# .Raspberry Pi Environment Setup

# for

# Video Streaming and CV

V1.1

目錄

[環境簡述 3](#_Toc459658856)

[Pi3作業系統安裝 3](#_Toc459658857)

[Pi3環境建置 6](#_Toc459658858)

[OpenCV環境安裝 6](#_Toc459658859)

[Pi Camera Module相機驅動安裝 7](#_Toc459658860)

[最後安裝確認 8](#_Toc459658861)

[Host PC環境建置(VM) 10](#_Toc459658862)

[Eclipse整合開發環境安裝 10](#_Toc459658863)

[Cross Compiler環境設置 10](#_Toc459658864)

[OpenCV環境設置 10](#_Toc459658865)

[FFmpeg環境設置 11](#_Toc459658866)

[配置Libx264 11](#_Toc459658867)

[配置FFmpeg 11](#_Toc459658868)

[Using Remote Debugging of Eclipse 12](#_Toc459658869)

# 環境簡述

* Pi3使用Ubuntu Mate作業系統，版本為16.04 x64 for armhf
* Host PC使用VM，並設置四核心，Ram為8GB，作業系統映像檔為Ubuntu 14.04 x64 for x86 arch
* 相機模組使用官方的Camera Module

# Pi3作業系統安裝

1. 取得ubuntu mate for Pi之映像檔。
2. 利用win32DiskImager或類似軟體將該映像檔寫入SD卡中。

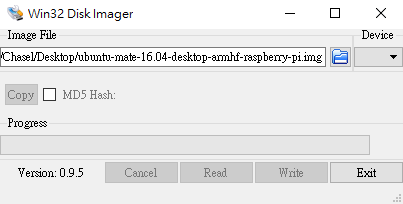


Figure 利用Win32DiskImager將寫入映像檔

註：下載位置已上傳到google雲端空間

1. Ubuntu mate

<https://drive.google.com/open?id=0B1cSyiRNLezAenpjSjhPSGxacm8>

1. Win32DiskImager

<https://drive.google.com/open?id=0B44qEXGn3V7fTDZPWDhEZ0R1TTg>

1. Pi3連接網路線、HDMI、滑鼠、鍵盤、攝影機(Camera module or Webcam)及SD卡



Figure Pi3連接圖(1)

1. 將電源插入Pi3後進行Ubuntu mate安裝。

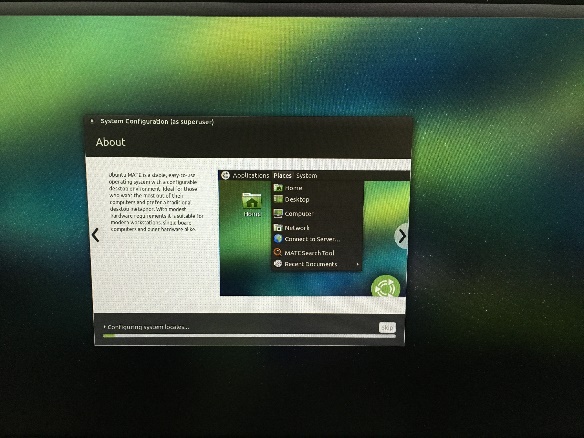
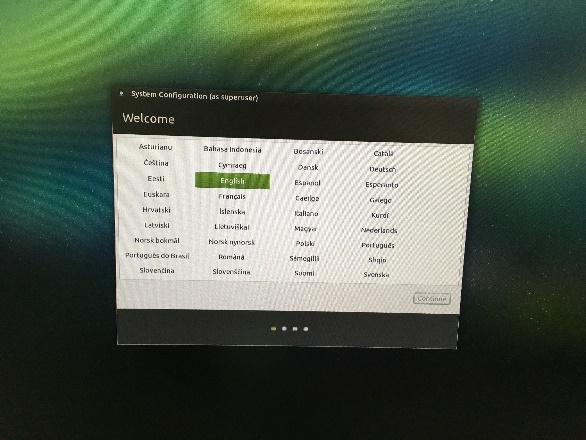


Figure 安裝過程圖

1. 安裝完成，進到Ubuntu mate桌面，這時我們先下載gparted，然後將未配置空間分配至我們所使用的磁區，首先我們先安裝gparted，如下：

$sudo apt-get install gparted

安裝完後，我們至terminal輸入：

$sudo gparted

後即可啟動gparted重新配置磁區，其配置流程圖如下幾張圖所示：

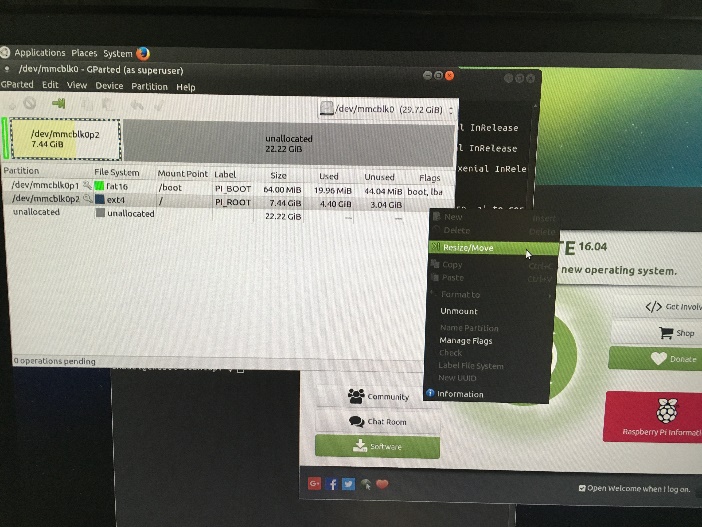


Figure Resize ext4格式之磁區

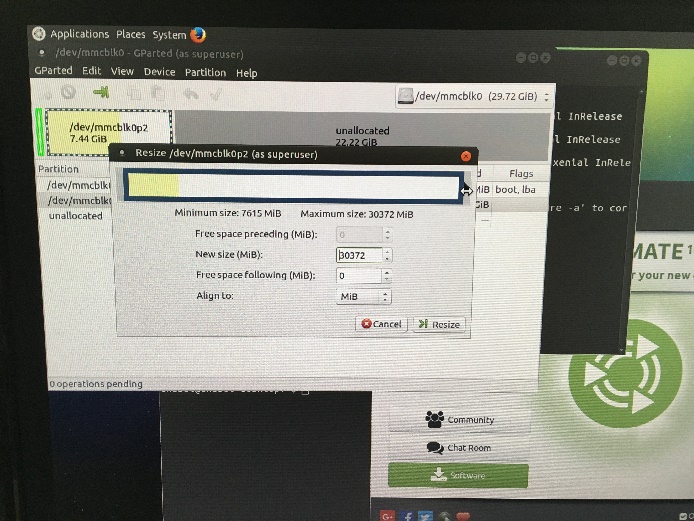


Figure 利用游標將未配置空間分配至選定的磁區中

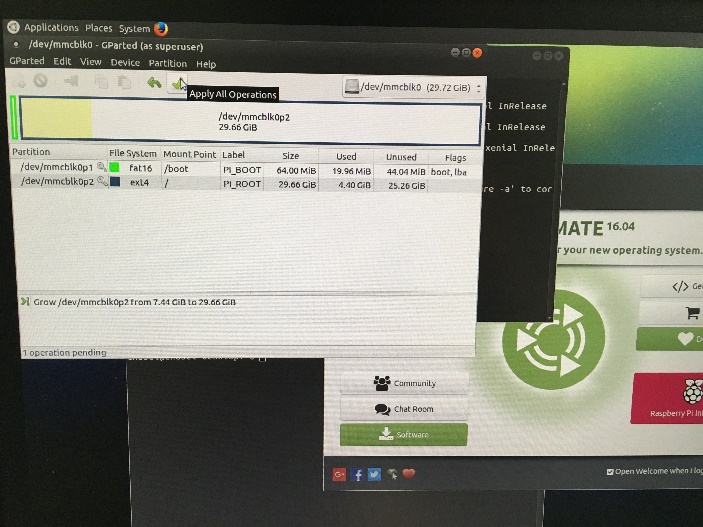


Figure 最後不要忘記按下apply

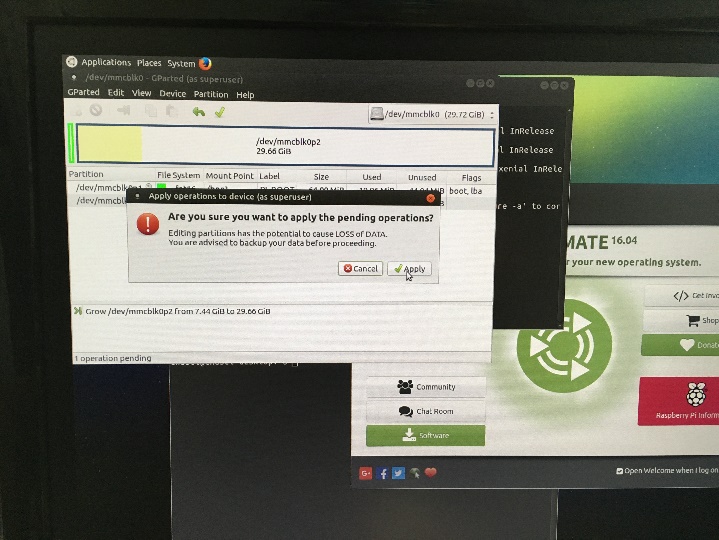
10

Figure 按下apply完成配置

1. 此時可拔掉HDMI、鍵盤及滑鼠等輸入設備(或者持續連接也可)，因為後續可直接透過Putty直接進行後續安裝操作，再關閉之前可以輸入ifconfig記一下IP，或者後續可以使用IP掃描工具來確認Pi3所在網路位置為何，方便我們後續進行SSH連線操作。

# Pi3環境建置

這一部分主要分三個階段，一為OpenCV環境安裝，二為板載攝影機驅動的安裝(如果使用Webcam則免)，最後由於在Ubuntu mate中，FFmpeg基本上應該已經安裝於作業系統中，故沒有FFmpeg的安裝流程，在進行後續流程前，我們先更新apt list及已安裝軟體的更新，鍵入指令如下：

$sudo apt-get update

$sudo apt-get upgrade

註1：如果遇到E: Unable to lock the administration directory (/var/lib/dpkg/), is another process using it?的問題，可以做以下動作(不見得會遇到)。

1. 鍵入指令sudo rm /var/lib/dpkg/lock後並重開機。
2. 接下來在輸入一次sudo apt-get update後可能會遇到dpkg was interrupted, you must manually run ‘sudo dpkg –configure –a’ to correct the problem，這時按照提示輸入指令即可。
3. 接下來在執行一次sudo apt-get update應該就無此問題了。
4. 如果還有第二點的問題，請重新開機後再輸入一次‘sudo dpkg –configure –a’後應該就沒問題了。

## OpenCV環境安裝

首先我們需要安裝必要環境，鍵入以下指令：

$sudo apt-get install build-essential cmake git pkg-config

$sudo apt-get install libtiff5-dev libjasper-dev libpng12-dev

$sudo apt-get install libv4l-dev

$sudo apt-get install libgtk2.0-dev

$sudo apt-get install libatlas-base-dev gfortran

接下來，我們從github下載OpenCV原始碼，鍵入指令如下(存放路徑請自行選擇)：

$git clone <https://github.com/Itseez/opencv.git>

接下來我們移動到下載回來的opencv資料夾內，然後開始進行configure->make->install三個流程

$cd opencv

$mkdir build

$cd build

$cmake --disable-libjpeg -D CMAKE\_INSTALL\_PREFIX=/usr/local -D WITH\_OPENGL=ON -D WITH\_V4L=ON -D WITH\_TBB=ON ..

$make –j $(nproc)

接下來會開始進行compile，時間稍長，請耐心等候。

註：使用多核心編譯可能會在途中遭遇記憶體用盡的情形，關鍵錯誤訊息為g++: internal compiler error: Killed (program cc1plus)，當看到編譯出錯，把terminal稍微往上拉至error處，可能可以發現此關鍵句，記憶體用盡有幾種解法(我自己的解法，請參考)：

1. 降低速度換穩定性，適合按下make就不見蹤影一段時間的人用，請在一開始make的時候降低核心數，以Pi3來說，可以選擇make或make –j 2指令來進行compile，或者心臟大點可以make –j 3。
2. 直接使用make –j $(nproc)，等到出錯時，再重開Pi3，進去build資料夾使用make –j 3繼續compile。

然後我們開始進行安裝，鍵入指令如下：

$sudo make install

$sudo ldconfig

## Pi Camera Module相機驅動安裝

首先，我們先更新Pi3的韌體，鍵入指令如下：

$sudo rpi-update

接下來，我們需要讓Pi camera module能夠被識別，而原本的手段是要利用raspi-config來打開相機模組，但目前除了raspbian OS外，其他系統暫時無法使用這種方式來打開相機模組，但據說日後Ubuntu mate更新後會加入這項功能。

我們利用vi來修正/boot/config.txt，修正過程如下：

$cd /boot

$sudo vi config.txt

然後我們在任一處加上以下三行(不要加在註解區)。

start\_x=1

gpu\_mem=128

disable\_camera\_led=1

其完成結果如下圖所示，之後回到vi的一般模式鍵入:!wq即可，然後我們重新啟動Pi3，來完成所有安裝程序。

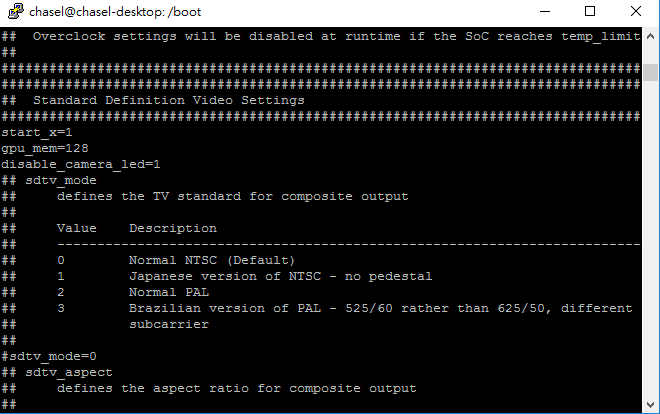


Figure 完成修改後的config.txt

## 最後安裝確認

* OpenCV部分請鍵入以下指令來確認安裝(結果如下圖)：

$pkg-config --libs opencv

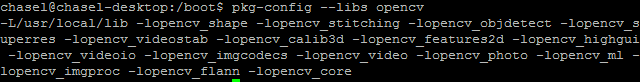


Figure OpenCV安裝完成

* FFmpeg請鍵入以下指令來確認安裝：

$ffmpeg

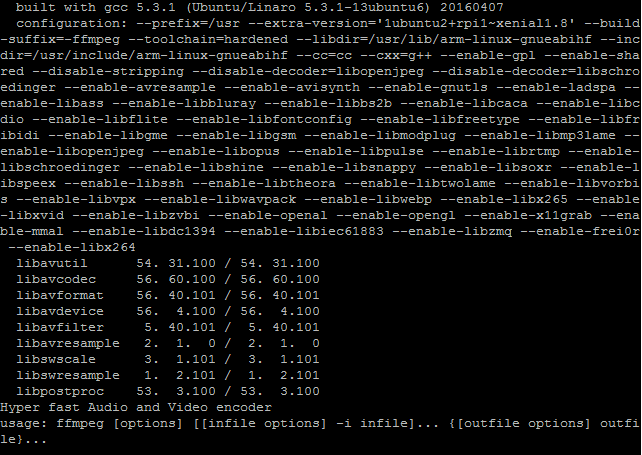


Figure FFmpeg安裝完成

* 相機模組請鍵入以下指令來測試：

$sudo raspistill -o test.jpg

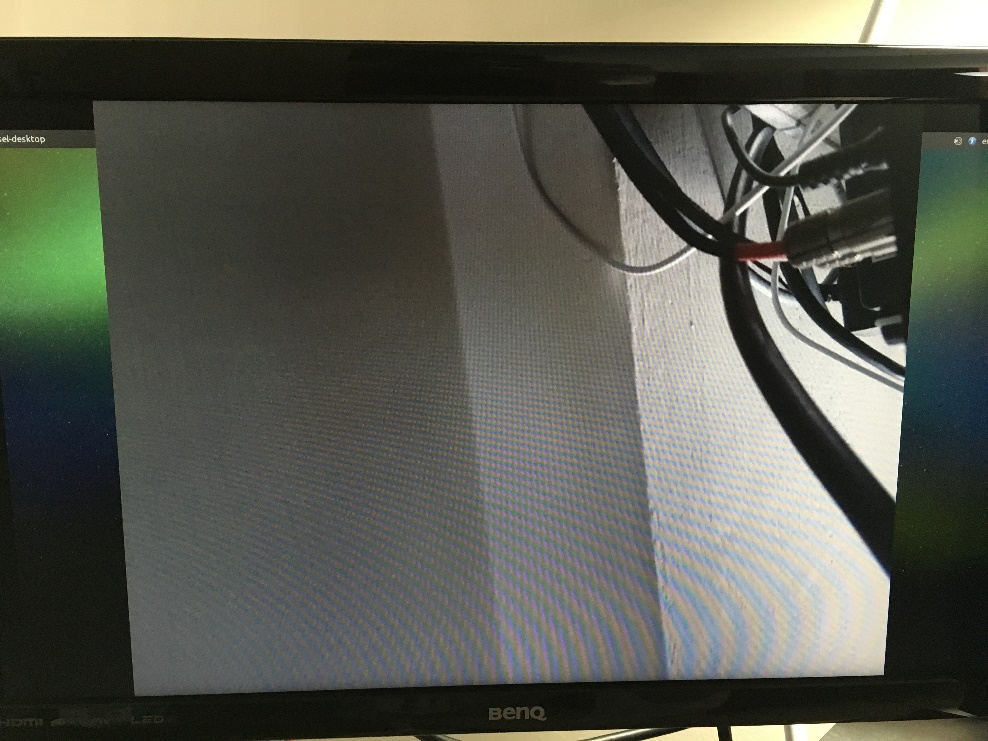


Figure 相機會打開預覽一小段時間後關閉，以上畫面為Pi3所顯示的畫面

註：請於每次開機時執行script，或手動鍵入以下指令以確保OpenCV API能抓到Pi Camera Module：

$sudo modprobe bcm2835-v4l2

# Host PC環境建置(VM)

首先，我們一樣先更新我們現有的套件以及更新軟體資料庫：

$sudo apt-get update

$sudo apt-get upgrade

## Eclipse整合開發環境安裝

接下來，我們利用最簡單的方式裝JDK以利我們安裝Eclipse來進行專案開發，指令如下：

$sudo apt-get install default-jdk

接下來我們從Eclipse官網下載Eclipse C/C++ IDE的壓縮檔，我們利用wget指令來進行下載：

$wget <http://mirrors.neusoft.edu.cn/eclipse/technology/epp/downloads/release/mars/2/eclipse-cpp-mars-2-linux-gtk-x86_64.tar.gz>

接下來解壓縮，實際檔名請自行更動。

$tar zxvf eclipse-cpp-mars-2-linux-gtk-x86\_64.tar.gz

至此，Eclipse for C/C++已安裝完成。

## Cross Compiler環境設置

接下來我們要為Embedded System構建Cross Compile環境，一樣從APT-Server下載套件來進行安裝，指令如下：

$sudo apt-get install libncurses5-dev

$sudo apt-get install gcc-arm-linux-gnueabihf

$sudo apt-get install g++-arm-linux-gnueabihf

$sudo apt-get install cmake git libgtk2.0-dev pkg-config

## OpenCV環境設置

首先我們先從github下載OpenCV source code，鍵入指令如下：

$git clone <https://github.com/Itseez/opencv.git>

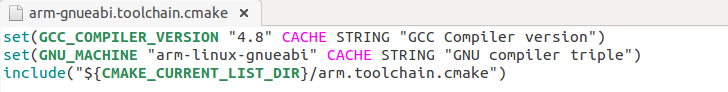
下載完畢後，我們移動至以下路徑：

$cd opencv/platforms/linux

我們可以在裡面發現裡面有一個arm-gnueabi.toolchain.cmake檔案，我們修改compiler的版本為4.8，實際版本可以使用下列指令得到：

$arm-linux-gnueabihf-g++ --version

而arm-gnueabi.toolchain.cmake修改完成後的結果如下圖所示：



接下來我們創建一個存放編譯結果的資料夾，名為build，然後開始configure，流程如下(注意，路徑還在opencv/platforms/linux底下)：

$mkdir build

$cd build

$cmake --disable-libjpeg -D CMAKE\_INSTALL\_PREFIX=/usr/local -DCMAKE\_TOOLCHAIN\_FILE=../arm-gnueabi.toolchain.cmake ../../..

待configure結束後，我們開始編譯並安裝，流程如下：

$make -j $(nproc)

$sudo make install

$sudo ldconfig

## FFmpeg環境設置

在這一部分，我們要建置包含H.264的FFmpeg library，故我們的建置流程分為兩部分，一為libx264的編譯，二為FFmpeg的編譯。

### 配置Libx264

首先，我們先下載x264的source code並移動至該資料夾內。

$git clone git://git.videolan.org/x264

$cd x264

再來，我們必須使用我們的cross compiler來編譯libx264，故鍵入以下configure來進行配置。

$./configure --host=arm-linux --enable-static --cross-prefix=arm-linux-gnueabihf- --prefix=/usr/local

然後我們開始進行編譯並安裝。

$make -j $(nproc)

$sudo make install

$sudo ldconfig

### 配置FFmpeg

首先，我們必須下載FFmpeg source code。

$git clone git://source.ffmpeg.org/ffmpeg.git

再來移動至ffmpeg資料夾裡面並進行配置、編譯及安裝，請鍵入如下指令：

$cd ffmpeg

$./configure --enable-cross-compile --cross-prefix=arm-linux-gnueabihf- --arch=armel --target-os=linux --prefix=/usr/local --enable-gpl --enable-libx264 --enable-nonfree --extra-cflags="-I/usr/local/include" --extra-ldflags="-L/usr/local/lib" --extra-libs=-ldl

$make –j $(nproc)

$sudo make install

$sudo ldconfig

# Using Remote Debugging of Eclipse

這裡使用Remote Debugging Tools有兩個目的，一為上傳檔案到板子，二為執行並從Eclipse的Console觀看文字輸出結果，設定前，我們先確保Host PC及板子都已經連線到同一台AP(有線無線網路皆可)。

首先，我們先與板子建立連線，步驟如下：

* 1. 點選Open Perspective，並新增Remote System Explorer

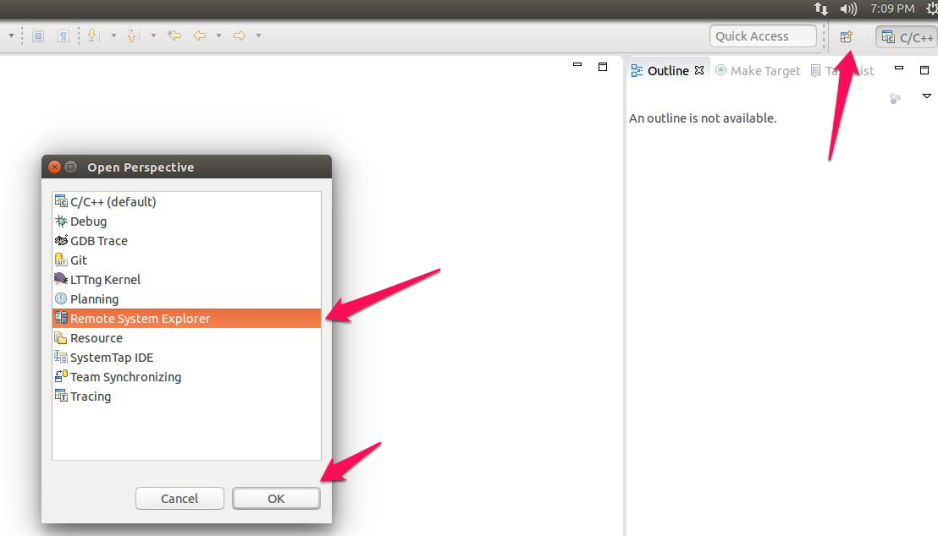


Figure Open Perspective

1. 移動到Remote System Explorer的Perspective後，我們開始建立新的連線，如下圖所示：

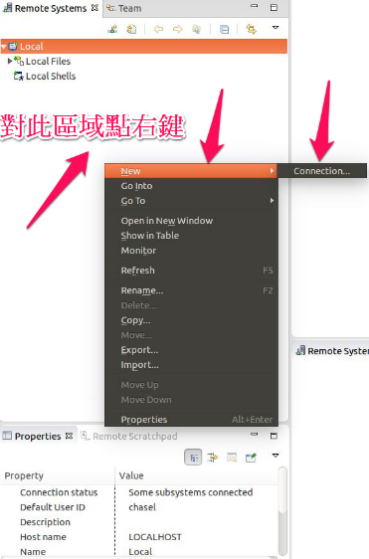


Figure 建立新連線

1. 選擇目標(Pi3)為Linux作業系統。

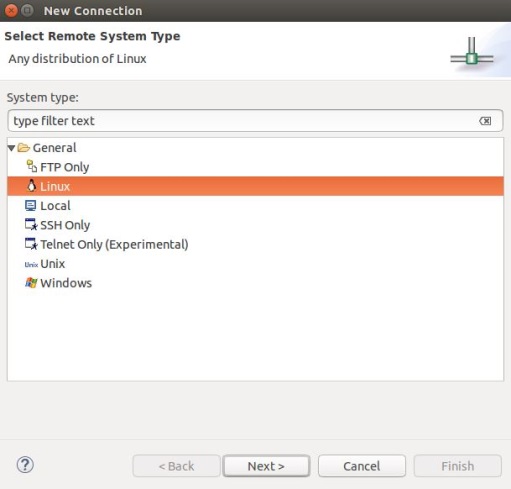


Figure 選擇目標作業系統

1. 輸入目標IP(Pi3)及連線名稱，IP可至Pi3上輸入ifconfig指令後取得，請注意，完成後不要按Finish，請按Next進行下一步。

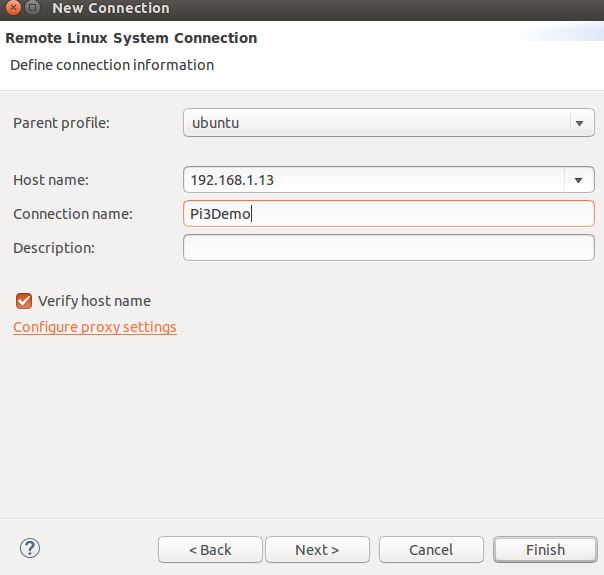


Figure 輸入IP及Connection Name

1. 接下來請按照下圖完成最後設定SSH連線設定，完成後請按Finish。

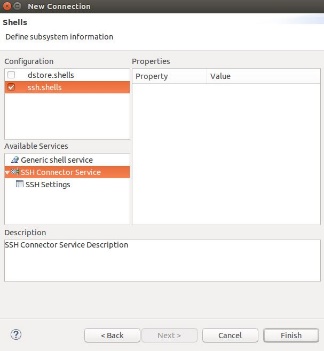
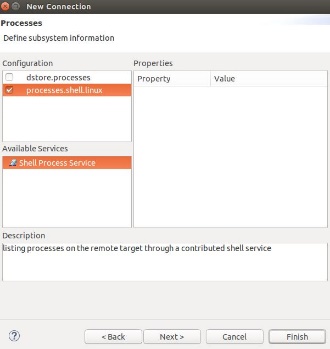
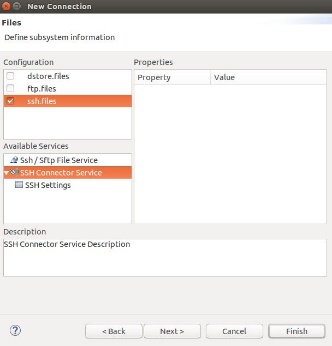


Figure SSH連線設定

1. 接下來我們就可以在Remote System Explorer發現我們剛剛設置的連線，現在，我們開始進行SSH連線。

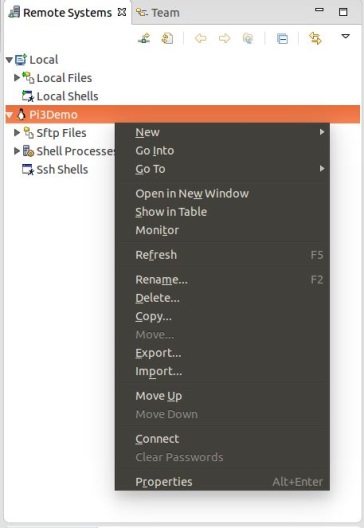


Figure 對連線按右鍵並按下Connect

1. 接下來，我們輸入Pi3上的帳戶跟密碼，按下OK進行連線，我們會發現會出現一堆通知框，請一律點選Yes或OK等正面回應。

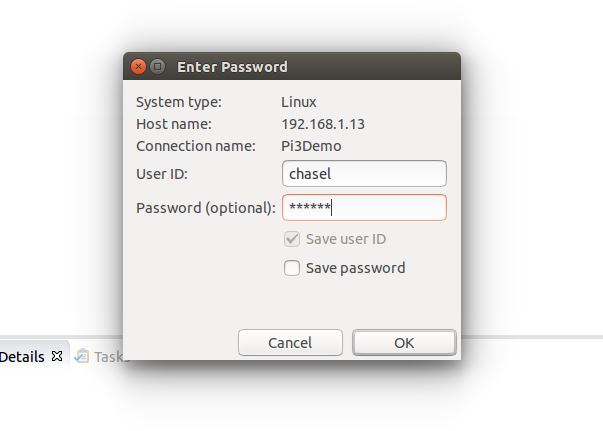


Figure 輸入帳號密碼

1. 最後，我們可以看剛剛建立的連線出現了綠色小箭頭，表示連線已經建立成功，如下圖所示：

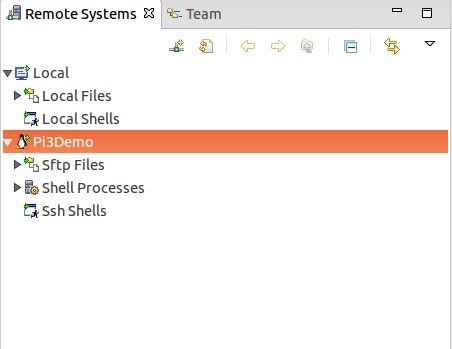


Figure 連線建立成功