『物聯網概論』實驗手冊

實驗模組名稱: 智慧電燈-紅外線感應自動照明實作

開發教師: 張志勇教授 (cychang@mail.tku.edu.tw)

開發學生: 王德仁 (deren921@gmail.com)

鄧振亞 (c22424193@gmail.com)

學校系所: 淡江大學資訊工程學系

實驗平台: Octopus II

實驗主軸: 紅外線感應自動照明實作

實驗內容關鍵字: Octopus II ,Tiny OS , Passive Infrared Motion Sensor,Relay

本實驗手冊乃『教育部網路通訊人才培育先導型計畫-課程發展計畫』支持所發展之實驗手冊

實驗目的

本實驗整合 Octopus II 嵌入式感測器與 Passive Infraresd Motion Sensor 紅外線物體移動感測器。利用 Passive Infraresd Motion Sensor 紅外線物體移動感測器偵測是否有人移動，當有人移動時，Passive Infraresd Motion Sensor 紅外線物體移動感測器將觸發 Octopus II，而 Octopus II 中的韌體將對 Relay 繼電器下達開關變換之命令，使得控制日光燈明滅。這個實驗可以讓學生練習 Octopus II 的輸入與輸出腳位之電路連結，利用 Passive Infraresd Motion Sensor 紅外線物體移動感測器當作 Octopus II 的輸入，而利用 Octopus II 連接的 Relay 繼電器當作 Octopus II 的輸出，透過本實驗，同學不但可以了解紅外線物體移動感測器之原理與應用，更可以熟悉 Octopus II 與其他模組透過輸出入腳位加以整合。

實驗器材

- Octopus II 無線感測節點
  - 中央處理器使用 TI 公司的 MSP430 處理器
  - 超低功耗架構，可延長電池使用時間
  - 儲存單元使用 ST 公司的 MP25P80 快閃記憶體，容量為 1MB
  - 具有 50 個擴充腳位，可整合其他感測器與模組
- Passive Infraresd Motion Sensor 紅外線物體移動感測器
  - 工作電壓為3.3和5V
  - 可以偵測區域內的物體移動
  - 3-Pin接腳(電源輸入、接地、輸出)
  - 感測範圍角度為 45 度

實驗所需軟體

- PC
  - Windows XP
  - Java 1.5 (Java 5) JDK 以上
  - TinyOS
  - Cygwin
  - MSP430 tool chain
  - nesC 編譯器
  - OctopusII 驅動程式
  - OctopusII 燒錄程式
  - OctopusII 編譯文件

Table of Contents

Part I 簡介 Octopus II 和 Passive Infraresd Motion Sensor、Relay 3
Step1 瞭解 Octopus II 3
Step2 瞭解 Passive Infraresd Motion Sensor 4
Step3 瞭解 Relay 6

Part II 環境建置 7
Step1 介紹實驗所需之軟體 7
Step2 安裝 JAVA 7
Step3 安裝 Cygwin1.7 (線上安裝) 13
Step4 安裝 TinyOS 2.1 23

Part III 硬體電路整合 30
Step1 選擇支援 GPIO 的腳位 30
Step2 整合 Passive Infrared Motion Sensor 至 Octopus II 31

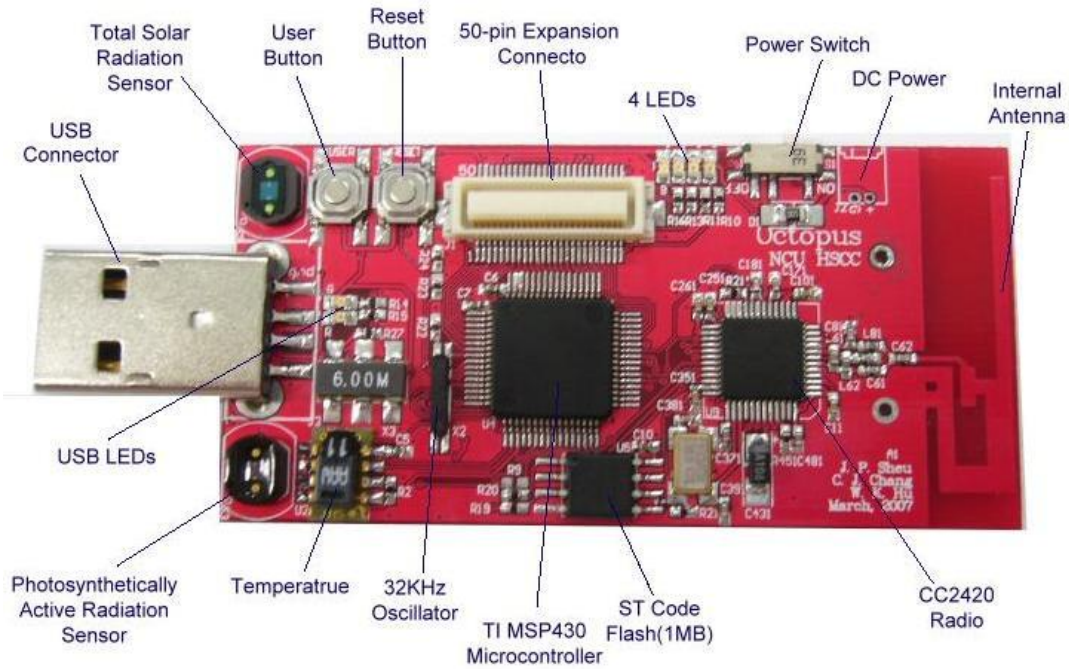
Part IV 設計與製作紅外線感應自動照明 32
Step1 編譯 Octopus II 程式 32
Step2 燒錄 Octopus II 程式 33
Step3 測試與展示系統 34

Part V –附錄程式碼 36

Part I 簡介 Octopus II 和 Passive Infrared Motion Sensor

Step 1 瞭解 Octopus II

Octopus 是由清華大學和中央大學團隊所共同研發的無線感測器，目前已開發出三個版本：Octopus I、Octopus II 以及 Octopus X，三個版本所使用的無線通訊模組皆為 CC2420 RF 晶片模組，第一版的 Octopus I 參照 Crossbow 的 MICAz 感測節點為其主要開發架構，並搭載 8-bit AVR MCU 為中央處理單元，具有 4 KB 的記憶體空間以及 128 KB 的快閃記憶體容量；第二版的 Octopus II 則參照 MoteIV 的 Tmote Sky 感測節點為其主要開發架構，搭載 16-bit MSP430 MCU 為中央處理單元，具有 10 KB 的記憶體空間以及 1MB 的快閃記憶體容量，相較於前一代的 Octopus I 具有更強的運算能力與更多的資料儲存空間；第三版的 Octopus X 全部架構皆為自行設計，其主要訴求為使感測節點體積變得更為輕巧，搭載了 CC2430 整合型晶片(整合了 8051 MCU 與 CC2420 RF)，使 Octopus X 感測節點體積縮小為 28mm×28mm×7mm，約為 Octopus II 體積(65mm×31mm×15mm, Octopus II－A)之 1/4 大小。



本範例實驗採用 Octopus II 感測節點，如上圖所示，其中央處理器晶片與儲存單元分別採用了 16 位元的 MSP430 RISC 處理器以及具有 1MByte 容量的 MP25P80 快閃記憶體。Octopus II 系列可分為 Octopus II－A 和 Octopus II－B，其尺寸分別為 65mm×31mm (Octopus II－A)及 80mm×31mm (Octopus II－B)，兩者皆採用雙天線設計，Octopus II－A 的傳輸距離可以達到 200 公尺，Octopus II－B 比 Octopus II－A 多了 RF 訊號放大處理，因此 Octopus II－B 的傳輸距離可

藉此增加到 450 公尺。這兩款 Octopus II 系列在訊號強度和傳輸距離上，均比市面上同類型無線感測器平台有更佳的線性化。

在感測器的部分，Octopus II 也採取跟 Octopus X 系列相同的分離式模組設計，採用 MICAz 的 PCB 線路設計，設置了 50 個擴充腳位，讓使用者可以靈活的運用以符合各種需求，藉此增加無線感測器平台的擴充性，而其感測器擴充板包含了四種基本的感測器，分別為溫溼度感測器、光感測器、陀螺儀和三軸加速度計。

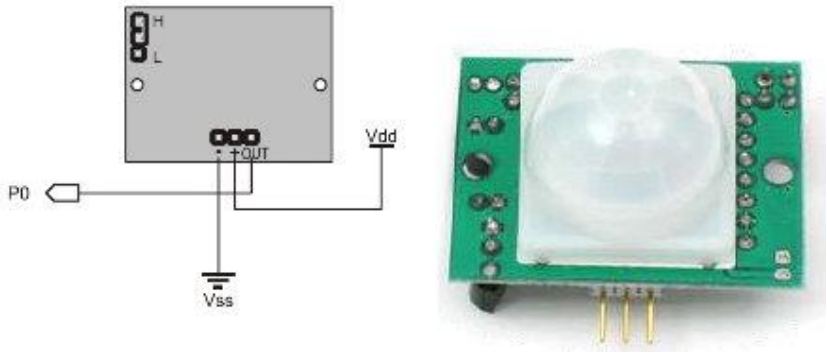
在電力管理的部分，Octopus II 的電力來源主要是兩顆 3 號電池，而 MSP430 處理器提供了 5 種省電模式，因此，使用者可藉由進入省電模式延長無線感測節點的使用時間，而其中一種省電模式更可以藉由關掉所有震盪器功能，讓 MSP430 達到極低的 0.2 μA 耗電量。

與 PC 端的橋接的部分，Octopus II 使用了 FTDI 的 FT232BL 為其 USB 與 UART 橋接晶片，並與處理單元 MSP430 F1611 MCU 銜接，只須將 Octopus II 的 USB 端直接插入 PC，再配合相關應用程式，便可進行程式燒錄及除錯。

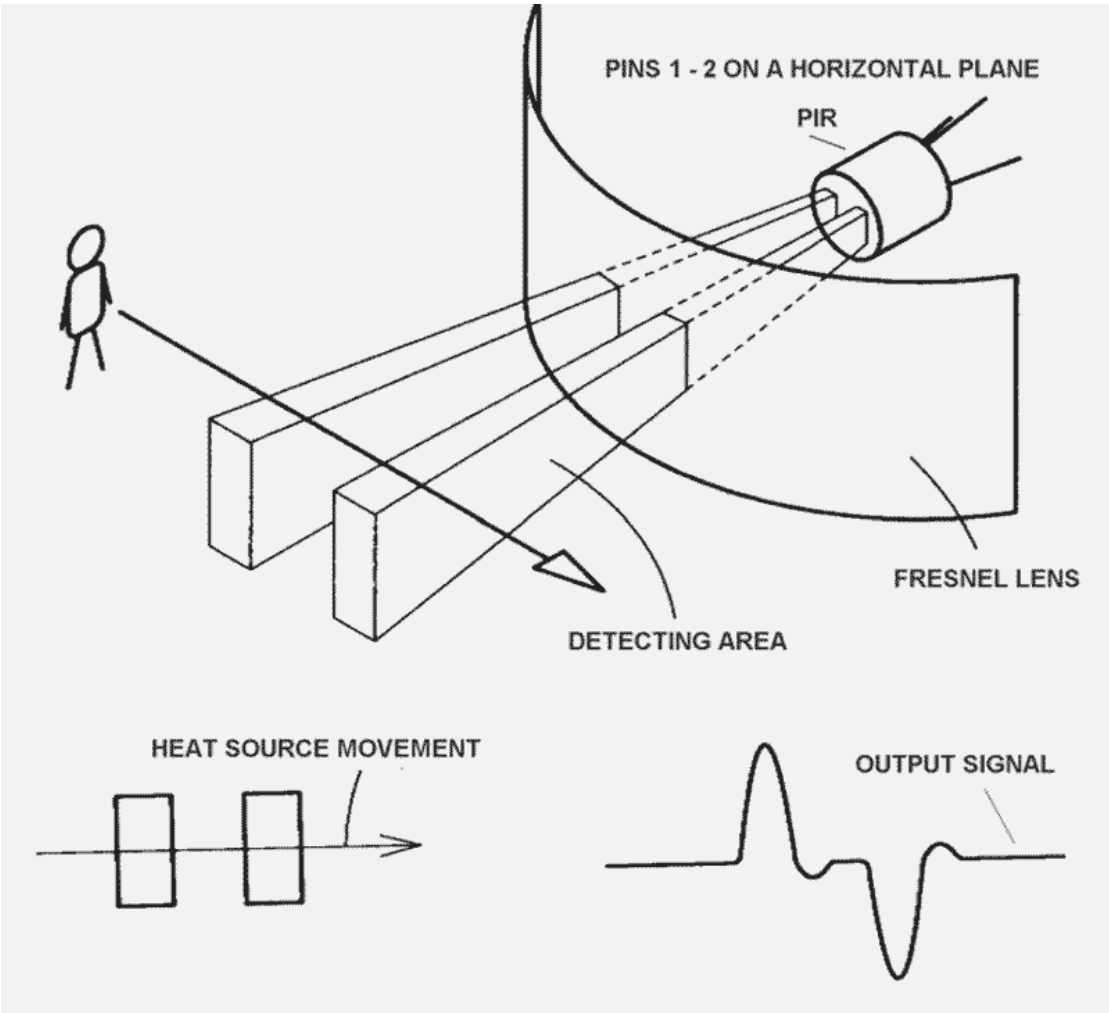
Step2 瞭解 Passive Infrared Motion Sensor

Passive Infrared Motion Sensor 也稱 PIR(Passive Infra-Red)，中文名稱為被動式紅外線感測器，被動式感測器本身不會發射紅外線，而是透過物體本身的熱源移動觸發感測器。

其主要功能在於利用物體發出的熱源來檢測物體是否有移動，並且利用 OUT 腳位的 High/Low 的波形來表示是否有物體的移動，此外，其感測具有方向性，感測角度為 45 度。



PIR 已被廣泛使用於室內物體移動的感測，例如廁所，當經過一段時間都沒有人進出廁所時，日光燈將自動關閉，以達到省電之目的。



Passive Infrared Motion Sensor 的工作原理如上圖所示，它會將感應到的紅外線光譜轉為訊號，由於不同溫度的物體發出的紅外線光譜也不一樣，以人體為例，體溫平均為 37 度左右，因此發出來的光譜波長大約為 10um[1]。

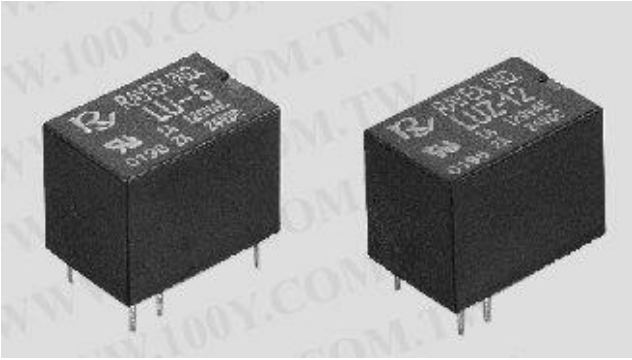
當發熱物體在 PIR 前面其移動時，PIR 會將結果反應在 OUT 腳位，我們可利用 PIR 感應器 OUT 腳位所送出的 high/low 電位判斷是否有物體移動。

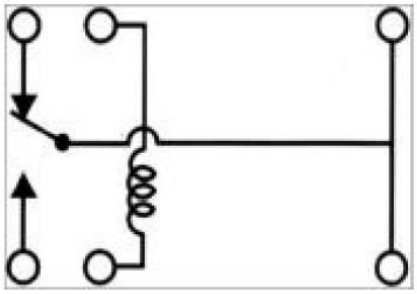
以下為 Passive Infrared Motion Senso 之規格：

- 單一位元輸出(不是 1 就是 0)
- 可偵測室內、區域
- 體積小容易隱藏
- 3-PIN SIP 接腳分別為 OUT、Vcc、GND
- 可相容 3.3V MCU 或其他裝置
- 工作電壓為 3.3 和 5 V DC，電流需求為 100uA

Step3 瞭解 Relay

Relay 繼電器其主要功能為利用電磁感應的的方式控制開關，它具有控制系統和被控制系統，通常應用於自動控制電路中。





本篇中所採用的繼電器為直流繼電器，其型號為 LU-3 需要 3V 以上才能驅動開關，其結構為鐵芯、線圈、銜鐵、觸點簧片等組成，其工作原理只需要在線圈兩端施加一定的電壓與電流，使其產生電磁感應，鐵芯將受到電磁感應的吸引，當電磁感應的吸引力大於簧片的拉力時，從而帶動銜鐵的動觸點與靜觸點吸合，當施加的電壓與電流消失時，鐵芯將會返回原來的位置。

Part II 環境建置

Step1 介紹實驗所需之軟體

- Java SE Development Kit
  提供 TinyOS 的執行引擎，並且提供了 JAVA 用戶端程式用以監控感測器目前所感測的資料。

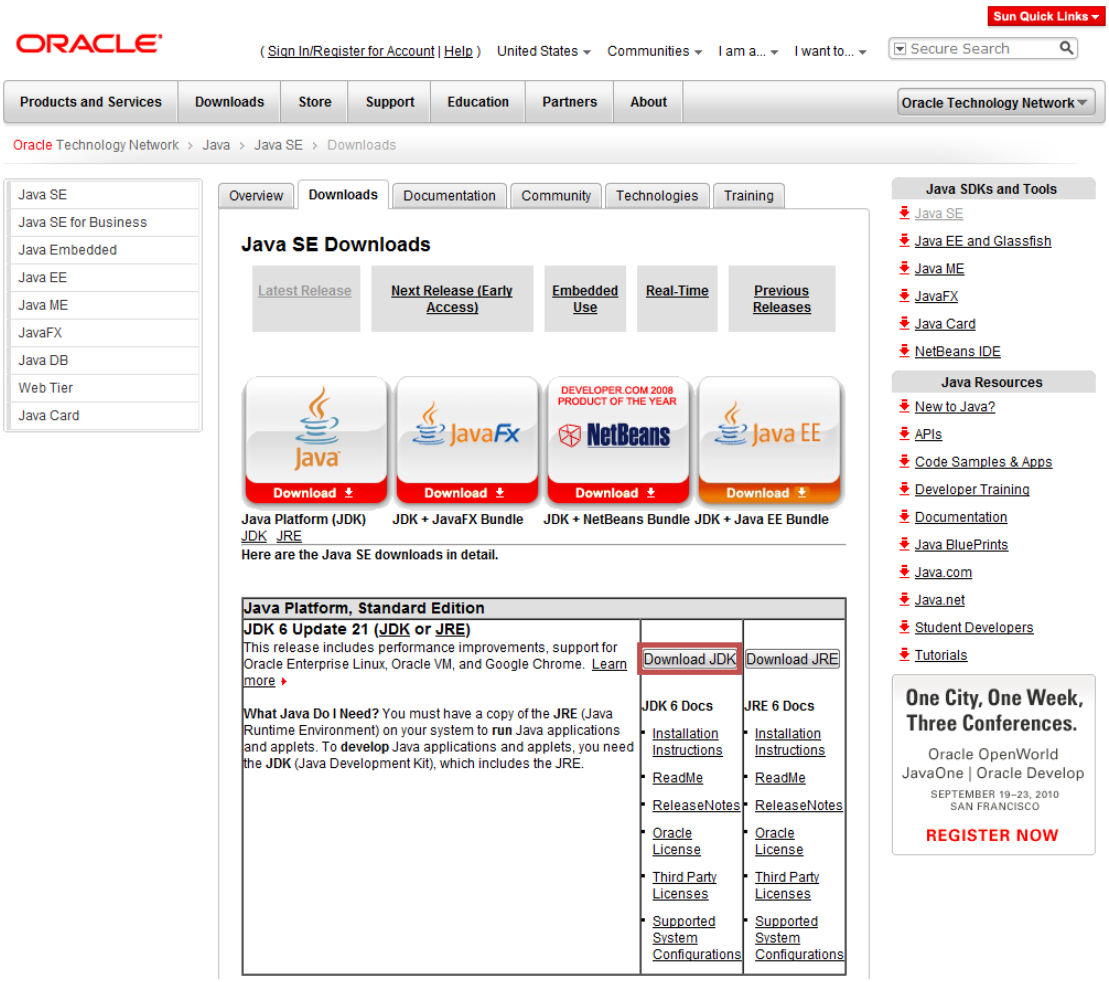
- Cygwin (with TinyOS)
  提供在 Windows 下模擬 UNIX 的 shell 環境以及開發 TinyOS 時需使用的 UNIX 工具(如 perl 和 shell 腳本)。在此模擬環境下，程式開發者將可直接使用 make 指令編譯 TinyOS 程式碼，並可在模擬環境下監控感測器所傳回之資料。

- Octopus II Programmer
  這是一套由清華大學和中央大學團隊共同開發之 Octopus II 燒錄程式，能將編譯完成後所產生的 TinyOS ihex 檔案直接利用 USB 燒錄到 Octopus II。

Step2 安裝 JAVA

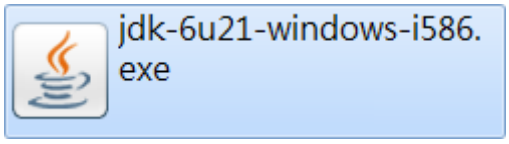
由於 TinyOS2.x 需要有 JAVA VM 環境的支援，因此我們將先介紹 JAVA JDK 之安裝。

- JAVA JDK 需使用 1.5 以上的版本
- 下載連結：
  http://www.oracle.com/technetwork/java/javase/downloads/index.html

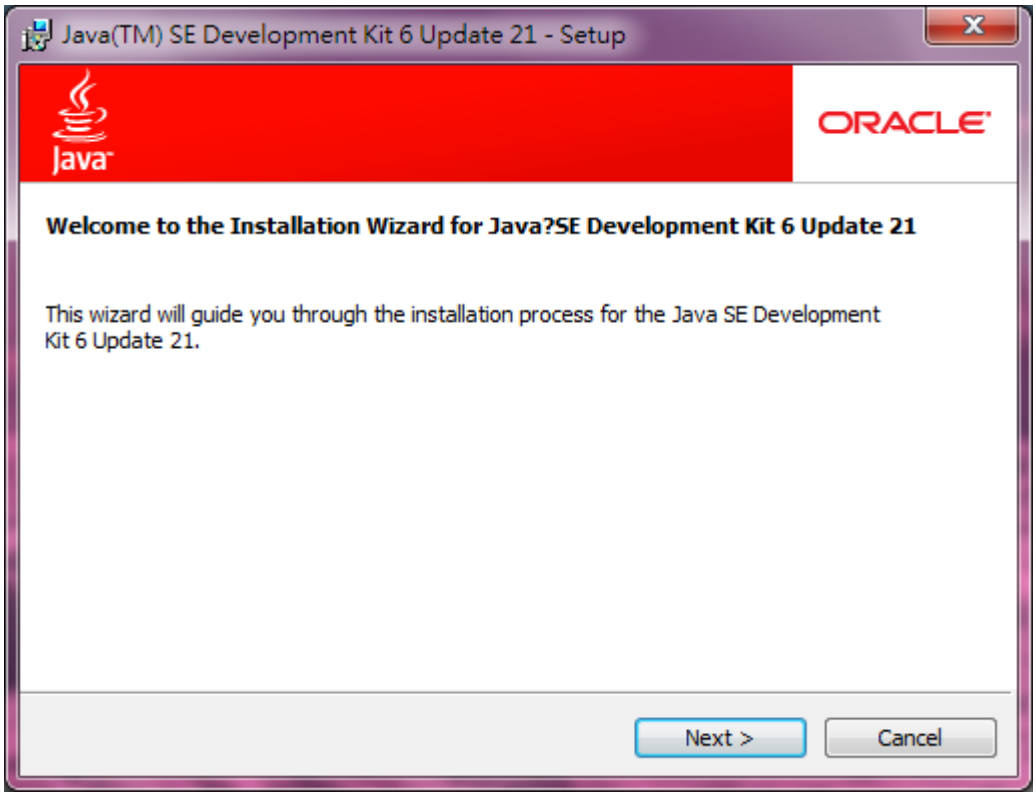


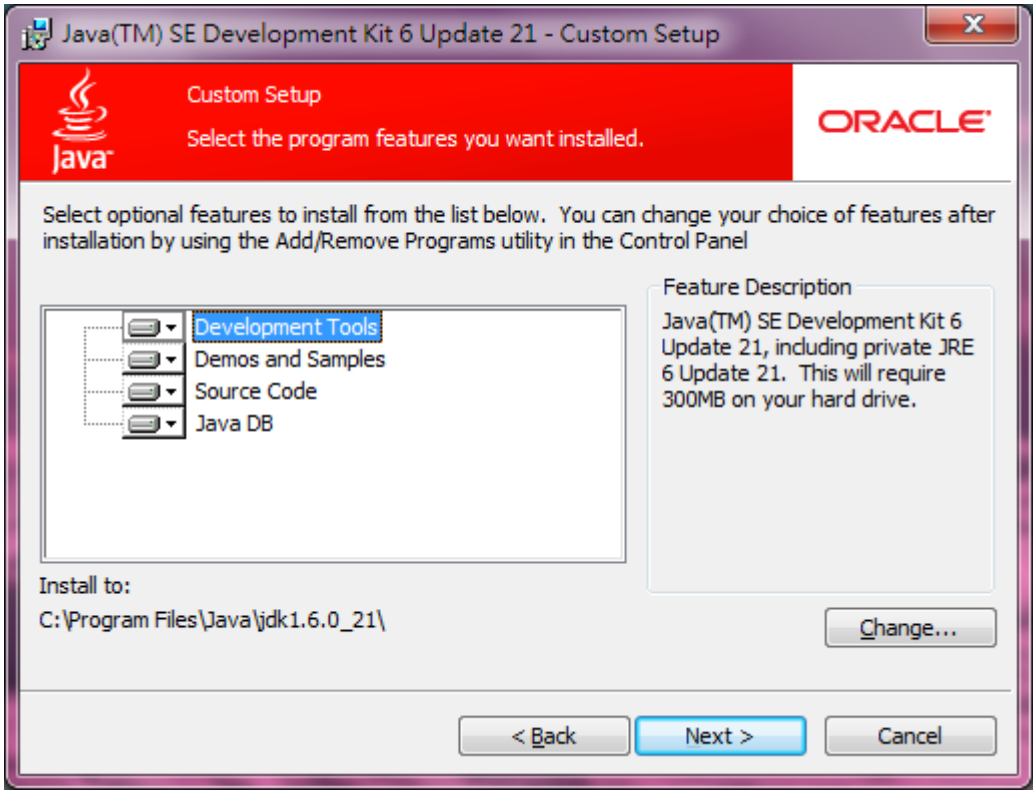
安裝 Java SDK

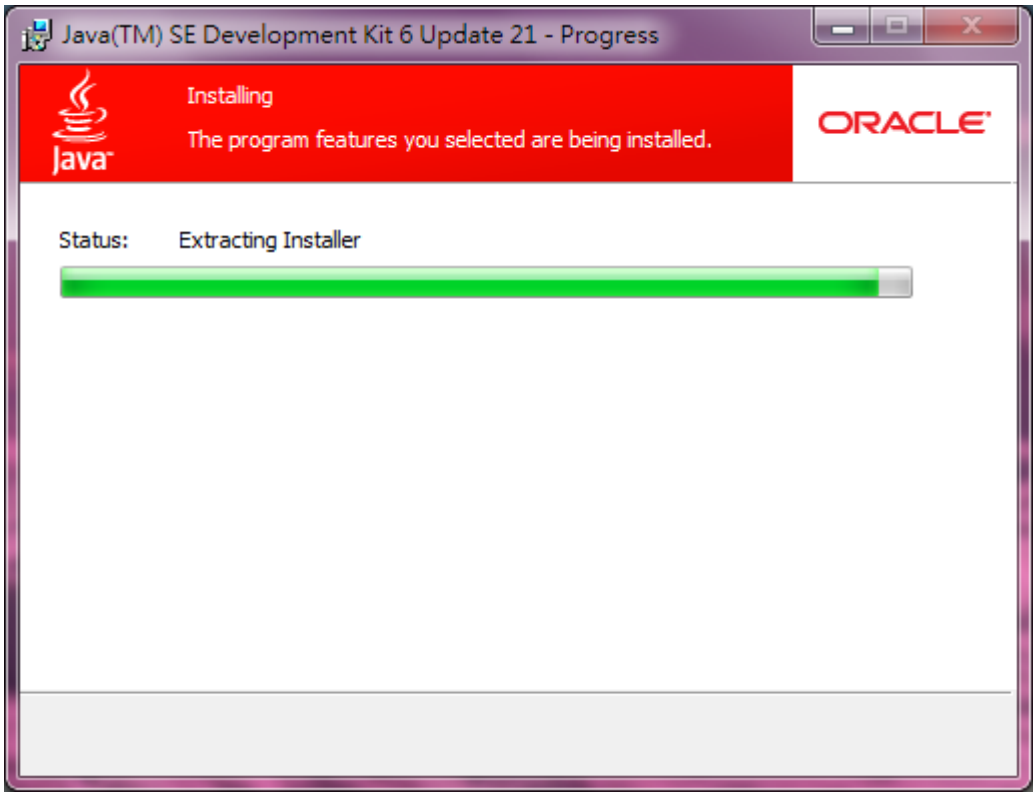
先下載安裝檔並執行：

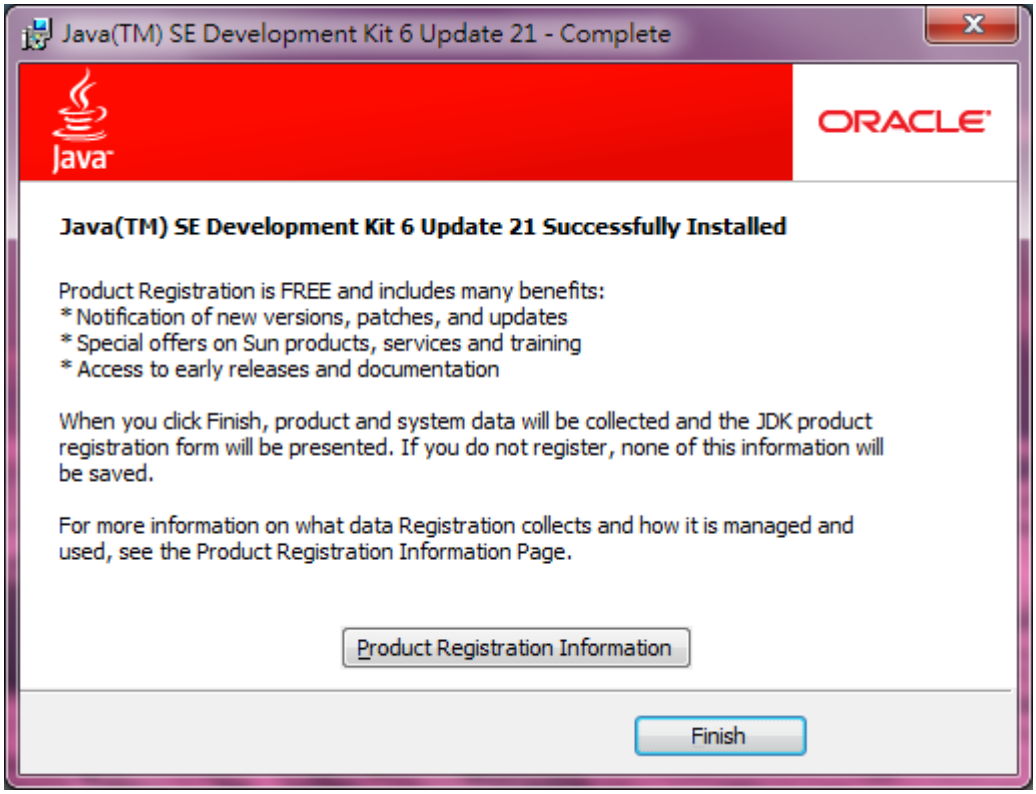


開啟後選擇 Next，順著安裝精靈的指示下一步直到完成安裝。


Java(TM) SE Development Kit 6 Update 21 - Setup
Java
ORACLE
Welcome to the Installation Wizard for Java?SE Development Kit 6 Update 21
This wizard will guide you through the installation process for the Java SE Development Kit 6 Update 21.
Next >
Cancel

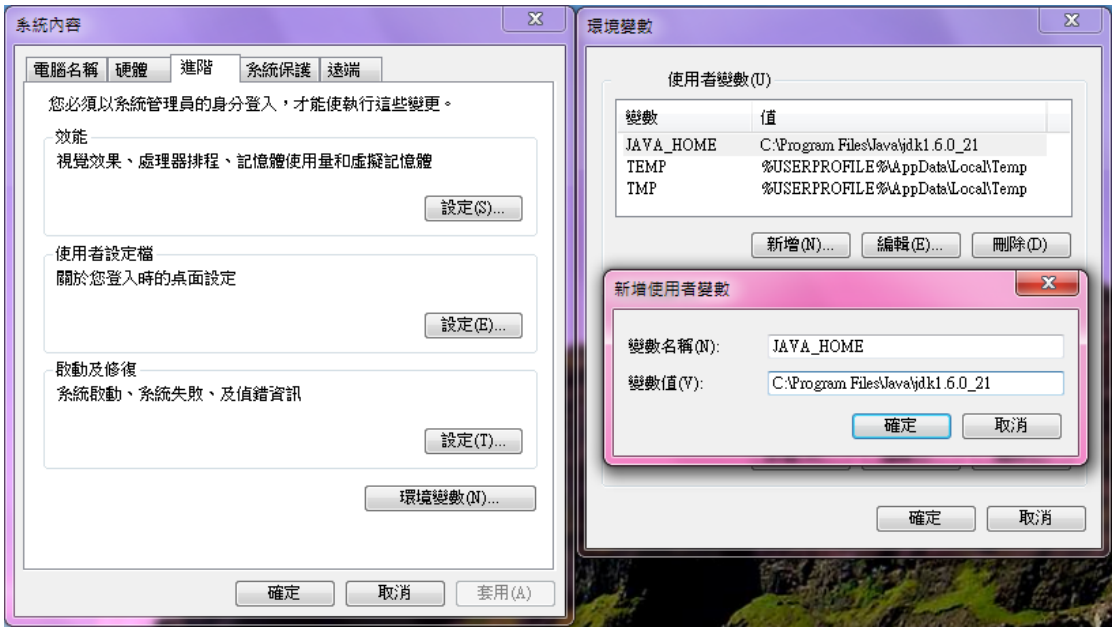

Java(TM) SE Development Kit 6 Update 21 - Custom Setup
Java
Custom Setup
Select the program features you want installed.
ORACLE
Select optional features to install from the list below. You can change your choice of features after installation by using the Add/Remove Programs utility in the Control Panel
Development Tools
Demos and Samples
Source Code
Java DB
Feature Description
Java(TM) SE Development Kit 6 Update 21, including private JRE 6 Update 21. This will require 300MB on your hard drive.
Install to:
C:\Program Files\Java\jdk1.6.0_21\
Change...
< Back
Next >
Cancel


Java(TM) SE Development Kit 6 Update 21 - Progress
Installing
The program features you selected are being installed.
ORACLE
Status:
Extracting Installer


Java(TM) SE Development Kit 6 Update 21 - Complete
ORACLE
Java(TM) SE Development Kit 6 Update 21 Successfully Installed
Product Registration is FREE and includes many benefits:
* Notification of new versions, patches, and updates
* Special offers on Sun products, services and training
* Access to early releases and documentation
When you click Finish, product and system data will be collected and the JDK product registration form will be presented. If you do not register, none of this information will be saved.
For more information on what data Registration collects and how it is managed and used, see the Product Registration Information Page.
Product Registration Information
Finish

設定環境變數

先在「**我的電腦**」上面按右鍵選「**內容**」→「**進階**」→「**環境變數**」，在下面的變數值貼上安裝 Java SDK 的所在位置。



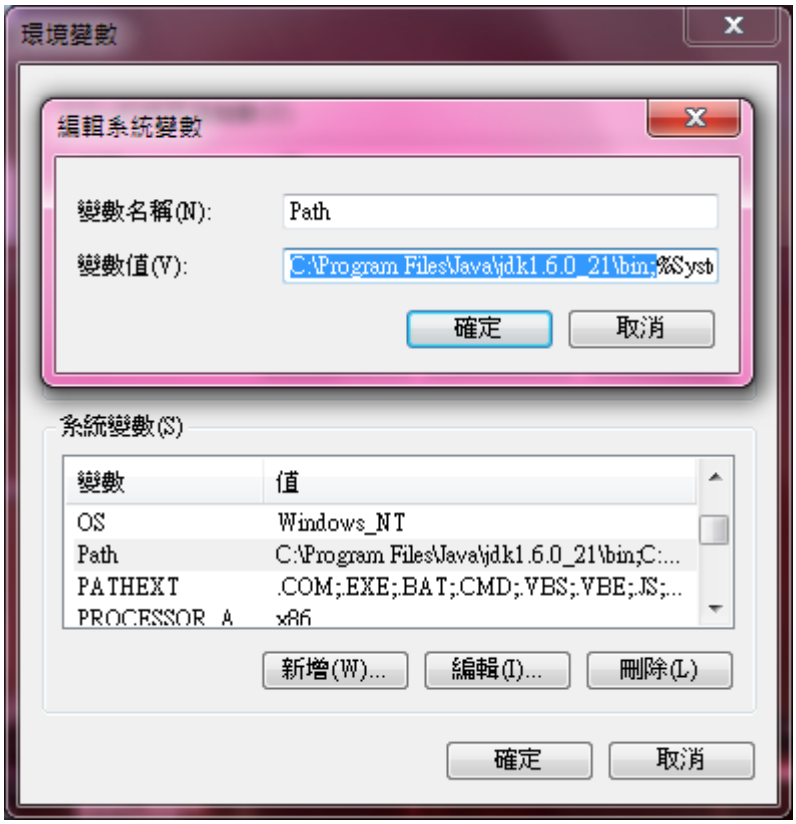
變數名稱：**JAVA_HOME**

路徑位置：**C:\Program Files\Java\jdk1.6.0_21**

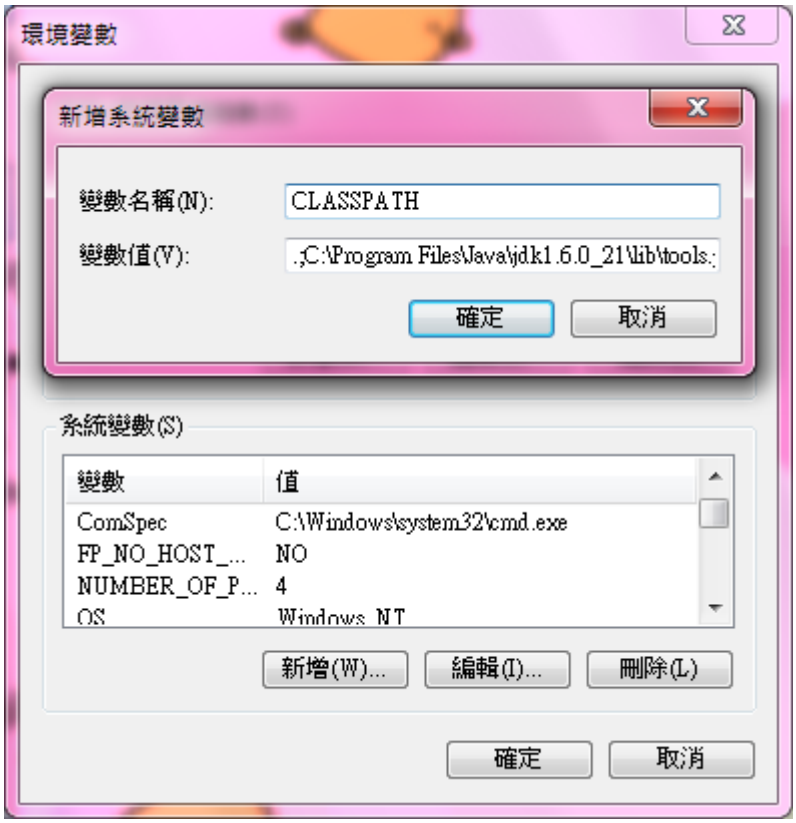
(此為安裝 Java 時選擇之安裝路徑)

接著在下方的系統變數選至「**Path**」，並選擇編輯。

在 Path 系統變數的最前面加入「**C:\Program Files\Java\jdk1.6.0_21\bin;**」，按下確定。



在下方的系統變數新增一個變數，名稱：「**CLASSPATH**」，
變數值：「**.;C:\Program Files\Java\jdk1.6.0_21\lib\tools.jar;**」，按下確定。



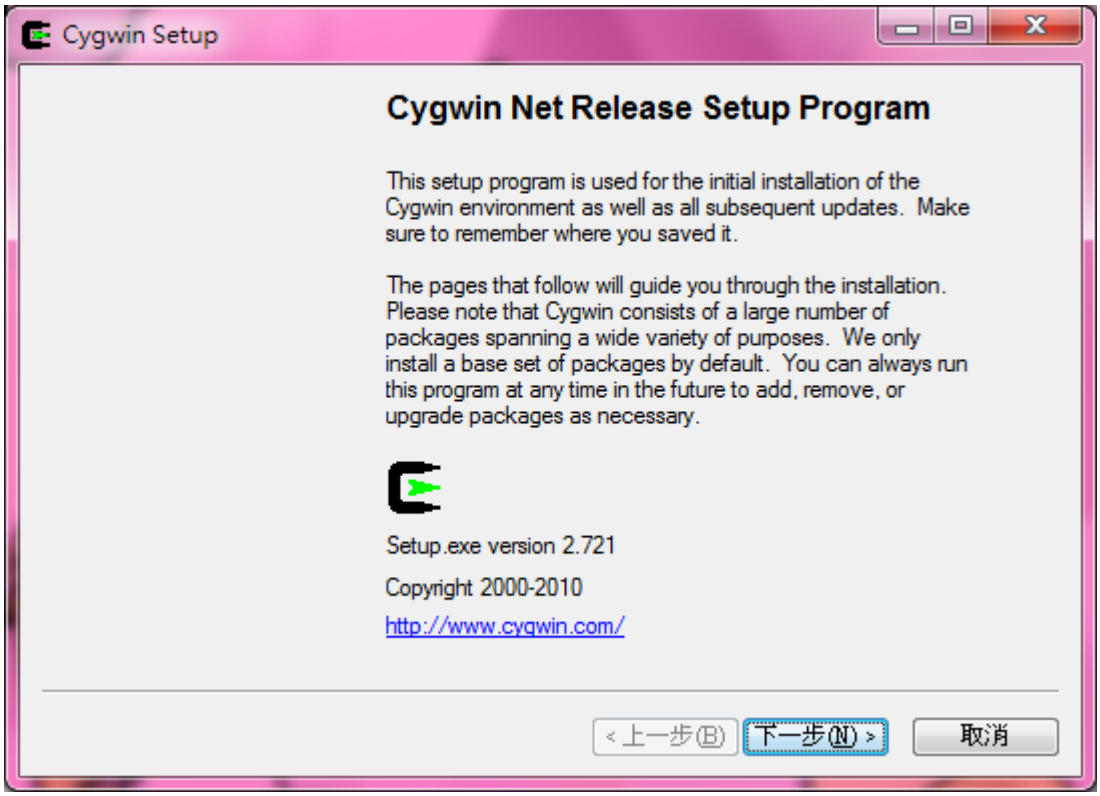
至此 Java SDK 的安裝與設定就告一段落。

Step3 安裝 Cygwin1.7 (線上安裝)

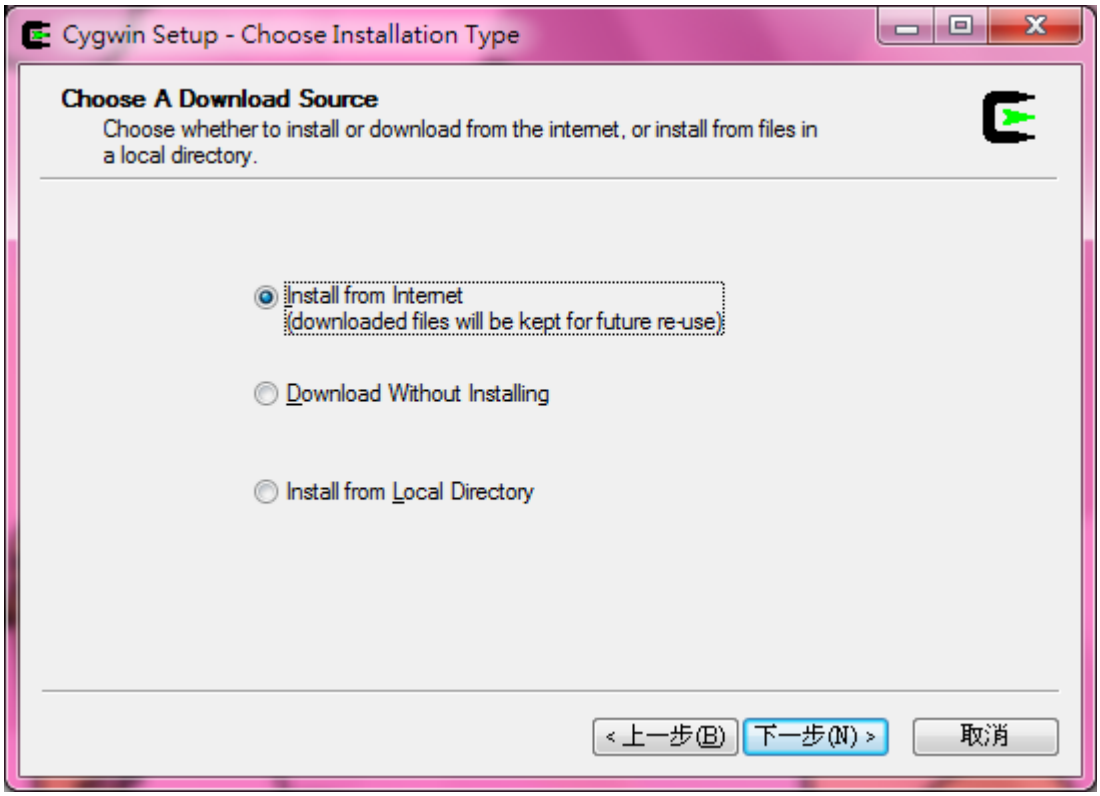
前往 http://www.cygwin.com

執行 **setup.exe**

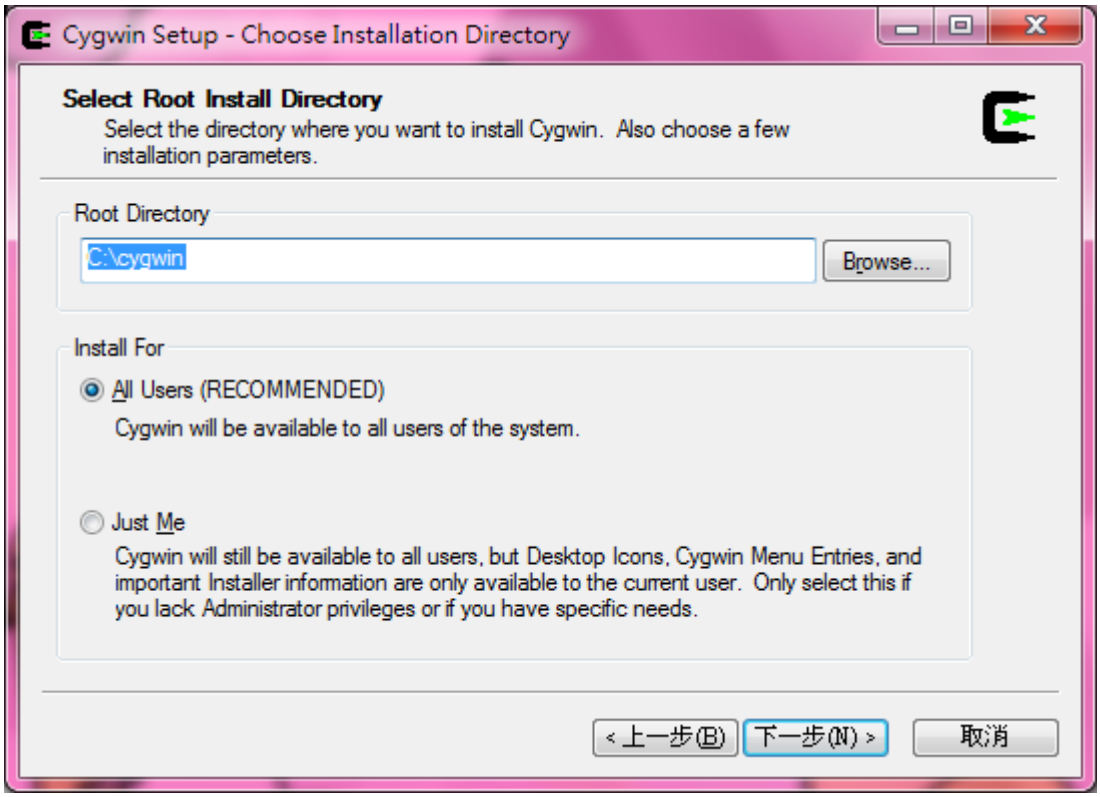
第一個 Setup 畫面直接按下下一步



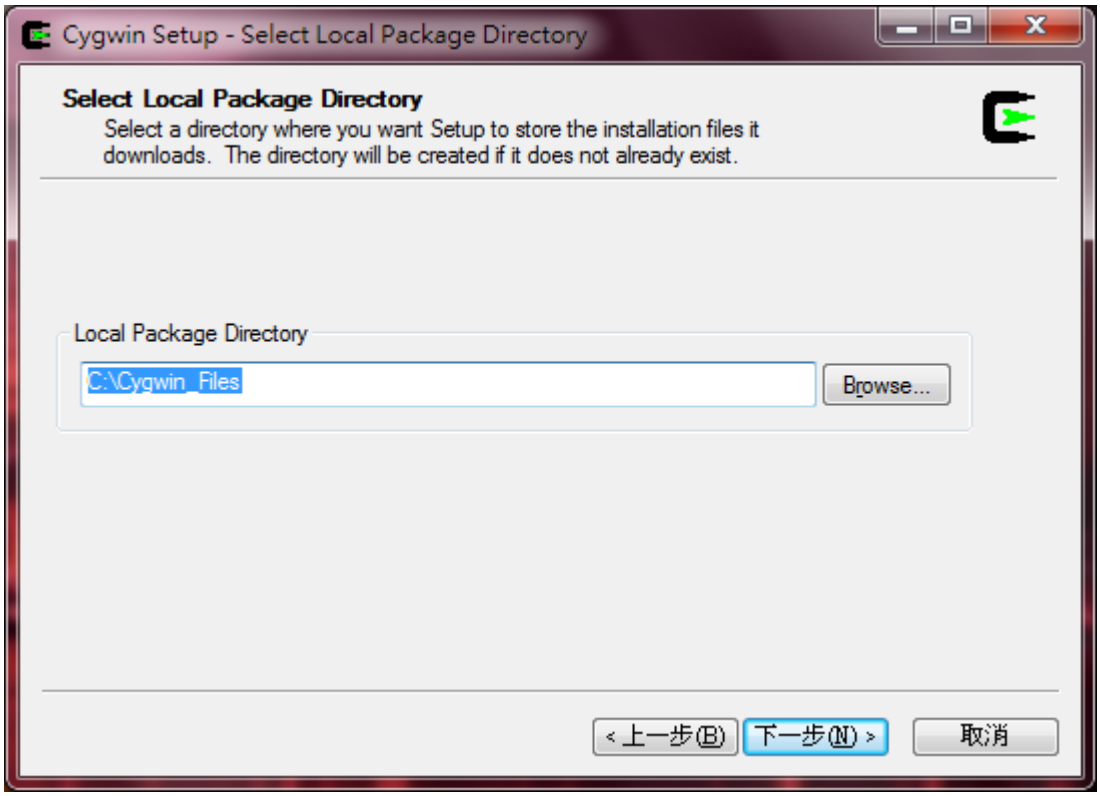
在 **Choose Installation Type** 時，選擇「**Install from Internet**」。



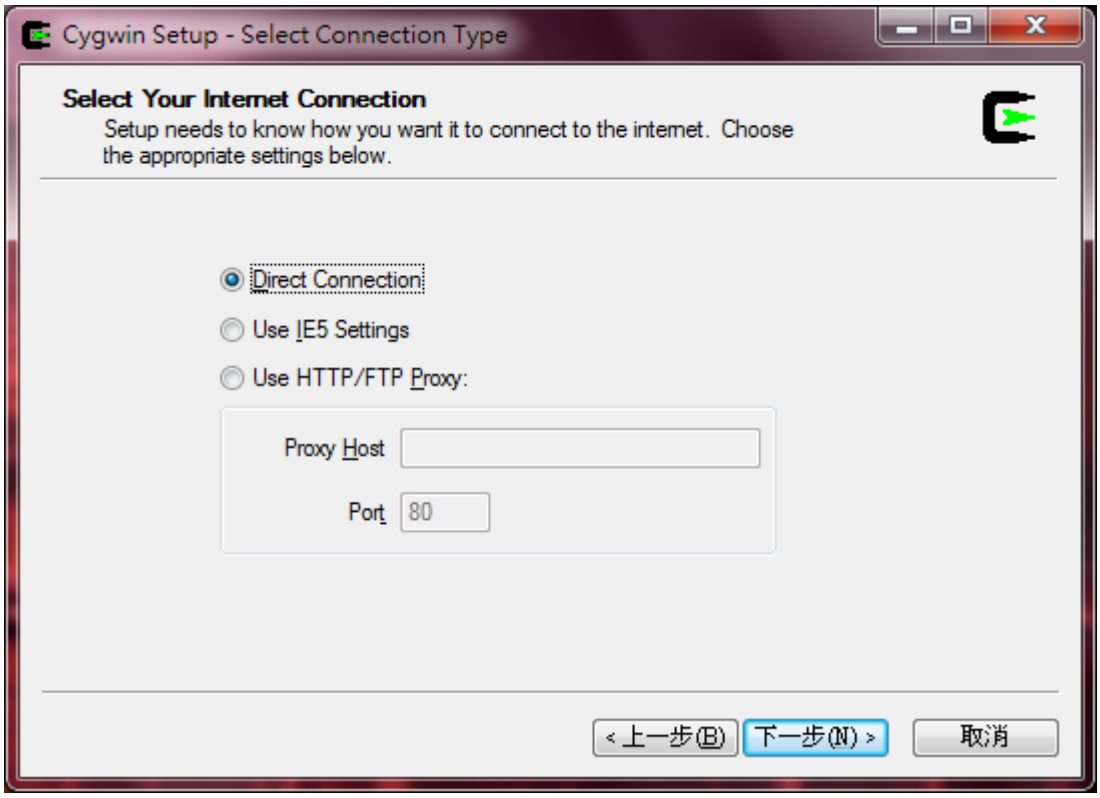
在 **Choose Installation Direction** 時，照預設值直接按下一步。



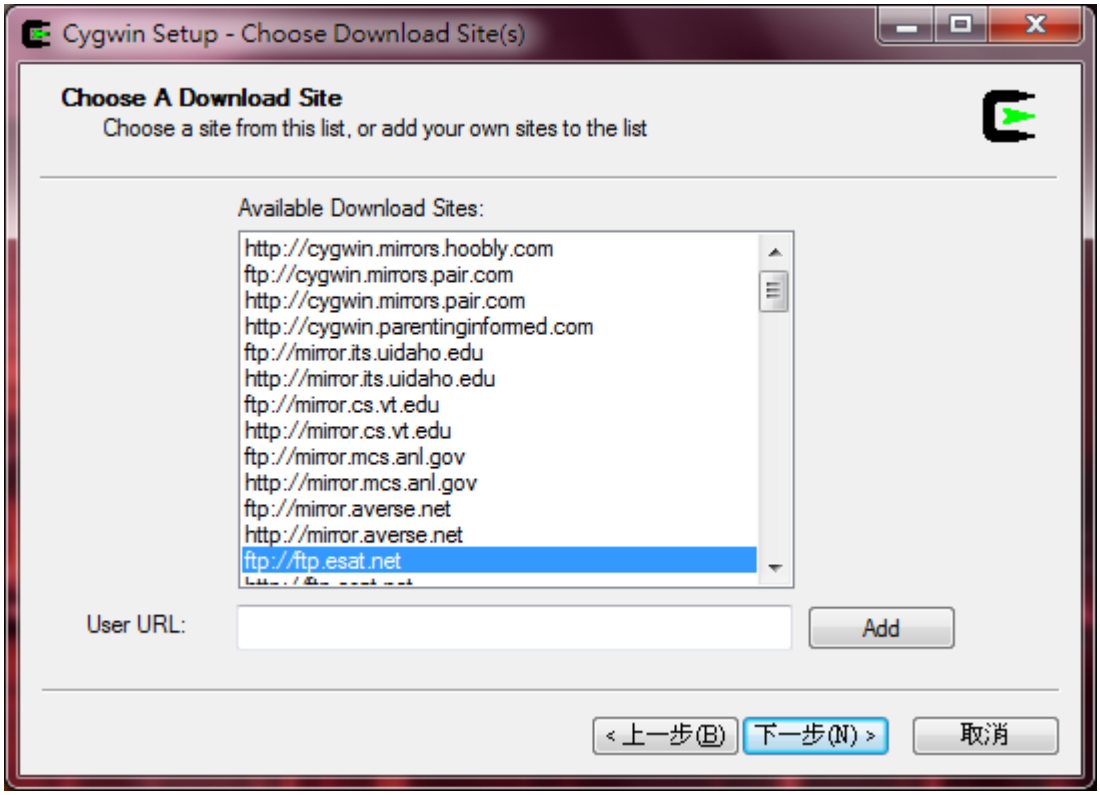
接著 **Select Local Package Direction** 為選擇「**存放由網路下載回來檔案的資料夾**」，設定完之後按下下一步。



在 **Select Connection Type** 時，選擇「**Direct Connection**」。

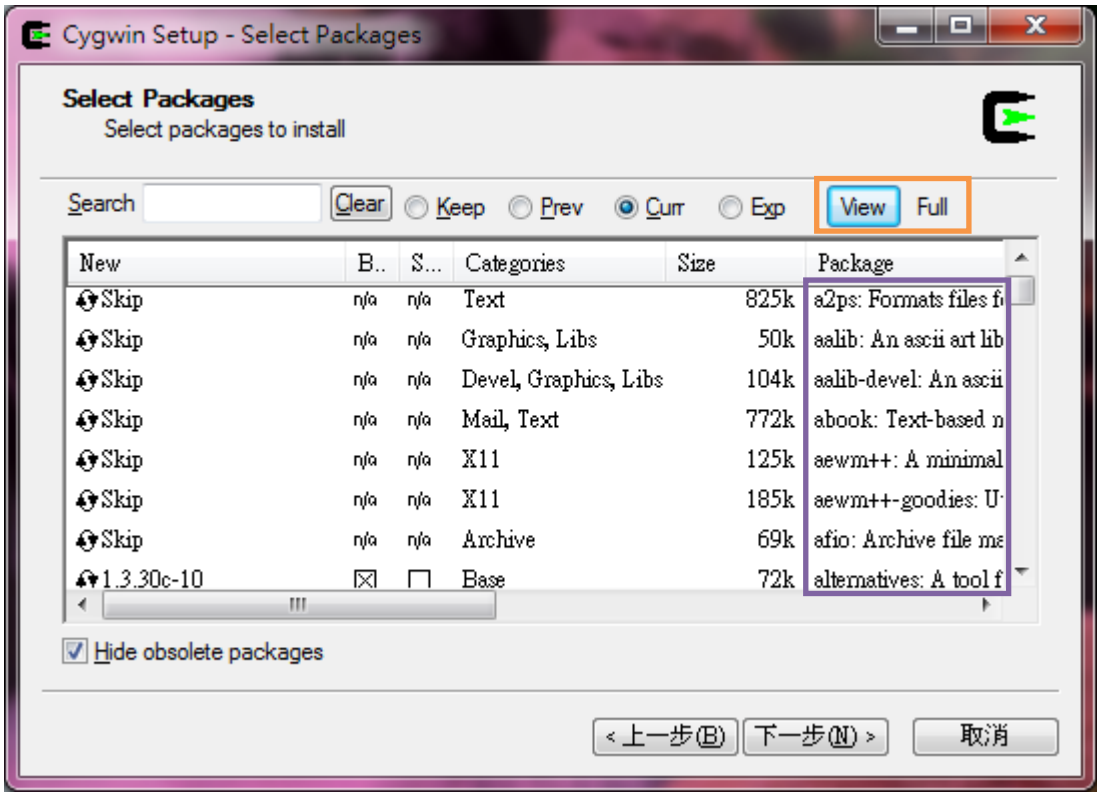


接著選擇下載來源 **Choose Download Sites**。建議選擇離自己較近的網站或台灣的網站（如 NTU 等）。



接著來到 **Select Packages** 部分。

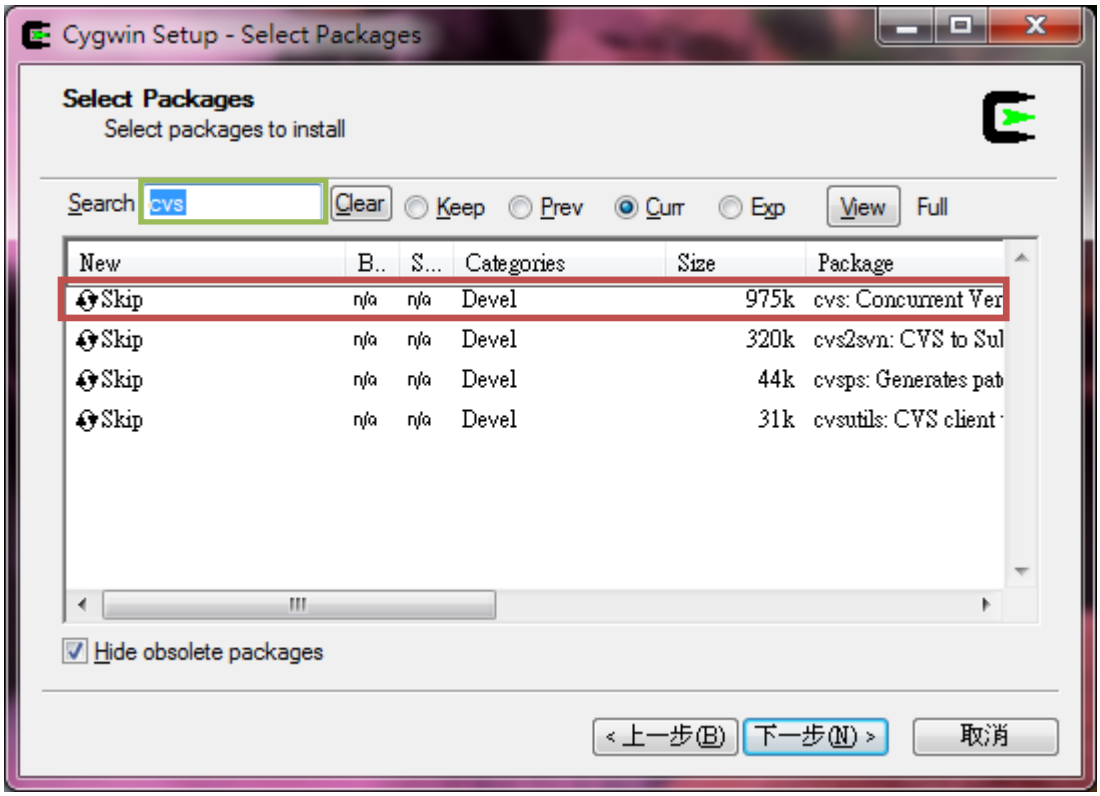
先按下右上角的“**View**”按鈕，使旁邊變成“**full**”，之後 packages 會變成由 a-z 順序列出。



安裝“**cvs**”，“**gcc**”，“**make**”，“**perl**”，“**rpm**”，和“**wget**”這六個 package。（點選 ***Skip*** 的圖案安裝）

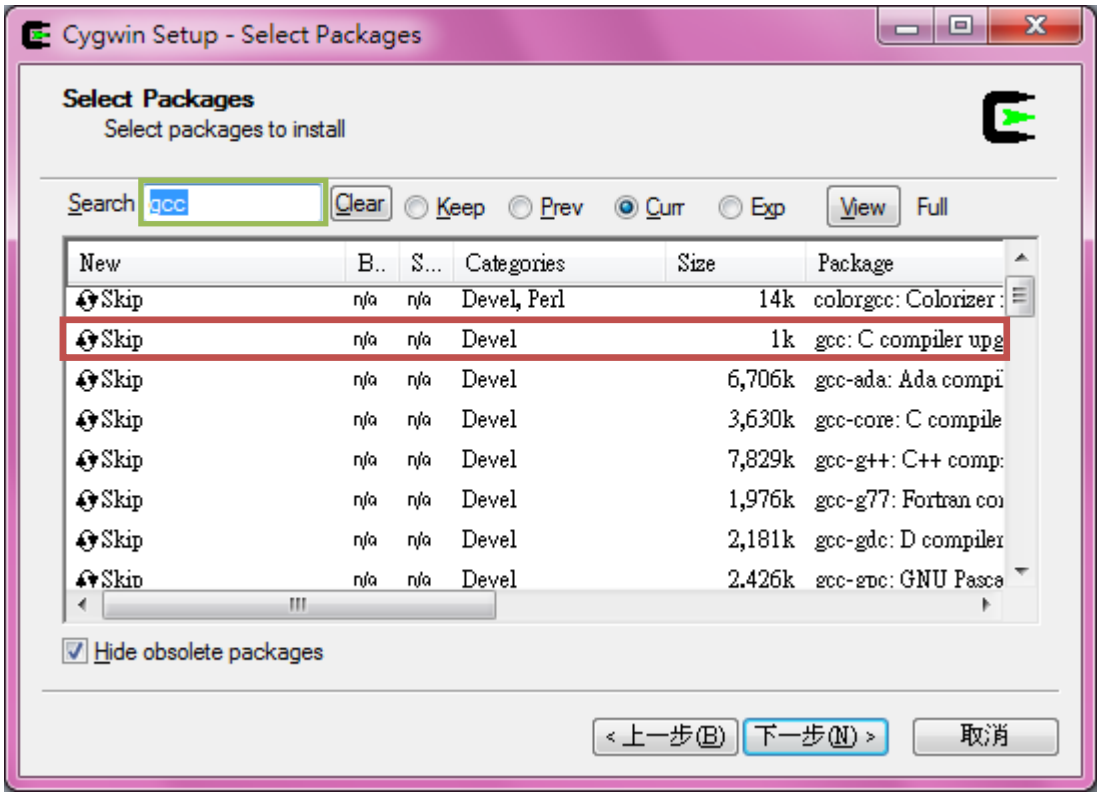
***cvs*：於“Search”欄位中輸入“cvs”，並點選“Skip”圖示進行安裝**

CVS（Concurrent Versions System）是一種版本控制系統，方便軟體的開發和使用者協同工作，很多開放原始碼或自由軟體皆使用 CVS 以便結合各程式設計師的改進和更改，這些項目包括 Gnome、KDE、GIMP、Wine 等，CVS 以獲得 GNU 的通用公共許可證授權，CVS 系統主要是利用將一組文件放在層次目錄樹中以保持同步，程式設計師可從 CVS 伺服器上更新他們本地層次樹的副本，並將修改結果或新文件發回，亦可刪除舊文件。



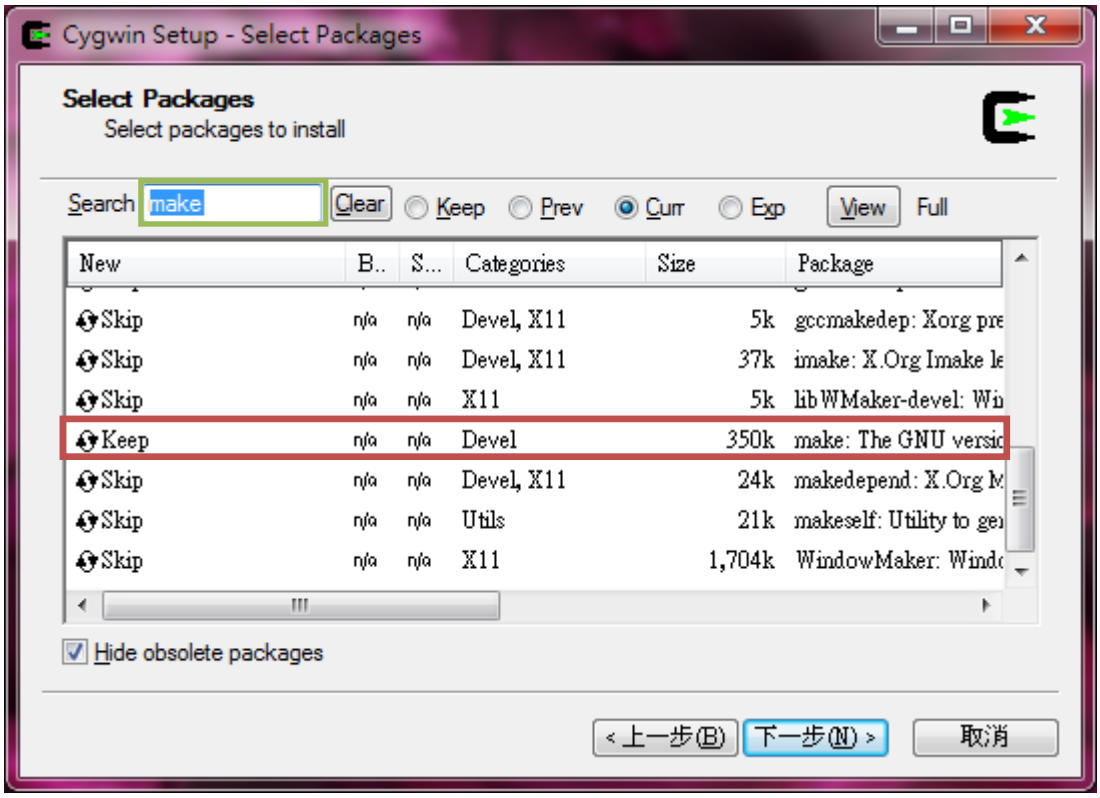
***gcc*：於“Search”欄位中輸入“gcc”，並點選“Skip”圖示進行安裝**

GCC（GNU Compiler Collection，GNU 編譯器套裝），是一套由 GNU 開發的編程語言編譯器，除了是一套擁有 GPL 及 LGPL 許可證的自由軟體外，亦是 Unix 及蘋果電腦 Mac OS X 作業系統的標準編譯器，GCC 原名為 GNU C 語言編譯器（GNU C Compiler），因為原本只能處理 C 語言，但後來快速地擴展，使其可處理 C++等語言，我們使用 GCC 主要是用來編譯 Octopus II 的 nes C 程式。



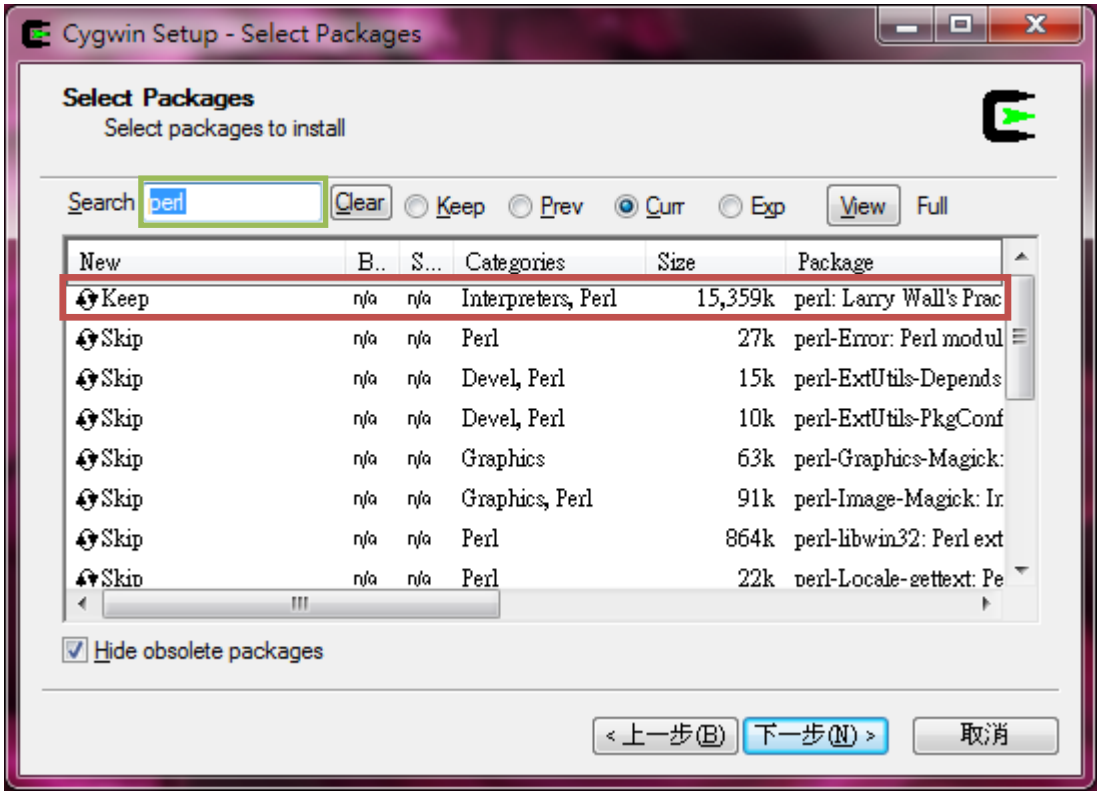
make：**於“Search”欄位中輸入“make”，並點選“Skip”圖示進行安裝**

MAKE 是一個會自動讀取原始碼、自動連結需要的編譯工具、並將原始碼轉譯為指定平台之專屬可執行檔。



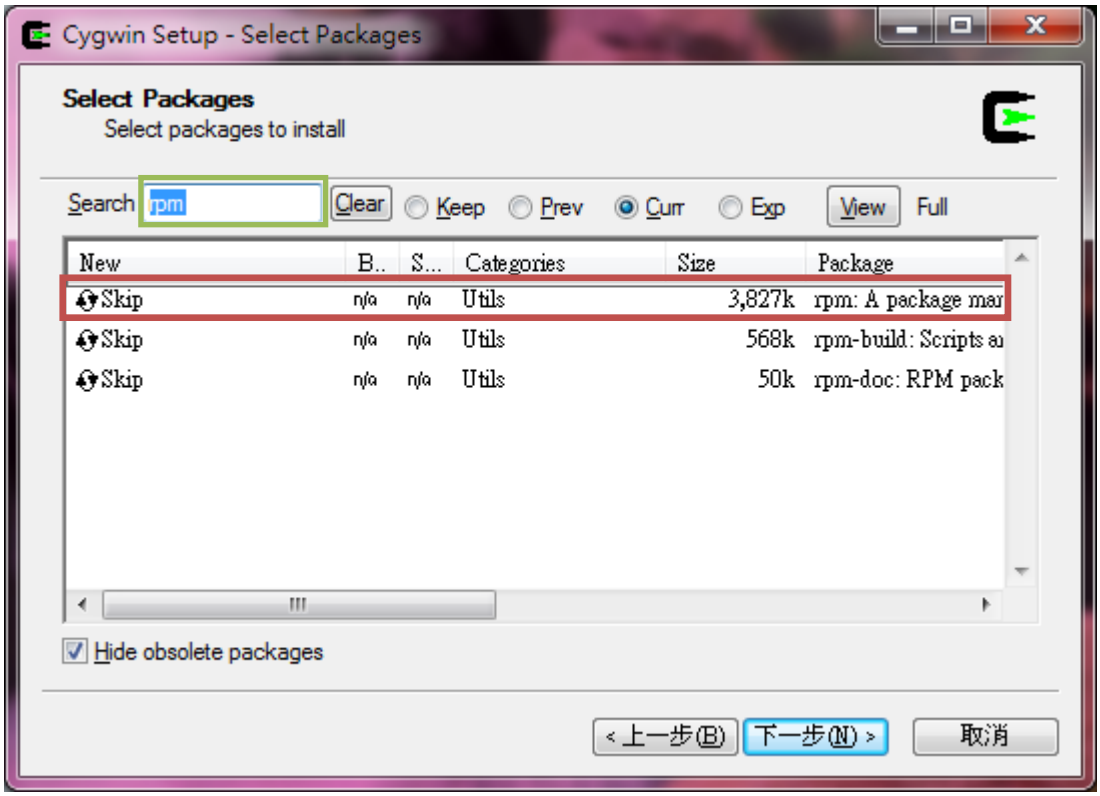
***perl*：於“Search”欄位中輸入“perl”，並點選“Skip”圖示進行安裝**

安裝 perl 的 Interpreter 主要是為了能夠順利使用 cygwin 在 windows 下模擬及執行 perl 程式。



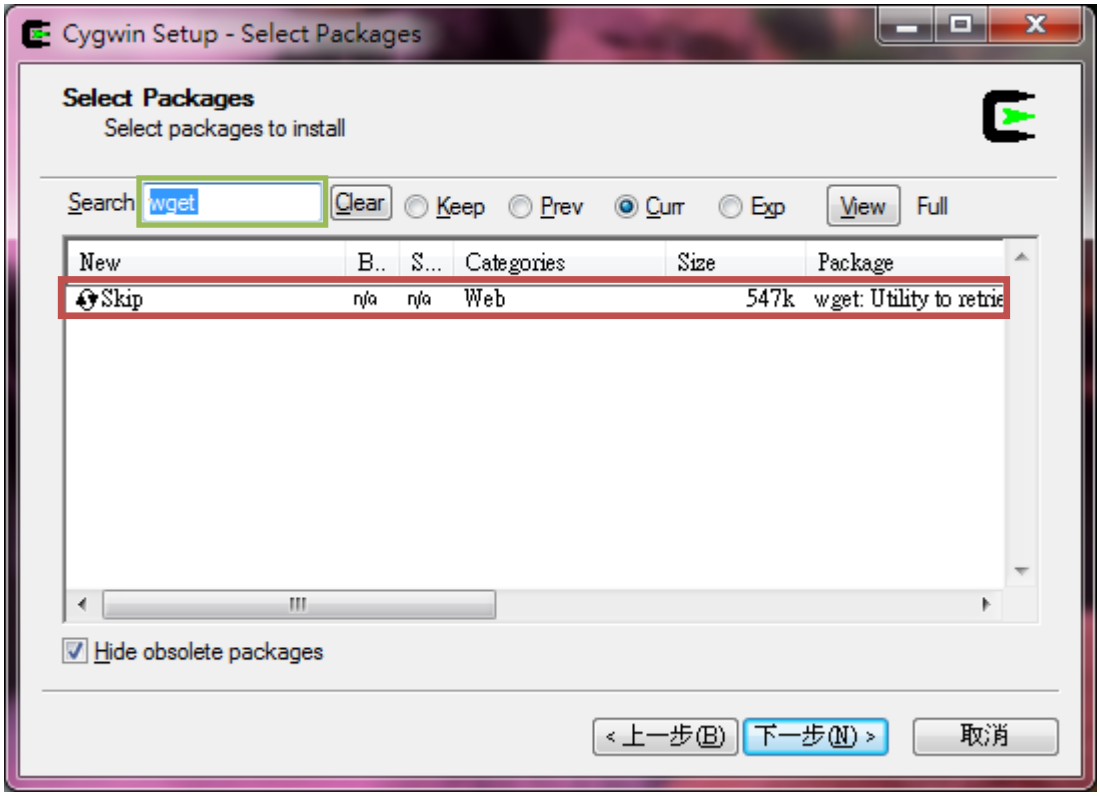
***rpm*：於“Search”欄位中輸入“rpm”，並點選“Skip”圖示進行安裝**

RPM 套件管理員（簡稱 RPM，全稱為 The RPM Package Manager），是在 Linux 下被廣泛使用的套件管理軟體，最早是由 Red Hat 公司研發，現在則由開源社群接續開發，RPM 套件分為二進制（Binary）、原始碼（Source）和 Delta 三種，二進制套件可直接安裝於電腦中，而原始碼套件將會由 RPM 自動編譯與安裝，原始碼套件經常以 src.rpm 作為副檔名。

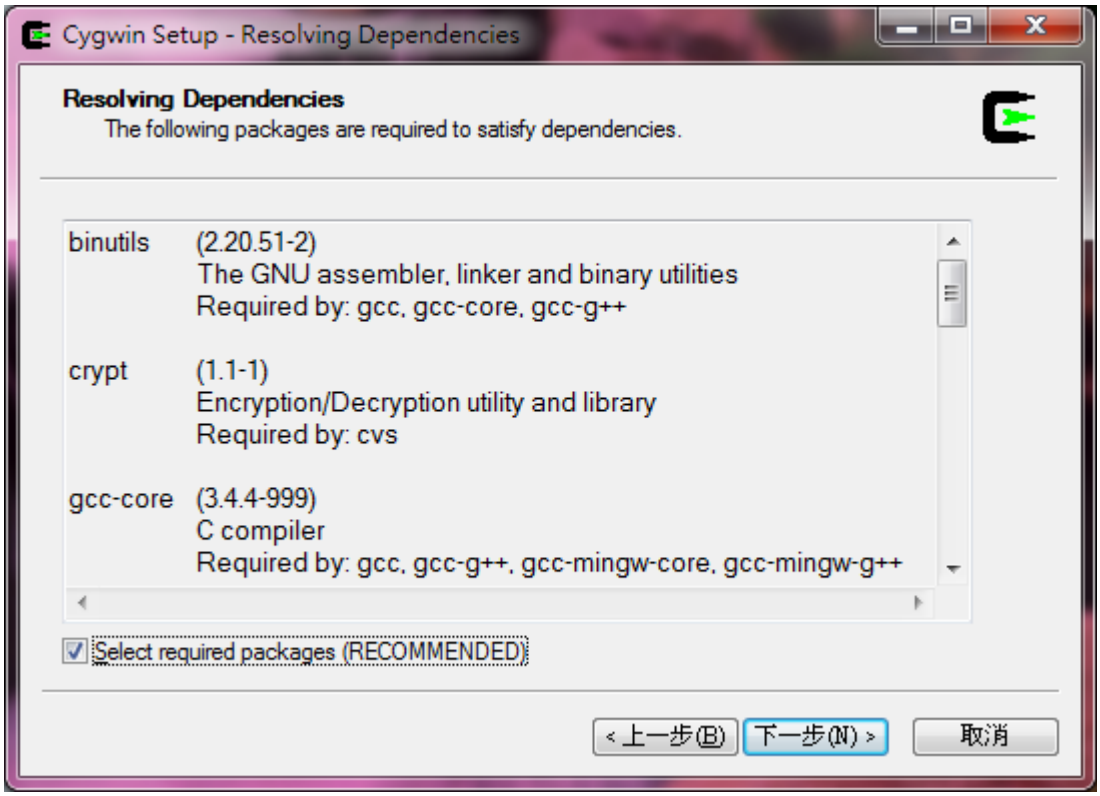


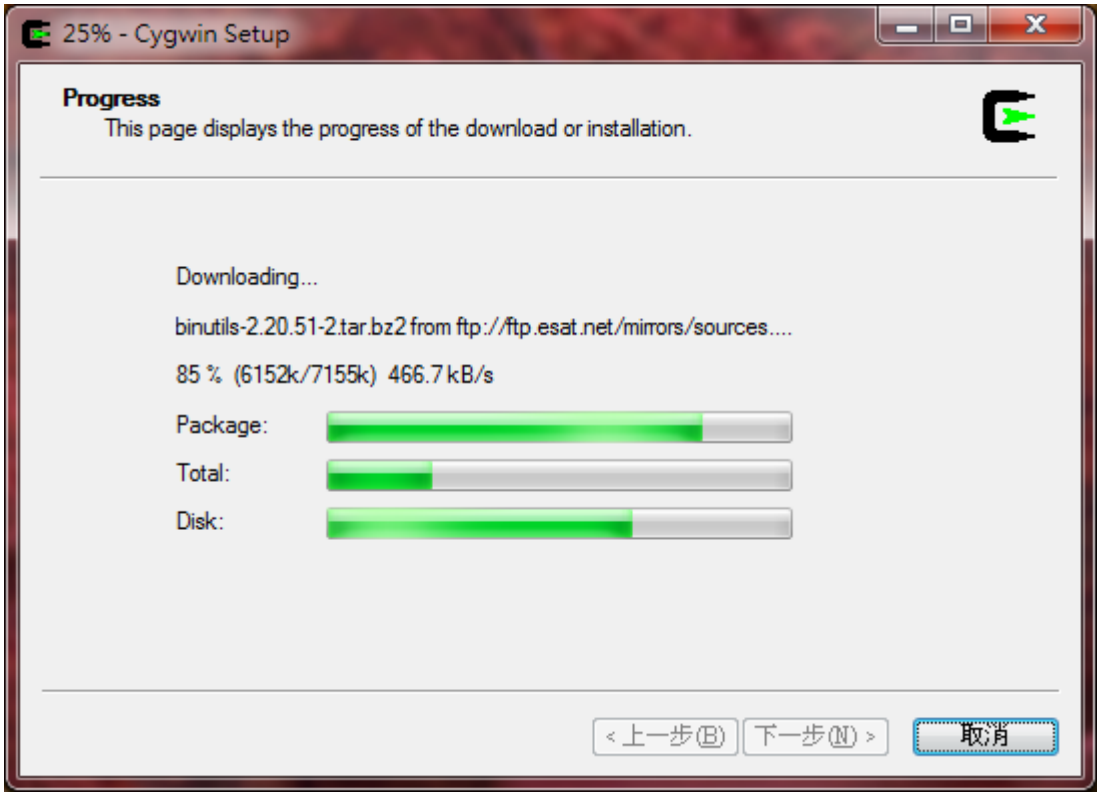
***wget*：於“Search”欄位中輸入“wget”，並點選“Skip”圖示進行安裝**

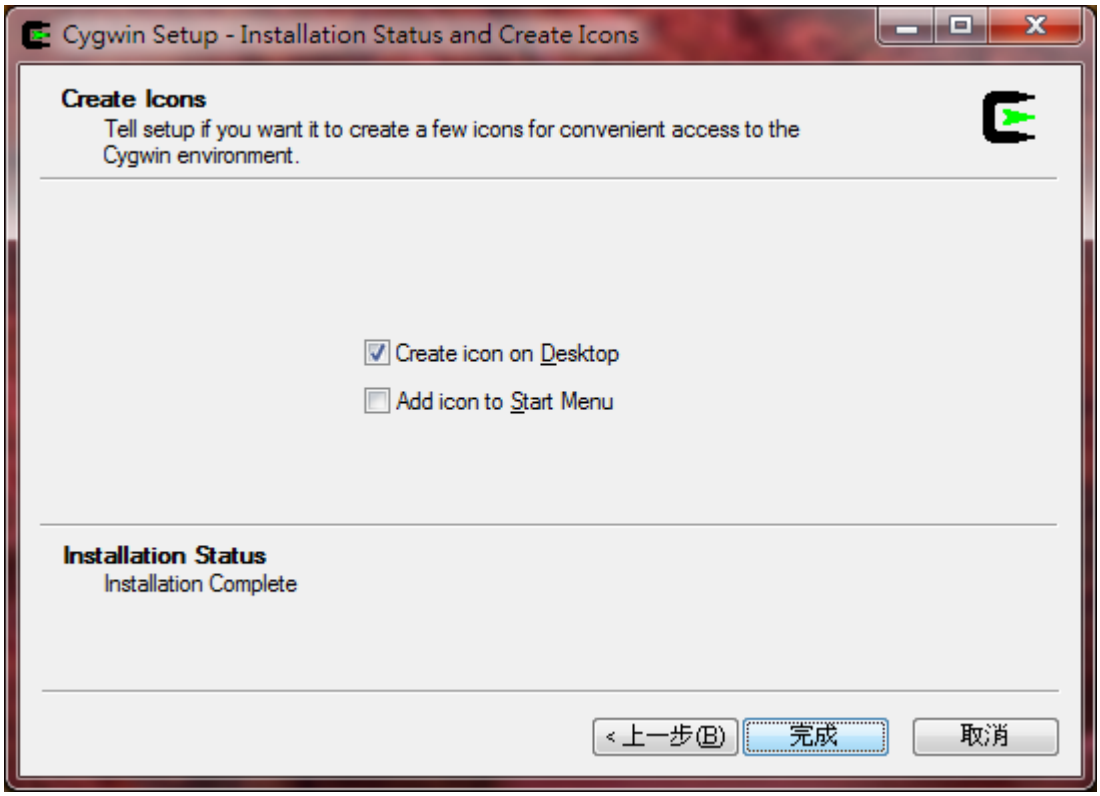
WGET 是一套在網路上進行程式下載的自由軟體，WGET 簡稱是由「World Wide Web」和「GET」結合，目前支援 HTTP、HTTPS 以及 FTP 三種最常見的網路下載。



選下一步以進行下載與安裝。







到此步驟便完成 Cygwin 環境的安裝。

Step4 安裝 TinyOS 2.1

安裝目標節點編譯器

因為我們要編譯 MCU 上的程序，所以需要能夠生成相應匯編語言的編譯器。如果你使用 mica 系列的節點，就需要 AVR 的 tool chain；如果是 telos 系列的節點，那就需要 MSP430 tool chain。

OctopusII 是使用 TI 公司的 MSP430 處理器，故要安裝 MSP430 的編譯器，請至 http://www.tinyos.net/dist-2.1.0/tools/windows/ 下載，共 5 個套件。

Base

http://www.tinyos.net/dist-2.1.0/tools/windows/msp430tools-base-0.1-20050607.cygwin.i386.rpm

python tools

http://www.tinyos.net/dist-2.1.0/tools/windows/msp430tools-python-tools-1.0-1.cygwin.noarch.rpm

binutils

http://www.tinyos.net/dist-2.1.0/tools/windows/msp430tools-binutils-2.16-20050607.cygwin.i386.rpm

gcc

http://www.tinyos.net/dist-2.1.0/tools/windows/msp430tools-gcc-3.2.3-20050607.cygwin.i386.rpm

libc

http://www.tinyos.net/dist-2.1.0/tools/windows/msp430tools-libc-20080808-1.cygwin.i386.rpm

先將下載的安裝檔存至「**c:\cygwin\rpm\msp430rpm**」下(自行新增 rpm 資料夾)，並開啟 Cygwin。

```
~
Copying skeleton files.
These files are for the user to personalise their cygwin experience.

They will never be overwritten nor automatically updated.

`./.bashrc' -> `/home/YiYu//.bashrc'
`./.bash_profile' -> `/home/YiYu//.bash_profile'
`./.inputrc' -> `/home/YiYu//.inputrc'

YiYu@YiYu   I x ~
$
```

輸入 **cd /rpm/msp430rpm/**，意思為切換至安裝檔 msp430rpm 目錄下。

輸入 **rpm -ivh --nodeps --ignoreos msp***以安裝所需的工具。

```
/rpm/msp430rpm
Copying skeleton files.
These files are for the user to personalise their cygwin experience.

They will never be overwritten nor automatically updated.

`./.bashrc' -> `/home/YiYu//.bashrc'
`./.bash_profile' -> `/home/YiYu//.bash_profile'
`./.inputrc' -> `/home/YiYu//.inputrc'

YiYu@YiYu   I x ~
$ cd /rpm/msp430rpm/

YiYu@YiYu   I x /rpm/msp430rpm
$ rpm -ivh --nodeps --ignoreos msp*
Preparing...                ########################################### [100%]
   1:msp430tools-python-tool########################################### [ 20%]
   2:msp430tools-base       ########################################### [ 40%]
   3:msp430tools-binutils   ########################################### [ 60%]
   4:msp430tools-gcc        ########################################### [ 80%]
   5:msp430tools-libc       ########################################### [100%]

YiYu@YiYu   I x /rpm/msp430rpm
$
```

安裝 nesC 編譯器

TinyOS 用 nesC 寫成。nesC 是 C 的一個分支，它支援 TinyOS 中的併發模型並且使用基於組件（component）的編程。nesC 編譯器是相對於節點而獨立的，它的編譯輸出會傳遞給目標節點的編譯器，所以你可以充分發揮你的主觀能動性對代碼進行優化。

請至 http://www.tinyos.net/dist-2.1.0/tinyos/windows/ 下載 nesC 與 TinyOS-tools 存至「c:\cygwin\rpm\tinyosrpm」下。

NesC

http://www.tinyos.net/dist-2.1.0/tinyos/windows/nesc-1.3.0-1.cygwin.i386.rpm

Deputy

http://www.tinyos.net/dist-2.1.0/tinyos/windows/tinyos-deputy-1.1-1.cygwin.i386.rpm

TinyOS-tools

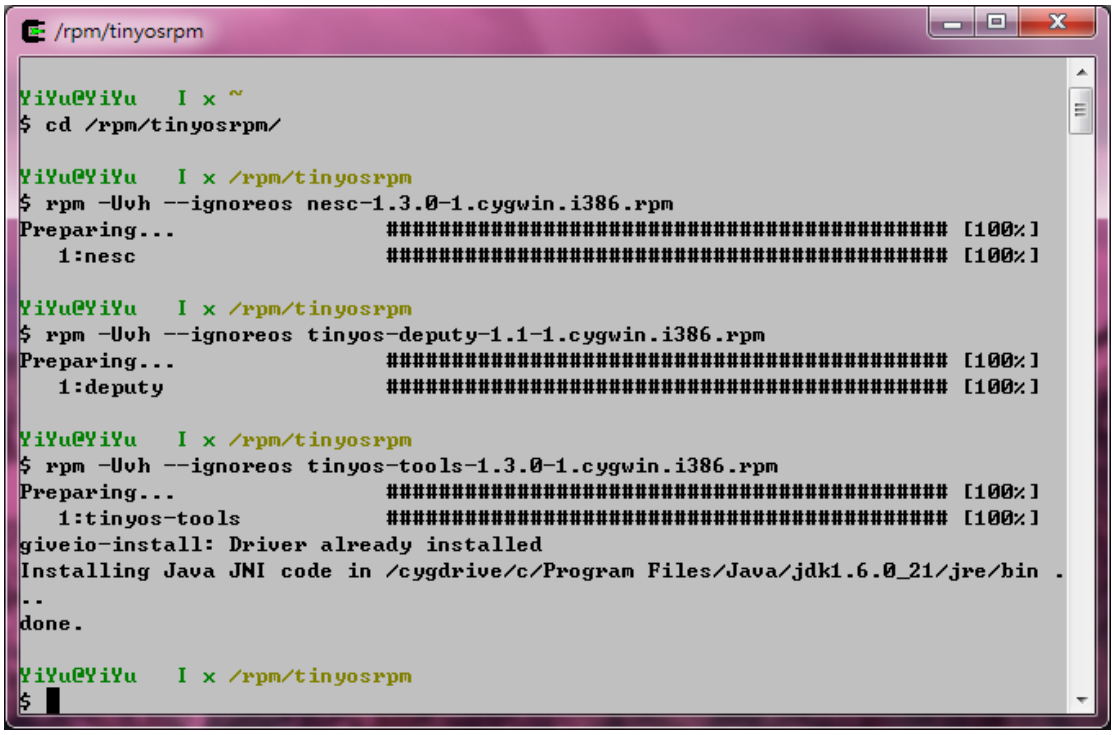
http://www.tinyos.net/dist-2.1.0/tinyos/windows/tinyos-tools-1.3.0-1.cygwin.i386.rpm

輸入 **cd /rpm/tinyosrpm/**切換至安裝檔 tinyosrpm 目錄下

輸入 **rpm -Uvh --ignoreos nesc-1.3.0-1.cygwin.i386.rpm**

輸入 **rpm -Uvh --ignoreos tinyos-deputy-1.1-1.cygwin.i386.rpm**

輸入 **rpm -Uvh --ignoreos tinyos-tools-1.3.0-1.cygwin.i386.rpm**



安裝 TinyOS Source Tree

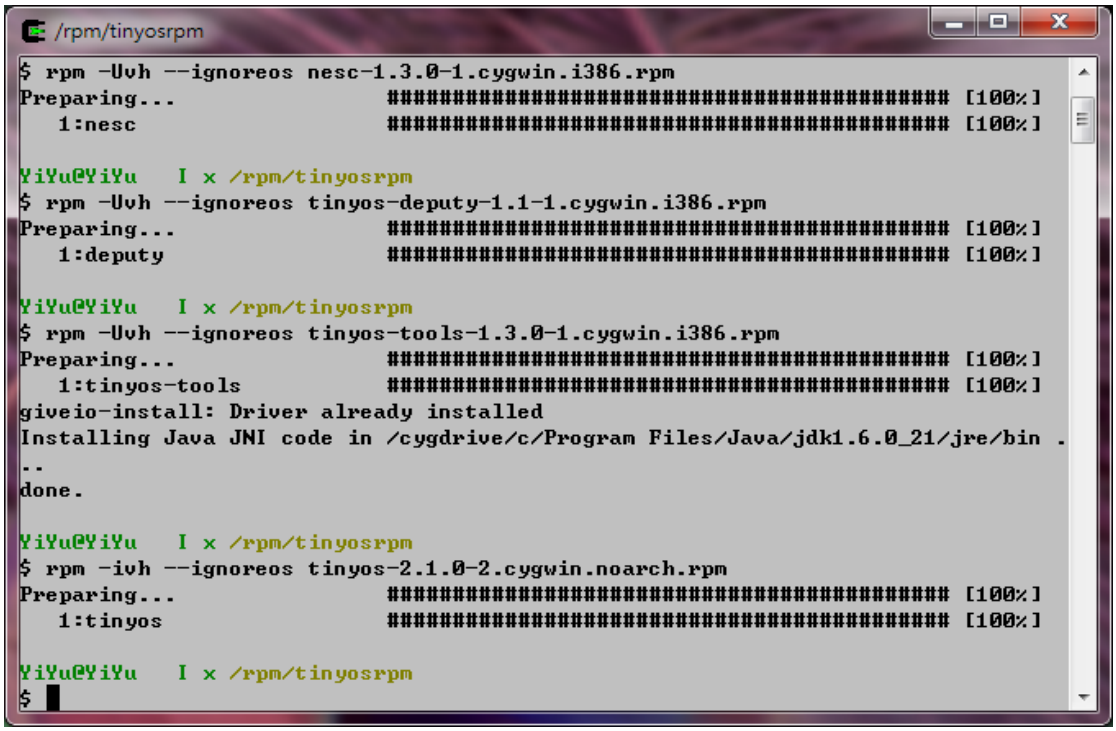
使用 TinyOS Source Tree 做為編譯和下載 TinyOS 的程式。

TinyOS 2.x

http://www.tinyos.net/dist-2.1.0/tinyos/windows/tinyos-2.1.0-2.cygwin.noarch.rpm

同樣在 tinyosrpm 目錄下：

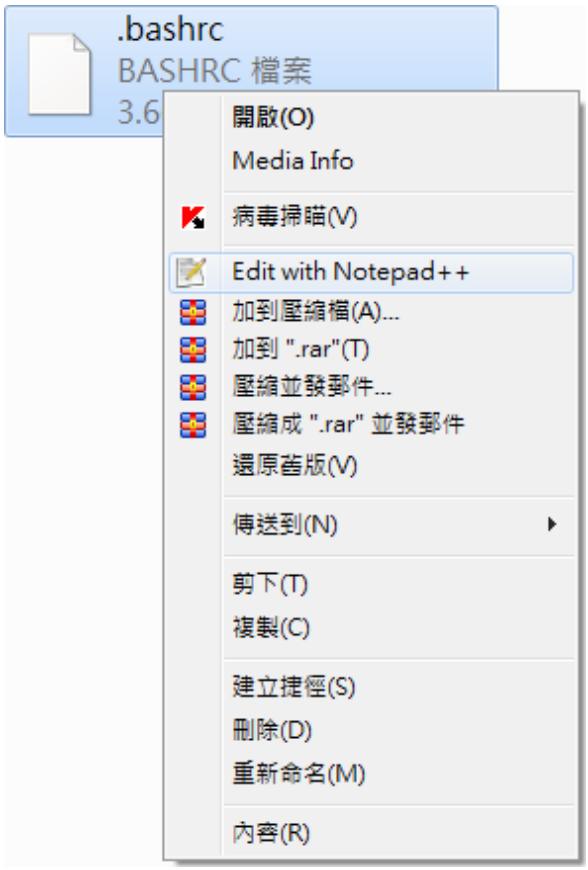
輸入 **rpm -ivh --ignoreos tinyos-2.1.0-2.cygwin.noarch.rpm**



完成後關閉 cygwin。

<u>修改環境變數</u>

在使用者目錄下（C:\cygwin\home\使用者\），用 Notepad++（或記事本）開啟**.bashrc**。



將下列文字貼在最後一行下面

TinyOS 2.x

export TOSROOT="/opt/tinyos-2.x"

export TOSDIR="$TOSROOT/tos"

export CLASSPATH="C:\cygwin\opt\tinyos-2.x\support\sdk\java\tinyos.jar;."

export MAKERULES="$TOSROOT/support/make/Makerules"

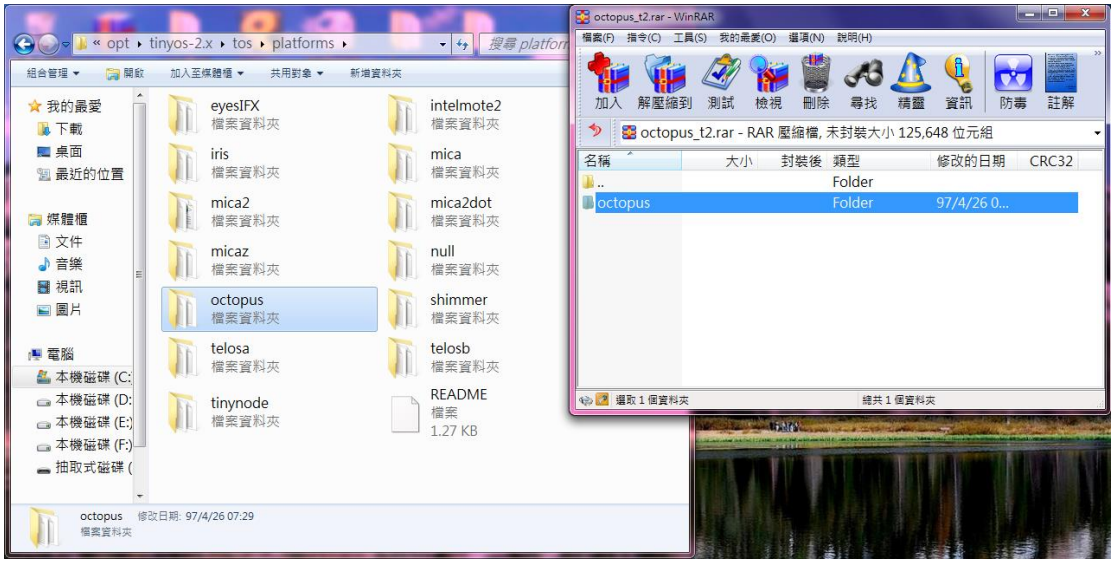
（紅色字體部分請根據安裝 Cygwin 的位置作變更，要注意換行符號的問題。)

```
124
125 # Functions
126 # #########
127
128 # Some example functions
129 # function settitle() { echo -ne "\e]2;$@\a\e]1;$@\a"; }
130
131 # TinyOS 2.x
132 export TOSROOT="/opt/tinyos-2.x"
133 export TOSDIR="$TOSROOT/tos"
134 export CLASSPATH="C:\cygwin\opt\tinyos-2.x\support\sdk\java\tinyos.jar;."
135 export MAKERULES="$TOSROOT/support/make/Makerules"
136
```

安裝 OctopusII 所需的編譯文件

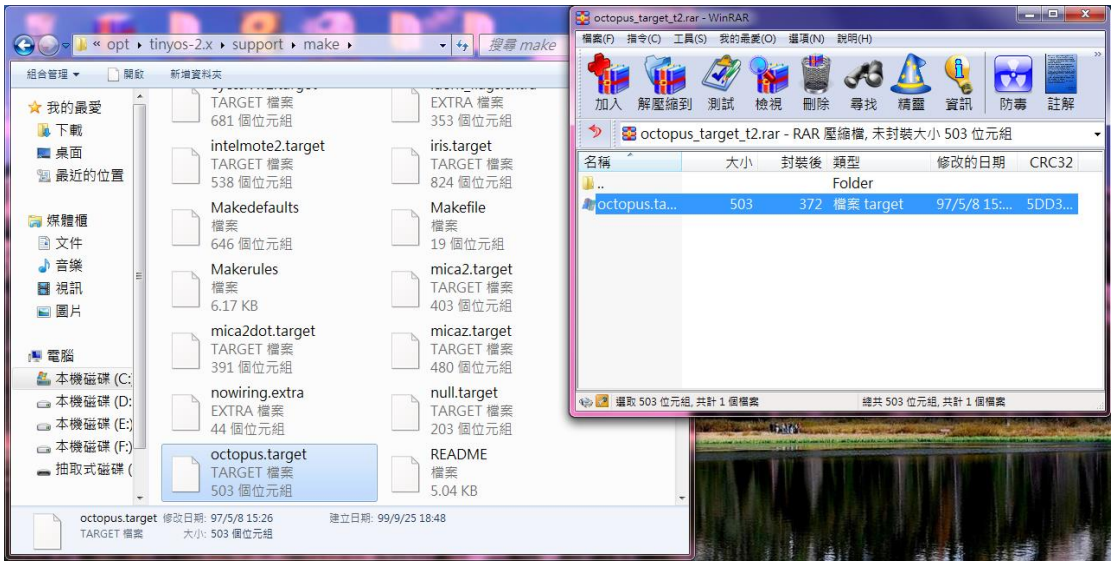
解壓縮 **octopus_t2.rar**，放至

/opt/tinyos-2.x/tos/platforms/ 中。



解壓縮 **octopus_target_t2.rar**，放至

/opt/tinyos-2.x/support/make/ 中。

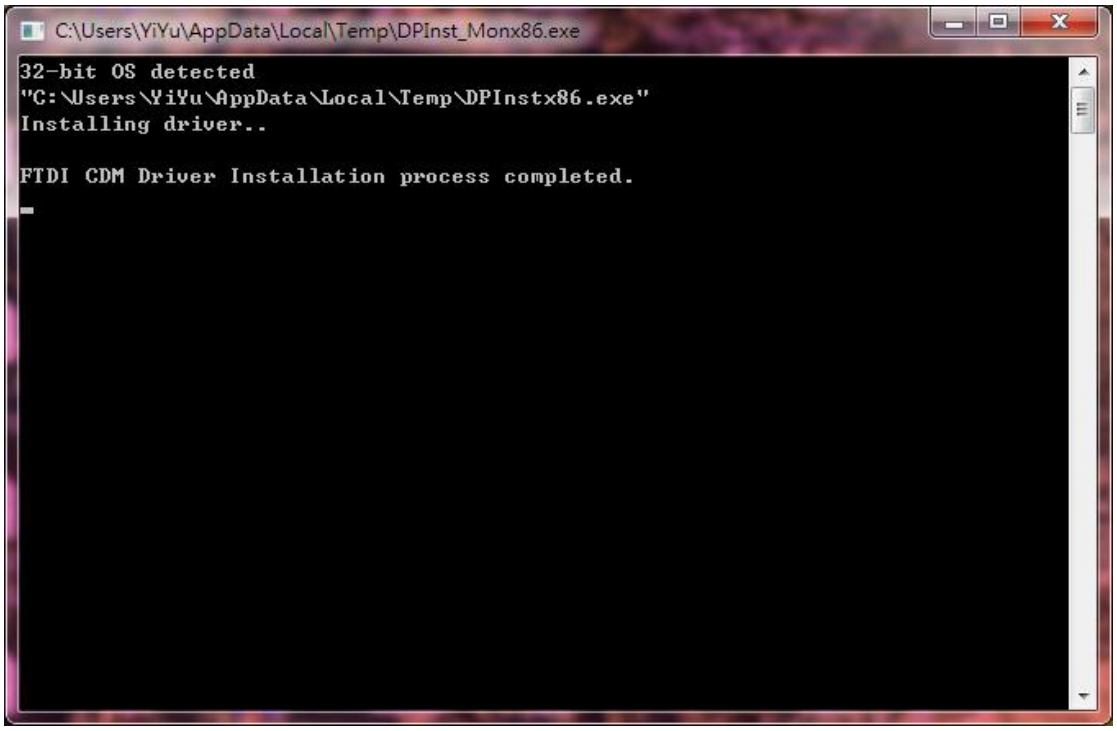


安裝 OctopusII Driver

至 FTDI VirtualComm USB Drivers（http://www.ftdichip.com/Drivers/VCP.htm）下載 USB 轉 RS232 驅動程式。

Operating System	Devices Supported	Driver Version	Release Date	Comments
Windows Server 2008 R2 Windows 7 Windows 7 x64 Windows Server 2008 Windows Server 2008 x64 Windows Vista Windows Vista x64 Windows XP Windows XP x64 Windows 2000 Windows Server 2003 Windows Server 2003 x64	FT2232H, FT4232H, FT232R, FT245R, FT2232, FT232B, FT245B, FT8U232AM, FT8U245AM	2.08.02 (zip file) 2.08.02 (setup exe)	11th August 2010	Microsoft WHQL certified. For custom VID and PID combinations see AN232R-03. Combined driver model (D2XX and VCP). Devices programmed as VCP will expose a COM port, as will AM and BM devices. Release Notes

下載完後安裝即可。



Part III 硬體電路整合

Step1 選擇支援 GPIO 的腳位

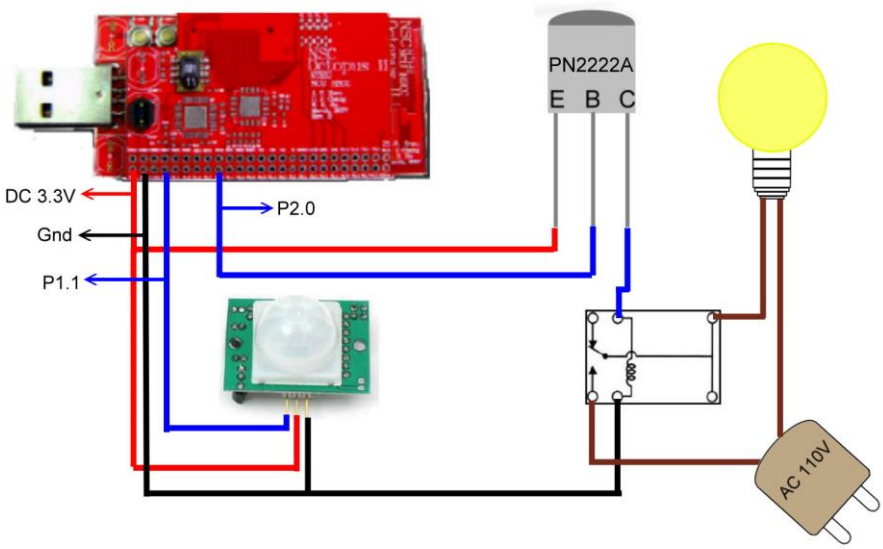
為了接收 Passive Infrared Motion Sensor 感測結果，我們需於 OctopusII 上選擇一個支援 GPIO 的腳位用以接收 Passive Infrared Motion Sensor 感應器的 OUT 腳位輸出，由於 Octopus 之 MCU 為 MSP430，我們可由 MSP430 的 datasheet 查出支援 GPIO 的腳位，經由下表可得知各腳位可支援的通訊介面，若 datasheet 中的 Description 欄位內有標示 General-Purpose digital I/O 即代表該腳位支援 GPIO，因此腳位 P1.0(NO.12)至 P5.3(NO.47)皆可支援 GPIO，而腳位 P2.6 亦額外支援 ADC 通訊介面。

Terminal Functions

TERMINAL NAME	NO.	I/O	DESCRIPTION
AV_{CC}	64		Analog supply voltage, positive terminal. Supplies only the analog portion of ADC12 and DAC12.
AV_{SS}	62		Analog supply voltage, negative terminal. Supplies only the analog portion of ADC12 and DAC12.
DV_{CC}	1		Digital supply voltage, positive terminal. Supplies all digital parts.
DV_{SS}	63		Digital supply voltage, negative terminal. Supplies all digital parts.
P1.0/TACLK	12	I/O	General-purpose digital I/O pin/Timer_A, clock signal TACLK input
P1.1/TA0	13	I/O	General-purpose digital I/O pin/Timer_A, capture: CCI0A input, compare: Out0 output/BSL transmit
P1.2/TA1	14	I/O	General-purpose digital I/O pin/Timer_A, capture: CCI1A input, compare: Out1 output
P1.3/TA2	15	I/O	General-purpose digital I/O pin/Timer_A, capture: CCI2A input, compare: Out2 output
P1.4/SMCLK	16	I/O	General-purpose digital I/O pin/SMCLK signal output
P1.5/TA0	17	I/O	General-purpose digital I/O pin/Timer_A, compare: Out0 output
P1.6/TA1	18	I/O	General-purpose digital I/O pin/Timer_A, compare: Out1 output
P1.7/TA2	19	I/O	General-purpose digital I/O pin/Timer_A, compare: Out2 output
P2.0/ACLK	20	I/O	General-purpose digital I/O pin/ACLK output
P2.1/TAINCLK	21	I/O	General-purpose digital I/O pin/Timer_A, clock signal at INCLK
P2.2/CAOUT/TA0	22	I/O	General-purpose digital I/O pin/Timer_A, capture: CCI0B input/Comparator_A output/BSL receive
P2.3/CA0/TA1	23	I/O	General-purpose digital I/O pin/Timer_A, compare: Out1 output/Comparator_A input
P2.4/CA1/TA2	24	I/O	General-purpose digital I/O pin/Timer_A, compare: Out2 output/Comparator_A input
P2.5/Rosc	25	I/O	General-purpose digital I/O pin/input for external resistor defining the DCO nominal frequency
P2.6/ADC12CLK/ DMAE0	26	I/O	General-purpose digital I/O pin/conversion clock – 12-bit ADC/DMA channel 0 external trigger
P2.7/TA0	27	I/O	General-purpose digital I/O pin/Timer_A, compare: Out0 output
P3.0/STE0	28	I/O	General-purpose digital I/O pin/slave transmit enable – USART0/SPI mode
P3.1/SIMO0/SDA	29	I/O	General-purpose digital I/O pin/slave in/master out of USART0/SPI mode, I^2C data – USART0/I^2C mode
P3.2/SOMI0	30	I/O	General-purpose digital I/O pin/slave out/master in of USART0/SPI mode
P3.3/UCLK0/SCL	31	I/O	General-purpose digital I/O pin/external clock input – USART0/UART or SPI mode, clock output – USART0/SPI mode, I^2C clock – USART0/I^2C mode
P3.4/UTXD0	32	I/O	General-purpose digital I/O pin/transmit data out – USART0/UART mode
P3.5/URXD0	33	I/O	General-purpose digital I/O pin/receive data in – USART0/UART mode
P3.6/UTXD1†	34	I/O	General-purpose digital I/O pin/transmit data out – USART1/UART mode
P3.7/URXD1†	35	I/O	General-purpose digital I/O pin/receive data in – USART1/UART mode
P4.0/TB0	36	I/O	General-purpose digital I/O pin/Timer_B, capture: CCI0A/B input, compare: Out0 output
P4.1/TB1	37	I/O	General-purpose digital I/O pin/Timer_B, capture: CCI1A/B input, compare: Out1 output
P4.2/TB2	38	I/O	General-purpose digital I/O pin/Timer_B, capture: CCI2A/B input, compare: Out2 output
P4.3/TB3†	39	I/O	General-purpose digital I/O pin/Timer_B, capture: CCI3A/B input, compare: Out3 output
P4.4/TB4†	40	I/O	General-purpose digital I/O pin/Timer_B, capture: CCI4A/B input, compare: Out4 output
P4.5/TB5†	41	I/O	General-purpose digital I/O pin/Timer_B, capture: CCI5A/B input, compare: Out5 output
P4.6/TB6†	42	I/O	General-purpose digital I/O pin/Timer_B, capture: CCI6A input, compare: Out6 output
P4.7/TBCLK	43	I/O	General-purpose digital I/O pin/Timer_B, clock signal TBCLK input
P5.0/STE1†	44	I/O	General-purpose digital I/O pin/slave transmit enable – USART1/SPI mode
P5.1/SIMO1†	45	I/O	General-purpose digital I/O pin/slave in/master out of USART1/SPI mode
P5.2/SOMI1†	46	I/O	General-purpose digital I/O pin/slave out/master in of USART1/SPI mode
P5.3/UCLK1†	47	I/O	General-purpose digital I/O pin/external clock input – USART1/UART or SPI mode, clock output – USART1/SPI mode

† 16x, 161x devices only

Step2 整合 Passive Infrared Motion Sensor 至 Octopus II



上圖為 Passive Infrared Motion Sensor 與 Octopus II 連接方式，由 Part1-Step2 的 Passive Infrared Motion Sensor 介紹，Passive Infrared Motion Sensor 的工作電壓以及驅動電壓為 3.3V 或 5V 直流電，因此我們必須於 Passive Infrared Motion Sensor 三個接腳中的電源輸入(標示為+3.3V)並將 GND 腳位接地(表示於圖中黑線)，而其感測結果經由 High(3V) / Low(0V)的電位輸入於 GPIO 腳位，而此輸出訊號必須利用 Octopus II 的 GPIO 腳位接收，我們選擇 Octopus II 上的 P1.1 (No.11) GPIO 腳位作為接收 Passive Infrared Motion Sensor 之感測結果。

當 Octopus II 收到 P1.1 為高電位時將會在 P2.0 輸出高電位，由於 OctopusII 腳位尚未能推動繼電器，因此需要加裝一顆電晶體作為放大之用。繼電器的部分請參照上圖所示，接上日光燈和 AC 110V。(切記繼電器接 AC110V 時請小心，接錯的結果可能會造成繼電器燒毀。)

Part IV 設計與製作紅外線感應自動照明

Step1 編譯 Octopus II 程式

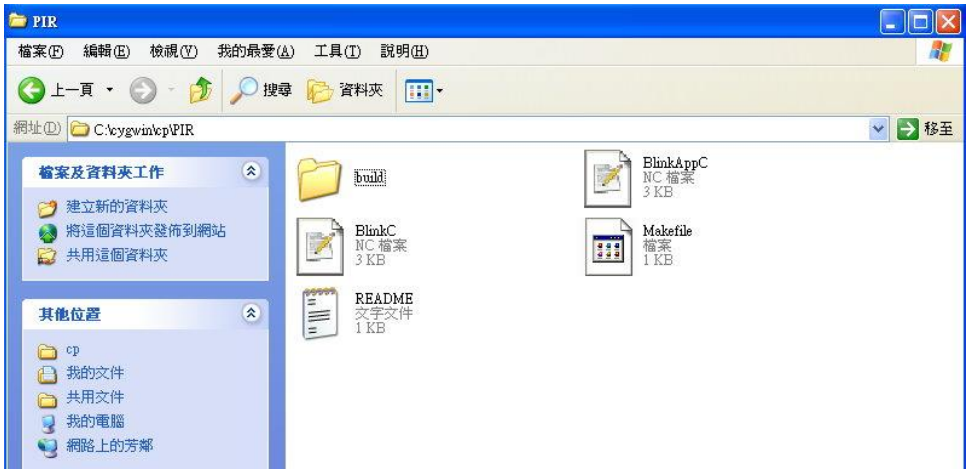
1.在 cygwin 命令列之下開啟 PIR 專案，此專案原始碼於 PartV 附錄程式碼中。

```
/cp/PIR

Administrator@ya319101-bd39fa ~
$ cd /cp/PIR

Administrator@ya319101-bd39fa /cp/PIR
$ ls
BlinkAppC.nc  BlinkC.nc  Makefile  README.txt  build

Administrator@ya319101-bd39fa /cp/PIR
$
```

2.輸入 make octopus，編譯 Octopus II 程式

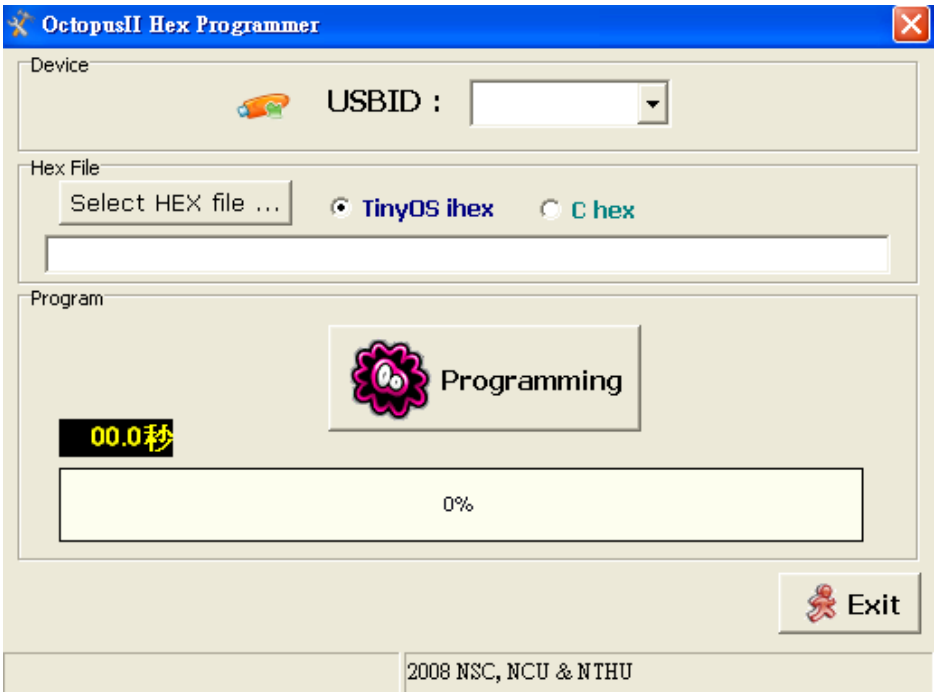
```
Administrator@ya319101-bd39fa /cp/PIR
$ make octopus
mkdir -p build/octopus
    compiling BlinkAppC to a octopus binary
ncc -o build/octopus/main.exe  -Os -O -mdisable-hwmul -Wall -Wshadow -Wnesc-all
-target=octopus -fnesc-cfile=build/octopus/app.c -board= -DDEFINED_TOS_AM_GROUP=
0x22 -DIDENT_APPNAME=\"BlinkAppC\" -DIDENT_USERNAME=\"Administrator\" -DIDENT_HO
STNAME=\"ya319101-bd39fa\" -DIDENT_USERHASH=0x1ff398c6L -DIDENT_TIMESTAMP=0x4e80
3577L -DIDENT_UIDHASH=0x03ebb906L  BlinkAppC.nc -lm
    compiled BlinkAppC to build/octopus/main.exe
            2718 bytes in ROM
              47 bytes in RAM
msp430-objcopy --output-target=ihex build/octopus/main.exe build/octopus/main.ih
ex
    writing TOS image

Administrator@ya319101-bd39fa /cp/PIR
$
```

3.編譯完成後，專案底下會生成 build 資料夾

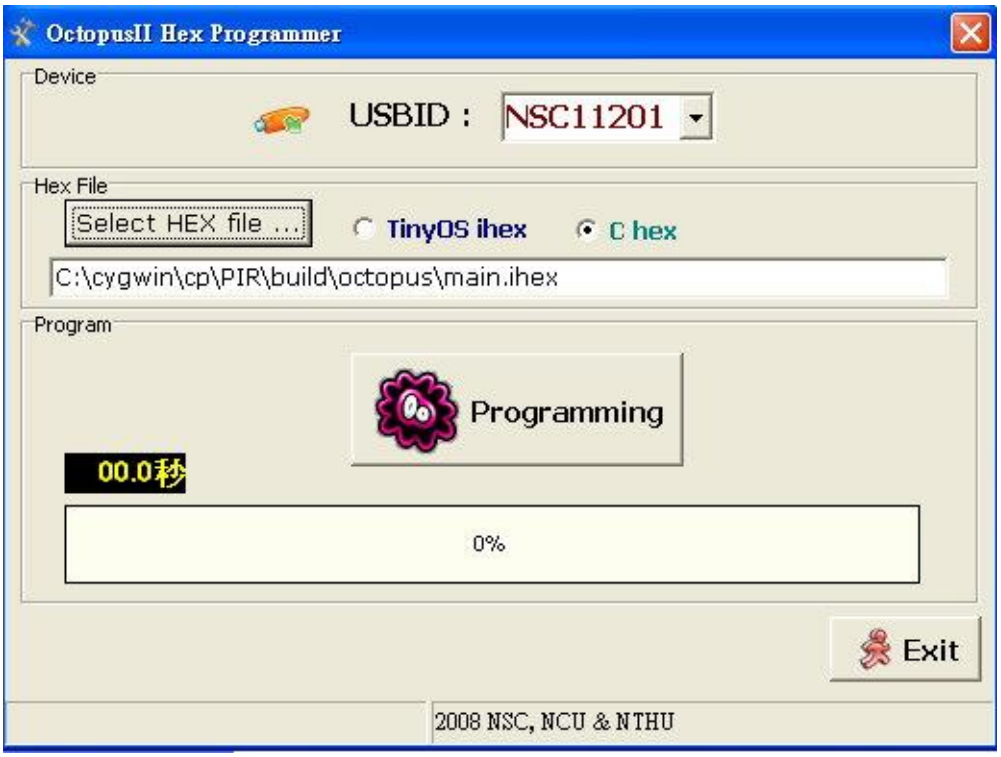


Step2 燒錄 Octopus II 程式

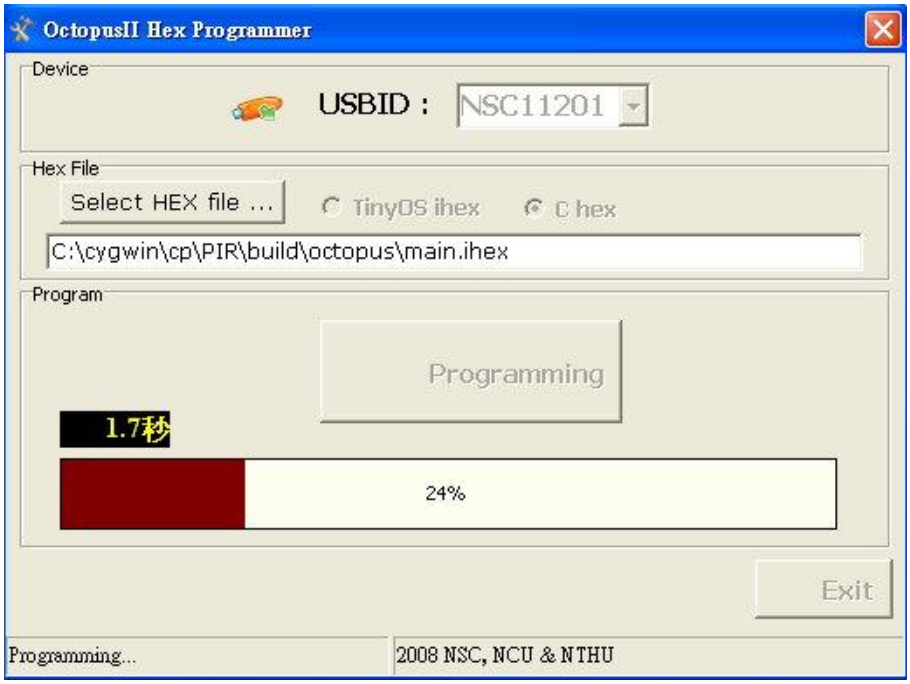
1. 開啟 Octopus II Hex Programmer 燒錄程式



2. 選擇 C:\cygwin\cp\PIR\build\octopus 底下的 main.ihex 檔案，並設定燒錄檔案為 C hex



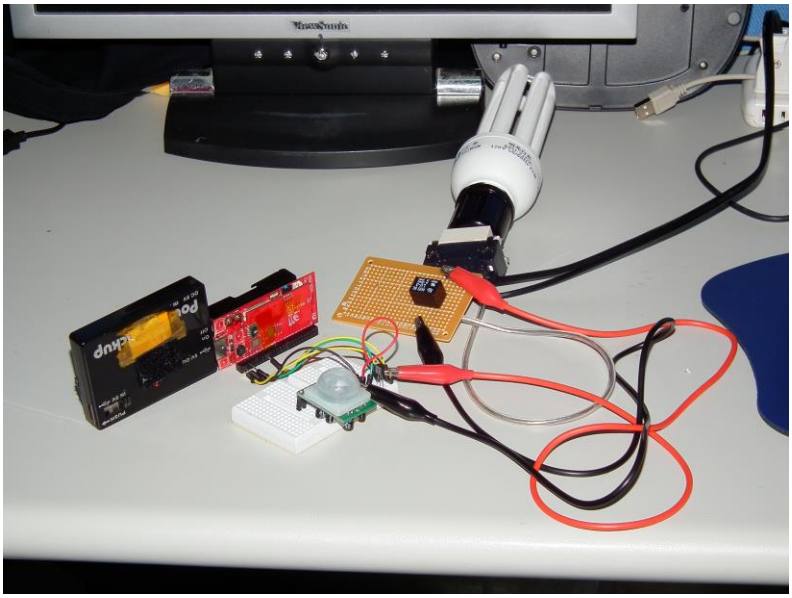
3. 按下 Programming 即可燒錄程式至 Octopus II



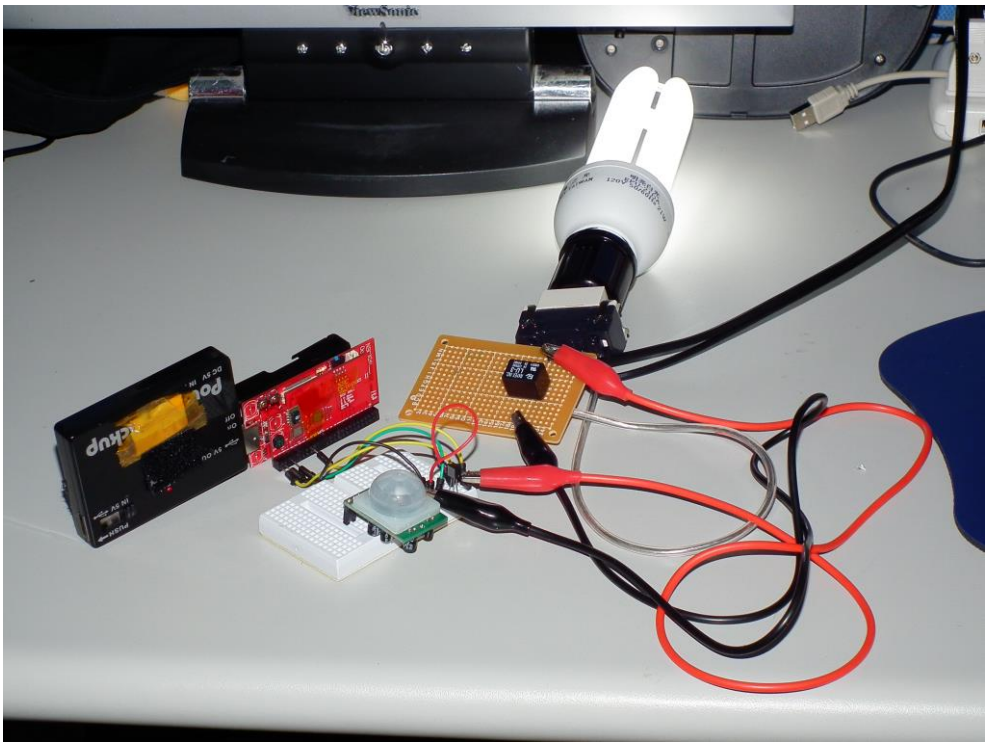
Step3 測試與展示系統

本實驗實際展示功能為利用 Passive Infraresd Motion Sensor 紅外線物體移動感測器偵測是否有物體移動，當有物體移動時，Passive Infraresd Motion Sensor 紅外線物體移動感測器將觸發 Octopus II，而 Octopus II 中的韌體將對 Octopus II 上的 P2.0 下達高電位(HIGH)之狀態命令。

當 Passive Infraresd Motion Sensor 紅外線物體移動感測器周圍沒有物體移動 5 秒時，OctopusII 上的 P2.0 為低電位(LOW)狀態，如下圖所示。



而若 Passive Infraresd Motion Sensor 偵測到其感測範圍內有物體移動時，將改變其輸出至 Octopus II，OctopusII 即下達命令使 P2.0 狀態改變，LED 將由 OFF(不亮)改變為 ON(亮)，如下圖所示。



Part V –附錄程式碼

在開始撰寫程式之前，我們應先思考本實驗的整體架構，本實驗是以 Octopus II 作為主要元件，利用 GPIO 通訊介面接收 Passive Infraresd Motion Sensor 感測器的事件通知，若接收到 Passive Infraresd Motion Sensor 感測器的訊號則表示有人於附近移動，此時 Octopus II 將改變 P2.0 高低電位的狀態，藉此表示偵測到有人移動。

其 Octopus II 可分成兩個程式：BlinkC.nc 以及 BlinkAppC.nc，最後撰寫 Makefile 以便進行編譯，其中 BlinkC.nc 主要功能在於設計主要程式之邏輯，BlinkAppC.nc 主要功能在於連結上層(Blink.nc)所使用到的介面(*interface*)與底層提供此介面的元件(*component*)，最後，Makefile 主要功能在於定義主要元件(*top-level configuration*)的程式進入點，以下我們將分別介紹這三個部分的程式。

PIR\BlinkC.nc

在這部分中，我們主要要達到的目的為撰寫程式之主要邏輯，我們必須列出會使用到的功能與相對應之介面，以便設計程式，以下列出本範例實驗所需要的 *interface*：

	Interface	*Function*
1	*Boot*	用途為啟動 Octopus II
2	*Leds*	當偵測到物體移動時，閃爍 Led
3	*Timer*	當偵測到物體移動時，倒數 *n* 秒後執行對應程式
4	*HplMsp430GeneralIO*	MSP430 提供 GPIO 的介面

宣告需要使用的 *interface* 程式碼如下：

```
1.  #include "Timer.h"
2.
3.  module BlinkC @safe()
4.  {
5.      uses interface Timer<TMilli> as Timer0; //宣告 use Timer interface
6.      uses interface HplMsp430GeneralIO as pin11;// 宣告use HplMsp430GeneralIO interface
7.      uses interface HplMsp430GeneralIO as pin20;
8.      uses interface Leds; // 宣告use Leds interface
9.      uses interface Boot; // 宣告use Boot interface
10. }
```

我們需將對應 *interface* 之 *event* 全部列出，以便程式撰寫，下表為 *interface* 對應之 *event*

	Interface	*event*
1	*Boot*	**event** void Boot.booted()
2	*Timer*	**event** void Timer0.fired()

implementation 內的程式碼如下：

```
1.  implementation
2.  {
3.  int counter=0;// 計數用
4.     event void Boot.booted()
5.     {
6.        call Timer0.startPeriodic( 100);// 掃描PIR高低電位
7.        call Timer1.startPeriodic( 1000 );
8.     }
9.     event void Timer0.fired()
10.    {
11.       if(call pin11.get()){// 當PIR輸入為高電位直
12.              call pin20.set();// 使P2.0輸出高電位
13.              call Leds.led1On();// LED綠色亮
14.              counter=0;// 計數歸零
15.              call Leds.led0Toggle();// LED紅閃爍
16.       }
17.    }
18.    event void Timer1.fired()
19.    {
20.       if(counter>=5){// 當counter大於5時(等於5秒後)
21.       call pin20.clr();// P2.0輸出低電位
22.       call Leds.led1Off();// LED綠關閉
23.       counter=0;// 計數歸零
24.       }
25.       else{
26.       counter++;
27.       }
28.       }
29. }
```

C:\cygwin\cp\PIR\BlinkAppC.nc

由於主程式 BlinkC.nc 所使用的 *interface* 是由底層相對應的元件(*component*)，因此我們必須在 BlinkAppC.nc 程式中定義主程式 BlinkC.nc 所使用到的 *interface* 與底層的 *component* 之關聯性。

首先，我們必須列出所有 BlinkC.nc 使用到的 *interface* 與提供此 *interface* 之 *component*，如下表所列：

	元件(*component*)	介面(*Interface*)
1	*MainC*	*Boot*
2	*LedsC*	*Leds*
3	*TimerMilliC*	*Timer*
4	*Msp430InterruptC*	*GpioInterrupt*
5	*HplMsp430GeneralIOC*	*HplMsp430GeneralIO*

宣告提供需要使用的 *interface* 程式碼如下：

```
configuration BlinkAppC
{
}
implementation
{
    components MainC, BlinkC, LedsC;// 宣告使用MainC,BlinkC,LedsC 元件
    components new TimerMilliC() as Timer0;// 宣告使用Timer元件
    components new TimerMilliC() as Timer1;// 宣告使用Timer元件
    components HplMsp430GeneralIOC;//宣告使用GPIO 元件
}
```

接下來我們需開始將 BlinkC.nc 所需要用到的 *interface* 連結到提供此 *interface* 之 *component*，其連結格式如下：

```
使用者元件.介面   ->   提供者元件.介面
```

若雙方的 *interface* 名稱相同，則可以省略 *interface* 名稱，如 BlinkC.Boot -> MainC.Boot，其介面名稱同樣為“Boot”，因此可以直接寫成 BlinkC.Boot -> MainC 或 BlinkC -> MainC.Boot 或 BlinkC -> MainC。

```
    BlinkC -> MainC.Boot;//wire Boot interface
    BlinkC.Timer0 -> Timer0;// wire Timer interface
    BlinkC.Timer1 -> Timer1;// wire Timer interface
    BlinkC.Leds -> LedsC;// wire Led interface
```

```
  BlinkC.pin11->HplMsp430GeneralIOC.Port11;//wire HplMsp430GeneralIOC    interface
  BlinkC.pin20->HplMsp430GeneralIOC.Port20;//wire HplMsp430GeneralIOC    interface
}
```

C:\cygwin\cp\PIR\Makefile

最後的步驟，我們需撰寫 *Makefile* 檔案，以完成主要元件（top-level configuration）的設定，在設定完畢後後即可進行編譯，編譯過程可參考 PartIII-Step1。

Makefile 設定內容如下：

```
COMPONENT=BlinkAppC //設定主元件)
include $(MAKERULES)
```


參考:

[1] **無線電技術月刊 406 期,熱釋電紅外線感應器的應用,黃海年**

http://www.electronictechnology.com/new/mgad.php?sublnk=article&mcontentid=1384&contentid=NDA2XzkvaW5kZXguaHRtbA==