

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

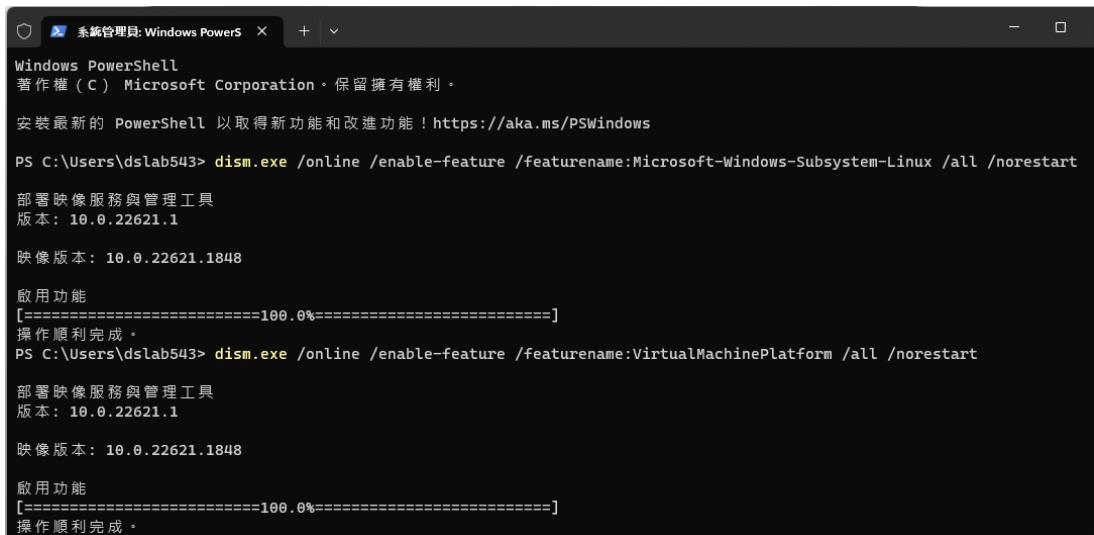
1.1 在 Windows 上安裝 WSL

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
```



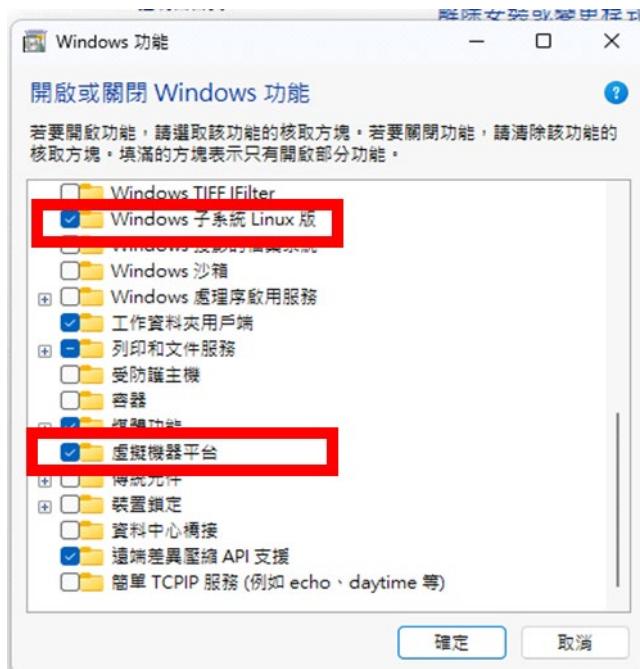
```
Windows PowerShell
著作權 (C) Microsoft Corporation。保留擁有權利。
安裝最新的 PowerShell 以取得新功能和改進功能！https://aka.ms/PSWindows
PS C:\Users\dslab543> dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
部署映像服務與管理工具
版本: 10.0.22621.1
映像版本: 10.0.22621.1848
啟用功能
[=====100.0%=====]
操作順利完成。
PS C:\Users\dslab543> dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
部署映像服務與管理工具
版本: 10.0.22621.1
映像版本: 10.0.22621.1848
啟用功能
[=====100.0%=====]
操作順利完成。
```

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】：

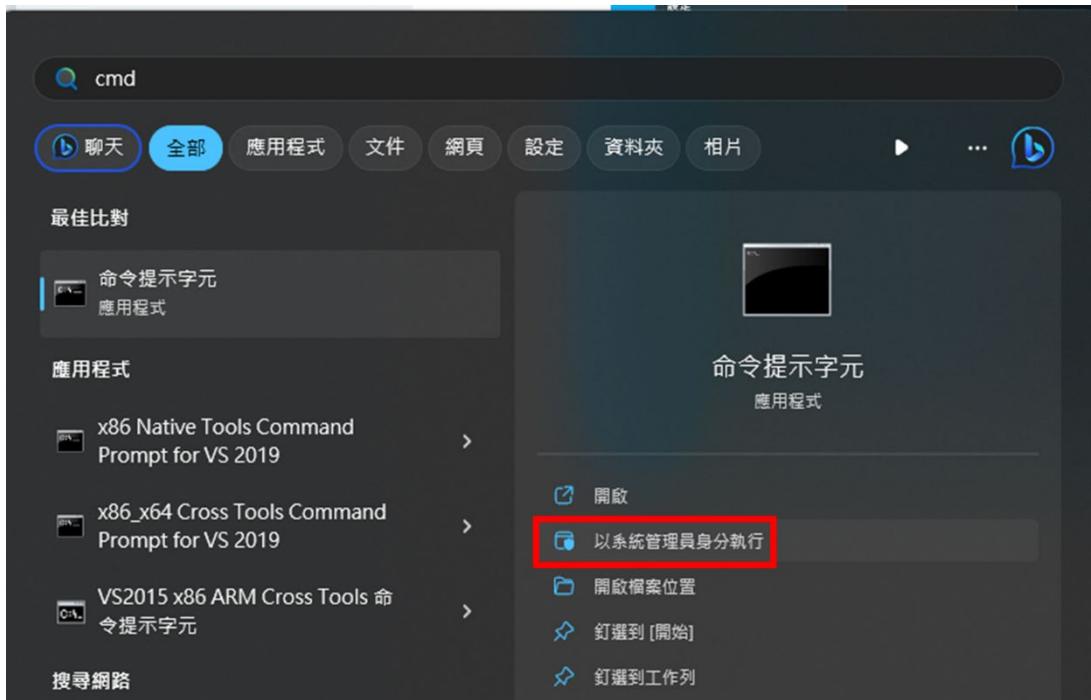


- ✿ 彈出下面這個視窗：需要勾選【適用於 Linux 的 Windows 子系統】和【虛擬機器平臺】這兩項。



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
```

```
C:\ 系統管理員: 命令提示字元
Microsoft Windows [版本 10.0.22621.1848]
(c) Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Windows\System32>wsl --install
正在安裝 Windows 子系統 Linux 版
已完成安裝 Windows 子系統 Linux 版。
正在安裝 Ubuntu
已完成安裝 Ubuntu。
已成功執行所要求的操作。請重新開機，變更才能生效。
```

- 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。

```
dslab8011@DESKTOP-EJ57QR1 ~ + | ~
已安裝 Ubuntu。
正在啟動 Ubuntu...
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: dslab8011
New password:
Retype new password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This message is shown once a day. To disable it please create the
/home/dslab8011/.hushlogin file.
```

➤ 按 Win+X，啟動 終端機(系統管理員)



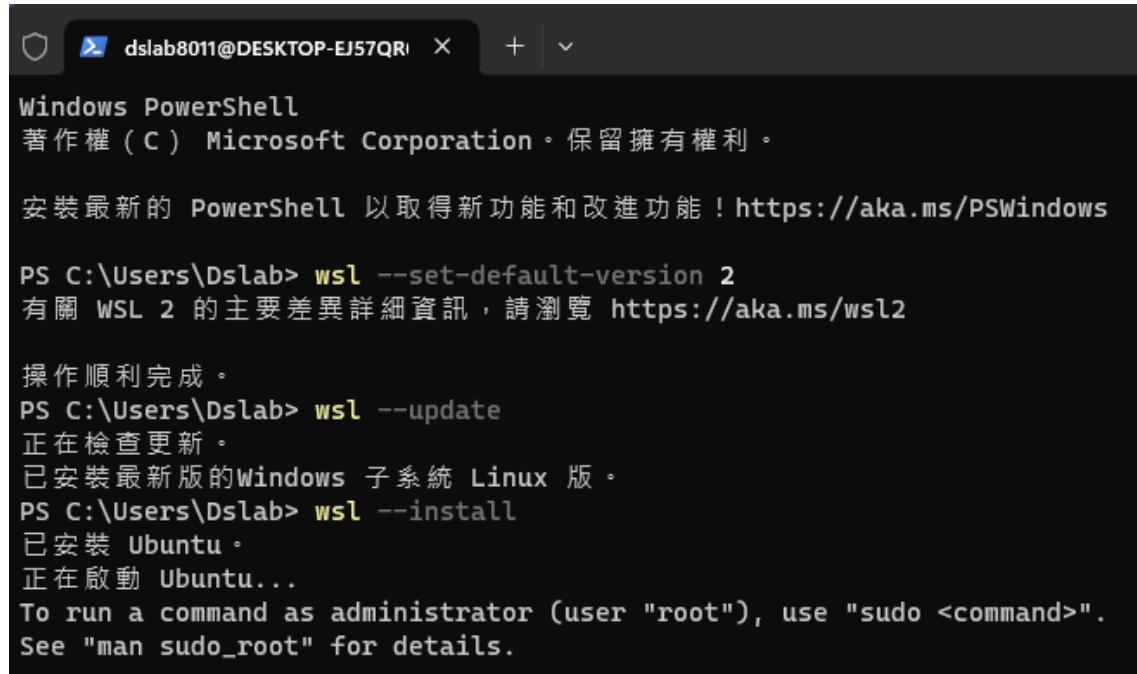
確認 WSL 是否為最新版

wsl --set-default-version 2

執行更新 wsl 命令

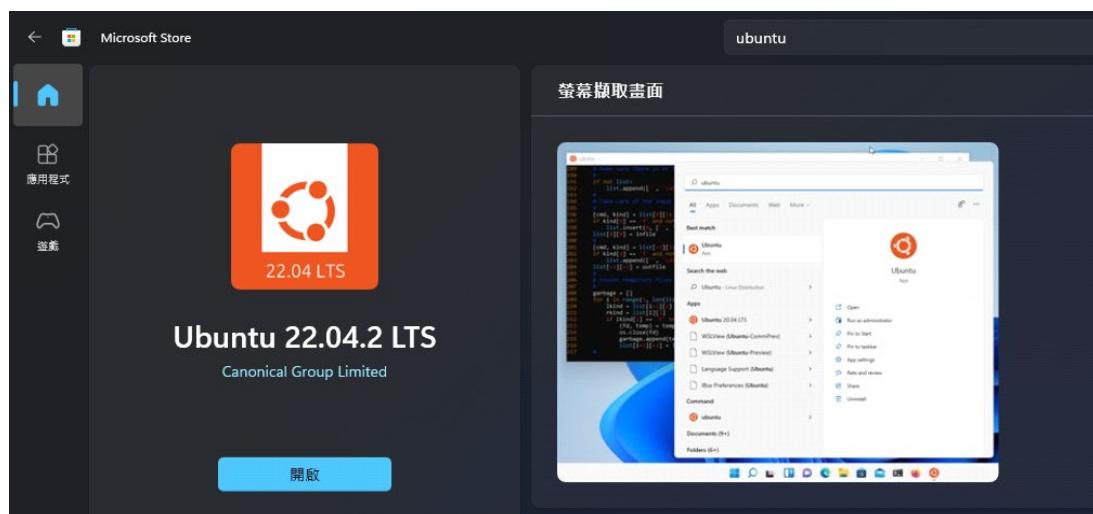
wsl --update

```
wsl --install
```

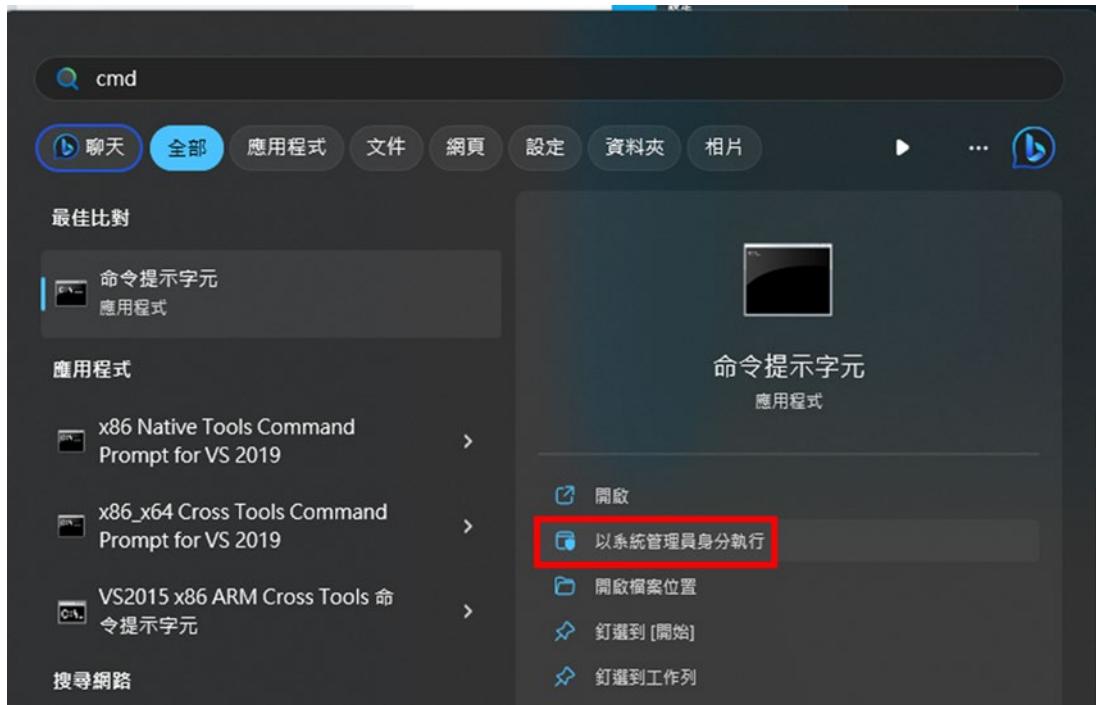


```
dslab8011@DESKTOP-EJ57QR: ~ + | v  
Windows PowerShell  
著作權 (C) Microsoft Corporation。保留擁有權利。  
安裝最新的 PowerShell 以取得新功能和改進功能！https://aka.ms/PSWindows  
PS C:\Users\DsLab> wsl --set-default-version 2  
有關 WSL 2 的主要差異詳細資訊，請瀏覽 https://aka.ms/wsl2  
  
操作順利完成。  
PS C:\Users\DsLab> wsl --update  
正在檢查更新。  
已安裝最新版的Windows 子系統 Linux 版。  
PS C:\Users\DsLab> wsl --install  
已安裝 Ubuntu。  
正在啟動 Ubuntu...  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.
```

⊕ 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

```
dslab8011@DESKTOP-EJ57QR6: /mnt/c/Windows/System32
Microsoft Windows [版本 10.0.22621.1848]
(c) Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Windows\System32>wsl
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

dslab8011@DESKTOP-EJ57QR6:/mnt/c/Windows/System32$
```

- ✚ /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

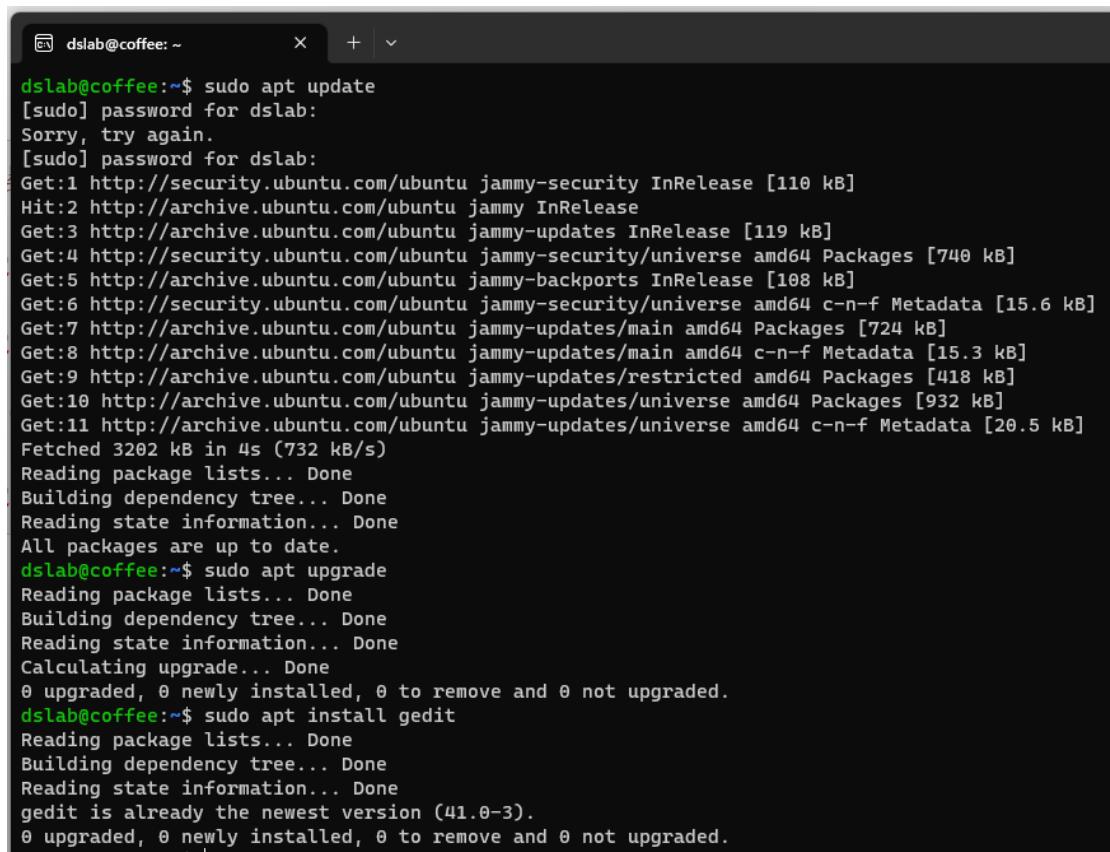
```
# 安裝 GUI 程式並測試
```

```
sudo apt update
```

```
sudo apt upgrade
```

```
# gnome 桌面下的編輯器
```

```
sudo apt install gedit
```

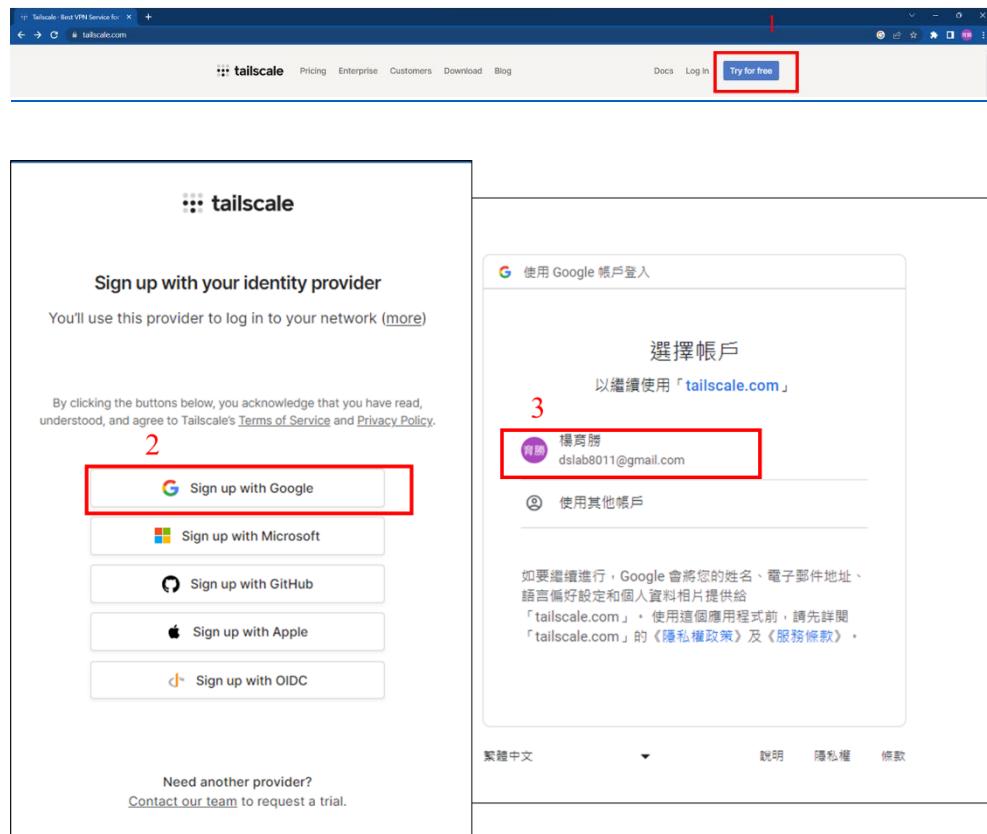


```
dslab@coffee:~$ sudo apt update
[sudo] password for dslab:
Sorry, try again.
[sudo] password for dslab:
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [740 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [108 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [15.6 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [724 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [15.3 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [418 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [932 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [20.5 kB]
Fetched 3202 kB in 4s (732 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
dslab@coffee:~$ sudo apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
dslab@coffee:~$ sudo apt install gedit
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gedit is already the newest version (41.0-3).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

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

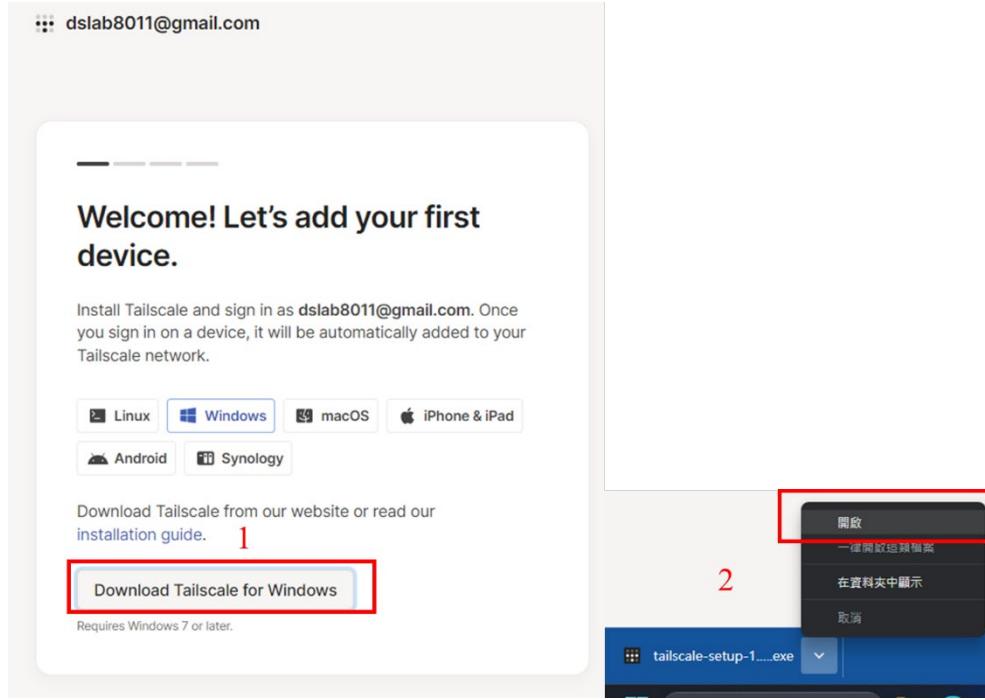
➔ 紅框 3：點擊自己帳號



- ✚ 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/0ac797413a56c6b2772f48a567a32cddd3b739f5b2af649fcf90be4245762ff/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



1.5 設定 Server 端上的程式

⊕ 打開 Visual Studio 2019

步驟 1. 建立新的專案



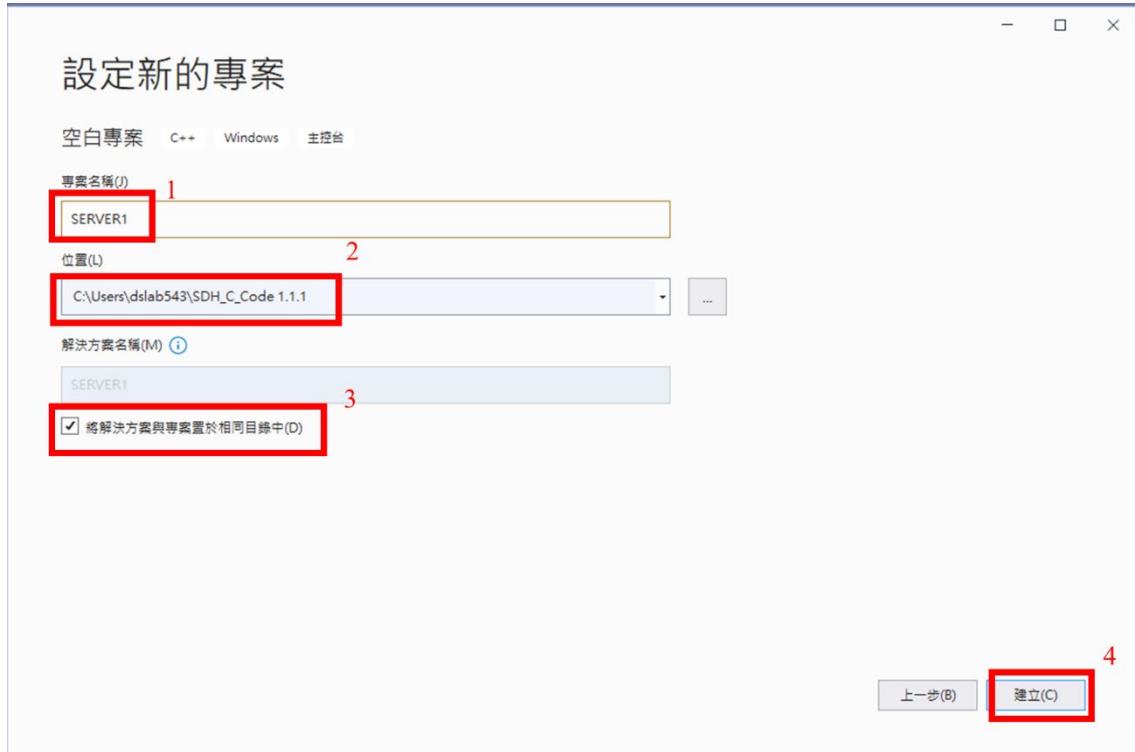
步驟 2. 空白專案 → 下一步



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

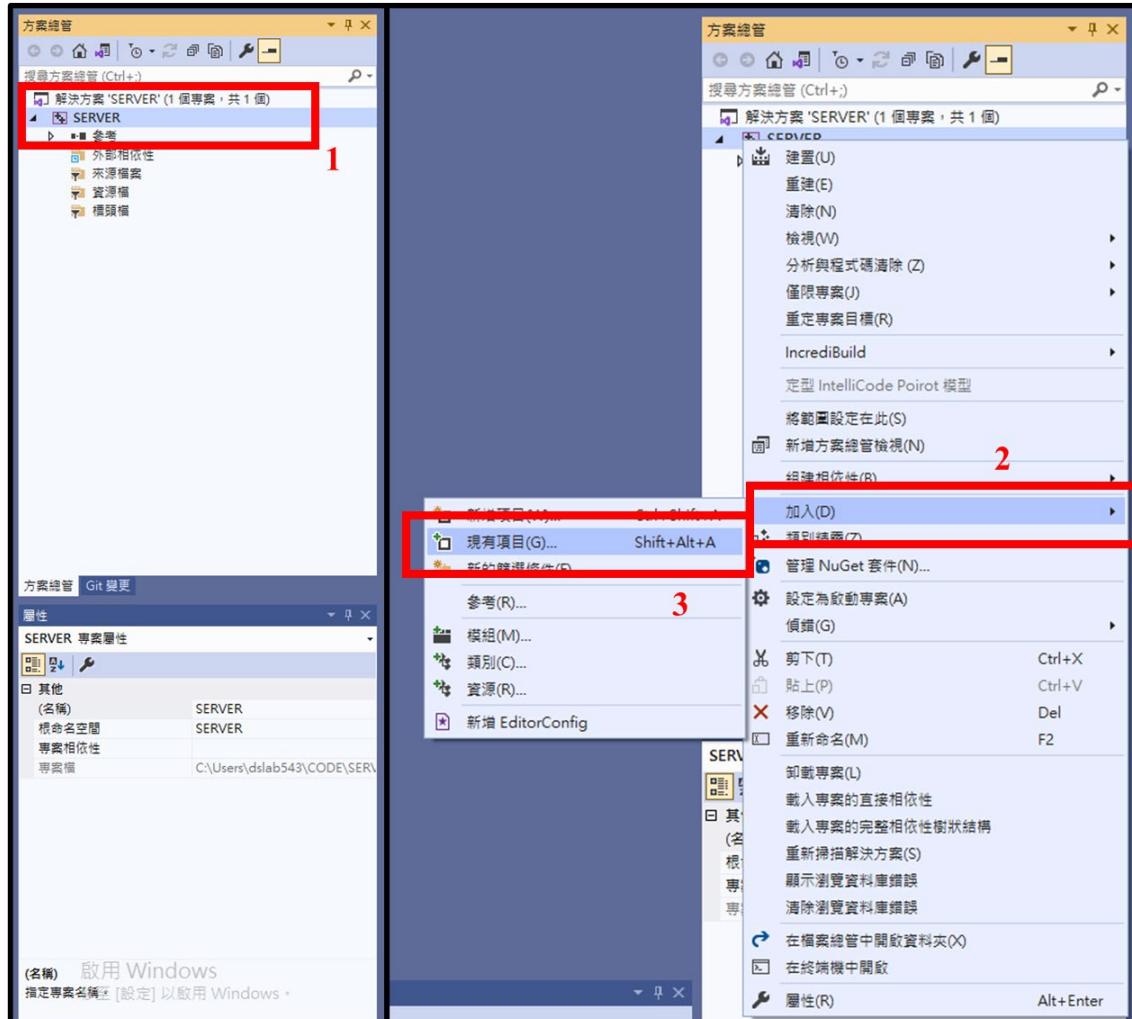
(C:\Users\dslab543\SDH_C_Code 1.1.1\SERVER1)底下

名稱	修改日期	類型	大小
.vs	2023/6/27 下午 01:39	檔案資料夾	
unsupported	2023/6/27 下午 01:43	檔案資料夾	
double2txt_server.cpp	2023/6/12 上午 02:28	CPP 檔案	2 KB
double2txt_server.h	2023/6/8 下午 06:51	C/C++ Header	1 KB
EmptyWeightPool_server.cpp	2023/6/12 上午 12:23	CPP 檔案	4 KB
EmptyWeightPool_server.h	2023/5/29 下午 04:57	C/C++ Header	1 KB
FileExists_server.cpp	2023/6/9 下午 01:58	CPP 檔案	1 KB
FileExists_server.h	2023/5/26 上午 08:58	C/C++ Header	1 KB
GetCurrentTimeStamp_server.cpp	2023/6/22 下午 11:42	CPP 檔案	3 KB
GetCurrentTimeStamp_server.h	2023/6/2 下午 01:42	C/C++ Header	1 KB
int2txt_server.cpp	2023/6/12 上午 03:25	CPP 檔案	2 KB
int2txt_server.h	2023/6/8 下午 06:52	C/C++ Header	1 KB
LoadData_server.cpp	2023/6/12 上午 03:24	CPP 檔案	2 KB
LoadData_server.h	2023/6/21 上午 12:36	C/C++ Header	1 KB
main_server.cpp	2023/6/25 下午 08:10	CPP 檔案	20 KB
main_server.h	2023/6/23 上午 01:40	C/C++ Header	1 KB
rtwtypes.h	2021/12/31 上午 10:35	C/C++ Header	2 KB
SEND_FILE.txt	2023/6/21 上午 01:10	文字文件	0 KB
SEND_FILE_CTRL_MSG.txt	2023/6/25 下午 08:14	文字文件	1 KB
SendOrReturnWeight.sh	2023/6/21 上午 03:32	sh_auto_file	3 KB
SendOrReturnWeight_server.cpp	2023/6/25 下午 06:37	CPP 檔案	2 KB
SendOrReturnWeight_server.h	2023/6/21 上午 12:36	C/C++ Header	1 KB
SERVER1.sln	2023/6/27 下午 01:39	Visual Studio Sol...	2 KB
SERVER1.vcxproj	2023/6/27 下午 01:39	VCXPROJ 檔案	8 KB
SERVER1.vcxproj.filters	2023/6/27 下午 01:39	VC++ Project Filt...	1 KB
SERVER1.vcxproj.user	2023/6/27 下午 01:39	Per-User Project ...	1 KB
splitData_server.cpp	2023/6/22 下午 11:48	CPP 檔案	7 KB
splitData_server.h	2023/6/8 下午 06:57	C/C++ Header	1 KB
unigl_labels_server.cpp	2023/6/22 下午 11:46	CPP 檔案	2 KB

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

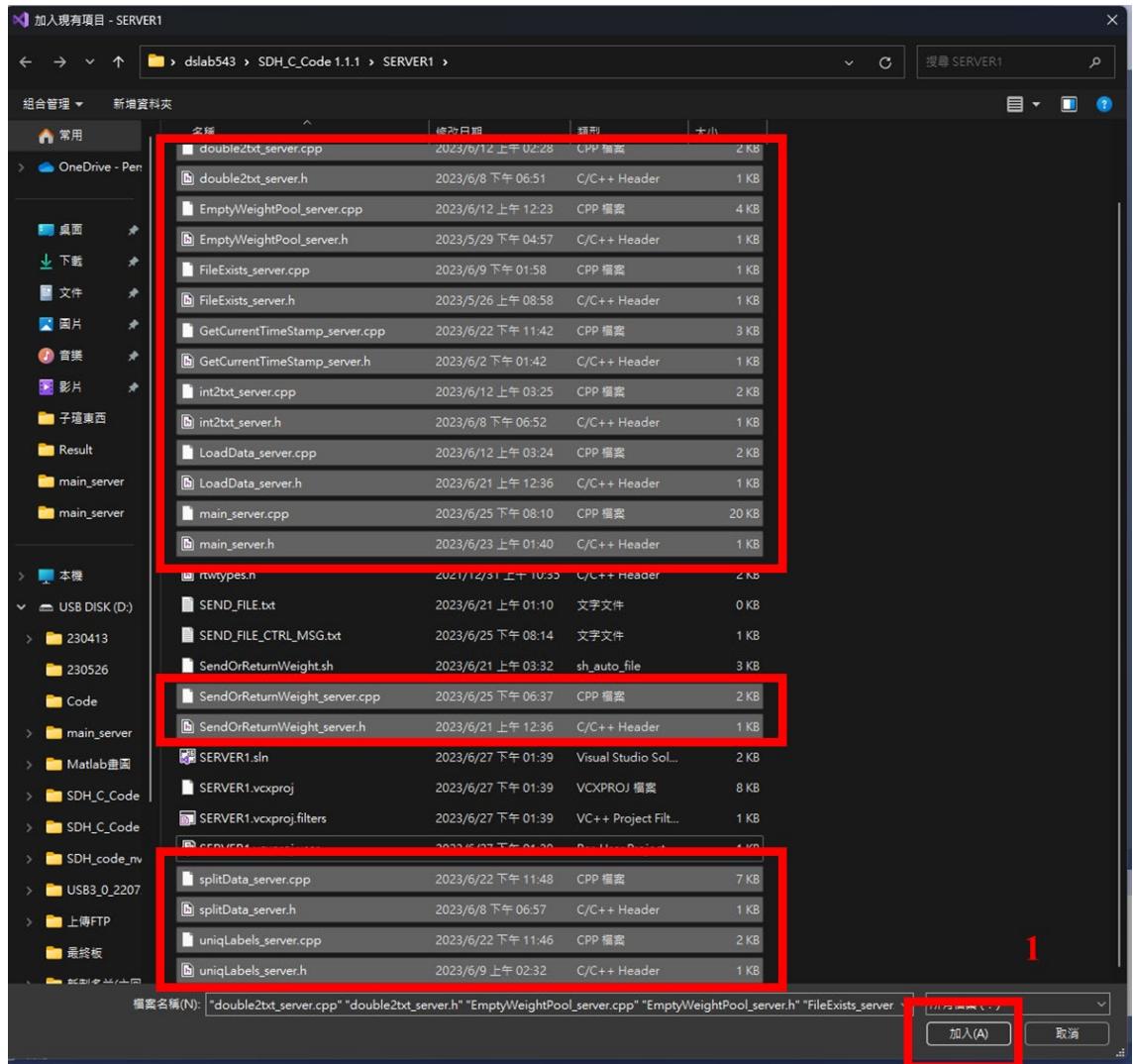
專案檔 ➔ 紅框 2：加入 ➔ 紅框 3：現有項目，將 SERVER 端程式加入專案

中，讓專案讀取



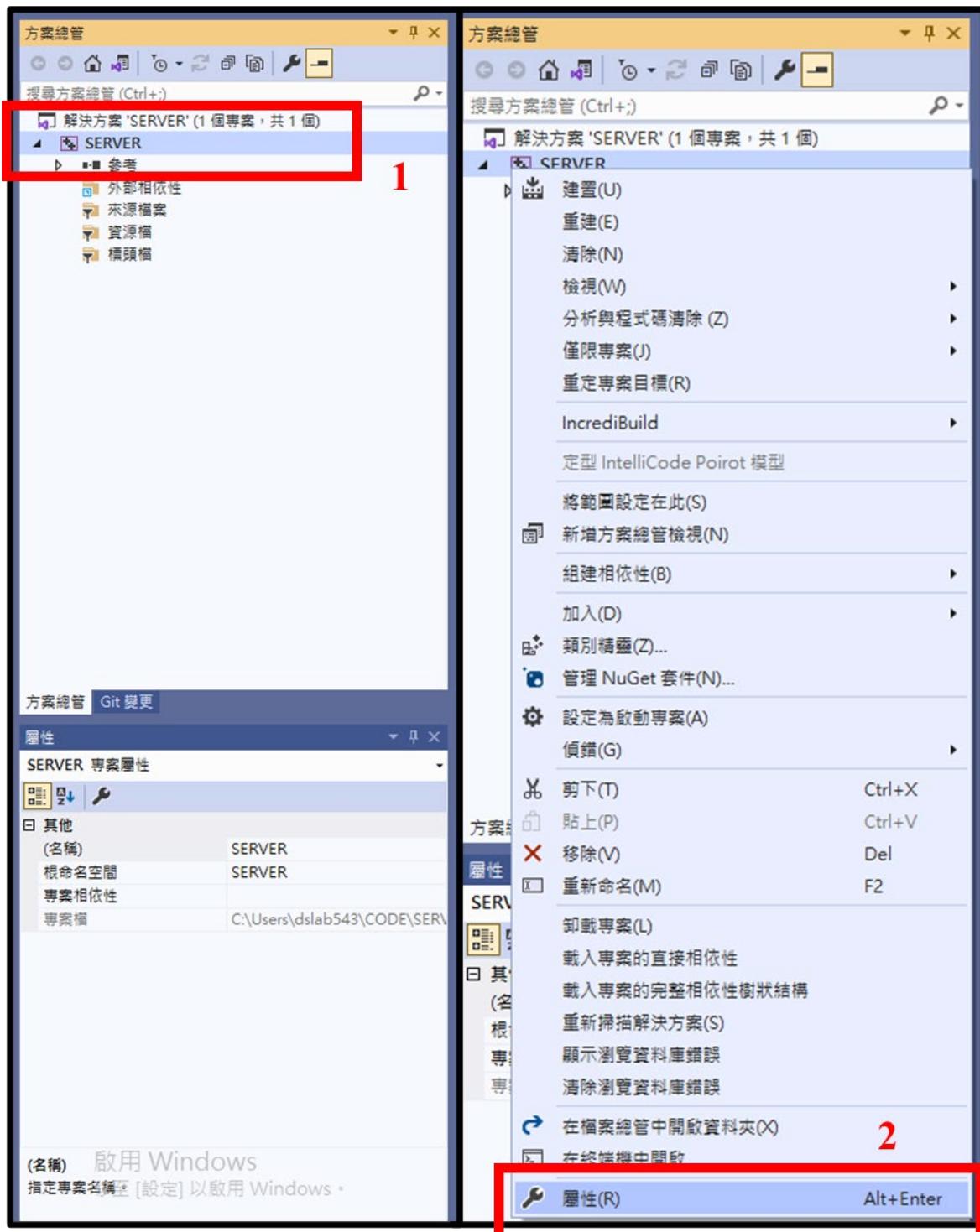
步驟 6. 加入 server 端程式至專案中，讓專案讀取 ➤ 只選取 CPP 檔案及

Header 檔案 → 紅框 1：加入



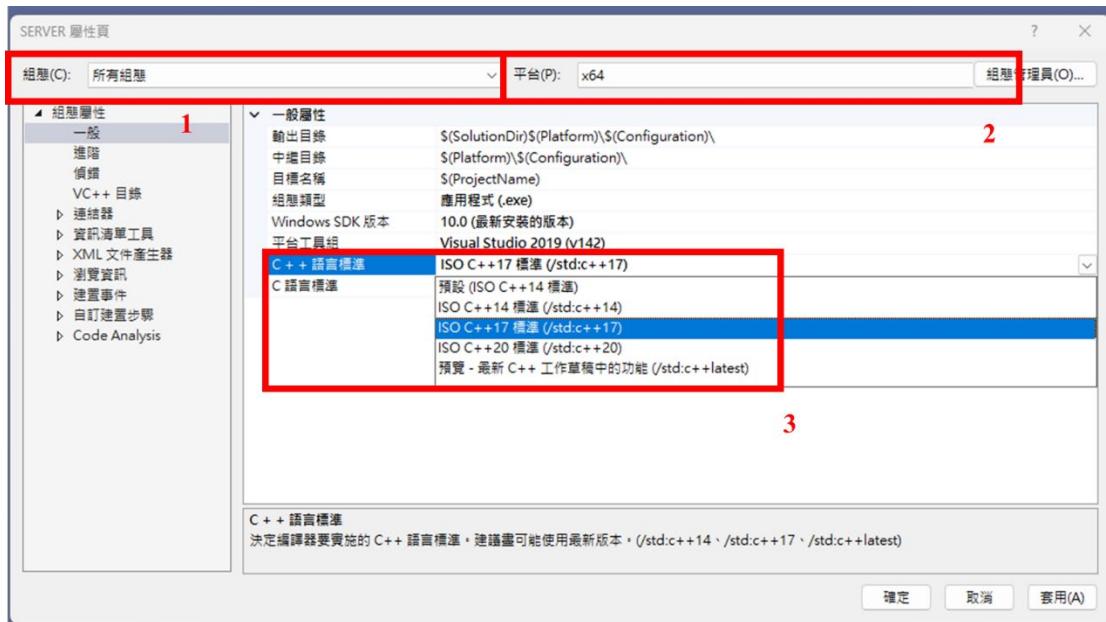
步驟 7. 將必需函式庫及套件包加入 ➤ 紅框 1：右鍵點擊旁邊專案檔 ➔ 紅框

2：屬性



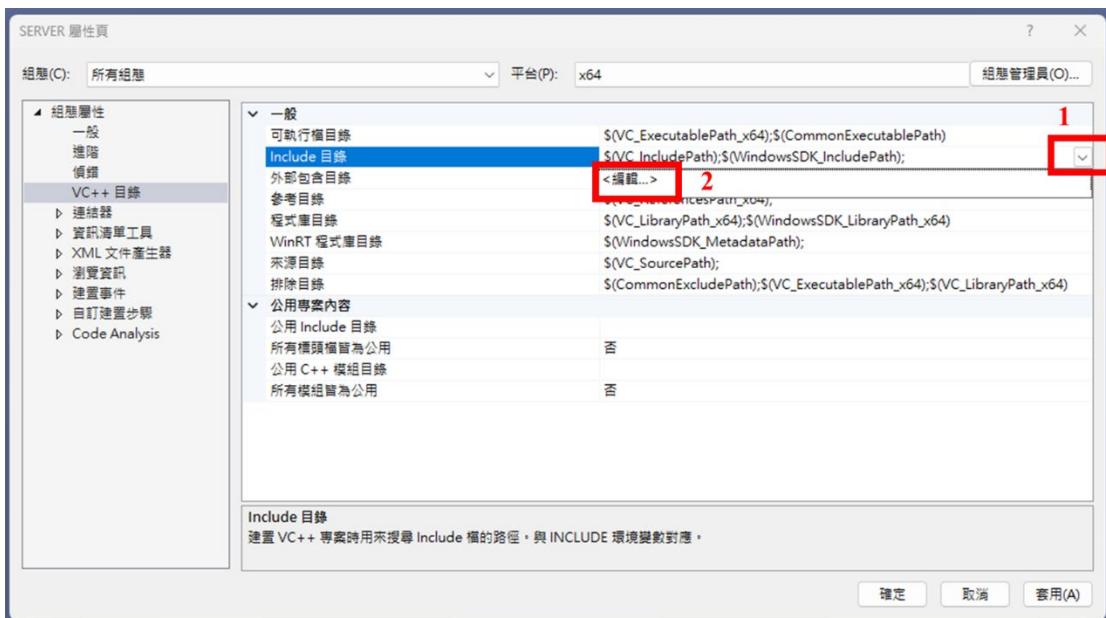
步驟 8. 將必需函式庫及套件包加入 ➤ 紅框 1：將組態選為 專案檔 → 紅框 2：

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**



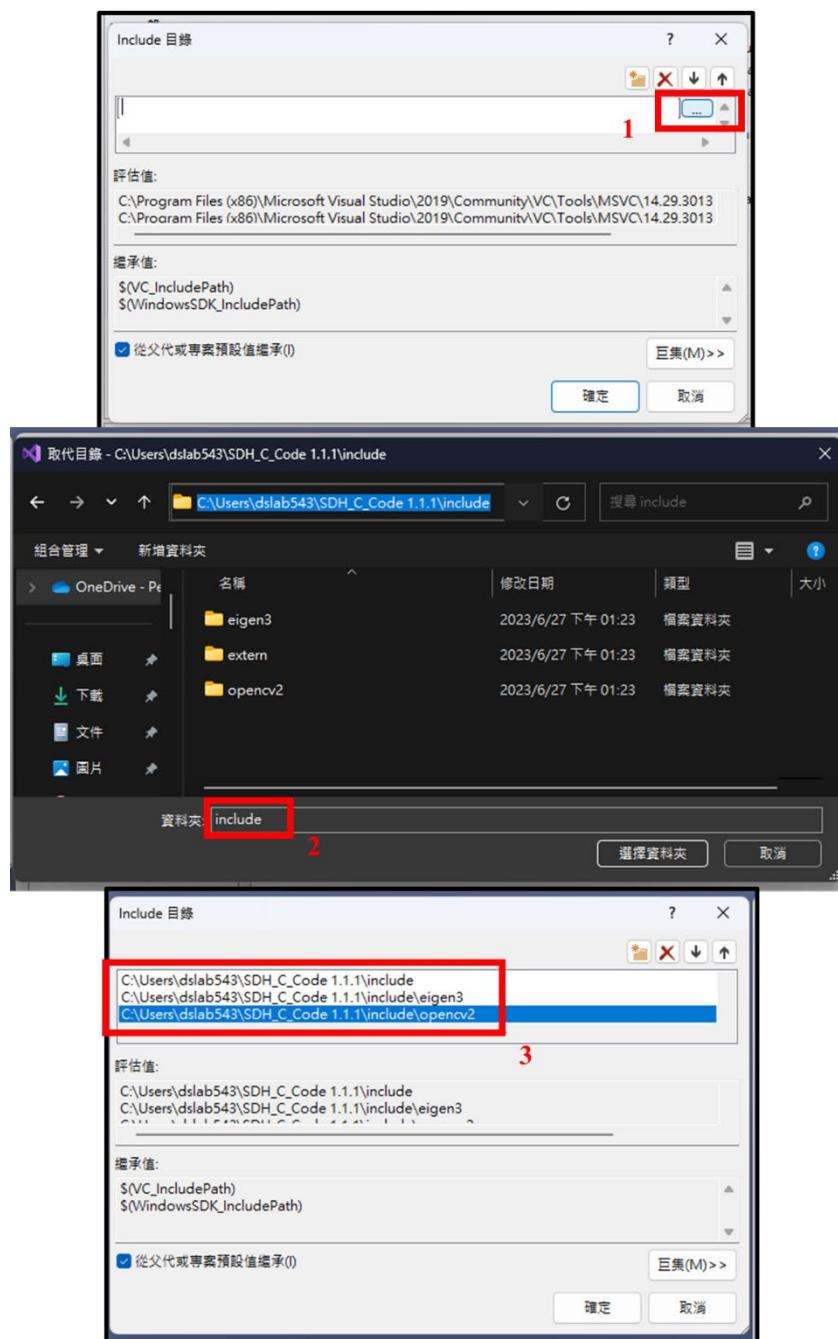
步驟 10. 將必需函式庫及套件包加入 ➤ 紅框 1：路徑按鈕 → 紅框 2：選取下面
指定的三個路徑 ➔ 紅框 3：指定完三個路徑的結果畫面 ➔ 點擊確認

指定的三個路徑

C:\程式包路徑\程式包名\include

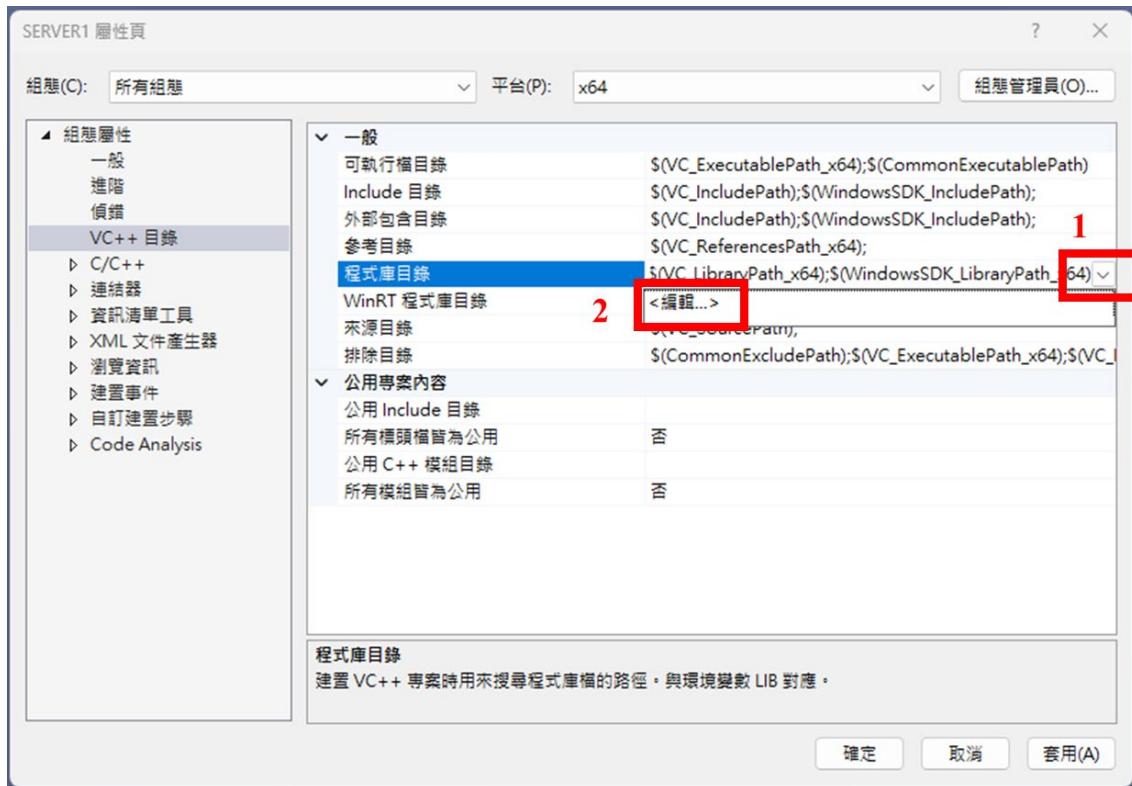
C:\程式包路徑\程式包名\include\opencv2

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

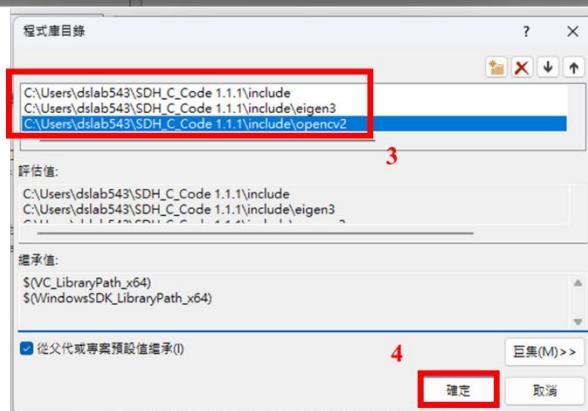
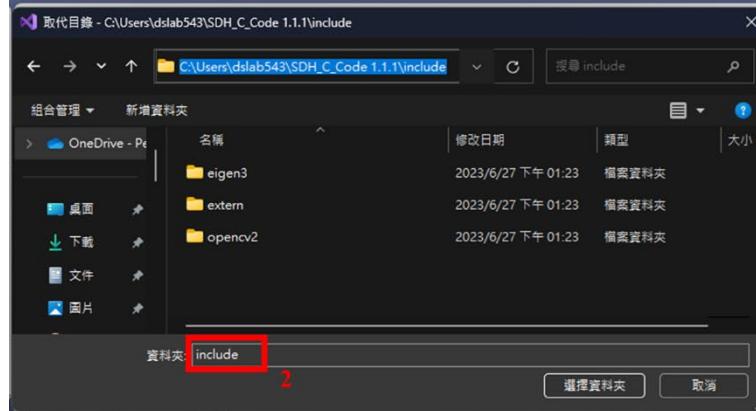
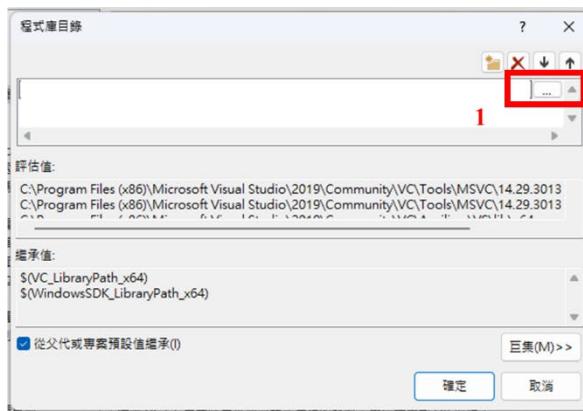
→ 點擊確定

指定的三個路徑

C:\程式包路徑\程式包名\include

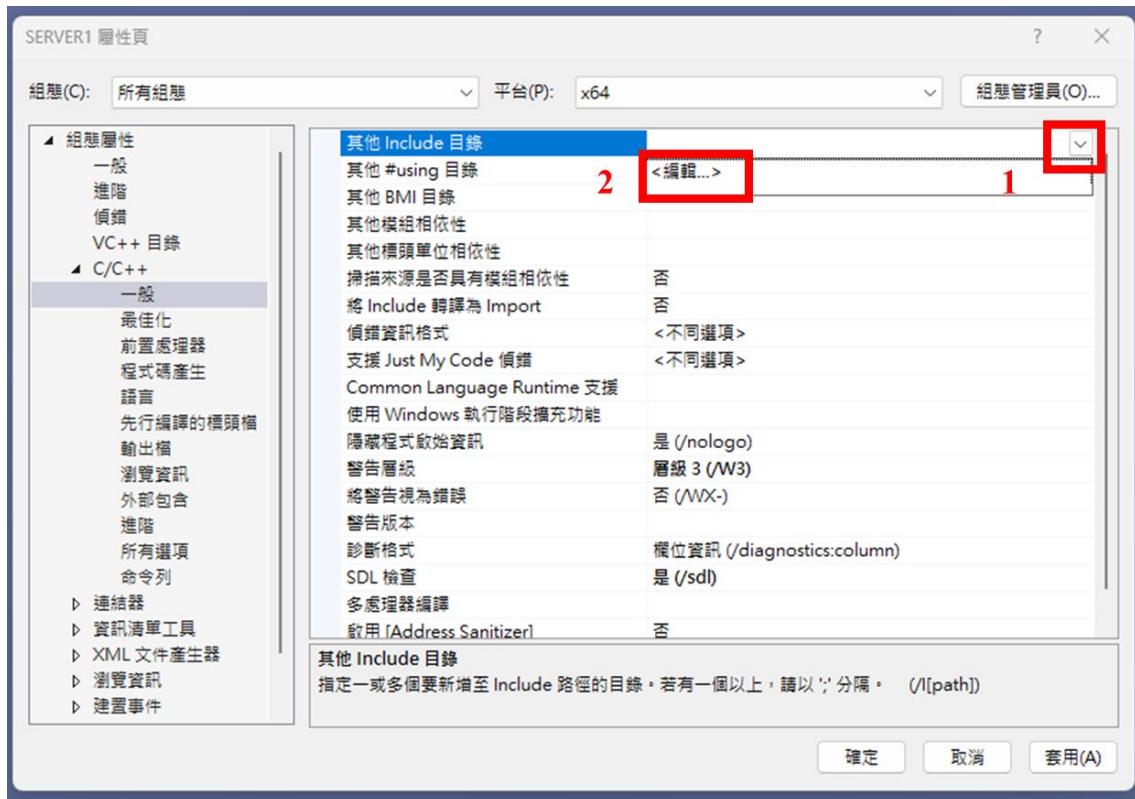
C:\程式包路徑\程式包名\include\opencv2

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

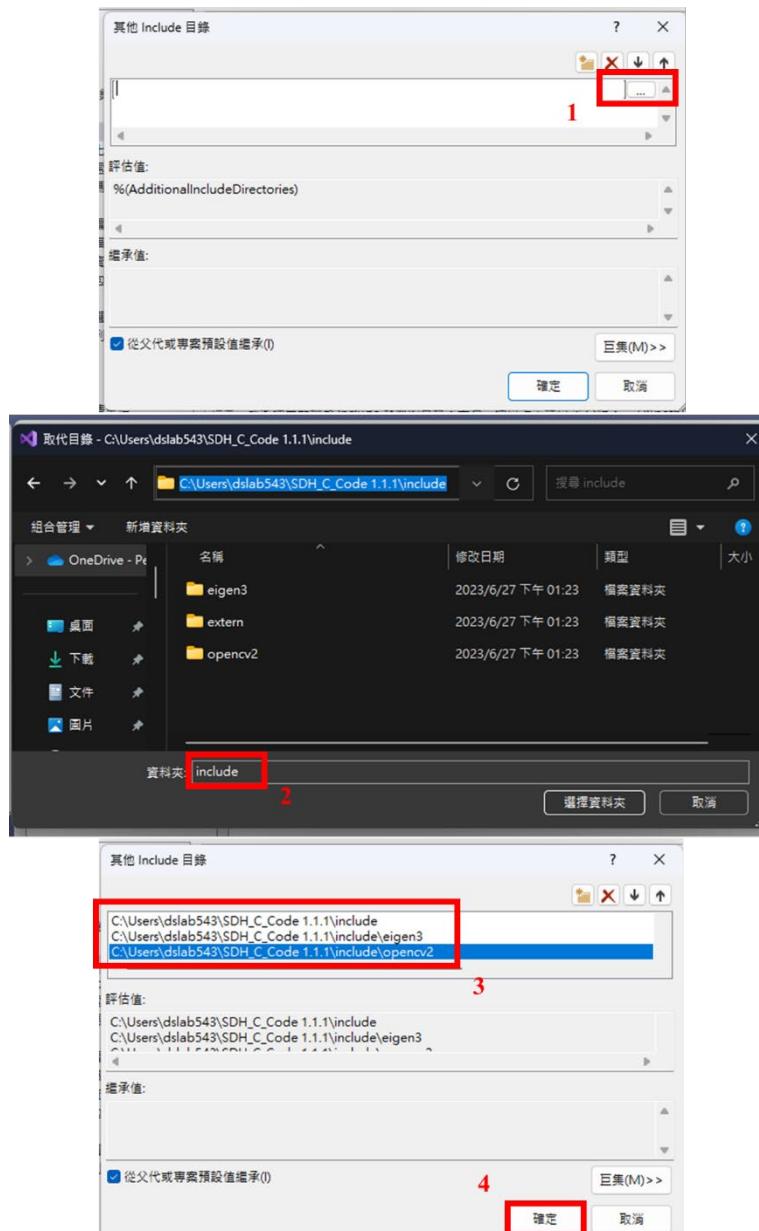
→ 點擊確定

指定的三個路徑

C:\程式包路徑\程式包名\include

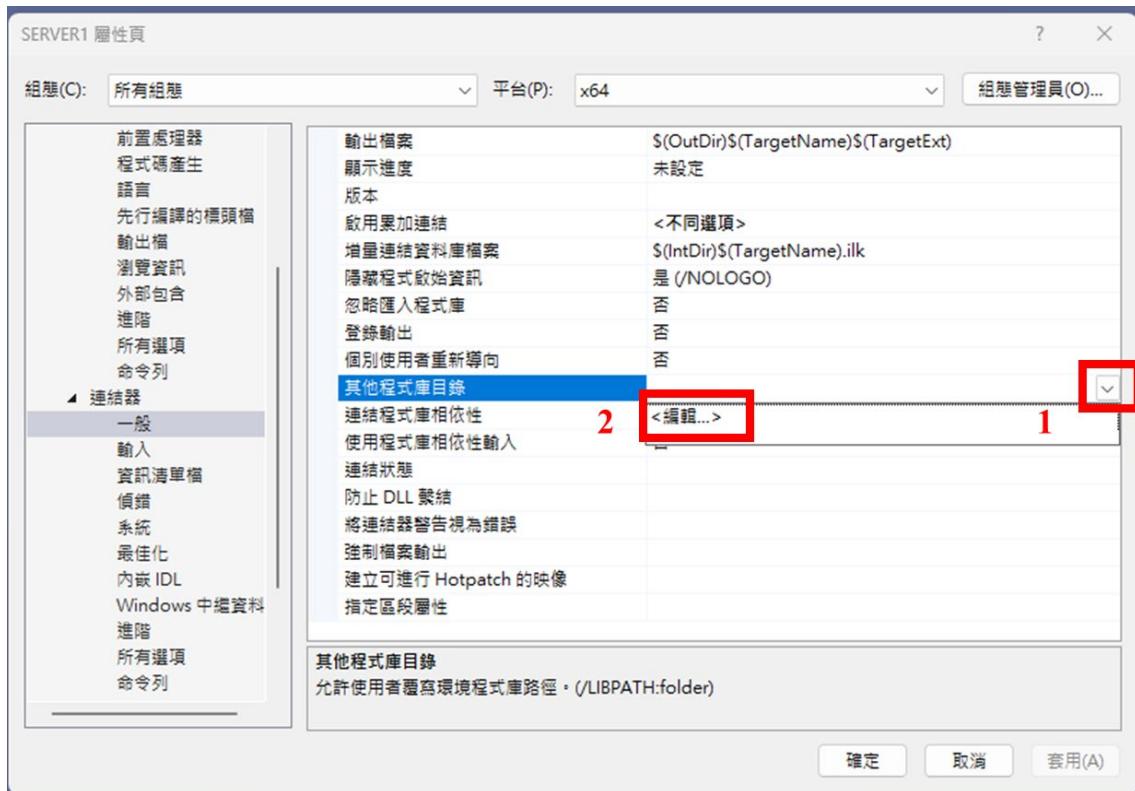
C:\程式包路徑\程式包名\include\opencv2

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

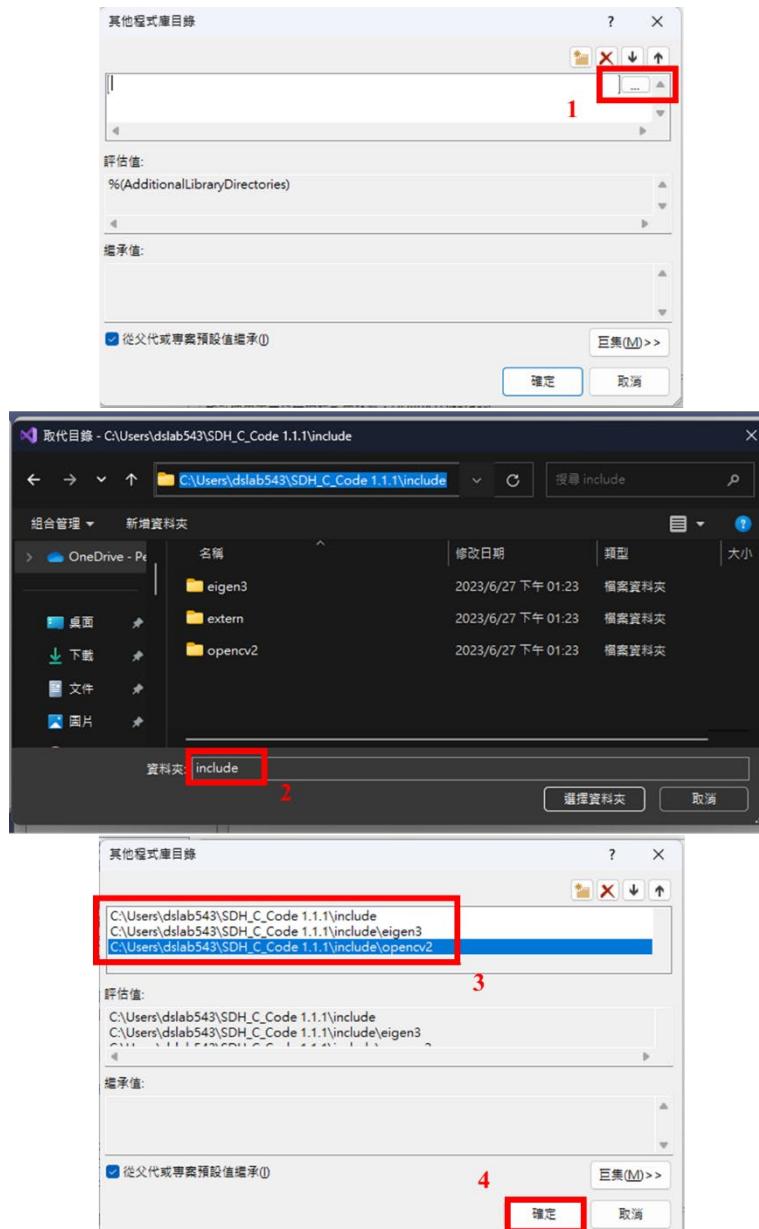
→ 點擊確定

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

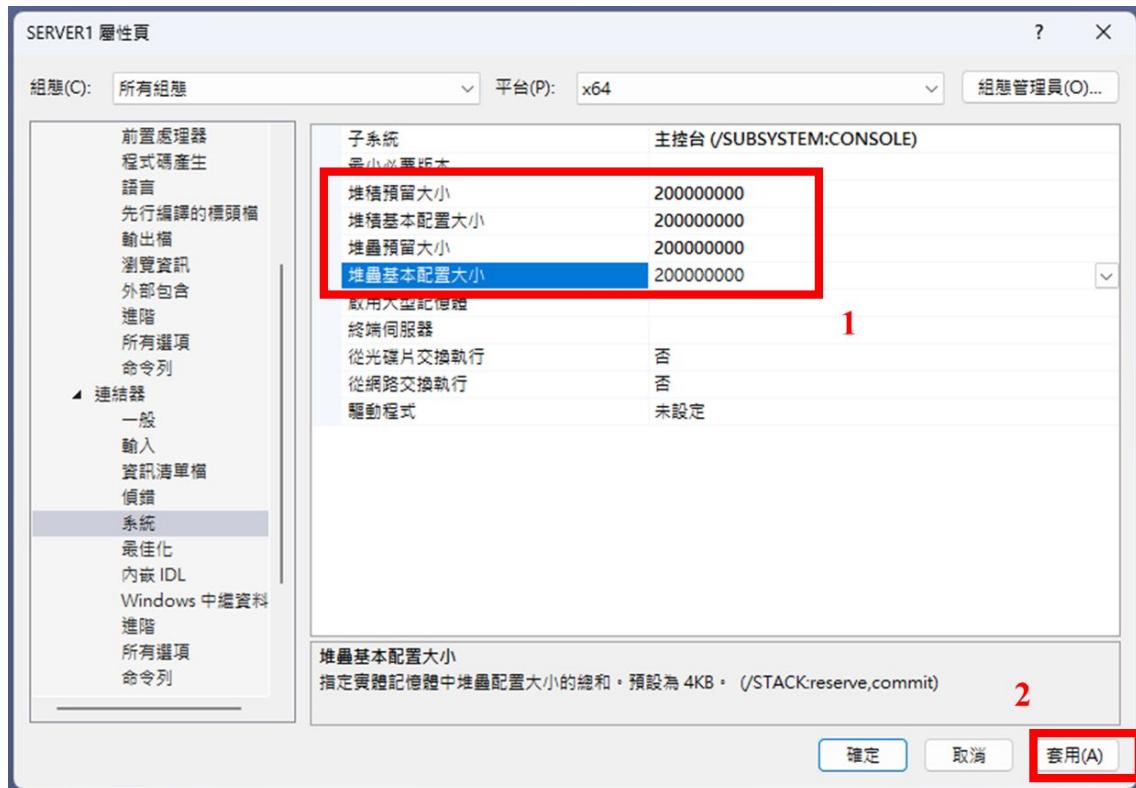
C:\程式包路徑\程式包名\include\eigen3



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 2000000000 ➔

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

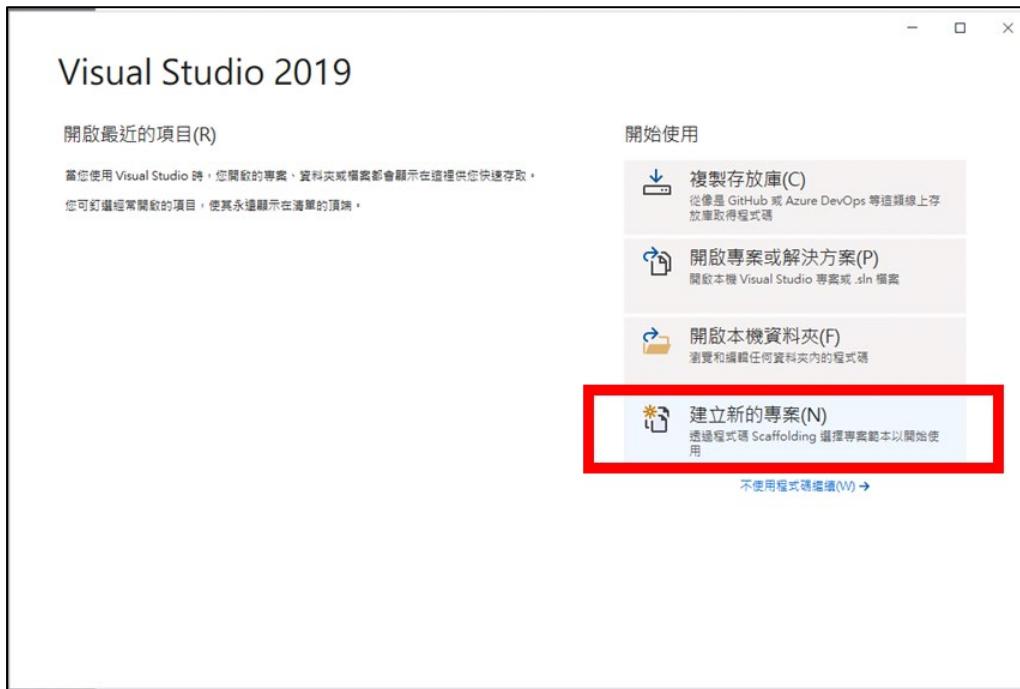


1.6 設定 Client 端上的程式

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

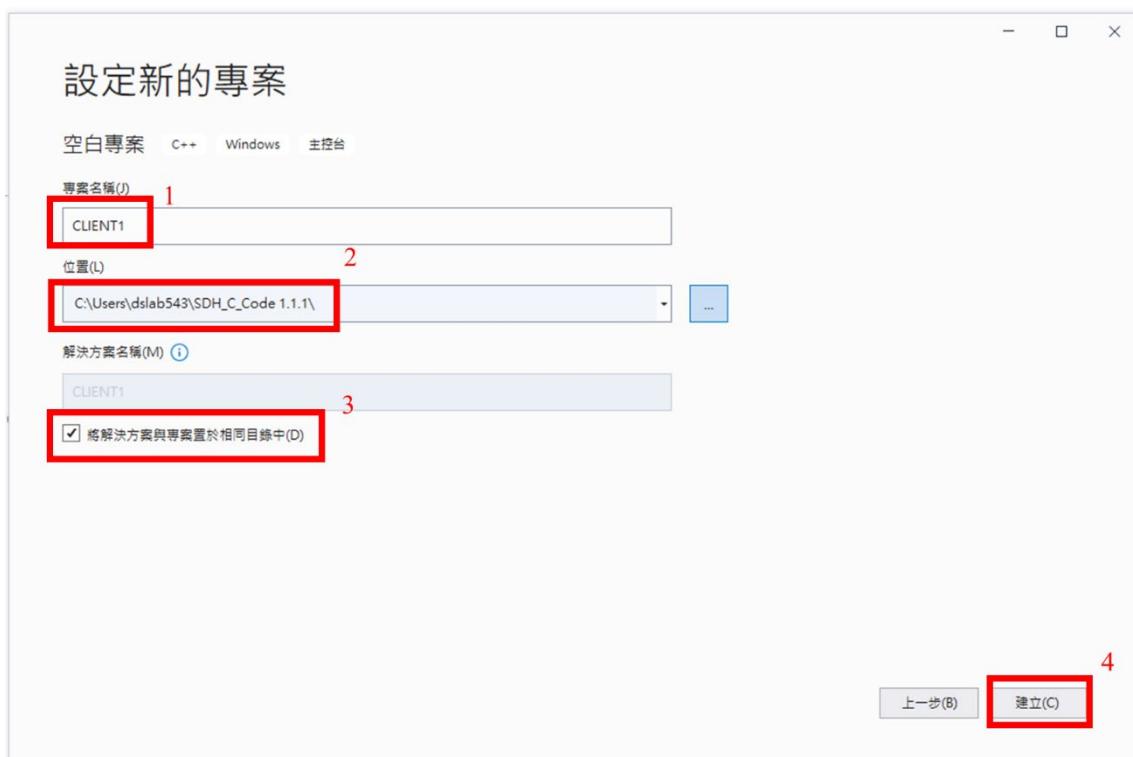
步驟 1. 建立新的專案(CLIENT1)



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



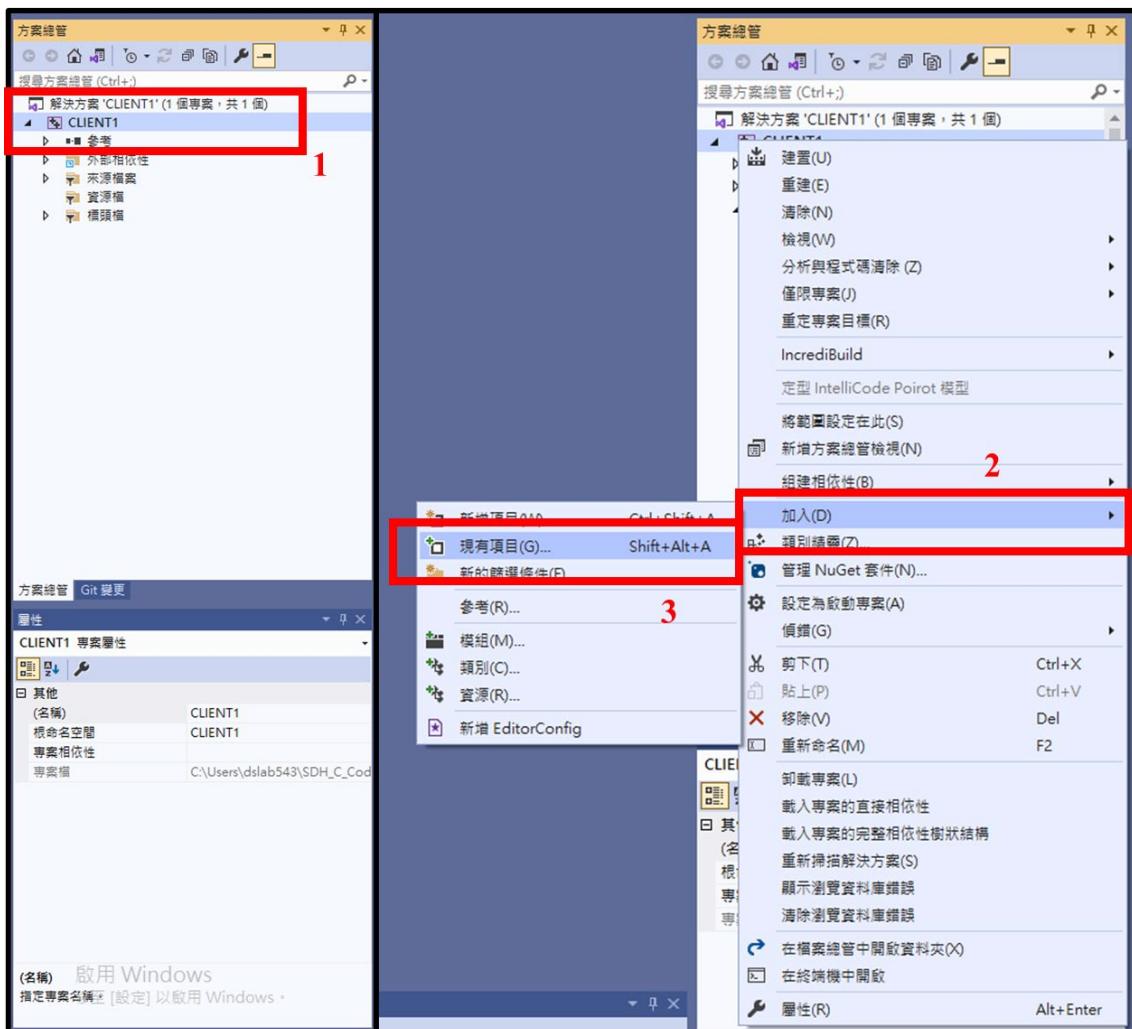
步驟 4. 將 client 端程式複製到剛創建專案 CLIENT1

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

C:\Users\dslab543\SDH_C_Code 1.1.1\CLIENT1				
	名稱	修改日期	類型	大小
sonal	GetCurrentTimeStamp_client.cpp	2023/6/26 下午 03:14	CPP 檔案	3 KB
1.0.9	GetCurrentTimeStamp_client.h	2023/6/26 下午 03:20	C/C++ Header	1 KB
1.1.1	getLabel_client.cpp	2023/6/26 下午 03:23	CPP 檔案	4 KB
	getLabel_client.h	2023/6/26 下午 03:13	C/C++ Header	1 KB
	hammingDist_client.cpp	2023/6/26 下午 03:20	CPP 檔案	3 KB
	hammingDist_client.h	2023/6/26 下午 03:20	C/C++ Header	1 KB
	LoadData_client.cpp	2023/6/26 下午 03:20	CPP 檔案	4 KB
	LoadData_client.h	2023/6/26 下午 03:20	C/C++ Header	1 KB
	main_client.cpp	2023/6/26 下午 03:47	CPP 檔案	37 KB
	main_client.h	2023/6/26 下午 03:20	C/C++ Header	2 KB
	main_client.pro	2023/6/18 下午 01:13	PRO 檔案	2 KB
	meanstd_client.cpp	2023/6/26 下午 03:20	CPP 檔案	2 KB
	meanstd_client.h	2023/6/26 下午 03:20	C/C++ Header	1 KB
	meanstdPerAggrelter_client.cpp	2023/6/21 下午 11:24	CPP 檔案	2 KB
	meanstdPerAggrelter_client.h	2023/6/21 下午 11:23	C/C++ Header	1 KB
	multilabelmaPrecision_client.cpp	2023/6/26 下午 03:20	CPP 檔案	6 KB
	multilabelmaPrecision_client.h	2023/6/26 下午 03:20	C/C++ Header	1 KB
	PWeight_client.cpp	2023/6/26 下午 03:20	CPP 檔案	5 KB
	PWeight_client.h	2023/6/26 下午 03:20	C/C++ Header	1 KB
	PWeightRRC_client.cpp	2023/6/26 下午 03:20	CPP 檔案	7 KB
	PWeightRRC_client.h	2023/6/26 下午 03:20	C/C++ Header	1 KB
	pythonRunner_client.cpp	2023/6/26 下午 03:20	CPP 檔案	2 KB
	pythonRunner_client.h	2023/6/26 下午 03:20	C/C++ Header	1 KB
	RemoveEvaLogFile_client.cpp	2023/6/23 下午 01:33	CPP 檔案	4 KB
	RemoveEvaLogFile_client.h	2023/6/19 下午 01:52	C/C++ Header	1 KB
	SDH_client.cpp	2023/6/26 下午 03:20	CPP 檔案	12 KB
	SDH_client.h	2023/6/26 下午 03:20	C/C++ Header	2 KB
	SDHWrapper_client.cpp	2023/6/26 下午 03:20	CPP 檔案	19 KB
	SDHWrapper_client.h	2023/6/26 下午 03:20	C/C++ Header	2 KB
vidia data				
	2 個項目			

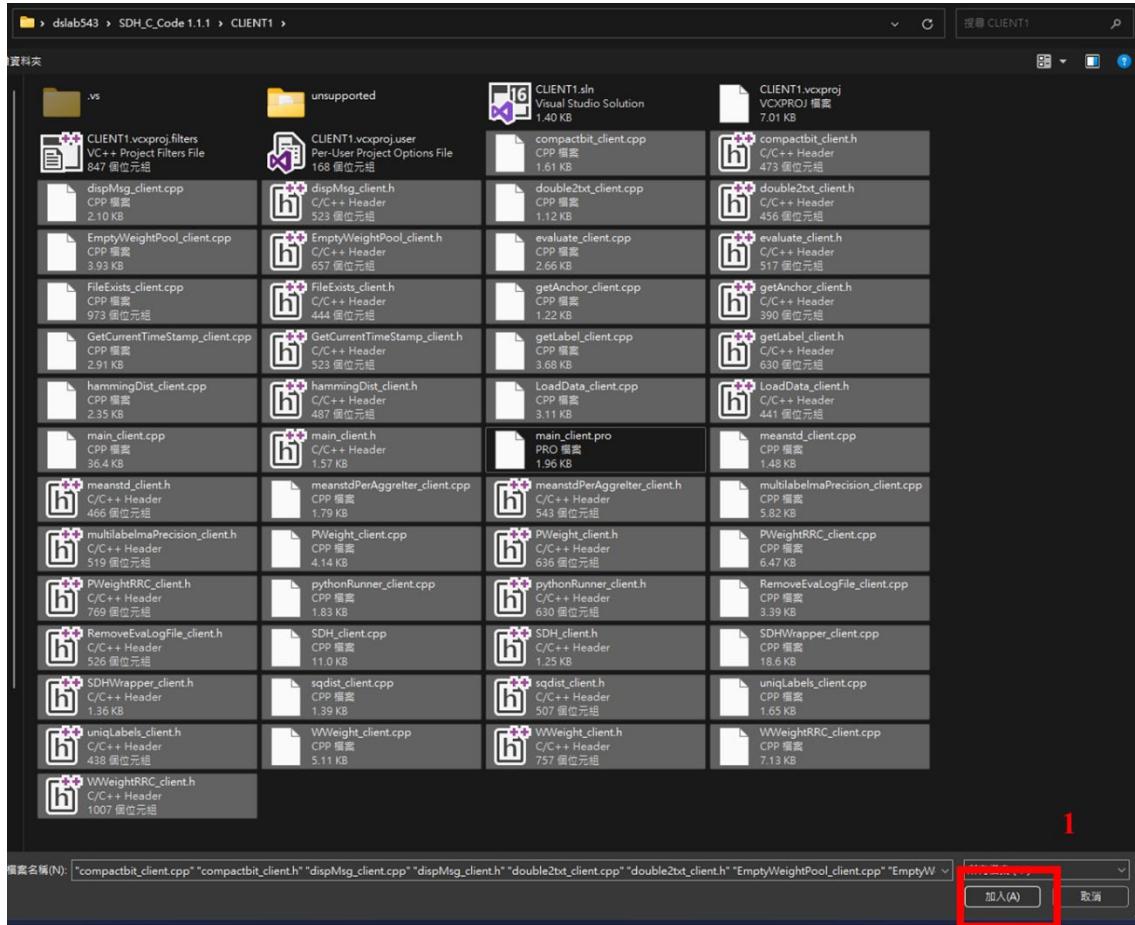
步驟 5. 加入 server 端程式加入專案 CLIENT1 中，讓專案讀取 ➤ 紅框 1：右鍵點擊旁邊專案檔 ➤ 紅框 2：加入 ➤ 紅框 3：現有項目，將 client 端程式

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)

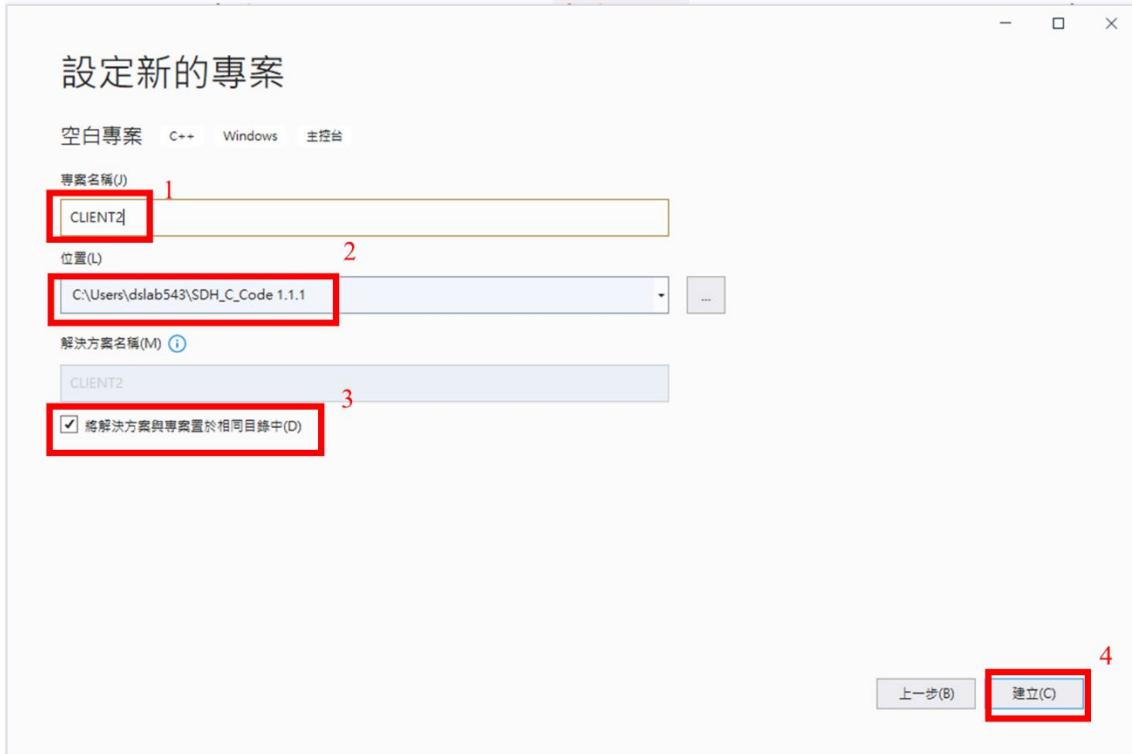
步驟 1. 建立新的專案(CLIENT2)



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

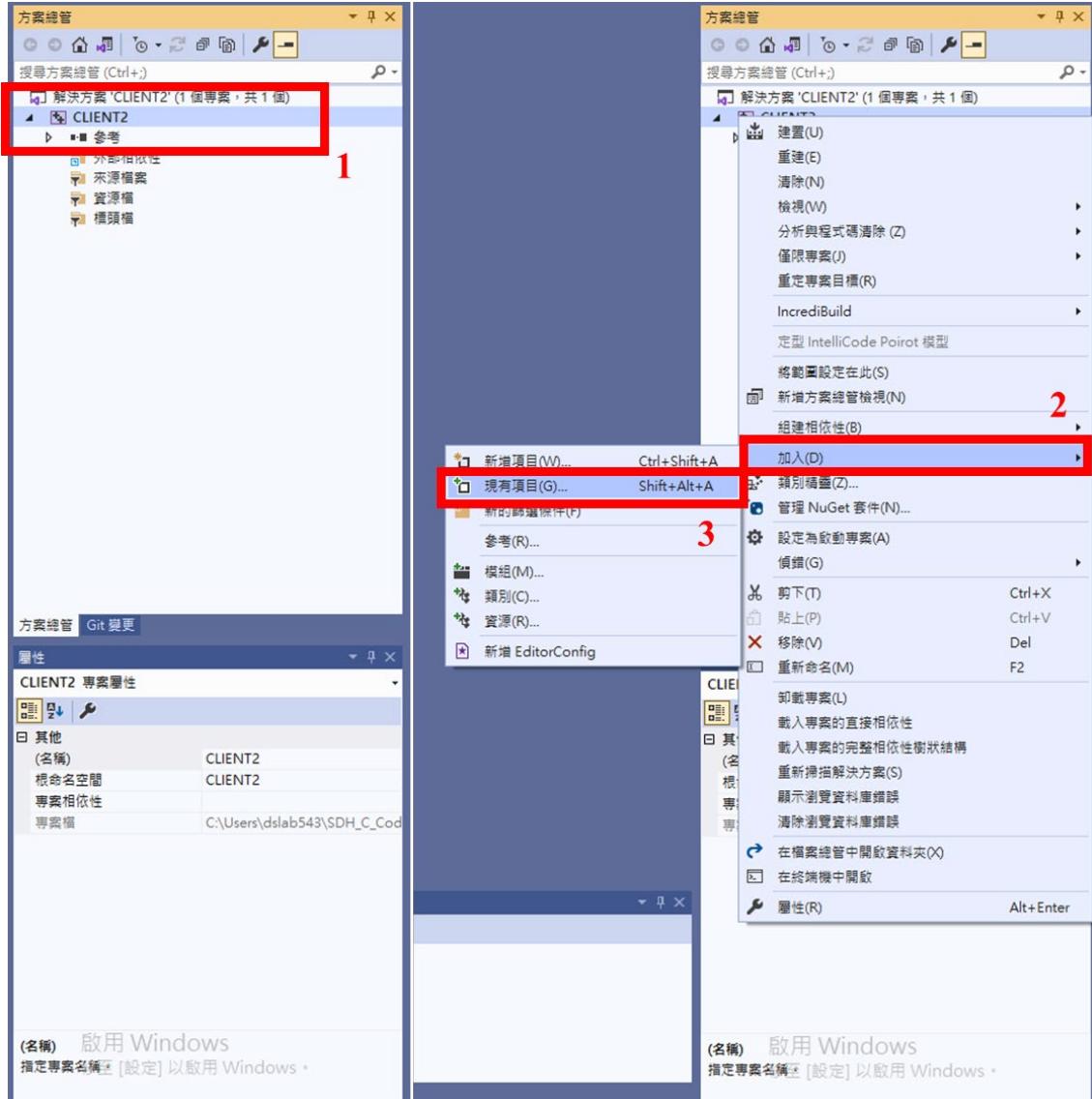
(C:\Users\dslab543\SDH_C_Code 1.1.1\CLIENT2)底下

名稱	修改日期	類型	大小
.vs	2023/6/27 下午 09:44	檔案資料夾	
unsupported	2023/6/27 下午 09:44	檔案資料夾	
CLIENT2.sln	2023/6/27 下午 09:44	Visual Studio Sol...	2 KB
CLIENT2.vcxproj	2023/6/27 下午 09:43	VCXPROJ 檔案	8 KB
CLIENT2.vcxproj.filters	2023/6/27 下午 09:44	VC++ Project Filt...	1 KB
CLIENT2.vcxproj.user	2023/6/27 下午 09:44	Per-User Project ...	1 KB
compactbit_client.cpp	2023/6/26 下午 03:14	CPP 檔案	2 KB
compactbit_client.h	2023/6/26 下午 03:16	C/C++ Header	1 KB
dispMsg_client.cpp	2023/6/23 下午 05:31	CPP 檔案	3 KB
dispMsg_client.h	2023/6/23 下午 05:26	C/C++ Header	1 KB
double2txt_client.cpp	2023/6/19 下午 03:41	CPP 檔案	2 KB
double2txt_client.h	2023/6/19 下午 03:43	C/C++ Header	1 KB
EmptyWeightPool_client.cpp	2023/6/25 下午 06:25	CPP 檔案	4 KB
EmptyWeightPool_client.h	2023/6/20 上午 10:45	C/C++ Header	1 KB
evaluate_client.cpp	2023/6/26 下午 03:14	CPP 檔案	3 KB
evaluate_client.h	2023/6/26 下午 03:20	C/C++ Header	1 KB
FileExists_client.cpp	2023/6/19 下午 03:41	CPP 檔案	1 KB
FileExists_client.h	2023/6/19 下午 03:43	C/C++ Header	1 KB
getAnchor_client.cpp	2023/6/26 下午 03:14	CPP 檔案	2 KB
getAnchor_client.h	2023/6/26 下午 03:20	C/C++ Header	1 KB
GetCurrentTimeStamp_client.cpp	2023/6/26 下午 03:14	CPP 檔案	3 KB
GetCurrentTimeStamp_client.h	2023/6/26 下午 03:20	C/C++ Header	1 KB
getLabel_client.cpp	2023/6/26 下午 03:23	CPP 檔案	4 KB
getLabel_client.h	2023/6/26 下午 03:13	C/C++ Header	1 KB
hammingDist_client.cpp	2023/6/26 下午 03:20	CPP 檔案	3 KB
hammingDist_client.h	2023/6/26 下午 03:20	C/C++ Header	1 KB
LoadData_client.cpp	2023/6/26 下午 03:20	CPP 檔案	4 KB
LoadData_client.h	2023/6/26 下午 03:20	C/C++ Header	1 KB
main_client.cpp	2023/6/26 下午 03:17	CPP 檔案	37 KB

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

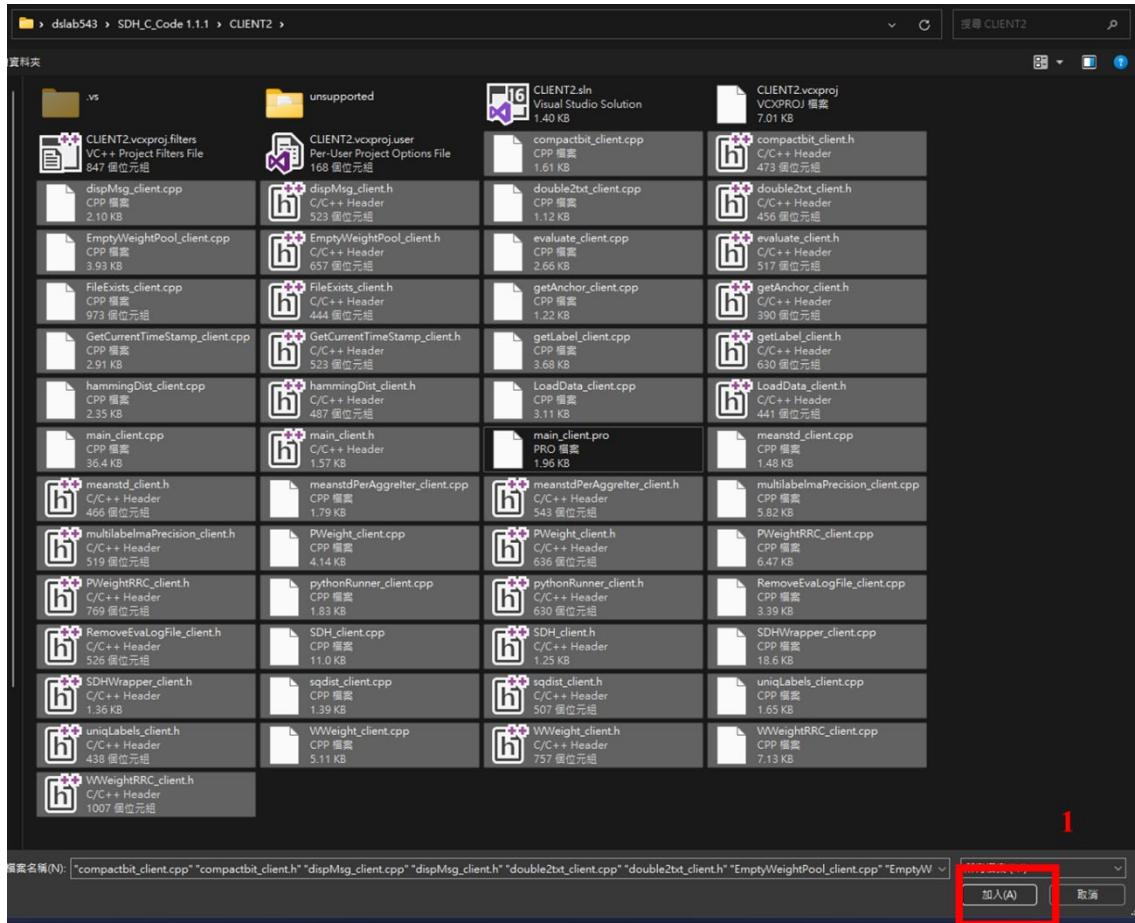
擊旁邊專案檔 ➔ 紅框 2：加入 ➔ 紅框 3：現有項目，將 client 端程式加入

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

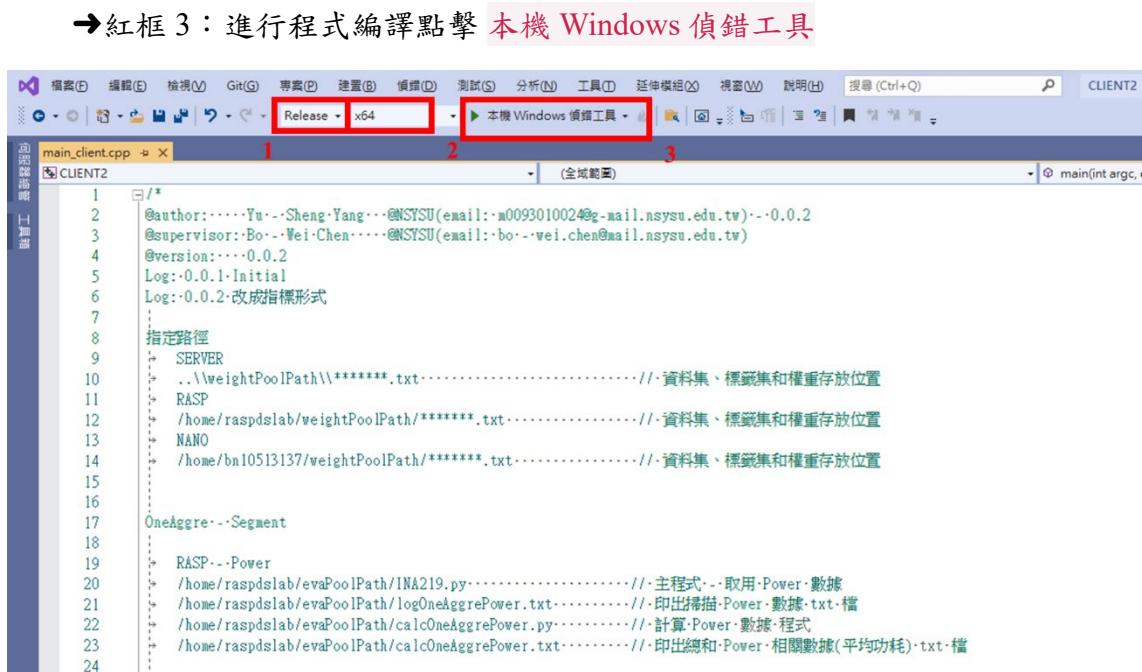
```

main_client.cpp //> X CLIENT2
124 {
125     //...參數設定-NANO_train&test_SPEED_FedProxFSDH_JAFFE_EXP10_ITRS_16B_SI1_MU10_5_1
126     string::datasetnam...:= "YALE"; //測試資料集("YALE", "JAFFE", "ORL", "SMALL_ARFACE")
127     string::SwitchDevice...:= "RASP"; //選擇在哪個裝置上進行權重更新(NANO, RASP)..
128     string::SwitchMethod...:= "AVG_FSDH"; //選擇何種聯邦學習式模型( AVG_SDH, AVG_FSDH, PROX_SDH 和 PROX_FSDH )
129     string::SwitchSegment...:= "TRAINTEST"; //選擇量測完整一回模型更新區段或是在模型訓練或模型測試區段進行( ALL 和 TRAINTEST )
130     string::SwitchEvaluation...:= "SPEED"; //選擇量測何種評估指標。( SPEED, POWER 和 MEMORY )
131     string::switchEval_METHOD:= "micro"; //Accuracy, Precision, Recall 和 F1 使用 micro 平均

main_client.cpp //> X CLIENT2
124 {
125     //...參數設定-NANO_train&test_SPEED_FedProxFSDH_JAFFE_EXP10_ITRS_16B_SI1_MU10_5_1
126     string::datasetnam...:= "YALE"; //測試資料集("YALE", "JAFFE", "ORL", "SMALL_ARFACE")
127     string::SwitchDevice...:= "NANO"; //選擇在哪個裝置上進行權重更新(NANO, RASP)..
128     string::SwitchMethod...:= "AVG_FSDH"; //選擇何種聯邦學習式模型( AVG_SDH, AVG_FSDH, PROX_SDH 和 PROX_FSDH )
129     string::SwitchSegment...:= "TRAINTEST"; //選擇量測完整一回模型更新區段或是在模型訓練或模型測試區段進行( ALL 和 TRAINTEST )
130     string::SwitchEvaluation...:= "SPEED"; //選擇量測何種評估指標。( SPEED, POWER 和 MEMORY )
131     string::switchEval_METHOD:= "micro"; //Accuracy, Precision, Recall 和 F1 使用 micro 平均

```

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



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**

```
main_server.cpp // SERVER1
92 extern int main(int argc, char **argv)
93 {
94     // 參數設定
95     string datasetnam = "SMALL_ARFACE"; 1 // 輸入需測試資料集("YALE", "JAFFE", "ORL", "SMALL_ARFACE")
96     vector<int> bitlen = {-16, -32, 64, 96, 128}; // 程式中須測試的主要碼長設定在param.L
97     int nChannels = 1; // if RGB, nChannels = -3
98     param.nExps = 1; // 執行幾回完整的模型更新
99     param.J = 0; // 鏡點
100    param.L = 64; // 編碼長度(先宣告, 64為預設值)
101    param.AggreIter = 5; // 最大同服聯合輪次

main_client1.cpp // CLIENT1
123 extern int main(int argc, char **argv)
124 {
125     // 參數設定-NANO_train&test_SPEED_EndProvPSDH_JAFFE_EXP10_1TR5_16B_SI1_MU10_5_1
126     string datasetnam = "SMALL_ARFACE"; 1 // 測試資料集("YALE", "JAFFE", "ORL", "SMALL_ARFACE")
127     string SwitchDevice = "RASP"; // 選擇在哪個裝置上進行權重更新(NANO-, RASP)
128     string SwitchMethod = "AVG_FSDH"; // 選擇何種聯邦學習式模型(AVG_SDH, AVG_FSDH, PROX_SDH 和 PROX_FSDH)
129     string SwitchSegment = "TRAINTEST"; // 選擇量測完整一回模型更新區段或是在模型訓練或模型測試區段進行(ALL 和 TRAINTEST)
130     string SwitchEvaluation = "SPEED"; // 選擇量測何種評估指標-(SPEED, POWER 和 MEMORY)
131     string switchEval_METHOD = "micro"; // Accuracy, Precision, Recall 和 F1 使用 micro 平均
132     vector<int> bitlen = {-16, -32, 64, 96, 128}; // 程式中須測試的碼長, 主要碼長設定在param.L
133     int nChannels = 1; // if RGB, nChannels = -3
134     param.nExps = 1; // 執行幾回完整的模型更新
135     param.J = 0; // 鏡點
136     param.L = 32; // 編碼長度(先宣告, 32為預設值)
137     param.AggreIter = 5; // 最大同服聯合輪次

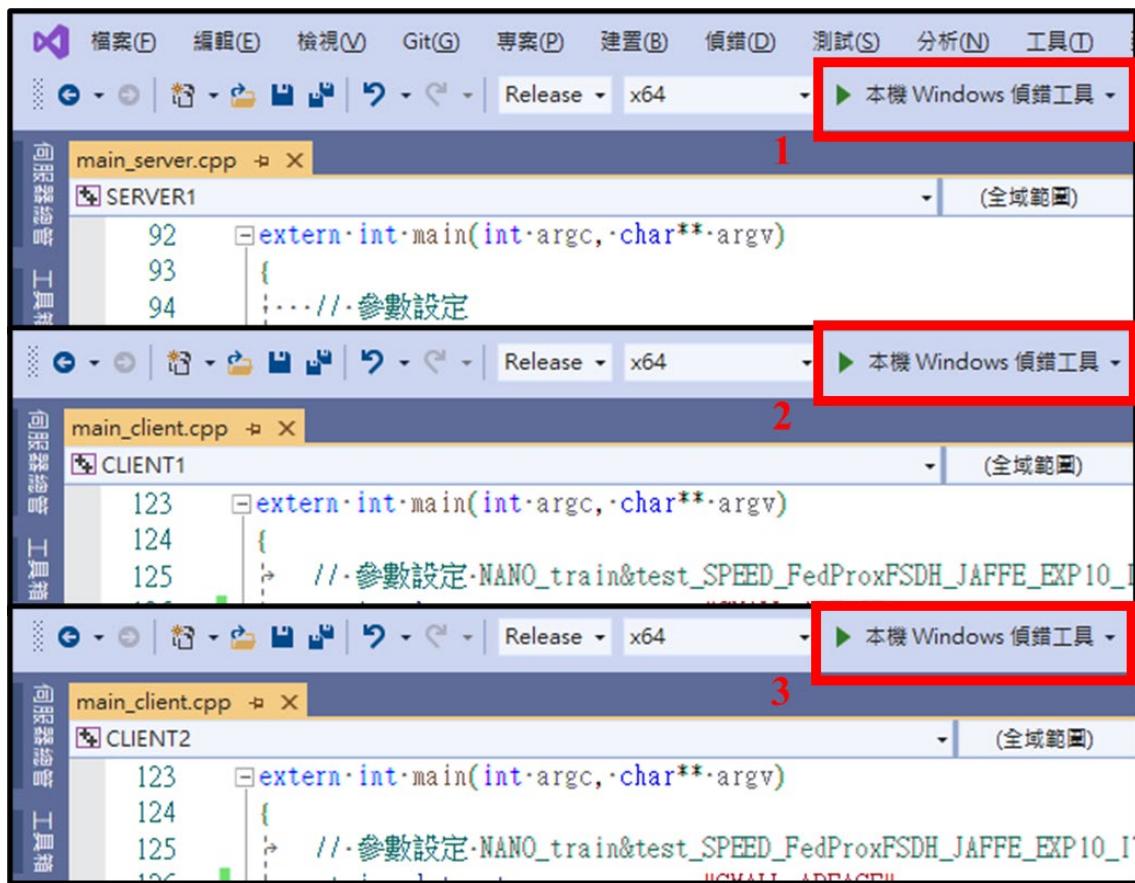
main_client2.cpp // CLIENT2
123 extern int main(int argc, char **argv)
124 {
125     // 參數設定-NANO_train&test_SPEED_EndProvPSDH_JAFFE_EXP10_1TR5_16B_SI1_MU10_5_1
126     string datasetnam = "SMALL_ARFACE"; 1 // 測試資料集("YALE", "JAFFE", "ORL", "SMALL_ARFACE")
127     string SwitchDevice = "NANO"; // 選擇在哪個裝置上進行權重更新(NANO-, RASP)
128     string SwitchMethod = "AVG_FSDH"; // 選擇何種聯邦學習式模型(AVG_SDH, AVG_FSDH, PROX_SDH 和 PROX_FSDH)
129     string SwitchSegment = "TRAINTEST"; // 選擇量測完整一回模型更新區段或是在模型訓練或模型測試區段進行(ALL 和 TRAINTEST)
130     string SwitchEvaluation = "SPEED"; // 選擇量測何種評估指標-(SPEED, POWER 和 MEMORY)
131     string switchEval_METHOD = "micro"; // Accuracy, Precision, Recall 和 F1 使用 micro 平均
132     vector<int> bitlen = {-16, -32, 64, 96, 128}; // 程式中須測試的碼長, 主要碼長設定在param.L
133     int nChannels = 1; // if RGB, nChannels = -3
134     param.nExps = 1; // 執行幾回完整的模型更新
135     param.J = 0; // 鏡點
136     param.L = 32; // 編碼長度(先宣告, 32為預設值)
137     param.AggreIter = 5; // 最大同服聯合輪次
```

步驟 2. 檢查專案 SERVER1 裡的資料集路徑是否正確

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

步驟 3. 依序點擊專案 SERVER1、CLIENT1 和 CLIENT2 本機 Windows 偵錯工具

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

SERVER1

```

成功刪除檔案 : ".../weightPoolPath\\SERVER2RASP_0_1_WWeight_matrix.txt"
成功刪除檔案 : ".../weightPoolPath\\SERVER2RASP_0_2_PWeight_matrix.txt"
成功刪除檔案 : ".../weightPoolPath\\SERVER2RASP_0_3_WWeight_matrix.txt"
成功刪除檔案 : ".../weightPoolPath\\SERVER2RASP_0_3_PWeight_matrix.txt"
成功刪除檔案 : ".../weightPoolPath\\SERVER2RASP_0_4_PWeight_matrix.txt"
成功刪除檔案 : ".../weightPoolPath\\SERVER2RASP_0_4_WWeight_matrix.txt"
成功刪除檔案 : ".../weightPoolPath\\SERVER2RASP_0_L_test_matrix.txt"
成功刪除檔案 : ".../weightPoolPath\\SERVER2RASP_0_L_tune_matrix.txt"
成功刪除檔案 : ".../weightPoolPath\\SERVER2RASP_0_X_test_matrix.txt"
成功刪除檔案 : ".../weightPoolPath\\SERVER2RASP_0_X_tune_matrix.txt"
成功刪除檔案 : ".../weightPoolPath\\SERVER2RASP_0_X_tran_matrix.txt"
文件夾 ..../weightPoolPath 中的所有文件已刪除。
$$$$$$$$$$
Server_assign_data
$$$$$$$$$$
SERVER_SEND_TRAIN_TO_NODE
第0_0次更新    wait a minutes until NODE send file
NODE_SEND_WEIGHT_TO_SERVER
第1次等待
2023-06-27 23:07:12:584:639:0
$$$$$$$$$$
Server_Aggregation_Matrix
$$$$$$$$$$
$$$$$$$$$$
Server_Assign_NEW_Matrix
$$$$$$$$$$
SERVER_SEND_AGGR_TO_NODE
2023-06-27 23:07:16:856:773:900
$$$$$$$$$$
Server_Aggregation_Matrix
$$$$$$$$$$
$$$$$$$$$$
Server_Assign_NEW_Matrix
$$$$$$$$$$
SERVER_SEND_AGGR_TO_NODE
2023-06-27 23:07:21:139:948:500
$$$$$$$$$$
Server_Aggregation_Matrix
$$$$$$$$$$
$$$$$$$$$$
Server_Assign_NEW_Matrix
$$$$$$$$$$
SERVER_SEND_AGGR_TO_NODE
2023-06-27 23:07:25:408:574:100
$$$$$$$$$$
Server_Aggregation_Matrix
$$$$$$$$$$
$$$$$$$$$$
Server_Assign_NEW_Matrix

```

CLIENT1

```

AVG_FSDH TestTime (ms)      mean : 199.9650100 std : 21.8422900
AVG_FSDH OneAggreTime (ms)  mean : 4648.8802500 std : 3516.9895837
TESTING_Evaluate
AVG_FSDH ACCURACY          mean : 0.9542124 std : 0.0158730
AVG_FSDH PRECISION          mean : 0.7856045 std : 0.0771406
AVG_FSDH RECALL             mean : 0.9529915 std : 0.0128205
AVG_FSDH F1                 mean : 0.8585777 std : 0.0042178
AVG_FSDH mAP                mean : 0.7935228 std : 0.0684675
AVG_FSDH TranTime (ms)      mean : 1683.5960000 std : 15.4618000
AVG_FSDH TestTime (ms)      mean : 198.3596500 std : 25.8355500
AVG_FSDH OneAggreTime (ms)  mean : 3588.2635000 std : 202.8724000
DataName : SMALL_ARFACE
SwitchMethod : AVG_FSDH
nExps : 1
L : 32
AggreIter : 5
LocalIteration : 3
RBFSIGMA : 2
HAMMRADIUS : 2
ProxMu : 1.0e-07
The elapsed time for Experiment is 29666.5 ms
The one time for Experiment is 5933.3 ms
2023-06-27 23:06:40:90:804:600

```

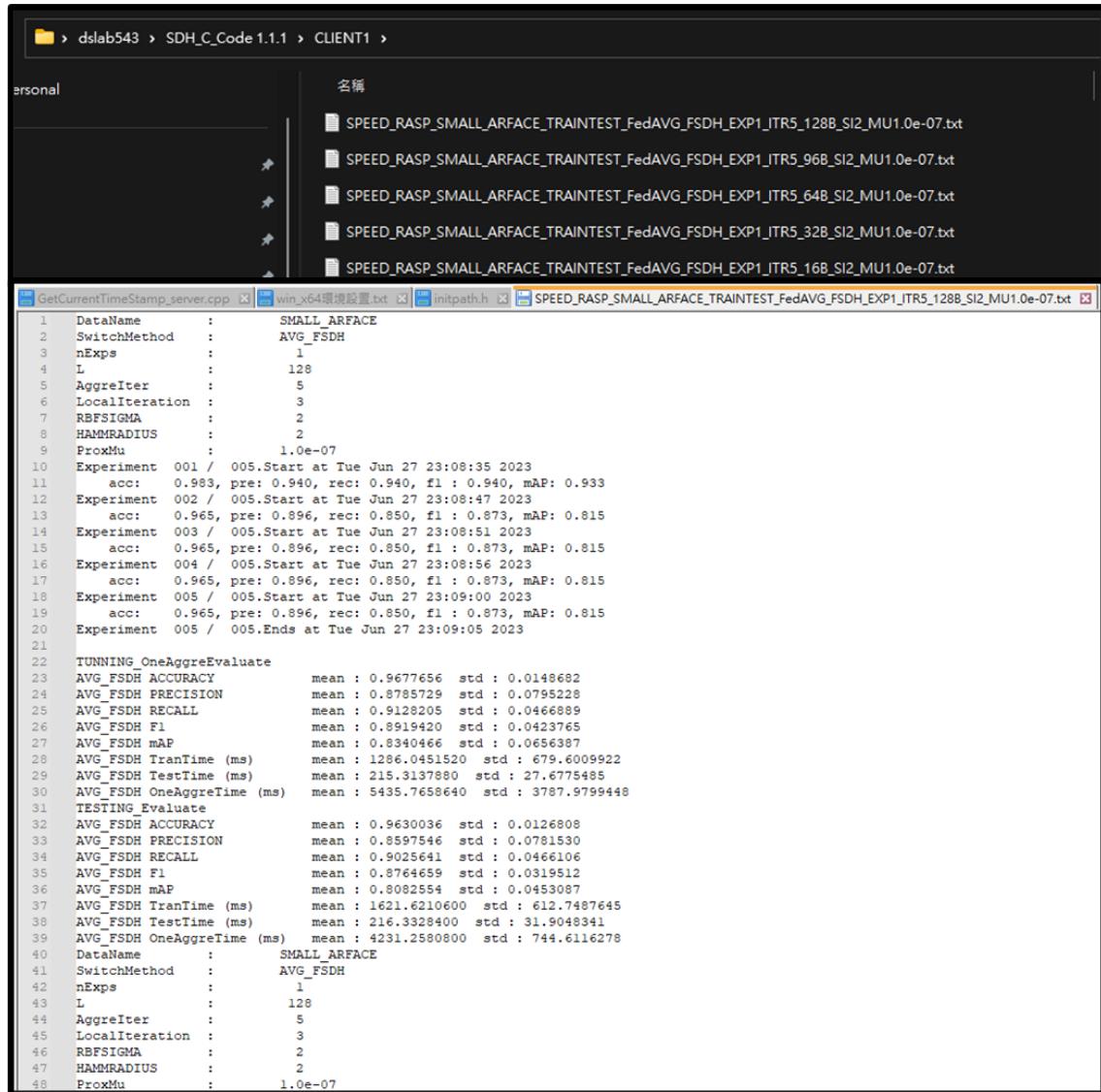
CLIENT2

```

AVG_FSDH mAP                mean : 0.8129274 std : 0.0547533
AVG_FSDH TranTime (ms)      mean : 7110.6838700 std : 17177.4171566
AVG_FSDH TestTime (ms)      mean : 210.6906300 std : 35.0132697
AVG_FSDH OneAggreTime (ms)  mean : 16530.2190400 std : 17625.6950447
TESTING_Evaluate
AVG_FSDH ACCURACY          mean : 0.9636752 std : 0.0076312
AVG_FSDH PRECISION          mean : 0.8395612 std : 0.0583112
AVG_FSDH RECALL             mean : 0.9316239 std : 0.0299145
AVG_FSDH F1                 mean : 0.8809279 std : 0.0188580
AVG_FSDH mAP                mean : 0.7899068 std : 0.0331305
AVG_FSDH TranTime (ms)      mean : 2132.5700500 std : 2.5159500
AVG_FSDH TestTime (ms)      mean : 219.1686000 std : 39.8525000
AVG_FSDH OneAggreTime (ms)  mean : 4620.6108000 std : 226.8018000
DataName : SMALL_ARFACE
SwitchMethod : AVG_FSDH
nExps : 1
L : 32
AggreIter : 5
LocalIteration : 3
RBFSIGMA : 2
HAMMRADIUS : 2
ProxMu : 1.0e-07
The elapsed time for Experiment is 29431.5 ms
The one time for Experiment is 5886.31 ms
2023-06-27 23:06:40:669:848:300

```

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。



The screenshot shows a Windows file explorer window and a code editor window side-by-side.

In the file explorer window, the path is `dslab543 > SDH_C_Code 1.1.1 > CLIENT1 >`. The right pane lists several text files:

- SPEED_RASP_SMALL_ARFACE_TRAINTEST_FedAVG_FSDH_EXP1_ITRS_128B_SI2_MU1.0e-07.txt
- SPEED_RASP_SMALL_ARFACE_TRAINTEST_FedAVG_FSDH_EXP1_ITRS_96B_SI2_MU1.0e-07.txt
- SPEED_RASP_SMALL_ARFACE_TRAINTEST_FedAVG_FSDH_EXP1_ITRS_64B_SI2_MU1.0e-07.txt
- SPEED_RASP_SMALL_ARFACE_TRAINTEST_FedAVG_FSDH_EXP1_ITRS_32B_SI2_MU1.0e-07.txt
- SPEED_RASP_SMALL_ARFACE_TRAINTEST_FedAVG_FSDH_EXP1_ITRS_16B_SI2_MU1.0e-07.txt

The code editor window displays a C++ file named `GetCurrentTimeStamp_server.cpp`. The code contains configuration parameters and experimental results. Key sections include:

- Configuration parameters (lines 1-21):

```
1 DataName      :      SMALL_ARFACE
2 SwitchMethod   :      AVG_FSDH
3 nExps          :      1
4 L              :      128
5 AggreIter     :      5
6 LocalIteration :      3
7 RBFSIGMA       :      2
8 HAMMRADIUS    :      2
9 ProxMu         :      1.0e-07
```
- Experimental results (lines 11-20):

```
11 Experiment 001 / 005.Start at Tue Jun 27 23:08:35 2023
12 acc: 0.983, pre: 0.940, rec: 0.940, f1 : 0.940, mAP: 0.933
13 Experiment 002 / 005.Start at Tue Jun 27 23:08:47 2023
14 acc: 0.965, pre: 0.896, rec: 0.850, f1 : 0.873, mAP: 0.815
15 Experiment 003 / 005.Start at Tue Jun 27 23:08:51 2023
16 acc: 0.965, pre: 0.896, rec: 0.850, f1 : 0.873, mAP: 0.815
17 Experiment 004 / 005.Start at Tue Jun 27 23:08:56 2023
18 acc: 0.965, pre: 0.896, rec: 0.850, f1 : 0.873, mAP: 0.815
19 Experiment 005 / 005.Start at Tue Jun 27 23:09:00 2023
20 acc: 0.965, pre: 0.896, rec: 0.850, f1 : 0.873, mAP: 0.815
21 Experiment 005 / 005.Ends at Tue Jun 27 23:09:05 2023
```
- Performance metrics (lines 22-30):

```
22 TUNNING_OneAggreEvaluate
23 AVG_FSDH ACCURACY      mean : 0.9677656 std : 0.0148682
24 AVG_FSDH PRECISION     mean : 0.8785729 std : 0.0795228
25 AVG_FSDH RECALL        mean : 0.9128205 std : 0.0466889
26 AVG_FSDH F1             mean : 0.8919420 std : 0.0423765
27 AVG_FSDH mAP            mean : 0.8340466 std : 0.0656387
28 AVG_FSDH TranTime (ms) mean : 1286.0451520 std : 679.6009922
29 AVG_FSDH TestTime (ms) mean : 215.3137880 std : 27.6775485
30 AVG_FSDH OneAggreTime (ms) mean : 5435.7658640 std : 3787.9799448
```
- Testing evaluation (lines 31-40):

```
31 TESTING Evaluate
32 AVG_FSDH ACCURACY      mean : 0.9630036 std : 0.0126808
33 AVG_FSDH PRECISION     mean : 0.8597546 std : 0.0781530
34 AVG_FSDH RECALL        mean : 0.9025641 std : 0.0466106
35 AVG_FSDH F1             mean : 0.8764659 std : 0.0319512
36 AVG_FSDH mAP            mean : 0.8082554 std : 0.0453087
37 AVG_FSDH TranTime (ms) mean : 1621.6210600 std : 612.7487645
38 AVG_FSDH TestTime (ms) mean : 216.3328400 std : 31.9048341
39 AVG_FSDH OneAggreTime (ms) mean : 4231.2580800 std : 744.6116278
```
- Final configuration (lines 41-48):

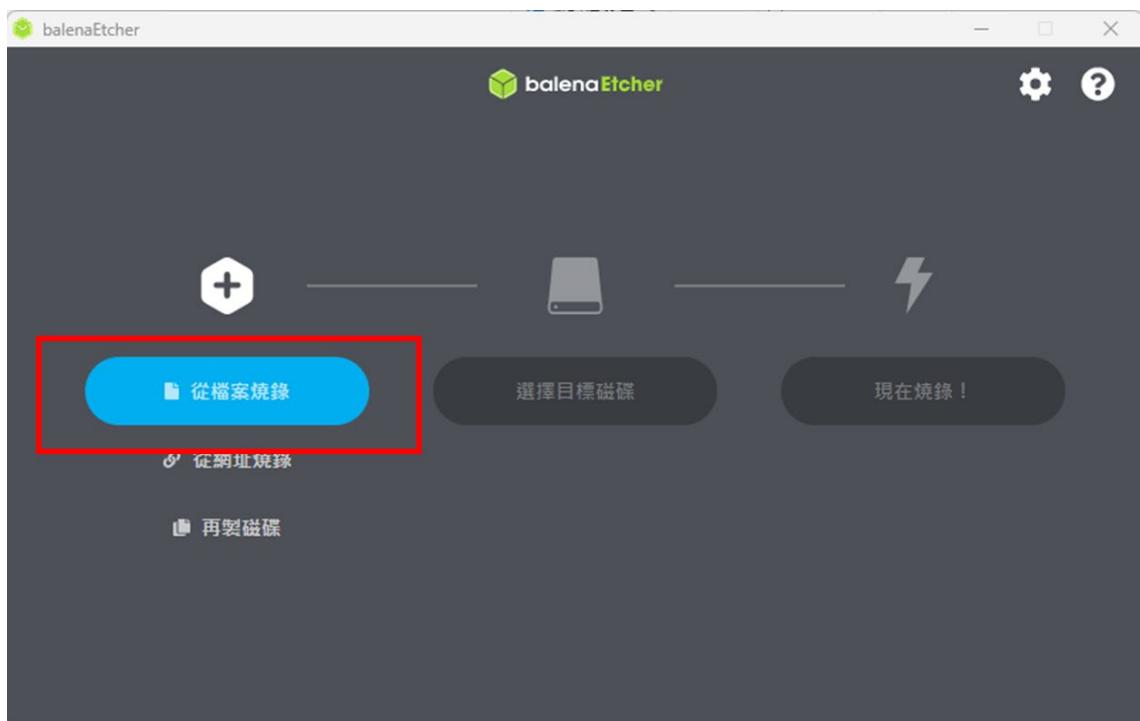
```
40 DataName      :      SMALL_ARFACE
41 SwitchMethod   :      AVG_FSDH
42 nExps          :      1
43 L              :      128
44 AggreIter     :      5
45 LocalIteration :      3
46 RBFSIGMA       :      2
47 HAMMRADIUS    :      2
48 ProxMu         :      1.0e-07
```

嵌入式版安裝步驟

2 進行 Raspberry Pi4 及 Jetson Nano 的環境設定

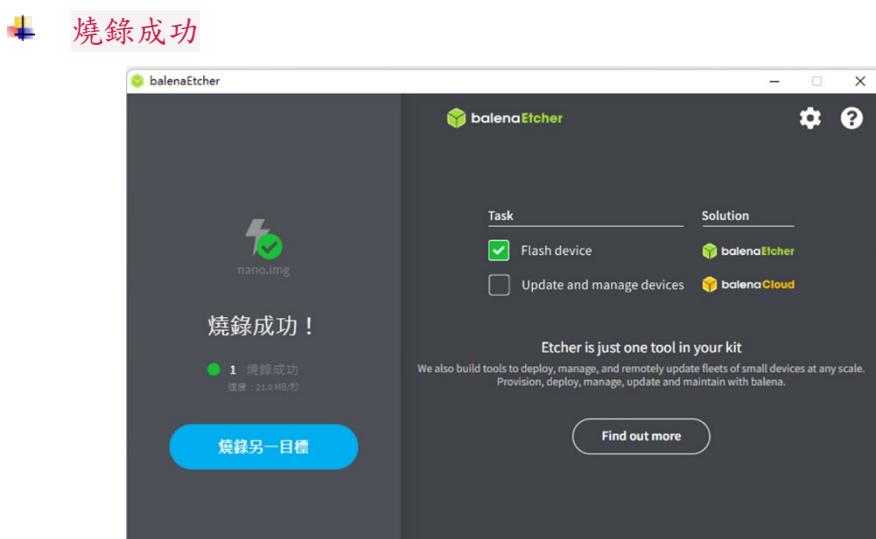
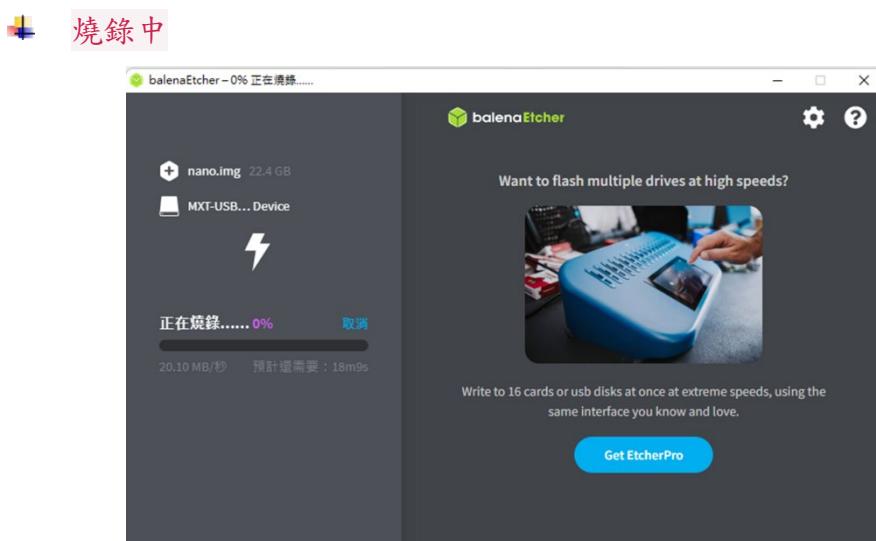
2.1 燒錄 Raspberry Pi4 及 Jetson Nano 的映像檔

- 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





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

Jetson Nano 的 OS

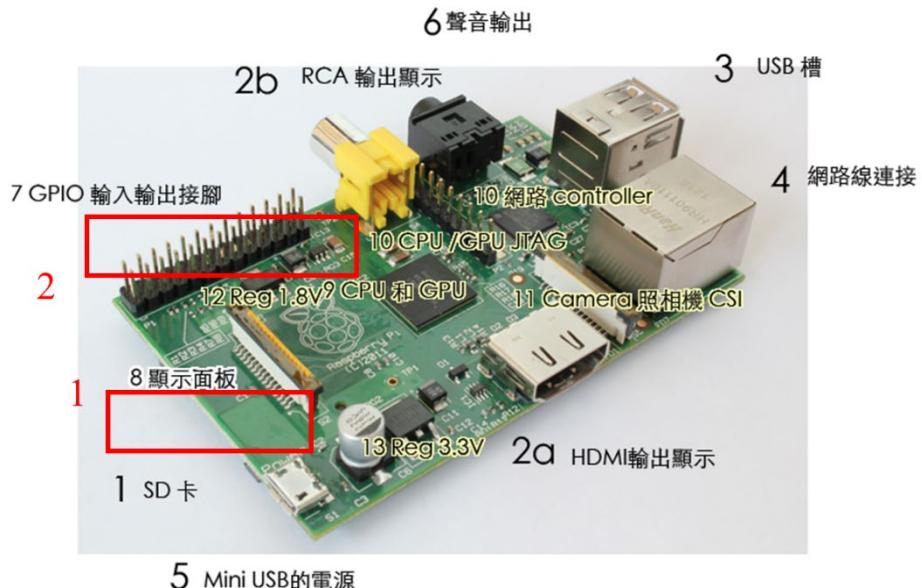
預設帳號 : bn10513137 密碼 : 6517

2.2 設定 Client 端程式在嵌入式版上

Raspberry Pi 4

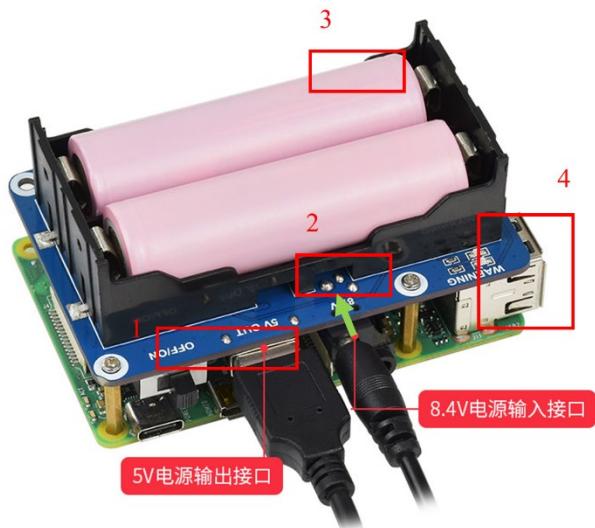
步驟 1. 組裝設備 ➤ 紅框 1：裝入燒錄完成的 SD 卡(需確認為 Raspberry Pi 4 的

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 WiFi 圖像 ➡ 紅框 2：點擊欲連接的 WiFi

名稱 ➔ 紅框 3：輸入 WiFi 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
```

```

raspdslab@raspberrypi:~ $ sudo apt-get update
已有:1 http://security.debian.org/debian-security bullseye-security InRelease
已有:2 http://deb.debian.org/debian bullseye InRelease
已有:3 http://deb.debian.org/debian bullseye-updates InRelease
已有:4 http://archive.raspberrypi.org/debian bullseye InRelease
下載:5 https://pkgs.tailscale.com/stable/debian bullseye InRelease
取得 6,525 B 用了 2s (4,109 B/s)
正在讀取套件清單... 完成
raspdslab@raspberrypi:~ $ sudo apt-get upgrade
正在讀取套件清單... 完成
正在重建相依關係... 完成
正在讀取狀態資料... 完成
籌備升級中... 完成
升級 0 個，新安裝 0 個，移除 0 個，有 0 個未被升級。
raspdslab@raspberrypi:~ $ sudo apt-get clean
raspdslab@raspberrypi:~ $ sudo apt-get autoremove
正在讀取套件清單... 完成
正在重建相依關係... 完成
正在讀取狀態資料... 完成
升級 0 個，新安裝 0 個，移除 0 個，有 0 個未被升級。
raspdslab@raspberrypi:~ $ 

```

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

Install with one command

```
curl -fSSL https://tailscale.com/install.sh | sh
```

[View script source](#)

Manually install on Raspberry Pi Bullseye ▾

Tailscale can run on Raspberry Pi boards running Raspbian. Packages are available in both 32-bit and 64-bit variants.

- 1 Install the `apt-transport-https` plugin:

```
sudo apt-get install apt-transport-https
```
- 2 Add Tailscale's package signing key and repository:

```
curl -fSSL https://pkgs.tailscale.com/stable/raspbian/bullseye.gpg | gpg --dearmor > /etc/apt/trusted.gpg.d/tailscale.gpg  
curl -fSSL https://pkgs.tailscale.com/stable/raspbian/bullseye.list | tee /etc/apt/sources.list.d/tailscale.list
```
- 3 Install Tailscale:

```
sudo apt-get update  
sudo apt-get install tailscale
```
- 4 Connect your machine to your Tailscale network and authenticate in your browser:

```
sudo tailscale up
```
- 5 You're connected! You can find your Tailscale IPv4 address by running:

```
tailscale ip -4
```

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  
          TX packets 305 bytes 34163 (33.3 KiB)  
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
2  
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 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
```

```
dslab8011@DESKTOP-BJ570R6:/mnt/c/Users/Dslab/Desktop/demol/ServerDemo1
$ ssh raspdslab@100.82.153.59
Linux raspberrypi 6.1.21-v8+ #1642 SMP PREEMPT Mon Apr  3 17:24:16 BST
2023 aarch64

The programs included with the Debian GNU/Linux system are free softwa
re;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

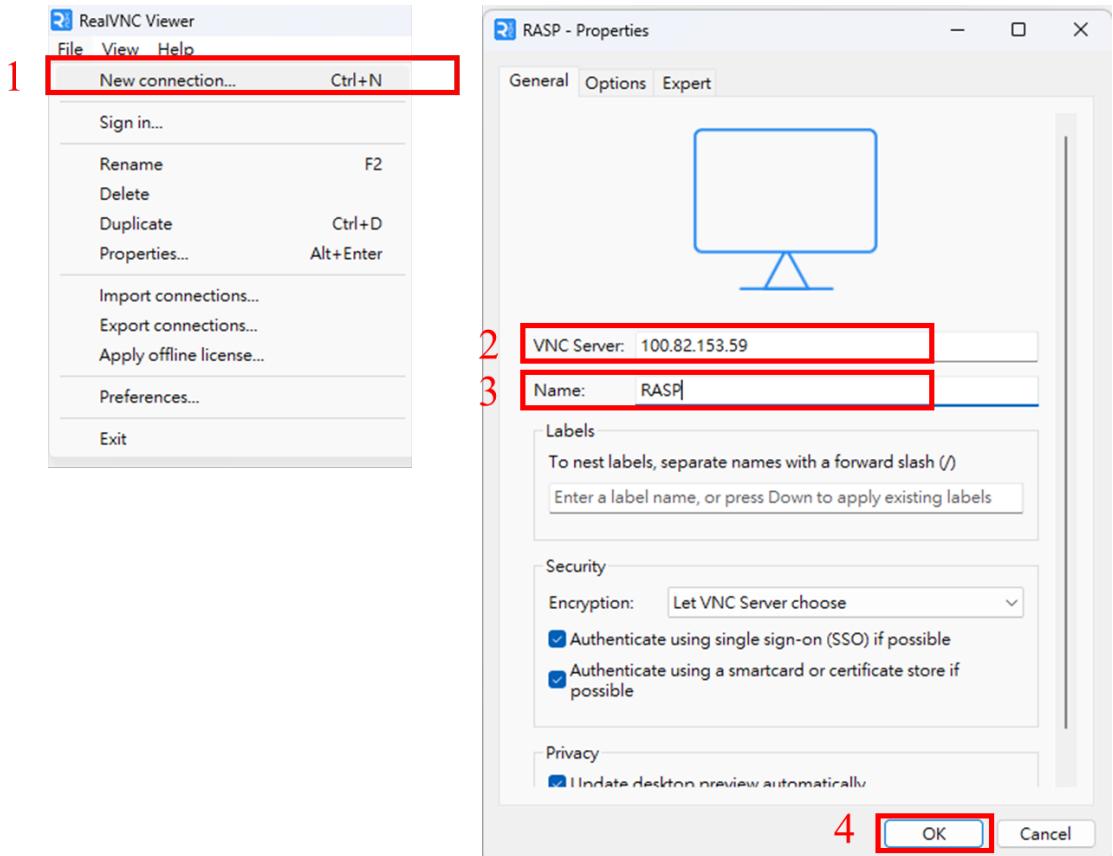
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Jul 13 14:14:21 2023 from 100.97.100.48
raspdslab@raspberrypi:~ $
```

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](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

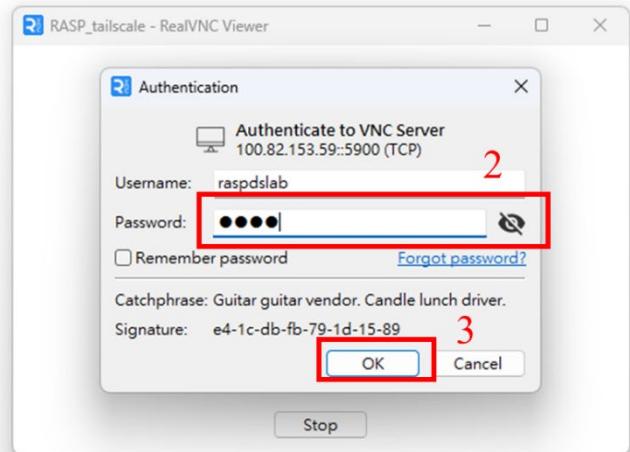
RASP 的密碼 ➔ 紅框 3：OK 。第二張圖為 VNC 連線後畫面



1

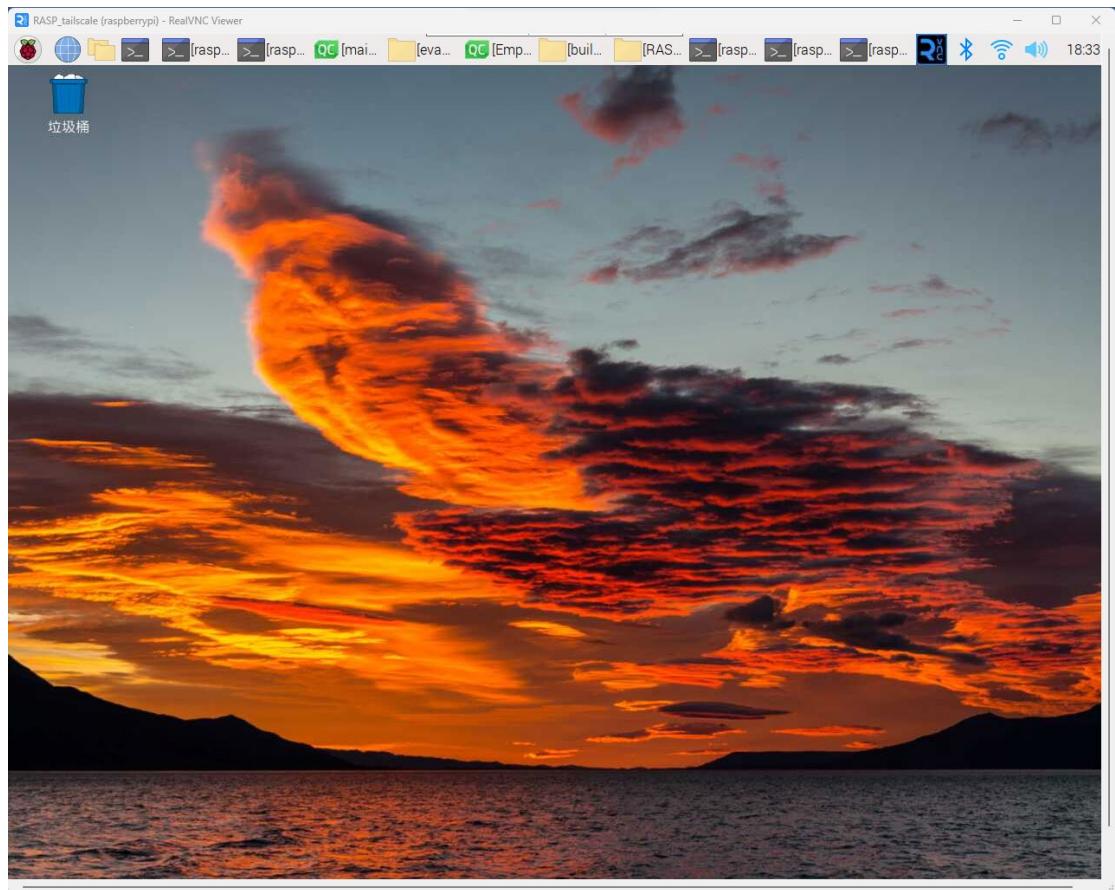


raspdslabETH0



2

3



步驟 13. 安裝 Qt - 5 程式

更新套件

```
sudo apt-get update
```

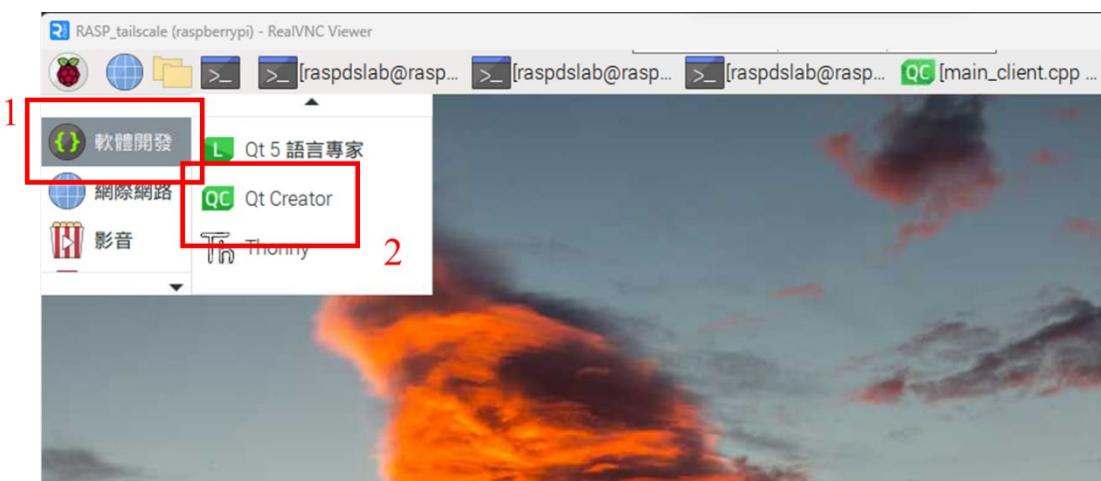
`sudo apt-get upgrade`

#安裝 Qt 套件

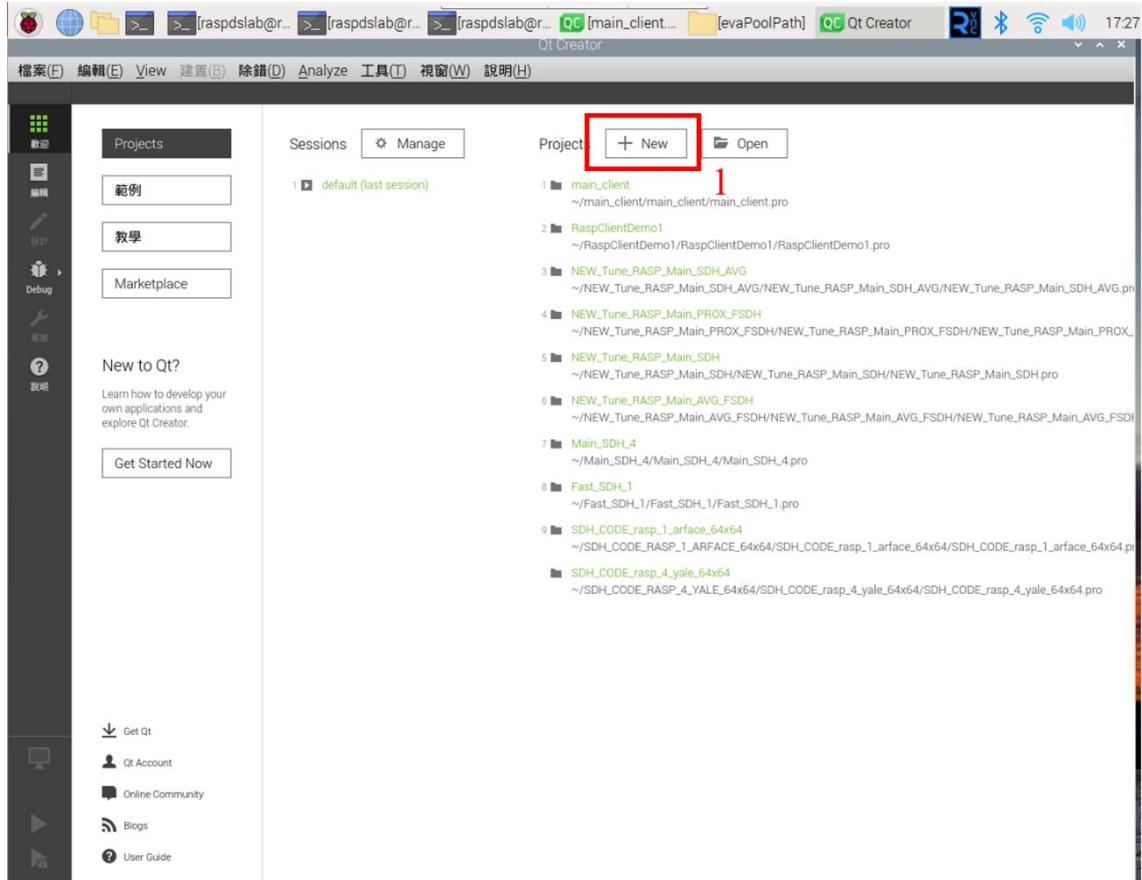
```
sudo apt-get install -y qt5-default
```

```
sudo apt-get install -y qtcreator
```

步驟 14. 開啟 Qt-5 程式的 Qt Creator 編輯器

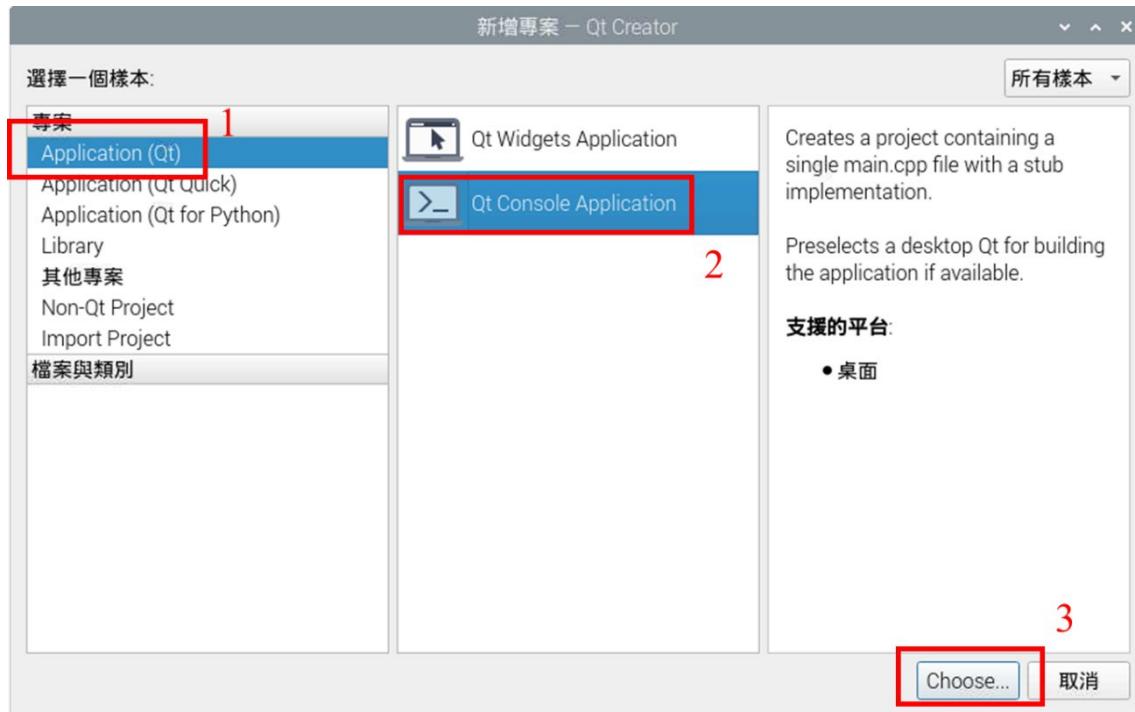


步驟 15. 創建新專案 + New



步驟 16. 創建專案 ➤ 紅框 1：選取 Application(Qt) ➔ 紅框 2：選取 Qt Console

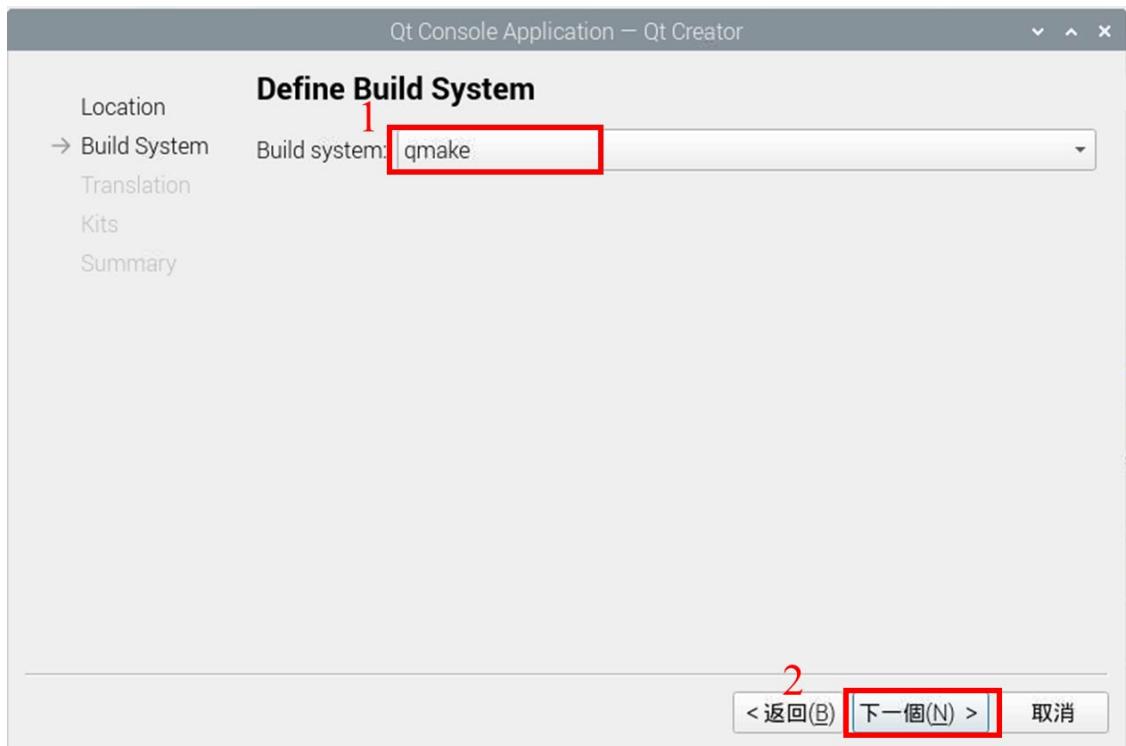
Application ➔ 紅框 3：Choose



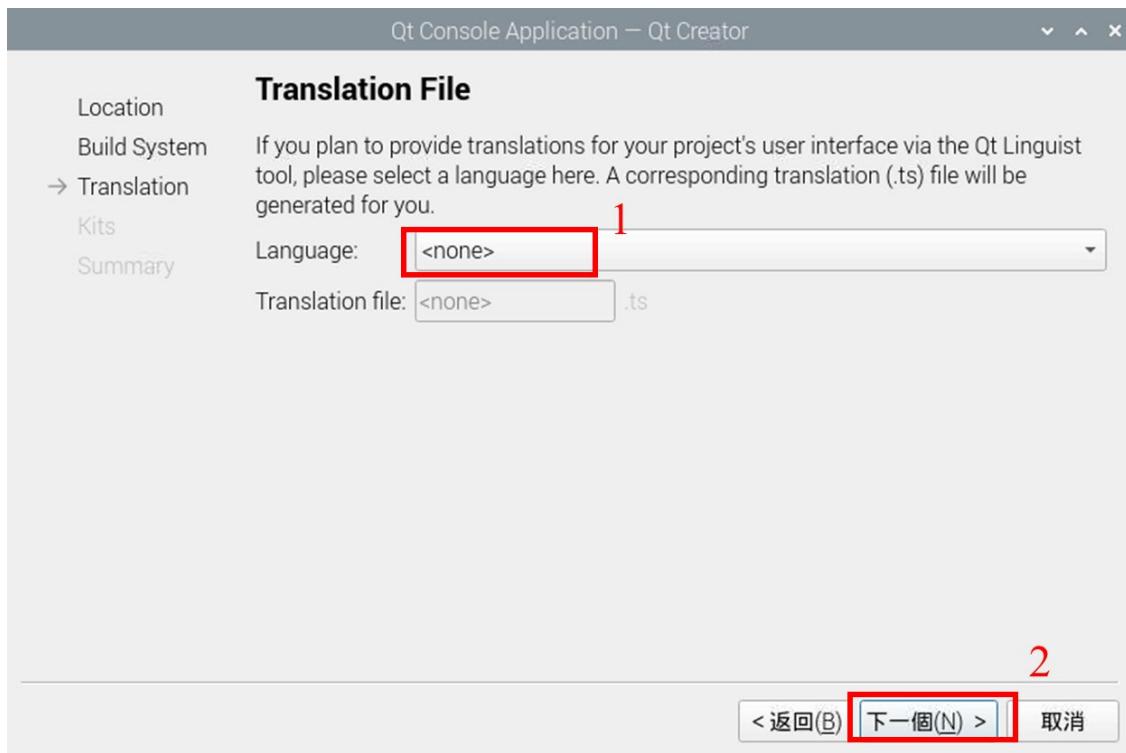
步驟 17. 創建專案，新增放置專案文件的資料夾 ➤ 紅框 1：新增專案名稱 ➔
紅框 2：作為預設的專案位置打勾 ➔ 紅框 3：按下瀏覽按鈕 ➔ 紅框 4：新
增資料夾 ➔ 紅框 5：將改名為剛創建的資料夾名 ➔ 紅框 6：選擇 ➔ 紅框
7：下一個



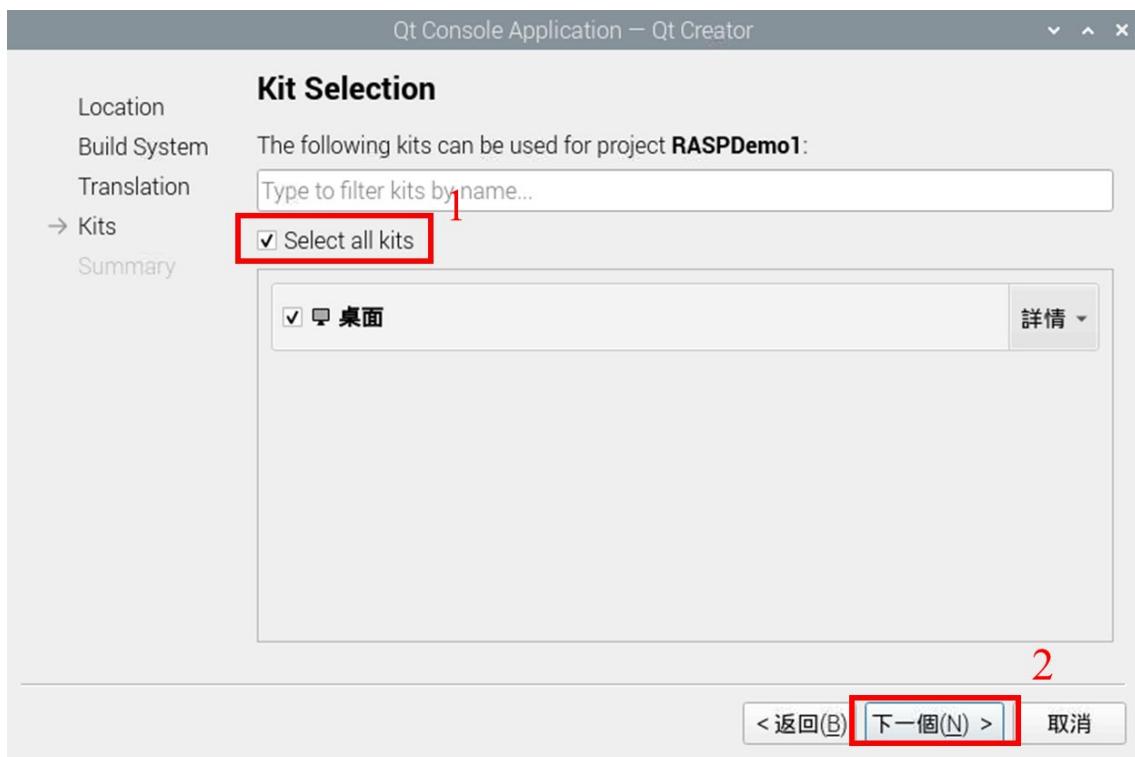
步驟 18. 創建專案 > 紅框 1：選取 qmake → 紅框 2：選取 下一個



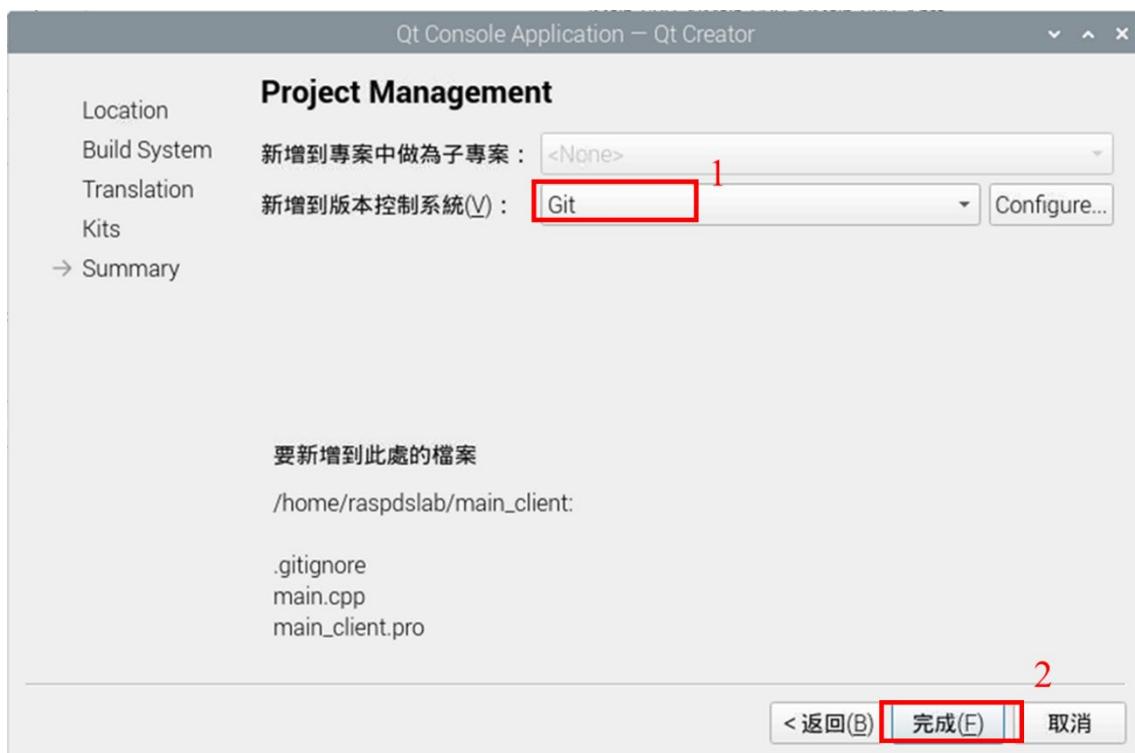
步驟 19. 創建專案 > 紅框 1：選取 none → 紅框 2：選取 下一個



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

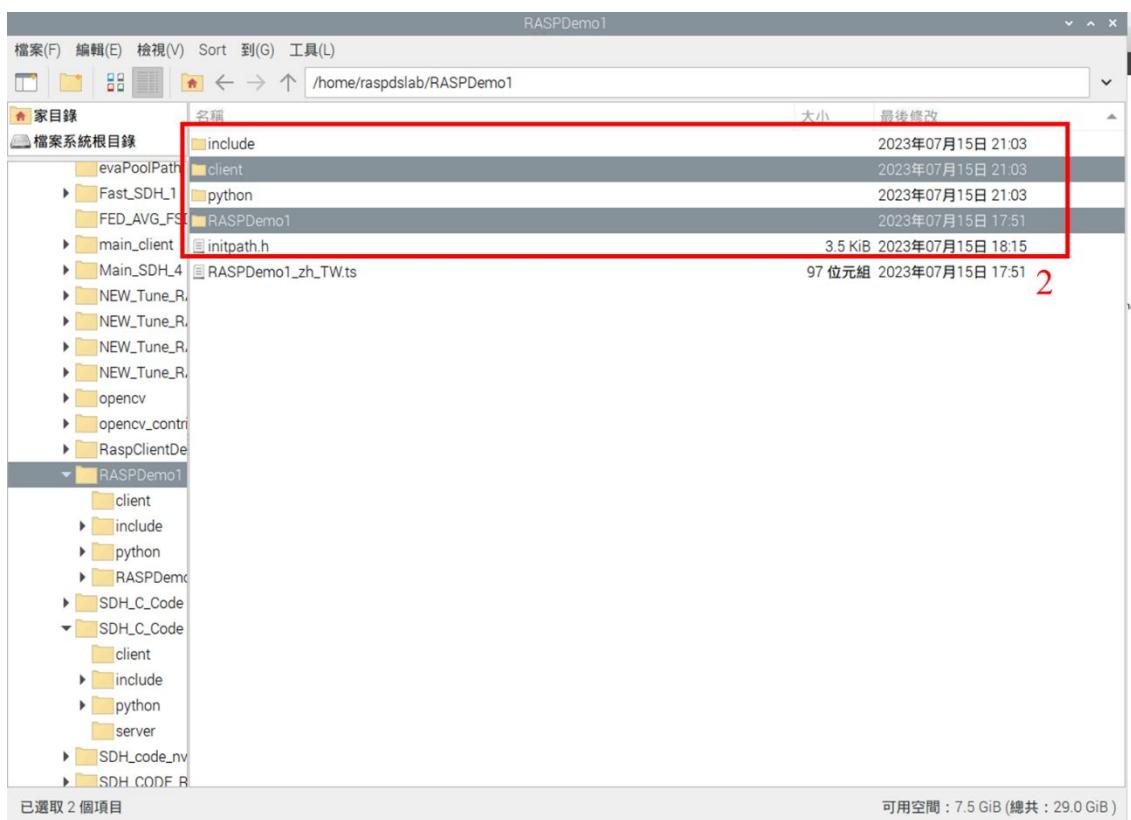
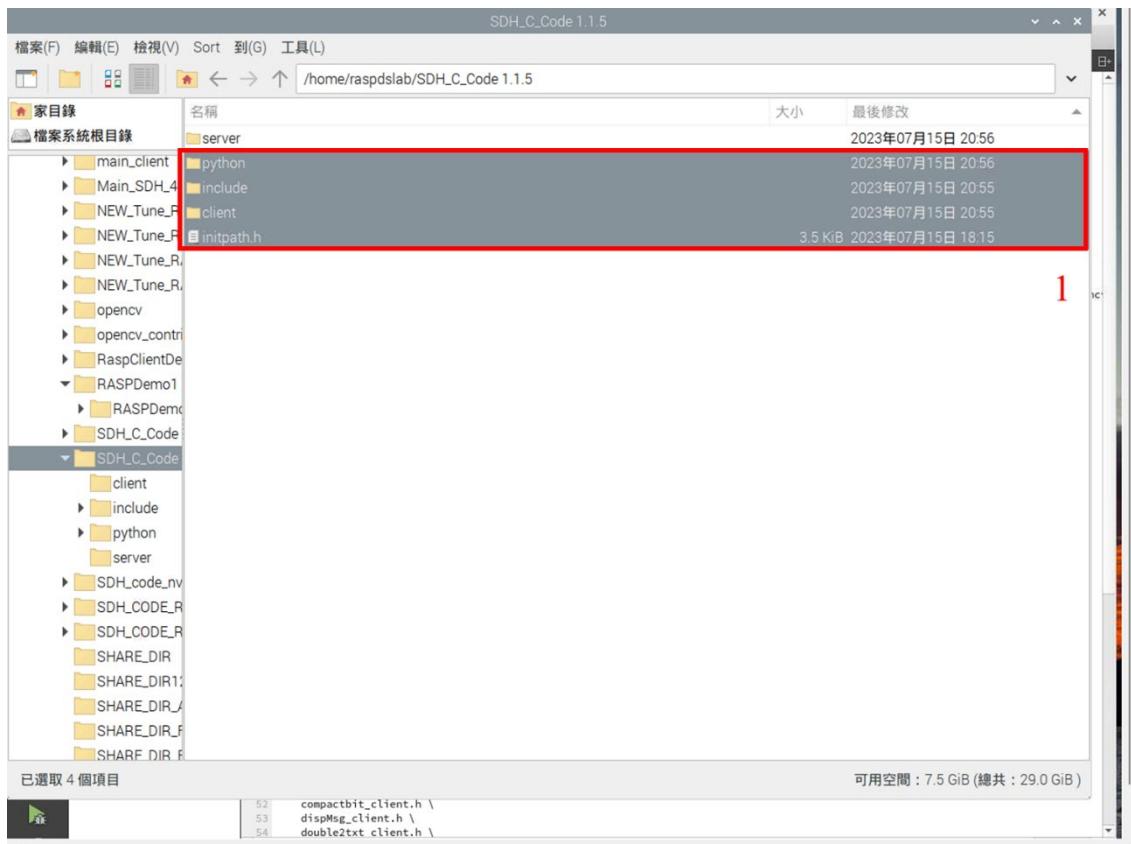


步驟 21. 創建專案 > 紅框 1：勾選 Git → 紅框 2：選取 完成



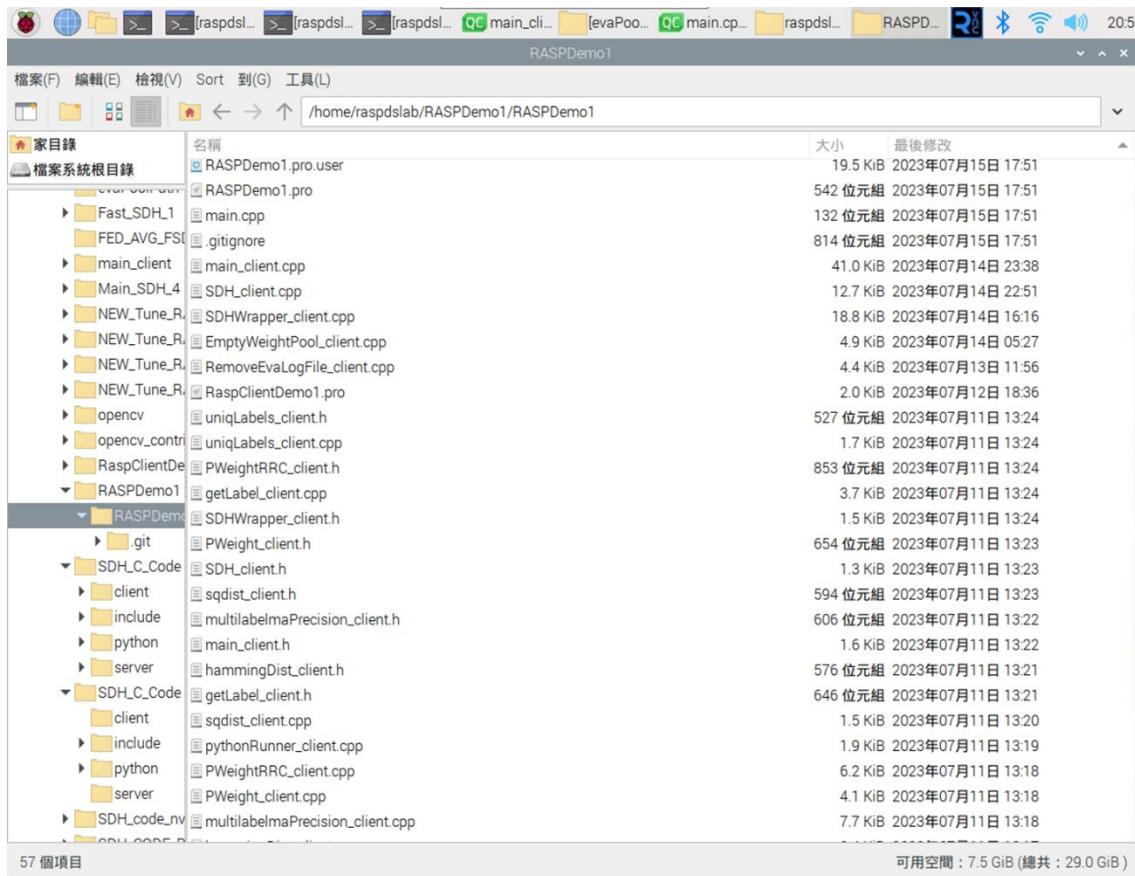
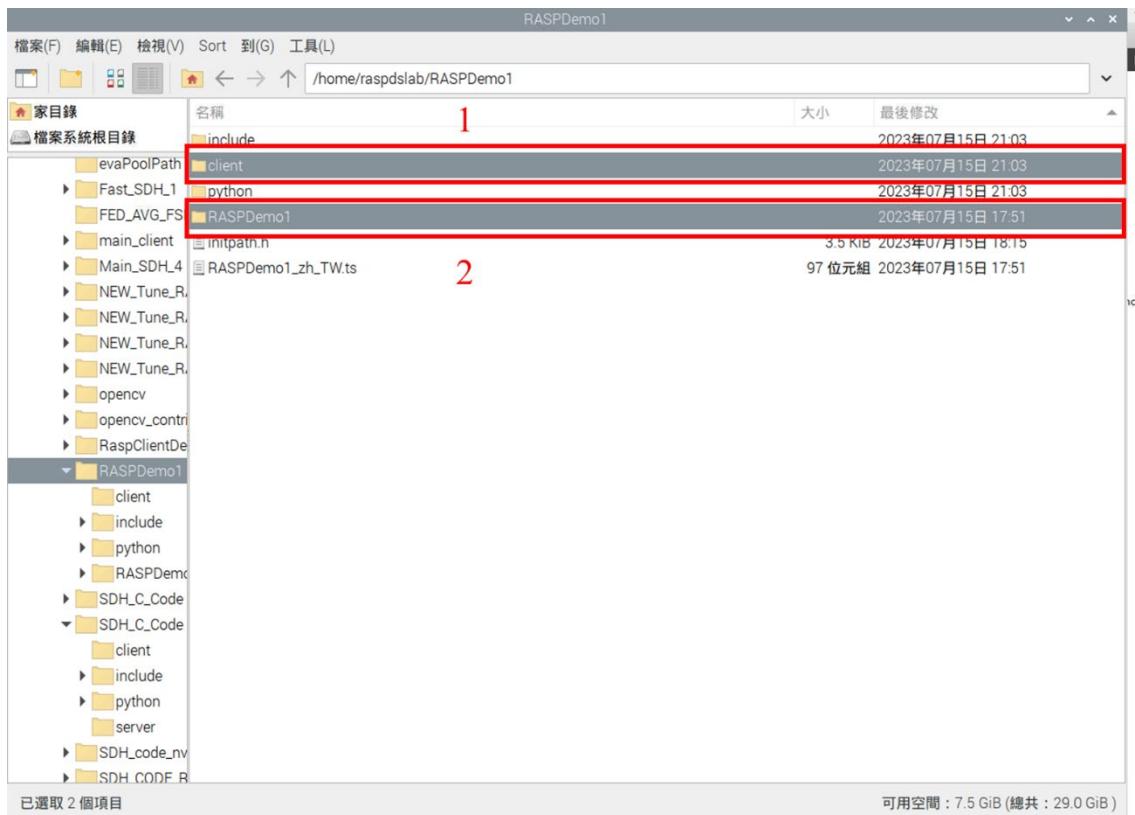
步驟 22. 放入所需檔案 ➤ 紅框 1：將紅框的四個資料夾或檔案複製到剛剛創建

專案資料夾底下 ➔ 紅框 2：複製完成畫面



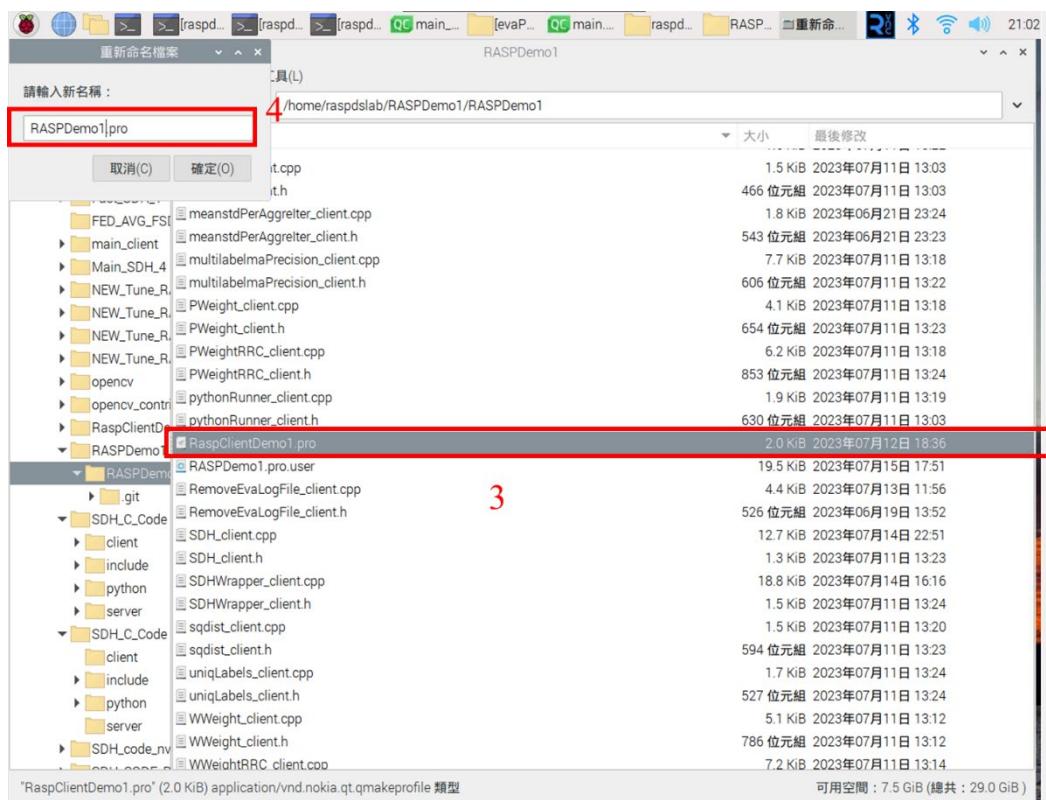
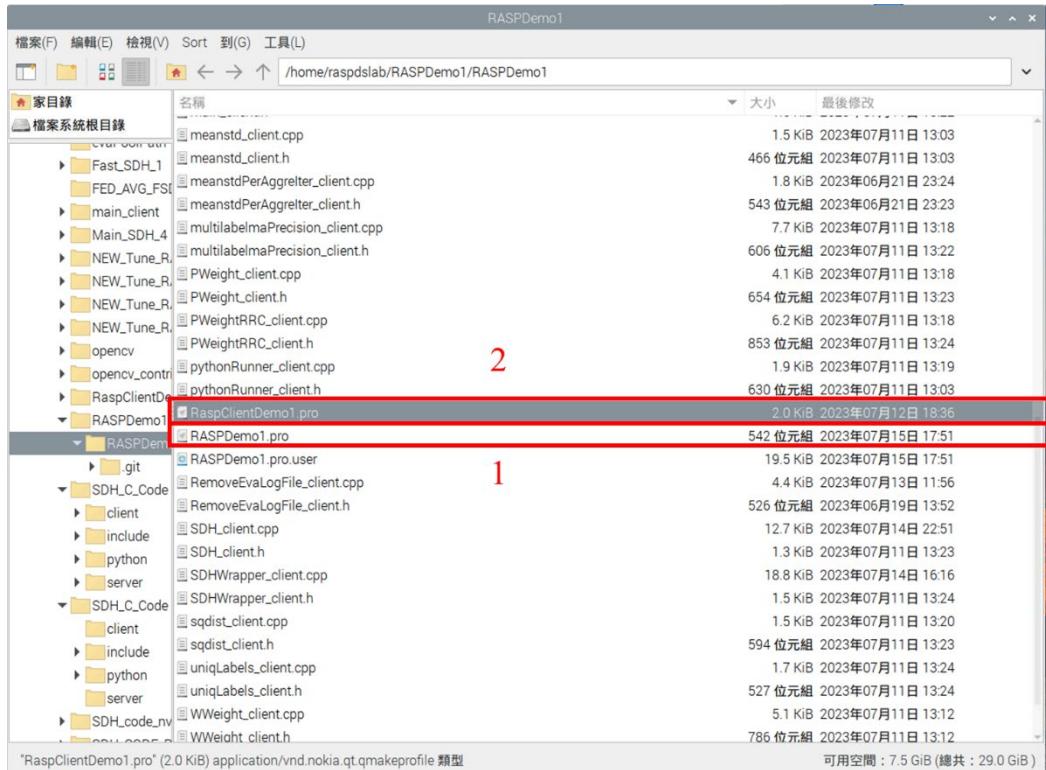
步驟 23. 複製 Client 端的檔案 > 將紅框 1 資料夾的所有檔案複製到紅框 2 放置

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

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.

The screenshot shows a desktop environment with a file browser window and a code editor window. The file browser window is titled 'RASP_tailscale (raspberrypi) - RealVNC Viewer' and shows a directory structure under '/home/raspdslab/RASPDemo1'. A file named 'initpath.h' is selected and highlighted with a red box. The code editor window shows the content of the 'initpath.h' file. The code defines various #define statements for paths, including one for the executable name at the bottom:

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

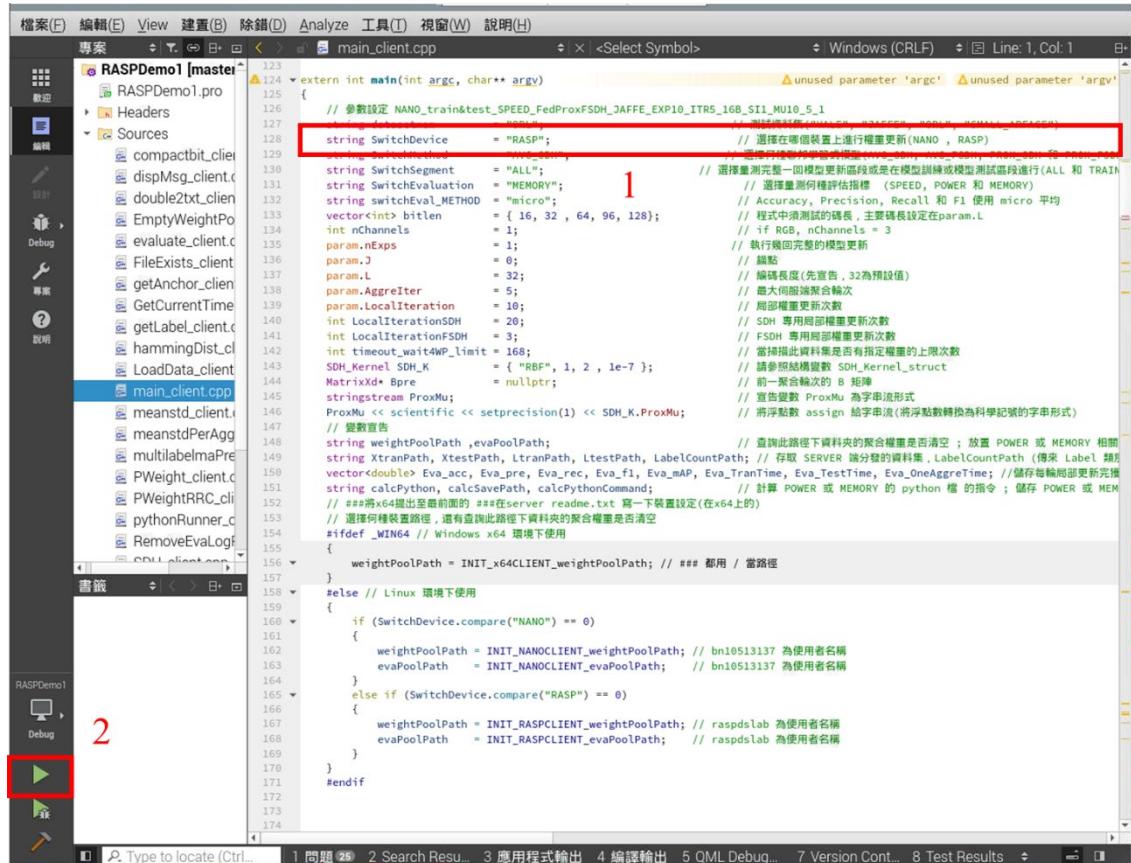
// Define the paths here

#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

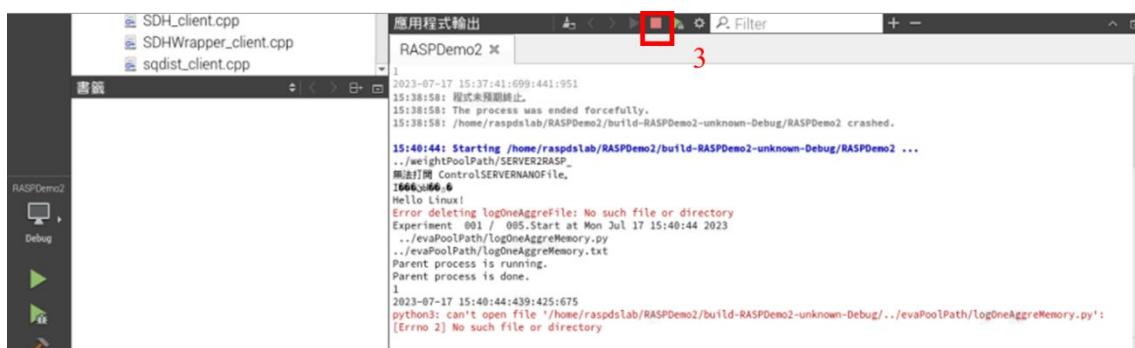
23  #define INIT_CLIENT_LabelCountMtxTxt  "_LabelCount_matrix.txt"
24  #define INIT_CLIENT_threadNamePath    "threadName.txt"
25  #define INIT_CLIENT_threadName       "./RaspDemo1" 4
```

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



```
124 extern int main(int argc, char** argv)
125 {
126     // 參數設定 NANO_train&test_SPEED_FedProxFSDH_JAFFE_EXP10_ITRS_16B_SI1_MU1_5_1
127     // SDH_Kernel_SDH_K = "NANO";
128     string SwitchDevice = "RASP"; // 選擇在那個裝置上進行權重更新(NANO , RASP)
129     string SwitchSegment = "ALL"; // 選擇量測範圍(ALL,部分)
130     string SwitchEvaluation = "MEMORY"; // 選擇量測範圍評估指標 (SPEED, POWER 和 MEMORY)
131     string switchEval_METHOD = "micro"; // 程式中須測試的長度，主要碼長設定在param.L
132     vector<int> bitlen = { 16, 32, 64, 96, 128 };
133     int nChannels = 1; // if RGB, nChannels = 3
134     param.NExps = 1; // 執行幾回完整的模型更新
135     param.J = 0; // 跳站
136     param.L = 32; // 碼長度(先宣告, 32為預設值)
137     param.Aggre_crite = 5; // 最大向量端點合併次
138     param.LocalIteration = 10; // 局部權重更新次數
139     int LocalIterationSDH = 20; // SDH 專用局部權重更新次數
140     int LocalIterationFSOH = 3; // FSOH 專用局部權重更新次數
141     int timeout_wait4WP_limit = 168; // 當掃描此資料集是否有指定權重的上限次數
142     SDH_Kernel SDH_K = { "RBF", 1, 2, 1e-7 }; // 請參照結構體 SDH_Kernel_struct
143     MatrixXd Bpre = nullptr; // 前一回合輸次的 B 矩陣
144     stringstream ProxMu; // 宣告變數 ProxMu 為字串流形式
145     ProxMu << scientific << setprecision(1) << SDH_K.ProxMu; // 將浮點數 assign 字串流(將浮點數轉換為科學記號的字串形式)
146     // 數據宣告
147     string weightPoolPath, evaPoolPath; // 查詢此路徑下資料夾的集合權重是否清空 ; 放置 POWER 或 MEMORY 相關
148     string XtrainPath, XtestPath, LtrainPath, LtestPath, LabelCountPath; // 存取 SERVER 清分發的資料集 , LabelCountPath (傳來 Label 請
149     vector<double> Eva_acc, Eva_pre, Eva_rec, Eva_f1, Eva_mA, Eva_TranTime, Eva_TestTime, Eva_OneAggreTime; // 藏存每輪局部更新完畢
150     string calPython, calcSavePath, calcPython; // 計算 POWER 或 MEMORY 的 python 指令 ; 儲存 POWER 或 MEM
151     // ##將x64提出至最前面的##在server readme.txt 還有查詢此路徑下資料夾的集合權重是否清空
152     #ifndef _WIN64 // Windows x64 環境下使用
153     {
154         weightPoolPath = INIT_X64CLIENT_weightPoolPath; // ### 都用 / 當路徑
155     }
156     else // Linux 環境下使用
157     {
158         if (SwitchDevice.compare("NANO") == 0)
159         {
160             weightPoolPath = INIT_NANOCLIENT_weightPoolPath; // bn10513137 為使用者名稱
161             evaPoolPath = INIT_NANOCLIENT_evaPoolPath; // bn10513137 為使用者名稱
162         }
163         else if (SwitchDevice.compare("RASP") == 0)
164         {
165             weightPoolPath = INIT_RASPCLOUD_weightPoolPath; // raspdslab 為使用者名稱
166             evaPoolPath = INIT_RASPCLOUD_evaPoolPath; // raspdslab 為使用者名稱
167         }
168     }
169     #endif
170 }
171
172
173
174 }
```



```
RASPDemo2 x
15:38:58: 程式未被順利終止。
15:38:58: The process was ended forcefully.
15:38:58: /home/raspdslab/RASPDemo2/build-RASPDemo2-unknown-Debug/RASPDemo2 crashed.

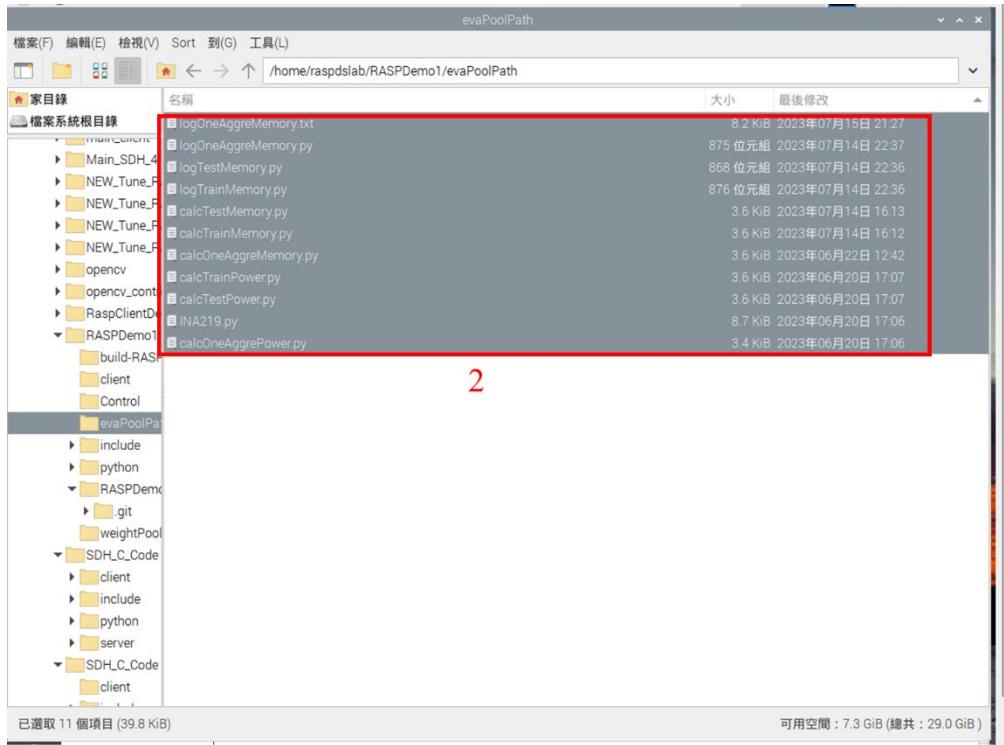
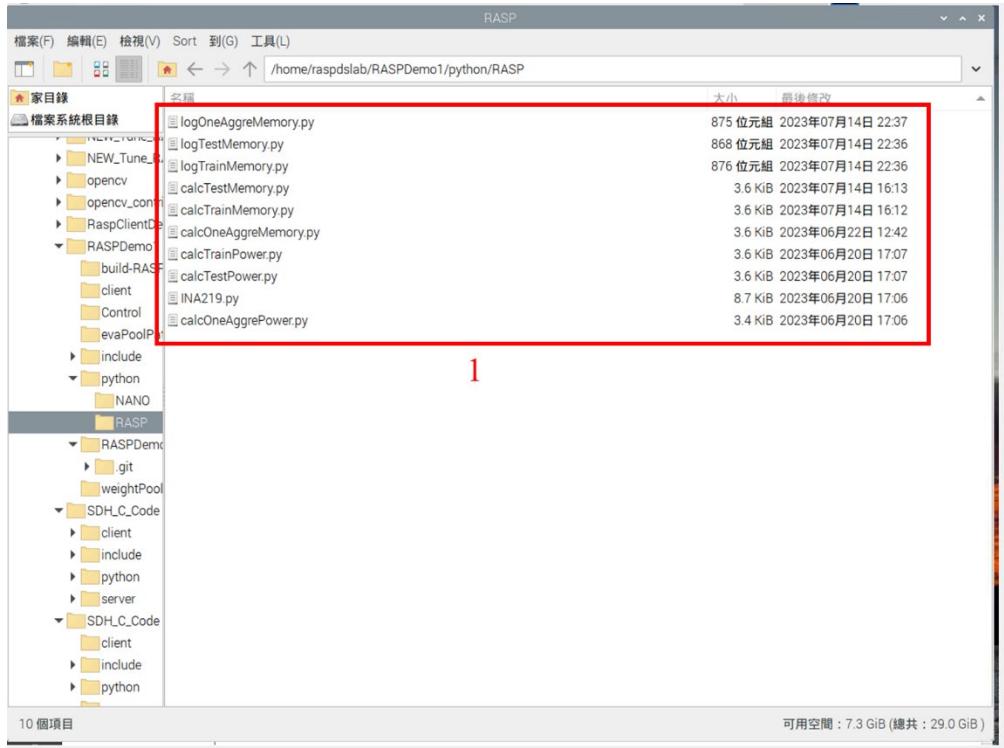
15:40:44: Starting /home/raspdslab/RASPDemo2/build-RASPDemo2-unknown-Debug/RASPDemo2 ...
./weightPoolPath/SERVER2RASP_
無法打開 ControlSERVERNANOfile,
1664:0:0:0:6
Hello world!
Error deleting logOneAggreFile: No such file or directory
Experiment 001 / 005.Start at Mon Jul 17 15:40:44 2023
./evaPoolPath/logOneAggreMemory.py
./evaPoolPath/logOneAggreMemory.txt
Parent process is running.
Parent process is done.
1
2023-07-17 15:40:44:439:425:675
python3: can't open file '/home/raspdslab/RASPDemo2/build-RASPDemo2-unknown-Debug/./evaPoolPath/logOneAggreMemory.py':
[Errno 2] No such file or directory
```

步驟 28. 複製檢測 Power 及 Memory 的 Python 檔 ➤ 紅框 1：將 python/RASP 資

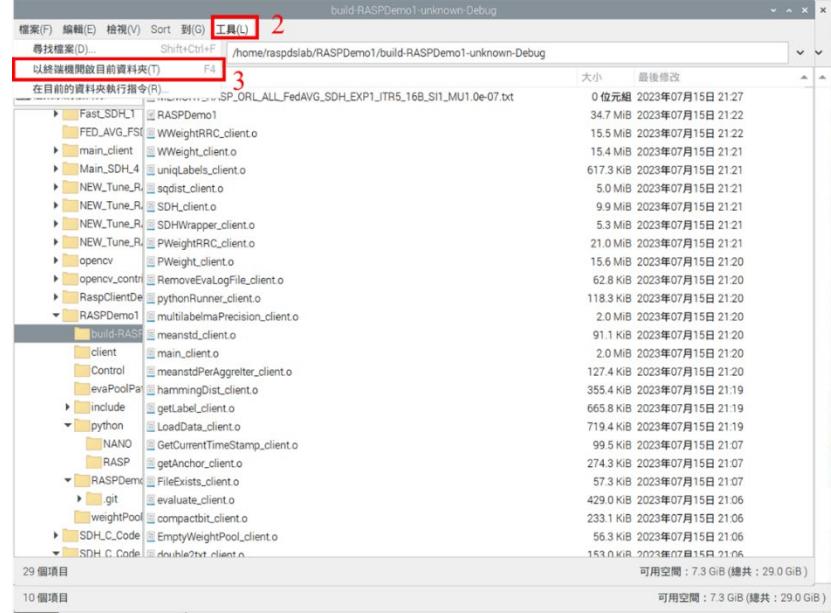
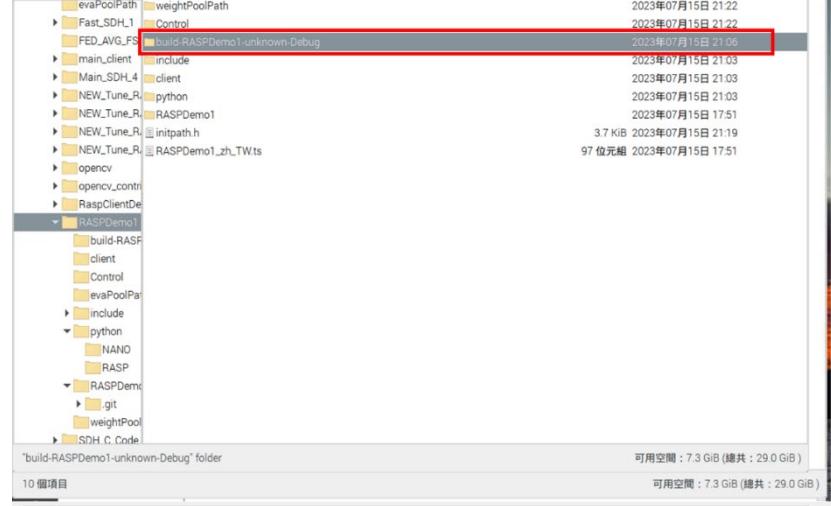
料夾裡的檔案複製到專案資料夾裡的 evaPoolPath (`/home/raspdslab/專案名`

`/evaPoolPath`)，evaPoolPath 資料夾剛編輯程式時會自動創建不用自己手動

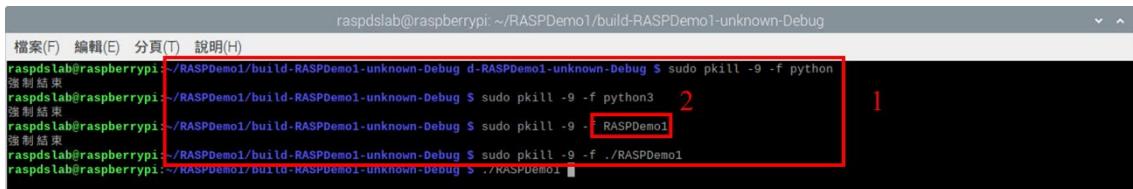
創建 ➔ 紅框 2：複製結果如下



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



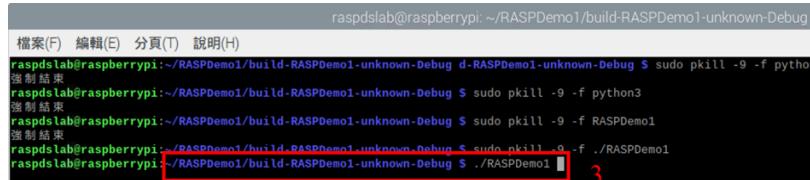
The screenshot shows a terminal window with the following command history:

```
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug$ sudo pkill -9 -f python
強制結束
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug$ sudo pkill -9 -f python3
強制結束
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug$ sudo pkill -9 -f RASPDemo1
強制結束
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug$ sudo pkill -9 -f ./RASPDemo1
強制結束
```

A red box labeled '1' highlights the first command: `sudo pkill -9 -f python`. A red box labeled '2' highlights the fourth command: `sudo pkill -9 -f ./RASPDemo1`.

➤ 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 ./)
```



The screenshot shows a terminal window with the following command history:

```
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug$ sudo pkill -9 -f python
強制結束
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug$ sudo pkill -9 -f python3
強制結束
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug$ sudo pkill -9 -f RASPDemo1
強制結束
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug$ sudo pkill -9 -f ./RASPDemo1
強制結束
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug$ ./RASPDemo1
```

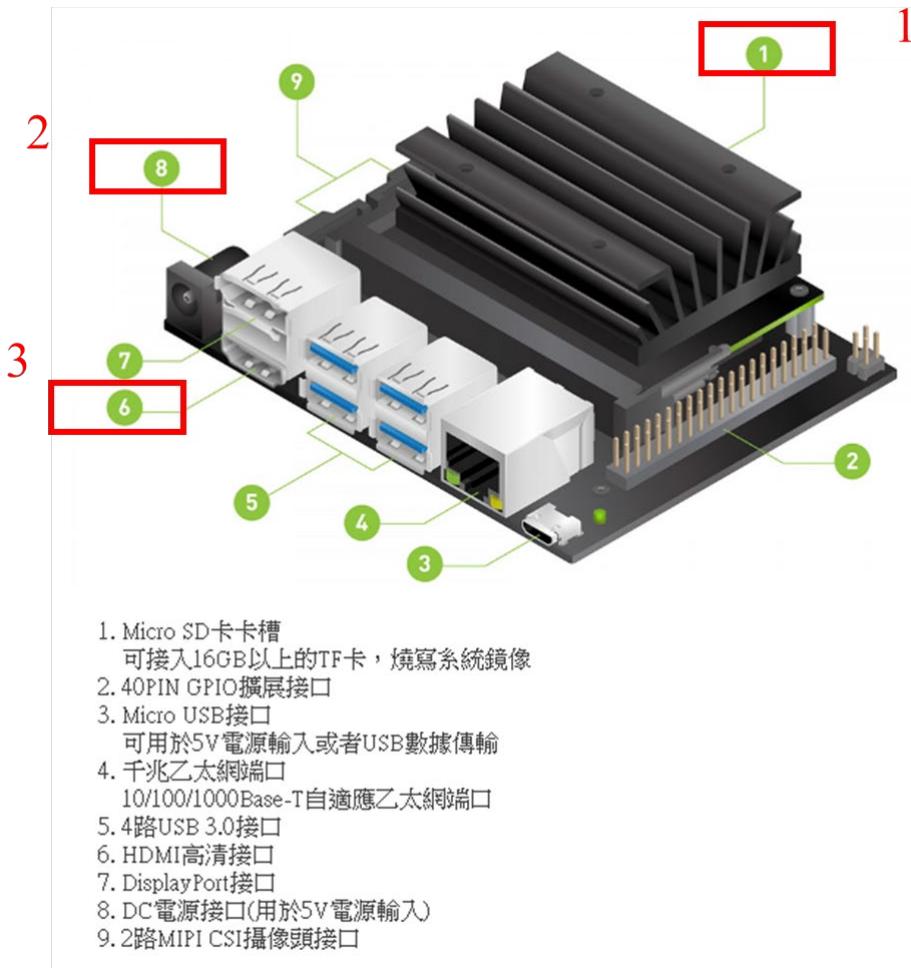
A red box labeled '3' highlights the final command: `./RASPDemo1`.

➤ 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 ./)
```

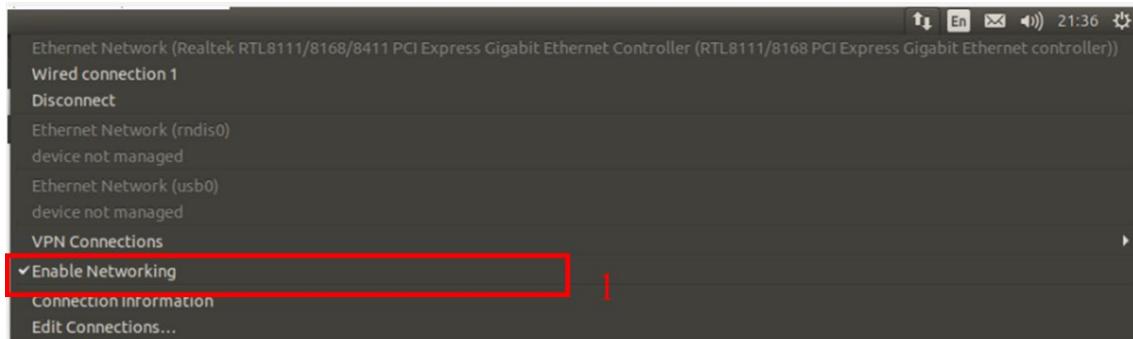
Jetson Nano

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，



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 Networking



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

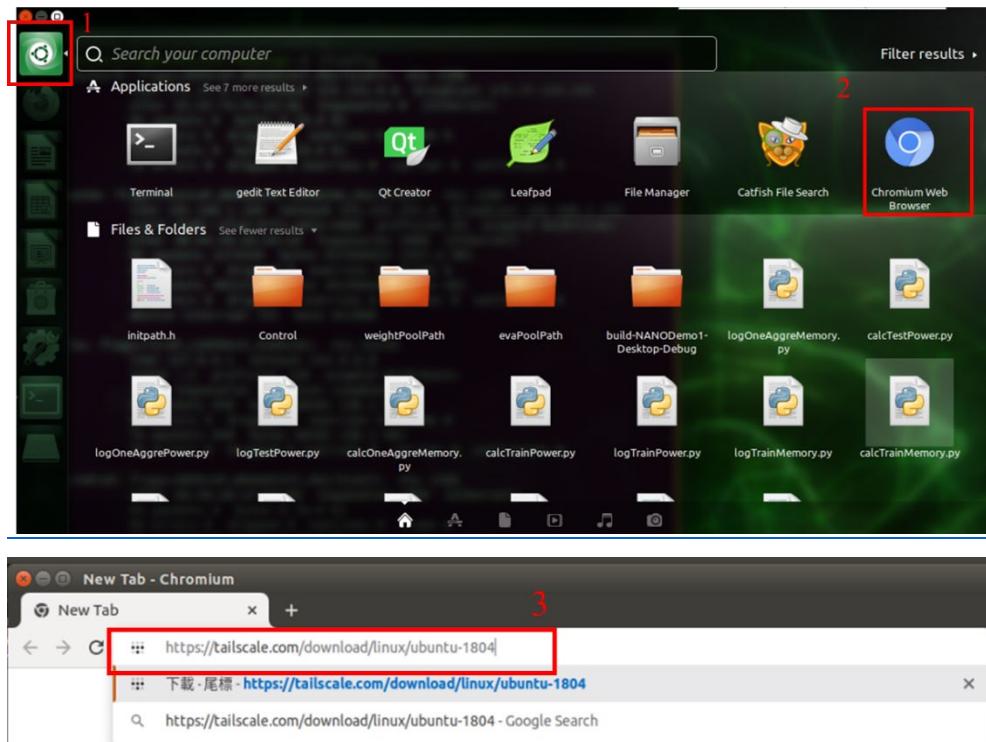
# 更新套件
sudo apt-get upgrade

# 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

Install with one command

```
curl -fsSL https://tailscale.com/install.sh | sh
```

View script source

Manually install on Ubuntu 18.04 LTS (Bionic) ▾

Packages are available for x86 and ARM CPUs, in both 32-bit and 64-bit variants.

1 Add Tailscale's package signing key and repository:

```
curl -fsSL https://pkgs.tailscale.com/stable/ubuntu/bionic.gpg | sudo apt-key add -
curl -fsSL https://pkgs.tailscale.com/stable/ubuntu/bionic.list | sudo tee /etc/apt/sources.list.d/tailscale.list
```

2 Install Tailscale:

```
sudo apt-get update
sudo apt-get install tailscale
```

3 Connect your machine to your Tailscale network and authenticate in your browser:

```
sudo tailscale up
```

4 You're connected! You can find your Tailscale IPv4 address by running:

```
tailscale ip -4
```

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)

```
bn@bn10513137:~$ ifconfig
bn10513137:~$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 brd 172.17.0.1 scope 0x1<host>
        ether 02:42:7e:91:e2:da txqueuelen 0 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.184 brd 192.168.1.255 scope 0x1<host>
        ether fe80::6fa9:87fa:82a:4ddd txqueuelen 64 (Ethernet)
        RX packets 2274549 bytes 957696854 (957.6 MB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 301257 bytes 33460442 (3.6 GB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
        device interrupt 151 base 0x1000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 brd 127.0.0.1 scope 0x1<host>
        loop txqueuelen 1 (Local Loopback)
        RX packets 840 bytes 88100 (88.1 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 840 bytes 88100 (88.1 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

rndis0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 96:3b:ab:a7:b3:d0 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

tailscaled: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1280
    inet 100.66.82.83 brd 100.66.82.83 scope 0x1<host>
        inet6 fd7a:115c:ale0:ab12:843:cd96:6242:s253 txqueuelen 128 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

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

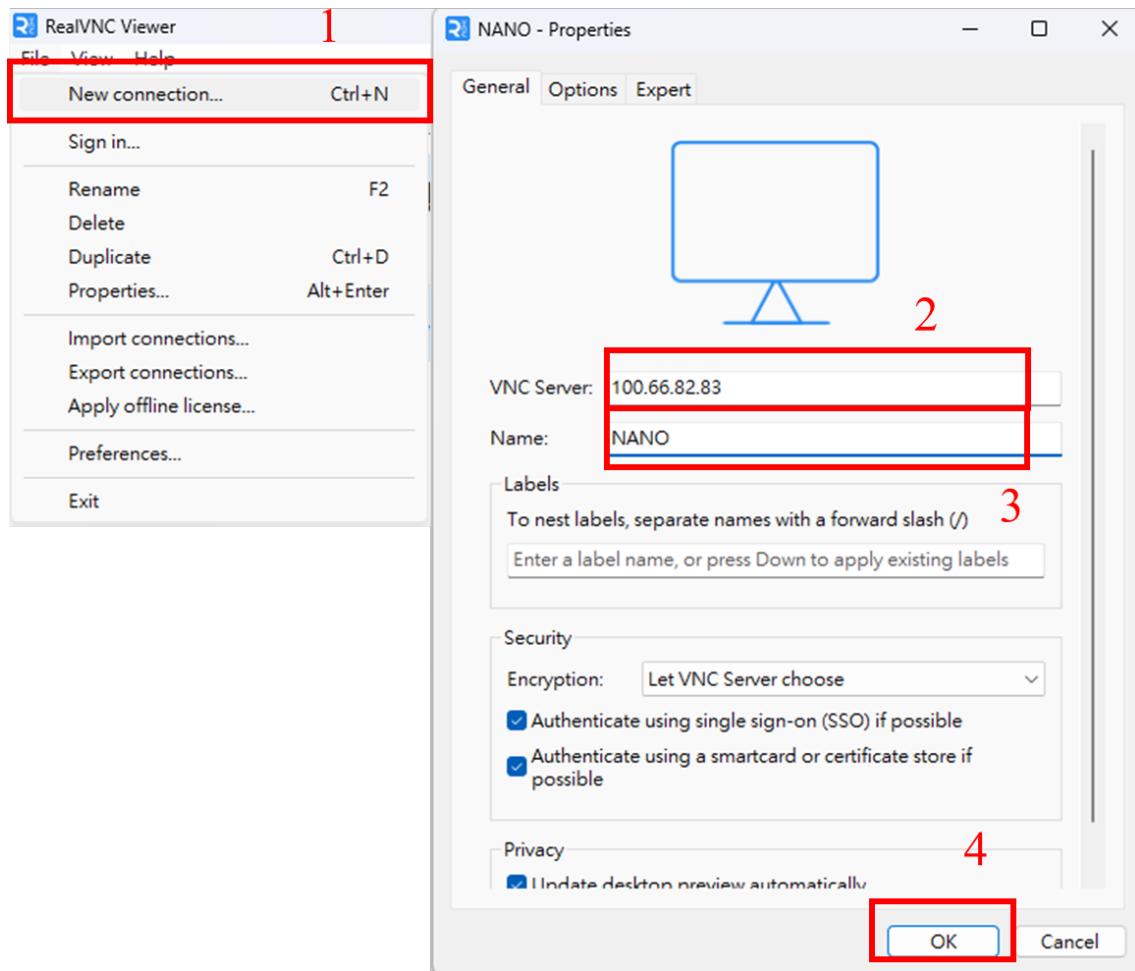
# 將你正在用的網路卡加入 VINO 服務
nmcli connection show

# 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
```

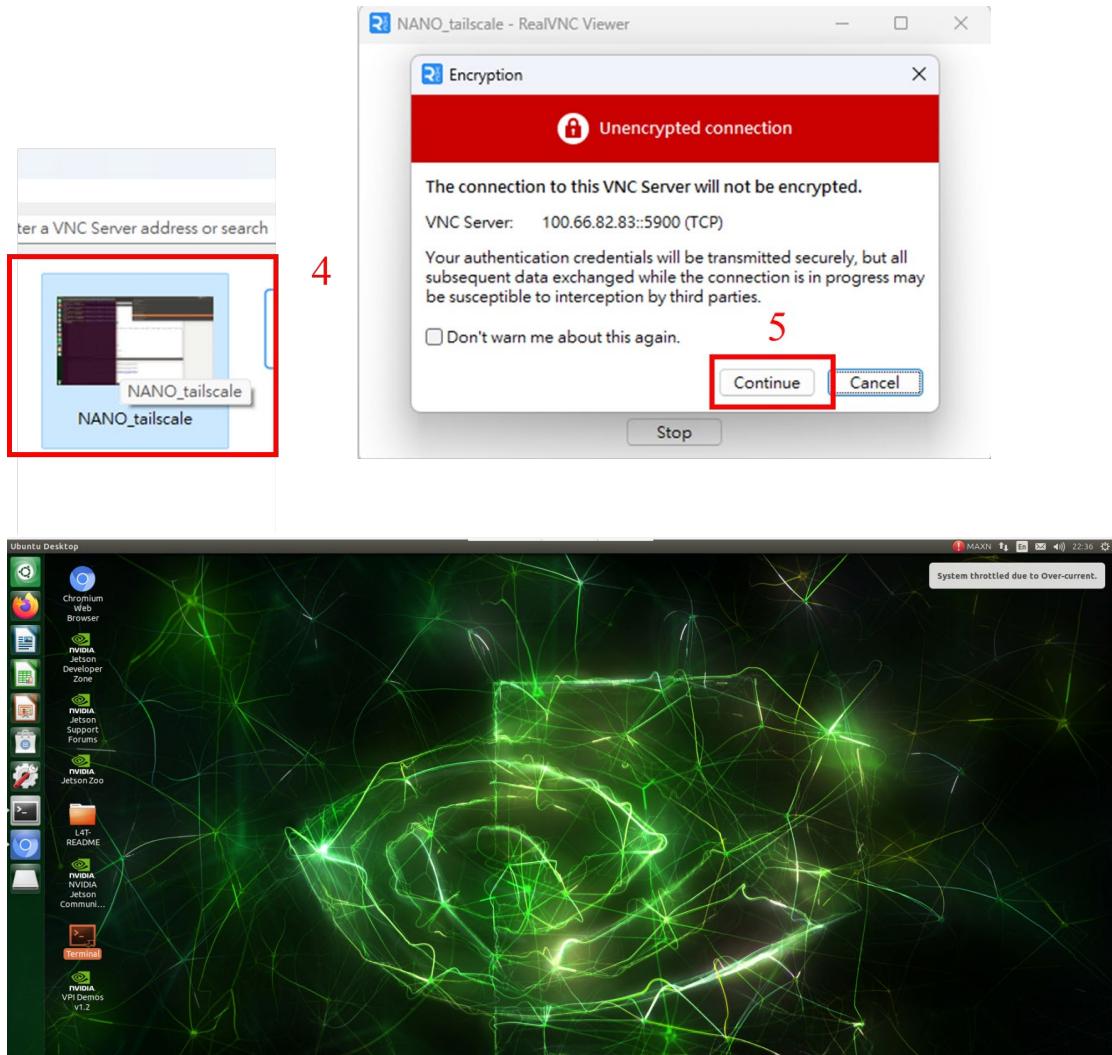
步驟 8. 打開電腦 VNC 軟體 ➤ 紅框 1：New connection ➔ 紅框 2：輸入

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
```

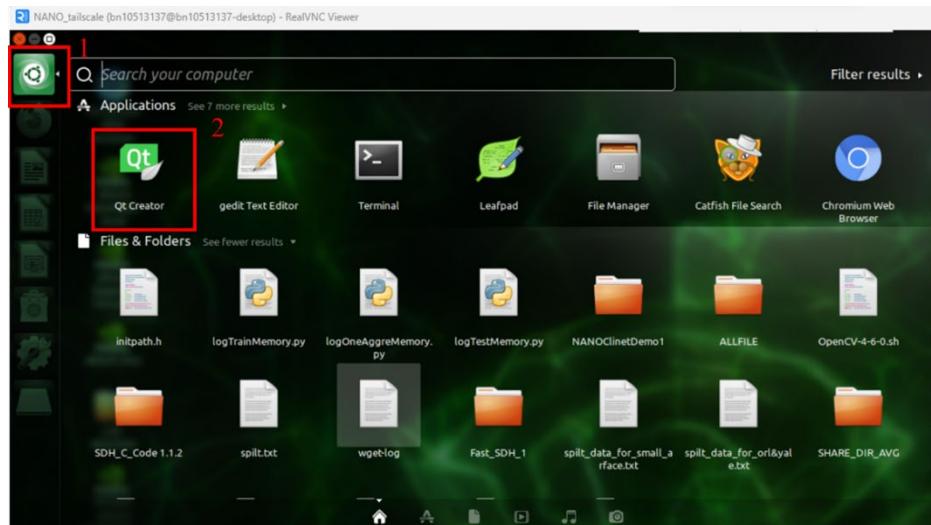
#安裝 Qt 套件

```
sudo apt-get install -y qt5-default
```

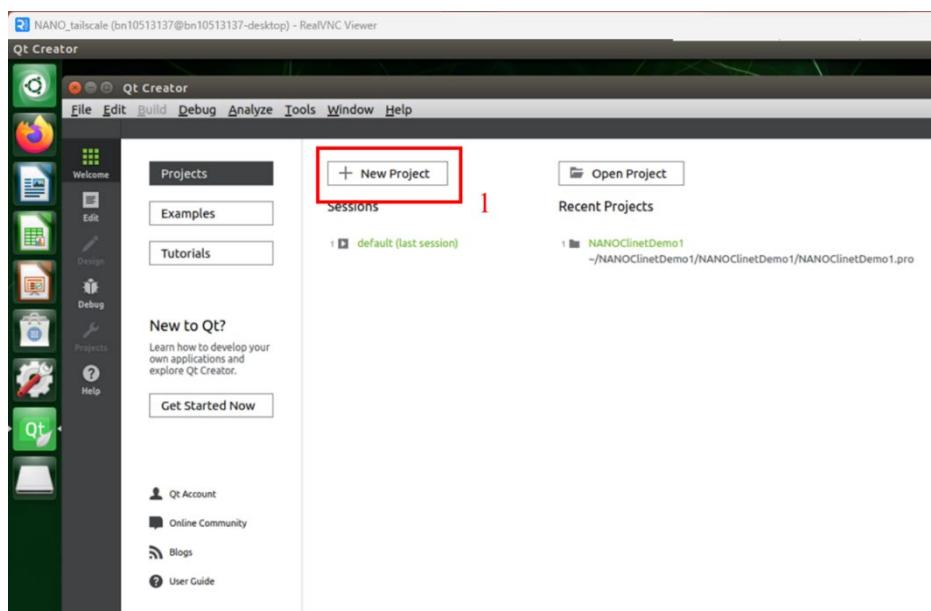
```
sudo apt-get install -y qtcreator
```

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

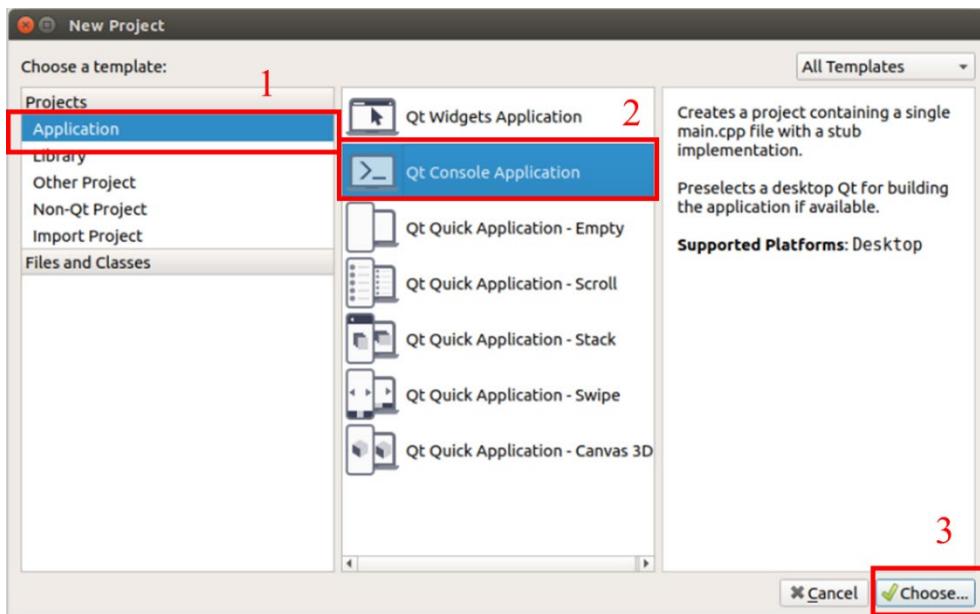


步驟 12. 創建新專案 > 紅框 1：+ New Project



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

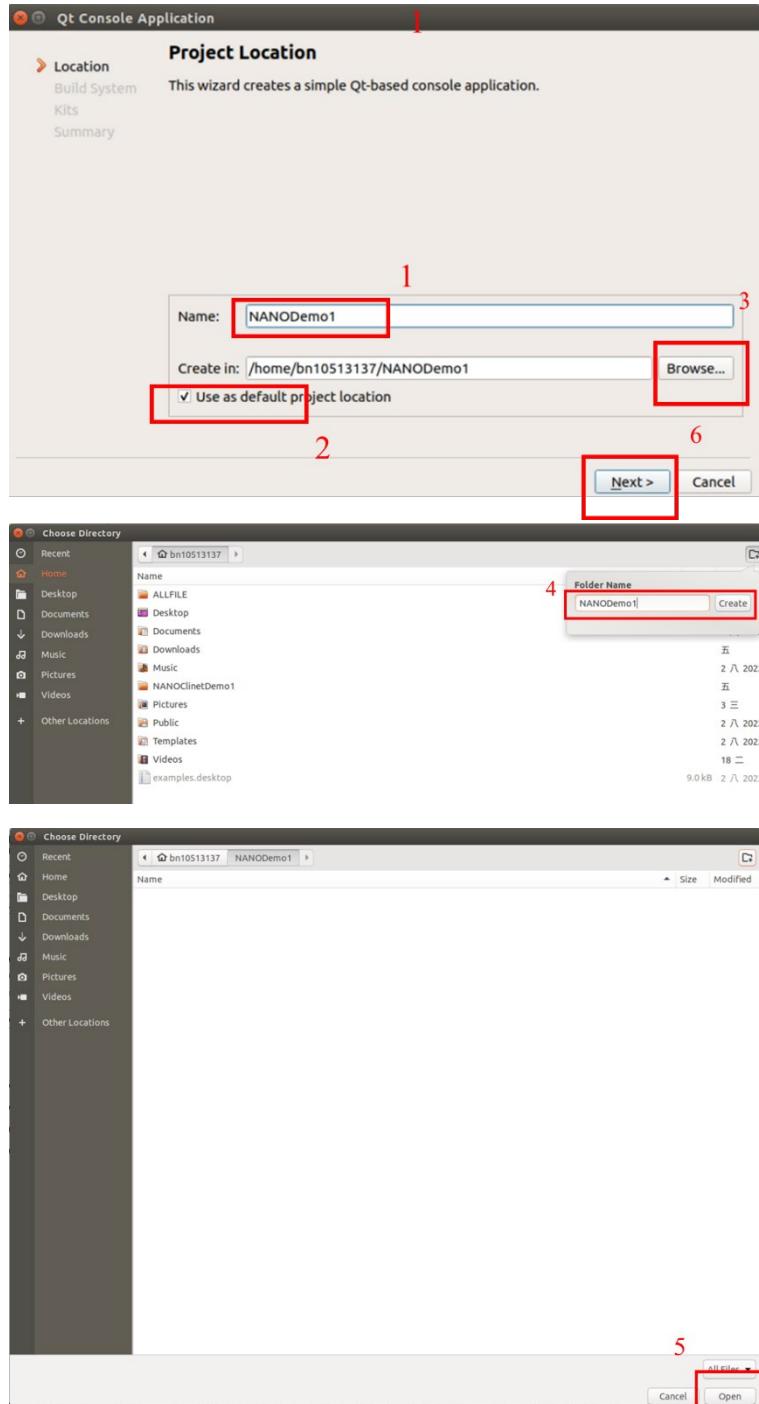


步驟 14. 創建專案，新增放置專案文件的資料夾 ➤ 紅框 1：新增專案名稱 ➔

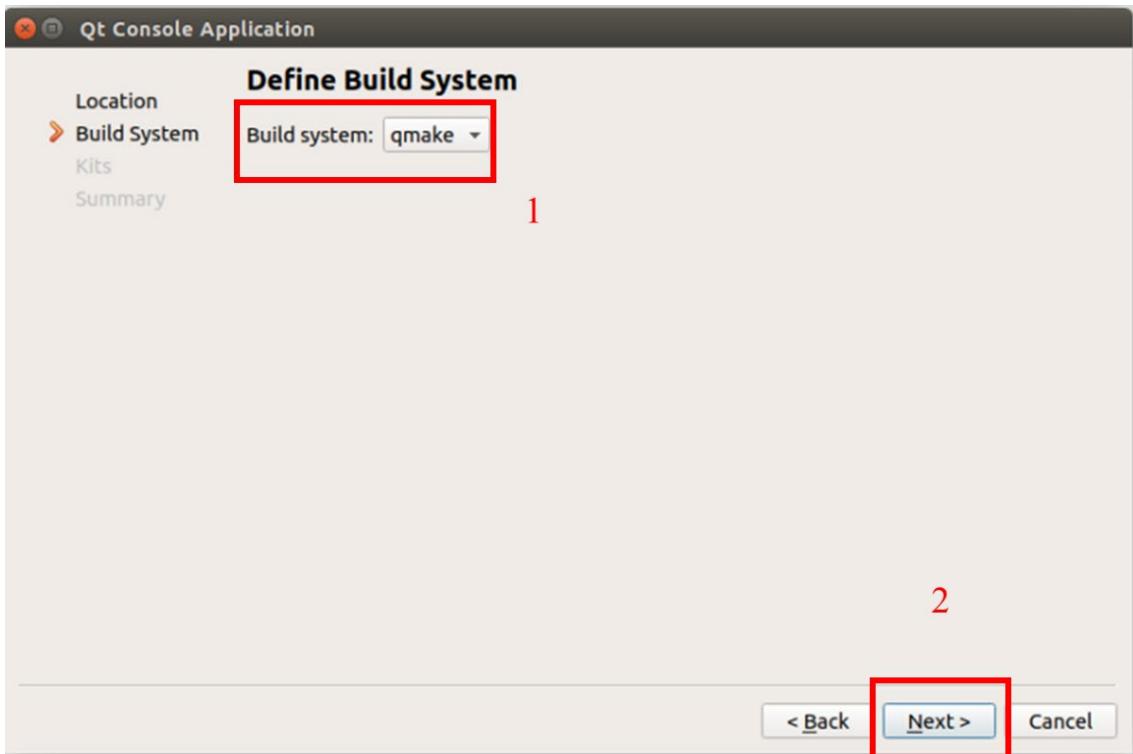
紅框 2：作為預設的專案位置打勾 ➔ 紅框 3：按下 Browse 按鈕 ➔ 紅框

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

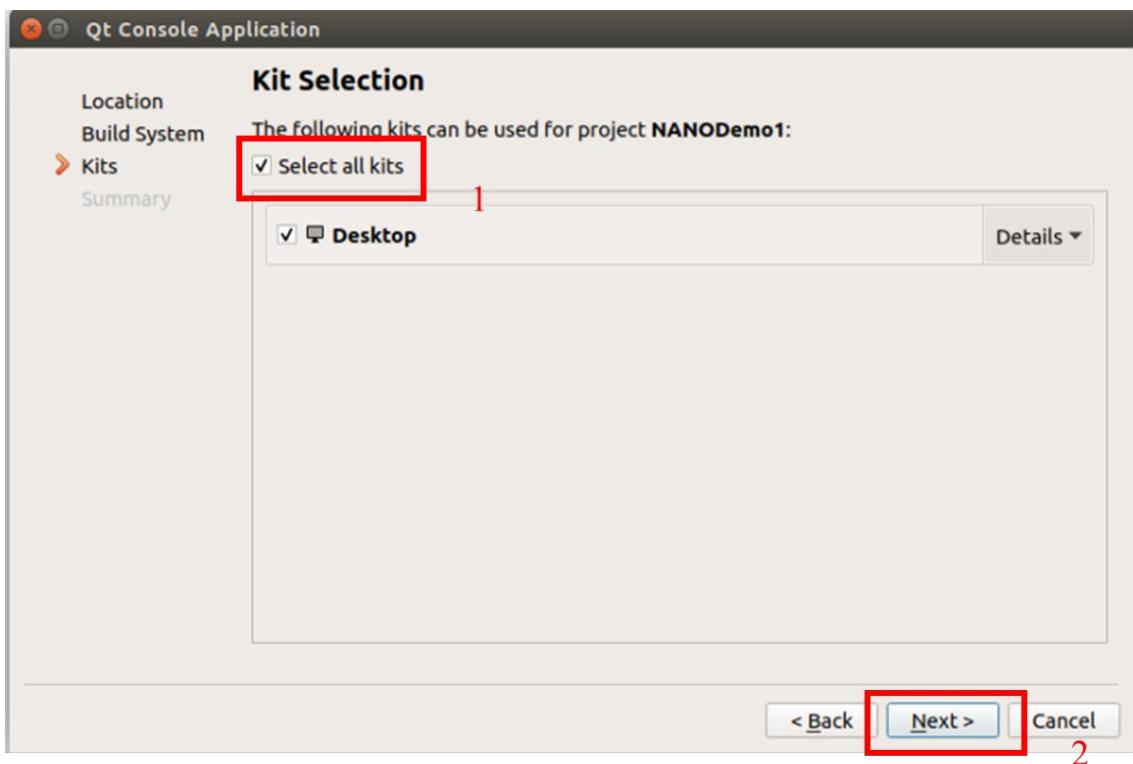
Open 按鈕 ➔ 紅框 6：Next



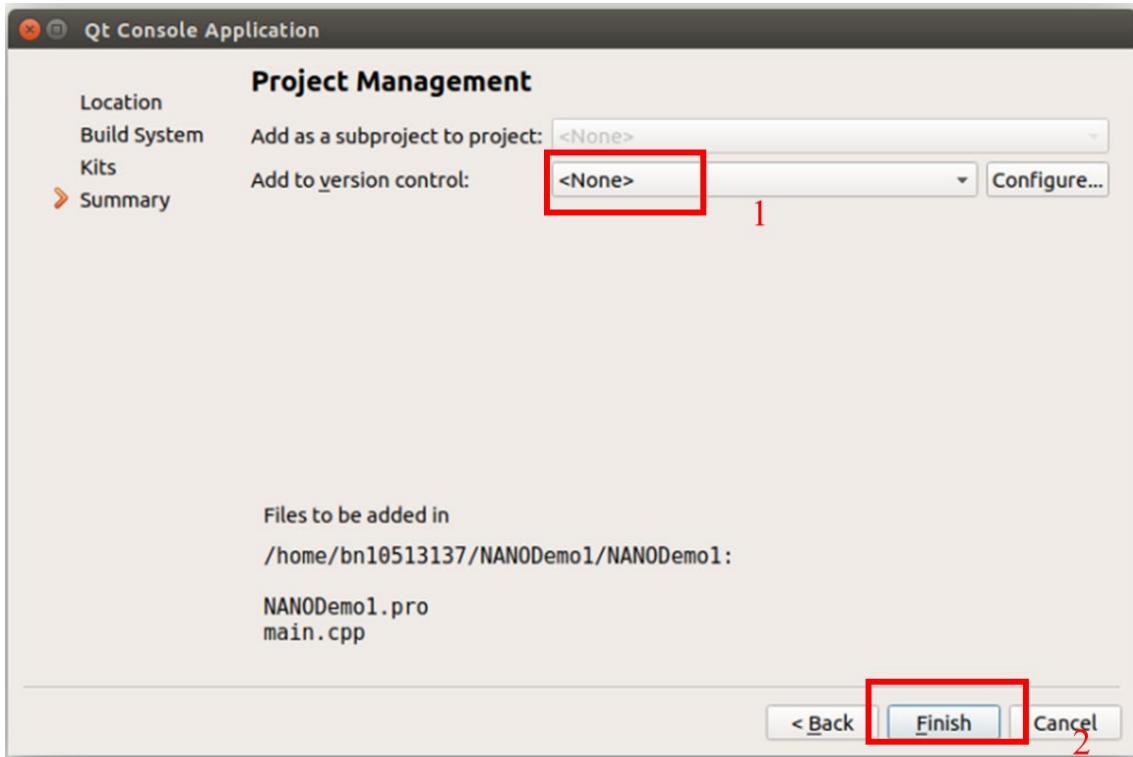
步驟 15. 創建專案 > 紅框 1：選取 qmake → 紅框 2：點擊 Next



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

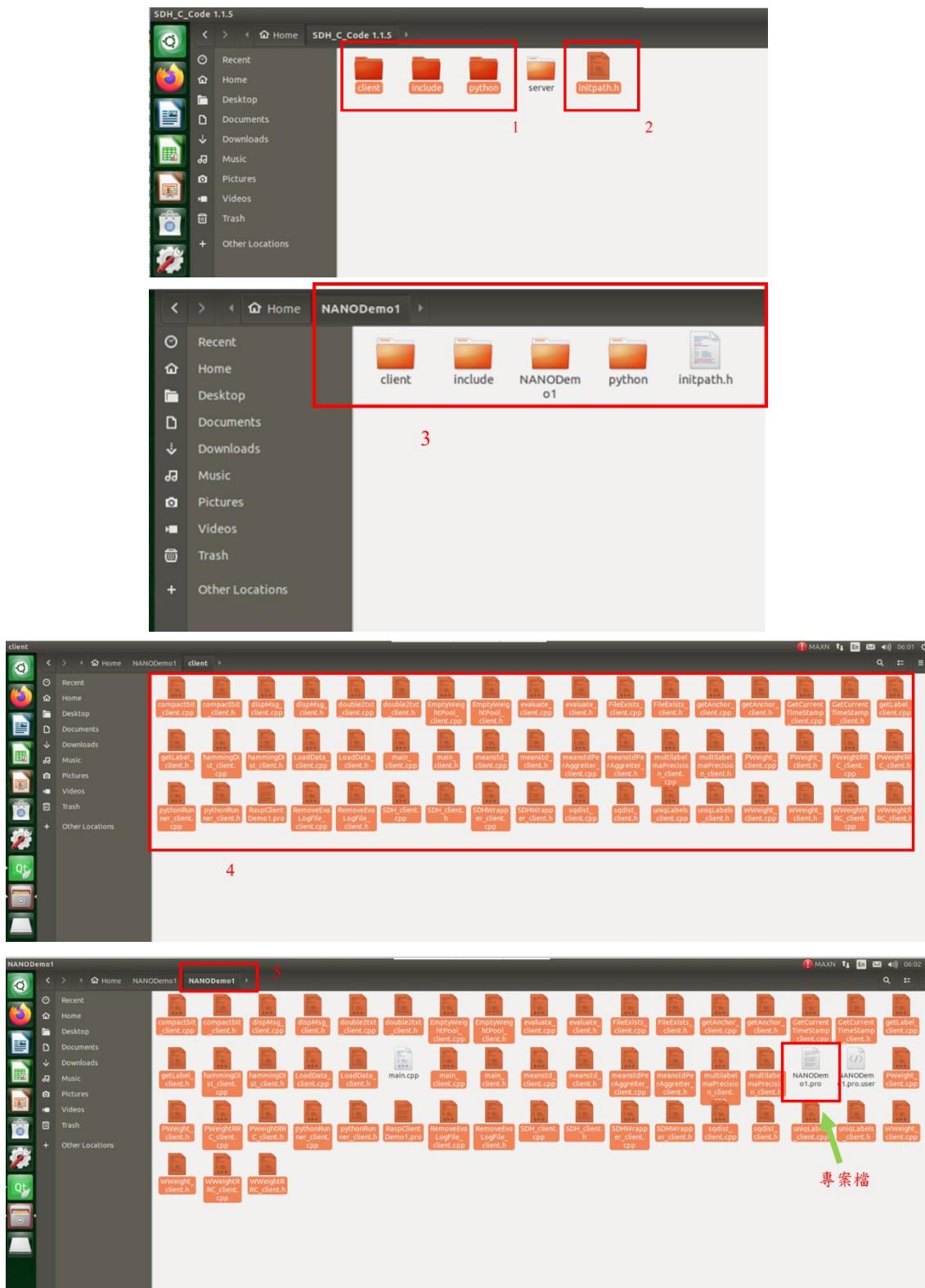


步驟 17. 創建專案 > 紅框 1：勾選 <None> → 紅框 2：點擊 Finish



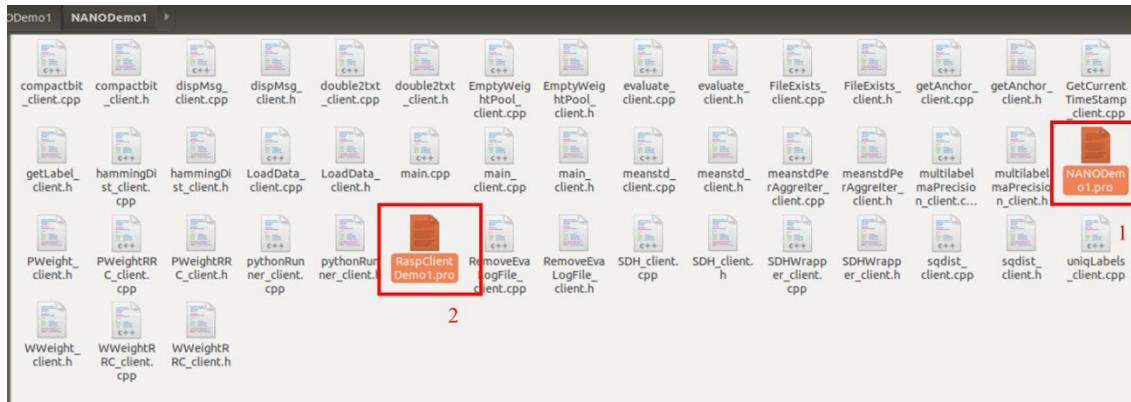
步驟 18. 放入所需檔案 > 將紅框 1 和紅框 2 的四個資料夾或檔案複製到專案資料夾底下，如紅框 3 所示 → 紅框 4：將 client 資料夾底下的所有檔案複製

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)

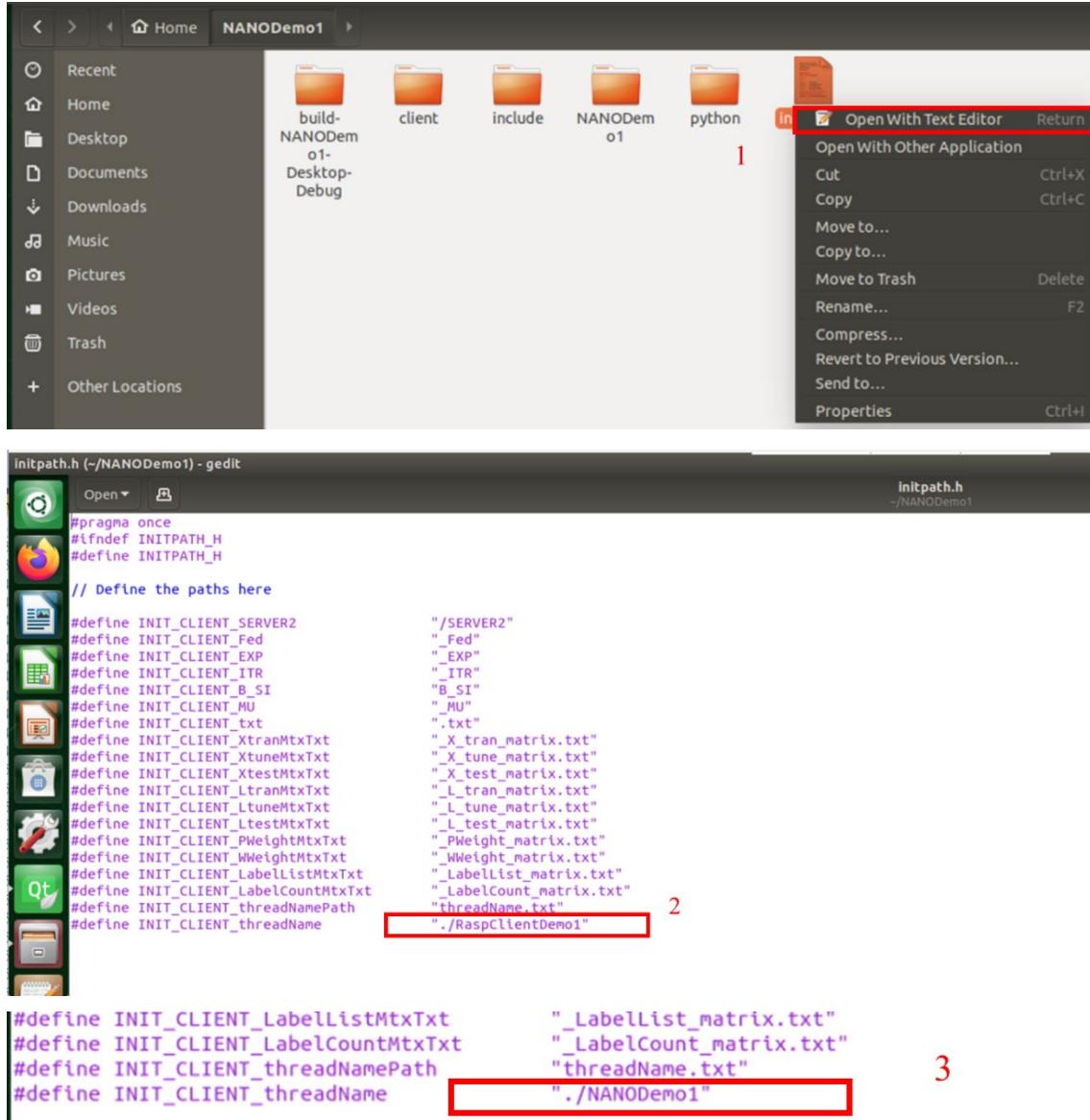


步驟 20. 因測量 Memory 時會去抓執行序名稱，才能獲得該執行序的 Memory

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

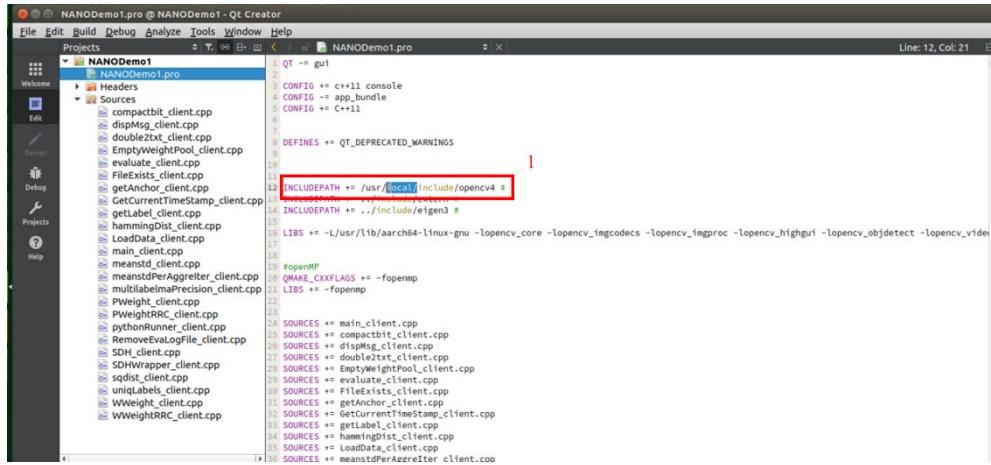
`initpath.h` 點擊右鍵，跳出選單，點選 `Open With Text Editor` ➔ 紅框 2：將

執行序名稱改為./專案檔名 ➔ 紅框 3：改完結果

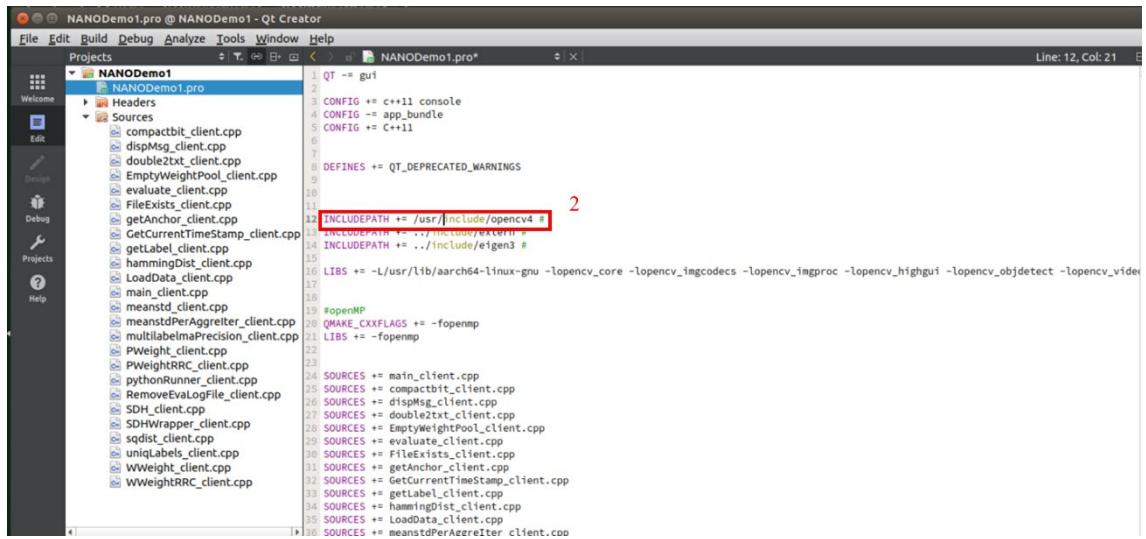


步驟 21. 打開創建的專案檔，點擊專案檔名.pro > 將紅框 1 修改為 紅框 2

2(/usr/local/includeopencv4)



```
QT -= gui
CONFIG += c++11 console
CONFIG -= app_bundle
CONFIG += C+11
DEFINES += QT_DEPRECATED_WARNINGS
INCLUDEPATH += /usr/local/include/opencv4 # 1
LIBS += -L/usr/lib/aarch64-linux-gnu -lopencv_core -lopencv_imgcodecs -lopencv_imgproc -lopencv_highgui -lopencv_objdetect -lopencv_video
INCLUDEPATH += ../include/eigen3 #
openMP
QMAKE_CXXFLAGS += -fopenmp
LIBS += -fopenmp
main_client.cpp
meanstdPerAggregator_client.cpp
multilabelmaPrecision_client.cpp
PWeight_RRC_client.cpp
pythonRunner_client.cpp
RemoveEvLogFile_client.cpp
SDH_client.cpp
SDHWrapper_client.cpp
double2txt_client.cpp
EmptyWeightPool_client.cpp
evaluate_client.cpp
FileExists_client.cpp
GetCurrentTimeStamp_client.cpp
getAnchor_client.cpp
getLabel_client.cpp
hammingDist_client.cpp
LoadData_client.cpp
main_client.cpp
meanstd_client.cpp
meanstdPerAggregator_client.cpp
multilabelmaPrecision_client.cpp
PWeight_RRC_client.cpp
pythonRunner_client.cpp
RemoveEvLogFile_client.cpp
SDH_client.cpp
SDHWrapper_client.cpp
double2txt_client.cpp
EmptyWeightPool_client.cpp
evaluate_client.cpp
FileExists_client.cpp
GetCurrentTimeStamp_client.cpp
getAnchor_client.cpp
getLabel_client.cpp
hammingDist_client.cpp
LoadData_client.cpp
meanstdPerAggregator.cpp
```



```
QT -= gui
CONFIG += c++11 console
CONFIG -= app_bundle
CONFIG += C+11
DEFINES += QT_DEPRECATED_WARNINGS
INCLUDEPATH += /usr/include/opencv4 # 2
INCLUDEPATH += ../include/eigen3 #
INCLUDEPATH += ./include/
LIBS += -L/usr/lib/aarch64-linux-gnu -lopencv_core -lopencv_imgcodecs -lopencv_imgproc -lopencv_highgui -lopencv_objdetect -lopencv_video
openMP
QMAKE_CXXFLAGS += -fopenmp
LIBS += -fopenmp
main_client.cpp
meanstdPerAggregator_client.cpp
multilabelmaPrecision_client.cpp
PWeight_RRC_client.cpp
pythonRunner_client.cpp
RemoveEvLogFile_client.cpp
SDH_client.cpp
SDHWrapper_client.cpp
double2txt_client.cpp
EmptyWeightPool_client.cpp
evaluate_client.cpp
FileExists_client.cpp
GetCurrentTimeStamp_client.cpp
getAnchor_client.cpp
getLabel_client.cpp
hammingDist_client.cpp
LoadData_client.cpp
main_client.cpp
meanstd_client.cpp
meanstdPerAggregator_client.cpp
multilabelmaPrecision_client.cpp
PWeight_RRC_client.cpp
pythonRunner_client.cpp
RemoveEvLogFile_client.cpp
SDH_client.cpp
SDHWrapper_client.cpp
double2txt_client.cpp
EmptyWeightPool_client.cpp
evaluate_client.cpp
FileExists_client.cpp
GetCurrentTimeStamp_client.cpp
getAnchor_client.cpp
getLabel_client.cpp
hammingDist_client.cpp
LoadData_client.cpp
meanstdPerAggregator.cpp
```

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

The screenshot shows the Qt Creator IDE interface with the following details:

- Project Tree:** Shows the project structure with files like main_client.cpp, main.cpp, and NANODemo1.pro.
- Code Editor:** Displays the main_client.cpp file with several code snippets highlighted by red boxes. One snippet is around line 128: `string databaseName = "#DB";` Another is around line 138: `string switchDevice = "NANO";` A third is around line 148: `int LocalIterationSDH = 265;` A fourth is around line 158: `string XtranPath, LtranPath, LtestPath, LabelCountPath;` A fifth is around line 168: `vector<double> Eva_acc, Eva_rec, Eva_f1, Eva_MAP, Eva_TrainTime, Eva_OneAggrTime;` A sixth is around line 178: `stringstream ProxMu;` A seventh is around line 188: `ProxMu << setprecision(1) << SDH_K.ProxMu;` A eighth is around line 198: `// 調整數值` A ninth is around line 208: `string weightPoolPath, evaPoolPath;` A tenth is around line 218: `string XtranPath, XtestPath, LtranPath, LtestPath, LabelCountPath;` A eleventh is around line 228: `vector<double> Eva_acc, Eva_rec, Eva_f1, Eva_MAP, Eva_TrainTime, Eva_OneAggrTime;` A twelfth is around line 238: `string logOneAggrMemory.py;` A thirteenth is around line 248: `string logOneAggrMemory.txt;` A fourteenth is around line 258: `Parent process is running. Parent process is done.

Toolbars: The toolbar includes standard icons for file operations, project management, and build.

Status Bar: Shows the current file path: main_client.cpp @ NANODemo1 - Qt Creator.

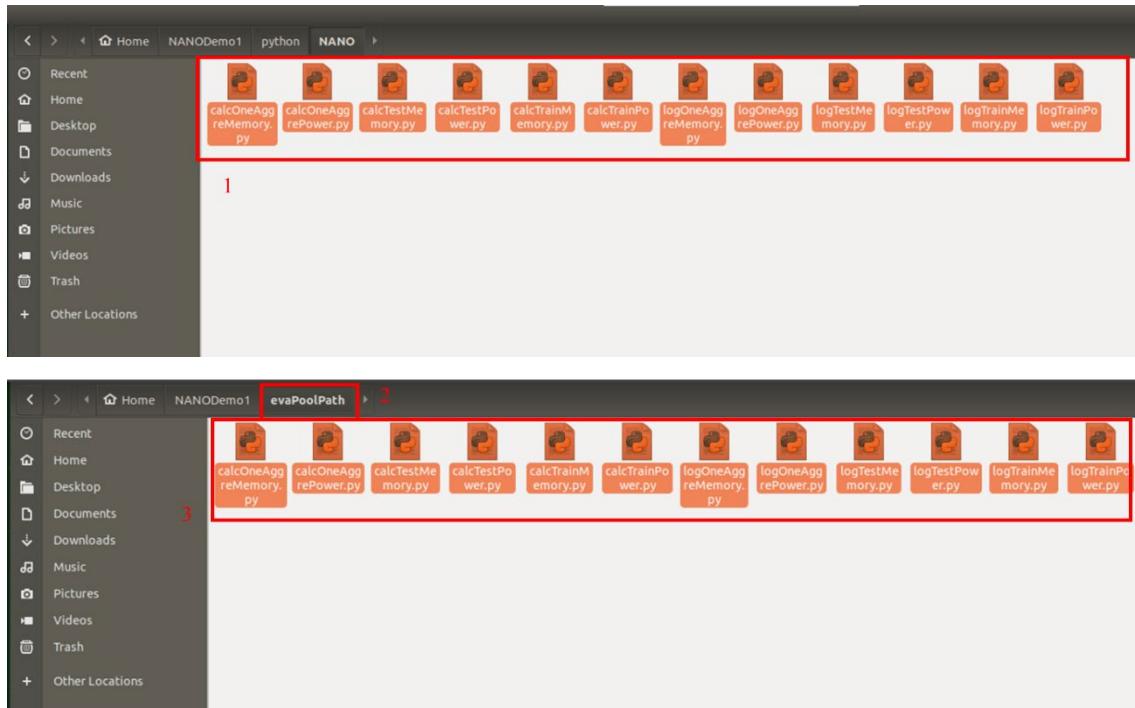
Bottom Panel: Shows the Application Output tab with the number 4 highlighted in red, displaying terminal logs.

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

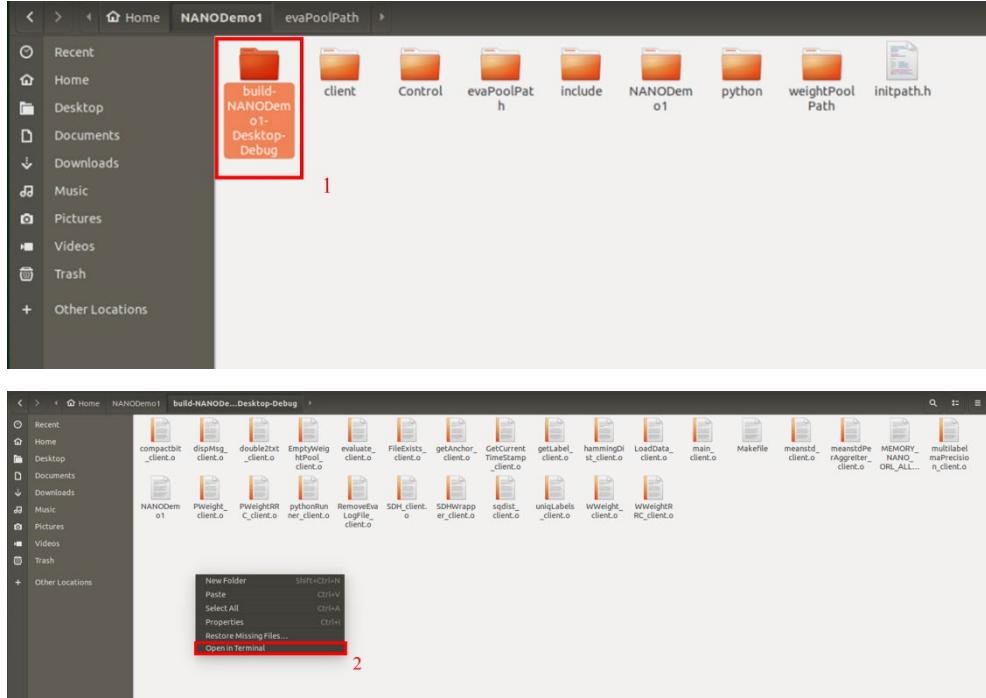


步驟 24. 打開終端機執行程式 ➤ 紅框 1：編輯完程式後會自動創建一個資料夾

build-NANODemo1-unknown-Debug ([/home/bn10513137/專案名/build-](#)

NANODemo1-unknown-Debug

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
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$ sudo pkill -9 -f python
[sudo] password for bn10513137:
Killed
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$ sudo pkill -9 -f python3
[sudo] password for bn10513137:
Killed
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$ sudo pkill -9 -f NANODemo1
[sudo] password for bn10513137:
Killed
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$ sudo pkill -9 -f ./NANODemo1
[sudo] password for bn10513137:
Killed
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$./NANODemo1
2

3
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$ sudo pkill -9 -f python
[sudo] password for bn10513137:
Killed
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$ sudo pkill -9 -f python3
[sudo] password for bn10513137:
Killed
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$ sudo pkill -9 -f NANODemo1.cpp
[sudo] password for bn10513137:
Killed
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$ sudo pkill -9 -f ./NANODemo1.cpp
[sudo] password for bn10513137:
Killed
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$./NANODemo1
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

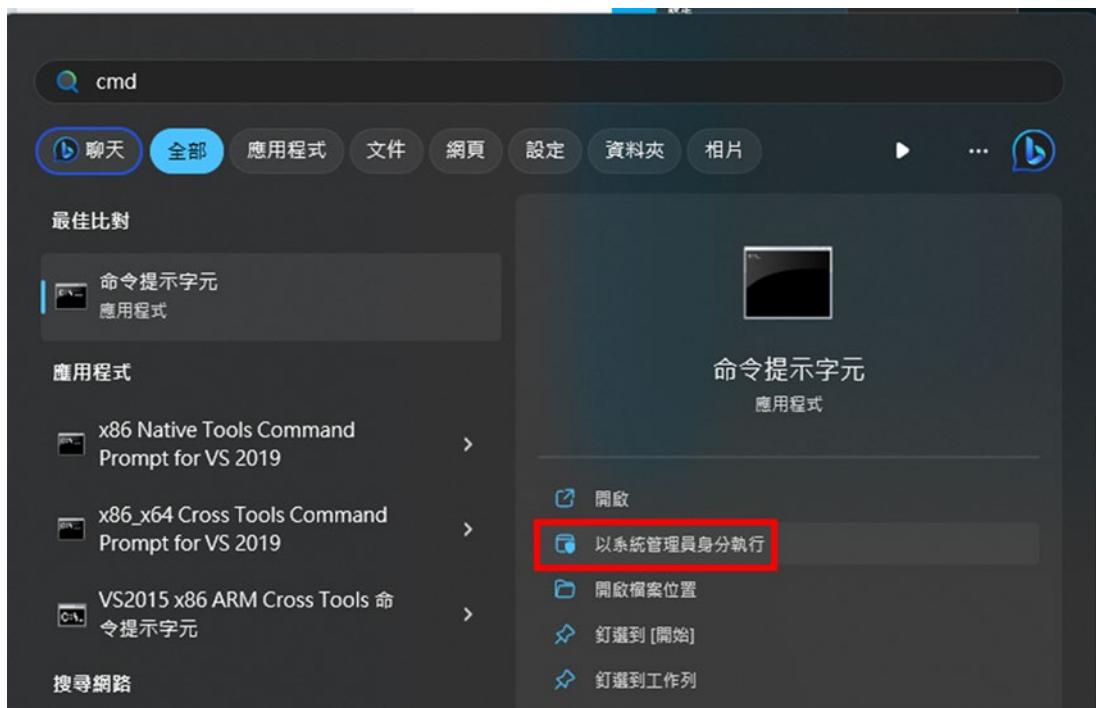
```
sudo pkill -9 -f python  
sudo pkill -9 -f python3  
sudo pkill -9 -f 專案檔名(例如紅框 2 的 RASPDemo1)  
sudo pkill -9 -f ./專案檔名(需在檔名前面加上 ./)
```

➤ 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)



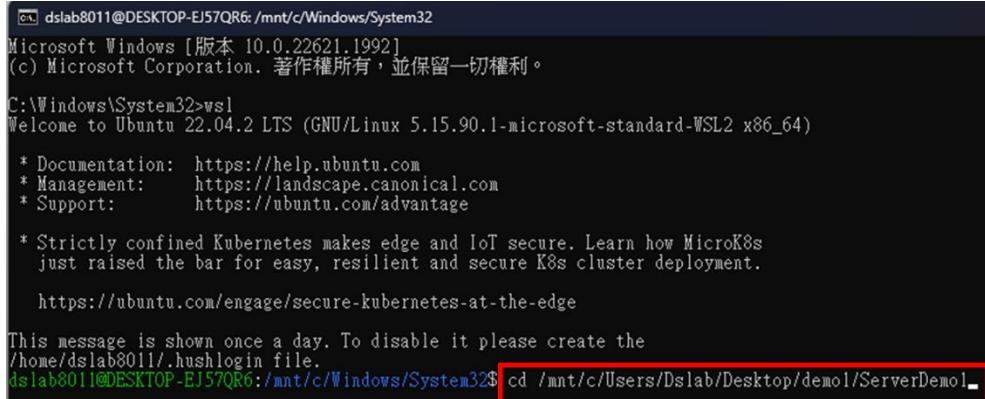
於 cmd 上開啟 WSL 系統

```
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

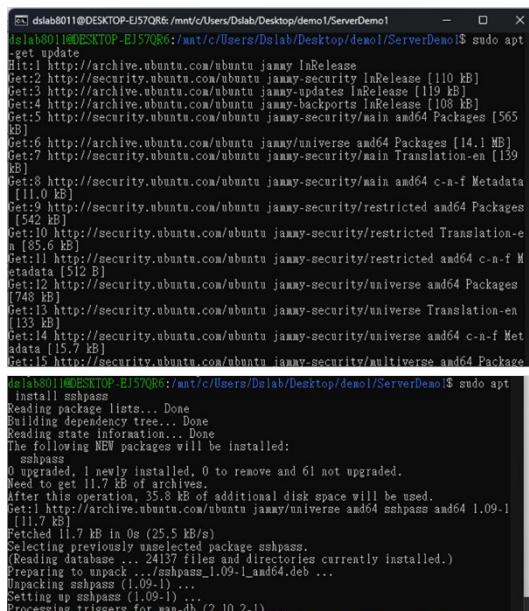
```
# 進入到剛創建的 Server 端專案 Server1 資料夾中  
# cd /mnt/原來 windows 專案路徑位置  
cd /mnt/c/Users/Dslab/Desktop/demo1/ServerDemo1
```



```
dslab8011@DESKTOP-EJ57QR6:/mnt/c/Windows/System32  
Microsoft Windows [版本 10.0.22621.1992]  
(c) Microsoft Corporation. 著作權所有，並保留一切權利。  
C:\Windows\System32>wsl  
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)  
  
* Documentation: https://help.ubuntu.com  
* Management: https://landscape.canonical.com  
* Support: https://ubuntu.com/advantage  
  
* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s  
just raised the bar for easy, resilient and secure K8s cluster deployment.  
https://ubuntu.com/engage/secure-kubernetes-at-the-edge  
This message is shown once a day. To disable it please create the  
/home/dslab8011/.hushlogin file.  
dslab8011@DESKTOP-EJ57QR6:/mnt/c/Windows/System32$ 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
```



```
dslab8011@DESKTOP-EJ57QR6:/mnt/c/Users/Dslab/Desktop/demo1/ServerDemo1$ sudo apt  
-get update  
Get:1 http://archive.ubuntu.com/ubuntu jammy InRelease  
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]  
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [19 kB]  
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [108 kB]  
Get:5 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [565 kB]  
Get:6 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]  
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [159 kB]  
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata  
[11.0 kB]  
Get:9 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages  
[542 kB]  
Get:10 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-e  
n [816 kB]  
Get:11 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 c-n-f M  
etadata [512 B]  
Get:12 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages  
[748 kB]  
Get:13 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en  
[133 kB]  
Get:14 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Met  
adata [15.7 kB]  
Get:15 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Package  
s [10 kB]  
dslab8011@DESKTOP-EJ57QR6:/mnt/c/Users/Dslab/Desktop/demo1/ServerDemo1$ sudo apt  
-install sshpass  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following NEW packages will be installed:  
  sshpass  
0 upgraded, 1 newly installed, 0 to remove and 61 not upgraded.  
Need to get 1.1 MB of additional disk space.  
After this operation, 35.8 kB of additional disk space will be used.  
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 sshpass amd64 1.09-1  
[11.7 kB]  
Fetched 11.7 kB in 0s (25.5 kB/s)  
Selecting previously unselected package sshpass.  
(Reading database ... 24137 files and directories currently installed.)  
Preparing to unpack .../sshpass_1.09-1_amd64.deb ...  
Unpacking sshpass (1.09-1)...  
Setting up sshpass (1.09-1)...  
Processing triggers for man-db (2.10.2-1) ...
```

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 rasdslab@100.82.153.59  
  
ssh-copy-id -i /home/dslab8011/.ssh/id_rsa bn10513137@100.66.82.83
```

```
dslab8011@DESKTOP-BJ57QR6:/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:nmcMowkKWuASeQa2AFeJ3aeCm9z35XAcc6IkouIdPk4 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-BJ57QR6:/mnt/c/Users/Dslab/Desktop/demo1/ServerDemo1$ ssh-copy-id -i /home/dslab8011/.ssh/id_rsa rasd  
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/ rasdslab@100.8  
2.153.59: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/ rapsdlslab@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

```
31 // SERVER PATH 使用，接著因為 SERVER 端程式只在 x64上使用，所以只在此處定義巨集
32 #define INIT_SERVER_weightPoolPath      ".../weightPoolPath"
33 #define INIT_SERVER_CTRL_MSG_1          "SERVER_SEND_TRAIN_TO_NODE"
34 #define INIT_SERVER_CTRL_MSG_2          "NODE_SEND_WEIGHT_TO_SERVER"
35 #define INIT_SERVER_CTRL_MSG_3          "SERVER_SEND_AGGR_TO_NODE"
36 #define INIT_SERVER_CTRL_MSG_4          "SERVER_SEND_SERVEREnd_TO_NODE"
37 #define INIT_SERVER_CTRL_MSG_5          "NODE_SEND_ClientEnd_TO_SERVER"
38 #define INIT_SERVER_SERVER2RASP         "/SERVER2RASP"
39 #define INIT_SERVER_SERVER2NANO         "/SERVER2NANO"
40 #define INIT_SERVER_XtranMtxTxt        "X_tran_matrix.txt"
41 #define INIT_SERVER_XtuneMtxTxt        "X_tune_matrix.txt"
42 #define INIT_SERVER_XtestMtxTxt        "X_test_matrix.txt"
43 #define INIT_SERVER_LtranMtxTxt        "L_tran_matrix.txt"
44 #define INIT_SERVER_LtuneMtxTxt        "L_tune_matrix.txt"
45 #define INIT_SERVER_LtestMtxTxt        "L_test_matrix.txt"
46 #define INIT_SERVER_RASP2SERVER        "/RASP2SERVER"
47 #define INIT_SERVER_NANO2SERVER        "/NANO2SERVER"
48 #define INIT_SERVER_FweightMtxTxt      "FWeight_matrix.txt"
49 #define INIT_SERVER_WweightMtxTxt      "WWeight_matrix.txt"
50 #define INIT_SERVER_LabelListMtxTxt    "Labellist_matrix.txt"
51 #define INIT_SERVER_LabelCountMtxTxt  "LabelCount_matrix.txt"
52 #define INIT_SERVER_ControlFile        "SEND_FILE_CTRL_MSG.txt"
53 #define INIT_SERVER_Control2SERVERPath 1 " /mnt/c/Users/Dslab/Desktop/demo1/weightPoolPath/ "
54 #define INIT_SERVER_Control2RASPPath   2 " raspdslab@100.82.153.59:RaspClientDemo1/weightPoolPath/ "
55 #define INIT_SERVER_Control2NANOPath   3 " bn10513137@100.66.82.83:NANOClientDemo1/weightPoolPath/ "
56 #define INIT_x64CLIENT_weightPoolPath  ".../weightPoolPath"
```

➤ 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:NANOClientDemo1/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
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
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
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

```
// 參數設定
string datasetnam
vector<int> bitlen
int nChannels
param.nExps
param.J
param.L
param.AggreIter
// 參數設定 NANO_train&test_SPEED_FedProxFSDH_JAFFE_EXP10_ITR5_16B_SI1_MU10_5_1
string datasetnam 1 = "ORL";
string SwitchDevice = "RASP";
string SwitchMethod = "AVG_SDH";
string SwitchSegment 3 = "ALL";
string SwitchEvaluation = "MEMORY";
string switchEval_METHOD = "micro";
vector<int> bitlen = { 16, 32 , 64, 96, 128 };
int nChannels
param.nExps
param.J
param.L
param.AggreIter
// 參數設定 NANO_train&test_SPEED_FedProxFSDH_JAFFE_EXP10_ITR5_16B_SI1_MU10_5_1
string datasetnam 1 = "ORL";
string SwitchDevice = "NANO";
string SwitchMethod = "AVG_SDH";
string SwitchSegment 3 = "ALL";
string SwitchEvaluation = "MEMORY";
string switchEval_METHOD = "micro";
vector<int> bitlen = { 16, 32 , 64, 96, 128 };
int nChannels
param.nExps
param.J
param.L
param.AggreIter
```

步驟 2. 檢查專案 SERVER1 裡的資料集路徑是否正確



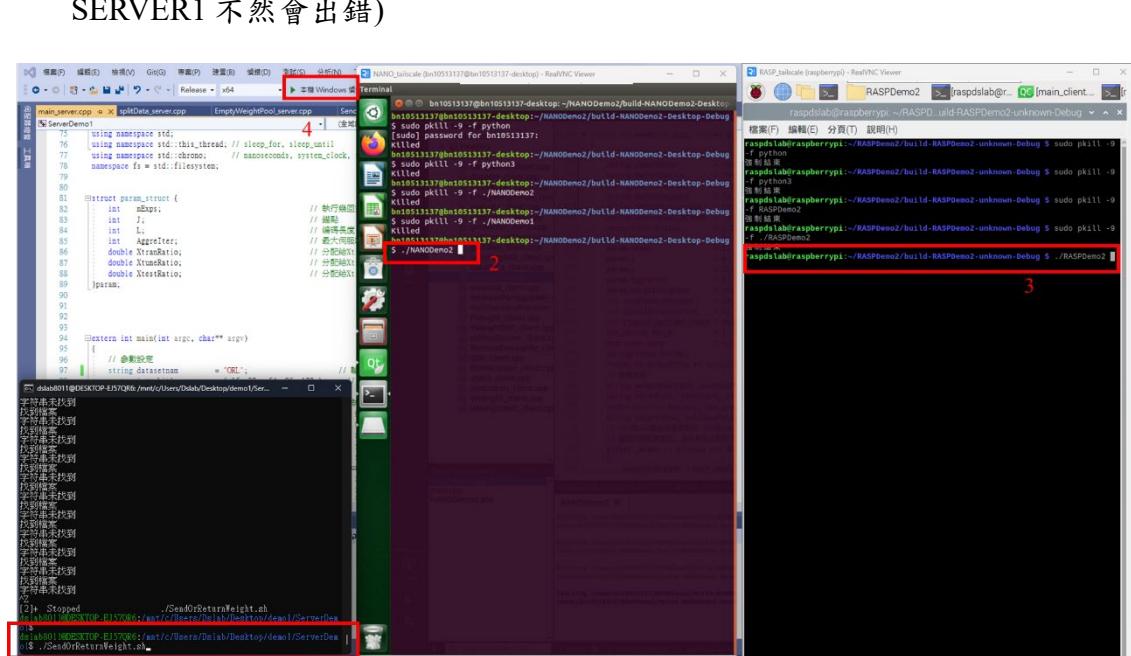
```
150
151
152
153     // 取用資料集(請看或是第 30 行)
154     // YALE
155     if (datasetnam.compare("YALE") == 0)
156     {
157         filepath_x_lin_data = "../Dataset/yale64x64_data_x_lin.txt";
158         filepath_t_data      = "../Dataset/yale64x64_data_t.txt";
159     }
160     // JAFFE
161     else if (datasetnam.compare("JAFFE") == 0)
162     {
163         filepath_x_lin_data = "../Dataset/jaffe64x64_data_x_lin.txt";
164         filepath_t_data      = "../Dataset/jaffe64x64_data_t.txt";
165     }
166     // ORL
167     else if (datasetnam.compare("ORL") == 0)
168     {
169         filepath_x_lin_data = "../Dataset/orl64x64_data_x_lin.txt";
170         filepath_t_data      = "../Dataset/orl64x64_data_t.txt";
171     }
172     // SMALL_ARFACE
173     else if (datasetnam.compare("SMALL_ARFACE") == 0)
174     {
175         filepath_x_lin_data = "../Dataset/SMALL_2340_ARface64x64_data_x_lin.txt";
176         filepath_t_data      = "../Dataset/SMALL_2340_ARface64x64_data_t.txt";
177     }
178 }
```

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

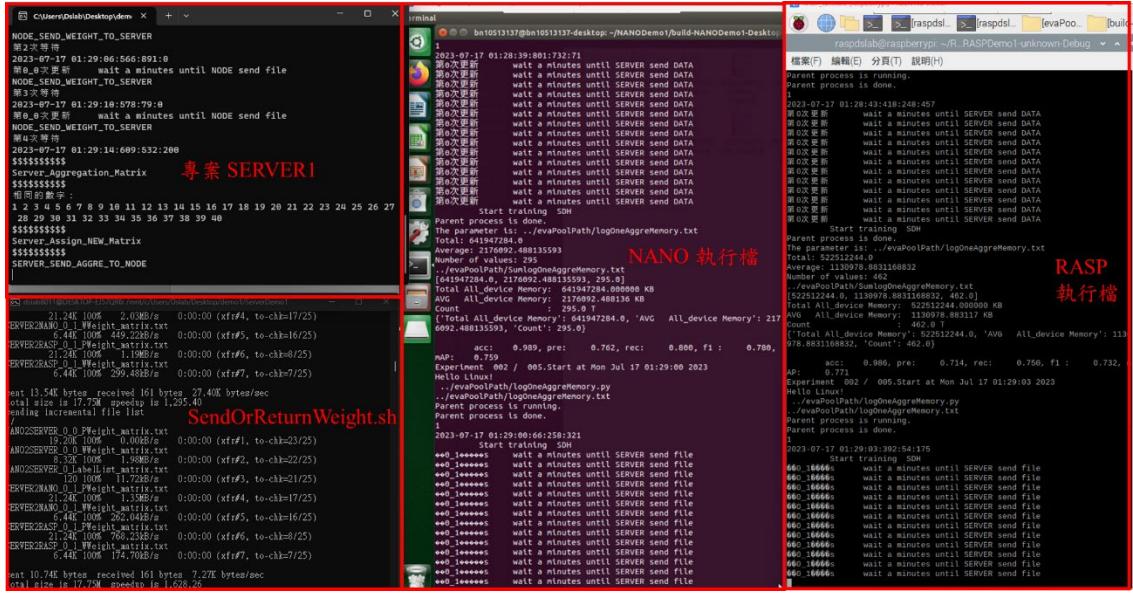
步驟 4. 依序執行紅框 1：SendOrReturnWeight.sh、紅框 2：NANO 執行檔、紅

框 3：RASP 執行檔和紅框 4：專案 SERVER1 (最後一個一定要是專案

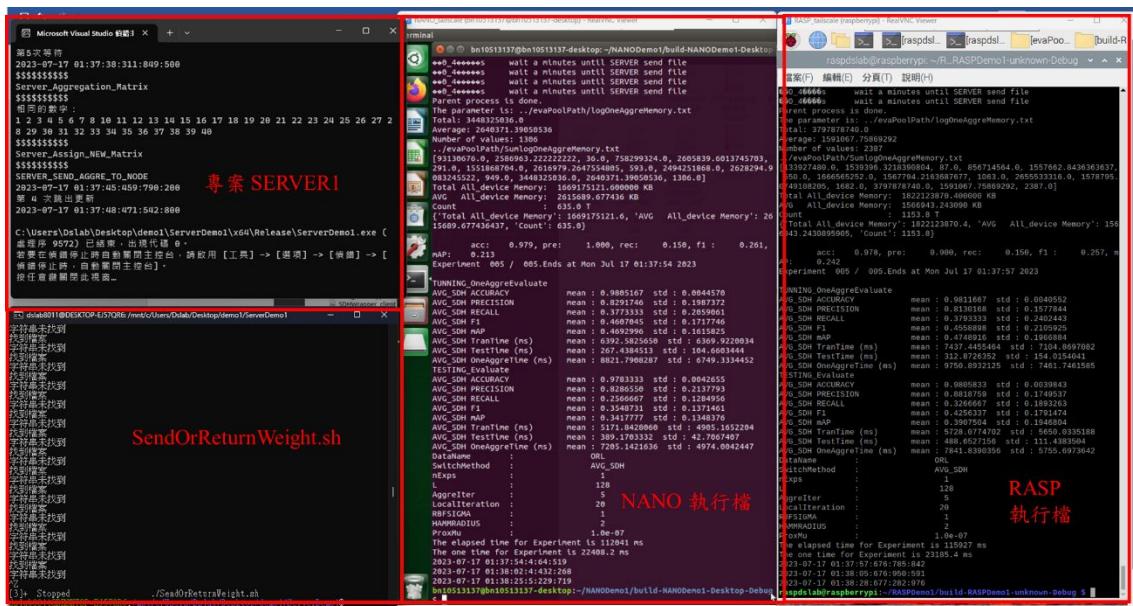
SERVER1 不然會出錯)



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。

The screenshot shows a file explorer window titled "evaPoolPath" and a code editor window titled "SumlogOneAggreMemory.csv".

File Explorer (Top Window):

- Path: /home/raspdslab/RASPDemo1/evaPoolPath
- Selected File: SumlogOneAggreMemory.csv
- Content List:

 - SumlogOneAggreMemory.txt
 - SumlogOneAggreMemory.csv
 - logOneAggreMemory.txt
 - threadName.txt
 - logOneAggreMemory.py
 - logTestMemory.py
 - logTrainMemory.py
 - calcTestMemory.py
 - calcTrainMemory.py
 - calcOneAggreMemory.py
 - calcTrainPower.py
 - calcTestPower.py
 - INA219.py
 - calcOneAggrePower.py

Code Editor (Bottom Window):

```

SumlogOneAggreMemory.csv - /
Total All device Memory, AVG All device Memory, Count
522512244.0, 1130978.8831168832, 462.0
754819790.0, 1140798.1222111224, 660.0
976819798.6666666, 1148070.9401756637, 847.333333333334
1179383757.0, 1154016.4234924035, 1016.75
1375686228.8, 1159505.770395891, 1179.2
160772412.0, 1246297.7674418604, 129.0
303990528.0, 1253020.213298395, 242.0
451548694.6666667, 1258626.3031966623, 357.333333333333
623408144.0, 1264269.6407833234, 490.25
817366892.8, 1270052.382756529, 638.6
511248384.0, 1341859.275590551, 381.0
713996442.0, 1346994.9032820011, 529.5
927294554.6666666, 1352017.655172575, 684.333333333334
1151315596.0, 1357011.4957059924, 845.5
1365423310.4, 1361787.4648059376, 998.2
720870228.0, 1441740.456, 500.0
1027679422.0, 1446924.97011099, 709.5
1300872978.6666667, 1451378.1993098299, 894.66666666666666
1582689281.0, 1455987.9757535588, 1084.0
1882155164.8, 1460946.0248336163, 1283.2
133927480.0, 1539396.3218390804, 87.0
495321022.0, 1548529.582737722, 318.5
885735765.3333334, 1554951.1272814039, 566.66666666666666
1328185153.0, 1560912.114188758, 845.5
1822123870.4, 1566943.2430895905, 1153.8

```

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。

	Total All device Memory	Avg All device Memory	Count
1	642049672.0	2599391.3846153845	247.0
2	9099090966.0	2609419.812574953	348.0
3	1168735434.66666667	2617420.994134732	445.66666666666667
4	1436841103.0	2624564.5968407183	546.0
5	1700548422.4	2631064.927231495	644.2
6	206940764.0	2722904.789473684	76.0
7	444584084.0	2731386.940921581	162.5
8	699953962.6666666	2738116.839402266	255.0
9	958541777.0	2743995.758532591	348.25
10	1231099937.6	2749875.8374114973	446.0
11	486222812.0	2826876.8139534886	172.0
12	845680850.0	2834592.756033348	298.0
13	1132049573.3333333	2839999.9755941383	398.0
14	1384310824.0	2844651.0417757104	485.75
15	1633504720.0	2849391.564936381	572.0
16	774358548.0	2911122.360902256	266.0
17	1297329090.0	2918824.5502260476	444.0
18	1695352433.3333333	2924053.0845244713	579.0
19	2031206357.0	2928462.884061311	692.5
20	2348406818.4	2932853.286694399	799.2
21	962189304.0	3016267.4106583074	319.0
22	1294029214.0	3021978.3273030827	428.0
23	1757020089.3333333	3028636.28471755	579.3333333333334
24	2304830456.0	3035459.7127641686	757.5
25	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

```

File Edit Build Debug Analyze Tools Window Help
Projects > NANODemo2 > Sources main_client.cpp
123 // 參數設定 NANO_train&test_SPEED_FedProxFSDH_JAFFE_EXP10_ITR5_16B_SI1_MU10_5_1
124 extern int main(int argc, char** argv)
125 {
126     string datasetnam = "ORL";
127     string SwitchDevice = "NANO";
128     string SwitchMethod = "AVG_SDH";
129     string SwitchSegment = "ALL";
130     string switchEvaluation = "MEMORY"; 1
131     string switchEval_METHOD = "micro";
132     vector<int> bitlen = { 16, 32 , 64, 96, 128 };
133     int nChannels;
134     param.nExps = 1;
135     param.J = 0;
136     param.L = 32;
137     param.AggreIter = 5;
138     param.Localization = 10;
139     int LocalizationSDH = 20;
140     int LocalizationFSDH = 3;
141     int timeout_wait4WP_limit = 168;
142 } // 測試資料集("YALE", "JAFFE", "ORL", "SMALL_ARFACE")
// 選擇在那個裝置上進行權重更新(NANO , RASP)
// 選擇何種類型學習或模型(AVG_SDH, AVG_FSDH, PROX_SDH 和 PROX_FSDH)
// 選擇量測完整一回合型更新區段或是量測訓練或模型測試區段進行(ALL 和 TRAINTEST)
// 選擇量測何種評估指標 (SPEED, POWER 和 MEMORY)
// Accuracy, Precision, Recall 和 F1 使用 micro 平均
// 程式中須測試的項目，主要碼長設定在param.L
// If RGB, nChannels = 3
// 執行幾回完整的模型更新
// 總點
// 編碼長度(先宣告, 32為預設值)
// 最大伺服器聚合轉次
// 局部權重更新次數
// SDH 專用局部權重更新次數
// FSDH 專用局部權重更新次數
// 當掃描此資料集是否有指定權重的上限次數

```



```

File Edit Build Debug Analyze Tools Window Help
Projects > NANODemo2 > Sources main_client.cpp
123 // 參數設定 NANO_train&test_SPEED_FedProxFSDH_JAFFE_EXP10_ITR5_16B_SI1_MU10_5_1
124 extern int main(int argc, char** argv)
125 {
126     string datasetnam = "ORL";
127     string SwitchDevice = "NANO";
128     string SwitchMethod = "AVG_SDH";
129     string SwitchSegment = "ALL";
130     string switchEvaluation = "POWER"; 2
131     string switchEval_METHOD = "micro";
132     vector<int> bitlen = { 16, 32 , 64, 96, 128 };
133     int nChannels;
134     param.nExps = 1;
135     param.J = 0;
136     param.L = 32;
137     param.AggreIter = 5;
138     param.Localization = 10;
139     int LocalizationSDH = 20;
140     int LocalizationFSDH = 3;
141     int timeout_wait4WP_limit = 168;
142 } // 測試資料集("YALE", "JAFFE", "ORL", "SMALL_ARFACE")
// 選擇在那個裝置上進行權重更新(NANO , RASP)
// 選擇何種類型學習或模型(AVG_SDH, AVG_FSDH, PROX_SDH 和 PROX_FSDH)
// 選擇量測完整一回合型更新區段或是量測訓練或模型測試區段進行(ALL 和 TRAINTEST)
// 選擇量測何種評估指標 (SPEED, POWER 和 MEMORY)
// Accuracy, Precision, Recall 和 F1 使用 micro 平均
// 程式中須測試的項目，主要碼長設定在param.L
// If RGB, nChannels = 3
// 執行幾回完整的模型更新
// 總點
// 編碼長度(先宣告, 32為預設值)
// 最大伺服器聚合轉次
// 局部權重更新次數
// SDH 專用局部權重更新次數
// FSDH 專用局部權重更新次數
// 當掃描此資料集是否有指定權重的上限次數

```

RASP

```

File Edit Build Debug Analyze Tools Window Help
Projects > double2txt_client > Sources main_client.cpp
124 extern int main(int argc, char** argv)
125 {
126     // 參數設定 NANO_train&test_SPEED_FedProxFSDH_JAFFE_EXP10_ITR5_16B_SI1_MU10_5_1
127     string datasetnam = "ORL";
128     string SwitchDevice = "RASP";
129     string SwitchMethod = "AVG_SDH";
130     string SwitchSegment = "ALL";
131     string switchEvaluation = "MEMORY"; 3
132     string switchEval_METHOD = "micro";
133     vector<int> bitlen = { 16, 32 , 64, 96, 128 };
134     int nChannels;
135     param.nExps = 1;
136     param.J = 0;
137     param.L = 32;
138     param.AggreIter = 5;
139     param.Localization = 10;
140     int LocalizationSDH = 20;
141     int LocalizationFSDH = 3;
142     int timeout_wait4WP_limit = 168;
143 } // 測試資料集("YALE", "JAFFE", "ORL", "SMALL_ARFACE")
// 選擇在那個裝置上進行權重更新(NANO , RASP)
// 選擇何種類型學習或模型(AVG_SDH, AVG_FSDH, PROX_SDH 和 PROX_FSDH)
// 選擇量測完整一回合型更新區段或是量測訓練或模型測試區段進行(ALL 和 TRAINTEST)
// 選擇量測何種評估指標 (SPEED, POWER 和 MEMORY)
// Accuracy, Precision, Recall 和 F1 使用 micro 平均
// 程式中須測試的項目，主要碼長設定在param.L
// If RGB, nChannels = 3
// 執行幾回完整的模型更新
// 總點
// 編碼長度(先宣告, 32為預設值)
// 最大伺服器聚合轉次
// 局部權重更新次數
// SDH 專用局部權重更新次數
// FSDH 專用局部權重更新次數
// 當掃描此資料集是否有指定權重的上限次數
unused parameter 'argc' □ unused parameter 'argv'

```



```

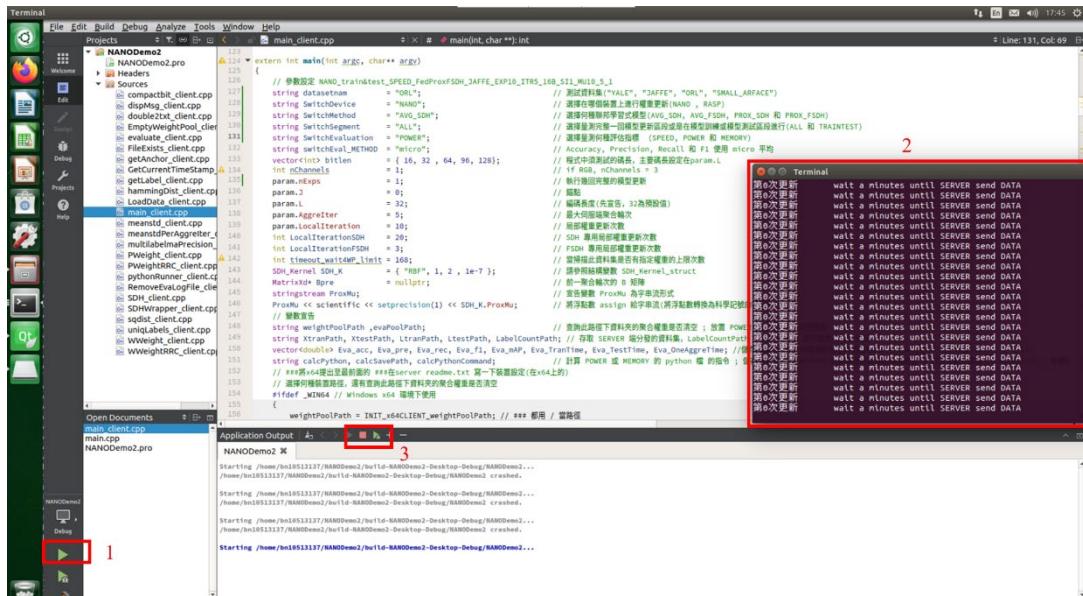
File Edit Build Debug Analyze Tools Window Help
Projects > compactbit_client > Sources main_client.cpp
124 extern int main(int argc, char** argv)
125 {
126     // 參數設定 NANO_train&test_SPEED_FedProxFSDH_JAFFE_EXP10_ITR5_16B_SI1_MU10_5_1
127     string datasetnam = "ORL";
128     string SwitchDevice = "RASP";
129     string SwitchMethod = "AVG_SDH";
130     string SwitchSegment = "ALL";
131     string switchEvaluation = "POWER"; 4
132     string switchEval_METHOD = "micro";
133     vector<int> bitlen = { 16, 32 , 64, 96, 128 };
134     int nChannels;
135     param.nExps = 1;
136     param.J = 0;
137     param.L = 32;
138     param.AggreIter = 5;
139     param.Localization = 10;
140     int LocalizationSDH = 20;
141     int LocalizationFSDH = 3;
142     int timeout_wait4WP_limit = 168;
143 } // 測試資料集("YALE", "JAFFE", "ORL", "SMALL_ARFACE")
// 選擇在那個裝置上進行權重更新(NANO , RASP)
// 選擇何種類型學習或模型(AVG_SDH, AVG_FSDH, PROX_SDH 和 PROX_FSDH)
// 選擇量測完整一回合型更新區段或是量測訓練或模型測試區段進行(ALL 和 TRAINTEST)
// 選擇量測何種評估指標 (SPEED, POWER 和 MEMORY)
// Accuracy, Precision, Recall 和 F1 使用 micro 平均
// 程式中須測試的項目，主要碼長設定在param.L
// If RGB, nChannels = 3
// 執行幾回完整的模型更新
// 總點
// 編碼長度(先宣告, 32為預設值)
// 最大伺服器聚合轉次
// 局部權重更新次數
// SDH 專用局部權重更新次數
// FSDH 專用局部權重更新次數
// 當掃描此資料集是否有指定權重的上限次數
unused parameter 'argc' □ unused parameter 'argv'
unused variable 'nChannels'
unused variable 'timeout_w...

```

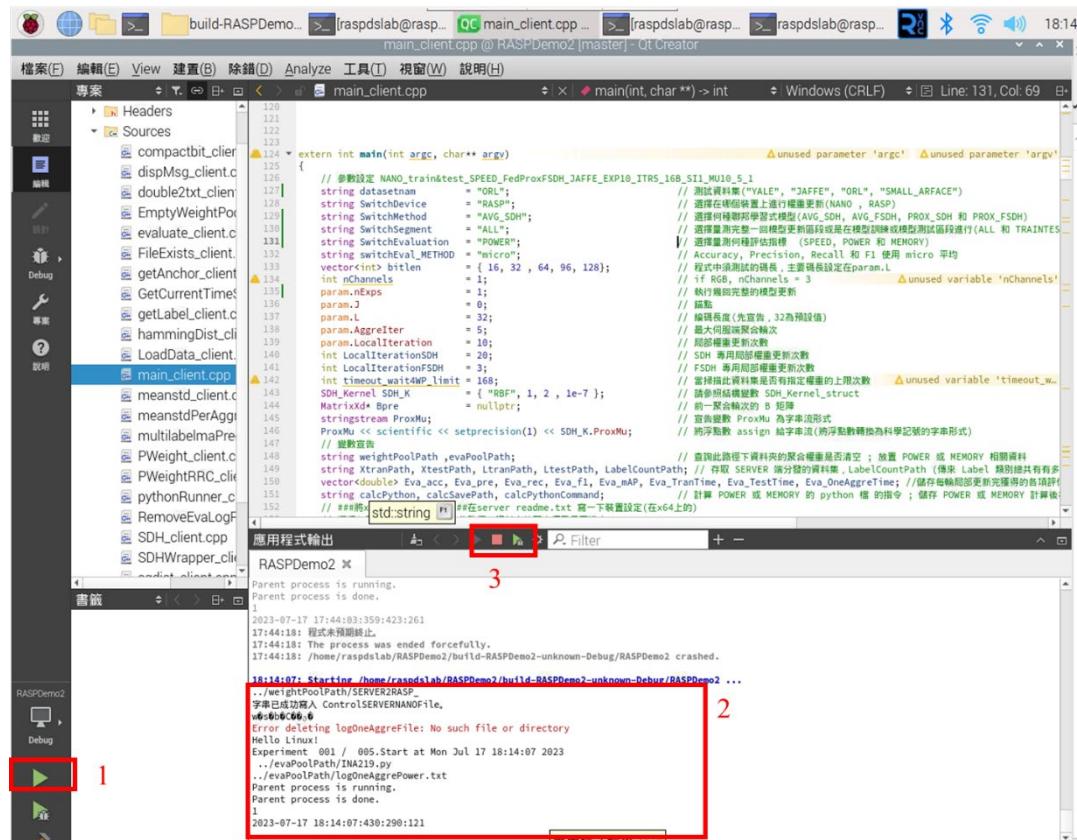
步驟 10. 將 NANO 專案和 RASP 專案進行編譯 ➤ 紅框 1：分別按下 NANO 專案

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



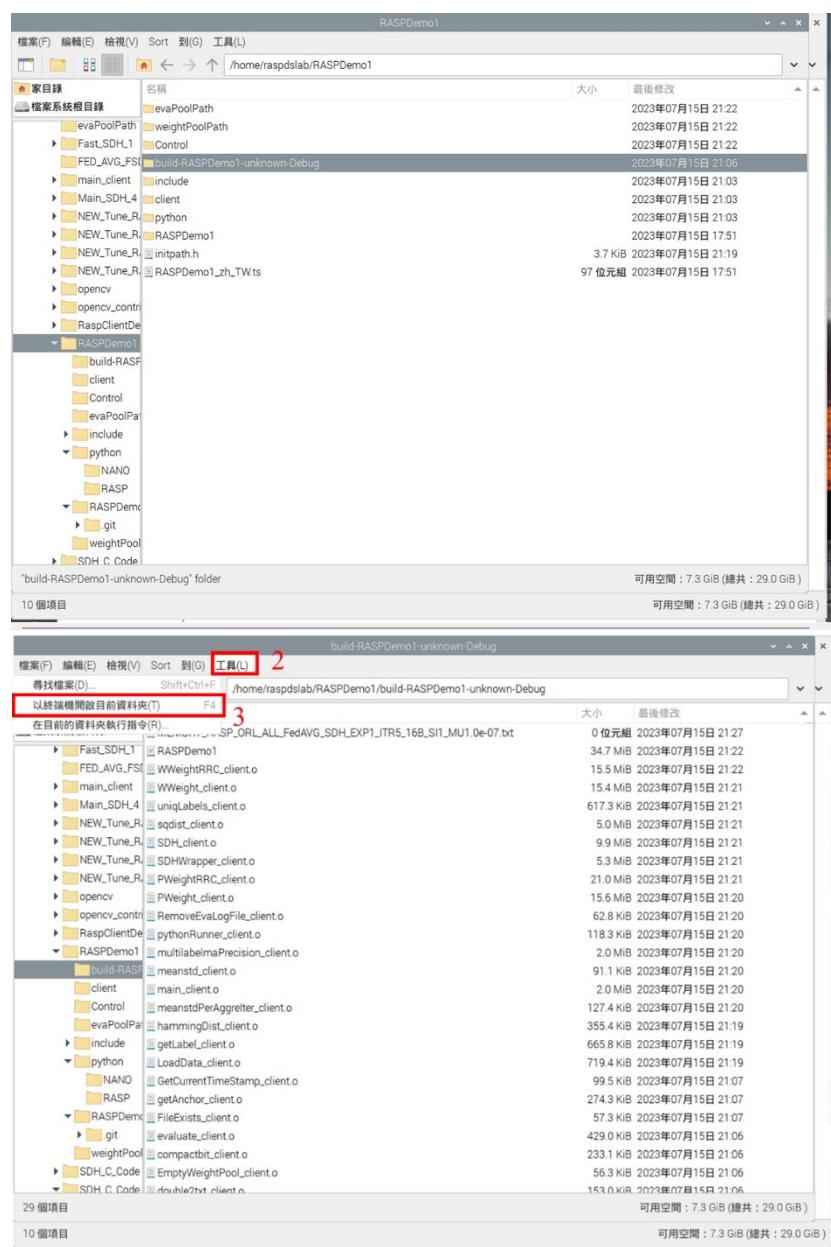
RASP



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



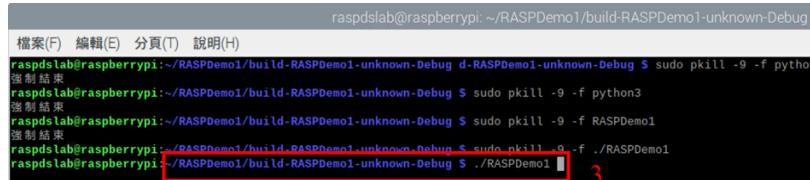
The screenshot shows a terminal window with the following text:

```
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug
強制結束
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug $ sudo pkill -9 -f python
強制結束
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug $ sudo pkill -9 -f python3
強制結束
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug $ sudo pkill -9 -f RASPDemo1
強制結束
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug $ sudo pkill -9 -f ./RASPDemo1
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug $ ./RASPDemo1
```

A red box highlights the first four lines of the command history, labeled '1' at the top right.

➤ 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 ./)
```



The screenshot shows a terminal window with the following text:

```
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug
強制結束
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug $ sudo pkill -9 -f python
強制結束
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug $ sudo pkill -9 -f python3
強制結束
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug $ sudo pkill -9 -f RASPDemo1
強制結束
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug $ sudo pkill -9 -f ./RASPDemo1
raspberrypi:~/RASPDemo1/build-RASPDemo1-unknown-Debug $ ./RASPDemo1
```

A red box highlights the last line of the command history, labeled '3' at the top right.

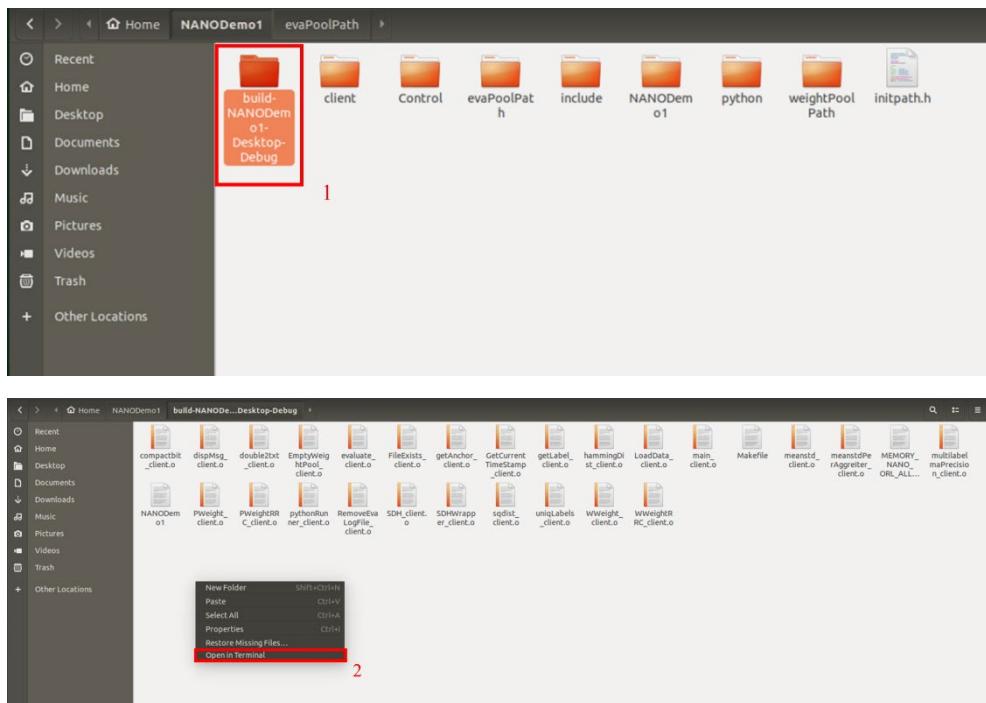
➤ 紅框 3：輸入執行程式指令(但先別按 **Enter** 執行)

```
# 執行程式執行檔
./專案檔名(需在檔名前面加上 ./)
```

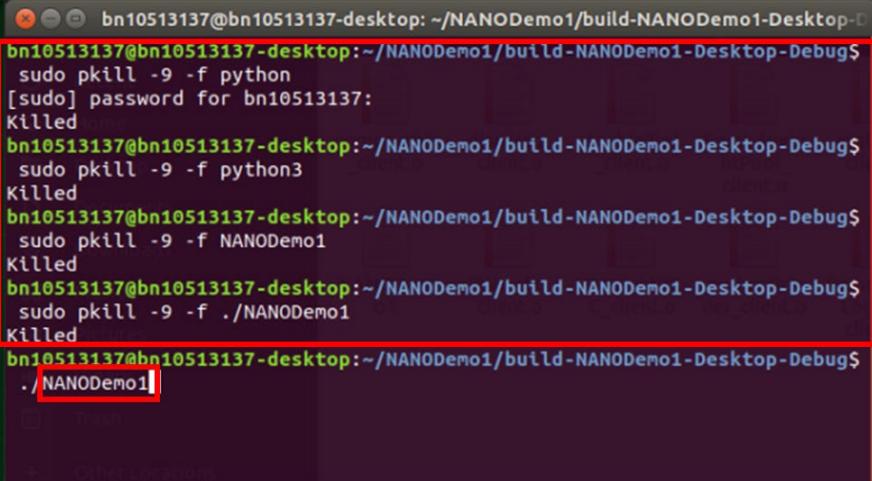
步驟 13. 打開 NANO 終端機執行程式 ➤ 紅框 1：編輯完程式後會自動創建一個資料夾 build-NANODemo1-unknown-Debug ([/home/bn10513137/專案名](#)

/build-NANODemo1-unknown-Debug)，裡面有我們編輯完的程式執行檔 ➔

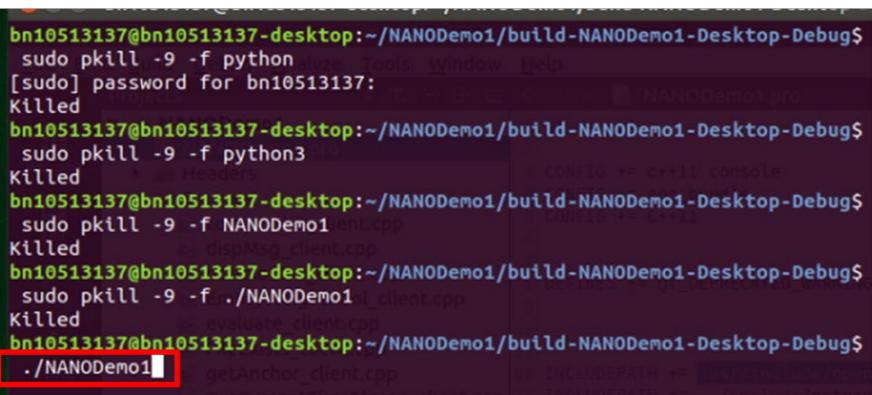
紅框 2：點擊右鍵 open in Terminal



步驟 14. 在終端機上輸入執行命令



1
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$ sudo pkill -9 -f python
[sudo] password for bn10513137:
Killed
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$ sudo pkill -9 -f python3
[sudo] password for bn10513137:
Killed
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$ sudo pkill -9 -f NANODemo1
Killed
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$ sudo pkill -9 -f ./NANODemo1
Killed
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$./NANODemo1



2
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$ sudo pkill -9 -f python
[sudo] password for bn10513137:
Killed
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$ sudo pkill -9 -f python3
[sudo] password for bn10513137:
Killed
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$ sudo pkill -9 -f NANODemo1.cpp
Killed
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$ sudo pkill -9 -f ./NANODemo1.cpp
Killed
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$./NANODemo1

3
bn10513137@bn10513137-desktop:~/NANODemo1/build-NANODemo1-Desktop-Debug\$./NANODemo1

➤ 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

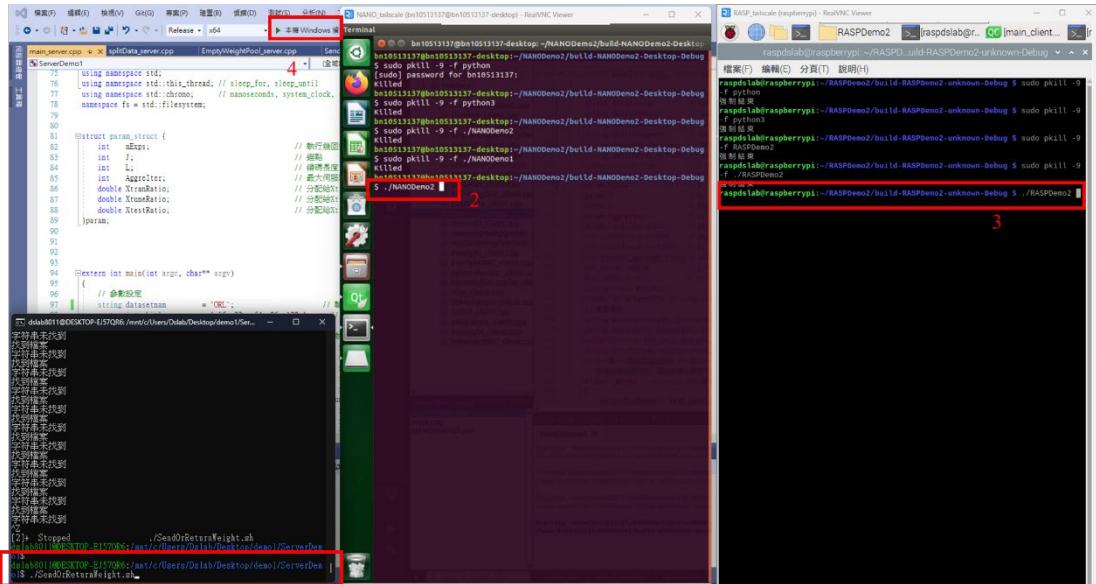
./專案檔名(需在檔名前面加上 ./)

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

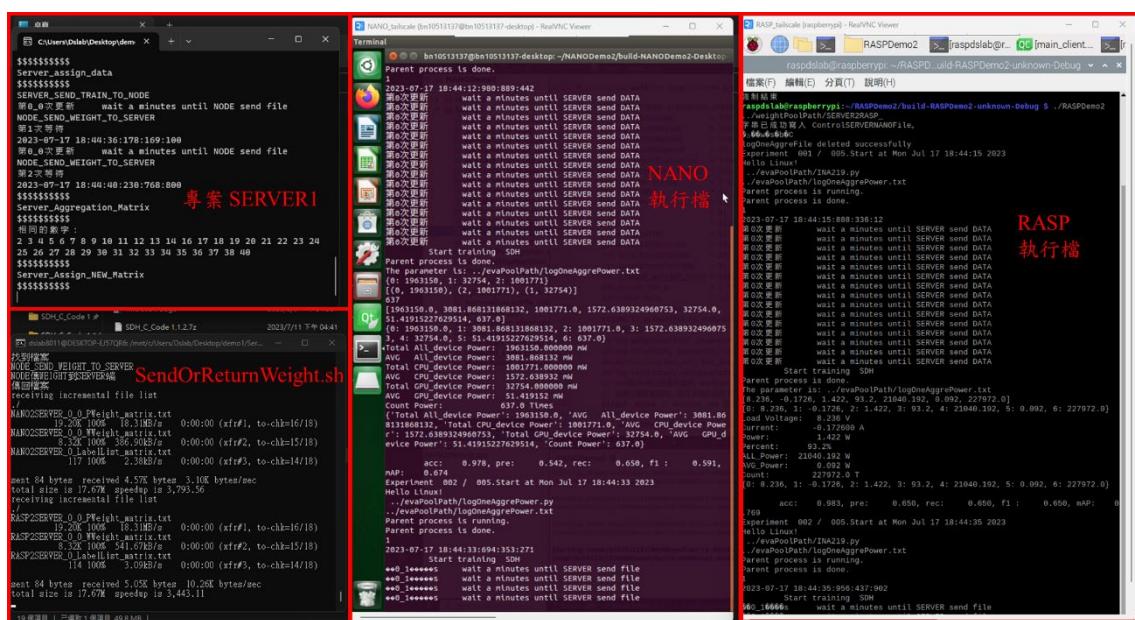
框 3：RASP 執行檔和紅框 4：專案 SERVER1 (最後一個一定要是專案

SERVER1 不然會出錯)



1

步驟 17. 程式執行中畫面



步驟 18. 程式執行結果如下

```

Microsoft Visual Studio 語譯 X + - ×
[Terminal]
nano_taiScale@bn10513137:~/Desktop - RealNC Viewer
$ nano_taiScale@bn10513137:~/Desktop - RealNC Viewer
$ ./ServerDemo1
[Terminal]
nano_taiScale@bn10513137:~/Desktop - RealNC Viewer
$ ./ServerDemo1
[Terminal]
nano_taiScale@bn10513137:~/Desktop - RealNC Viewer
$ ./RASPDemo2
[Terminal]
raspberrypi:~$ ./RASPDemo2

```

專業 SERVER

```

2023-07-17 18:52:32:332:288:i [INFO] [ServerDemo1] 次執行更新
2023-07-17 18:52:35:345:150:600 [INFO] [ServerDemo1] Experiment 005 / 005. Ends at Mon Jul 17 18:52:40 2023

```

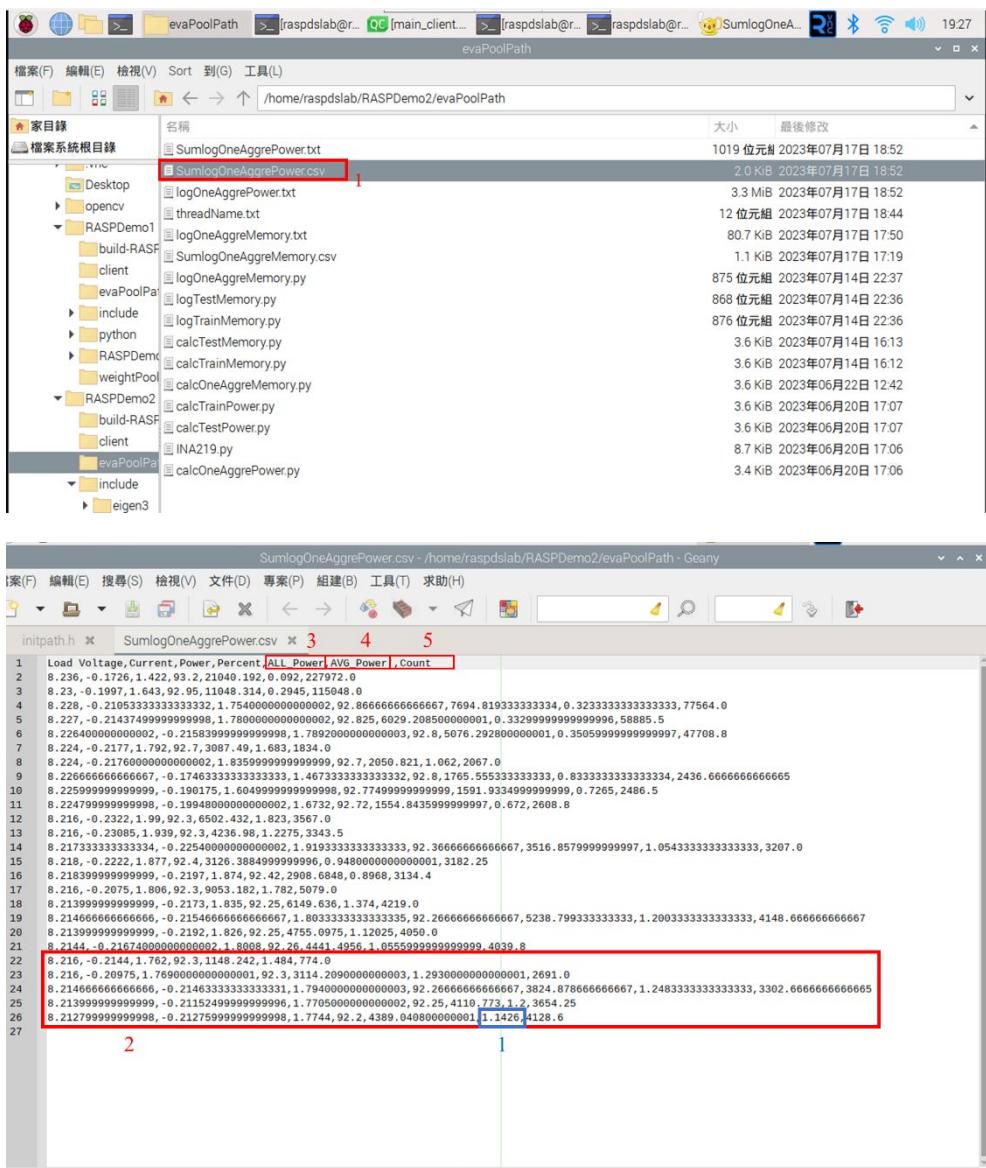
RASP

```

2023-07-17 18:52:32:332:288:i [INFO] [RASPDemo2] 次執行更新
2023-07-17 18:52:35:345:150:600 [INFO] [RASPDemo2] Experiment 005 / 005. Ends at Mon Jul 17 18:52:40 2023

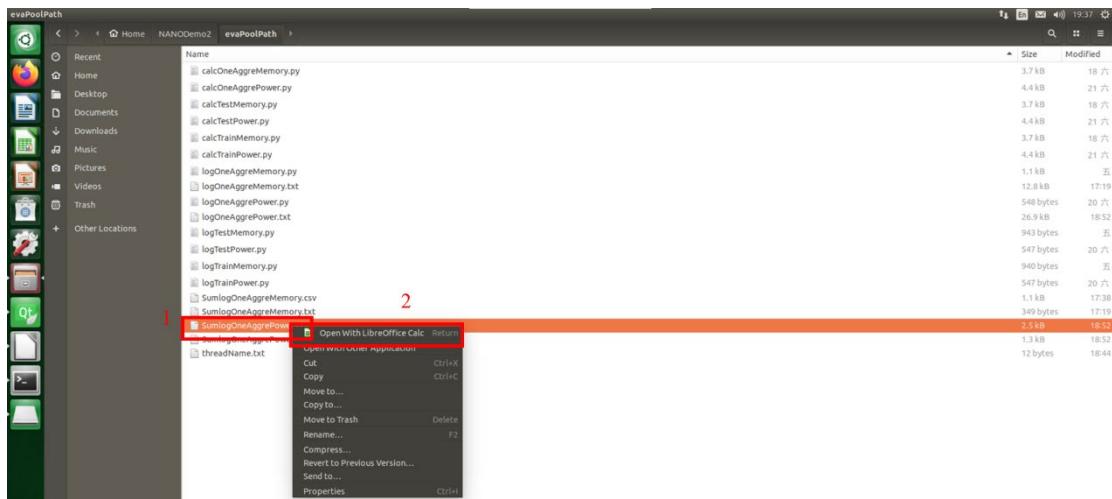
```

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。



Text Import - [SumlogOneAggrePower.csv]

Import
Character set: Western Europe (Windows-1252/WinLatin 1)
Language: Default - English (USA)
From row: 1

Separator Options
Fixed width (unchecked)
Tab (checked)
Comma (unchecked)
Semicolon (checked)
Space (unchecked)
Merge delimiters (unchecked)

Other Options
Format quoted field as text (unchecked)
Detect special numbers (unchecked)

Fields
Column type: Standard (dropdown set to 4)
Total All device Power (highlighted with blue box, Red Box 1)
Standard (highlighted with blue box, Red Box 2)
Total CPU_device Power (highlighted with blue box, Red Box 3)
Standard (highlighted with blue box, Red Box 4)
Total GPU_device Power (highlighted with blue box, Red Box 5)
Standard (highlighted with blue box, Red Box 6)

	Total All device Power	Avg All device Power	Total CPU_device Power	Avg CPU_device Power	Total GPU_device Power	Avg GPU_device Power	Count Power
1	1963158.6	3081.868131868132	1901771.0	1572.6389324960753	32754.0	51.41915227629514	637.7
2	1963158.6	3109.4797134924524	1260588.5	1592.7897422565392	41695.0	52.58643388761678	789.5
3	2459145.0	3139.526924126481	1554123.0	1663.3894184593357	51711.66666666666664	53.248662652424255	965.666666666666666
4	3026619.66666666665	3139.526924126481	1869963.25	1611.5844564112847	62662.75	53.74286647829823	1154.75
5	3639673.0	3150.114962549548	2195335.6	1617.9165312944413	73873.4	54.1737443706085	1349.4
6	4276449.0	4616.181818181818	344549.0	3132.2636363636366	6402.0	58.2	116.0
7	507780.0	4132.166649643628	617546.0	2611.783118418225	15867.0	59.04326241134752	266.5
8	1025471.5	3996.378285803097	934038.66666666666	2378.229222669729	27147.0	59.421496974796646	453.0
9	1634873.0	3781.8952656450665	1254356.0	2245.577539798154	38448.5	59.65223615695483	639.5
10	2247836.5	3760.655203634446	1589498.6	2156.646129571986	50872.2	59.658377307886166	835.2
11	2889673.8	4737.432558139535	709391.0	3299.49323255814	9242.0	42.986646511627994	215.0
12	1018548.0	4347.11276764119	1041857.0	2878.826618778764	18583.5	46.42665897009967	387.5
13	1617176.0	4160.499548456963	1376807.66666666667	2672.566621953371	28153.333333333332	48.389127808975	559.666666666666666
14	2219349.0	4050.369510399326	1727188.0	2550.4739651442734	38807.25	49.57191601144823	737.75
15	2847453.75	4336.779963382492	1963443.0	2876.5808183841918	28386.0	36.1293911123831	725.0
16	3453197.2	3978.9648382377513	2666875.4	2471.886539577392	47519.2	50.413912924667985	988.4
17	1586123.0	4821.64255319149	110401.0	3486.7655997264436	8929.0	27.03554911569568	329.0
18	294247.0	4513.621976875857	1534643.0	3679.765299841093	18364.5	33.03344011569715	521.5
19	3635169.66666666665	4336.779963382492	1963443.0	2876.5808183841918	28386.0	57.926233992613386	38.202285805176435
20	377709.0	4234.004160447512	2391183.25	2760.0236338646745	38861.5	55.575234793651354	924.25
3	4007.605451320734	4640.75	147885.0	3098.9375	45144.0	34.6620377789362	442.2
4	222756.0	4253.24224137931	747247.5	2761.339439655172	17394.5	70.041666666666667	48.0
5	1232441.0	4132.375374800807	1283148.0	2586.6617285388834	28097.333333333332	57.926233992613386	542.3333333333334
6	3024422.25	4067.28287888996	1830736.25	39067.25	55.575234793651354	777.75	
7	4018157.6	4820.1828706496486	2421099.2	2478.666896485366	52525.2	54.65453107557305	1039.4

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
```