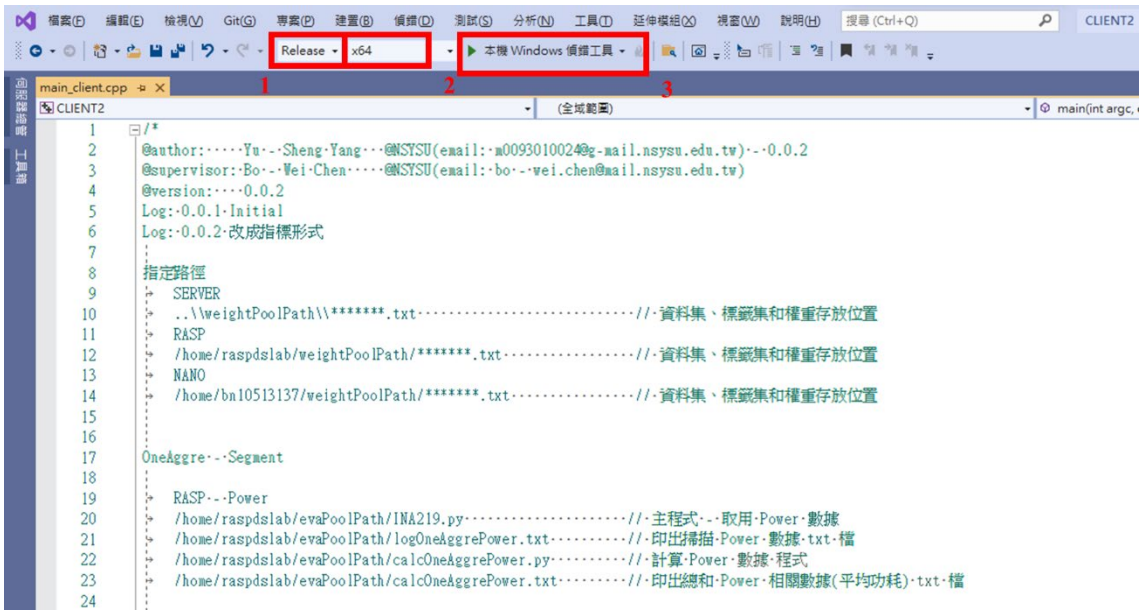


重複設定 server 端程式的**步驟 7 ～ 步驟 17**

步驟 18. 需在專案 CLIENT2 中的主程式 main_client.cpp ➢ 將紅框 1 的 RASP 值改為紅框 2 的 NANO



步驟 19. 轉換組態和平台 ➢ 紅框 1：組態選取 Release ➜紅框 2：平台選取 x64 ➜紅框 3：進行程式編譯點擊 本機 Windows 偵錯工具



1.7　執行聯邦式雜湊學習模型

- 進行最終檢查

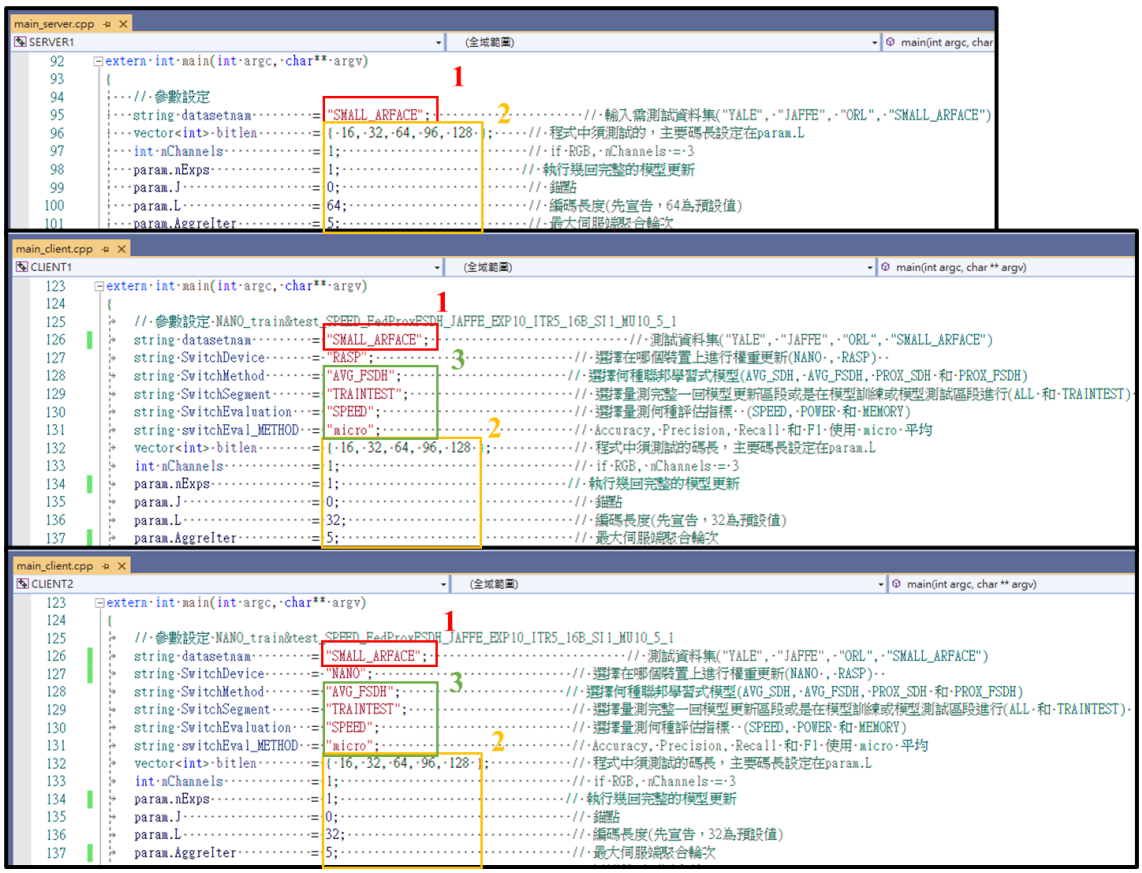
步驟 1.　檢查專案 SERVER1、CLIENT1 和 CLIENT2 主程式裡的參數是否一致

➢ 紅框 1：檢查三個專案中 datasetnam(數據集)變數之值是否一致

➢ 橘框 2：檢查三個專案中 nChannel、param.nExps、param.J、param.L 和 param.AggreIter 變數之值是否一致

➢ 綠框 3：檢查專案 CLIENT1 和 CLIENT2 SwitchMethod、SwitchSegment 和 SwitchEvaluation 變數之值是否一致

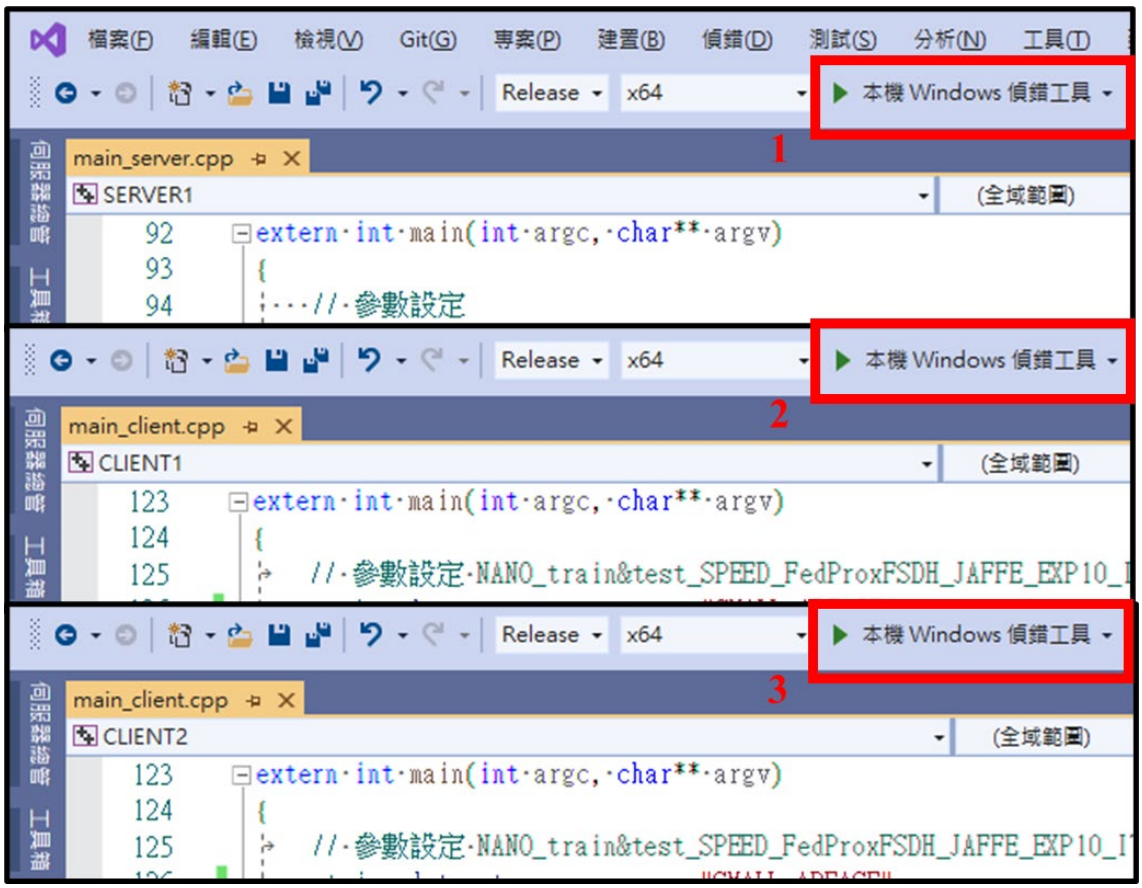
進行程式編譯點擊 本機 Windows 偵錯工具



步驟 2.　檢查專案 SERVER1 裡的資料集路徑是否正確

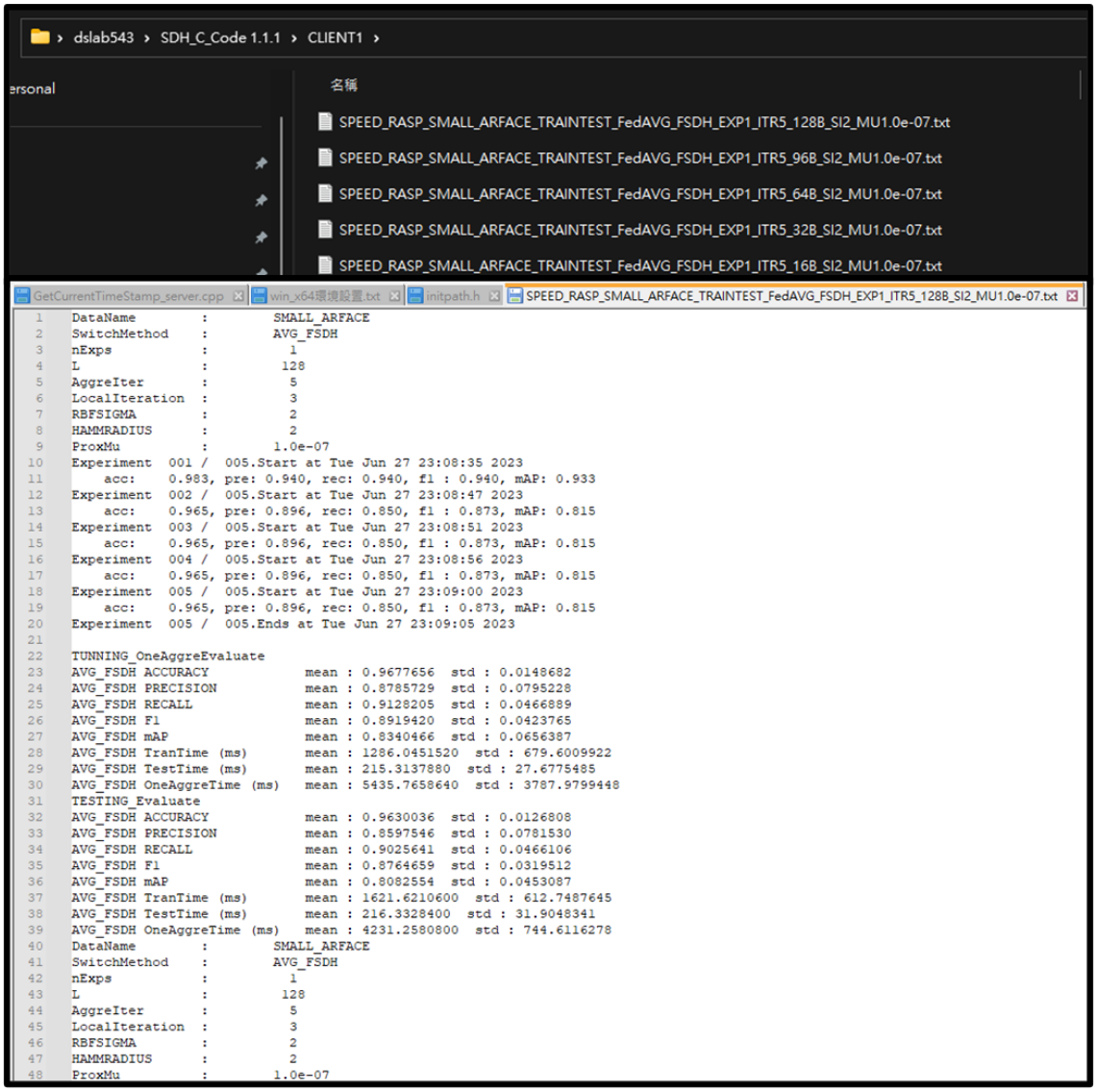
```
150
151  // 取用資料集(請看成是第 30 行)
152  // YALE
153  if (datasetnam.compare("YALE") == 0)
154  {
155      infilepath_x_lin_data = "../Dataset/yale64x64_data_x_lin.txt";
156      infilepath_t_data     = "../Dataset/yale64x64_data_t.txt";
157  }
158  // JAFFE
159  else if (datasetnam.compare("JAFFE") == 0)
160  {
161      infilepath_x_lin_data = "..//Dataset/jaffe64x64_data_x_lin.txt";
162      infilepath_t_data     = "../Dataset/jaffe64x64_data_t.txt";
163  }
164  // ORL
165  else if (datasetnam.compare("ORL") == 0)
166  {
167      infilepath_x_lin_data = "../Dataset/orl64x64_data_x_lin.txt";
168      infilepath_t_data     = "../Dataset/orl64x64_data_t.txt";
169  }
170  // SMALL_ARFACE
171  else if (datasetnam.compare("SMALL_ARFACE") == 0)
172  {
173      infilepath_x_lin_data = "../Dataset/SMALL_2340_ARface64x64_data_x_lin.txt";
174      infilepath_t_data     = "../Dataset/SMALL_2340_ARface64x64_data_t.txt";
175  }
```

步驟 3.　依序點擊專案 SERVER1、CLIENT1 和 CLIENT2 本機 Windows 偵錯工具 (第一個一定要點 server 端不然會出錯)



步驟 4.　程式執行結果如下

步驟 5.　在專案 CLIENT1 和 CLIENT2 的資料夾中會生成各碼長生成的執行結果，但只會有模型好壞評估指標及速度的測試結果。

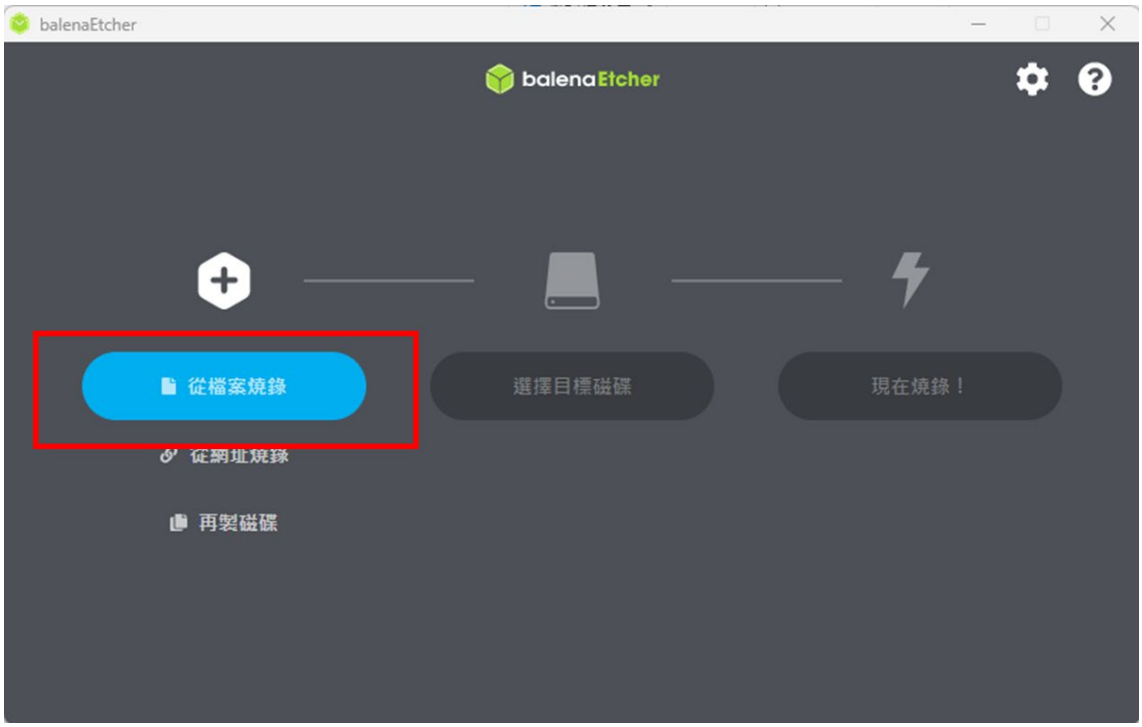


嵌入式版安裝步驟

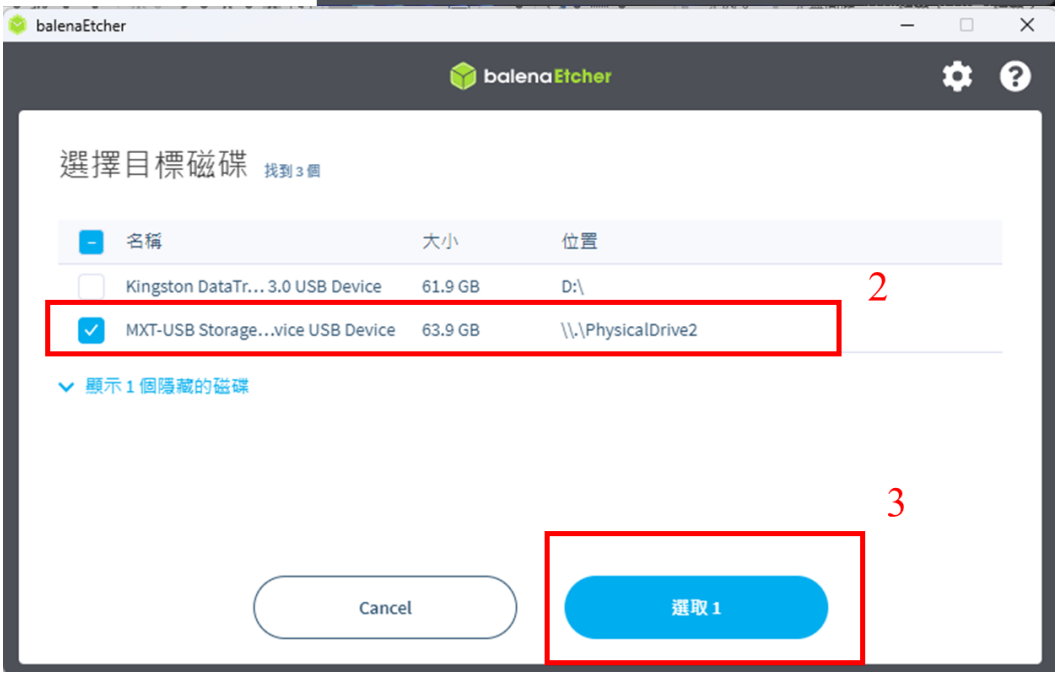
2 進行 Raspberry Pi4 及 Jetson Nano 的環境設定

2.1 燒錄 Raspberry Pi4 及 Jetson Nano 的映像檔

- 準備兩張乾淨的 SD 卡(至少為 64GB)
- 在 Windows 上安裝燒錄程式 **balenaEtcher**
- 到以下網址進行下載 **https://etcher.balena.io/#download-etcher**
- 打開程式後，點選**從檔案燒錄**選項選擇 Raspberry Pi4 或 Jetson Nano 的映像檔

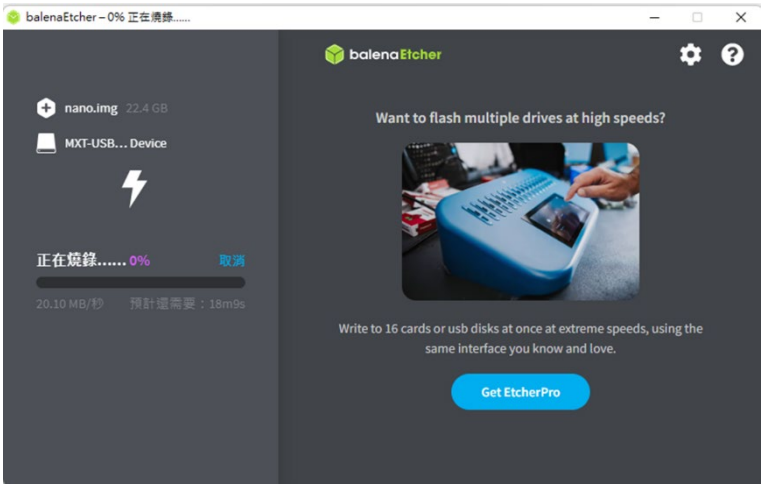


- 選擇要燒錄的 SD 卡 ➢ 紅框 1：選擇目標磁碟 ➜紅框 2：找到要燒錄的磁碟位置 ➜紅框 3：選取 1

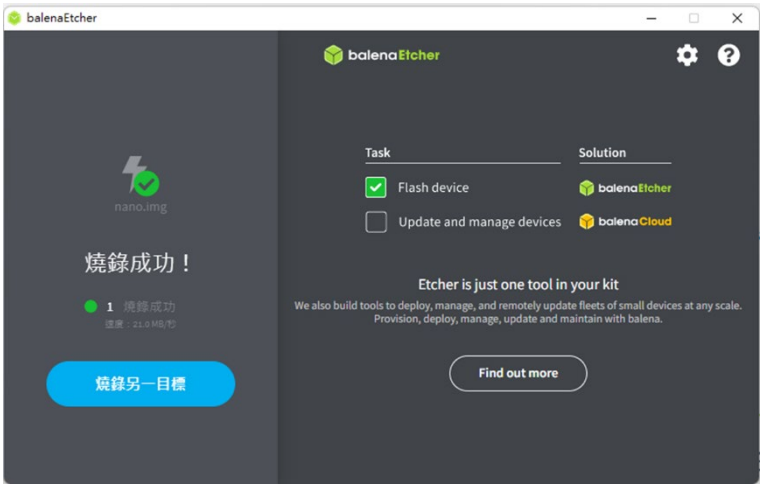


- 現在燒錄!

- 燒錄中



- 燒錄成功



Raspberry Pi4 的 OS 預設帳號：raspdslab 密碼：6517

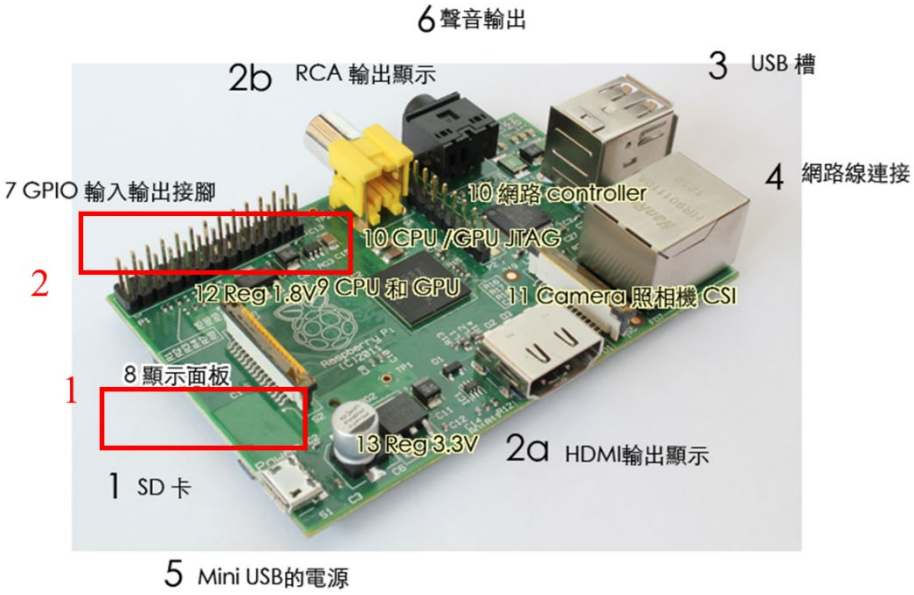
Jetson Nano 的 OS

預設帳號：bn10513137 密碼：6517

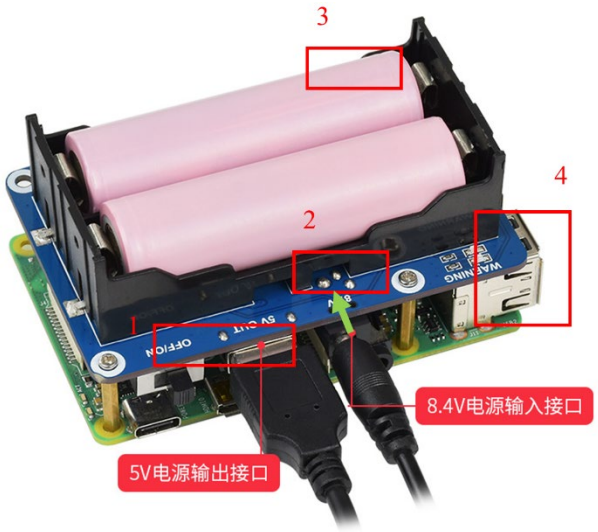
2.2 設定 Client 端程式在嵌入式版上

- Raspberry Pi 4

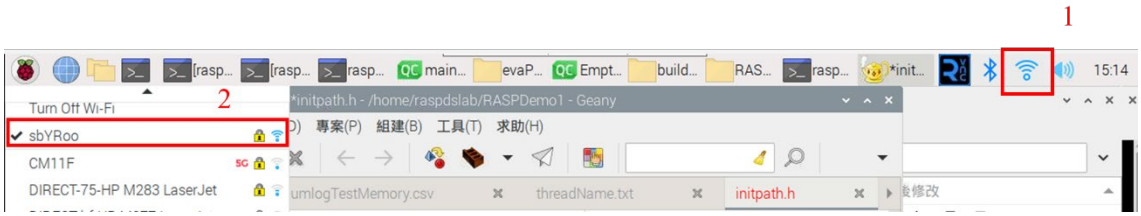
步驟 1. 組裝設備➢紅框 1：裝入燒錄完成的 SD 卡(需確認為 Raspberry Pi 4 的映像檔)➜紅框 2：將樹莓派不斷電模組(UPS HAT)插入 GPIO 接腳：



步驟 2. 連接電源及螢幕➢紅框 1：插入 HDMI 線，另一頭連接螢幕，因第一次使用需使用螢幕查看裝置 IP 位址，以進行 VNC 連線➜ 紅框 2：將 8.4V 電源充電器插入綠色箭頭的地方，不要插原板子上的電源孔，會被壓克力板遮住➜ 紅框 3：開啟不斷電模組開關到 ON➜ 紅框 4：連接網路線或是等下設定 Wi- Fi

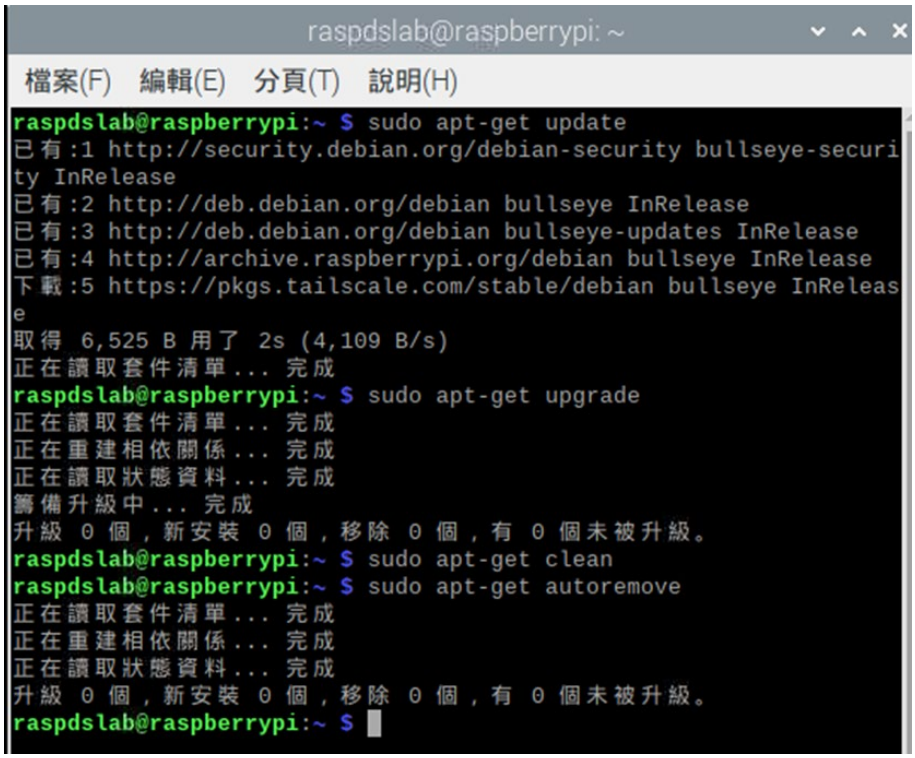


步驟 3.　連接 Wi-Fi ➢紅框 1：點擊 Wi-Fi 圖像➜ 紅框 2：點擊欲連接的 Wi-Fi 名稱➜ 紅框 3：輸入 Wi-Fi 密碼



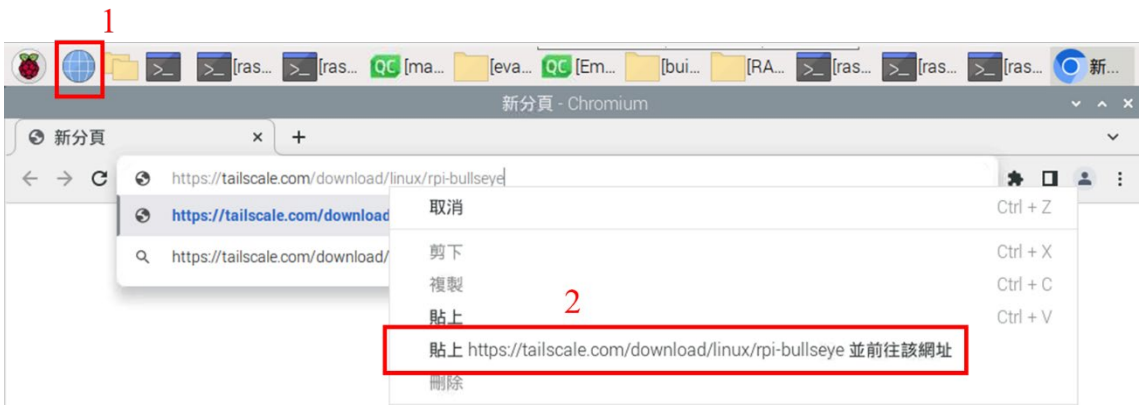
步驟 4.　進行 Ubuntu 更新及清除更新後用不到的舊版本檔案

```
# 用來取得遠端更新伺服器的套件檔案清單

sudo apt-get update

# 更新套件

sudo apt-get upgrade

#清除更新時所下載回來的更新(安裝)檔案

sudo apt-get clean

#自動清除更新後用不到的舊版本檔案（例如舊的核心程式）。

sudo apt-get autoremove
```

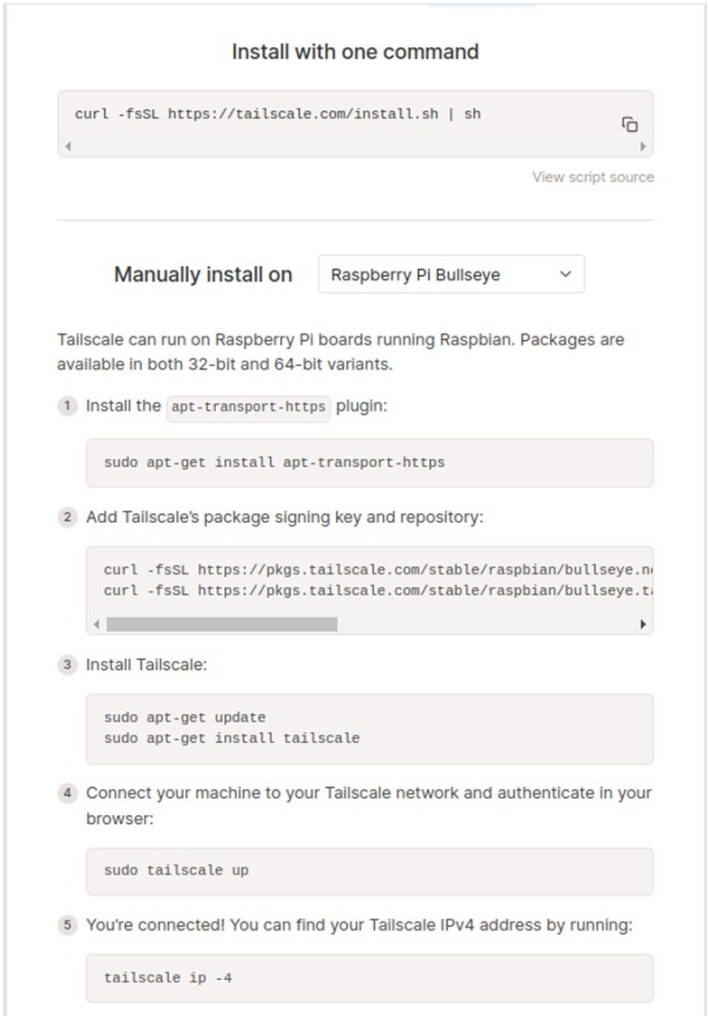


步驟 5.　下載 tailscale(Raspberry Pi 版)，需到下面網址網站上有所有安裝 tailscale 的步驟➢紅框 1：點擊瀏覽器圖示 ➜紅框 2：在瀏覽器中貼上下面的網址

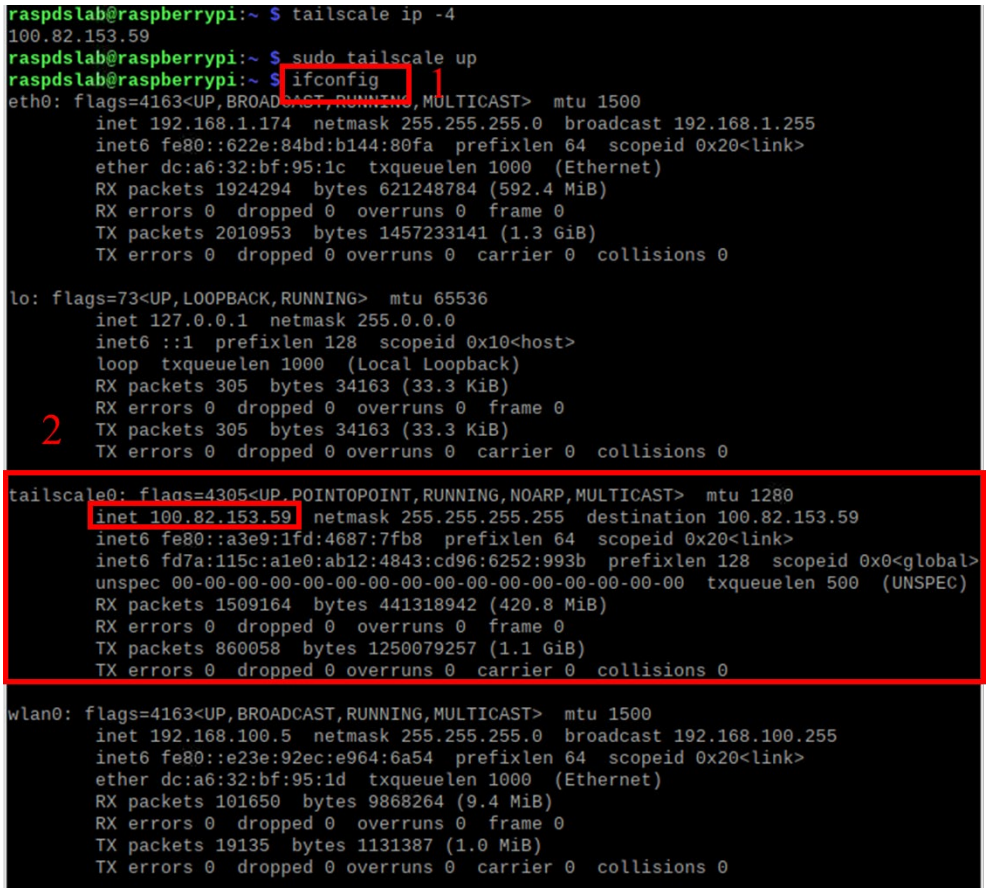
https://tailscale.com/download/linux/rpi-bullseye



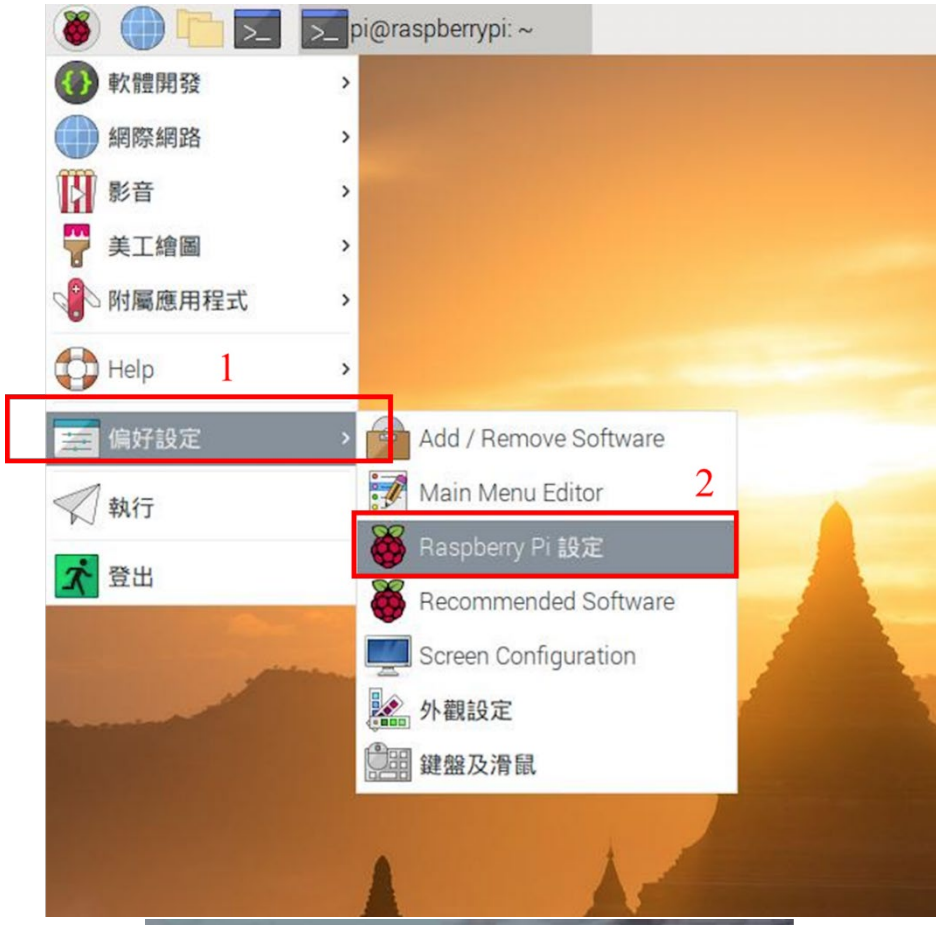
步驟 6.　網頁內容有詳細的安裝指令，照著步驟做就能成功安裝



步驟 7.　尋找 Rasp 的 IP，打開終端機 ➢ 紅框 1：輸入 ifconfig ➜紅框 2：查看 tailscale 區塊的 IP(100.82.153.59)


raspdslab@raspberrypi:~ $ tailscale ip -4
100.82.153.59
raspdslab@raspberrypi:~ $ sudo tailscale up
raspdslab@raspberrypi:~ $ ifconfig
1
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.1.174 netmask 255.255.255.0 broadcast 192.168.1.255
inet6 fe80::622e:84bd:b144:80fa prefixlen 64 scopeid 0x20<link>
ether dc:a6:32:bf:95:1c txqueuelen 1000 (Ethernet)
RX packets 1924294 bytes 621248784 (592.4 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 2010953 bytes 1457233141 (1.3 GiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 305 bytes 34163 (33.3 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
2
TX packets 305 bytes 34163 (33.3 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
tailscale0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1280
inet 100.82.153.59 netmask 255.255.255.255 destination 100.82.153.59
inet6 fe80::a3e9:1fd:4687:7fb8 prefixlen 64 scopeid 0x20<link>
inet6 fd7a:115c:a1e0:ab12:4843:cd96:6252:993b prefixlen 128 scopeid 0x0<global>
unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
RX packets 1509164 bytes 441318942 (420.8 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 860058 bytes 1250079257 (1.1 GiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.100.5 netmask 255.255.255.0 broadcast 192.168.100.255
inet6 fe80::e23e:92ec:e964:6a54 prefixlen 64 scopeid 0x20<link>
ether dc:a6:32:bf:95:1d txqueuelen 1000 (Ethernet)
RX packets 101650 bytes 9868264 (9.4 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 19135 bytes 1131387 (1.0 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

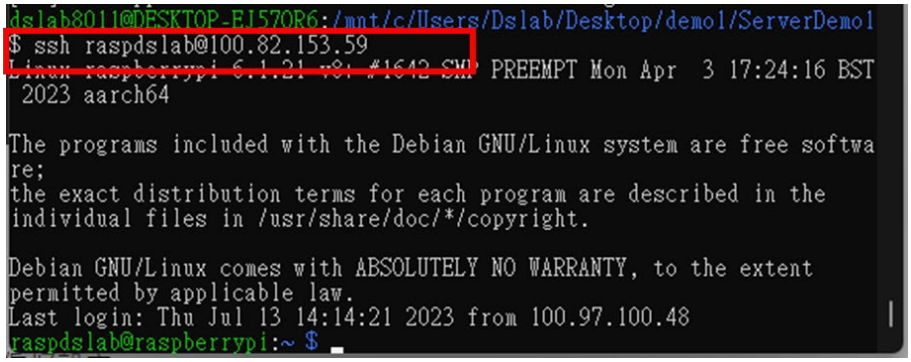
步驟 8.　開啟 SSH 開關及 VNC 開關➢ 紅框 1：偏好設定 ➜紅框 2：Raspberry Pi 設定➜紅框 3：點擊 SSH 開關及 VNC 開關



步驟 9.　從電腦連接 SSH 到 RASP 裝置

```
#ssh 帳號@ IP address(輸入剛查到 tailscale 的位址)

ssh raspdslab@100.82.153.59
```

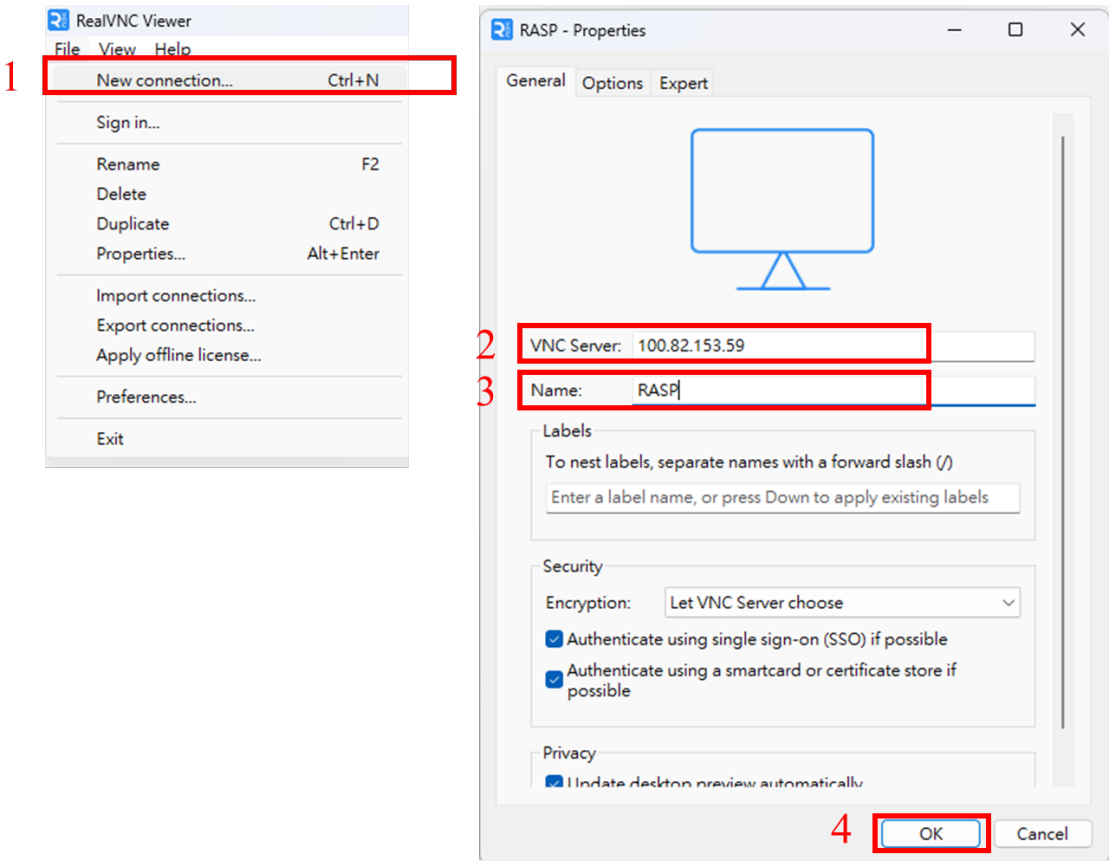
1



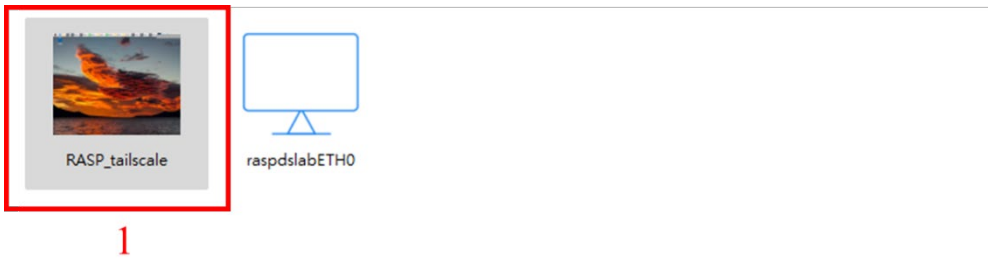
步驟 10. 使用 VNC 從電腦連接到 RASP 裝置，點擊下面超連結下載 windows 版的 VNC-7.5.1

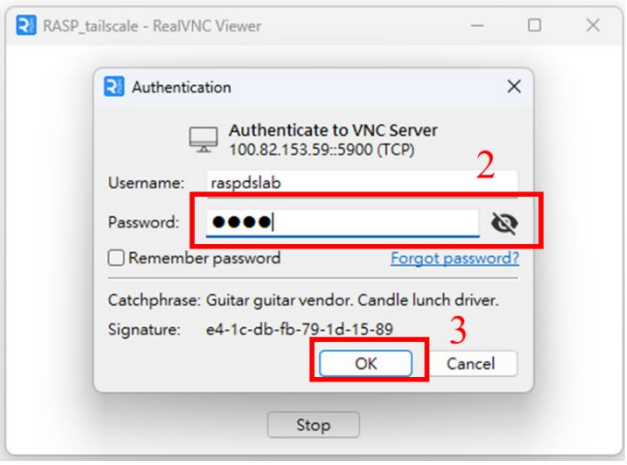
https://downloads.realvnc.com/download/file/viewer.files/VNC-Viewer-7.5.1-Windows.exe

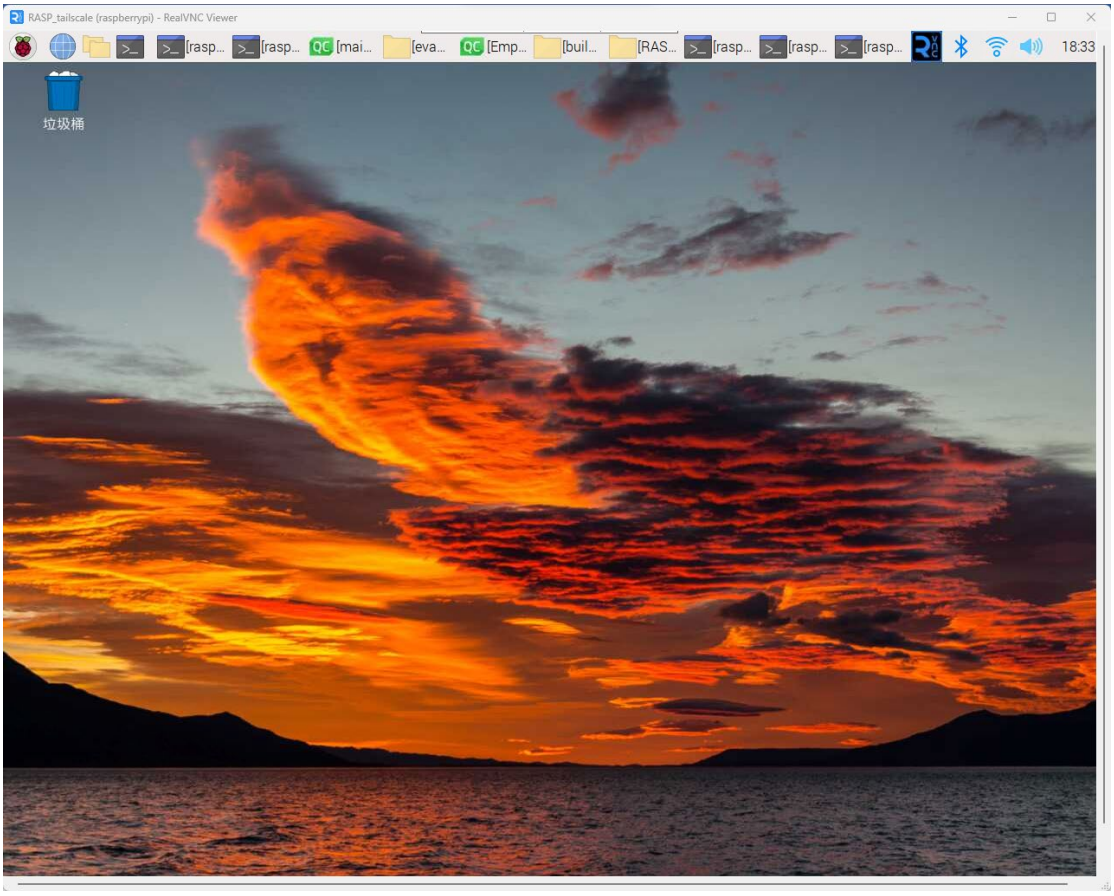
步驟 11. 打開電腦 VNC 軟體➢ 紅框 1：New connection ➜紅框 2：輸入 100.82.153.59➜紅框 3：輸入代稱(看得懂就好) ➜紅框 4：OK



步驟 12. 回到 VNC 軟體主畫面➢ 紅框 1：點擊設定好的連線 ➜紅框 2：輸入 RASP 的密碼➜紅框 3：OK 。第二張圖為 VNC 連線後畫面

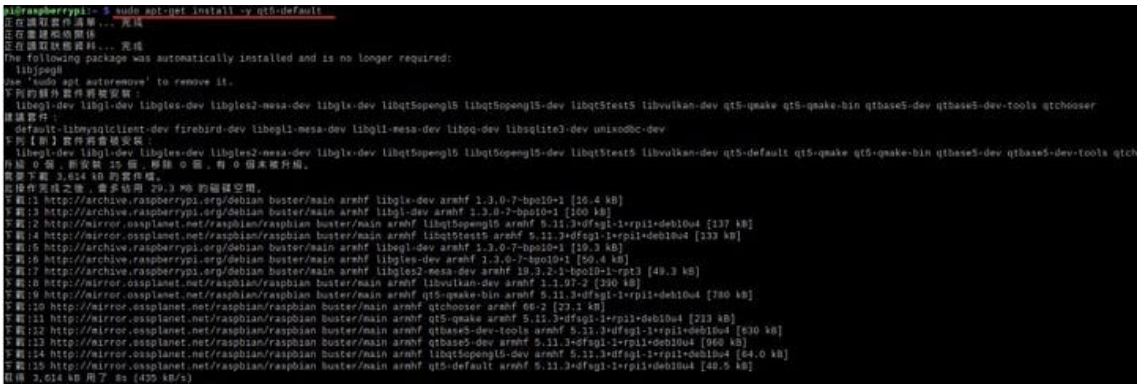


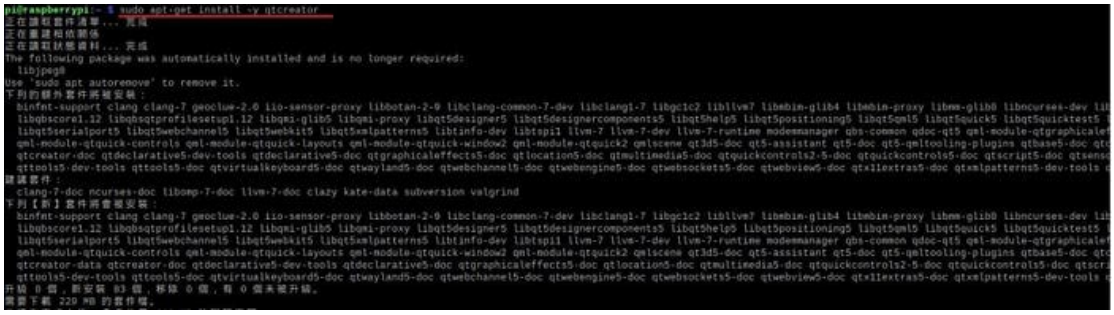




步驟 13. 安裝 Qt – 5 程式

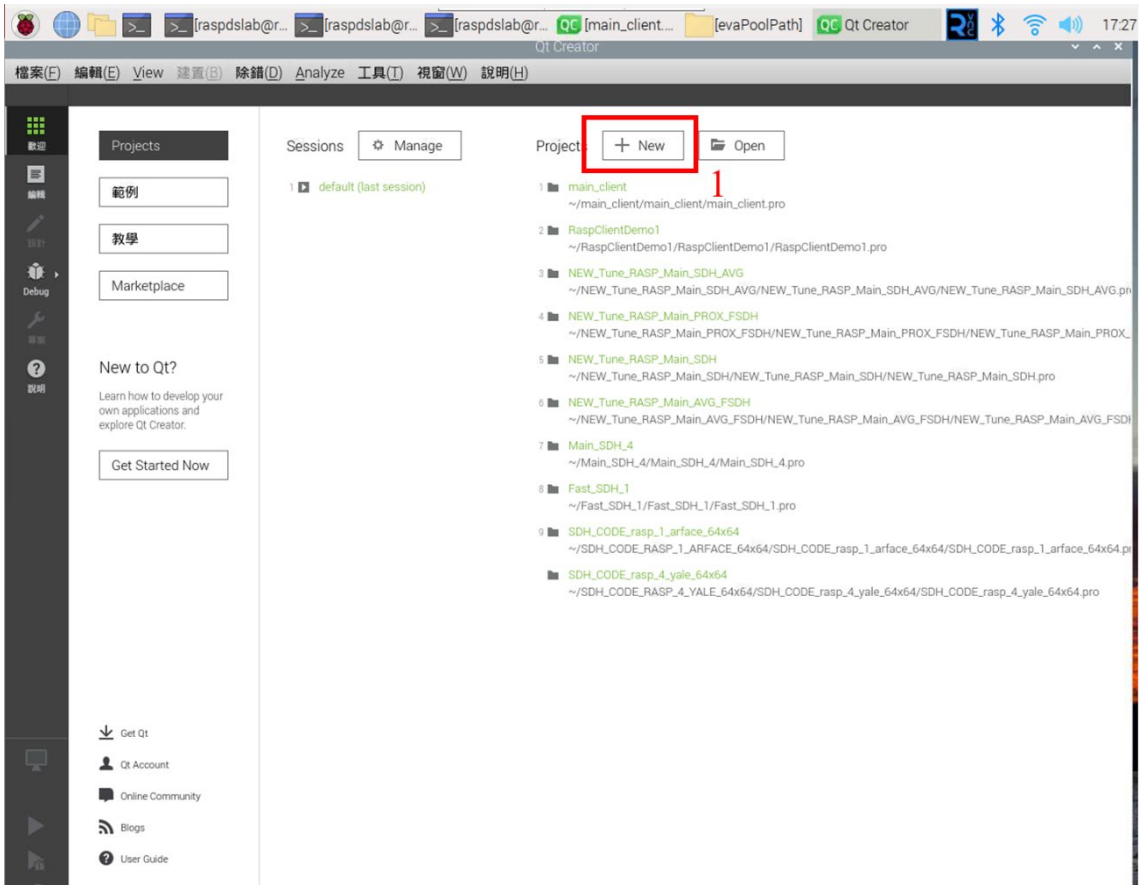
```
# 更新套件
sudo apt-get update
sudo apt-get upgrade
#安裝 Qt 套件
sudo apt-get install -y qt5-default
sudo apt-get install -y qtcreator
```



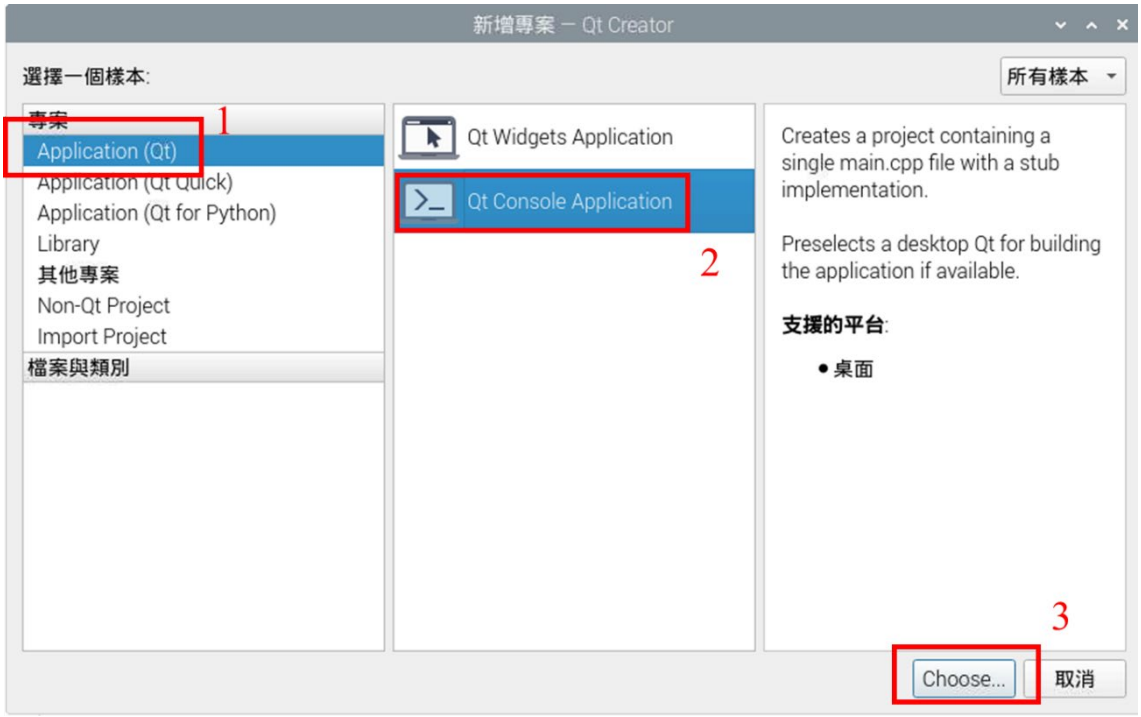


步驟 14. 開啟 Qt – 5 程式的 Qt Creator 編輯器

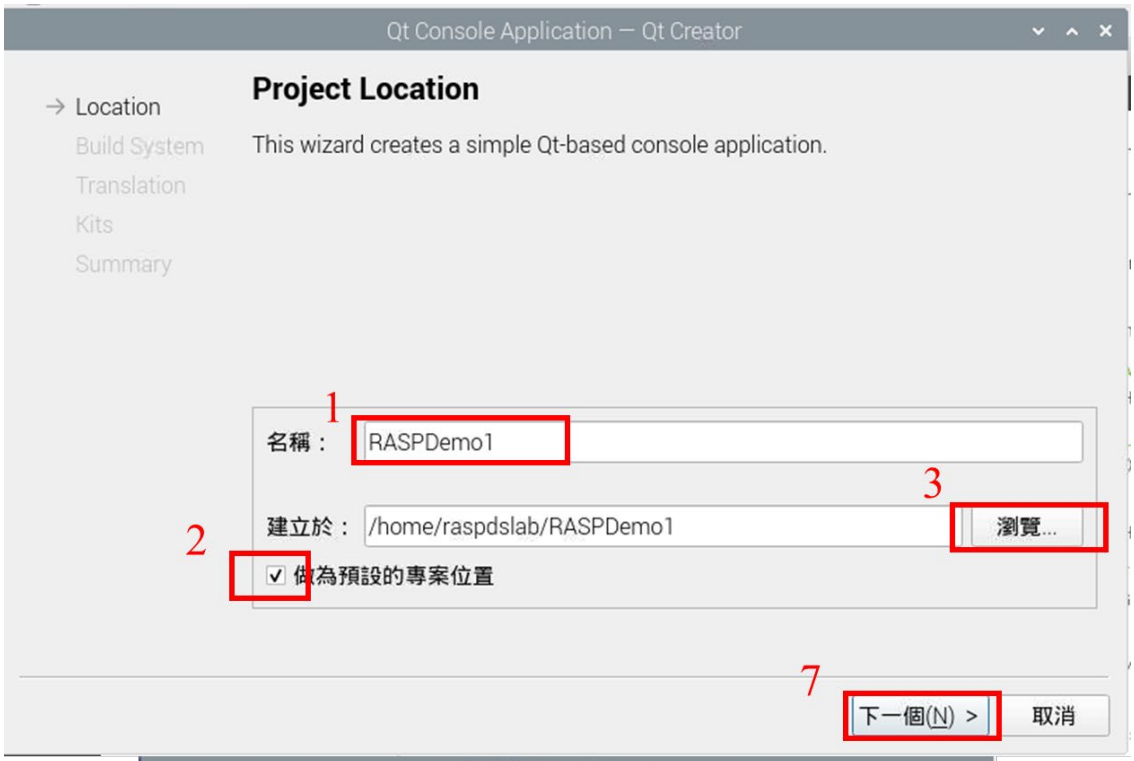
步驟 15. 創建新專案 + New

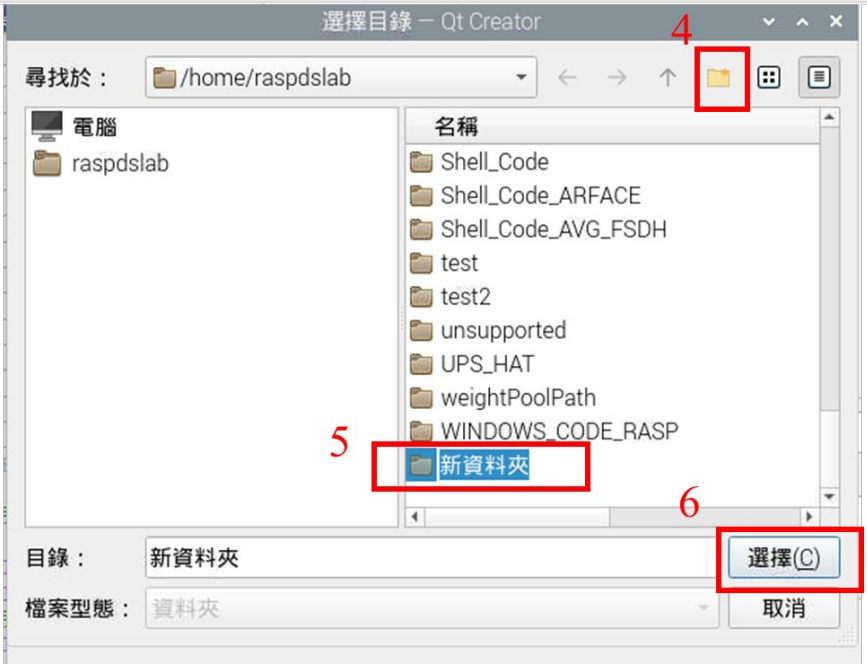


步驟 16. 創建專案 ➢ 紅框 1：選取 Application(Qt) ➜紅框 2：選取 Qt Console Application ➜紅框 3：Choose

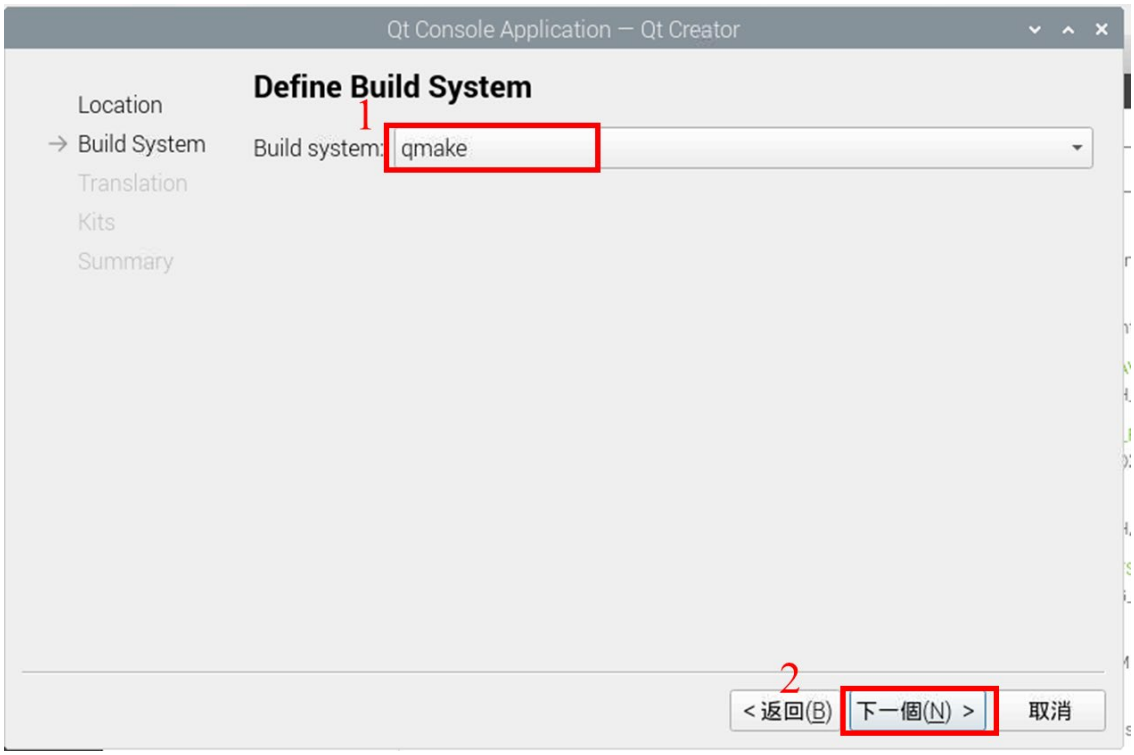


步驟 17. 創建專案，新增放置專案文件的資料夾➢ 紅框 1：新增專案名稱 ➜ 紅框 2：作為預設的專案位置打勾 ➜ 紅框 3：按下瀏覽按鈕 ➜ 紅框 4：新增資料夾 ➜ 紅框 5：將改名為剛創建的資料夾名 ➜ 紅框 6：選擇 ➜ 紅框 7：下一個

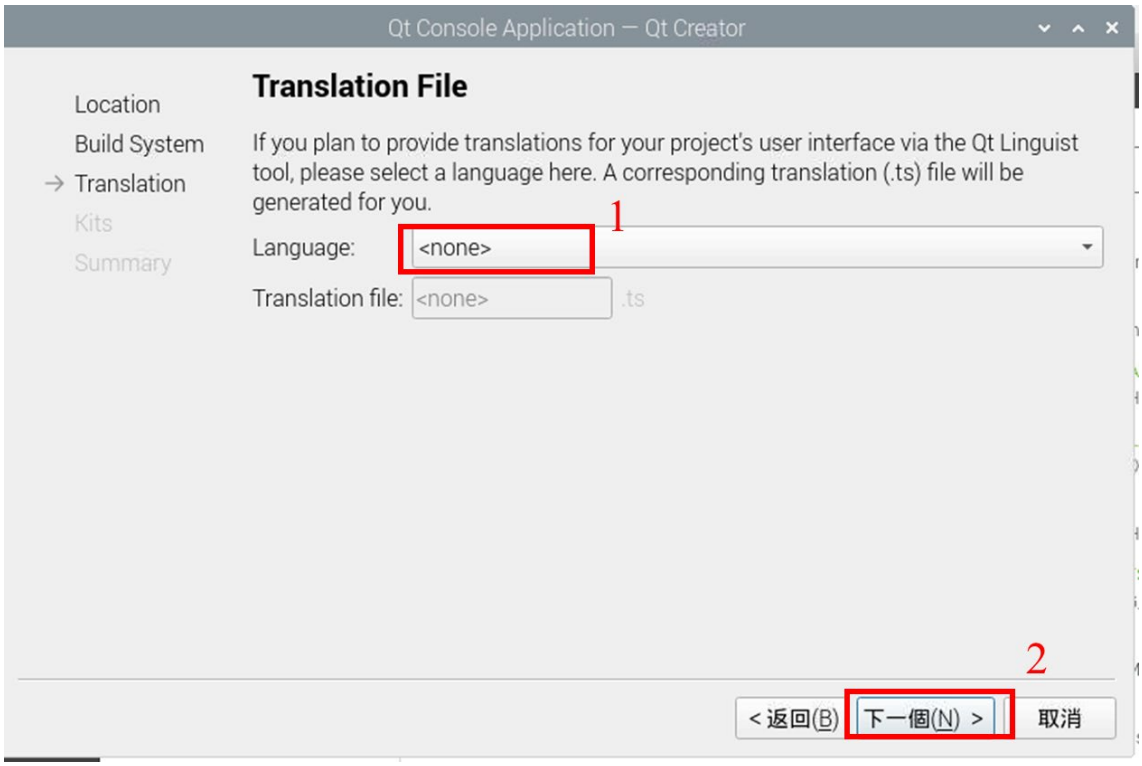




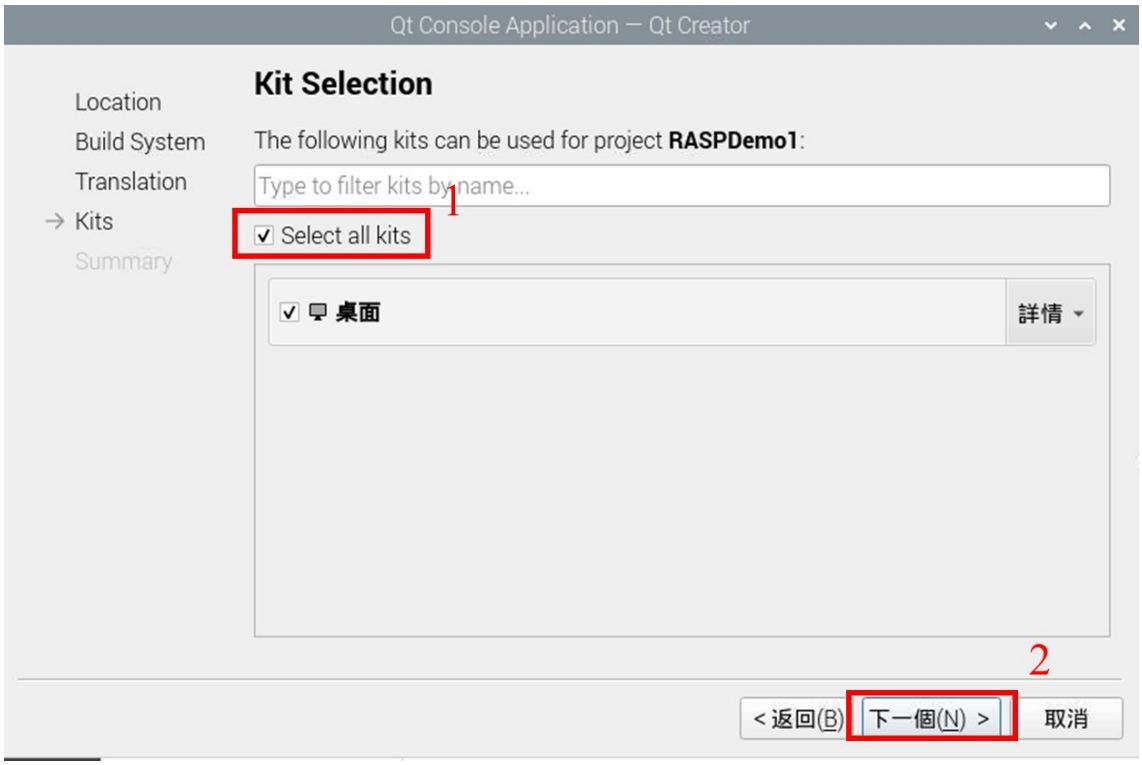
步驟 18. 創建專案 ➢ 紅框 1：選取 qmake ➜ 紅框 2：選取 下一個



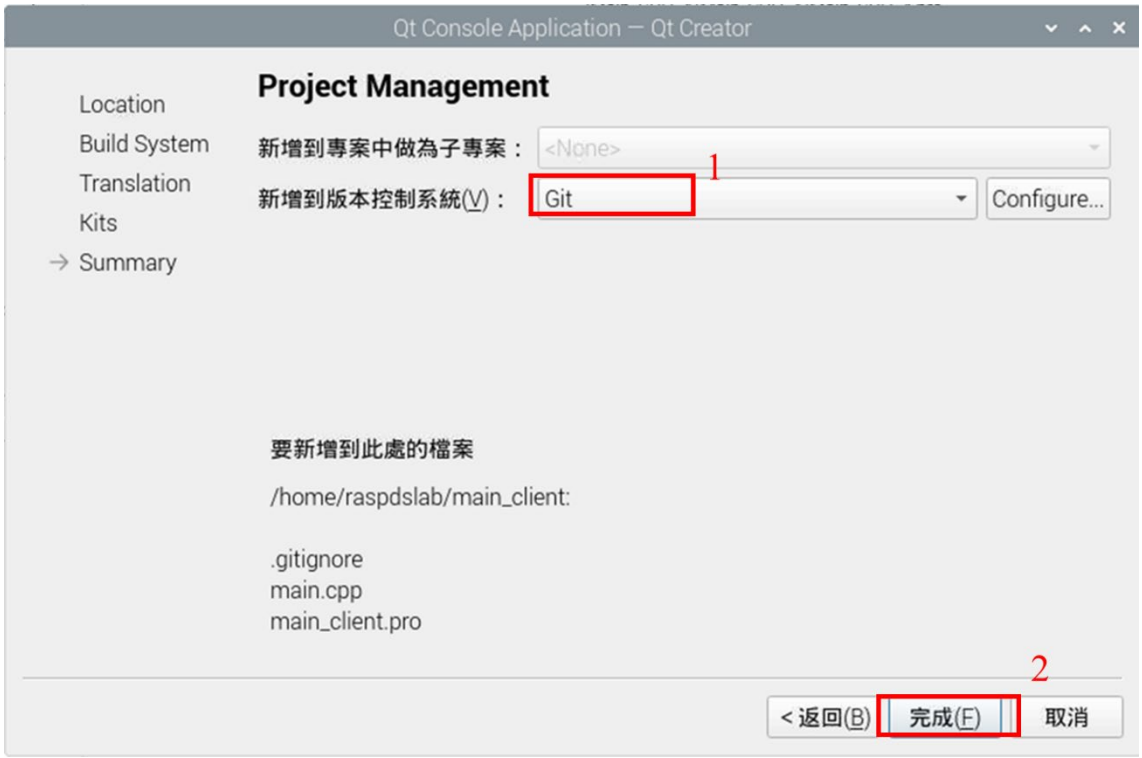
步驟 19. 創建專案 ➢ 紅框 1：選取 none ➜ 紅框 2：選取 下一個



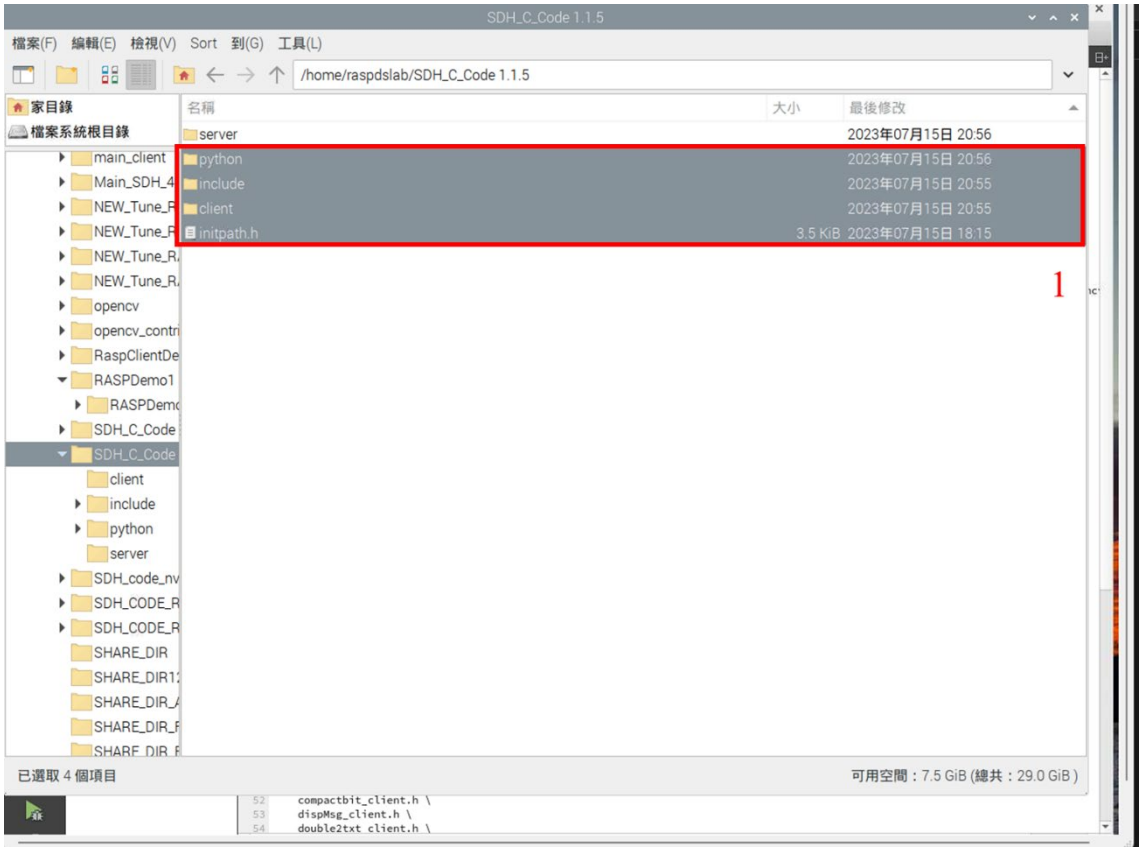
步驟 20. 創建專案 ➢ 紅框 1：勾選 select all kits ➜ 紅框 2：選取 下一個

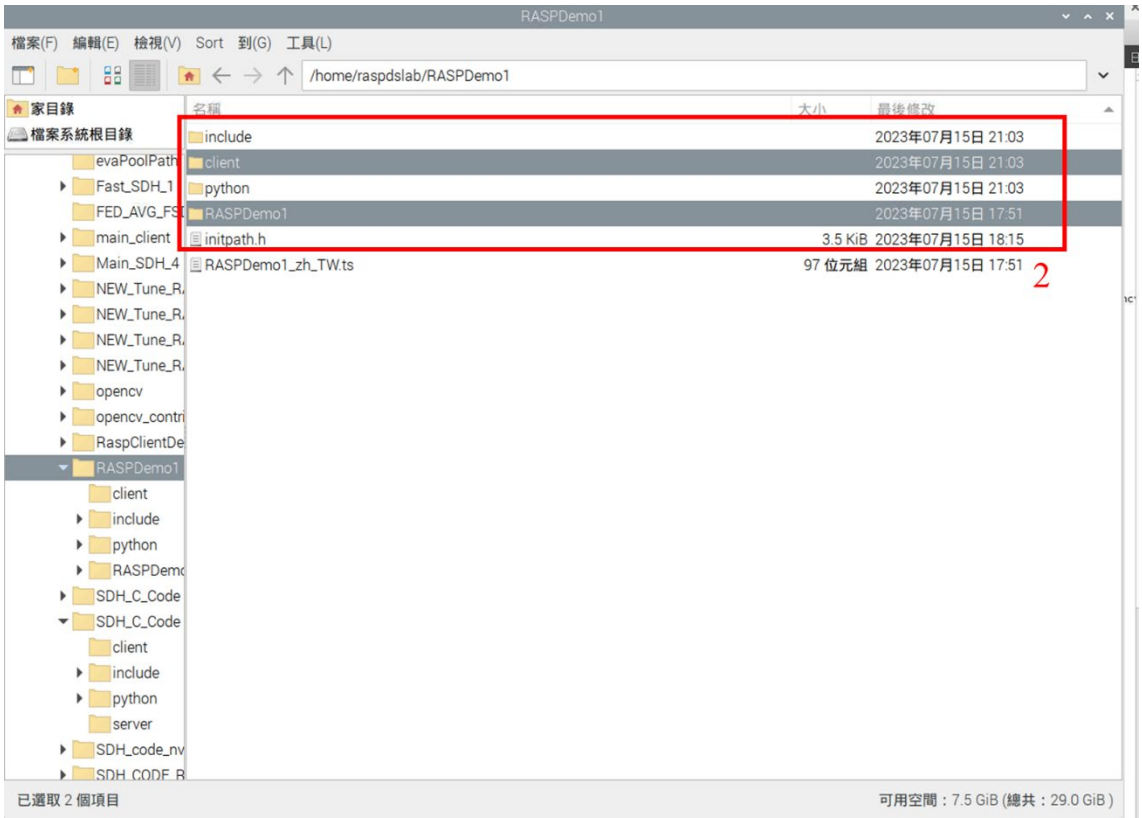


步驟 21. 創建專案 ➢ 紅框 1：勾選 Git ➜ 紅框 2：選取 完成

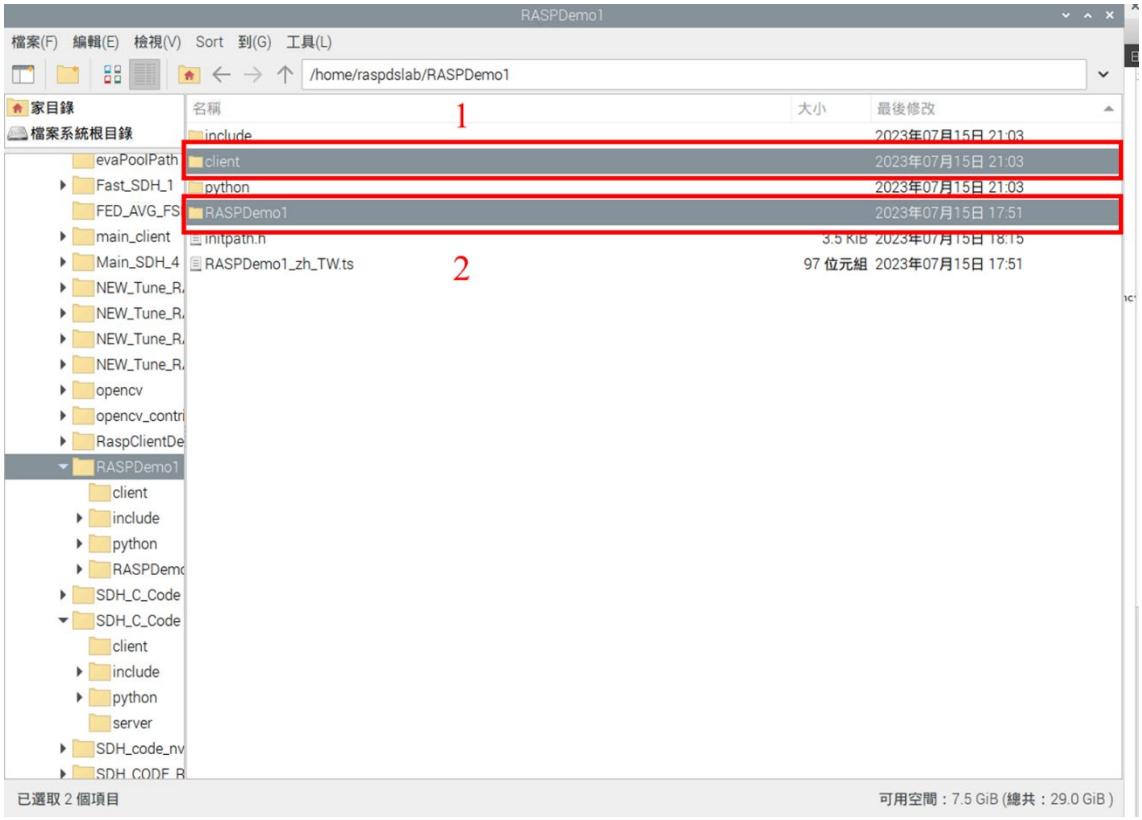


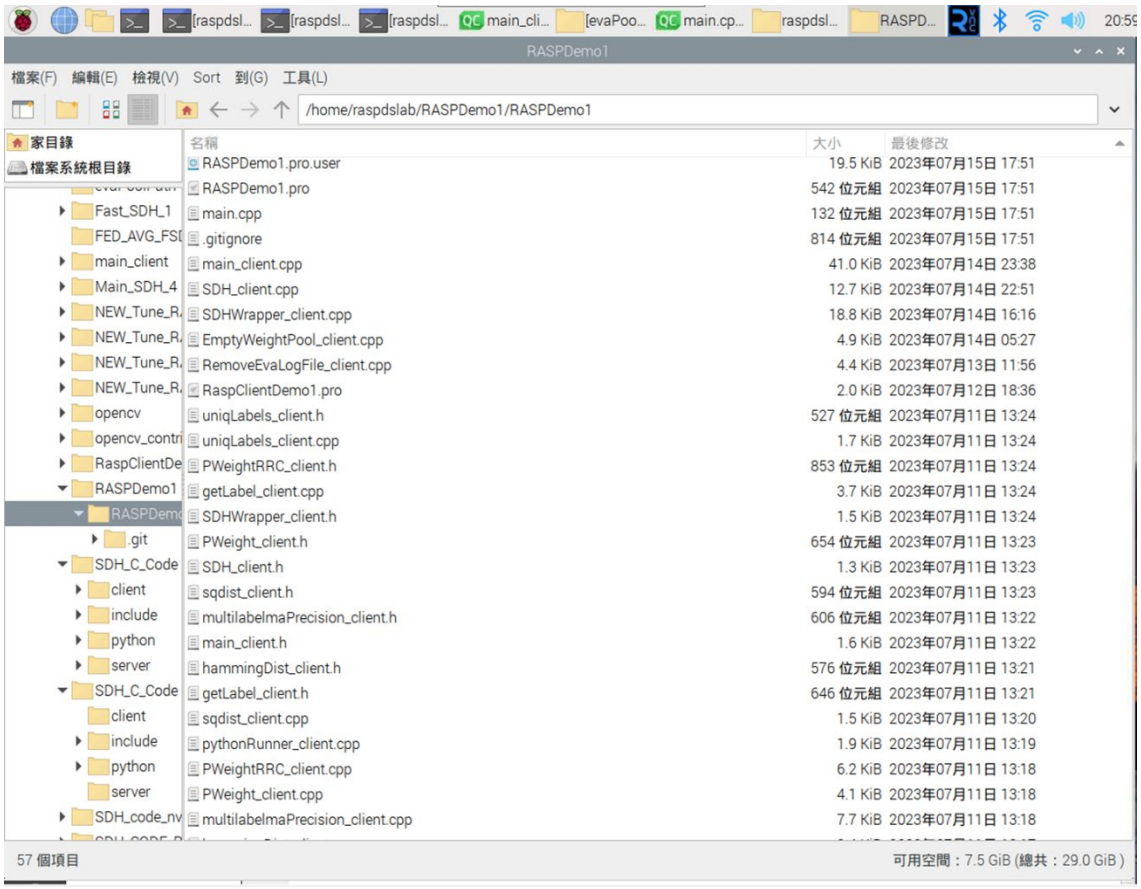
步驟 22. 放入所需檔案 ➢ 紅框 1：將紅框的四個資料夾或檔案複製到剛剛創建專案資料夾底下 ➜ 紅框 2：複製完成畫面



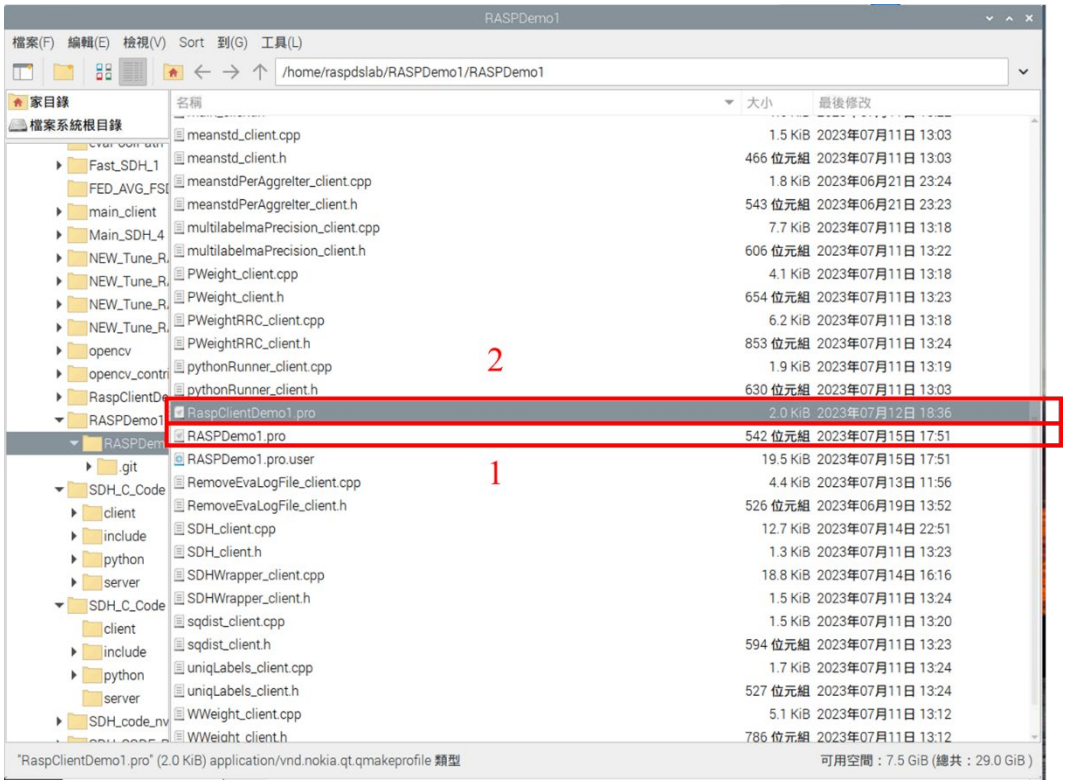


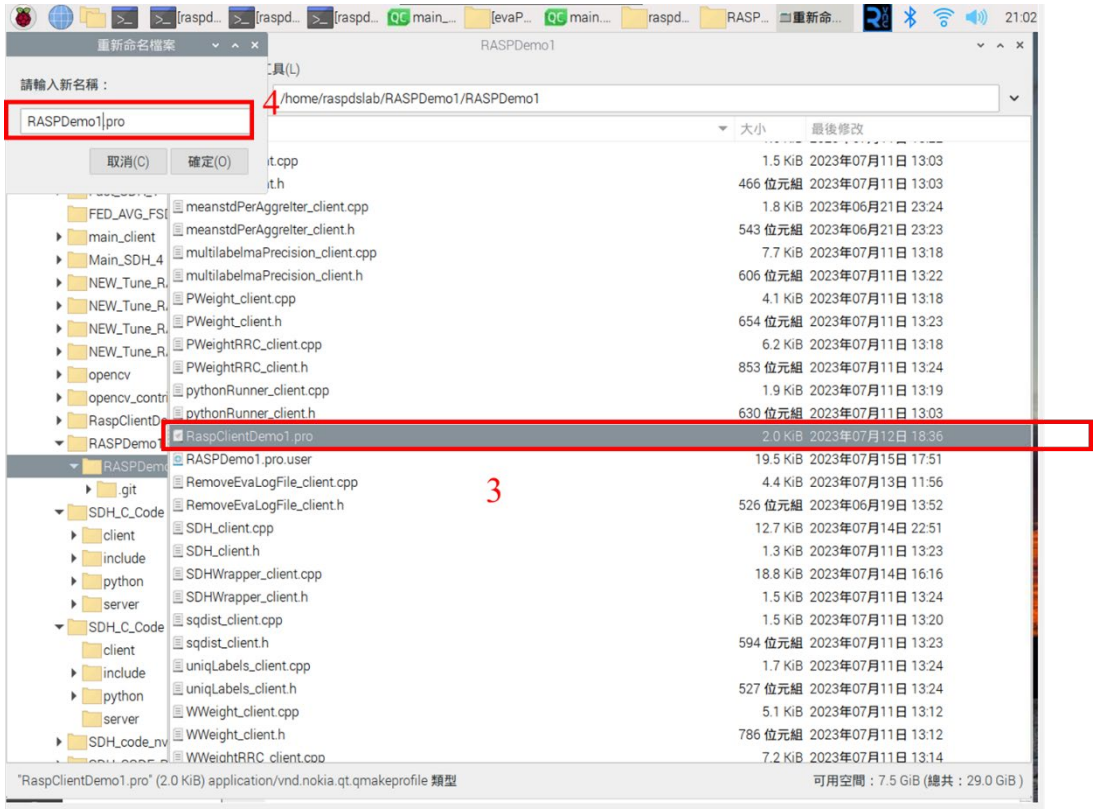
步驟 23. 複製 Client 端的檔案 ➢ 將紅框 1 資料夾的所有檔案複製到紅框 2 放置專案檔的資料夾底下，如第二張圖所示





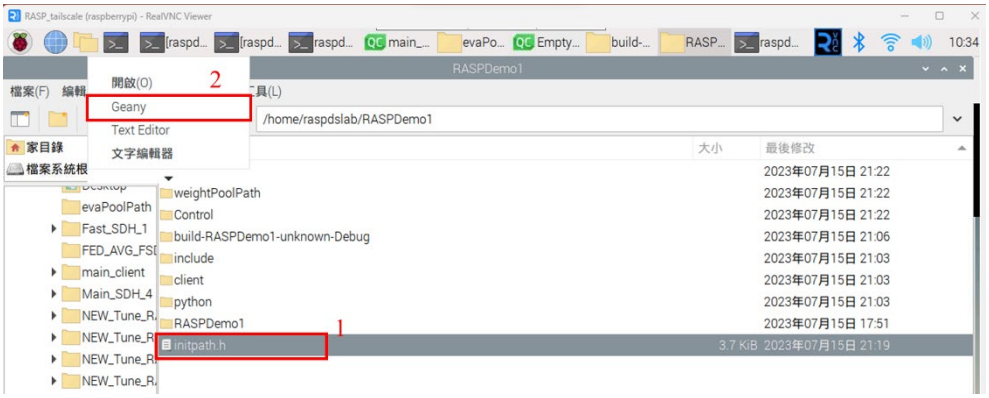
步驟 24. 複製 Client 端的檔案 ➢ 將紅框 1 移除，並將紅框 2 的檔名改為專案檔名.pro，也就是紅框 1 的名字，如第 2 張圖所示 ➜ 紅框 3 ：點取右鍵重新命名 ➜ 紅框 4：將名字改為專案檔名





步驟 25. 因測量 Memory 時會去抓執行序名稱，所以必須修改路徑檔 initpath.h 中，執行序名稱，➢ 紅框 1：對路徑檔 initpath.h 點擊右鍵➜ 紅框 2：跳出選單，點選 Geany ➜ 紅框 3：將執行序名稱改為./專案檔名 ➜ 紅框 4：改完結果

步驟 26.



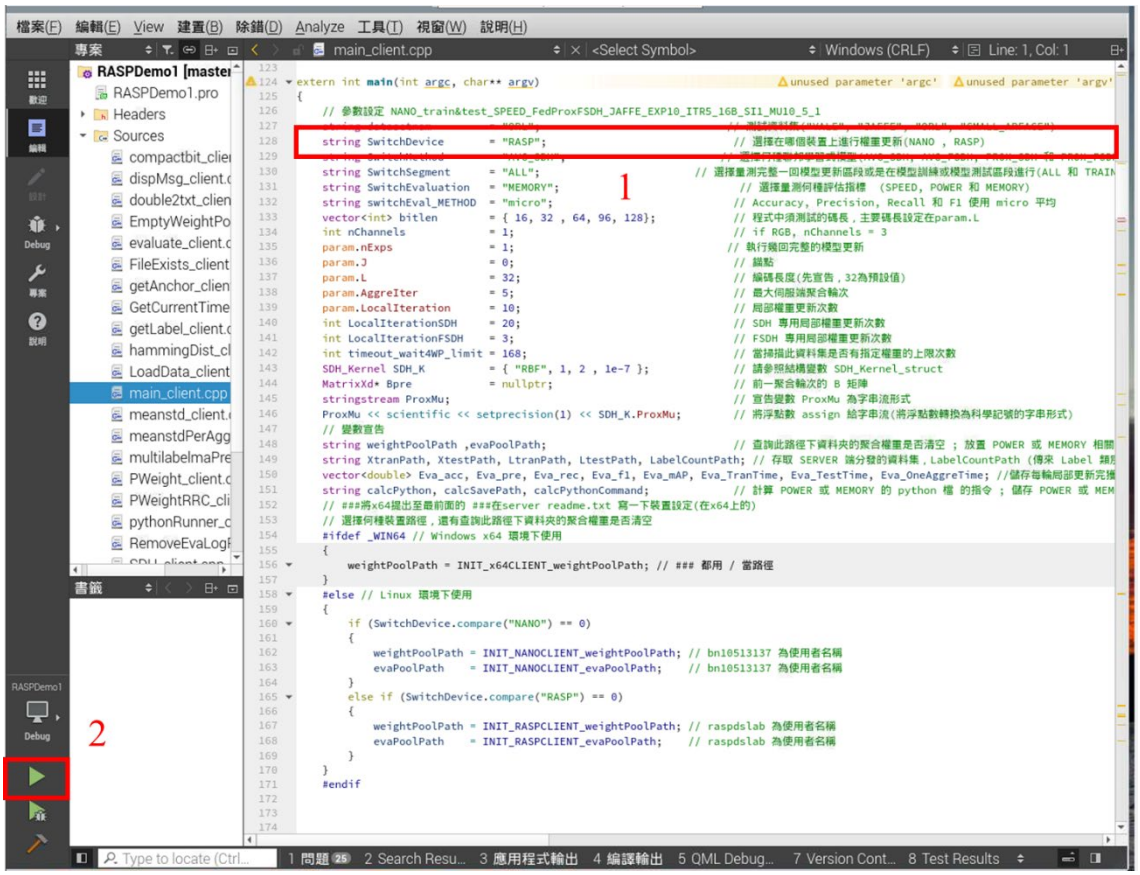

```
#pragma once
#ifndef INITPATH_H
#define INITPATH_H

// Define the paths here

#define INIT_CLIENT_SERVER2              "/SERVER2"
#define INIT_CLIENT_Fed                  "_Fed"
#define INIT_CLIENT_EXP                  "_EXP"
#define INIT_CLIENT_ITR                  "_ITR"
#define INIT_CLIENT_B_SI                 "B_SI"
#define INIT_CLIENT_MU                   "_MU"
#define INIT_CLIENT_txt                  ".txt"
#define INIT_CLIENT_XtranMtxTxt          "_X_tran_matrix.txt"
#define INIT_CLIENT_XtuneMtxTxt          "_X_tune_matrix.txt"
#define INIT_CLIENT_XtestMtxTxt          "_X_test_matrix.txt"
#define INIT_CLIENT_LtranMtxTxt          "_L_tran_matrix.txt"
#define INIT_CLIENT_LtuneMtxTxt          "_L_tune_matrix.txt"
#define INIT_CLIENT_LtestMtxTxt          "_L_test_matrix.txt"
#define INIT_CLIENT_PWeightMtxTxt        "_PWeight_matrix.txt"
#define INIT_CLIENT_WWeightMtxTxt        "_WWeight_matrix.txt"
#define INIT_CLIENT_LabelListMtxTxt      "_LabelList_matrix.txt"
#define INIT_CLIENT_LabelCountMtxTxt     "_LabelCount_matrix.txt"
#define INIT_CLIENT_threadNamePath       "threadName.txt"
#define INIT_CLIENT_threadName           "./RaspClientDemo1"
```

3

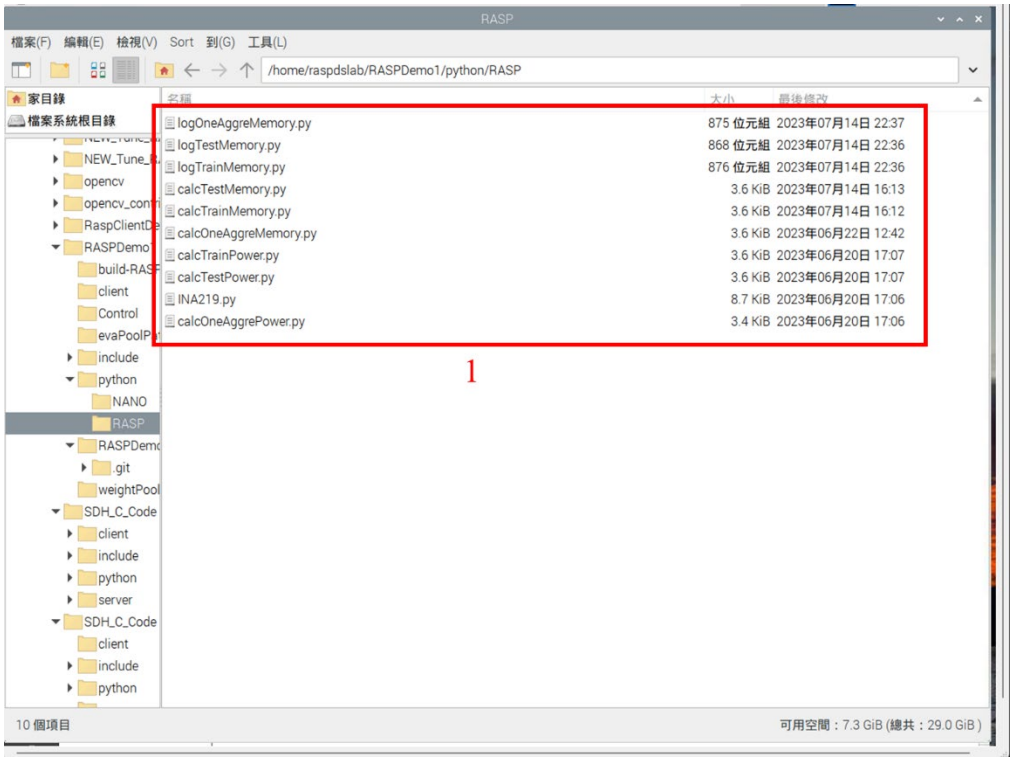
```
#define INIT_CLIENT_LabelCountMtxTxt     "_LabelCount_matrix.txt"
#define INIT_CLIENT_threadNamePath       "threadName.txt"
#define INIT_CLIENT_threadName           "./RaspDemo1"
```

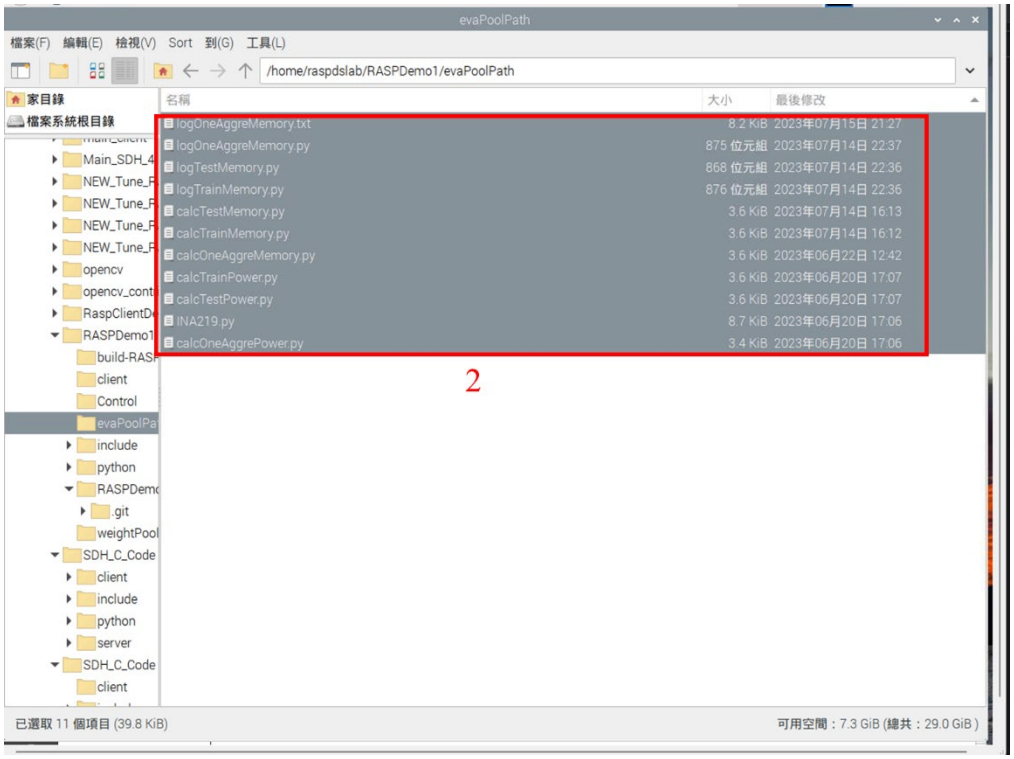
4

步驟 27. 打開 RASP 專案檔 ➢ 紅框 1：檢查是否為"RASP" ➜ 紅框 2：點選綠色三角形按鈕編輯程式，等待跑完。跑完結果如第二張圖所示 ➜ 紅框 3：等程式跳出以下畫面就能按紅色正方形按鈕中斷，輸出內容有錯是因為還沒放入檢測 Power 及 Memory 的 Python 檔，所以是正常的

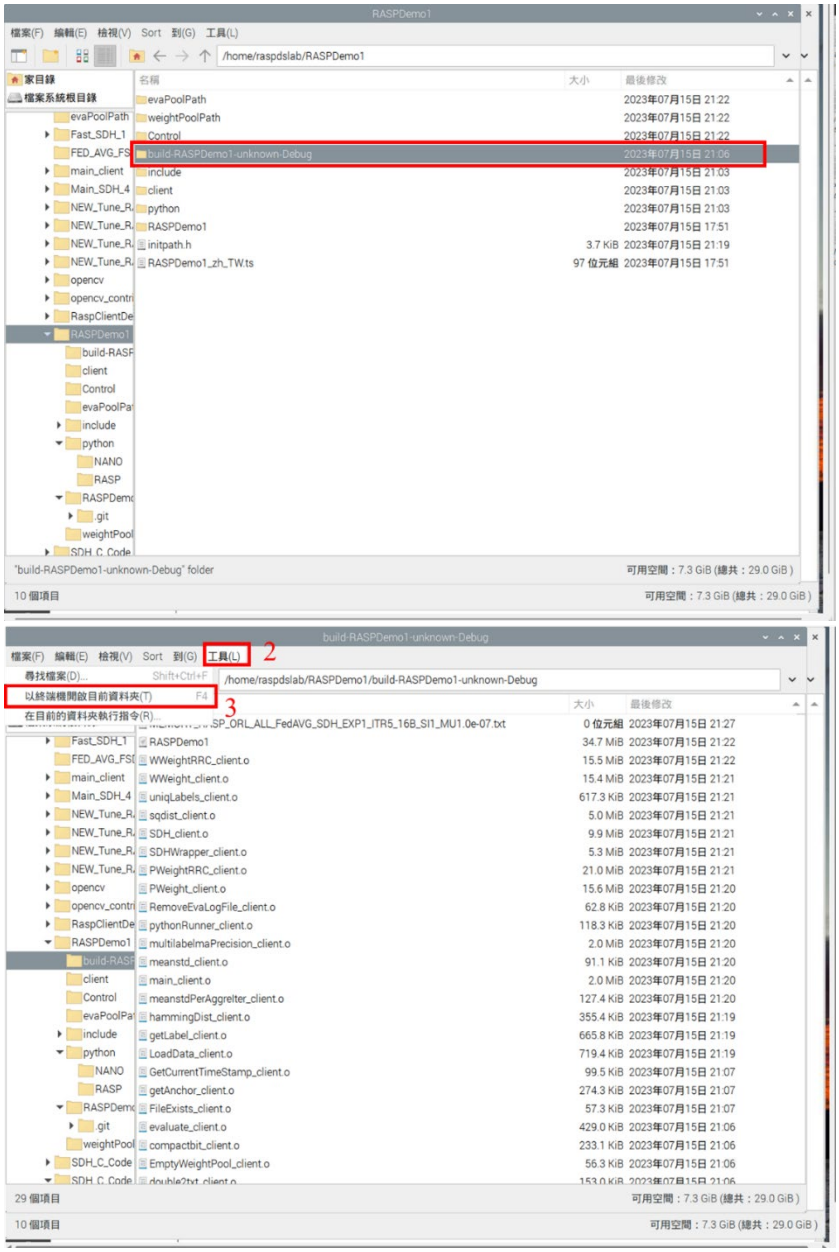


步驟 28. 複製檢測 Power 及 Memory 的 Python 檔➢ 紅框 1：將 python/RASP 資料夾裡的檔案複製到專案資料夾裡的 evaPoolPath (/home/raspdslab/專案名/evaPoolPath)，evaPoolPath 資料夾剛編輯程式時會自動創建不用自己手動創建 ➜ 紅框 2：複製結果如下

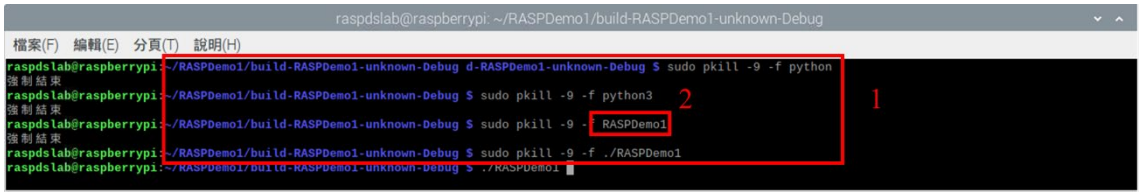




步驟 29. 打開終端機執行程式 ➢ 紅框 1：到編輯完程式後自動創建的一個資料夾中 build-RASPDemo1-unknown-Debug (/home/raspdslab/專案名/build-RASPDemo1-unknown-Debug)，裡面有我們編輯完的程式執行檔➜ 紅框 2：點擊工具➜ 紅框 3：以磁碟機開啟目前資料夾

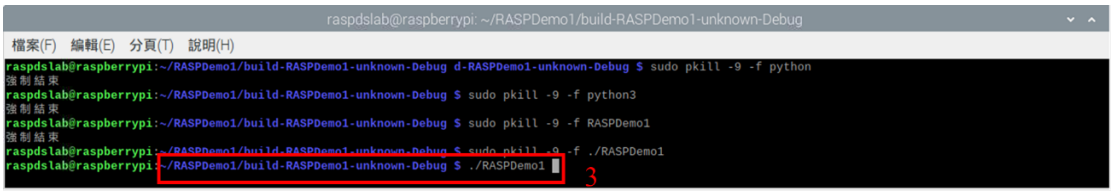


步驟 30. 在終端機上輸入執行命令



➢ 紅框 1：輸入下面四行中斷指令

```
# 四行中斷指令用於刪除執行序，如果程式執行到一半未跑完，有可能程式會在背景執行，尤其是檢測 Power 及 Memory 的 Python 檔，最有可能

sudo pkill -9 -f python

sudo pkill -9 -f python3

sudo pkill -9 -f 專案檔名(例如紅框 2 的 RASPDemo1)

sudo pkill -9 -f ./專案檔名(需在檔名前面加上 ./)
```

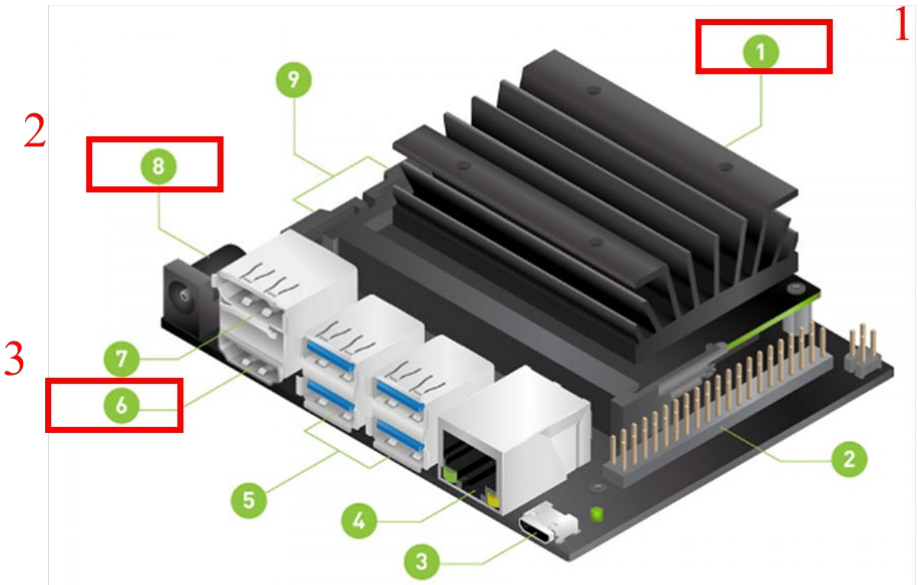


➢ 紅框 3：輸入執行程式指令(但先別按 Enter 執行)

```
# 執行程式執行檔

./專案檔名(需在檔名前面加上 ./)
```

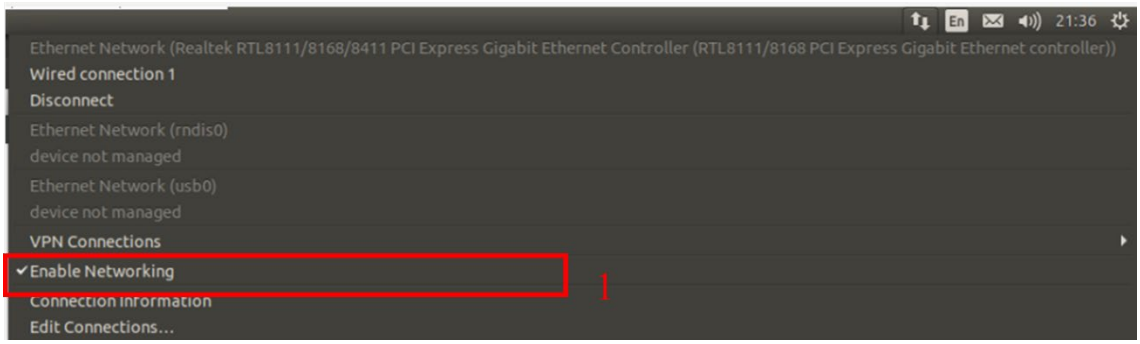
- Jetson Nano

步驟 1.　組裝裝置 ➢ 紅框 1：裝入燒錄完成的 SD 卡(需確認為 Jetson Nano 的映像檔) ➜ 紅框 2：插入電源線(5V) ➜ 紅框 3：插入 HDMI 線另一頭連接螢幕，因第一次使用需使用螢幕查看裝置 IP 位址，



1. Micro SD卡卡槽
   可接入16GB以上的TF卡，燒寫系統鏡像
2. 40PIN GPIO擴展接口
3. Micro USB接口
   可用於5V電源輸入或者USB數據傳輸
4. 千兆乙太網端口
   10/100/1000Base-T自適應乙太網端口
5. 4路USB 3.0接口
6. HDMI高清接口
7. DisplayPort接口
8. DC電源接口(用於5V電源輸入)
9. 2路MIPI CSI攝像頭接口

步驟 2.　因 Jetson Nano 本身沒有 Wi-Fi，只能連接乙太網路➢紅框 1：開啟網路 Enable Netwoking

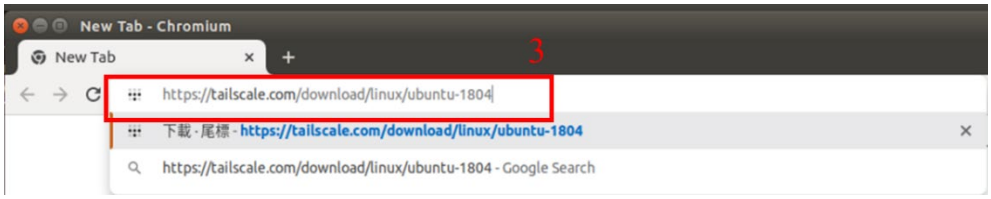


步驟 3.　進行 Ubuntu 更新及清除更新後用不到的舊版本檔案(由於這邊跟前面 RASP 一致不再放圖片了)

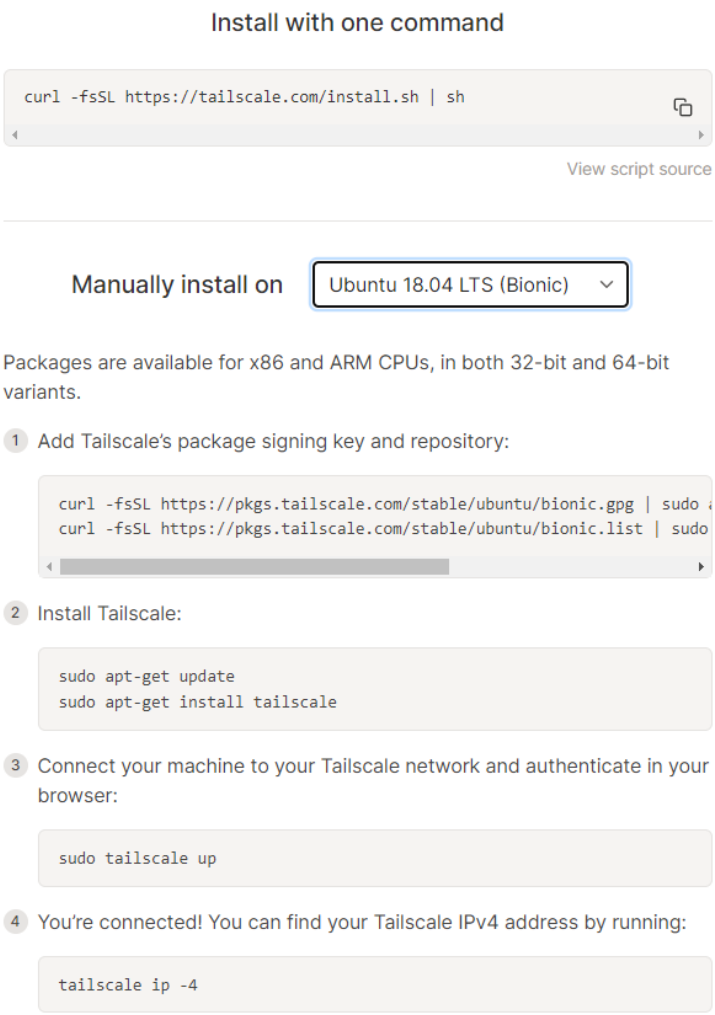
```
# 用來取得遠端更新伺服器的套件檔案清單
sudo apt-get update
# 更新套件
sudo apt-get upgrade
#清除更新時所下載回來的更新(安裝)檔案
sudo apt-get clean
#自動清除更新後用不到的舊版本檔案（例如舊的核心程式）。
sudo apt-get autoremove
```

步驟 4.　下載 tailscale(Jetson Nano 板上)，需到下面網址網站上有所有安裝 tailscale 的步驟➢紅框 1：點擊綠色圖示 ➜紅框 2：點擊藍色瀏覽器➜紅框 3：瀏覽器中貼上下面的網址

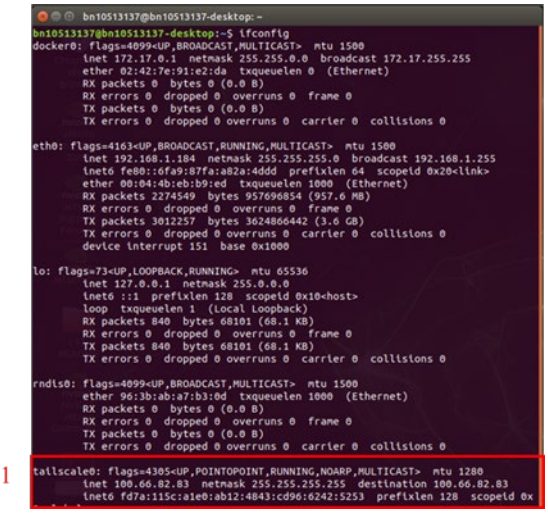
https://tailscale.com/download/linux/ubuntu-1804



步驟 5.　網頁內容有詳細的安裝指令，照著步驟做就能成功安裝



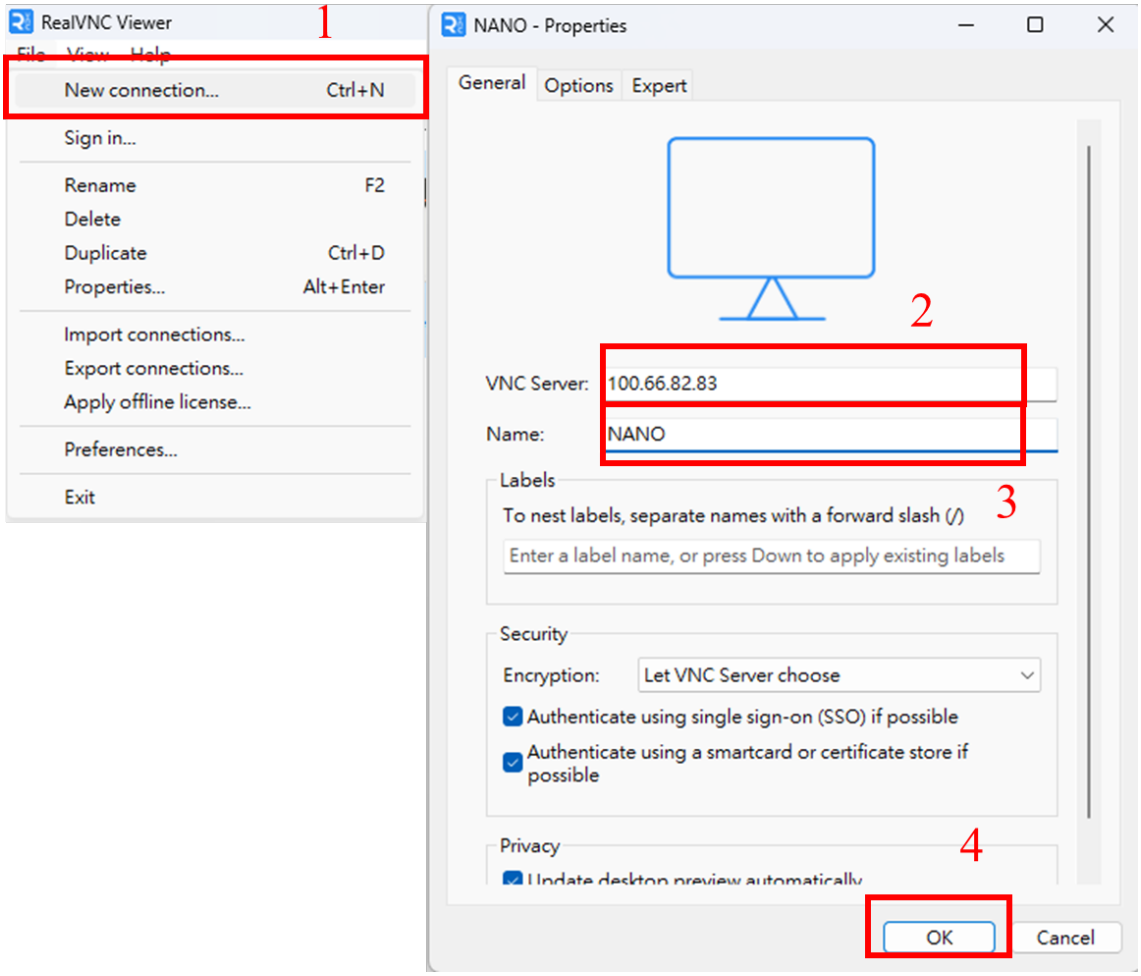
步驟 6.　尋找 NANO 的 IP，打開終端機 ➢ 紅框 1：輸入 ifconfig ➜紅框 2：查看 tailscale 區塊的 IP(100.66.82.83)



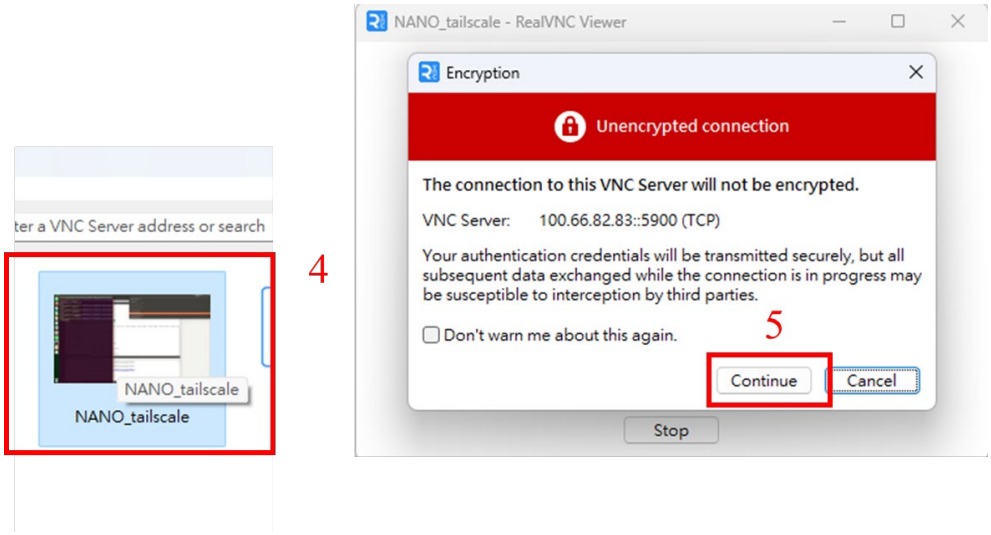
步驟 7.　SSH(可以直接使用)，但須安裝 VNC ➢ 直接輸入下面指令，即可安裝 VNC

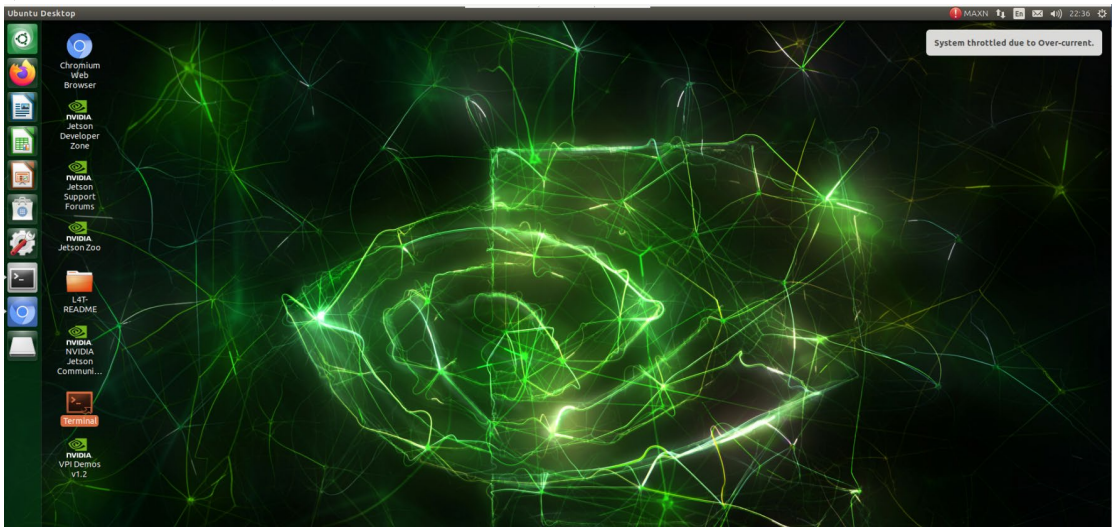
```
# 用來取得遠端更新伺服器的套件檔案清單
sudo apt-get update
# 更新套件
sudo apt-get upgrade
# 安裝 Vino-VNC Server
sudo apt install vino
#使用 gsettings 來調整 GNOME 桌面設定，將 Vino 的 prompt-enabled（啟用提示）以及 require-encryption（需要加密）都設為 false，讓 VNC 遠端連線不需要經過認證階段。
gsettings set org.gnome.Vino prompt-enabled false
gsettings set org.gnome.Vino require-encryption false
#將你正在用的網路卡加入 VINO 服務
nmcli connection show
#會顯示網路卡的 UUID，把它填入下方的單引號’　‘之間
dconf write /org/gnome/settings-daemon/plugins/sharing/vino-server/enabled-connections "['填入這裏']"
export DISPLAY=:0
```

步驟 8.　打開電腦 VNC 軟體➢ 紅框 1：New connection ➜紅框 2：輸入 100.82.153.59➜紅框 3：輸入代稱(看得懂就好) ➜紅框 4：OK



步驟 9.　回到 VNC 軟體主畫面➢ 紅框 4：點擊設定好的連線 ➜紅框 5：Continue，第二張圖為 VNC 連線後畫面

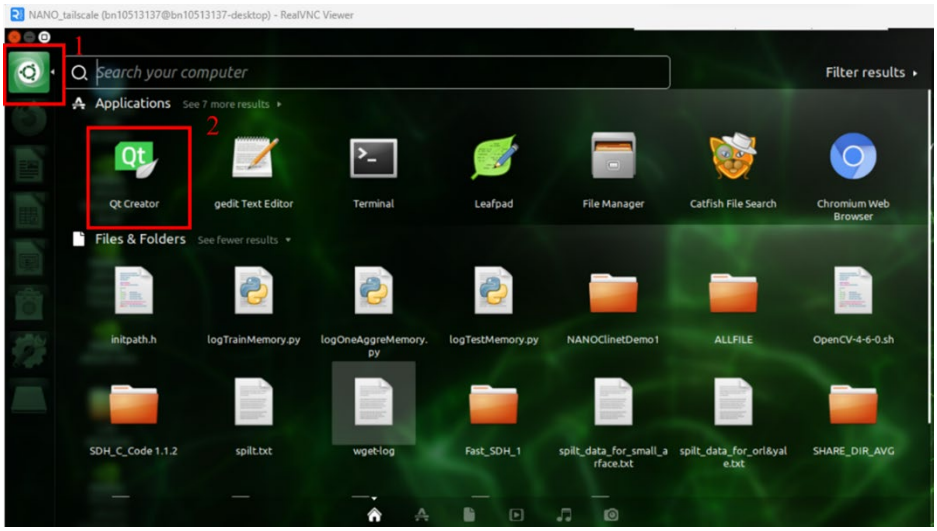




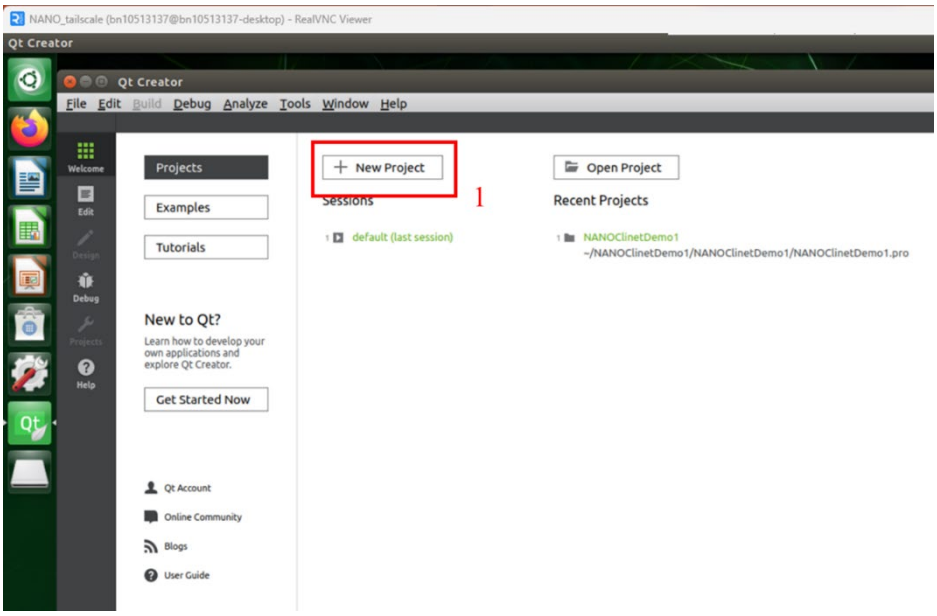
步驟 10. 安裝 Qt – 5 程式

```
# 更新套件
sudo apt-get update
sudo apt-get upgrade
#安裝 Qt 套件
sudo apt-get install -y qt5-default
sudo apt-get install -y qtcreator
```

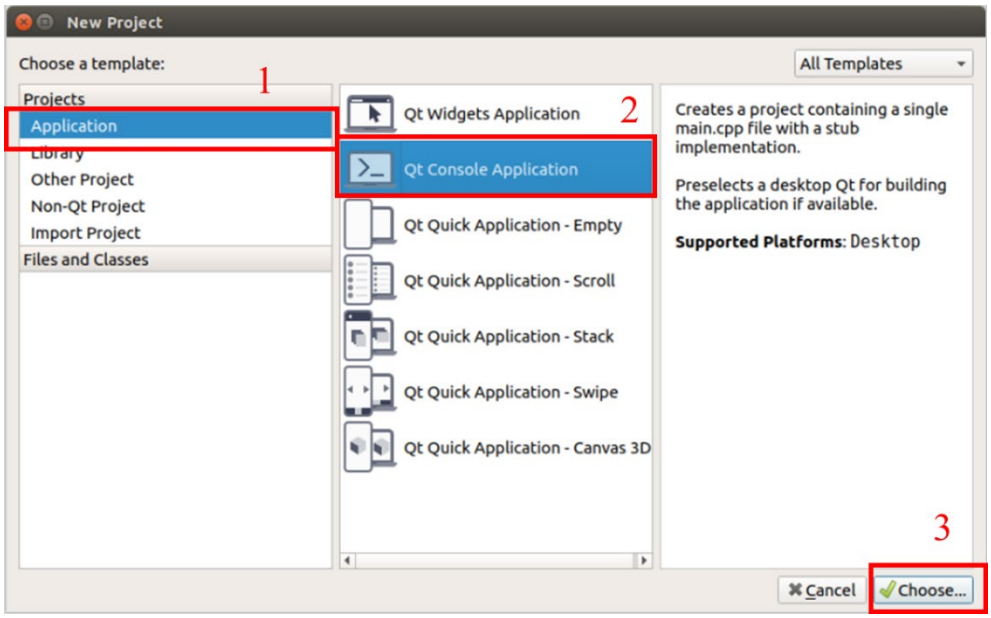
步驟 11. 開啟 Qt – 5 程式的 Qt Creator 編輯器➢紅框 1：點擊裏頭圖像 ➜ 紅框 2：點擊 Qt Creator 編輯器



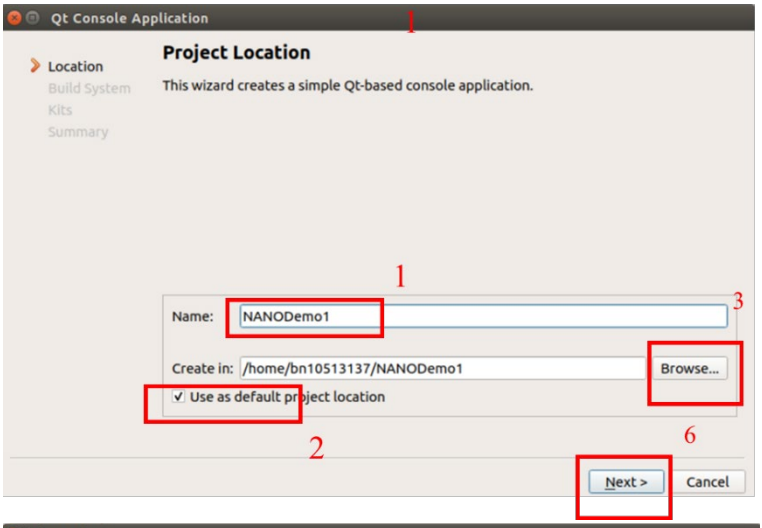
步驟 12. 創建新專案 ➢紅框 1：+ New Project

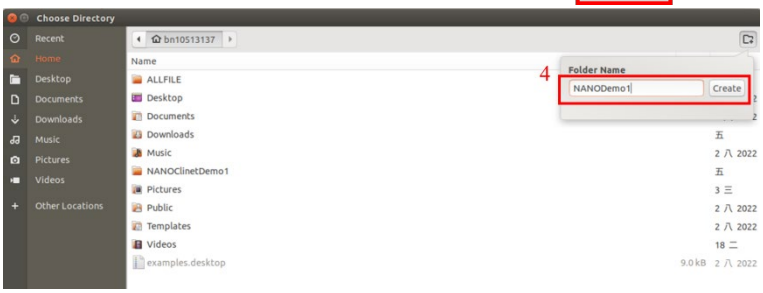


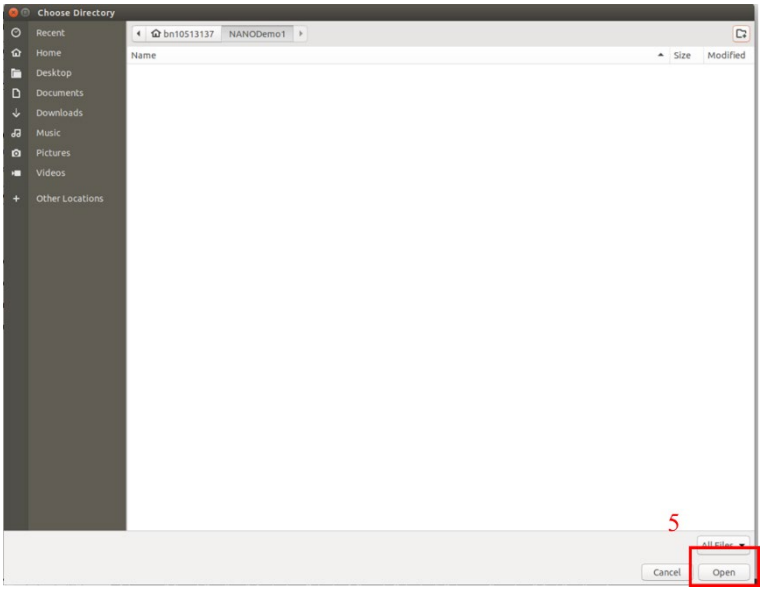
步驟 13. 創建專案 ➢ 紅框 1：選取 Application➜紅框 2：選取 Qt Console Application ➜紅框 3：Choose



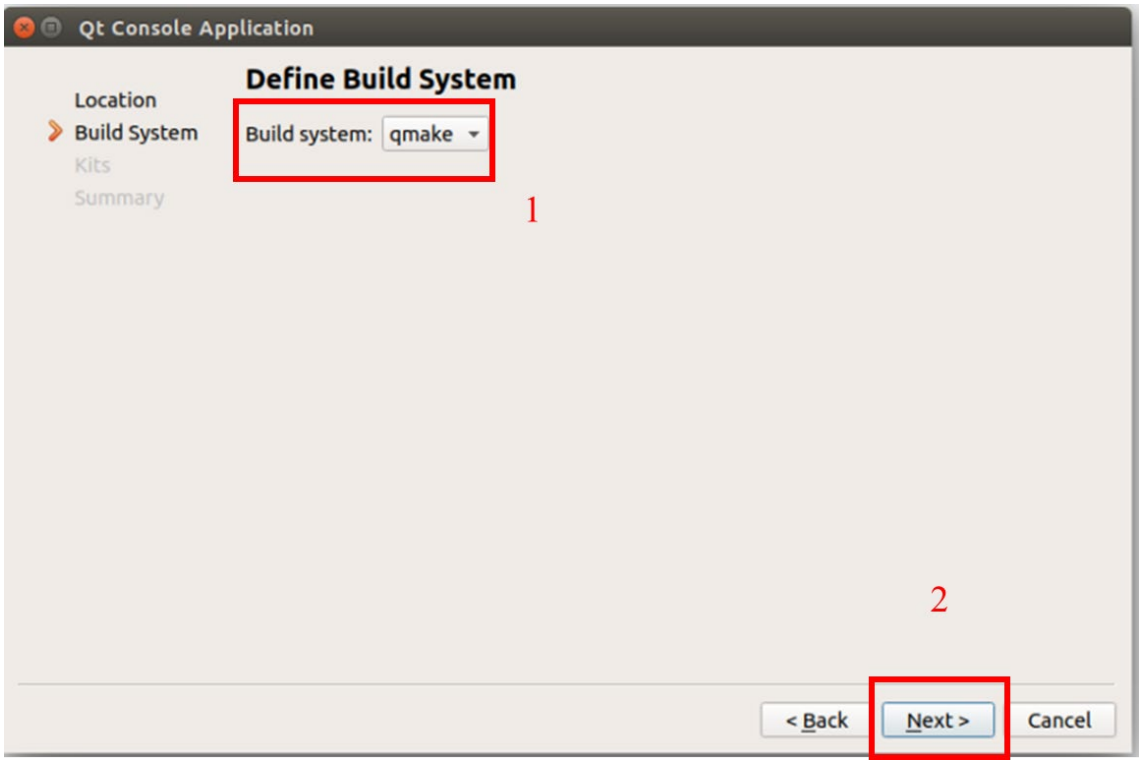
步驟 14. 創建專案，新增放置專案文件的資料夾➢ 紅框 1：新增專案名稱 ➜ 紅框 2：作為預設的專案位置打勾 ➜ 紅框 3：按下 Browse 按鈕 ➜ 紅框 4：新增資料夾，改名為專案名稱 ➜ 紅框 5：到新增資料夾的路徑下按下 Open 按鈕➜ 紅框 6：Next



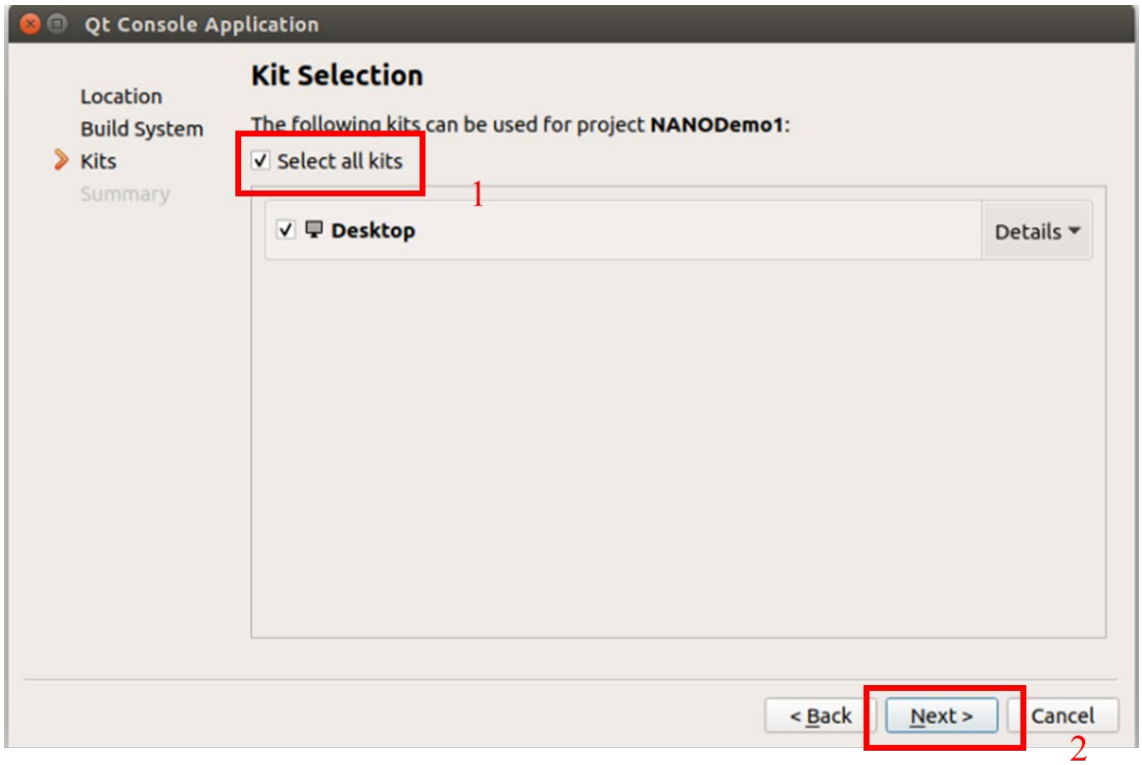




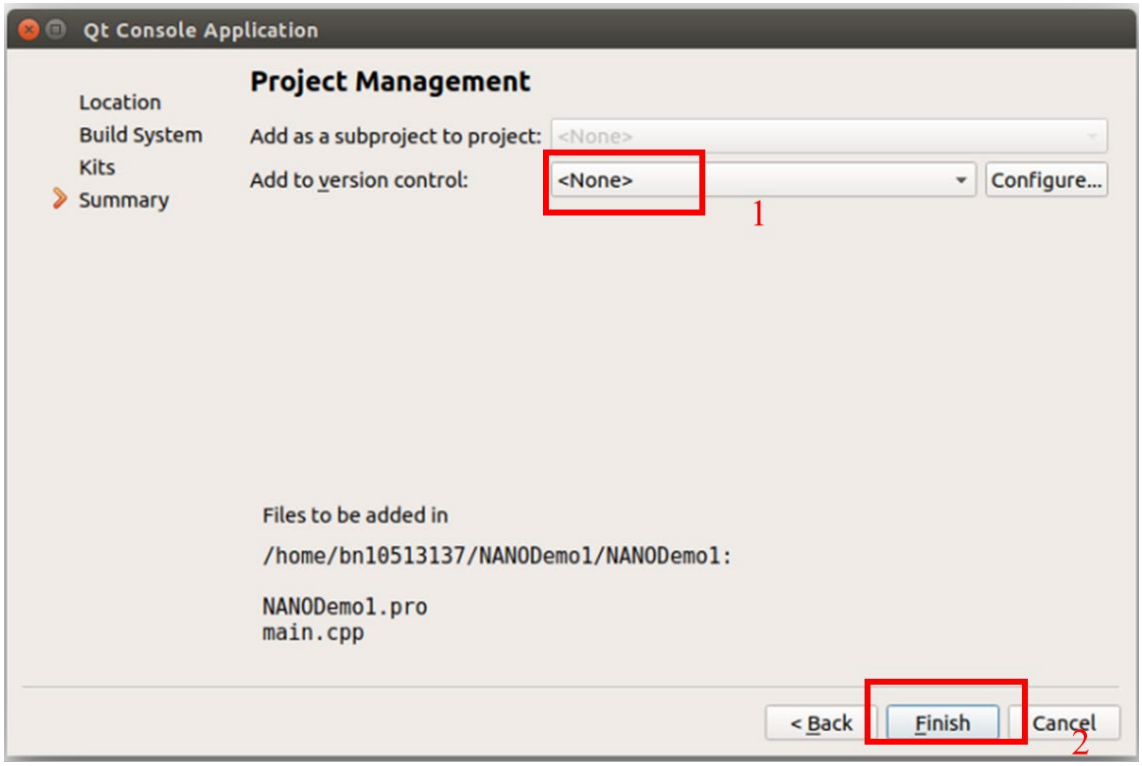
步驟 15. 創建專案 ➢ 紅框 1：選取 qmake ➜ 紅框 2：點擊 Next



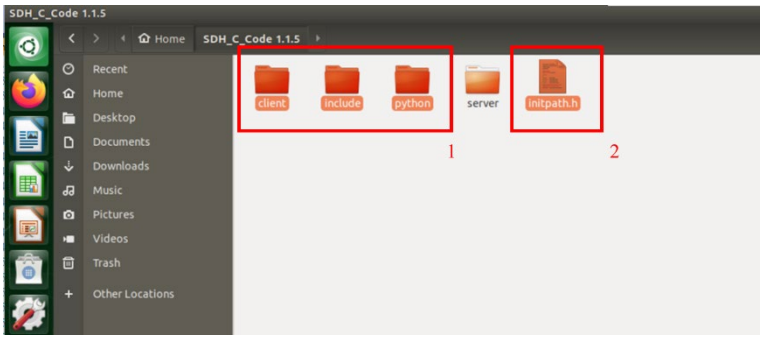
步驟 16. 創建專案 ➢ 紅框 1：勾選 select all kits ➜ 紅框 2：點擊 Next

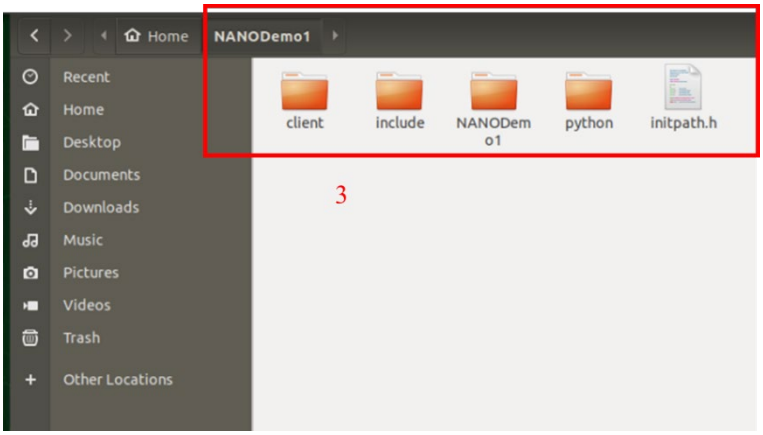


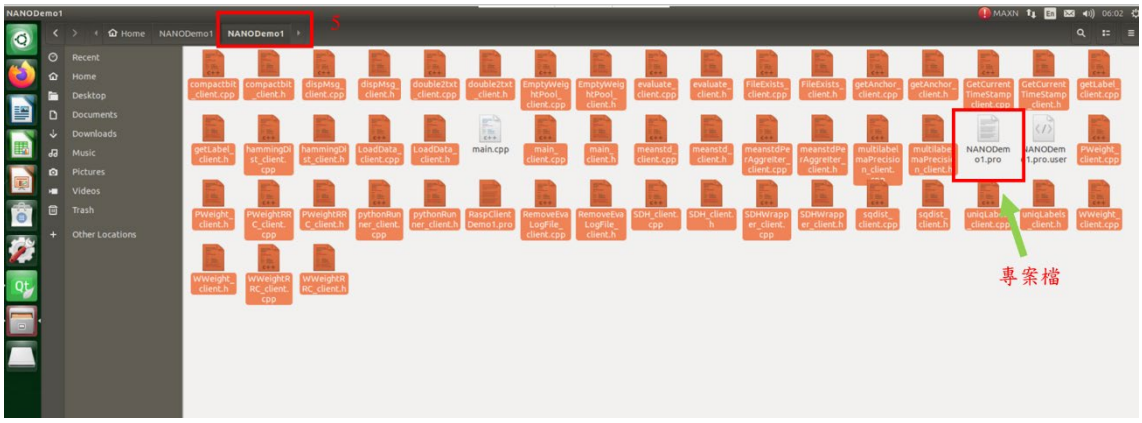
步驟 17. 創建專案 ➢ 紅框 1：勾選 <None> ➜ 紅框 2：點擊 Finish



步驟 18. 放入所需檔案 ➢ 將紅框 1 和紅框 2 的四個資料夾或檔案複製到專案資料夾底下，如紅框 3 所示 ➜ 紅框 4：將 client 資料夾底下的所有檔案複製到紅框 5 放置專案檔的資料夾底下

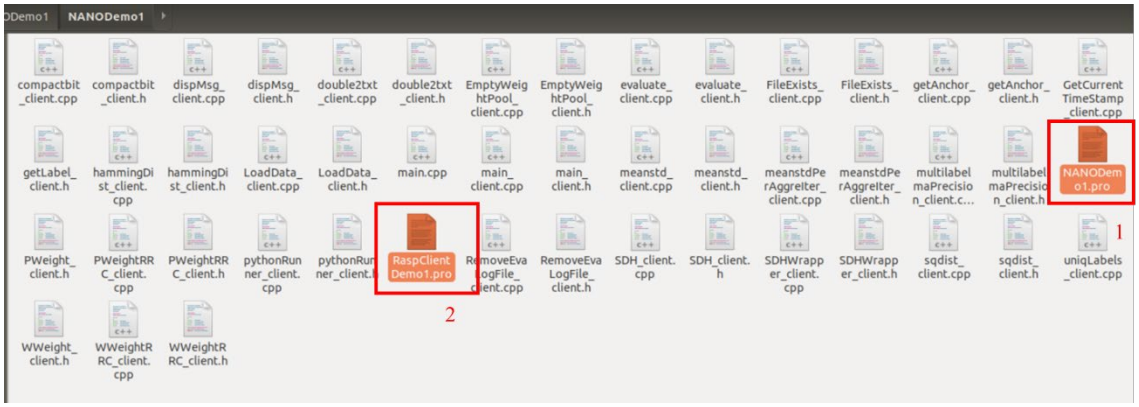




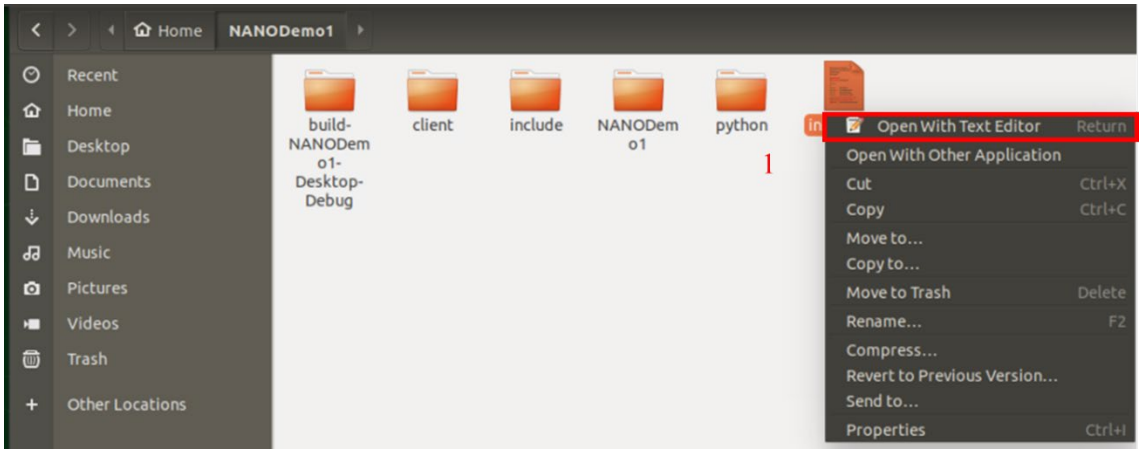


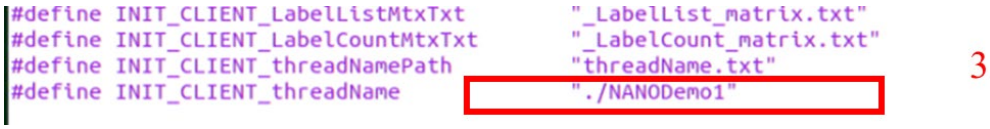
步驟 19. 將 include 函式庫及標頭檔的.pro 改名 ➢ 紅框 1：將此檔案移除

➜ 紅框 2 ：點取右鍵重新命名，名字改為專案檔名(也就是紅框 1 的檔案名)

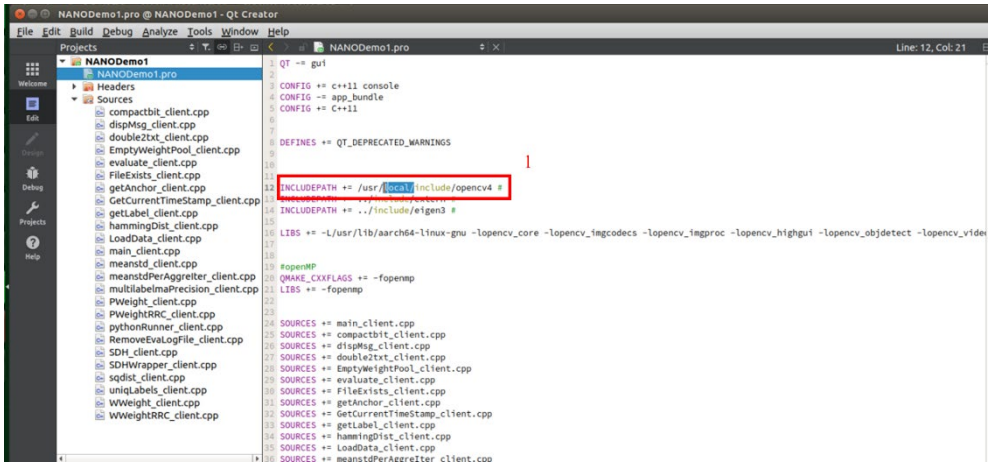


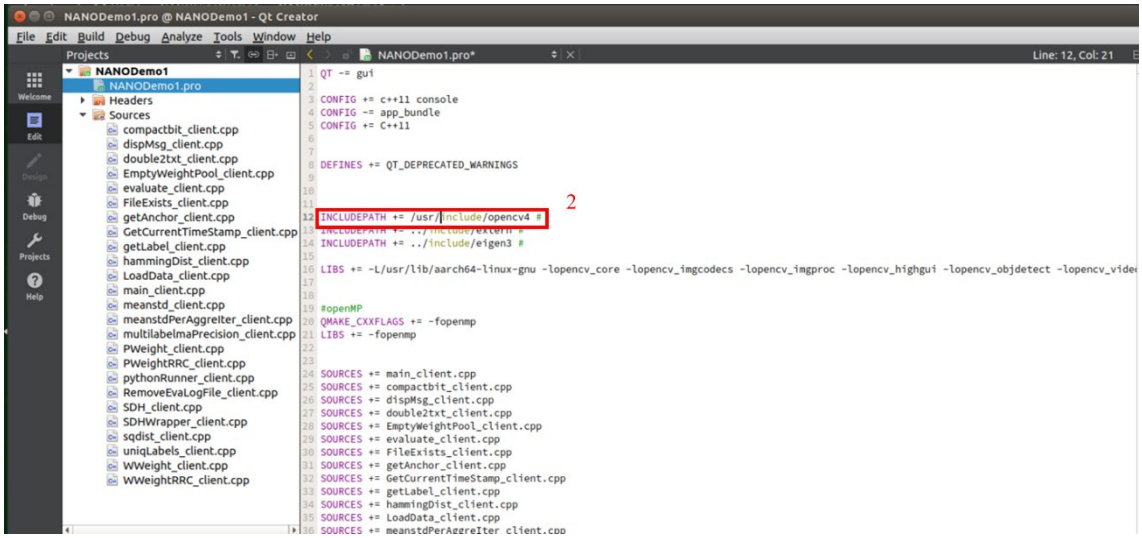
步驟 20. 因測量 Memory 時會去抓執行序名稱，才能獲得該執行序的 Memory 值，所以必須修改路徑檔 initpath.h 中，執行序名稱 ➢ 紅框 1：對路徑檔 initpath.h 點擊右鍵，跳出選單，點選 Open With Text Editor ➜ 紅框 2：將執行序名稱改為./專案檔名 ➜ 紅框 3： 改完結果



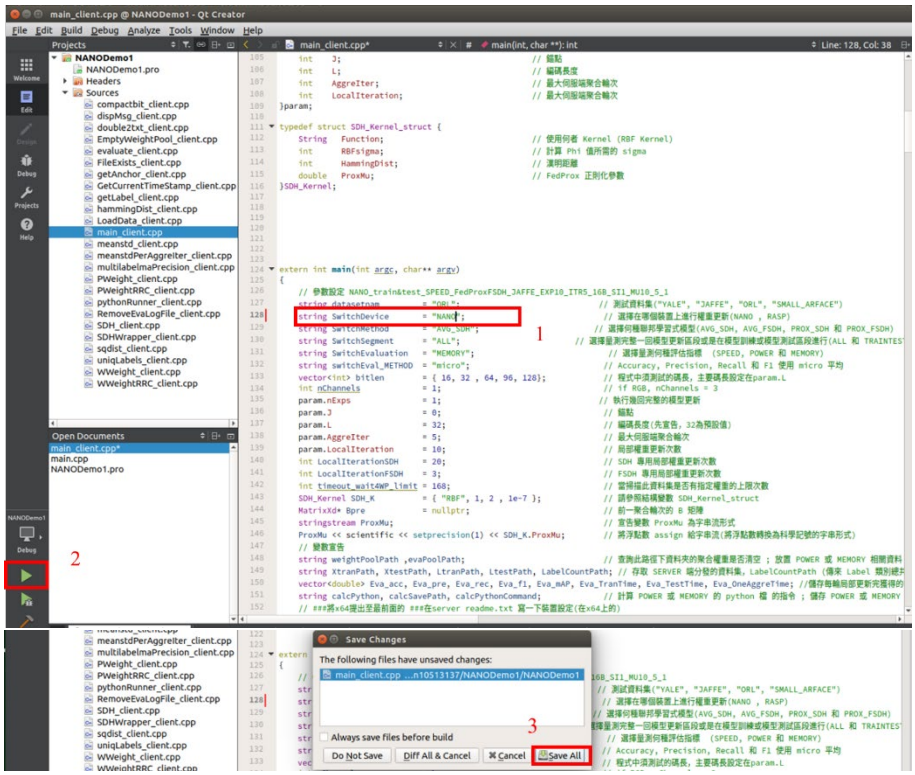


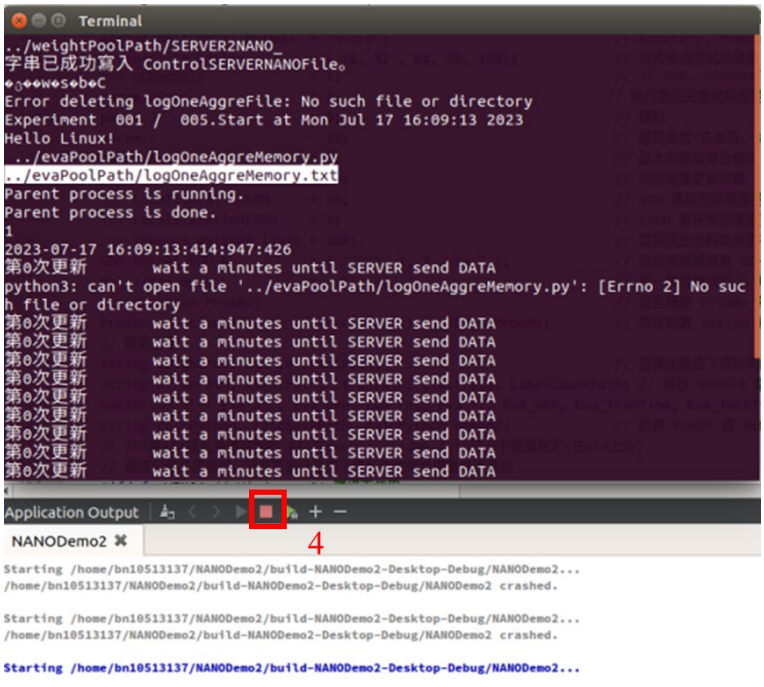
步驟 21. 打開創建的專案檔，點擊專案檔名.pro ➢ 將紅框 1 修改為 紅框 2(/usr/local/opencv4)





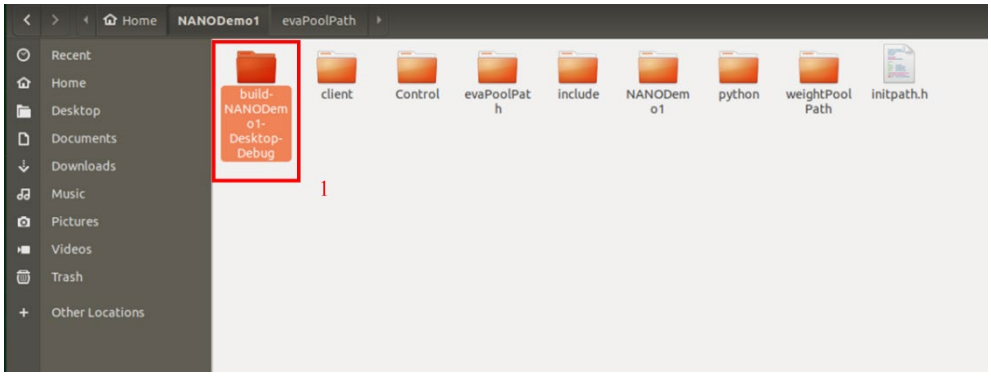
步驟 22. 打開剛創建專案檔➢ 紅框 1：檢查是否為"NANO" ➜ 紅框 2：點擊綠色三角形按鈕，編輯程式➜ 紅框 3：點擊 Save All 等待程式編輯完成➜ 紅框 4：執行過程如第三張圖所示，輸出內容有錯是因為還沒放入檢測 Power 及 Memory 的 Python 檔，所以是正常的，等程式跳出以下畫面就能按紅色正方形按鈕中斷

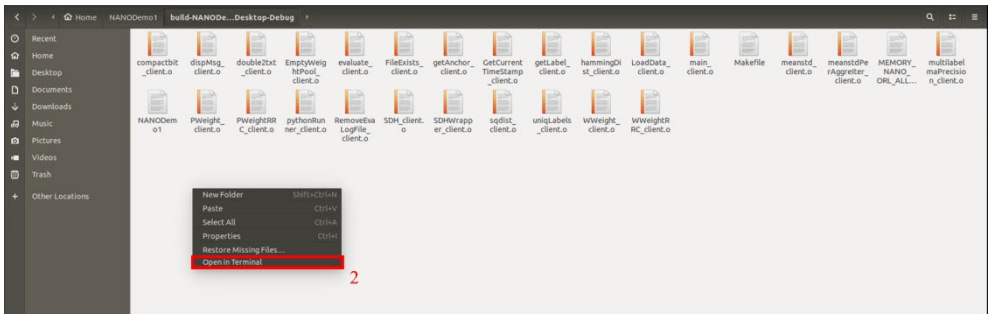




步驟 23. 複製檢測 Power 及 Memory 的 Python 檔➢ 紅框 1：將底下檔案複製到專案資料夾裡的 evaPoolPath (/home/bn10513137/專案名/evaPoolPath)，evaPoolPath 資料夾剛編輯程式時會自動創建不用自己手動創建 ➜ 紅框 3：複製複製到紅框 2 evaPoolPath 資料夾底下，結果如下

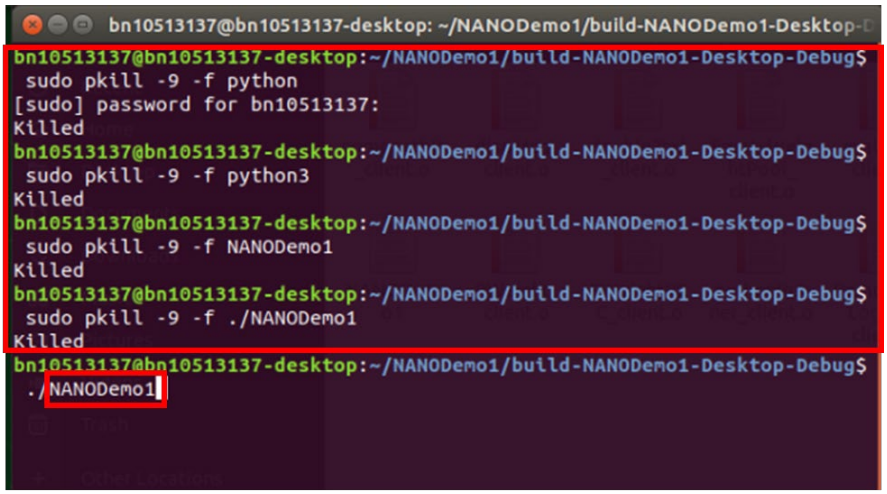
步驟 24. 打開終端機執行程式 ➢ 紅框 1：編輯完程式後會自動創建一個資料夾 build-NANODemo1-unknown-Debug (/home/bn10513137/專案名/build-NANODemo1-unknown-Debug)，裡面有我們編輯完程式的程式執行檔 ➜ 紅框 2：點擊右鍵 open in Terminal





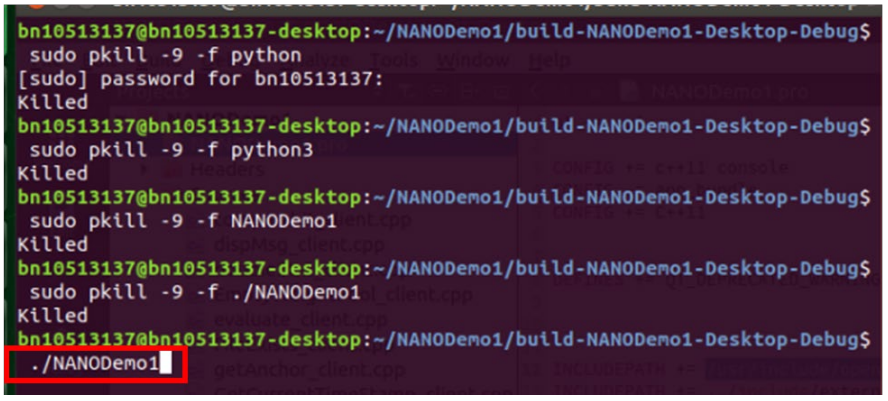
步驟 25. 在終端機上輸入執行命令

1



2

3



- 紅框 1：輸入下面四行中斷指令

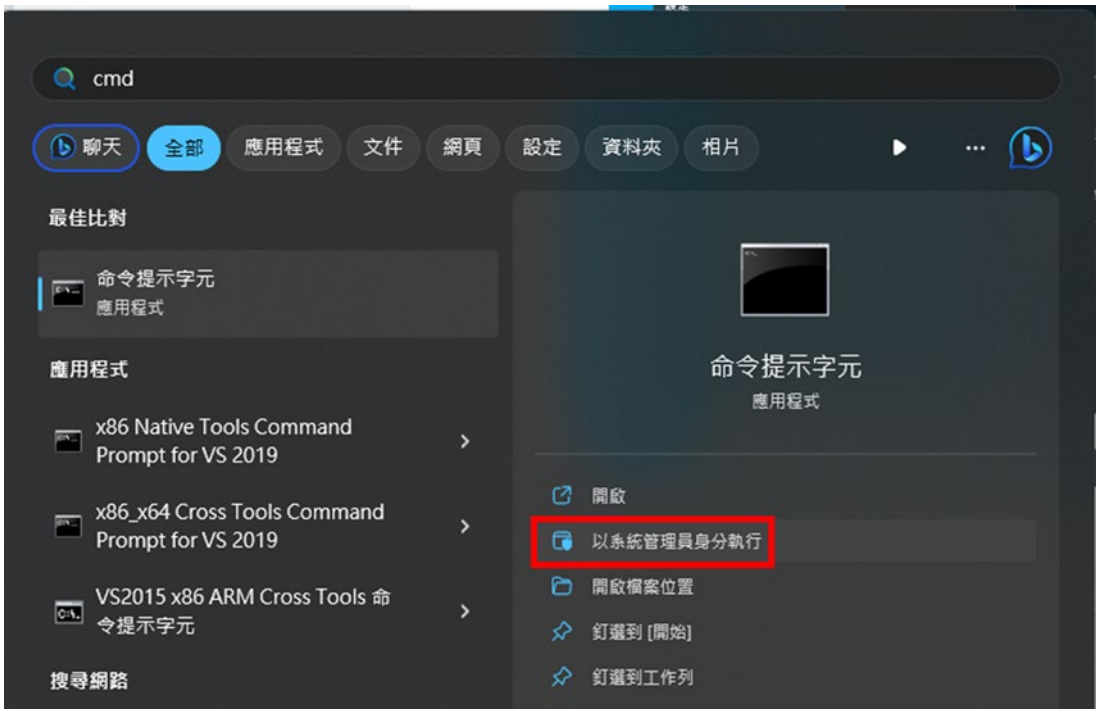
```
# 四行中斷指令用於刪除執行序的，如果程式執行到一半未跑完，有可能程式會在背景執行，尤其是檢測 Power 及 Memory 的 Python 檔，最有可能

sudo pkill -9 -f python

sudo pkill -9 -f python3

sudo pkill -9 -f 專案檔名(例如紅框 2 的 RASPDemo1)

sudo pkill -9 -f ./專案檔名(需在檔名前面加上 ./)
```

- 紅框 3：輸入執行程式指令(但先別按 Enter 執行)

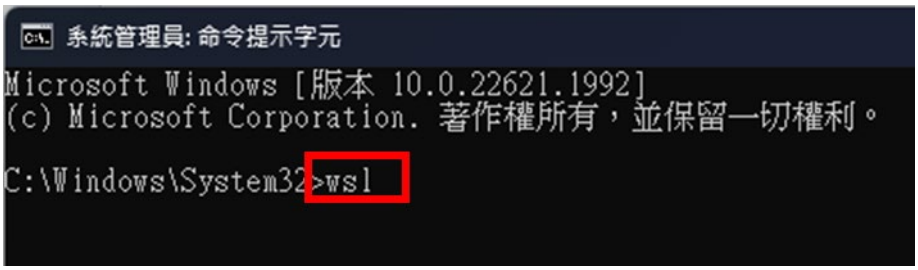
```
# 執行程式執行檔

./專案檔名(需在檔名前面加上 ./)
```

- Win64 Server 端程式

步驟 1.　以系統管理員身份執行 Windows 的命令提示字元(cmd)

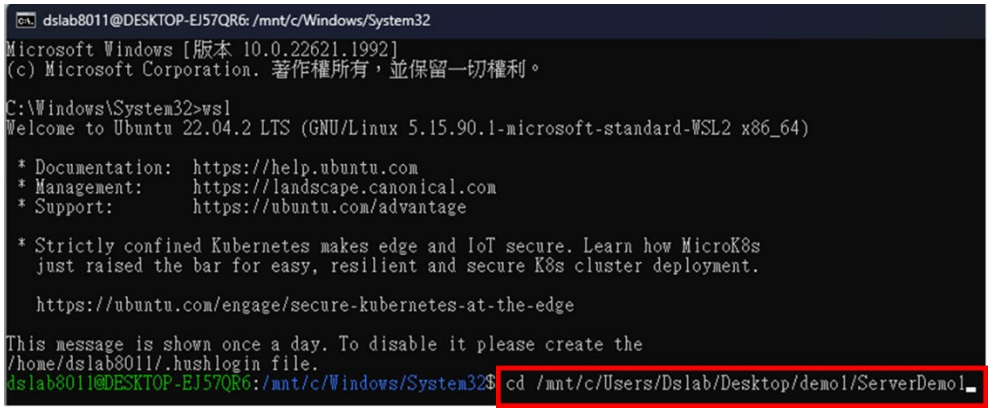



```
# 於 cmd 上開啟 WSL 系統

wsl
```



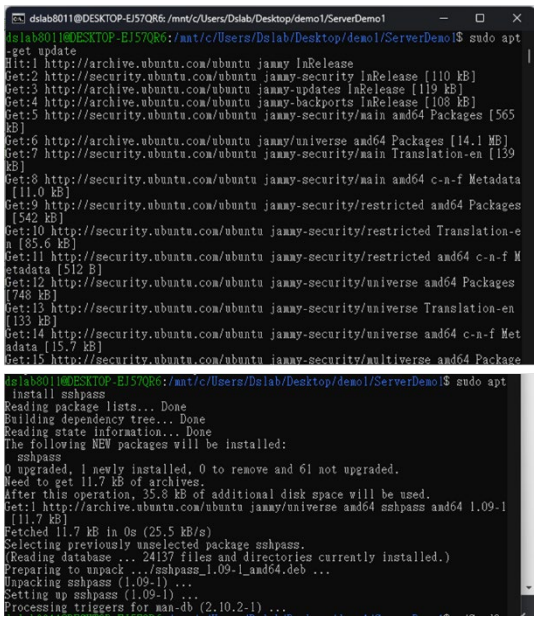
步驟 2.　開啟後可以看到在 Linux 子系統中，Windows 的磁碟機會被掛載到 Linux 作業系統的/mnt 目錄下

```
# 進入到剛創建的 Server 端專案 Server1 資料夾中
# cd /mnt/原來 windows 專案路徑位置
cd /mnt/c/Users/Dslab/Desktop/demo1/ServerDemo1
```



步驟 3.　安裝 sshpass 套件

```
# 先進行更新
sudo apt-get update
# 安裝 sshpass 套件
sudo apt install sshpass
```



步驟 4.　設定 RASP、NANO 和 SERVER 之間透過 SSH 的公鑰私鑰來使用 rsync 指令傳遞更新權重

首先在 server 端輸入指令來產生 key(RASP 已產生不需再做一次)，

dslab8011@100.97.100.48 為附加至公開金鑰檔案結尾以便輕鬆識別的註解，有加沒加沒有差

#100.97.100.48 為 Server 端的 IP，dslab8011 為 Server 端帳號名稱

#100.82.153.59 為 RASP 端的 IP，raspdslab 為 RASP 端帳號名稱

#100.82.153.59 為 NANO 端的 IP ，bn10513137 為 NANO 端帳號名稱

ssh-keygen -t rsa -m PEM -b 4096 -C dslab8011@100.97.100.48

#將公鑰設定放到 CLIENT(RASP 和 NANO)端上

ssh-copy-id -i your_key_path username@server_host

ssh-copy-id -i /home/dslab8011/.ssh/id_rsa raspdslab@100.82.153.59

ssh-copy-id -i /home/dslab8011/.ssh/id_rsa bn10513137@100.66.82.83

```
dslab8011@DESKTOP-EJ57QR6:/mnt/c/Users/Dslab/Desktop/demo1/ServerDemo1$ ssh-keygen -t rsa -m PEM -b 4096 -C TEST2
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dslab8011/.ssh/id_rsa):
/home/dslab8011/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dslab8011/.ssh/id_rsa
Your public key has been saved in /home/dslab8011/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:nmcMowkKWuASe0A2AFeJ3aeCm9z35XAcc6IkouldPk4 TEST2
The key's randomart image is:
+---[RSA 4096]----+
|=+.+.o           |
|o.o o . .        |
|+  .   o         |
|.=. o o . + .    |
|+oo*.o oSo =     |
|o+*...o+o=+      |
|.... +E.o=+      |
|  . ..o .o.      |
|     ...         |
+----[SHA256]-----+
dslab8011@DESKTOP-EJ57QR6:/mnt/c/Users/Dslab/Desktop/demo1/ServerDemo1$ ssh-copy-id -i /home/dslab8011/.ssh/id_rsa raspd
slab@100.82.153.59
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/dslab8011/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
raspdslab@100.82.153.59's password:

Number of key(s) added: 1
```

#傳檔案

#RASP 端

rsync -avzhP /mnt/c/Users/Dslab/Desktop/demo1/weightPoolPath/ raspdslab@100.82.153.59:RASPDemo1/weightPoolPath/

#NANO 端

rsync -avzhP /mnt/c/Users/Dslab/Desktop/demo1/weightPoolPath/ bn10513137@100.66.82.83:NANODemo1/weightPoolPath/

#免密傳檔案(其中指令中間的 6517 為 RASP 和 NANO 裝置的密碼，每次用此 key 登入都需要輸入因此用 sshpass 傳遞)

sshpass -p 6517 rsync -avzhP /mnt/c/Users/Dslab/Desktop/demo1/weightPoolPath/ raspdslab@100.82.153.59:RASPDemo1/weightPoolPath/

sshpass -p 6517 rsync -avzhP /mnt/c/Users/Dslab/Desktop/demo1/weightPoolPath/ bn10513137@100.66.82.83:/NANODemo1/weightPoolPath/

步驟 5.　打開 Server 端的路徑檔 initpath.h

- 紅框 1：Server 端 weightPoolPath 的路徑位置(/mnt/weightPoolPath 路徑)
- 紅框 2：Rasp 上 weightPoolPath 的路徑位置

(主機名稱@ Rasp 端 IP: /weightPoolPath 路徑)

- 紅框 3：Nano 上 weightPoolPath 的路徑位置

(主機名稱@ Nano 端 IP: /weightPoolPath 路徑)

以下為範例，請依照實際位置去設置

紅框 1："/mnt/c/Users/Dslab/Desktop/demo1/weightPoolPath/"

紅框 2："raspdslab@100.82.153.59:RaspClientDemo1/weightPoolPath/"

紅框 3："bn10513137@100.66.82.83:NANOClinetDemo1/weightPoolPath/"

步驟 6.　打開 Server 端的 SendOrReturnWeight.sh(這個 shell code 檔主要為取 SEND_FILE_CTRL_MSG.txt 裡不同的控制指令傳遞權重資料)➢ 紅框 1、2 和 3：這串密碼需改為 RASP 和 NANO 所設的系統密碼

initpath.h　SendOrReturnWeight.sh

```
67
68  # 如果文件存在且不為空
69  while true; do
70      if [ -f SEND_FILE_CTRL_MSG.txt ]; then
71          echo "找到檔案"
72          # 使用 grep 命令搜索文件
73          if grep $searchString1 SEND_FILE_CTRL_MSG.txt; then      #CTRL_MSG_1
74              echo "SERVER傳DATA到NODE端"
75              echo "傳送檔案"
76              sshpass -p 6517 rsync -avzhP $SERVER_fixed $NANO_fixed
77              sshpass -p 6517 rsync -avzhP $SERVER_fixed $RASP_fixed
78              sleep 2                                              1
79              > SEND_FILE_CTRL_MSG.txt # 清除檔案內容
80          elif grep $searchString2 SEND_FILE_CTRL_MSG.txt; then    #CTRL_MSG_2
81              echo "NODE傳WEIGHT到SERVER端"
82              echo "傳回檔案"
83              sshpass -p 6517 rsync -avzhP $NANO_fixed $SERVER_fixed
84              sshpass -p 6517 rsync -avzhP $RASP_fixed $SERVER_fixed
85              sleep 2                                              2
86              > SEND_FILE_CTRL_MSG.txt # 清除檔案內容
87          elif grep $search_string_3 SEND_FILE_CTRL_MSG.txt; then  #CTRL_MSG_3
88              echo "SERVER端傳聚合WEIGHT到NODE端"
89              echo "傳送聚合檔案"
90              sshpass -p 6517 rsync -avzhP $SERVER_fixed $NANO_fixed
91              sshpass -p 6517 rsync -avzhP $SERVER_fixed $RASP_fixed
92              sleep 2                                              3
93              > SEND_FILE_CTRL_MSG.txt # 清除檔案內容
94          else
95              echo "字符串未找到"
96          fi
97      else
98          echo "文件不存在或為空"
99      fi
100     sleep 1
101 done
```

2.3 執行聯邦式雜湊學習模型

- 進行最終檢查

步驟 1. 檢查 SERVER 端、RASP 端和 NANO 端專案夾主程式的參數是否一致

➢ 紅框 1：檢查三個專案中 datasetnam(數據集)變數之值是否一致

➢ 黃框 2：檢查三個專案中 nChannel、param.nExps、param.J、param.L 和 param.AggreIter 變數之值是否一致

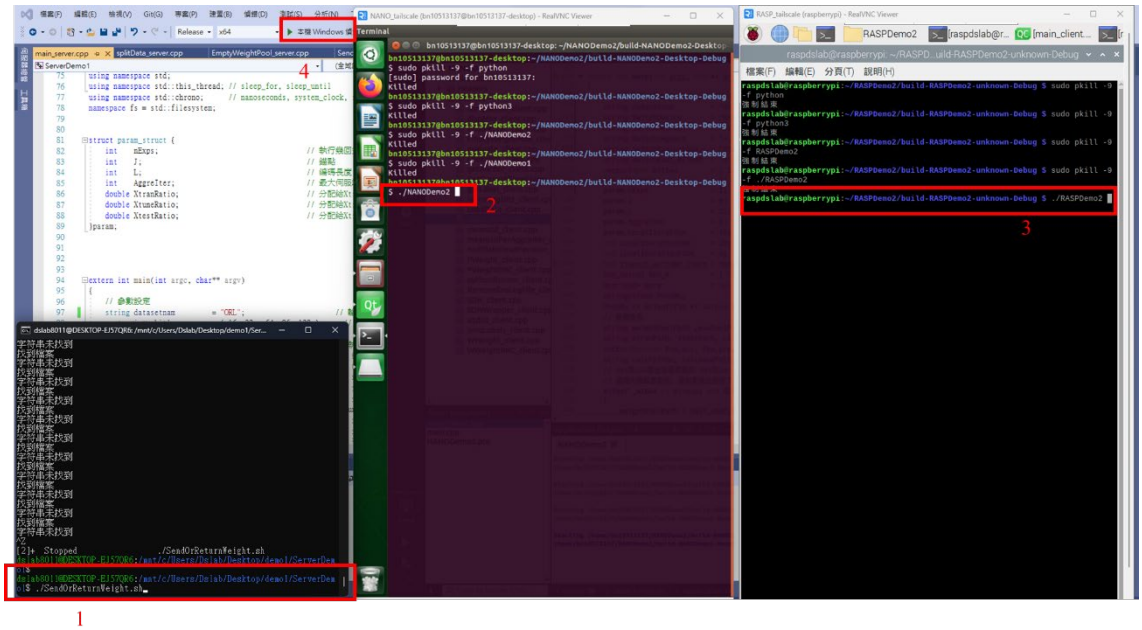
➢ 藍框 3：檢查專案 CLIENT1 和 CLIENT2 SwitchMethod、SwitchSegment、SwitchEvaluation 和 SwitchEva_METHOD 變數之值是否一致

步驟 2.　檢查專案 SERVER1 裡的資料集路徑是否正確

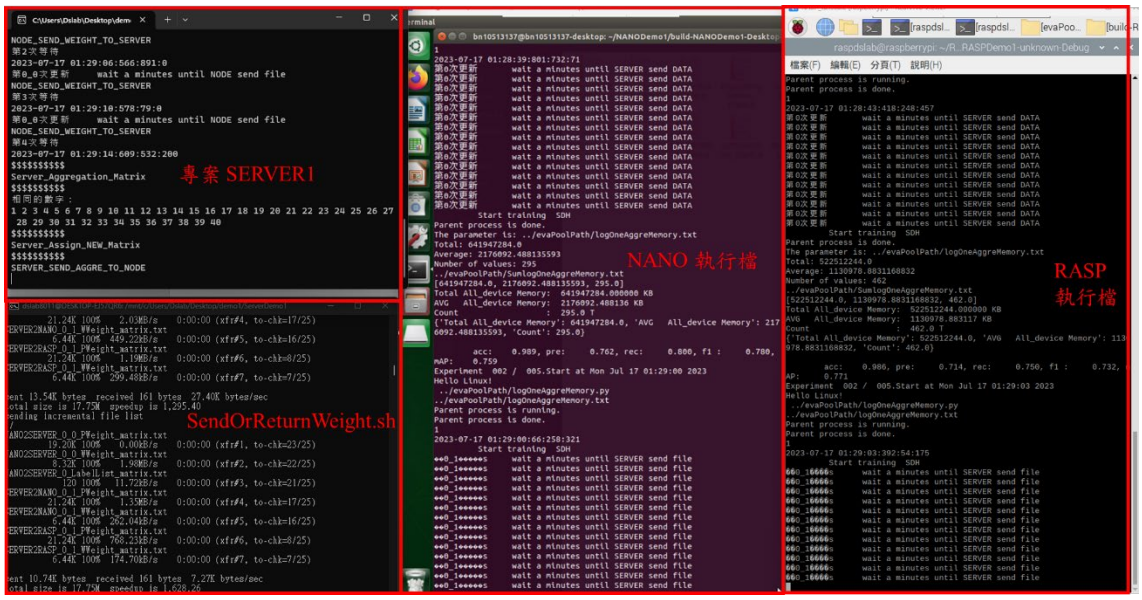
```
150
151  // 取用資料集(請看成是第 30 行)
152  // YALE
153  if (datasetnam.compare("YALE") == 0)
154  {
155      infilepath_x_lin_data = "../Dataset/yale64x64_data_x_lin.txt";
156      infilepath_t_data     = "../Dataset/yale64x64_data_t.txt";
157  }
158  // JAFFE
159  else if (datasetnam.compare("JAFFE") == 0)
160  {
161      infilepath_x_lin_data = "..//Dataset/jaffe64x64_data_x_lin.txt";
162      infilepath_t_data     = "../Dataset/jaffe64x64_data_t.txt";
163  }
164  // ORL
165  else if (datasetnam.compare("ORL") == 0)
166  {
167      infilepath_x_lin_data = "../Dataset/orl64x64_data_x_lin.txt";
168      infilepath_t_data     = "../Dataset/orl64x64_data_t.txt";
169  }
170  // SMALL_ARFACE
171  else if (datasetnam.compare("SMALL_ARFACE") == 0)
172  {
173      infilepath_x_lin_data = "../Dataset/SMALL_2340_ARface64x64_data_x_lin.txt";
174      infilepath_t_data     = "../Dataset/SMALL_2340_ARface64x64_data_t.txt";
175  }
```

步驟 3.　需要同時運作四個程式分別為專案 SERVER1、RASP 執行檔和 NANO 執行檔和傳遞更新權重的 SendOrReturnWeight.sh，開啟他們各自的資料夾

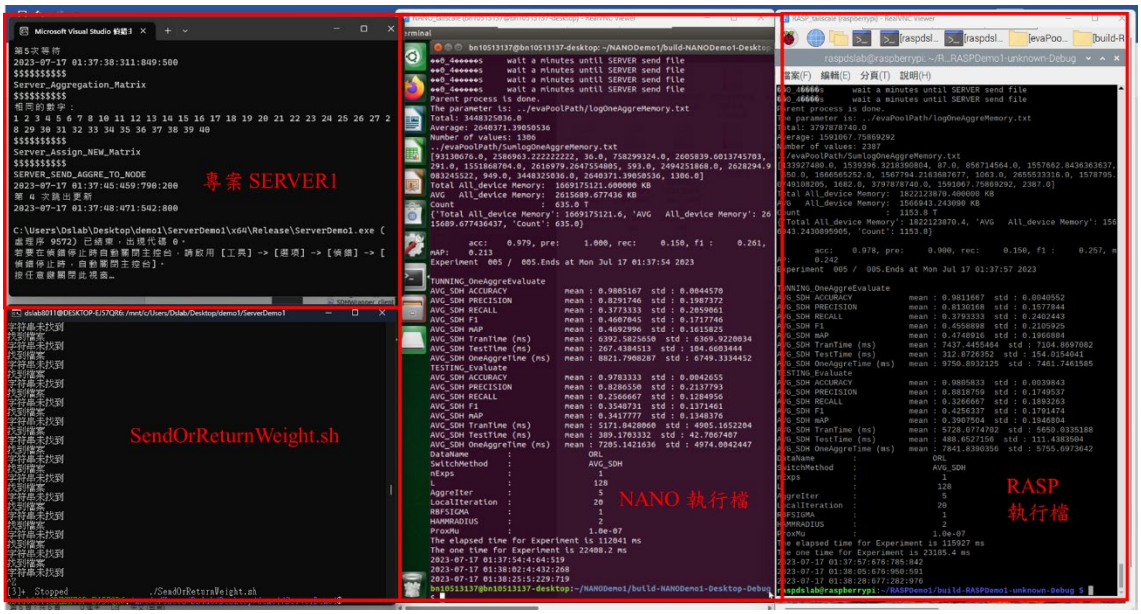
步驟 4.　依序執行紅框 1：SendOrReturnWeight.sh、紅框 2：NANO 執行檔、紅框 3：RASP 執行檔和紅框 4：專案 SERVER1 (最後一個一定要是專案 SERVER1 不然會出錯)



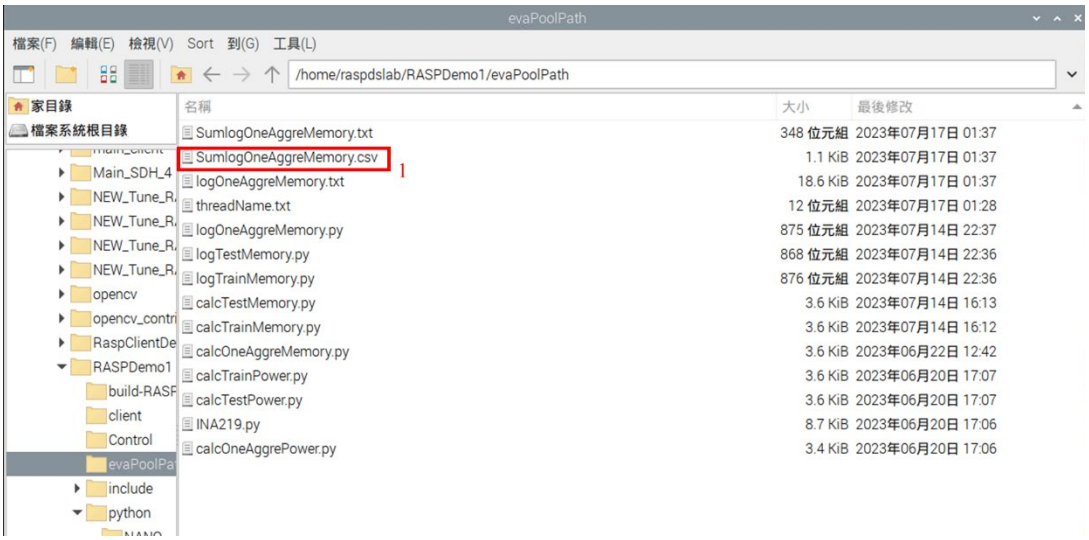
步驟 5.　程式執行中畫面

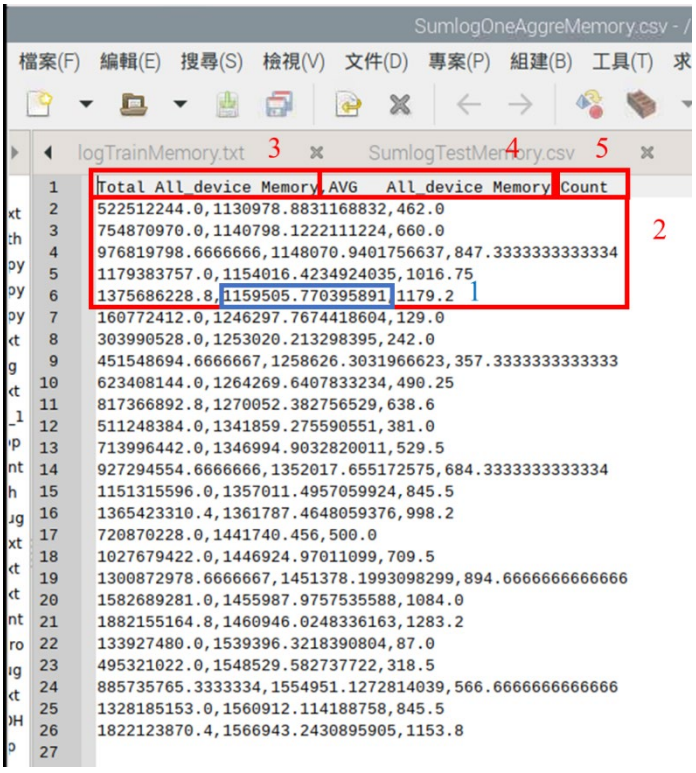


步驟 6.　程式執行結果如下

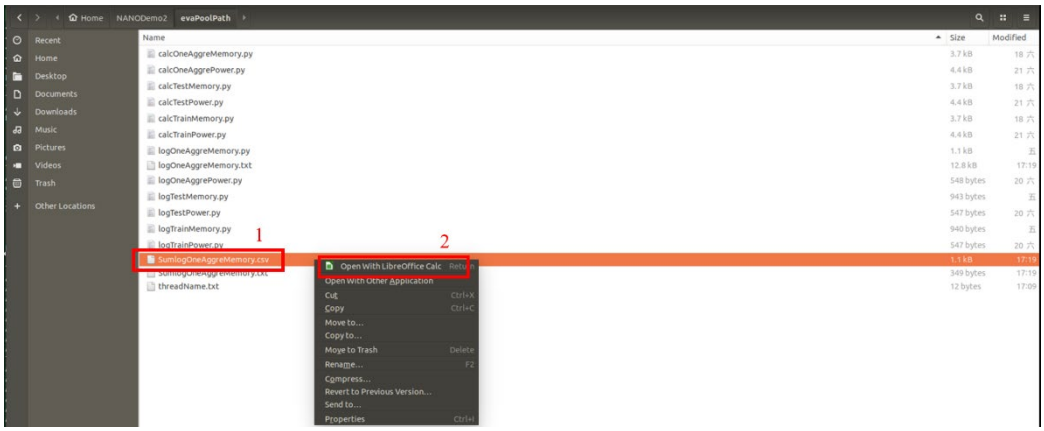


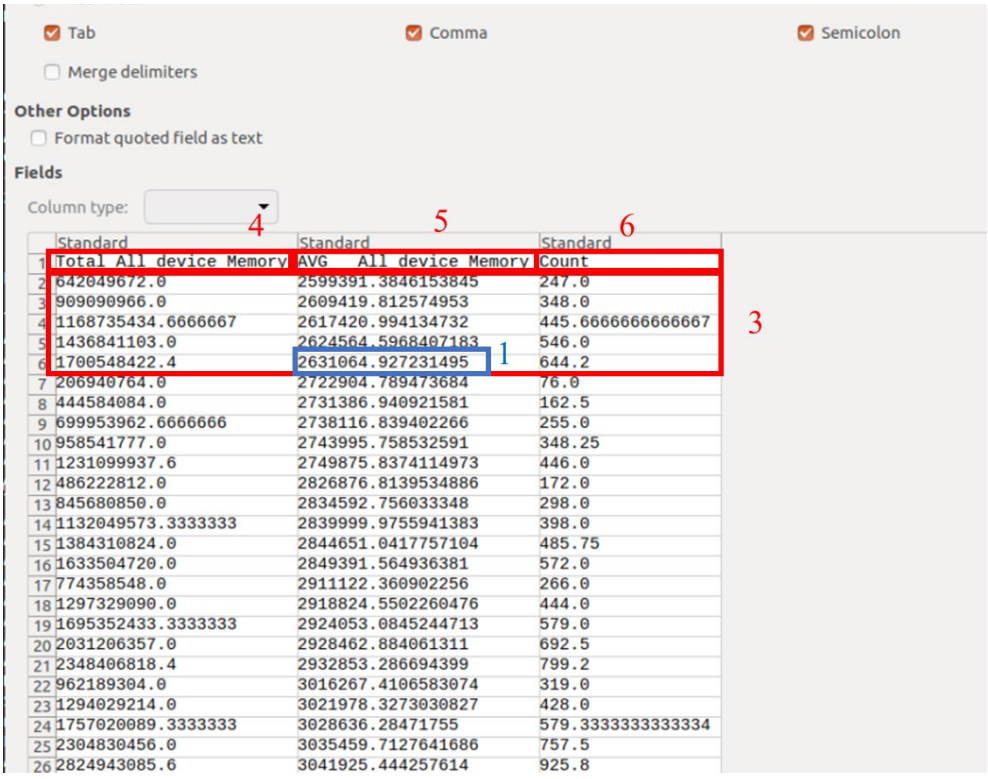
步驟 7.　測量數據(Memory)可以至 RASP 帳號名/專案檔名稱/ evaPoolPath (/home/raspdslab/RASPDemo1/evaPoolPath) 尋找，例如這次測的是區段為 ALL 及評估為 Memory➢ 紅框 1：為此次 RASP 端產生的最終結果➔ 紅框 2：每個碼長都跑五回，所以 16bits 的平均占用記憶體量為 1159.505Megabyte(藍框 1)➔ 紅框 3：每一回累計到的占用記憶體量➔ 紅框 4：每一回的平均占用記憶體量➔ 紅框 5：從第一回計次到第五回結束，記錄到 Memory 的總次數。





步驟 8.　而 NANO 也是到相同位置尋找測量數據(Memory) NANO 帳號名/專案檔名稱/ evaPoolPath (/home/bn10513137/NANODemo1/evaPoolPath) 尋找，例如這次測的是區段為 ALL 及評估為 Memory➢紅框 1：為此次 NANO 端產生的最終結果，點擊右鍵➔ 紅框 2：點擊 Open With LibreOffice Calc➔ 紅框 3：每個碼長都跑五回，所以 16bits 的平均占用記憶體量為 2631.064Megabyte(藍框 1)➔ 紅框 4：每一回累計到的占用記憶體量➔ 紅框 5：每一回的平均占用記憶體量➔ 紅框 6：從第一回計次到第五回結束，記錄到 Memory 的總次數。

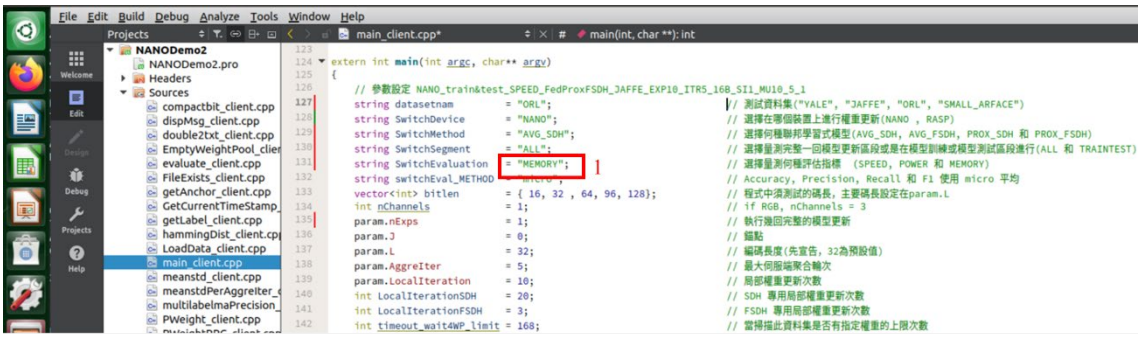


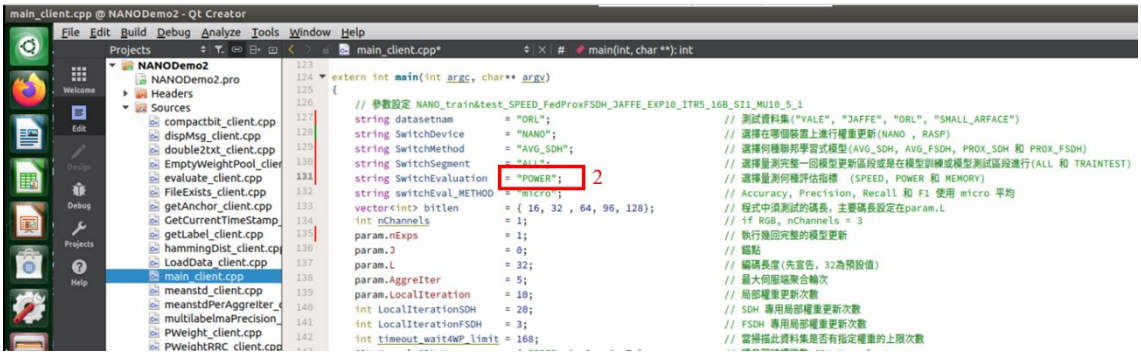


	Standard	Standard	Standard
1	Total_All_device_Memory	AVG_All_device_Memory	Count
2	642049672.0	2599391.3846153845	247.0
3	909090966.0	2609419.812574953	348.0
4	1168735434.6666667	2617420.994134732	445.6666666666667
5	1436841103.0	2624564.5968407183	546.0
6	1700548422.4	2631064.927231495	644.2
7	206940764.0	2722904.789473684	76.0
8	444584084.0	2731386.940921581	162.5
9	699953962.6666666	2738116.839402266	255.0
10	958541777.0	2743995.758532591	348.25
11	1231099937.6	2749875.8374114973	446.0
12	486222812.0	2826876.8139534886	172.0
13	845680850.0	2834592.756033348	298.0
14	1132049573.3333333	2839999.9755941383	398.0
15	1384310824.0	2844651.0417757104	485.75
16	1633504720.0	2849391.564936381	572.0
17	774358548.0	2911122.360902256	266.0
18	1297329090.0	2918824.5502260476	444.0
19	1695352433.3333333	2924053.0845244713	579.0
20	2031206357.0	2928462.884061311	692.5
21	2348406818.4	2932853.286694399	799.2
22	962189304.0	3016267.4106583074	319.0
23	1294029214.0	3021978.3273030827	428.0
24	1757020089.3333333	3028636.28471755	579.3333333333334
25	2304830456.0	3035459.7127641686	757.5
26	2824943085.6	3041925.444257614	925.8

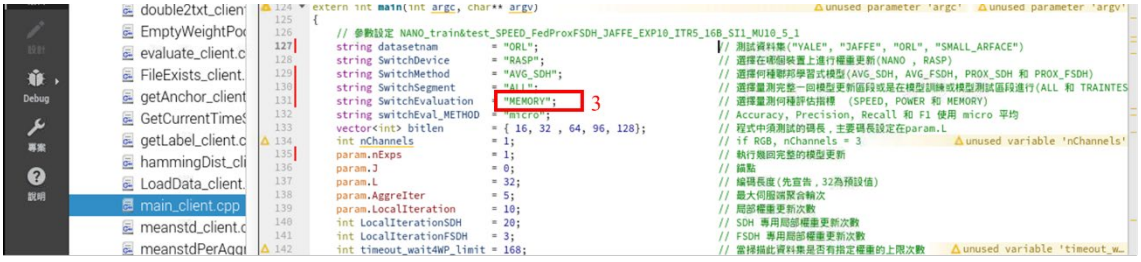
步驟 9.　如果要測量 Power 則需要回 Qt – 5 專案檔裡更改評估方法，並重新編譯➢在 NANO 專案將紅框 1(Memory)改為紅框 2(Power)，在 RASP 專案將紅框 1(Memory)改為紅框 2 (Power)。

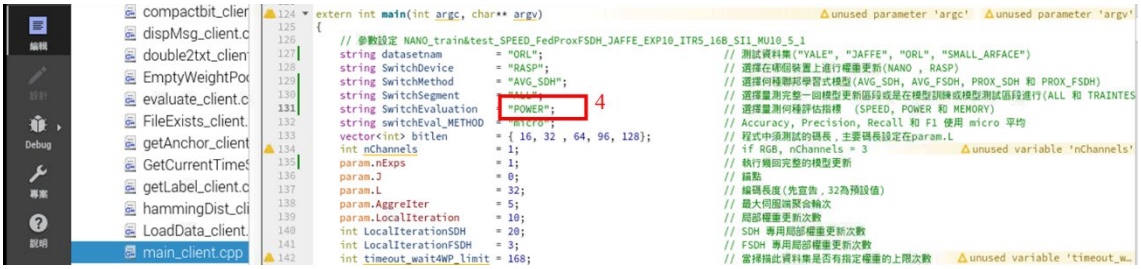
NANO





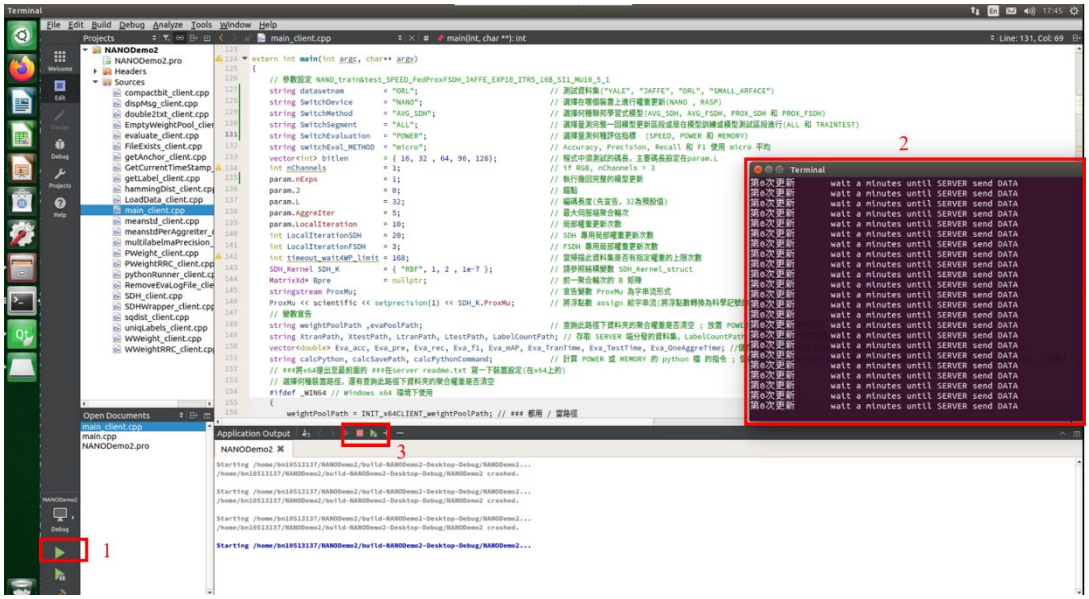
RASP



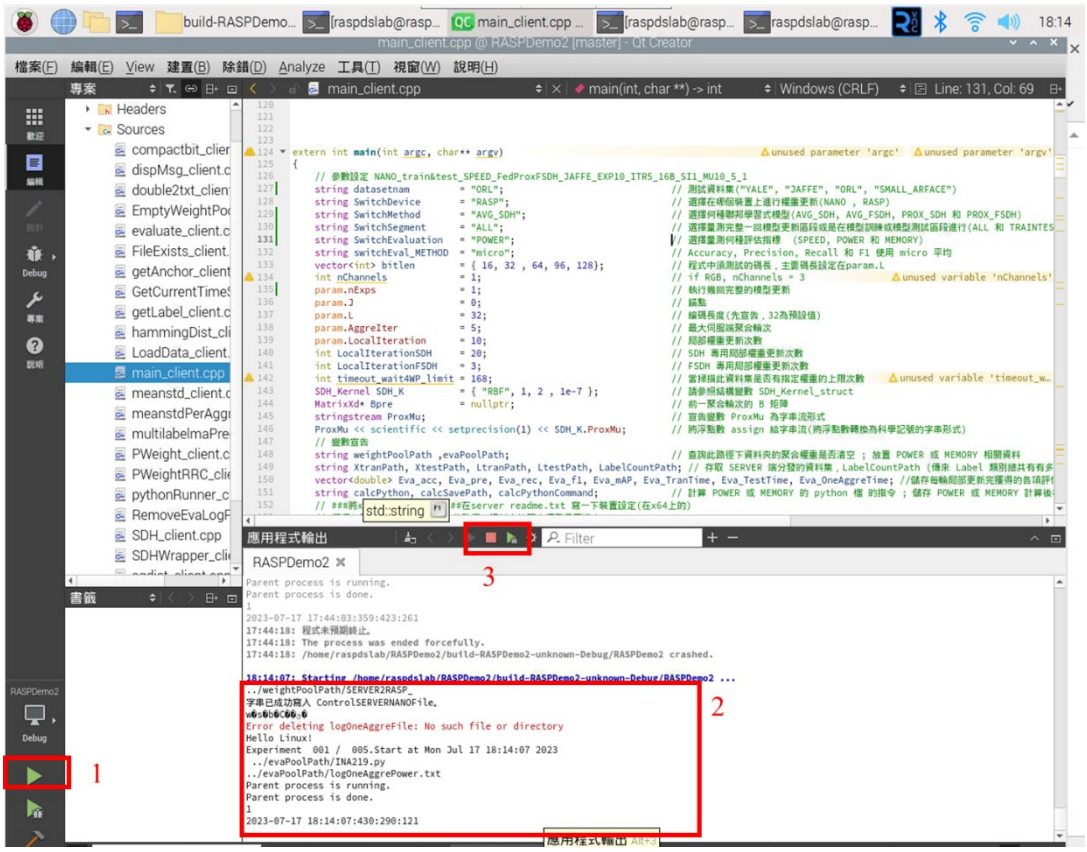


步驟 10. 將 NANO 專案和 RASP 專案進行編譯➢紅框 1：分別按下 NANO 專案和 RASP 專案的編譯按鈕➔ 紅框 2：會各自輸出執行結果➔ 紅框 3：等到像紅框 2 畫面的狀態時就能按下停止按鈕。

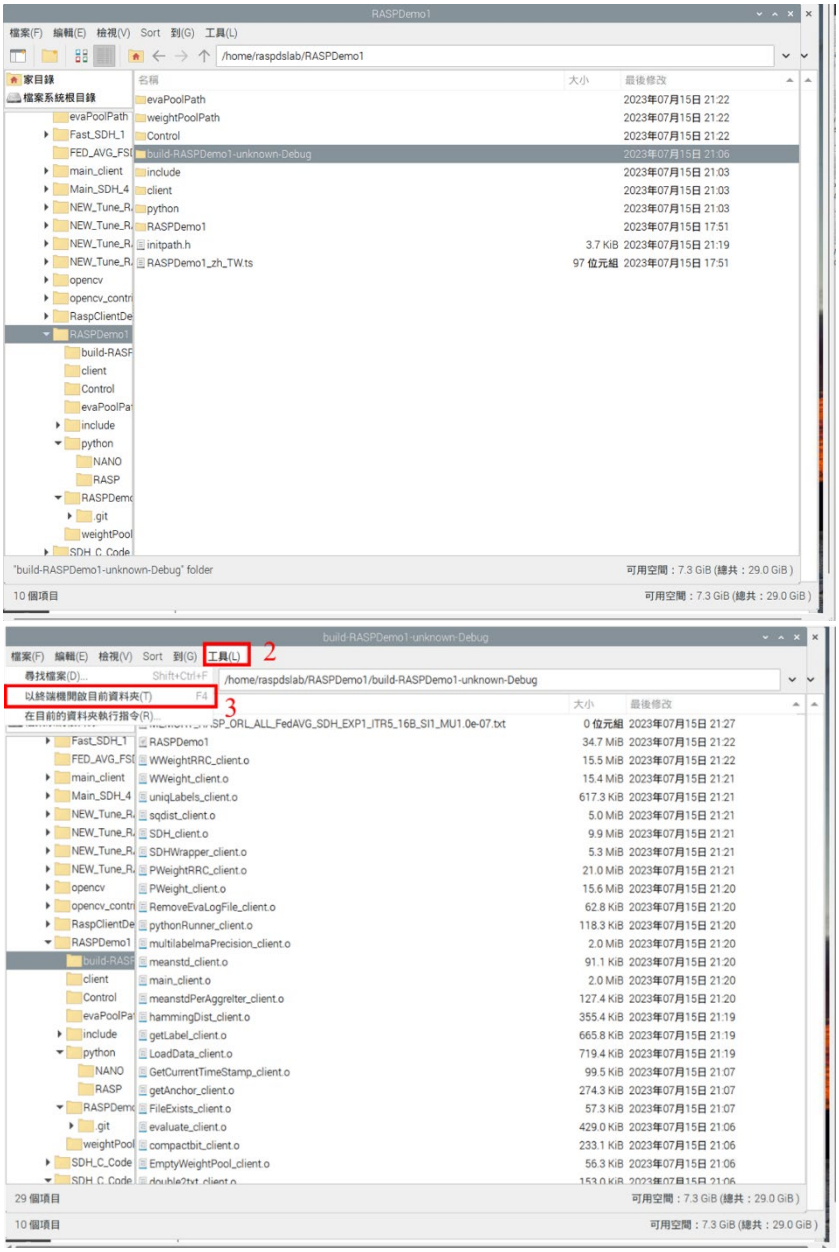
NANO



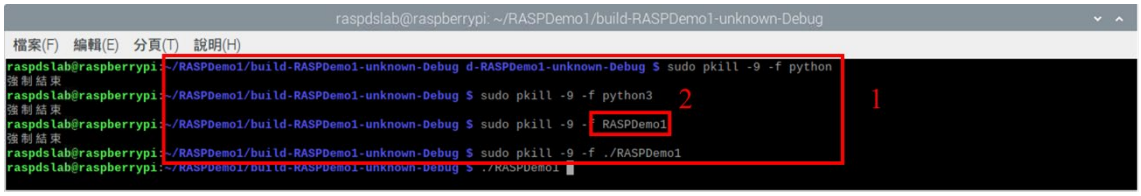
RASP



步驟 11. 打開 RASP 終端機執行程式 ➢ 紅框 1：到編輯完程式後自動創建的一個資料夾中 build-RASPDemo1-unknown-Debug (/home/raspdslab/專案名/build-RASPDemo1-unknown-Debug)，裡面有我們編輯完的程式執行檔➜紅框 2：點擊工具➜ 紅框 3：以終端機開啟目前資料夾

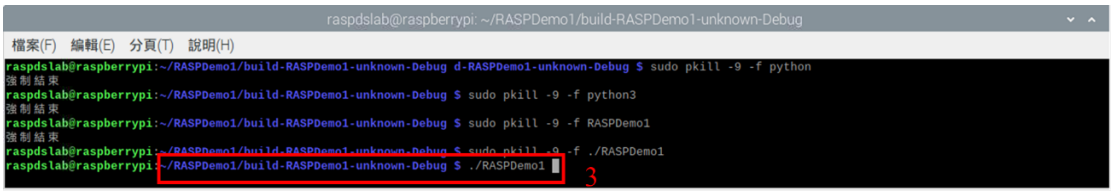


步驟 12. 在 RASP 終端機上輸入執行命令



➢ 紅框 1：一樣輸入下面四行中斷指令

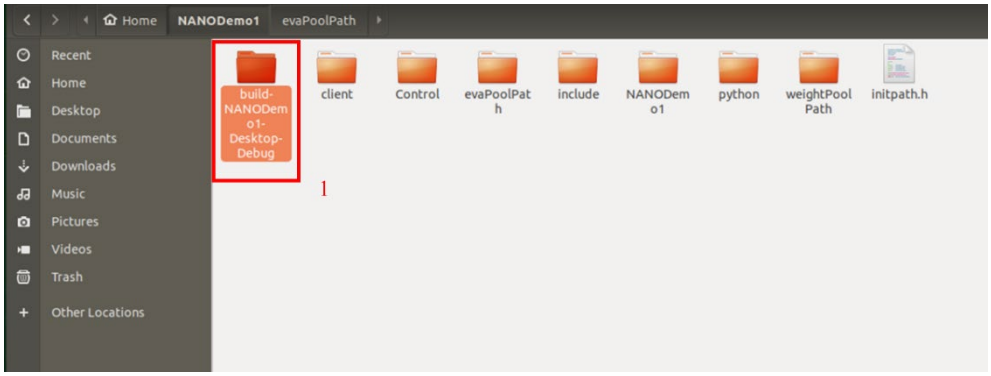
```
# 四行中斷指令用於刪除執行序，如果程式執行到一半未跑完，有可能程式會在背景執行，尤其是檢測 Power 及 Memory 的 Python 檔，最有可能

sudo pkill -9 -f python

sudo pkill -9 -f python3

sudo pkill -9 -f 專案檔名(例如紅框 2 的 RASPDemo1)

sudo pkill -9 -f ./專案檔名(需在檔名前面加上 ./)
```

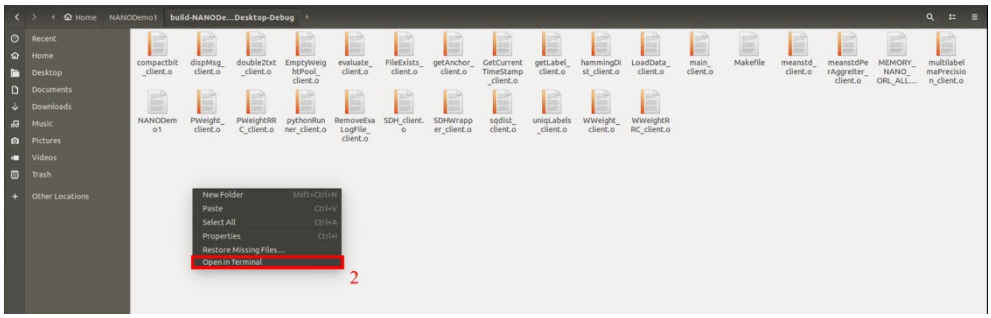


➢ 紅框 3：輸入執行程式指令(但先別按 Enter 執行)

```
# 執行程式執行檔

./專案檔名(需在檔名前面加上 ./)
```

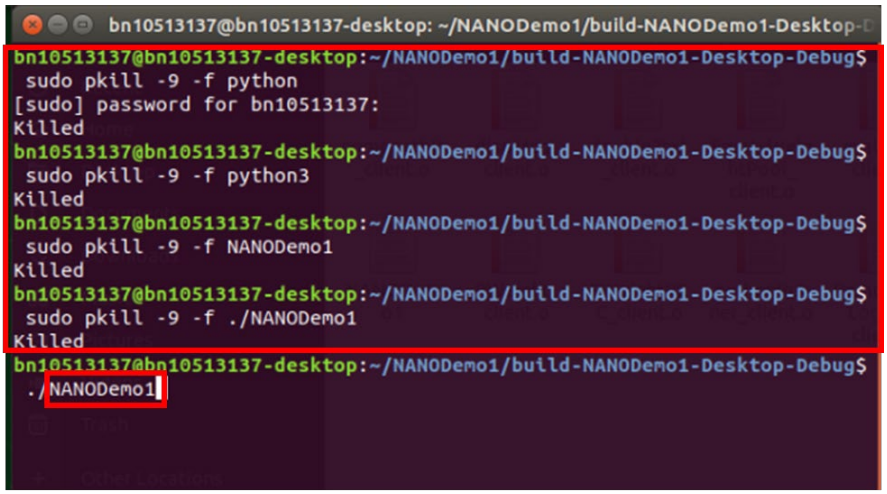
步驟 13. 打開 NANO 終端機執行程式 ➢ 紅框 1：編輯完程式後會自動創建一個資料夾 build-NANODemo1-unknown-Debug (/home/bn10513137/專案名/build-NANODemo1-unknown-Debug)，裡面有我們編輯完的程式執行檔 ➜ 紅框 2：點擊右鍵 open in Terminal





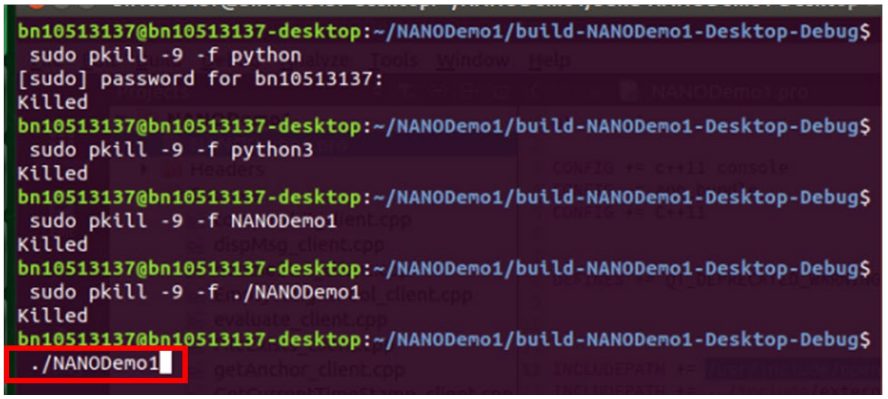
步驟 14. 在終端機上輸入執行命令

1



2

3



- 紅框 1：輸入下面四行中斷指令

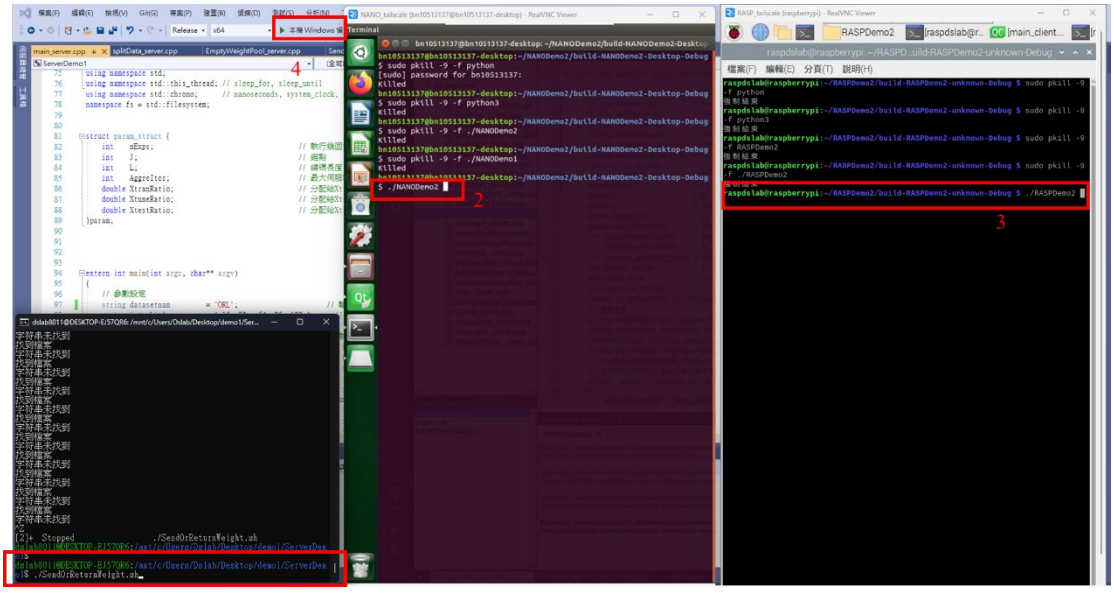
```
# 四行中斷指令用於刪除執行序的，如果程式執行到一半未跑完，有可能程式會在背景執行，尤其是檢測 Power 及 Memory 的 Python 檔，最有可能

sudo pkill -9 -f python

sudo pkill -9 -f python3

sudo pkill -9 -f 專案檔名(例如紅框 2 的 RASPDemo1)

sudo pkill -9 -f ./專案檔名(需在檔名前面加上 ./)
```

- 紅框 3：輸入執行程式指令(但先別按 Enter 執行)

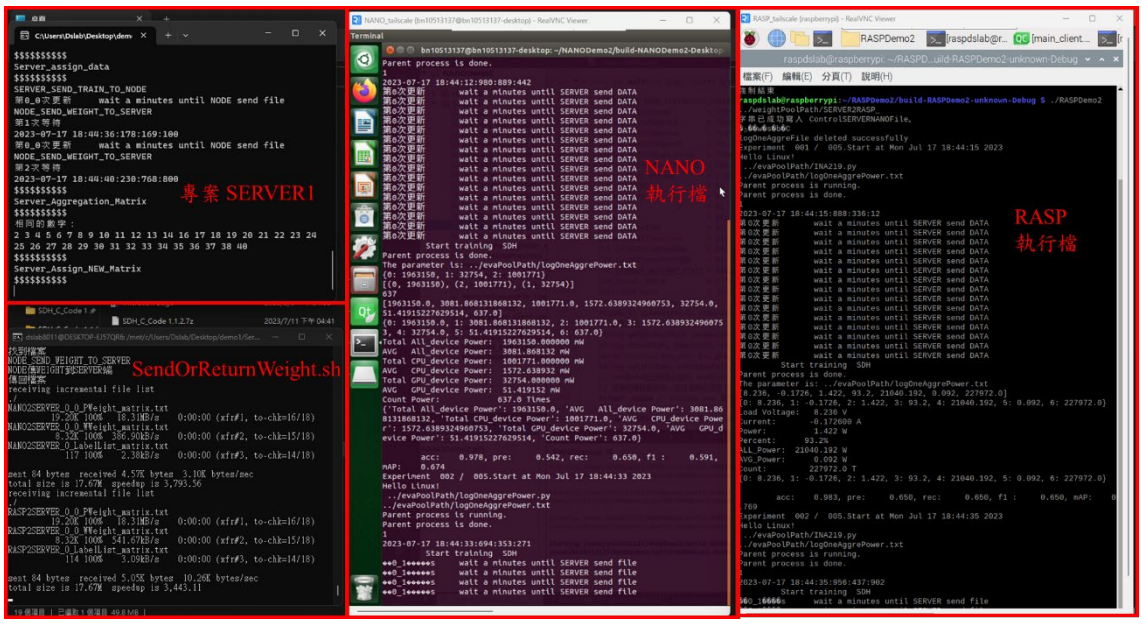
```
# 執行程式執行檔

./專案檔名(需在檔名前面加上 ./)
```

步驟 15. 檢查 SERVER 端、RASP 端和 NANO 端專案夾主程式的參數是否一致

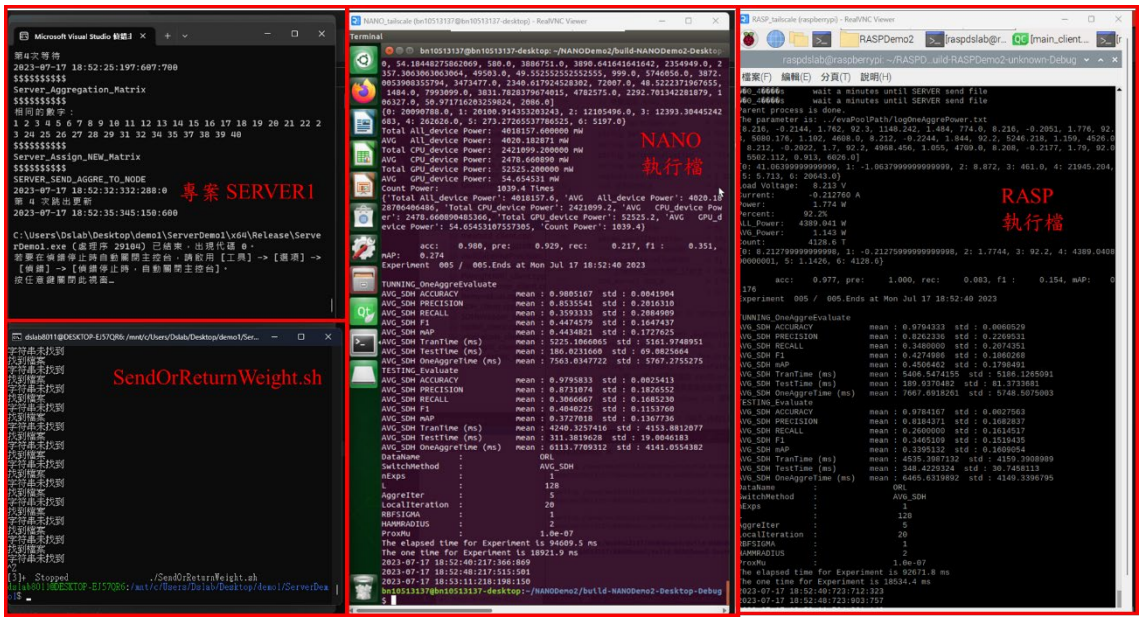
步驟 16. 依序執行紅框 1：SendOrReturnWeight.sh、紅框 2：NANO 執行檔、紅框 3：RASP 執行檔和紅框 4：專案 SERVER1 (最後一個一定要是專案 SERVER1 不然會出錯)



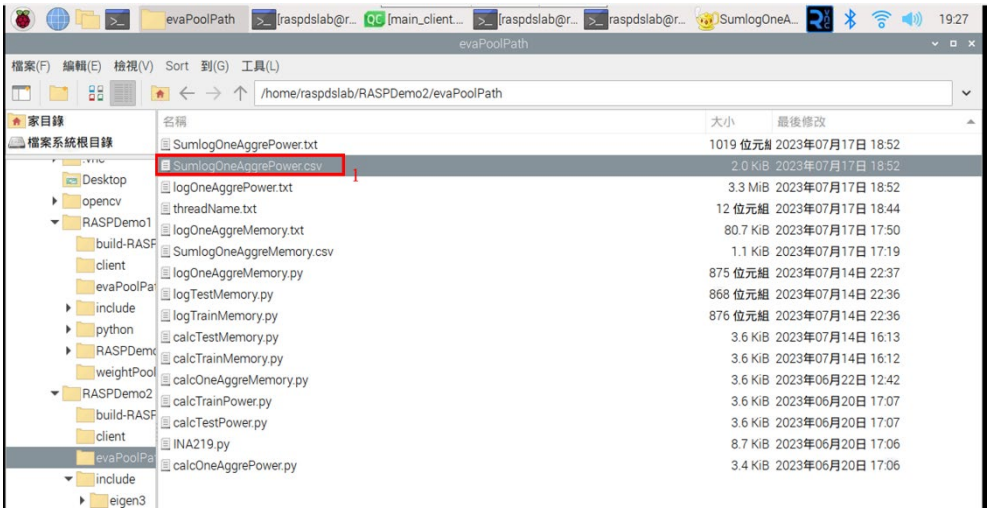
步驟 17. 程式執行中畫面

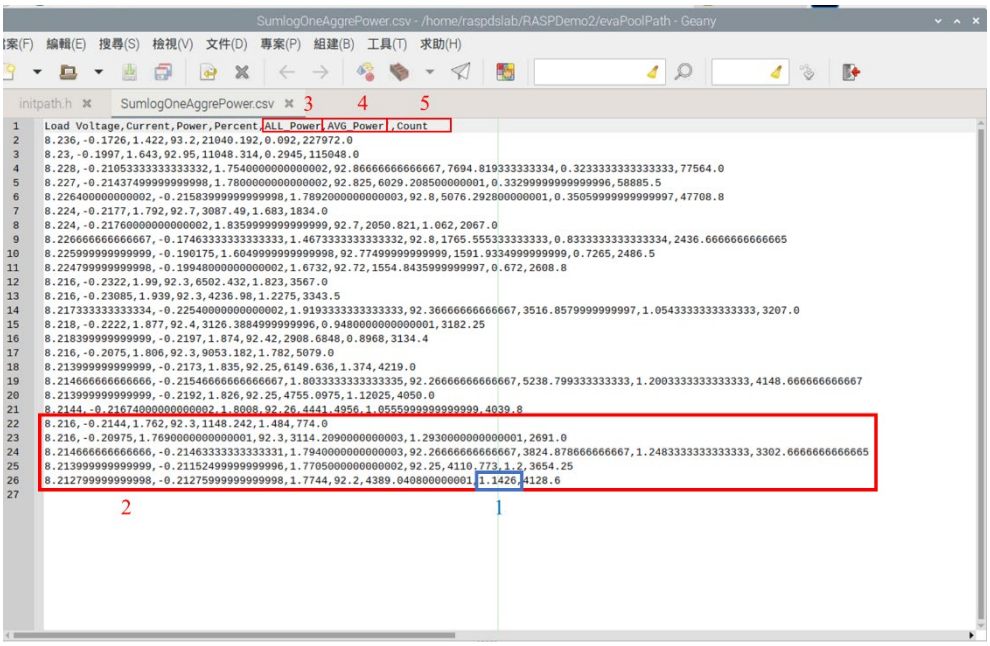


步驟 18. 程式執行結果如下

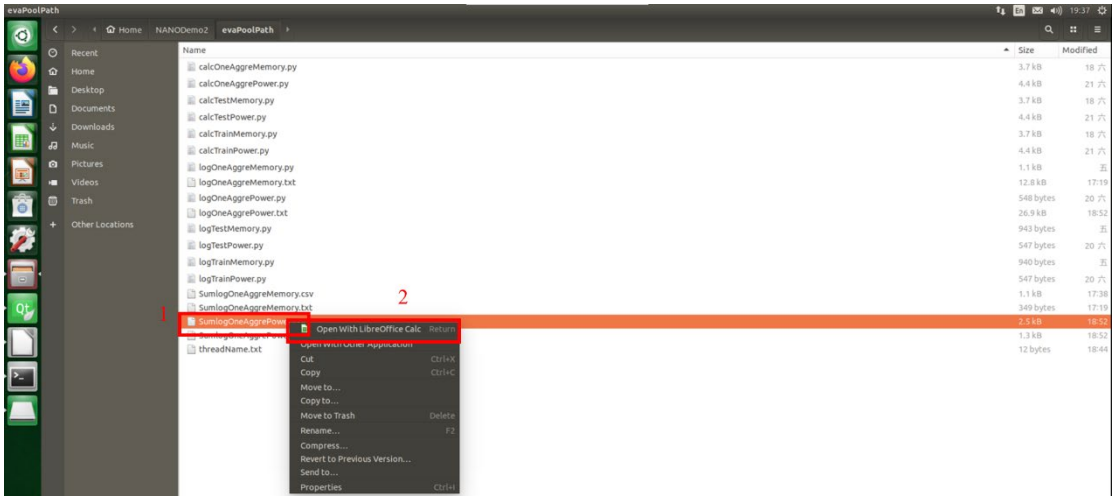


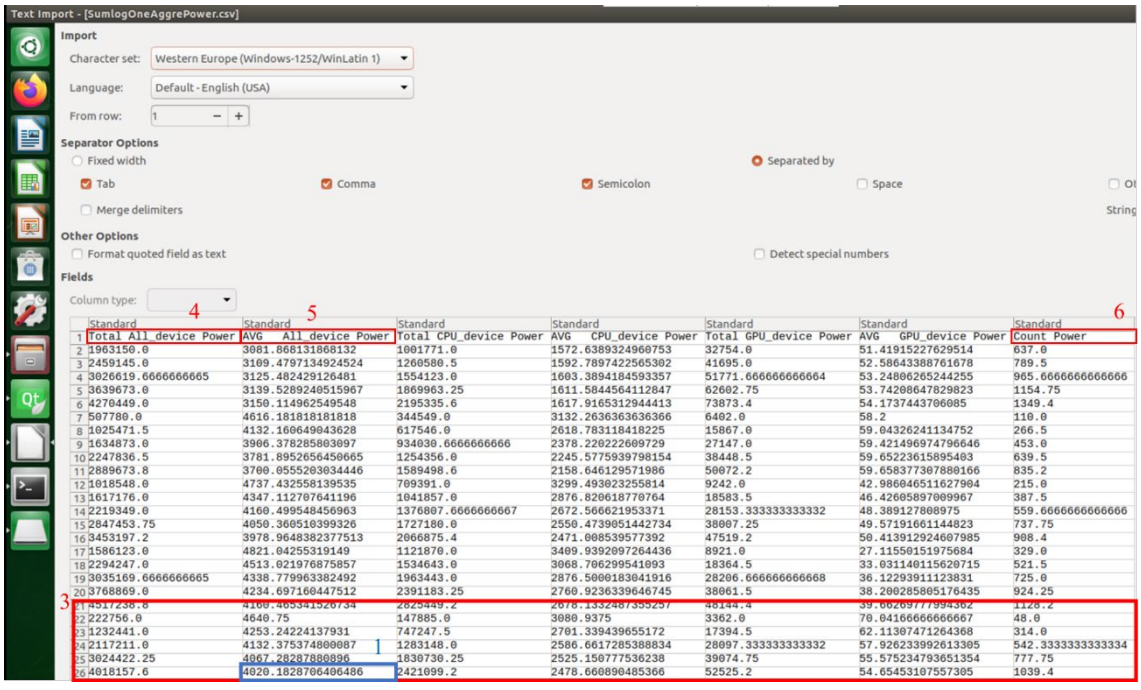
步驟 19. 測量數據(Power)可以至 RASP 帳號名/專案檔名稱/ evaPoolPath (/home/raspdslab/RASPDemo1/evaPoolPath) 尋找，例如這次測的區段為 ALL 及評估為 Power ➢ 紅框 1：為此次 RASP 端產生的最終結果➔ 紅框 2：點擊進來後，因每個碼長都跑五回，所以 128 bits 輸出結果為 22~26 行，而碼長 128 bits 的平均功耗為 1.1426 Watt (藍框 1)➔ 紅框 3：每一回累計到的功耗➔ 紅框 4：每一回累計後的平均功耗➔ 紅框 5：從第一回計次到第五回結束，記錄到 Power 的總次數。





步驟 20. 而 NANO 也是到相同位置尋找測量數據(Power)NANO 帳號名/專案檔名稱/ evaPoolPath (/home/bn10513137/NANODemo1/evaPoolPath) 尋找，例如這次測的區段為 ALL 及評估為 Power ➢紅框 1：為此次 NANO 端產生的最終結果，點擊右鍵➔ 紅框 2：點擊 Open With LibreOffice Calc➔紅框 3：點擊進來後，因每個碼長都跑五回，所以 128 bits 輸出結果為 22~26 行，而碼長 128 bits 的平均功耗為 4.0201 Watt (藍框 1)➔ 紅框 4：每一回累計到的功耗➔ 紅框 5：每一回累計後的平均功耗➔ 紅框 6：從第一回計次到第五回結束，記錄到 Power 的總次數。





2.4 備份 RASP 及 NANO 系統

- 備份前請先備好乾淨的 SD 卡(格式化)、讀卡機和在虛擬基裝一台 Linux 系統(UBUNTU 22 版本)

```
# lsblk 看 SD 路徑(ex. /dev/sdb)
lsblk
#把 RASP 現有系統備份成映像檔的指令，會產生壓縮檔 NCS-raspbian.zip
sudo dd bs=4M if=/dev/sda status=progress | zip NCS-raspbian.zip -
# 最終解壓縮的檔名為 -，需要自己改名為 raspbian.img
#下載 pishrink 腳本。
wget https://raw.githubusercontent.com/Drewsif/PiShrink/master/pishrink.sh
*.img 可以透過 pishrink 來縮減映像檔大小
chmod +x pishrink.sh
sudo ./pishrink.sh raspbian.img
```