『物聯網概論』實驗手冊

實驗模組名稱: Bluetooth 與 ZigBee 異質網路閘道器

開發教師: 張志勇教授 (cychang@mail.tku.edu.tw)

開發學生: 謝嘉濬 (shoker0511@hotmail.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: 異質網路通訊模組整合

實驗內容關鍵字: Octopus II ,Tiny OS , Microwave motion sensor module , Arduino

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、Arduino 實驗平台以及藍芽無線傳輸模組。手機或電腦透過藍芽無線傳輸模組，下達命令給 Octopus II 嵌入式感測器、Arduino 實驗平台及藍芽無線傳輸模組所組成的物聯網閘道器。本實驗將可讓學生練習 Arduino 和 Octopus II 程式寫作並利用 Arduino 和 Octopus II 輸入與輸出腳位之電路連結與控制，我們利用藍芽無線模組當作物聯網閘道器的輸入，並利用 OctopusII 作為物聯網閘道器的輸出，透過本實驗，同學將可以更加熟悉 Octopus II 嵌入式感測器、Arduino 實驗平台的操作、控制以及如何透過輸出入腳位整合其他模組。

實驗器材

- Arduino 實驗平台
  - 基於 創用CC 開放原始碼的電路圖設計
  - Arduino 可使用 ICSP 線上燒入器，將 Bootloader 燒入新的 IC 晶片。
  - 可依據官方電路圖，簡化 Arduino 模組，完成獨立運作的微處理控制。
  - 可簡單地與感測器，各式各樣的電子元件連接，如 紅外線、超音波、熱敏電阻、光敏電阻、伺服馬達…等。
  - 支援多樣的互動程式，如 Adobe Flash, Max/MSP, VVVV, Pure Data, C, Processing… 等。
  - USB 介面，不需外接電源。另外有提供 9V 直流電源輸入
  - 使用低價格的微處理控制器 (ATMEGA 8–328)

- 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

- 藍牙無線傳輸模組
  - IC為class 1 bluegiga WT11
  - 啟動電壓為1.2V
  - 傳遞距離為100公尺
  - 增加兩倍的資料寫入量，而且有14KB的儲存空間
  - 電源可由USB供電或將5V直流電源連結於Vcc供電
  - 提供另外兩組Analog輸入端，總共八個類比輸入

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 Arduino、Octopus II 4
Step 1 瞭解 Arduino 實驗平台 4
Step2 瞭解 Octopus II 6
Step3 瞭解藍芽無線傳輸模組 6

Part II 環境建置 8
Step1 介紹實驗所需之軟體 8
Step2 安裝 Arduino 9
Step3 安裝 JAVA 10
Step4 安裝 Cygwin1.7 (線上安裝) 16
Step5 安裝 TinyOS 2.1 26

Part III 硬體電路整合 33
Step1 整合藍芽無線傳輸模組至 Arduino 33
Step2 整合 OctopusII 至 Arduino 34

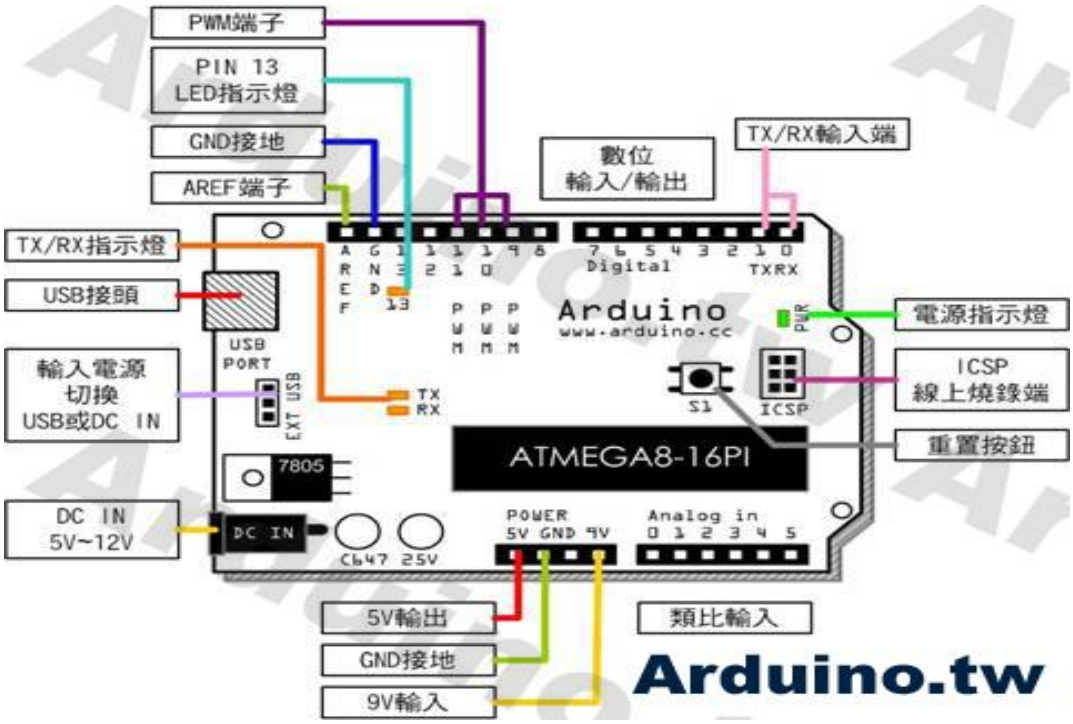
Part IV 設計與製作物聯網閘道器 35
Step1 編譯 Octopus II 程式 35
Step2 燒錄 Octopus II 程式 36
Step3 測試與展示系統 37

Part V –附錄程式碼 38

Part I 簡介 Arduino、Octopus II

Step1 瞭解 Arduino 實驗平台

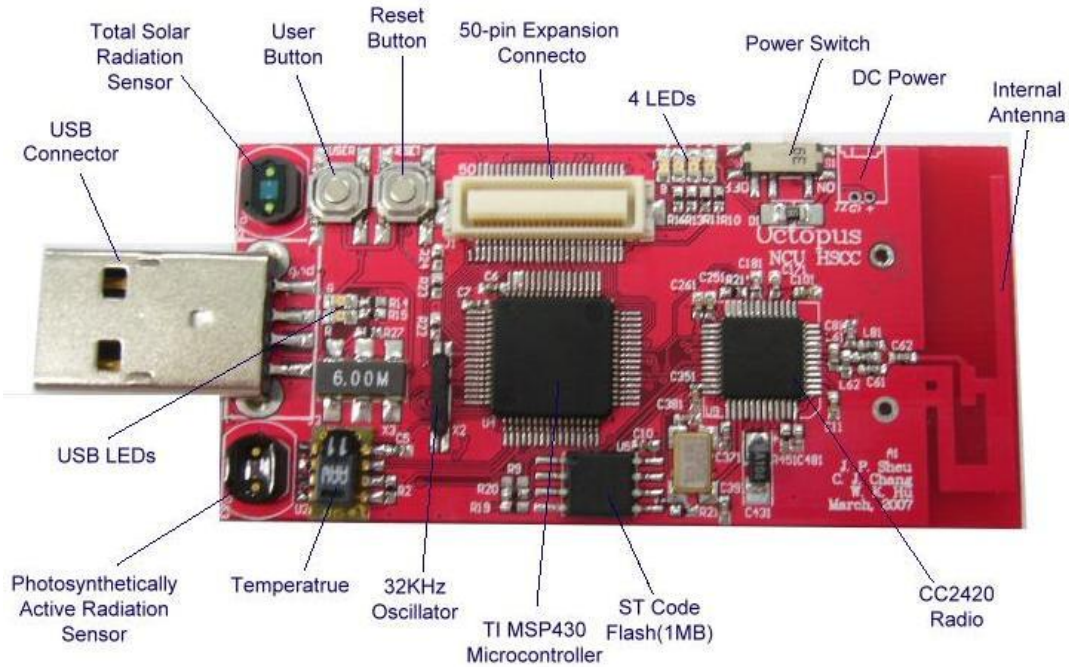
Arduino是由義大利高科技設計學校的老師和西班牙籍晶片工程師所共同研發的實驗平台，Arduino可以獨立運作成為一個可以跟軟體溝通的介面，Arduino採用Atmel Atmega8/168/328 單晶片，參照AVR-GCC編譯器與avr-libc, AVRlib函式庫與Processing程式語言為基礎 ，並搭載8-bit AVR MCU為中央處理單元，具有1 KB的記憶體空間以及16 KB的快閃記憶體容量；其主要訴求讓我們可以快速使用 Arduino 語言與 Macromedia Flash, Processing, Max/MSP, Pure Data, SuperCollider…等軟體，作出互動作品。Arduino可以使用開發完成的電子元件例如Switch或感測器或其他控制器件、LED、步進馬達或其他輸出裝置。
Arduino大小尺寸：寬70mm X 高54mm。



Step2 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。

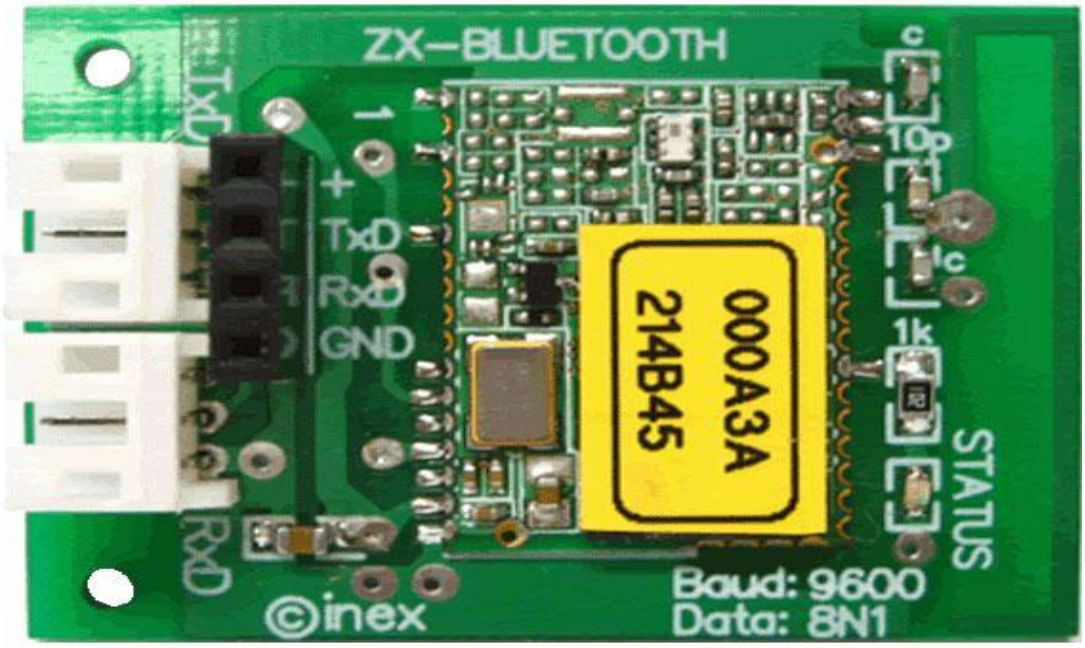
在電力管理的部分，Octopus II 的電力來源主要是兩顆 3 號電池，而 MSP430 處理器提供了 5 種省電模式，因此，使用者可藉由進入省電模式延長無線感測節

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。

Step3 瞭解藍芽無線傳輸模組

藍芽無線傳輸模組主要功能在於利用 9600 bps speeds and 8N1 data transfer 與其他藍芽裝置進行通訊，並且只需要連接 RX, TX 就可以直接傳資料, 不需額外設定程式；使用者, 程式與原來的 Serial 傳輸程式一樣，可以不用修改，且與其他藍芽協定的裝置相容，所以常常應用於其他藍芽裝置進行配對，使用 UART 通訊。



低單價, 適用任何微處理器(包括 BASIC Stamp, Arduino, Atmel, 8051, 等等) 與其他藍牙裝置連線，體積小：3cm x 4cm，標準 TTL 電壓輸入 5V，最大有效通訊距離 30 公尺。

以下為 X-Band Motion Detector 之特色與規格：

- 內建天線
- 最大傳輸距離 30M
- 支援串列傳輸 Serial Port Profile (SPP)
- 9600 bps speeds and 8N1 data transfer
- 只需要連接 RX, TX 就可以直接傳資料, 不需額外設定程式 ; 使用者, 程式與原來的 Serial 傳輸程式一樣, 可以不用修改
- 使用 UART 通訊
- 本身只能當成 Slave bluetooth 通訊
- 可以於其他藍牙協定的裝置相容
- 標準輸入電壓 +5v
- Able to work with Computers with SSP Profile based Bluetooth Devices.
- Compatible with ESD-02 and ESD100 Devices
- 尺寸: 3cm by 4cm

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。

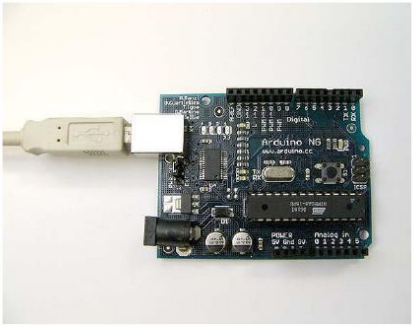
- Cygwin (with TinyOS)
  提供在 Windows 下模擬 UNIX 的 shell 環境以及開發 TinyOS 時需使用的 UNIX 工具(如 perl 和 shell 腳本)。在此模擬環境下，程式開發者將可直接使用 make 指令編譯 TinyOS 程式碼，並可在模擬環境下監控感測器所傳回之資料。

- 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 Arduino

安裝Arduino

- 將 Arduino 接上方頭的USB線，另外一端連接電腦
- 接上之後會出現FTDI的驅動程式安裝畫面



安裝Arduino

- 指定驅動程式安裝目錄，一般下載的Arduino資料夾裡面即會包含
- （EX：C:\Arduino-0022\drivers\FTDI USB Drivers）

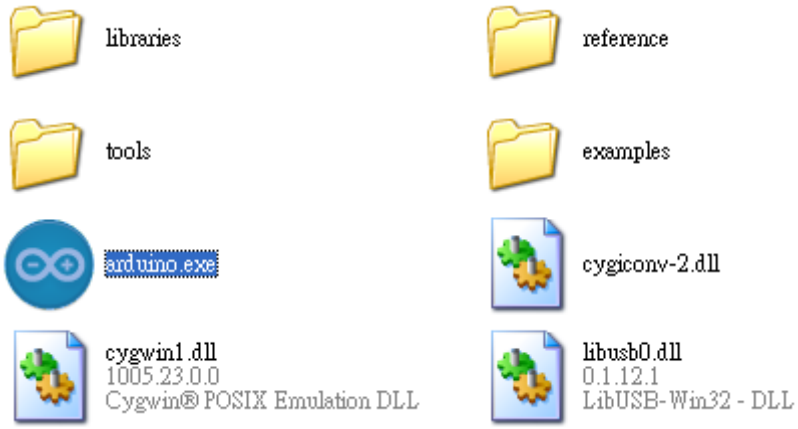
連接好硬體後，可以檢查所連接的 Arduino 目前是位於哪個 COM Port

你可以從控制台 → 系統 → 切換至「硬體」標籤 → 裝置管理員

→ 連接埠 (COM 和 LPT)→ 檢查 USB Serial Port (COM4)，括號中的數字即為

你目前所連接的 Arduino Port

Arduino 程式開發工具

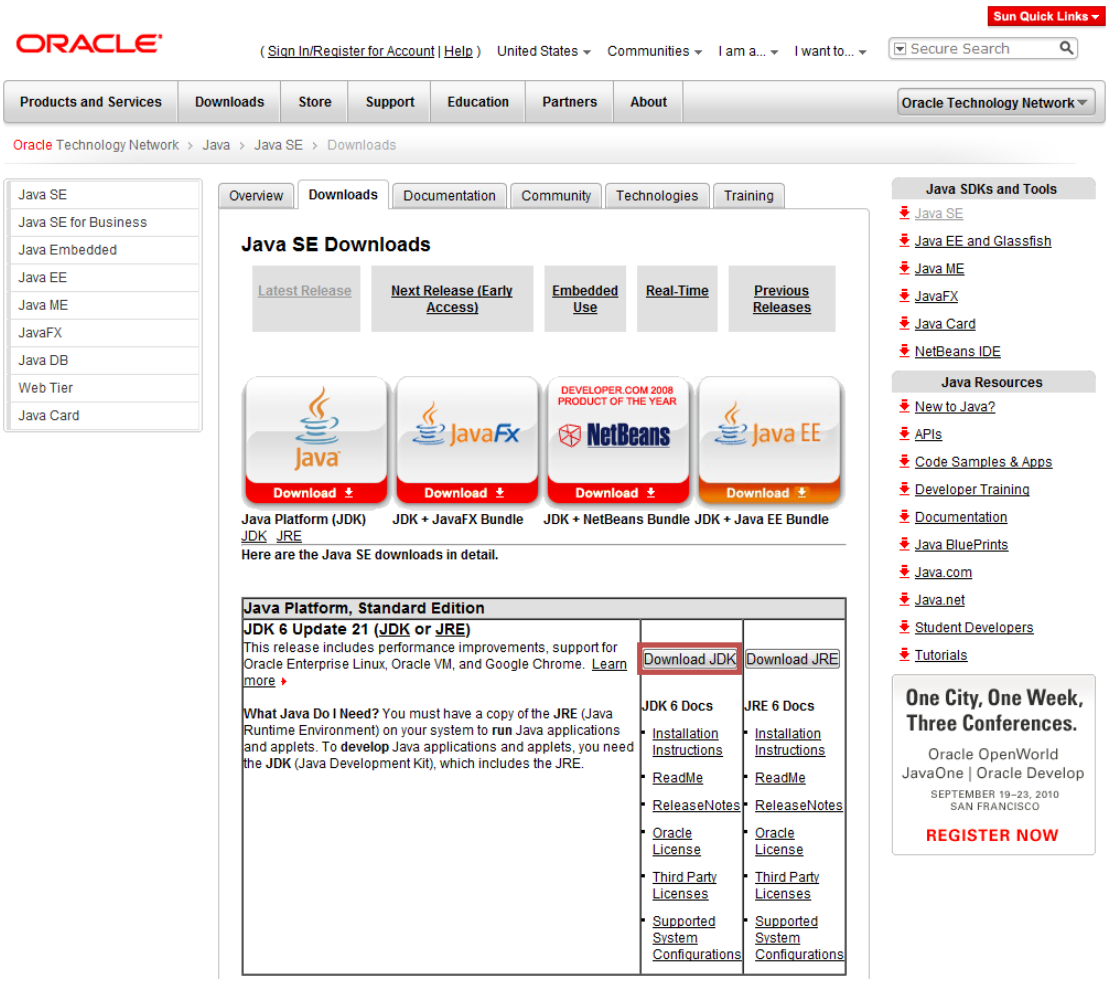


即可進入到 Arduino

Step3 安裝 JAVA

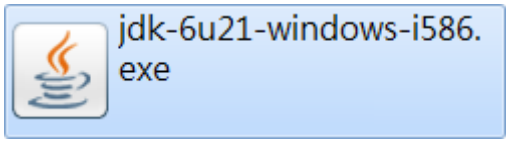
由於 TinyOS2.x 需要有 JAVA VM 環境的支援，因此我們將先介紹 JAVA JDK 之安裝。

- 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

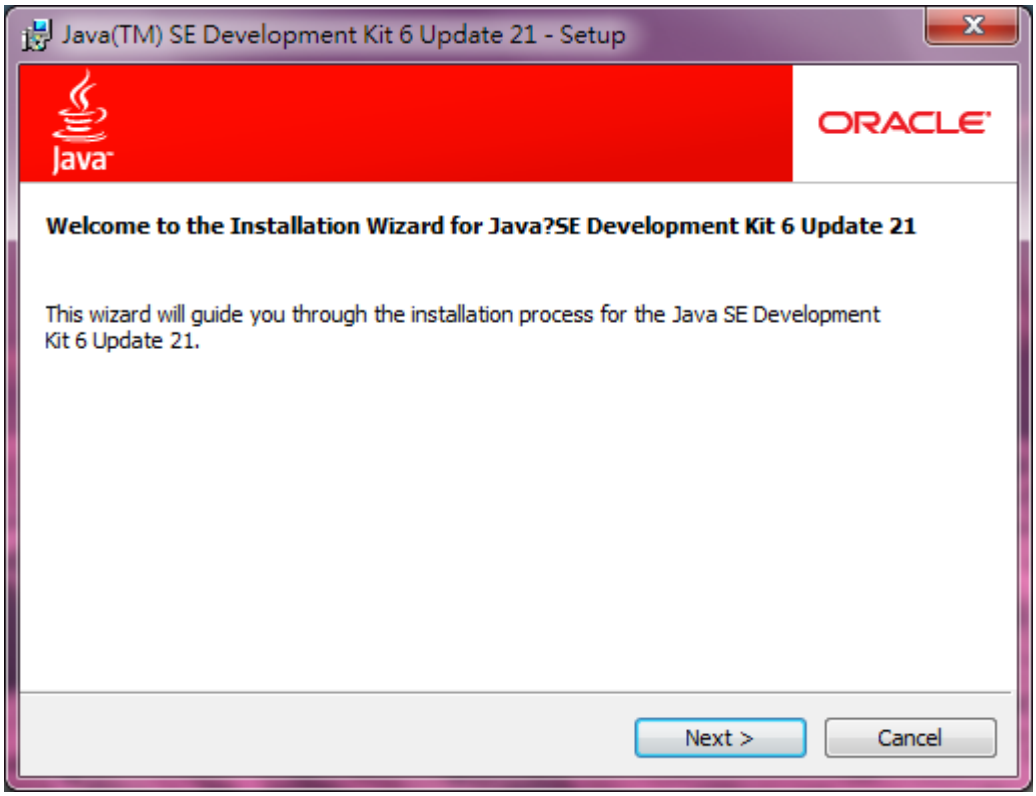


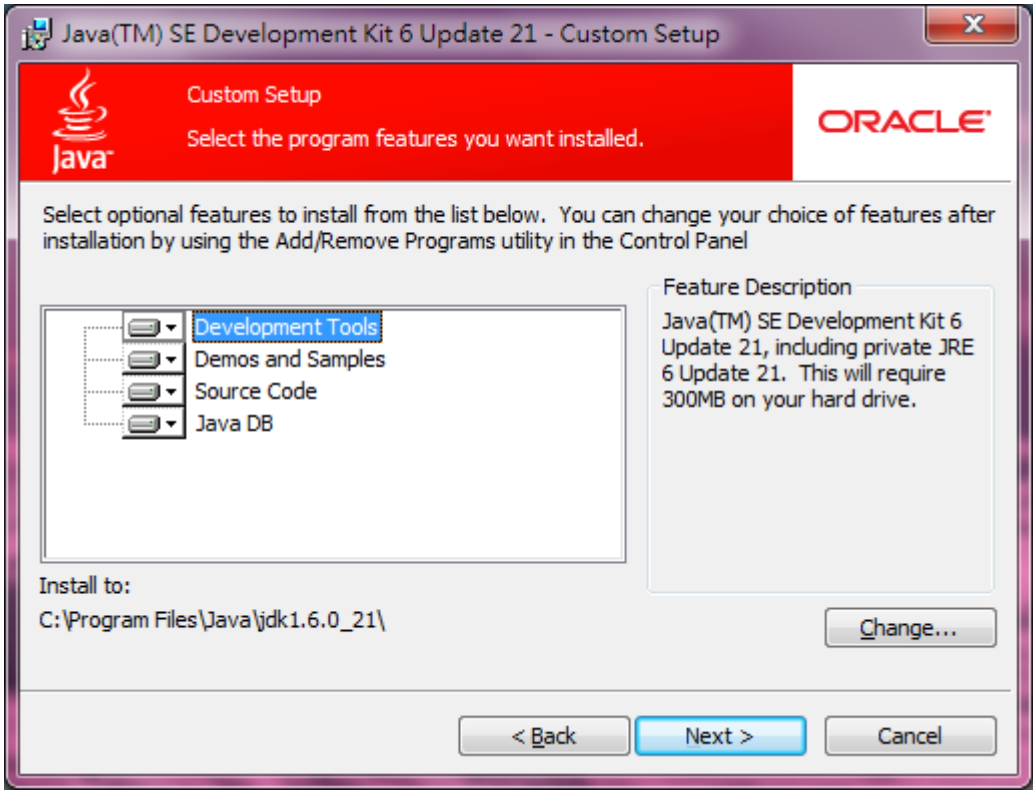
安裝 Java SDK

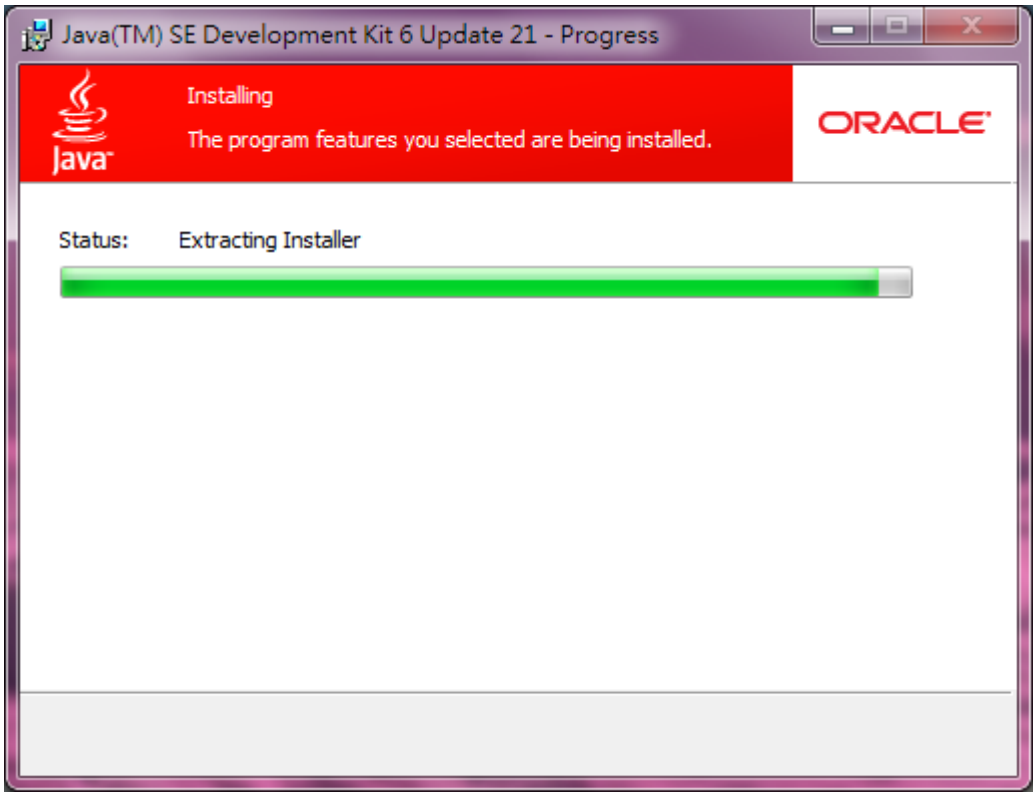
先下載安裝檔並執行：

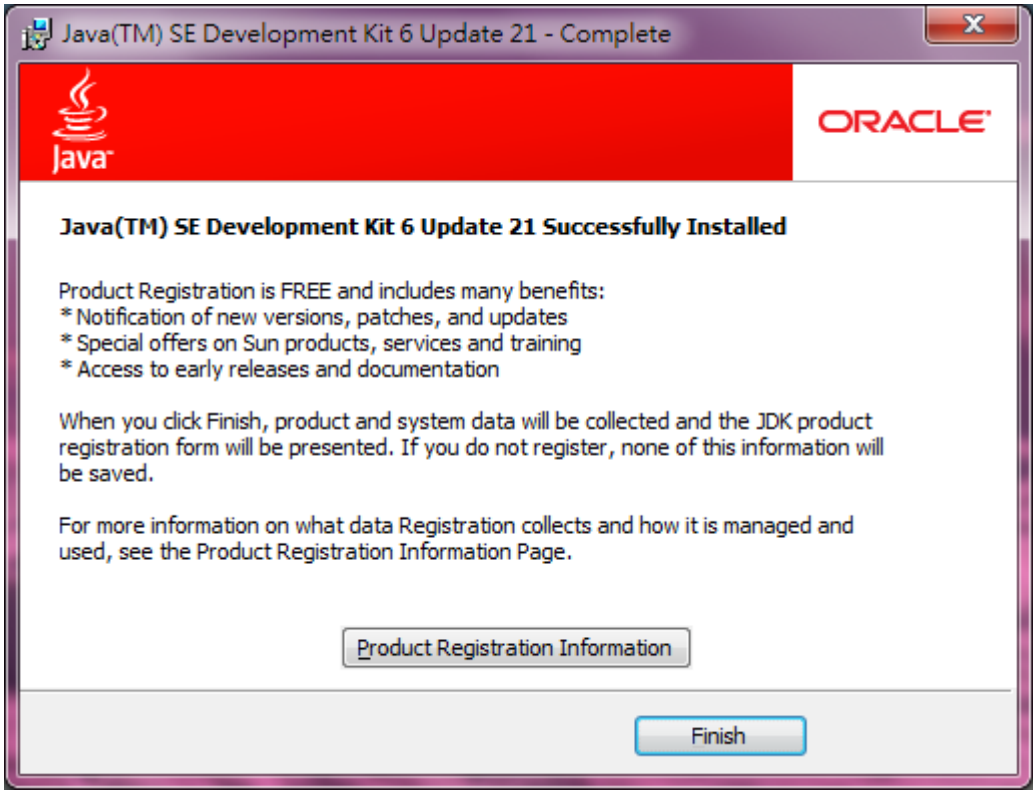


開啟後選擇 Next，順著安裝精靈的指示下一步直到完成安裝。


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

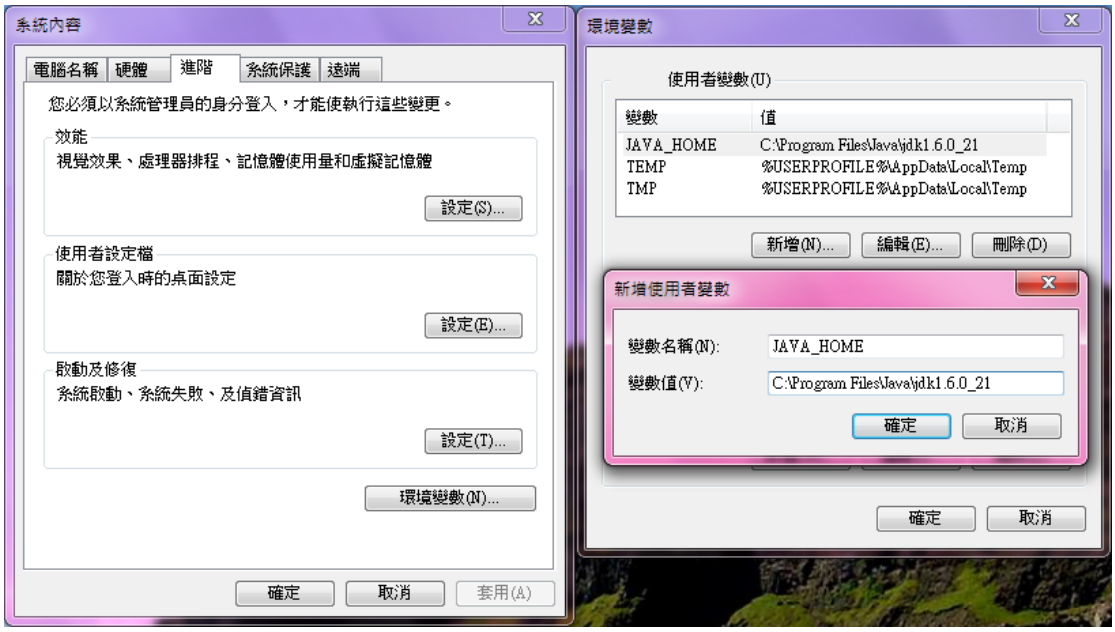

Java(TM) SE Development Kit 6 Update 21 - Custom Setup
Java
Custom Setup
Select the program features you want installed.
ORACLE
Select optional features to install from the list below. You can change your choice of features after installation by using the Add/Remove Programs utility in the Control Panel
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。



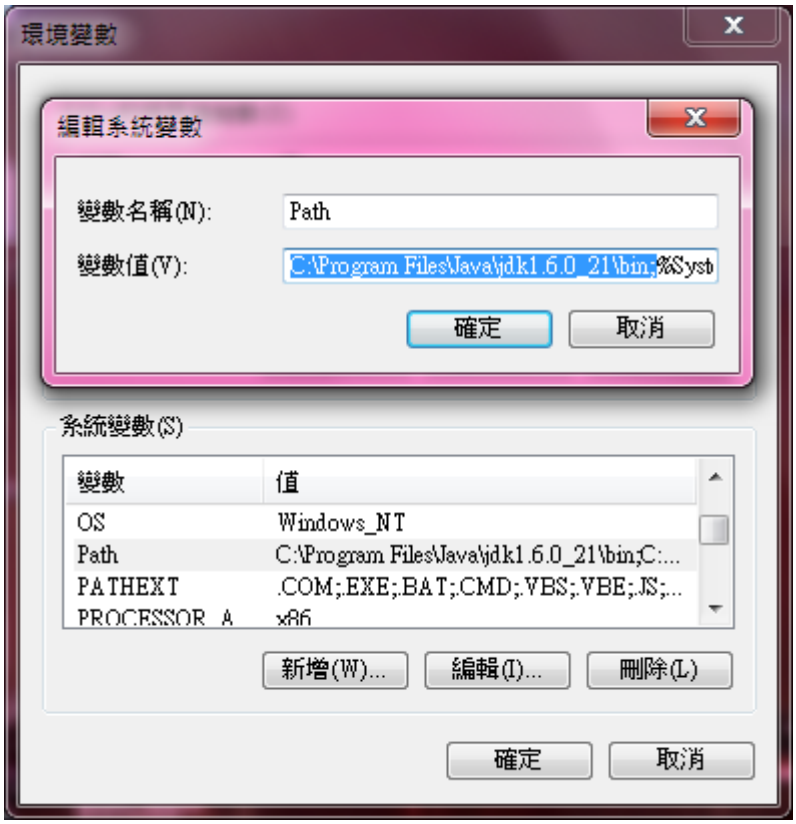
變數名稱：**JAVA_HOME**

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

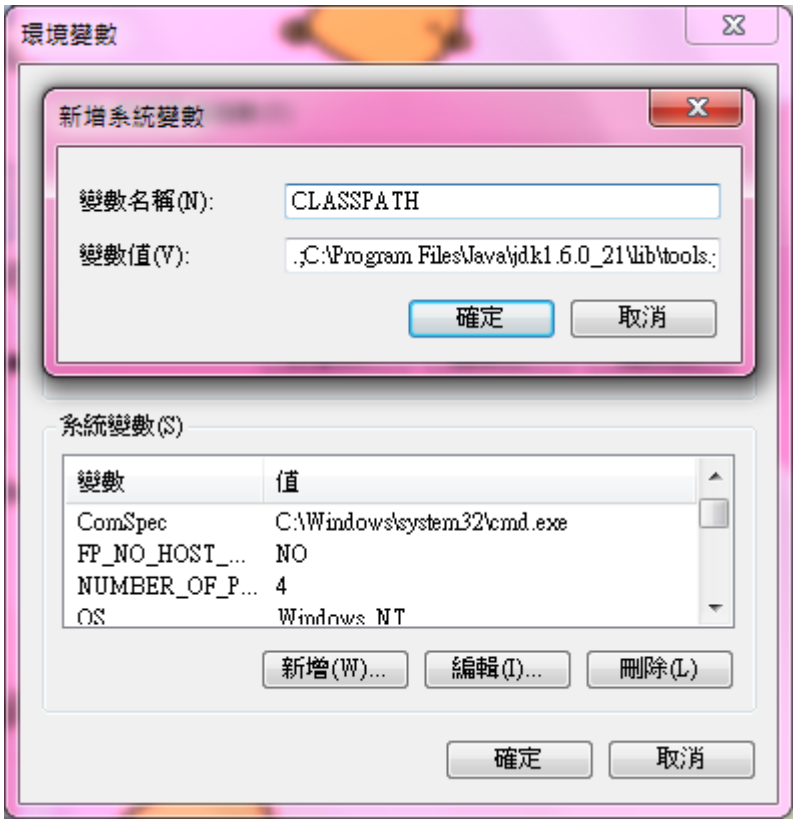
(此為安裝 Java 時選擇之安裝路徑)

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。



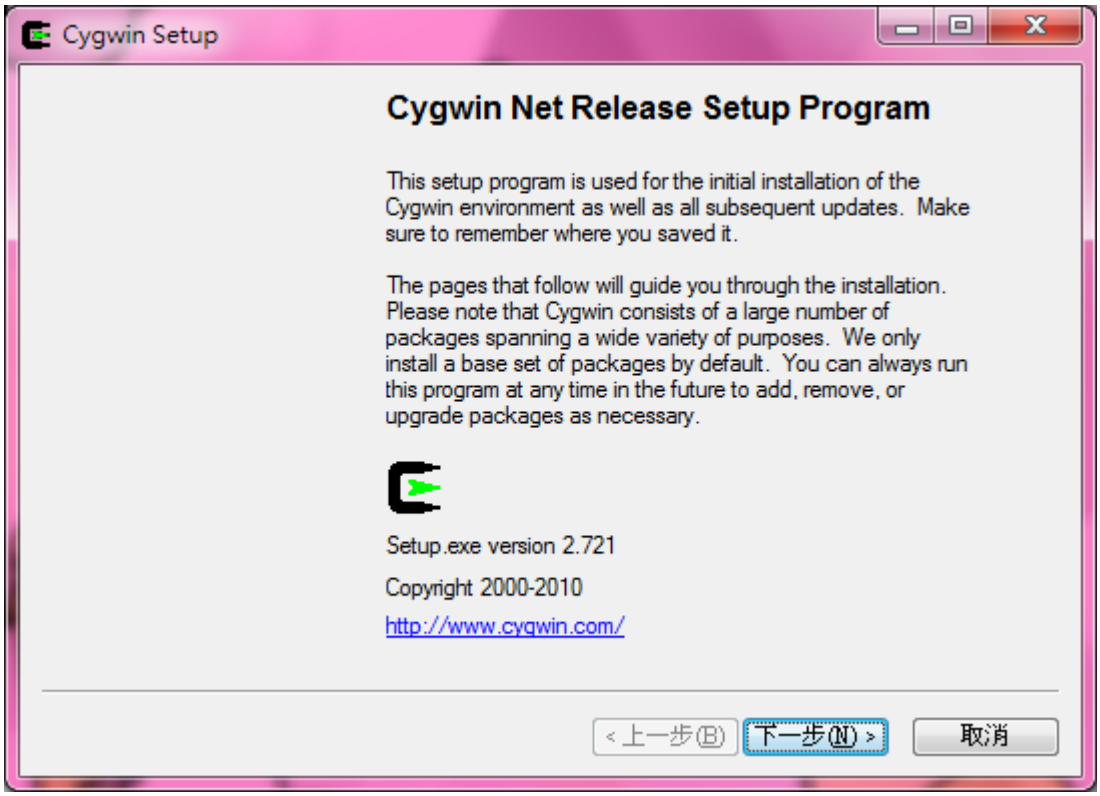
至此 Java SDK 的安裝與設定就告一段落。

Step4 安裝 Cygwin1.7 (線上安裝)

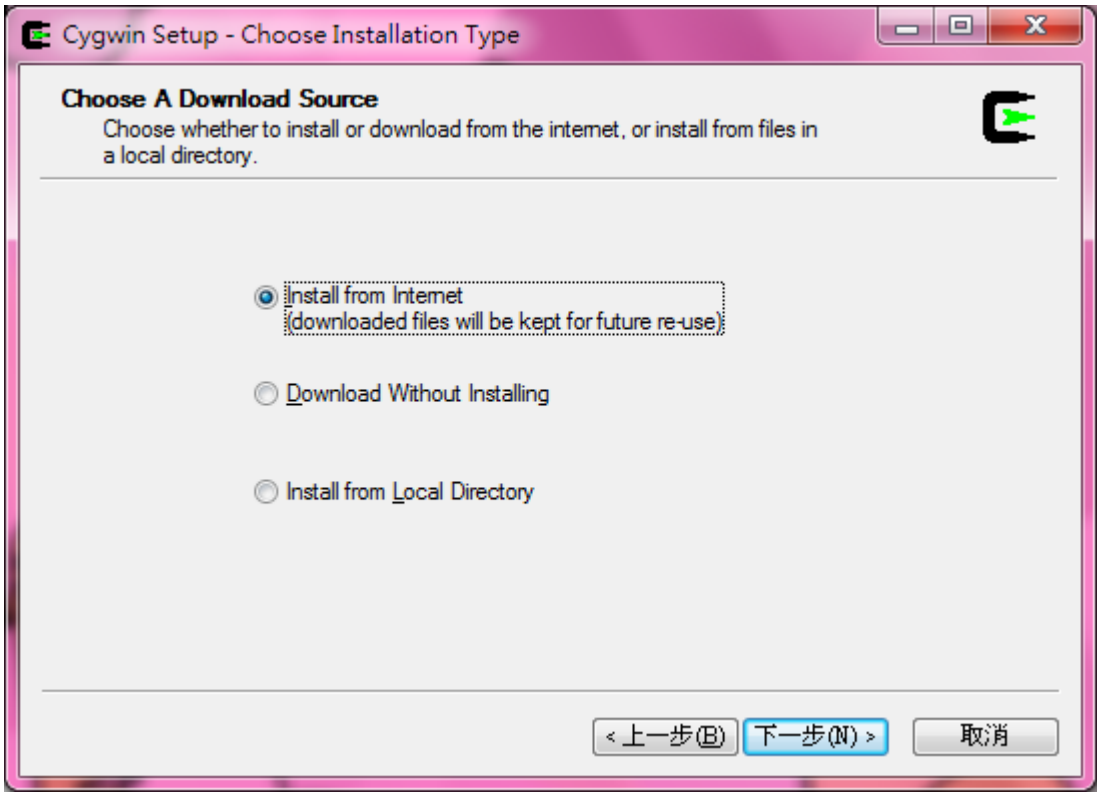
前往 http://www.cygwin.com

執行 **setup.exe**

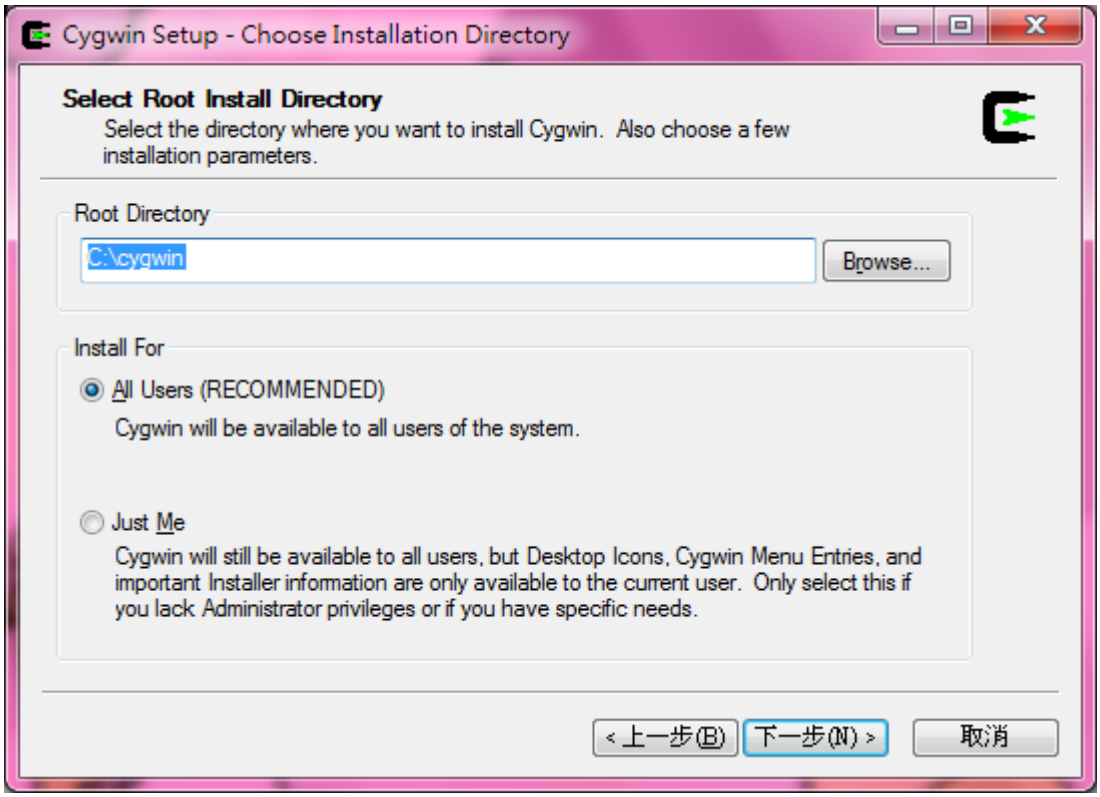
第一個 Setup 畫面直接按下下一步



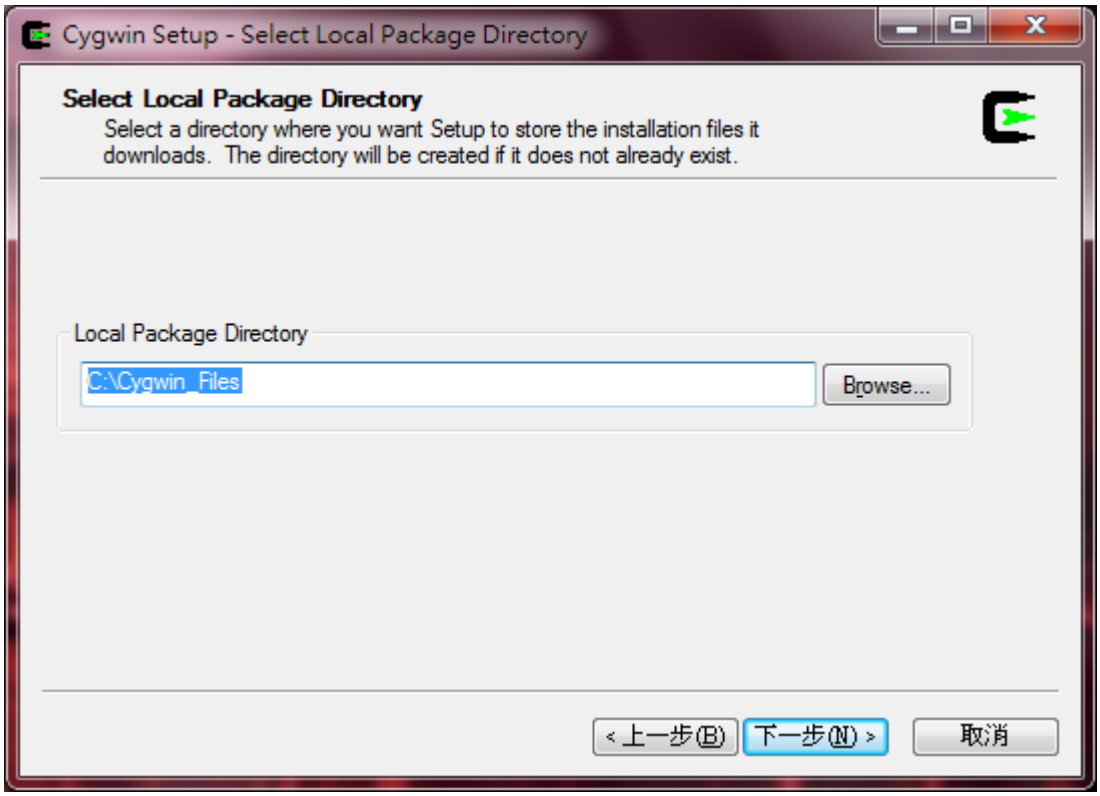
在 **Choose Installation Type** 時，選擇「**Install from Internet**」。



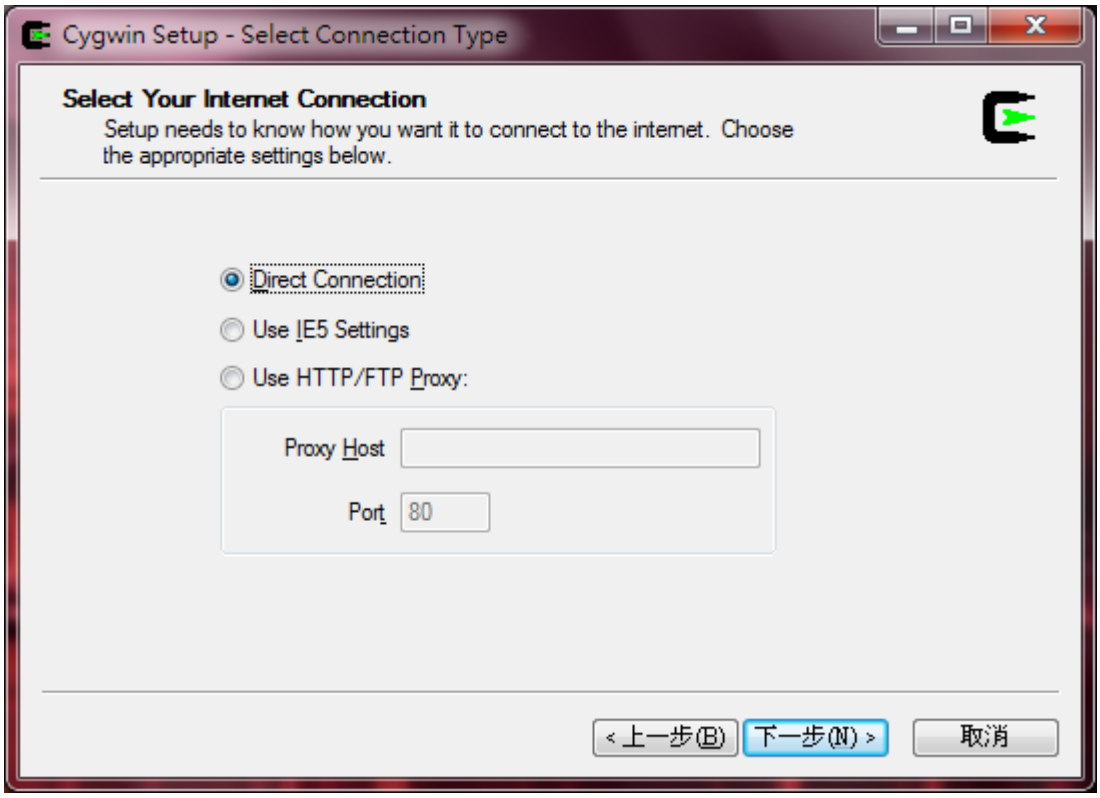
在 **Choose Installation Direction** 時，照預設值直接按下一步。



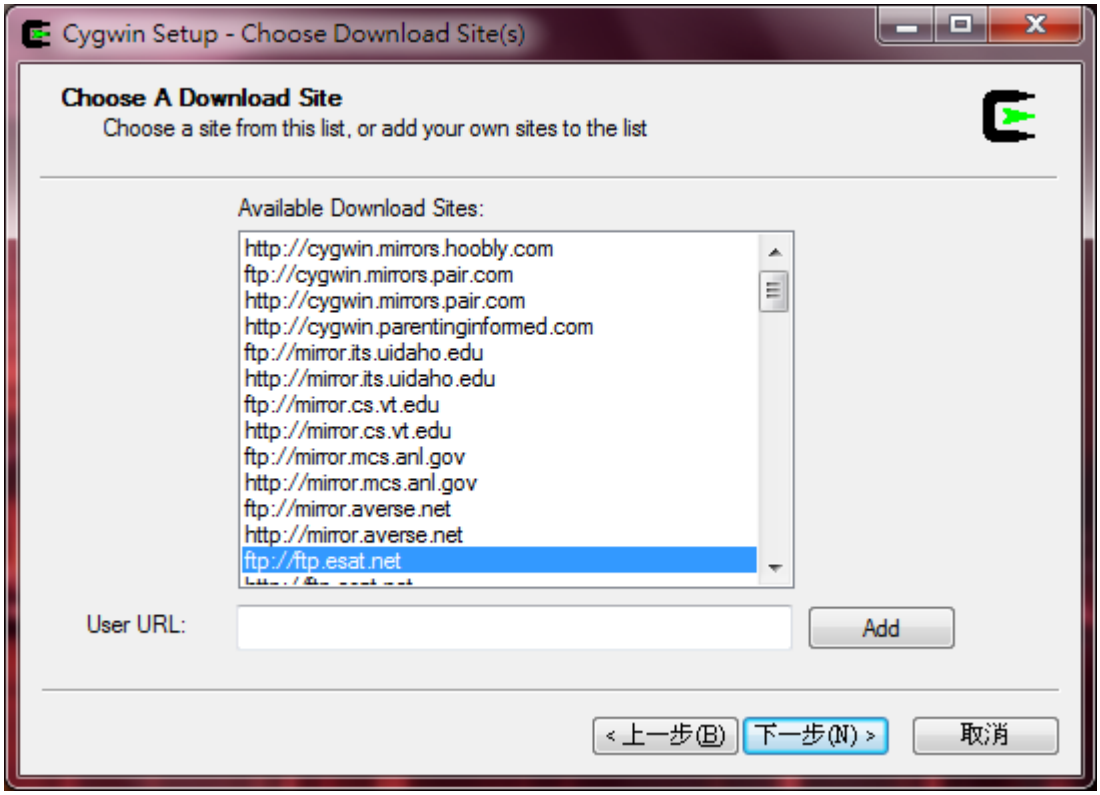
接著 **Select Local Package Direction** 為選擇「**存放由網路下載回來檔案的資料夾**」，設定完之後按下下一步。



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

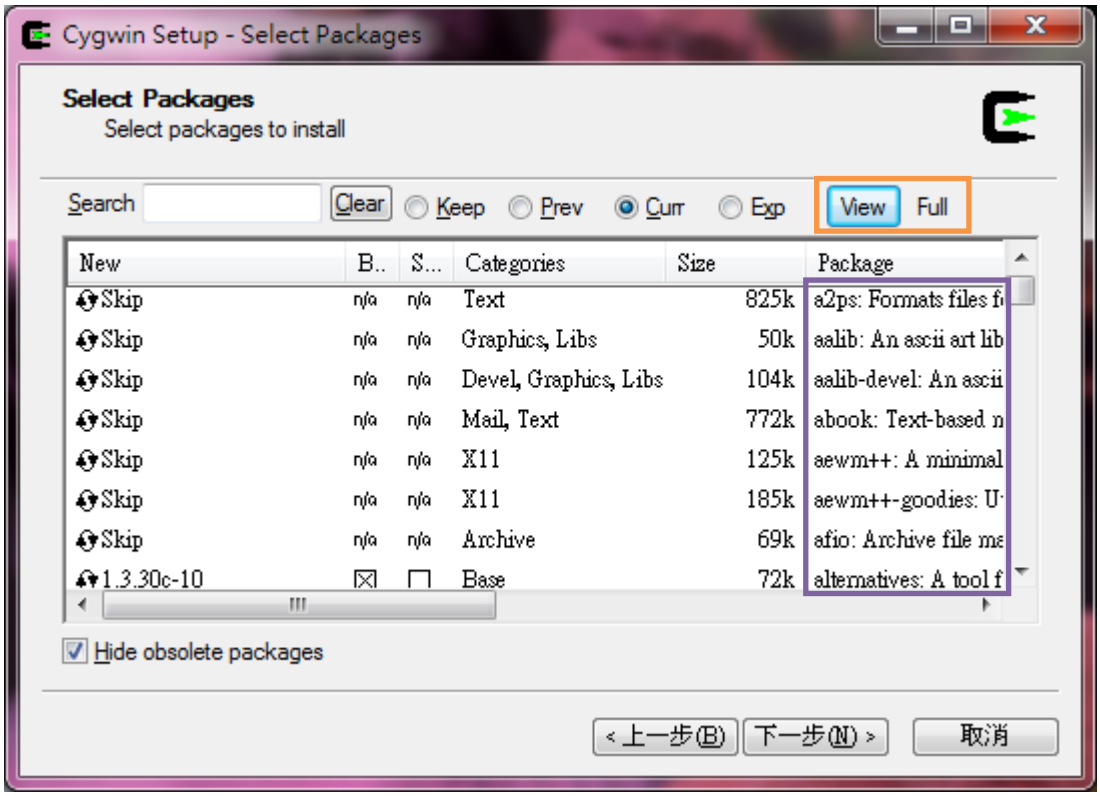


接著選擇下載來源 **Choose Download Sites**。建議選擇離自己較近的網站或台灣的網站（如 NTU 等）。



接著來到 **Select Packages** 部分。

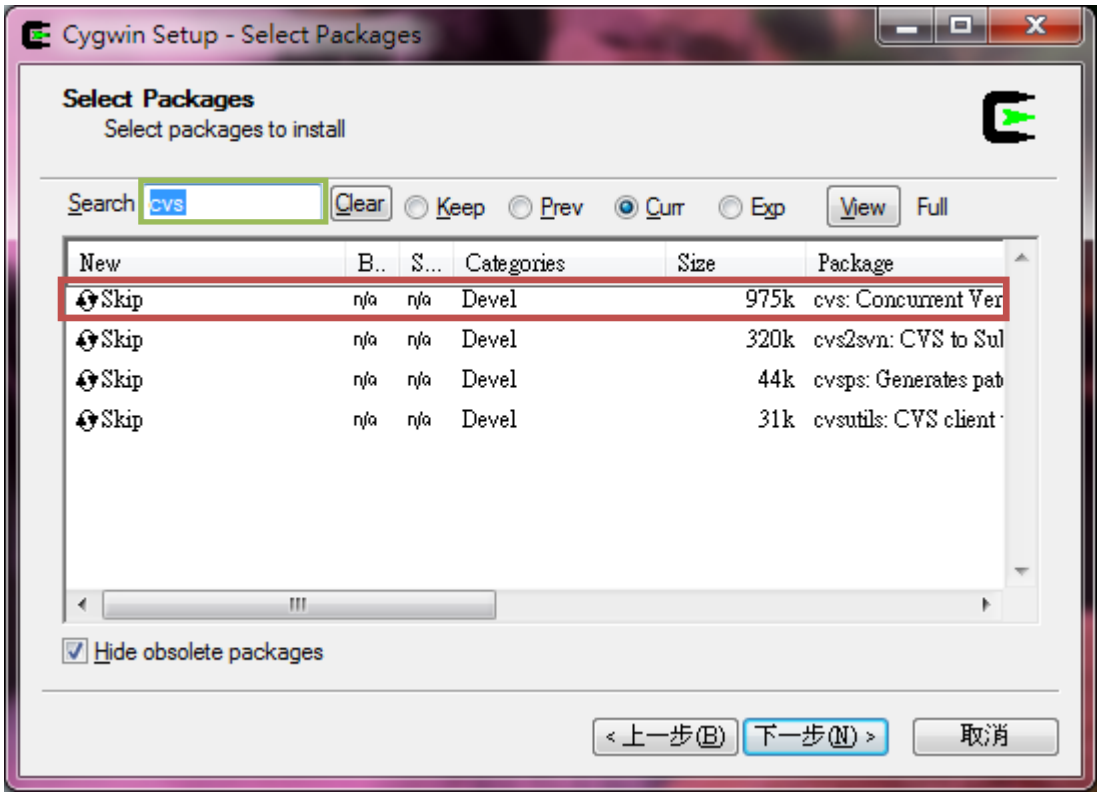
先按下右上角的“**View**”按鈕，使旁邊變成“**full**”，之後 packages 會變成由 a-z 順序列出。



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

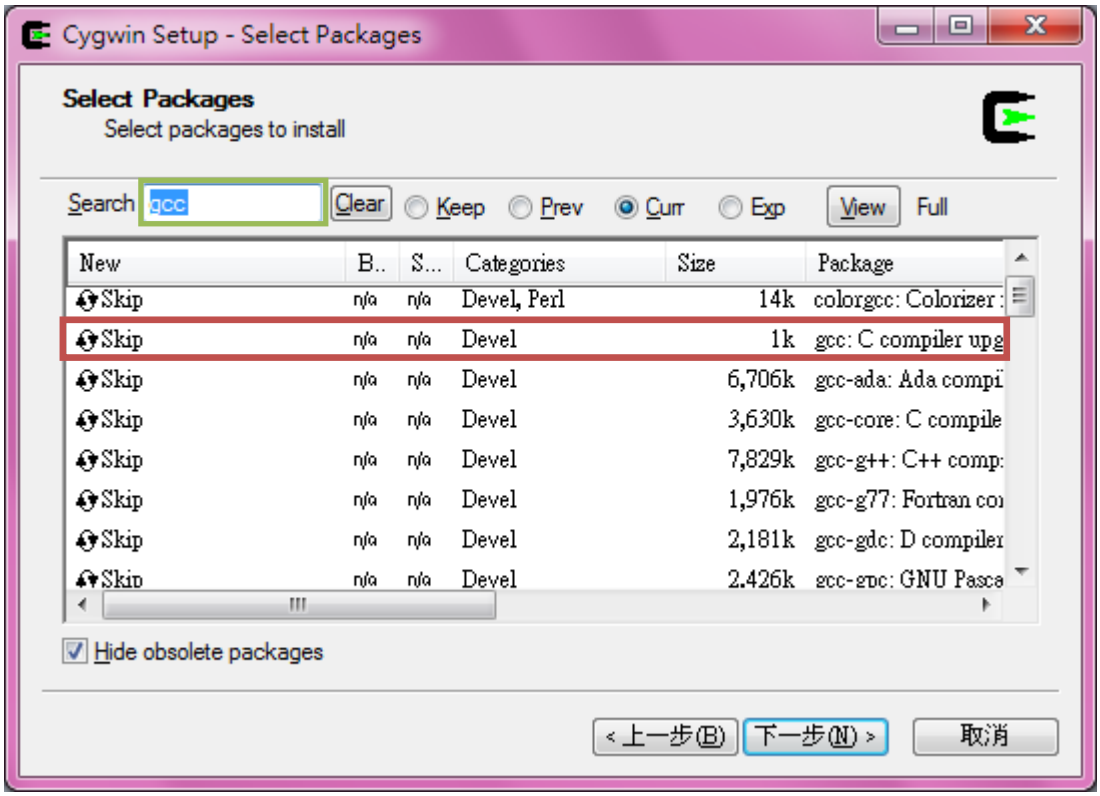
***cvs*：於“Search”欄位中輸入“cvs”，並點選“Skip”圖示進行安裝**

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。



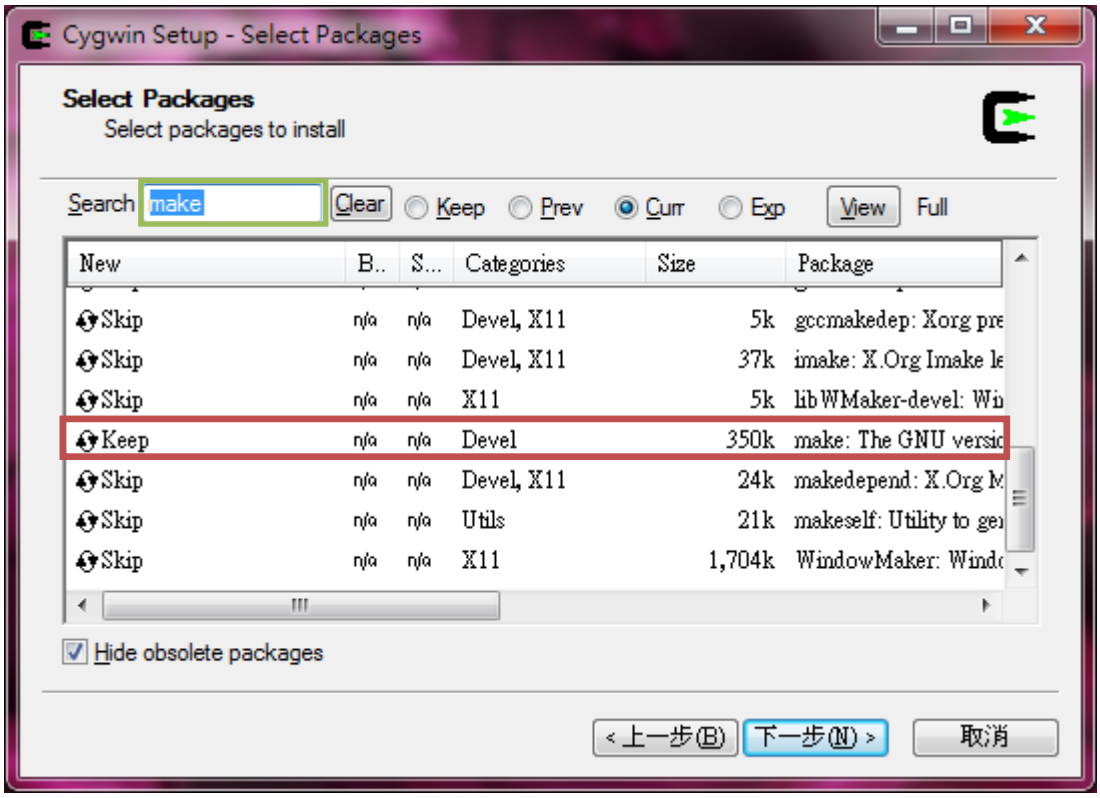
***gcc*：於“Search”欄位中輸入“gcc”，並點選“Skip”圖示進行安裝**

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。



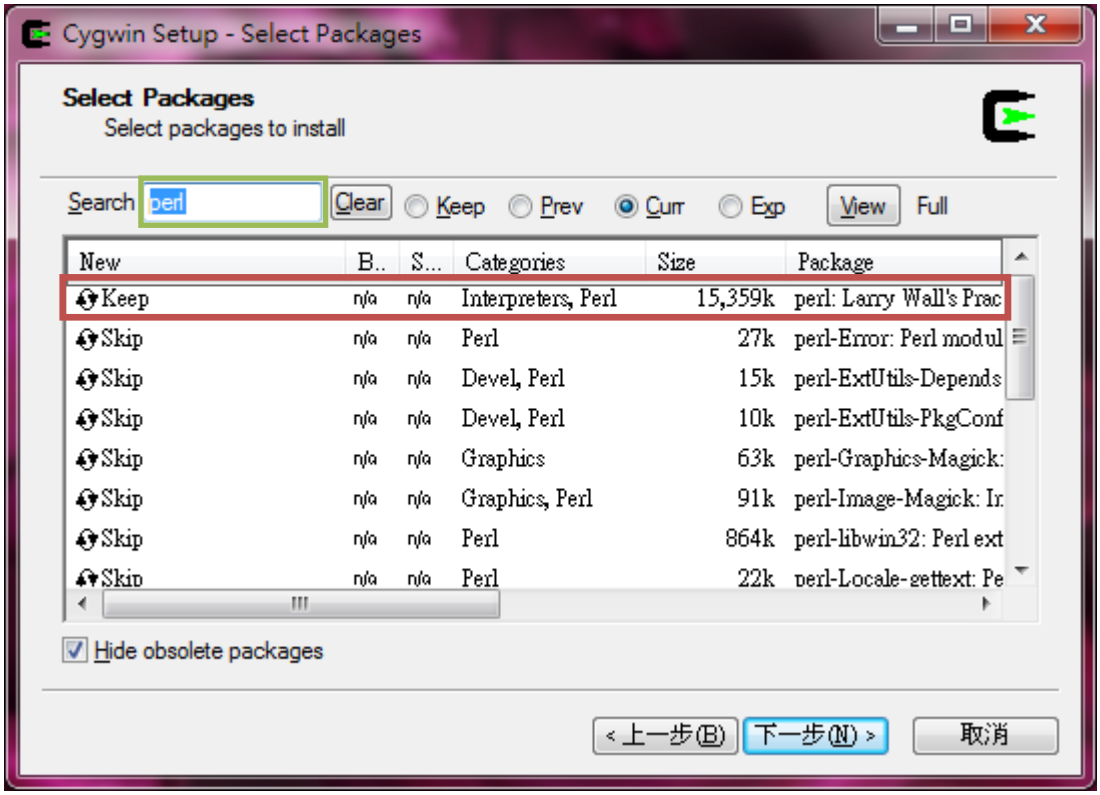
make：**於“Search”欄位中輸入“make”，並點選“Skip”圖示進行安裝**

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。



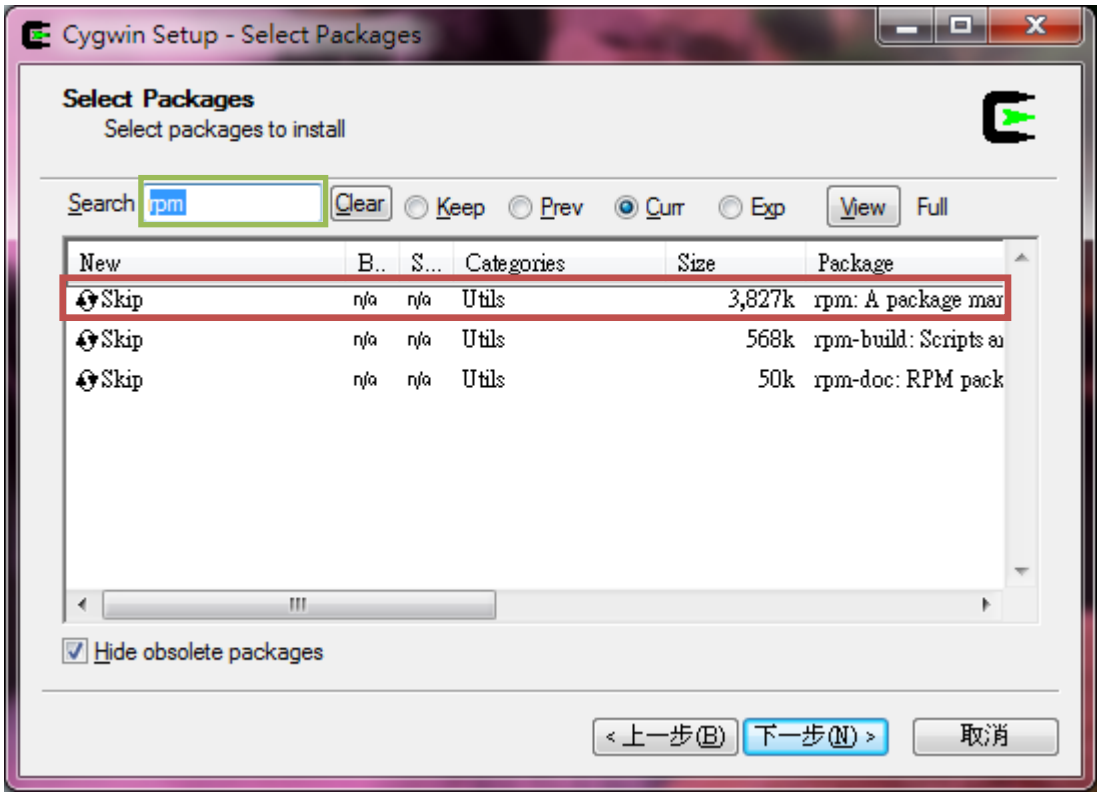
***perl*：於“Search”欄位中輸入“perl”，並點選“Skip”圖示進行安裝**

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。



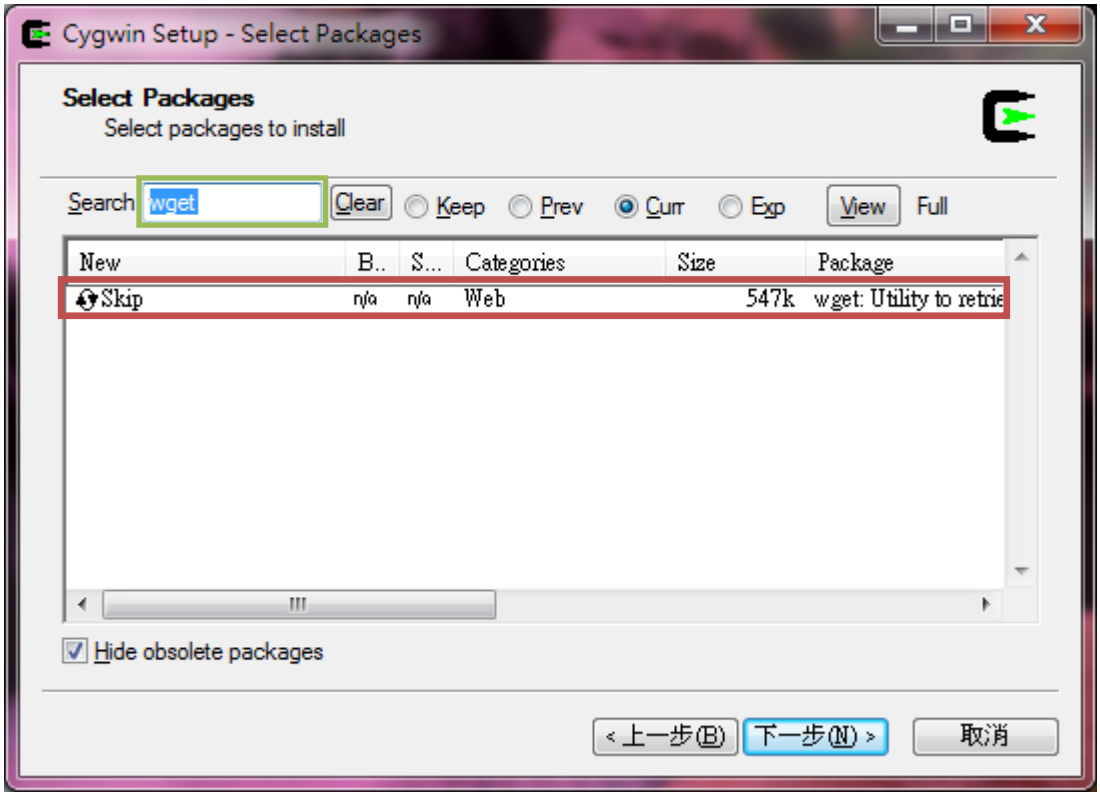
***rpm*：於“Search”欄位中輸入“rpm”，並點選“Skip”圖示進行安裝**

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。

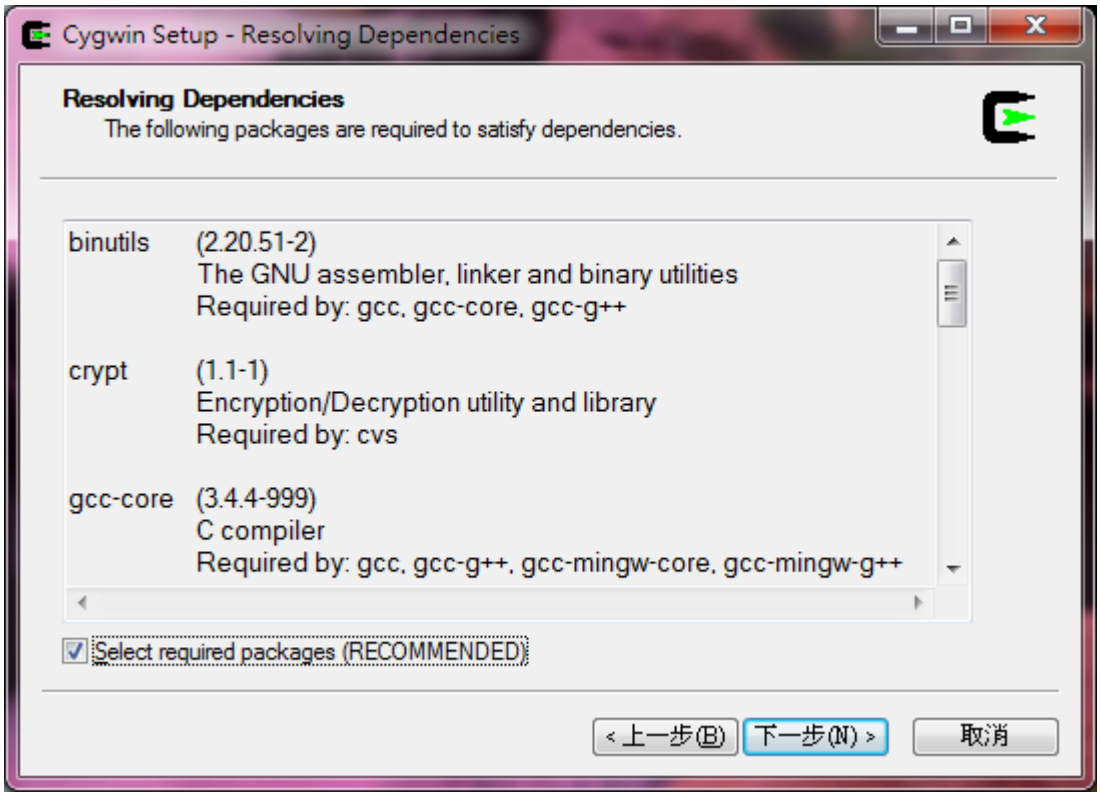


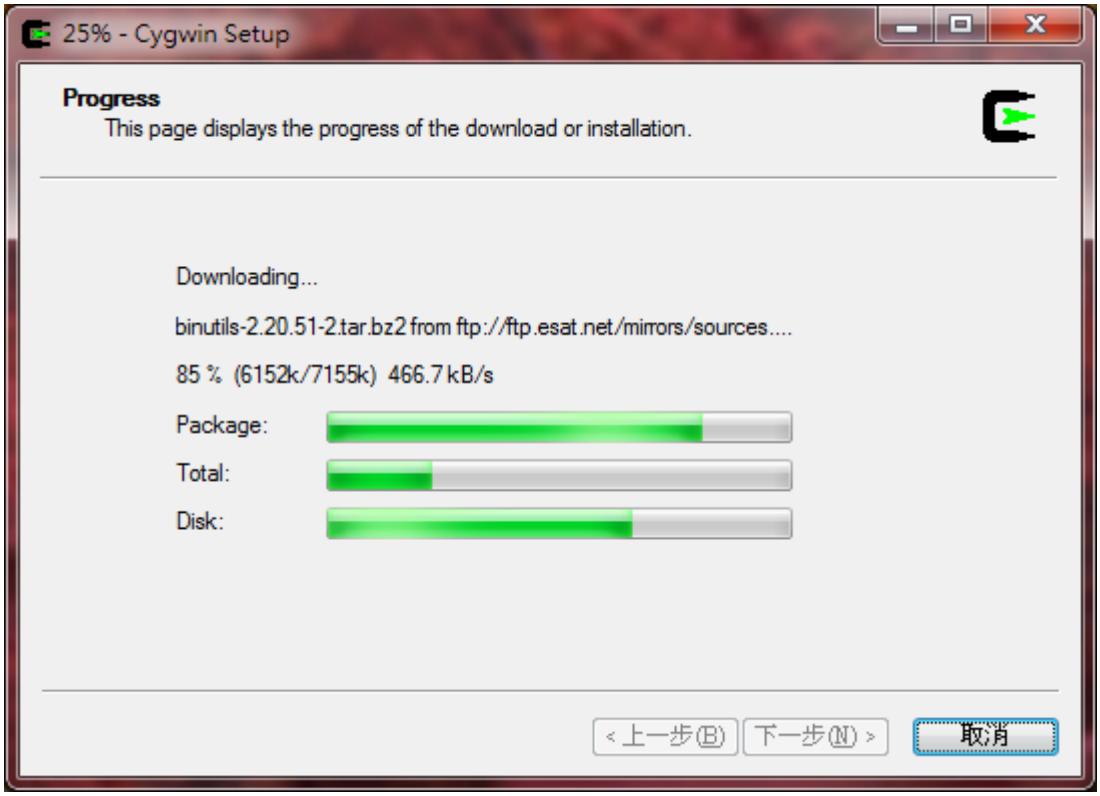
***wget*：於"Search"欄位中輸入"wget"，並點選"Skip"圖示進行安裝**

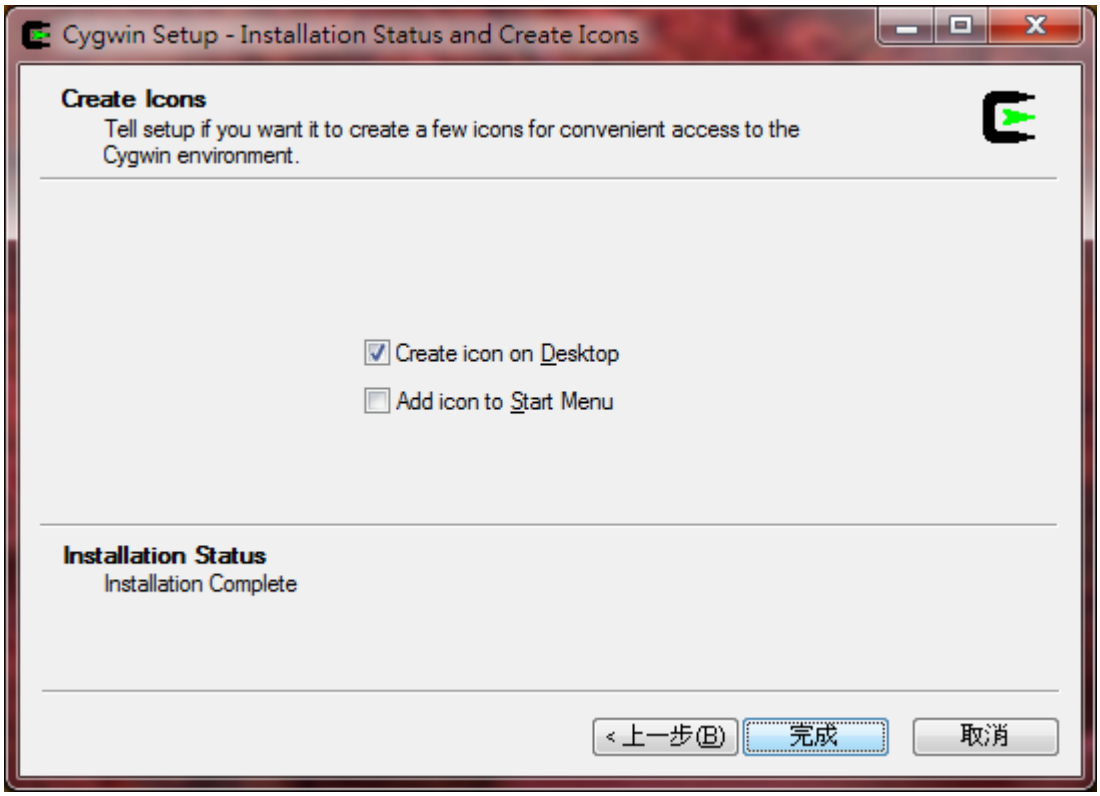
WGET 是一套在網路上進行程式下載的自由軟體，WGET 簡稱是由「World Wide Web」和「GET」結合，目前支援 HTTP、HTTPS 以及 FTP 三種最常見的網路下載。



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。

Step5 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

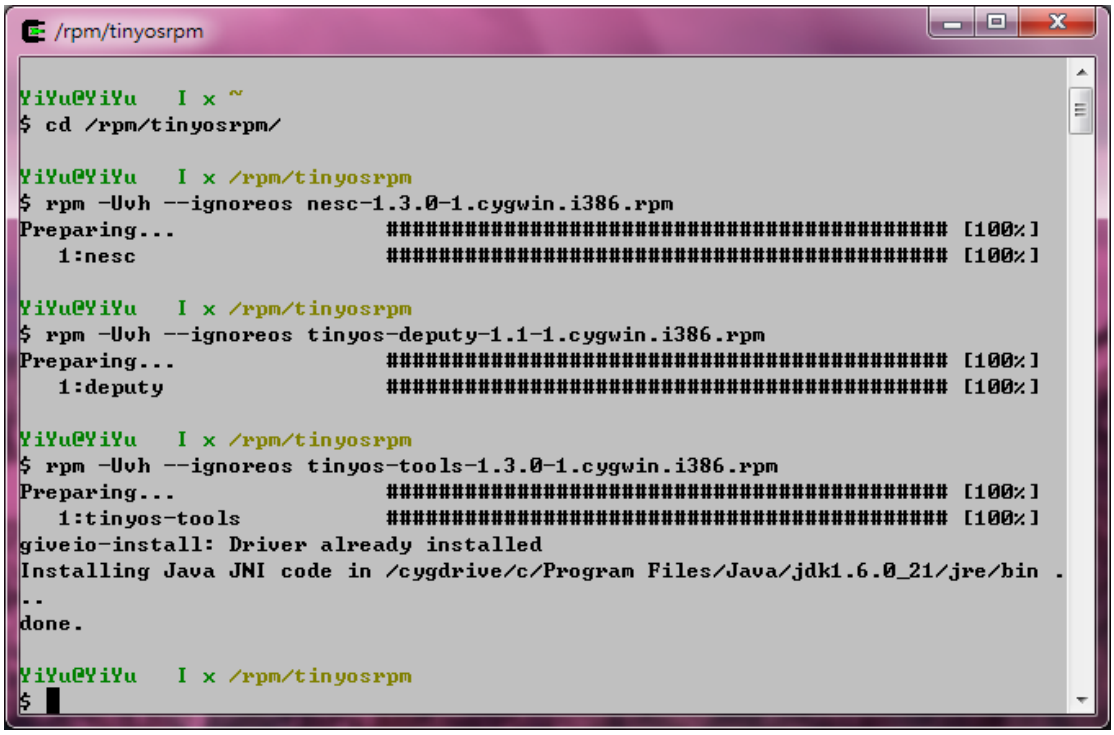
http://www.tinyos.net/dist-2.1.0/tinyos/windows/tinyos-tools-1.3.0-1.cygwin.i386.rpm

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

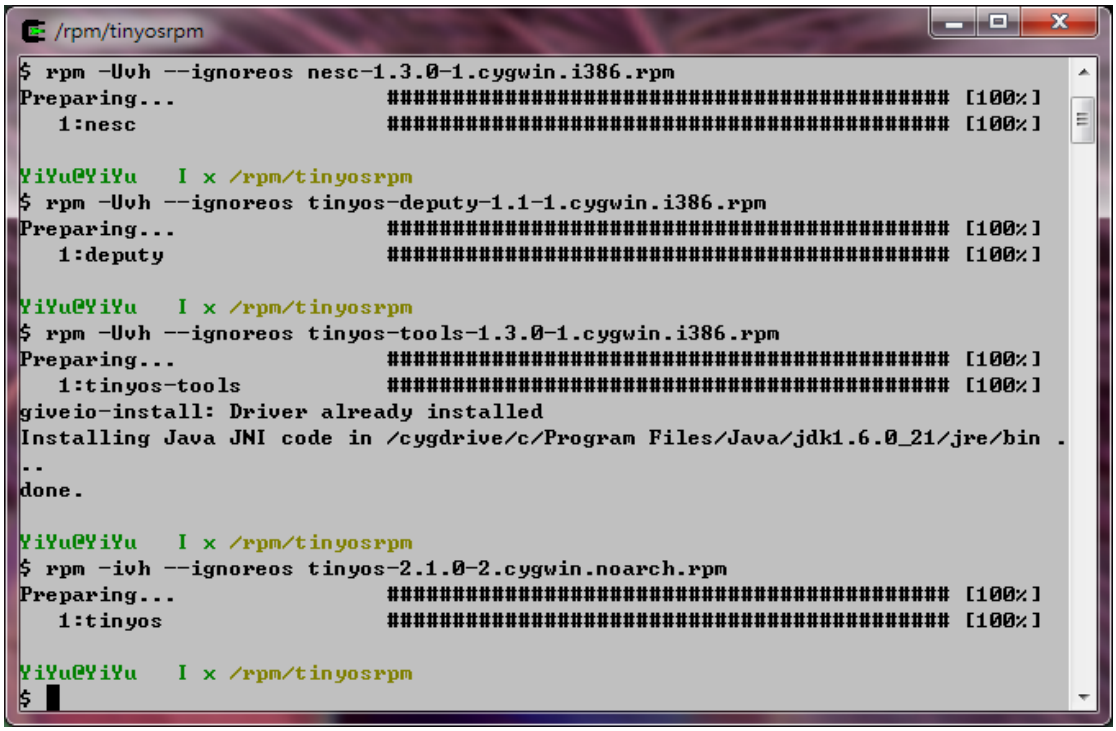
使用 TinyOS Source Tree 做為編譯和下載 TinyOS 的程式。

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：

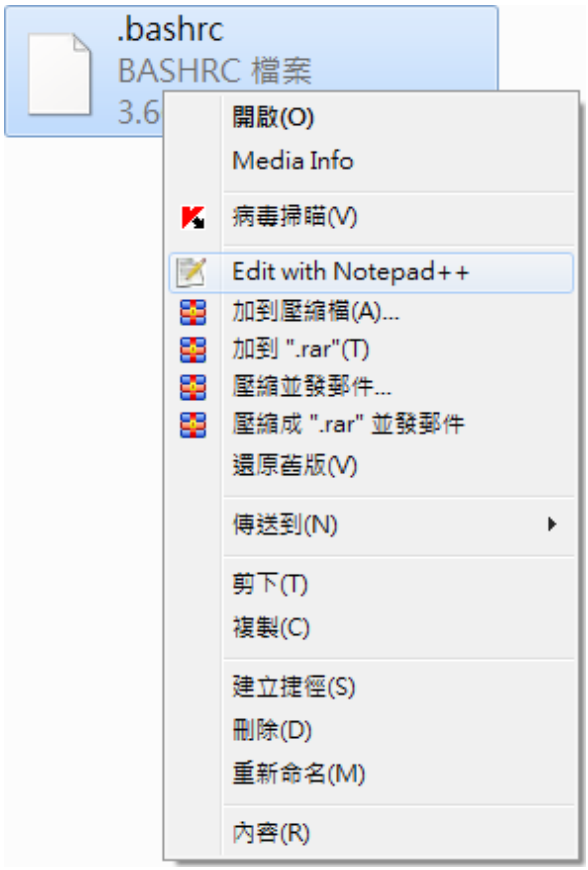
輸入 **rpm -ivh --ignoreos tinyos-2.1.0-2.cygwin.noarch.rpm**



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"

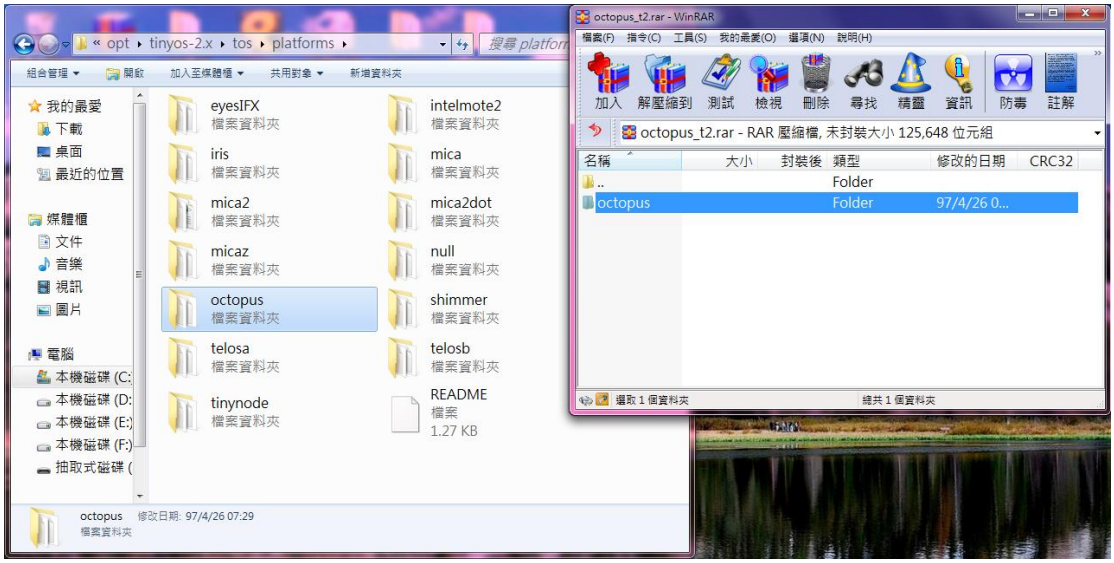
（紅色字體部分請根據安裝 Cygwin 的位置作變更，要注意換行符號的問題。)

```
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

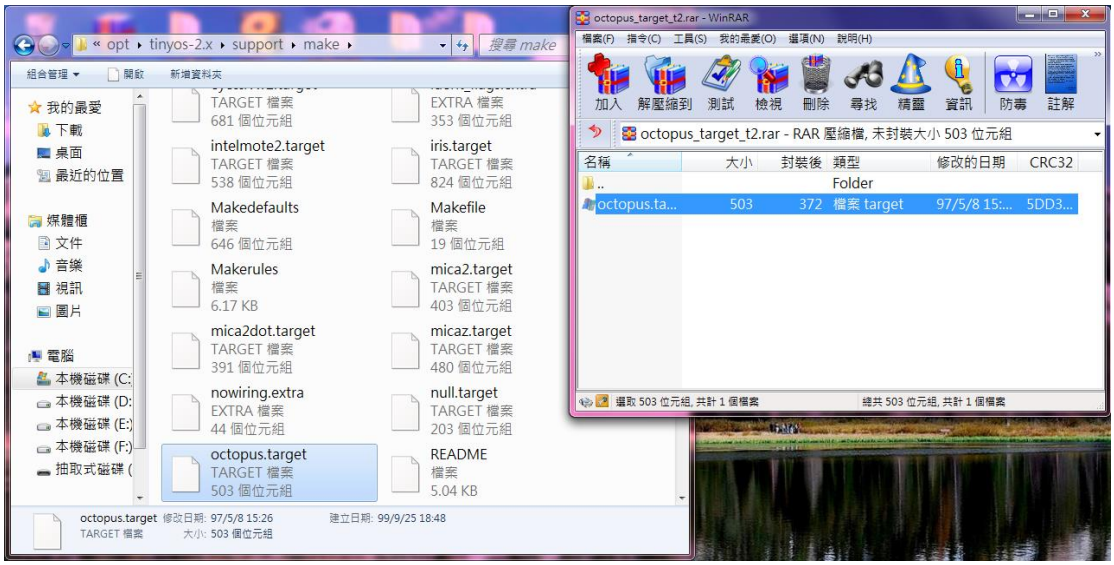
解壓縮 **octopus_t2.rar**，放至

/opt/tinyos-2.x/tos/platforms/ 中。



解壓縮 **octopus_target_t2.rar**，放至

/opt/tinyos-2.x/support/make/ 中。

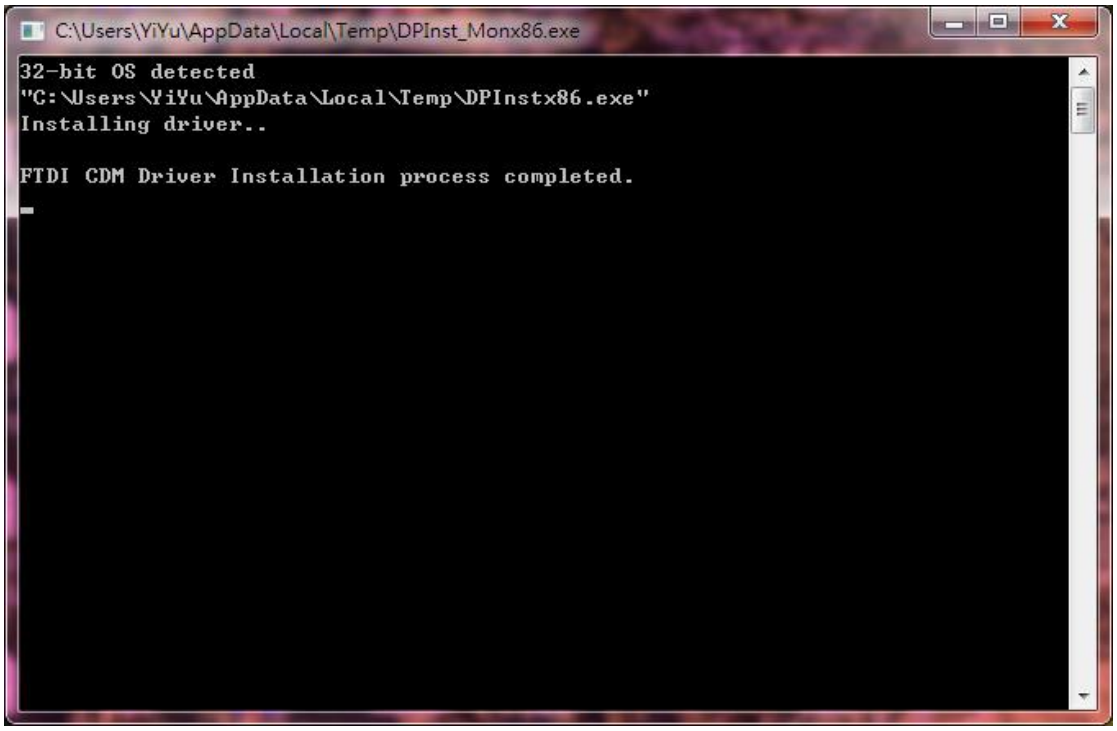


安裝 OctopusII Driver

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

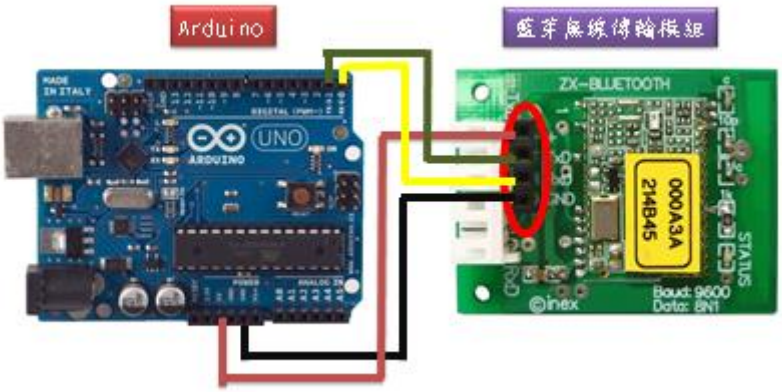
Operating System	Devices Supported	Driver Version	Release Date	Comments
Windows Server 2008 R2 Windows 7 Windows 7 x64 Windows Server 2008 Windows Server 2008 x64 Windows Vista Windows Vista x64 Windows XP Windows XP x64 Windows 2000 Windows Server 2003 Windows Server 2003 x64	FT2232H, FT4232H, FT232R, 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。



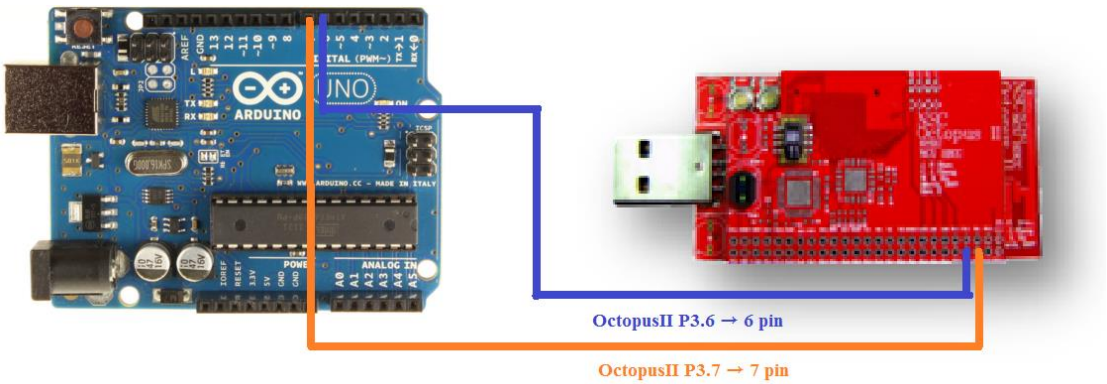
Part III 硬體電路整合

Step1 整合藍芽無線傳輸模組至 Arduino



上圖為藍芽無線傳輸模組與 Arduino 連接方式，由 Part1-Step2 的藍芽無線傳輸模組介紹，藍芽無線傳輸模組的工作電壓以及驅動電壓為 5V 直流電，因此我們必須於藍芽無線傳輸模組四個接腳中的電源輸入(標示為+5V)及 Enable(標示為 EN)腳位接上 5V 的直流電源(表示於圖中紅線)，並將 GND 腳位接地(表示於圖中黑線)，而其感測結果經由 RX / TX 的波形震盪輸出於 OUT 腳位，而此輸出訊號必須利用 Arduino 的 RX/TX 腳位接收。

Step2 整合 OctopusII 至 Arduino



上圖為 OctopusII 與 Arduino 連接方式，為了從 OctopusII 接收與傳送資料，我們必須將 OctopusII 上的 RX/TX 腳位連接至 Arduino 上，在 Arduino 的軟體撰寫中，我們透過 NewSoftSerial 之函式庫，使其避開 Bluetooth 傳送資料時所使用的 RX/TX 通道，所以我們必須將 OctopusII 上的 P3.6 腳位連接至 Arduino 上的 Pin 6 腳位；OctopusII 上的 P3.7 腳位連接至 Arduino 上的 Pin7 腳位，即完成 OctopusII 與 Arduino 的硬體連接。

Part IV 設計與製作物聯網閘道器

Step1 編譯 Octopus II 程式

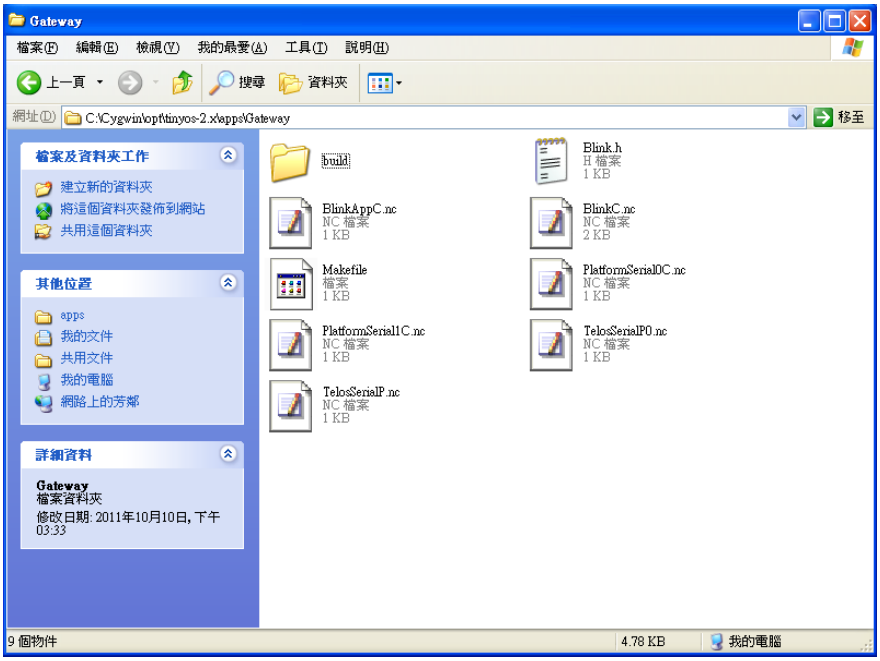
1.在 cygwin 命令列之下開啟 Octopus II 專案，此專案原始碼於 PartV 附錄程式碼中。

```
root@Shrimp ~
$ cd /opt/tinyos-2.x/apps

root@Shrimp /opt/tinyos-2.x/apps
$ cd Gateway

root@Shrimp /opt/tinyos-2.x/apps/Gateway
```

2.輸入 make octopus，編譯 Octopus II 程式

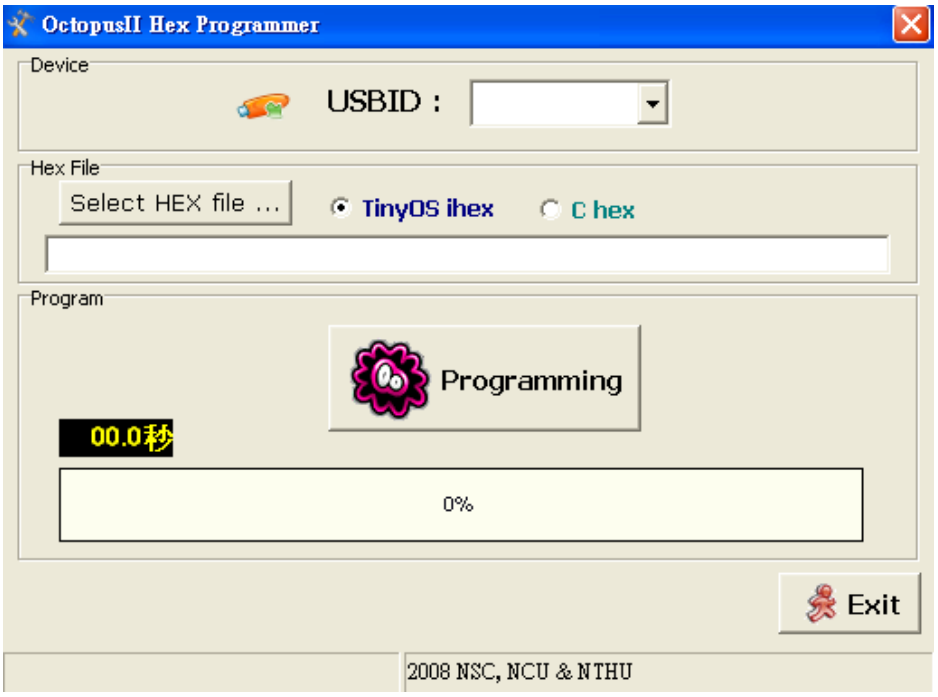
```
root@Shrimp /opt/tinyos-2.x/apps/Gateway
$ make octopus
mkdir -p build/octopus
    compiling BlinkAppC to a octopus binary
ncc -o build/octopus/main.exe  -Os -O -mdisable-hwmul -fnesc-separator=__ -Wall
-Wshadow -Wnesc-all -target=octopus -fnesc-cfile=build/octopus/app.c -board= -DD
EFINED_TOS_AM_GROUP=0x22 -DIDENT_PROGRAM_NAME=\"BlinkAppC\" -DIDENT_USER_ID=\"ro
ot\" -DIDENT_HOSTNAME=\"Shrimp\" -DIDENT_USER_HASH=0xa3bcf819L -DIDENT_UNIX_TIME
=0x4e929fd7L -DIDENT_UID_HASH=0x5a8aa5d4L  BlinkAppC.nc -lm
/opt/tinyos-2.x/tos/chips/cc2420/lpl/DummyLplC.nc:39:2: warning: #warning "*** L
OW POWER COMMUNICATIONS DISABLED ***"
    compiled BlinkAppC to build/octopus/main.exe
            11096 bytes in ROM
              267 bytes in RAM
msp430-objcopy --output-target=ihex build/octopus/main.exe build/octopus/main.ih
ex
    writing TOS image

root@Shrimp /opt/tinyos-2.x/apps/Gateway
$
```

3.編譯完成後，專案底下會生成 build 資料夾

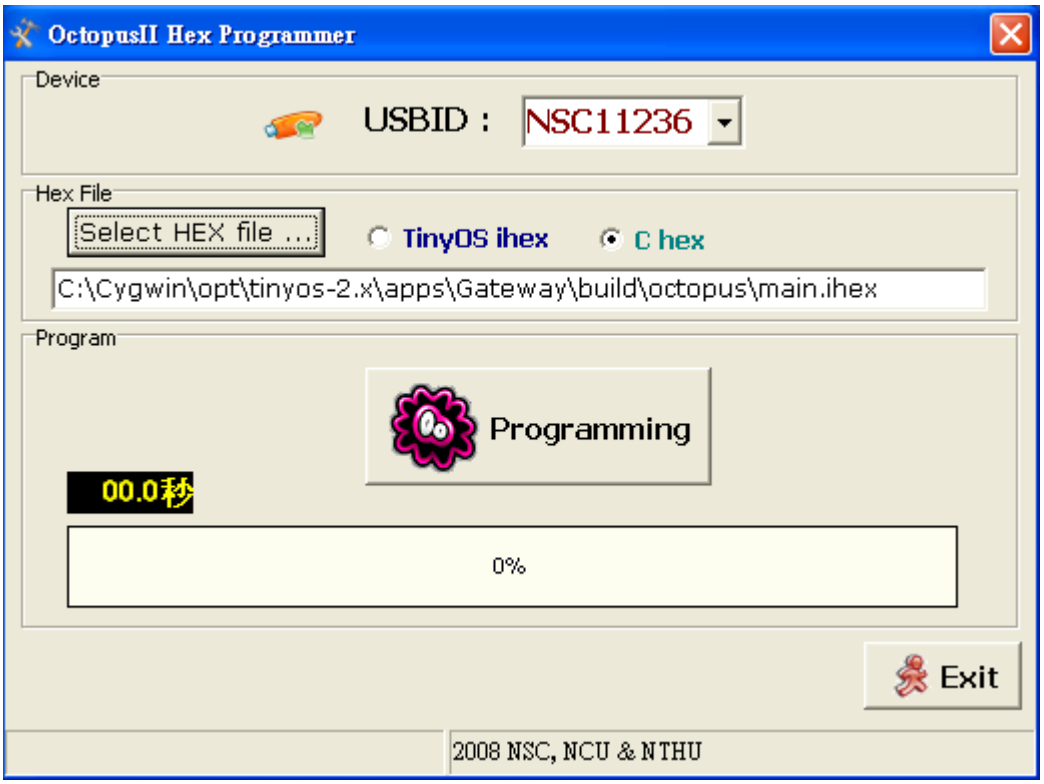


Step2 燒錄 Octopus II 程式

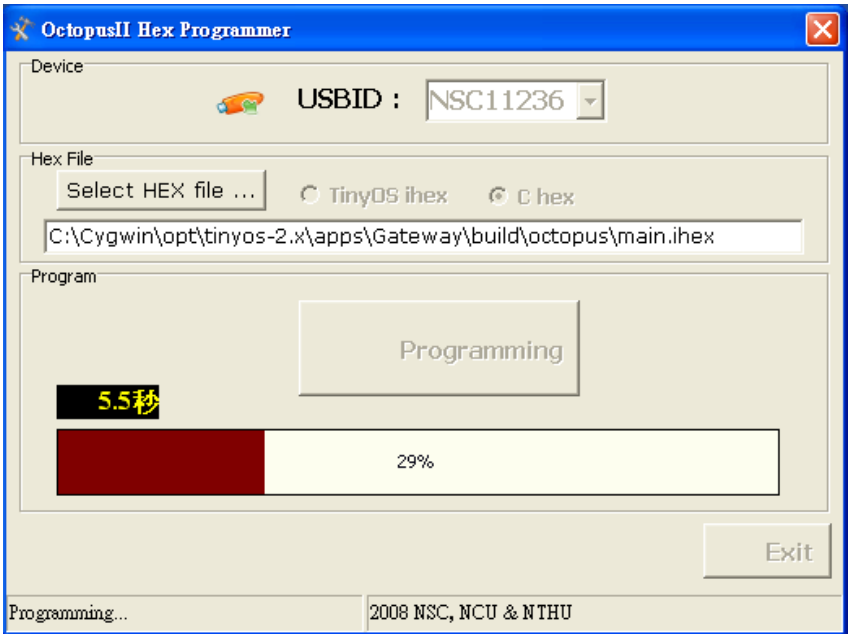
1. 開啟 Octopus II Hex Programmer 燒錄程式



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

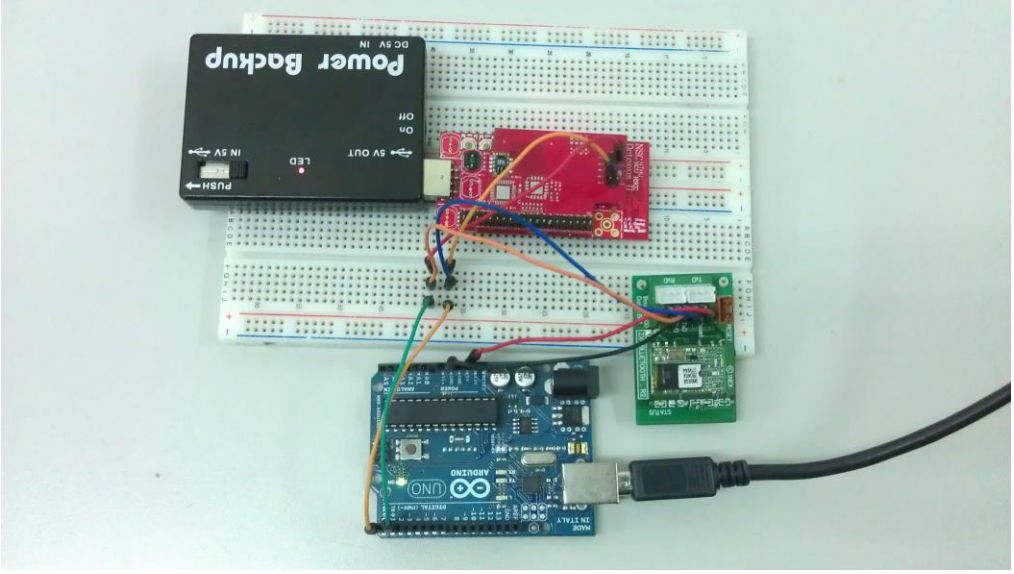


3. 按下 Programming 即可燒錄程式至 Octopus II



Step3 測試與展示系統

本實驗實際展示功能為利用 Arduino、Octopus II 以及 Bluetooth 模組來實現雙網合一的功能，我們可以接收 ZigBee 的封包並透過 Bluetooth 傳送至其他 Bluetooth 裝置外，亦可接收 Bluetooth 裝置的封包並透過 ZigBee 傳送至其他 Zigbee 裝置，當 OctopusII 感測器欲傳送封包給其他 OctopusII 感測器時，其板子上的紅燈 LED 即會閃爍，表示封包利用廣播的方式進行傳送；而當 OctopusII 接收到此封包時，板子上的藍色 LED 即會閃爍，表示封包接收成功。



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，我們利用 Arduino、Octopus II 與藍芽的結合來時做這次我們用的 Gateway，以下，我們分別對 OctopusII 以及 Arduino 之程式碼進行解釋。

Octopus II 上可分成兩個程式：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。

Gateway\Blink.h

```
#ifndef BLINKTORADIO_H
#define BLINKTORADIO_H

enum {
   AM_BLINKTORADIOMSG= 6,
   TIMER_PERIOD_MILLI = 250
};

typedef nx_struct BlinkToRadioMsg {
   nx_uint16_t nodeid;
   nx_uint8_t cmd;
} BlinkToRadioMsg;

#endif
```

Gateway\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*	當接收到資料時，閃爍 Led
3	*Timer*	當接收到資料時，倒數 *n* 秒後執行對應程式

宣告需要使用的 *interface* 程式碼如下：

```
#include "Timer.h"
#include "Blink.h"
module BlinkC @safe() {

	uses interface HplMsp430GeneralIO as Port26;
	uses interface Leds;
	uses interface Boot;
	uses interface Receive;
	uses interface SplitControl as AMControl;

}
implementation {

	uint8_t cmd;
	message_t pkt;
	bool busy = FALSE;
	bool isOn = FALSE;

	event void Boot.booted() {
		call Leds.led0On();
		call AMControl.start();

	}
	event void AMControl.startDone(error_t err) {
		if(err == SUCCESS) {
			//call Timer0.startPeriodic(TIMER_PERIOD_MILLI);

		}
		else {
			call AMControl.start();
		}
	}

	event void AMControl.stopDone(error_t err) {

				call Leds.led3Toggle();
```

```
                        if(isOn){
                                call Port26.set();

                        }else{
                                call Port26.clr();

                        }
                call    AMControl.start();
        }

        event message_t * Receive.receive(message_t * msg, void * payload,
                uint8_t len) {
            if(len == sizeof(BlinkToRadioMsg)) {
                BlinkToRadioMsg * btrpkt = (BlinkToRadioMsg * ) payload;
                //    setLeds(btrpkt->counter);
                // check cmd
                //....
                cmd = btrpkt->cmd;
                call Leds.led1Toggle();

                //call AMControl.stop();

                if(cmd == 0x29) {//41
                        isOn=TRUE;
                        call AMControl.stop();
                }else if(cmd == 0x28){ //40
                        isOn=FALSE;
                        call AMControl.stop();
                }
                //call AMControl.start();

            }
            return msg;
        }
}
```

在宣告完需要使用的 *interface* 後於 BlinkC.nc 後，我們需在 *implementation* 內撰寫程式運作邏輯，程式一開始會執行 *Boot.booted*()，我們於其中設定 P2.6

支援中斷，並將其腳位設定為輸入腳位。當 P2.6 收到 X-Band Motion Detector 因偵測到物體移動而傳來的訊號後，將會執行 *Interrupt26.fired*()，我們在這區間取消中斷和致能中斷以避免收到重複的中斷，再利用計時器使 Led 閃爍。

我們首先需將對應 *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()
3	*GpioInterrupt*	**async event** void Interrupt26.fired()

implementation 內的程式碼如下：

```
implementation
{
   event void Boot.booted() // booted event
   {
       call Interrupt26.enableFallingEdge();       //設定port26為下緣觸發

       call Port26.makeInput();        //設定port26為輸入腳位
   }

   async event void Interrupt26.fired(){ //設定port26被觸發
   call Interrupt26.disable();//disable port26

      call Timer0.startOneShot(250);        //0.25秒後執行Timer0.fired()

   call Interrupt26.enableFallingEdge();//enable port26
   }
   event void Timer0.fired()
   {
      call Leds.led1Toggle();  //改變LED之狀態(明變暗或暗變明)
   }
}
```

Gateway\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：

```
#include "Blink.h"

configuration BlinkAppC {
}
implementation {
	components MainC, BlinkC, LedsC;
	components new TimerMilliC() as Timer0;

	components HplMsp430InterruptC as InterruptC;
	components HplMsp430GeneralIOC as GeneralIOC;

	components new Msp430InterruptC() as IntUserSwitchC26;

	components ActiveMessageC;
	components new AMReceiverC(AM_BLINKTORADIOMSG);



	//UserSwitchC26->GeneralIOC.Port26;
	BlinkC.Port26->GeneralIOC.Port26;
	BlinkC->MainC.Boot;
	BlinkC.Leds->LedsC;

	BlinkC.AMControl->ActiveMessageC;
	BlinkC.Receive->AMReceiverC;
```

```
}
```

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
```

由於 Msp430InterruptC 提供的 HplInterrupt 介面是由 HplMsp430InterruptC 元件提供的，固必須將其連結(wire)，連結記為“->”，其 Port26 介面即為 HplInterrupt 介面的別名。

```
    IntUserSwitchC26.HplInterrupt -> InterruptC.Port26; //wire HplMsp430 Interrupt interface
```

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.Interrupt26 -> IntUserSwitchC26.Interrupt;    //wire GpioInterrupt interface
  BlinkC.Port26 -> GeneralIOC.Port26;    //wire HplMsp430GeneralIOC    interface
  BlinkC.Boot -> MainC.Boot;    //wire Boot interface
  BlinkC.Timer0 -> Timer0; // wire Timer interface
  BlinkC.Leds -> LedsC; // wire Led interface
}
```

Gateway\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)
```

在 Ardunio 上，因為 Bluetooth 裝置與 OctopusII 感測器所使用的傳輸介面均為 UART，但在 Arduino 控制板上僅有一組 RX/TX 介面，故我們必須利用軟體的方式模擬出一組 RX/TX 介面，所使用到的函式庫為「NewSoftSerial」，並宣告此組 RX/TX 的輸入腳位為 Pin6 與 Pin7。

```
#include <NewSoftSerial.h>
#include <IRremote.h>

// for all
   byte incomingByte;
// soft uart for octopus
 NewSoftSerial mySerial(6, 7);            //  模擬RX/TX介面於Pin6與Pin7

void setup()
{
   Serial.begin(9600);
   Serial.println("gateway start");
   // set the data rate for the NewSoftSerial port
   mySerial.begin(115200);
 //mySerial.println("Hello, world?");

}
void loop()                              // run over and over again
{
    if(!isDemoing){ // if is not demoing

          byte incomingByte;
            while (Serial.available() && !isDemoing) {
                    incomingByte = Serial.read();
                    Serial.print("receive byte = ");
                    Serial.println(incomingByte);
            }
    }
}
```