『物聯網概論』實驗手冊

實驗模組名稱: 無線感測器基礎程式教學

開發教師: 翁仲銘教授 (cmown@mail.sju.edu.tw**)**

學校系所: 聖約翰科技大學電腦與通訊工程學系

實驗平台: Octopus II

實驗環境：Windows 7 32bits, Windows XP

實驗主軸: 無線感測節點程式語法練習

實驗內容關鍵字: Octopus II ,Tiny OS

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

本實驗主要在教導學生如何在 TinyOS 環境中，練習 nesC 的程式，為入門的實驗教材。本實驗教材總共分成四個部分，藉由這四個實驗來讓學生瞭解入門的 nesC 元件和介面的概念，並且熟悉整體的程式撰寫。

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

實驗所需軟體

● PC

- Windows 7 或 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

實驗一 HelloWorld 4
實驗目的 4
實驗器材 4
實驗過程 4
Step 1 瞭解 Octopus II 4
Step 2 建立 Helloworld 目錄 5
Step 3 Helloworld 程式碼 8
Step 4 編譯和燒錄程式碼 9
實驗結果 11
實驗練習 11
實驗二 HelloWorld 12
實驗目的 12
實驗器材 12
實驗過程 12
Step 1 瞭解 Octopus II 12
Step 2 瞭解 Timer 元件 13
Step 3 建立 hellotimer 目錄 14
Step 4 hellotimer 程式碼 14
實驗練習 16
實驗 三 Serial Port 輸出 17
實驗目的 17
實驗器材 17
實驗過程 17
Step 1 瞭解 Octopus II 17
Step 2 瞭解 Printf 函數 18
Step 3 建立 testprintf 目錄 18
Step 4 testprintf 程式碼 19
實驗結果 20
實驗練習 22

實驗一 HelloWorld

實驗目的

本實驗為熟悉 nesC 語法的入門第一支程式。每次的實驗中，皆分成三個程式部分讓同學編寫，分別為設定檔（Configuraton），模組檔（Module），及編譯檔（Makefile），透過此三個主要檔案來讓學生瞭解 TinyOS 的運作與開發。而在實驗一的部分，則是仿效一般坊間的程式設計手冊，教導學生用燈號代替輸出字串，來寫出一個簡短的 nesC 程式。

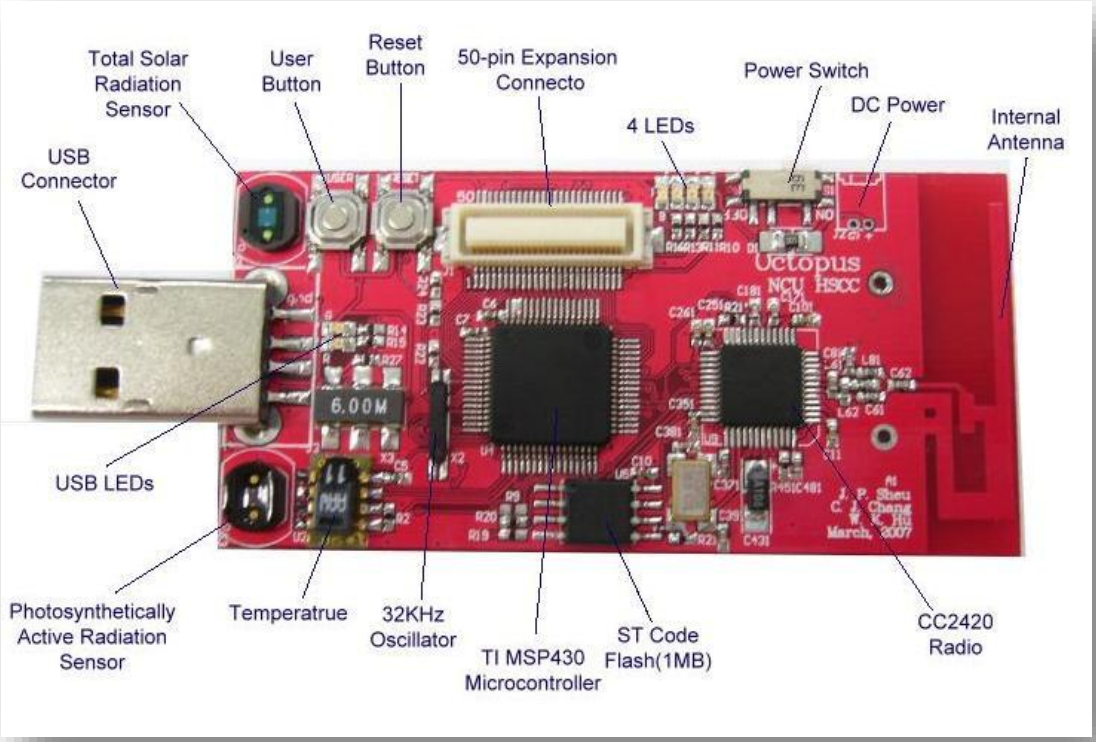
實驗器材

- Octopus II 開發平台
- TinyOS 開發環境請參考前一份實驗教材

實驗過程

Step 1 瞭解 Octopus II

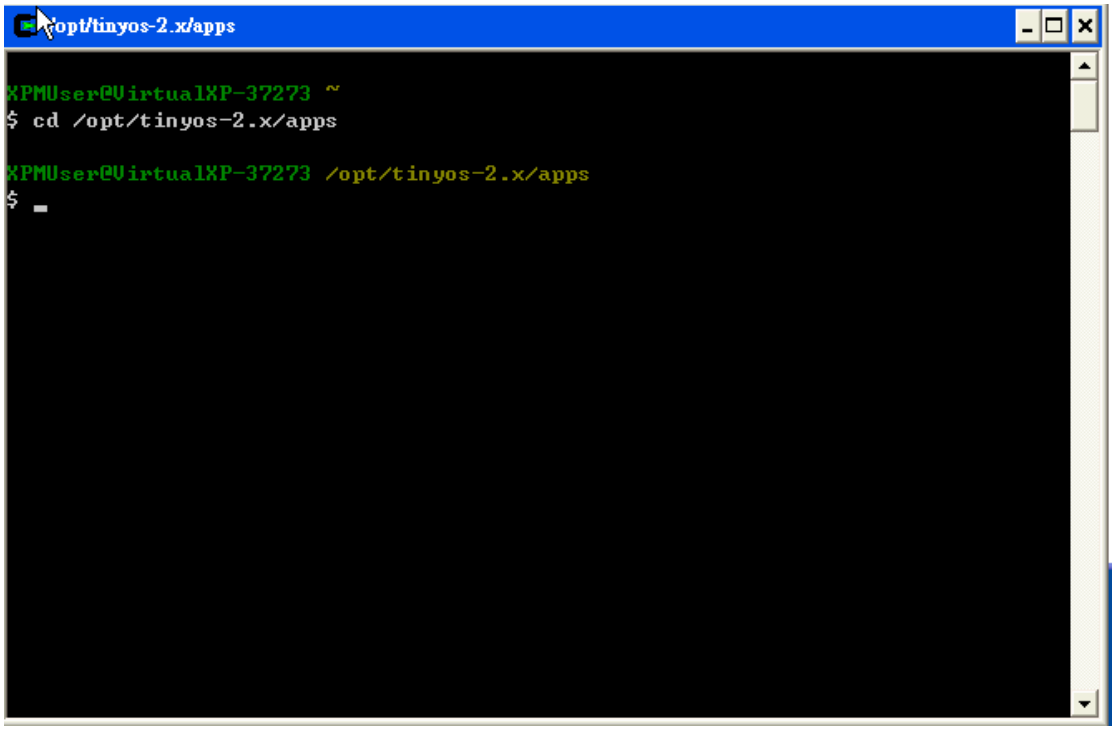
Octopus 是由清華大學和中央大學團隊所共同研發的無線感測器，目前已開發出三個版本：Octopus I、Octopus II 以及 Octopus X，三個版本所使用的無線通訊模組皆為 CC2420 RF 晶片模組，第一版的 Octopus I 參照 Crossbow 的 MICAz 感測節點為其主要開發架構，並搭載 8-bit AVR MCU 為中央處理單元，具有 4 KB 的記憶體空間以及 128 KB 的快閃記憶體容量；第二版的 Octopus II 則參照 MoteIV 的 Tmote Sky 感測節點為其主要開發架構，搭載 16-bit MSP430 MCU 為中央處理單元，具有 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。



Step 2 建立 Helloworld 目錄

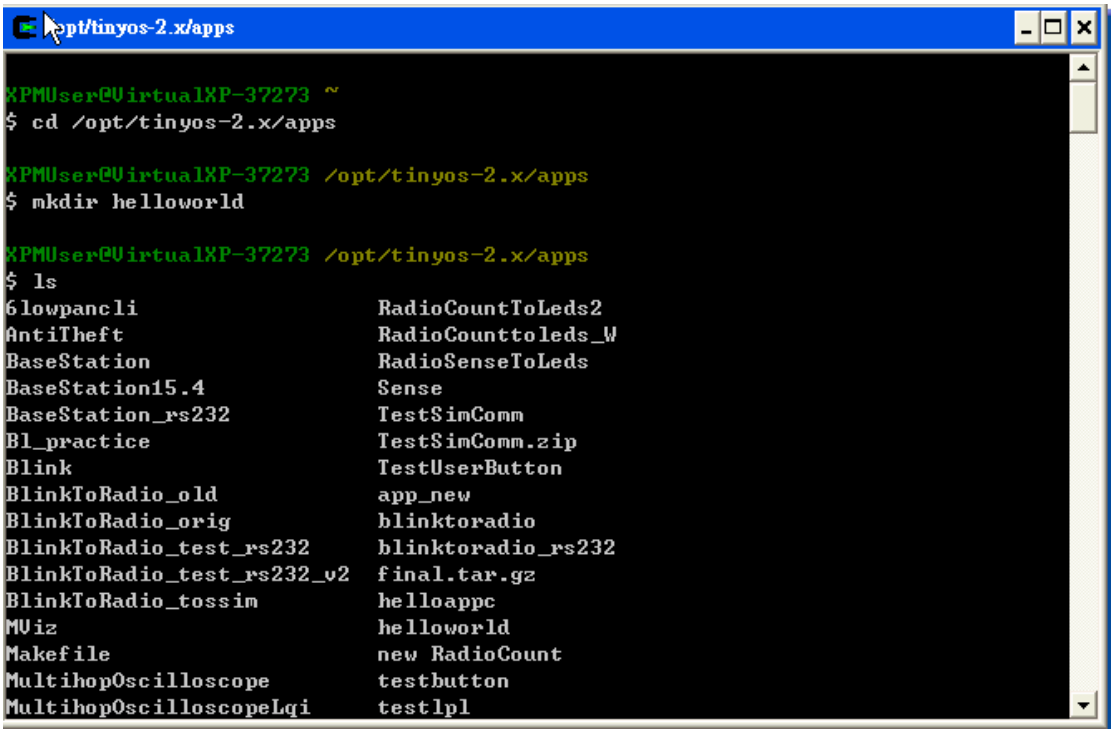
在實驗開始時，請先建立本實驗所需的目錄，可以利用兩種方式：

1. 首先執行 Cygwin 程式，並且改變目錄到 /opt/tinyos-2.x/apps（使用 cd 指令）



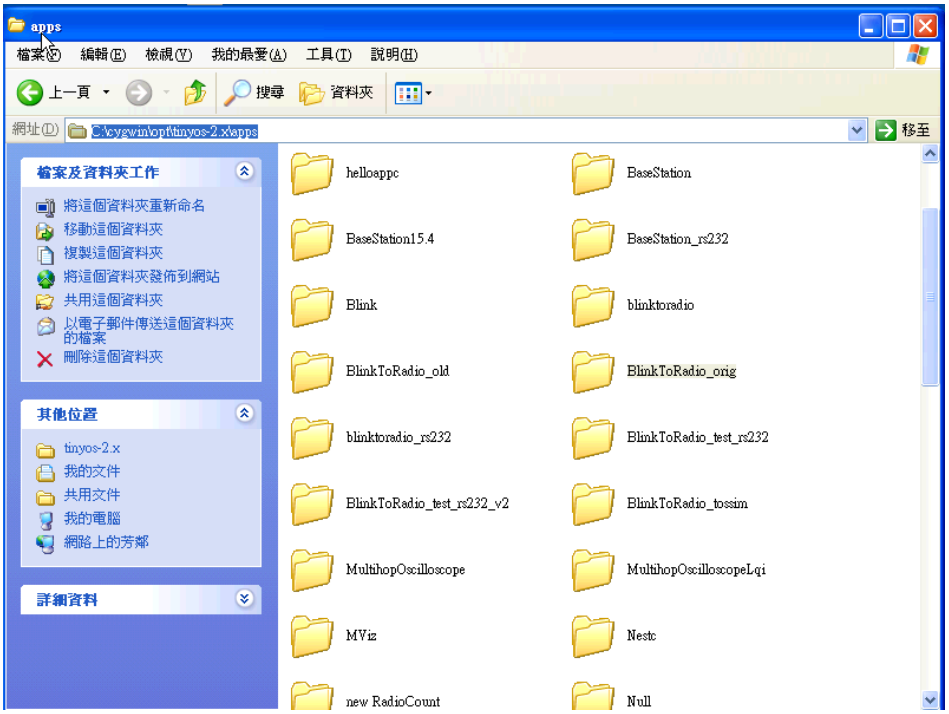
2. 接著在 Cygwin 程式中，鍵入 mkdir helloworld，以建立一個新目錄。

3. 接著鍵入指令碼 ls，以查詢目錄是否正確建立。

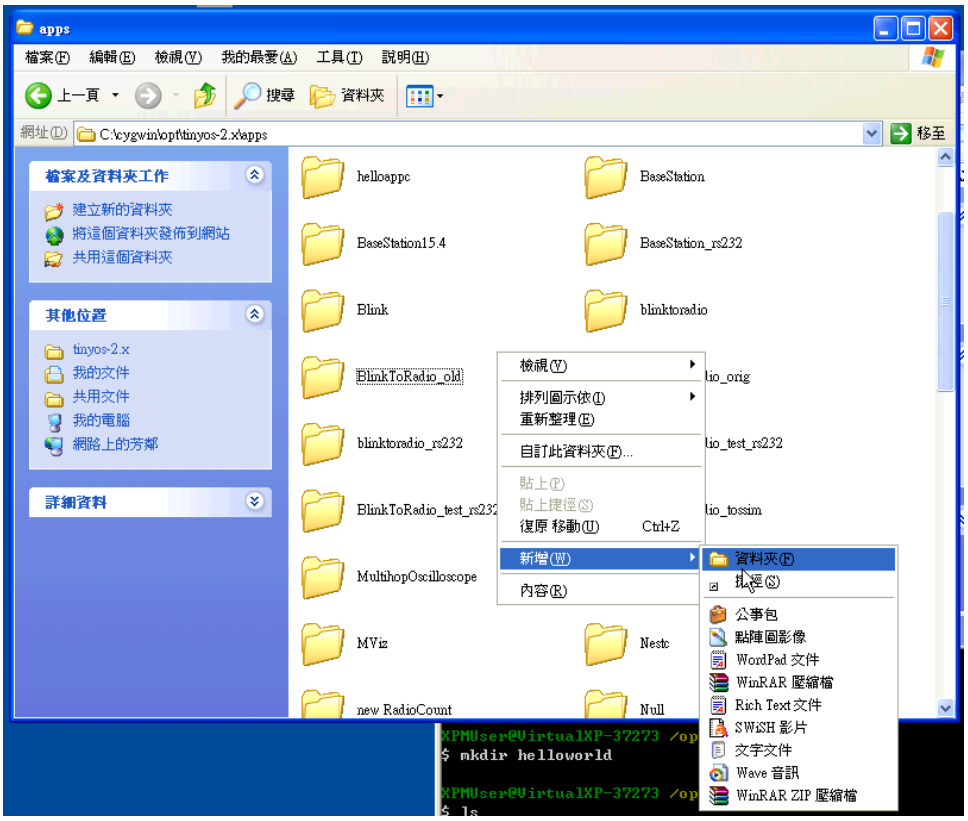


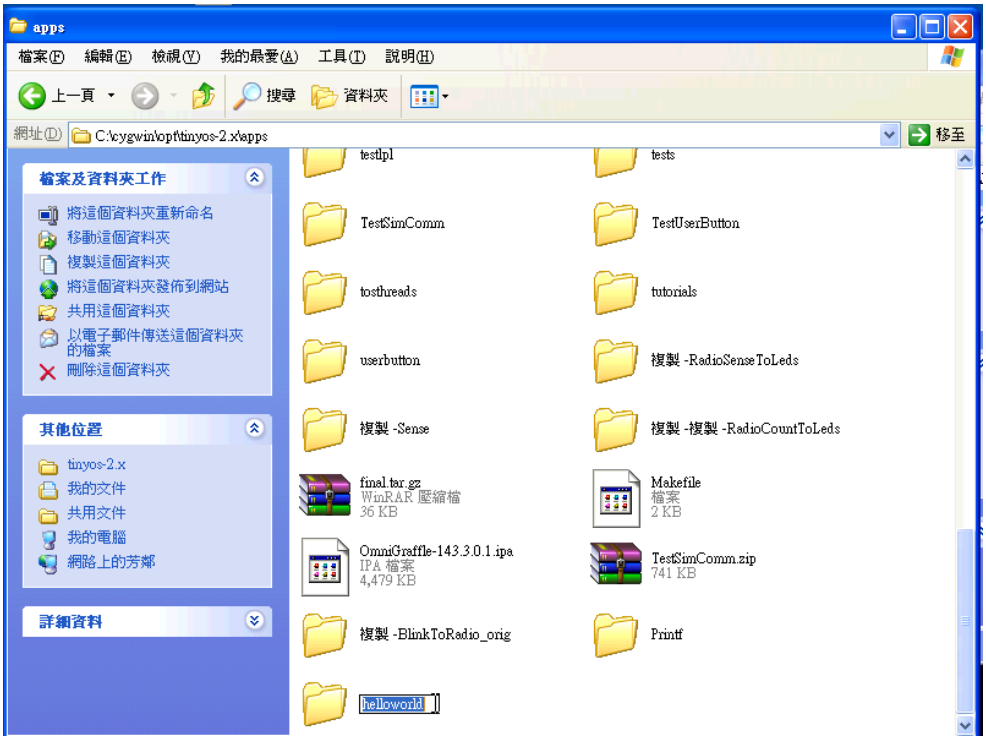
或是

1. 直接使用 Window 滑鼠，雙擊「我的電腦」，並點選到目錄「C:\cygwin\opt\tinyos-2.x\apps」



2. 按滑鼠右鍵以建立一個新目錄，名稱為「helloworld」





Step 3 Helloworld 程式碼

以下為本實驗的程式碼，總共分成三個部分。主要目的是讓 Octopus 模組亮 Led 燈表示開機成功。其中，共用三個元件，分別是程式本身元件 helloC，系統元件包含 MainC 和 LedsC，分別負責開機和模組上燈號的控制。請使用小作家或其他文字編輯軟體，打開一個文件檔，並編輯和存檔下列程式碼到文字檔中，請記得副檔名必須一致。

- 設定檔： helloAppC.nc，設定檔內宣告所有用到的元件（component）及元件間的線性關係。

```
configuration helloAppC{
}
implementation {
    components MainC, helloC, LedsC;     //設定所用到的元件
            // 以下皆為 Interface 線性關係的定義
    helloC.Boot -> MainC.Boot;
    helloC.Leds -> LedsC.Leds;
}
```

- 模組檔　helloC.nc，則是實際要撰寫的應用程式碼

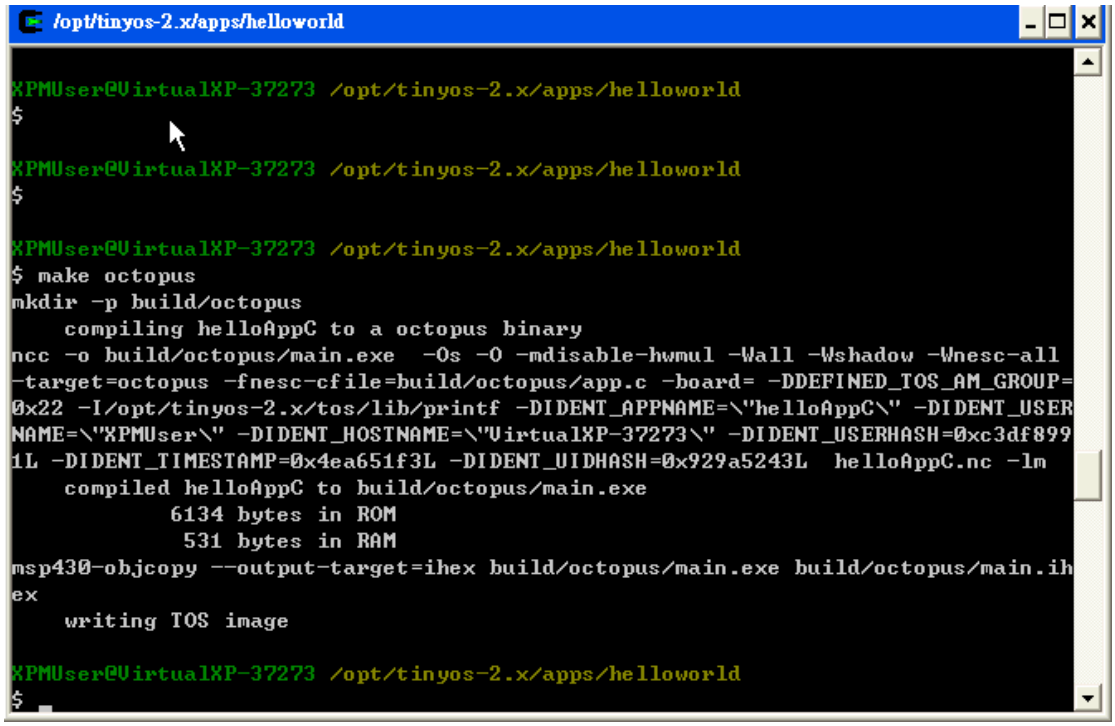
```
module helloC {
   uses interface Leds;
   uses interface Boot;
}
implementation {
   event void Boot.booted() {
      call Leds.led0Toggle();
      //經過開機的程序後，程式接收系統響應的 Boot 事件，並亮第一顆燈號
   }
}
```

- 編譯檔 Makefile

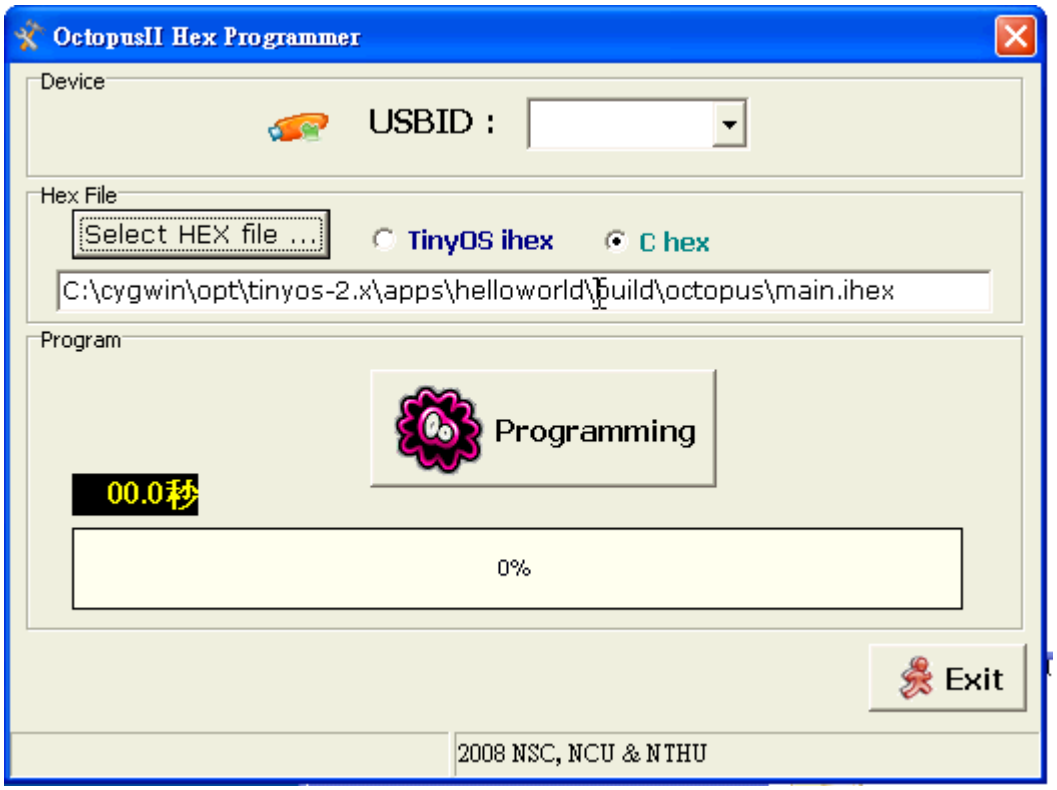
```
COMPONENT=helloAppC
include $(MAKERULES)
```

Step 4 編譯和燒錄程式碼

請在 Cygwin 目錄下打入「make octopus」，以執行編譯的動作，請參考下圖：

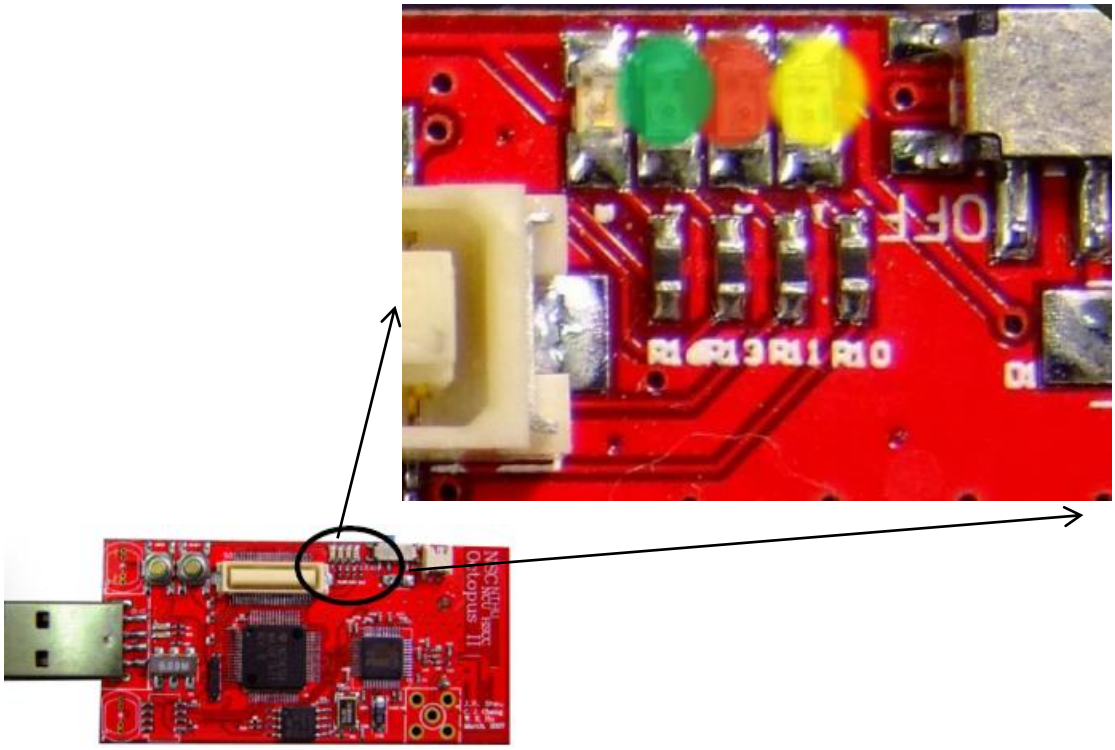


並且點選「octopusII_prog.exe」程式以執行燒錄的動作，請參考下圖：



實驗結果

下圖則為 Octopus 模組上的燈號顯示狀況



實驗練習

1. 請參考 C:\cygwin\opt\tinyos-2.x\tos\interfaces 目錄下的資料，瞭解 LedsC Interface 的函數有哪些。
2. 請根據不同的指令函數，來讓模組上的閃爍不同 Leds 燈號。
3. 請同學辨識出模組上的燈號編號、顏色及對應的位置。

實驗二 HelloWorld

實驗目的

本實驗為熟悉 nesC 語法的入門第二支程式。Counter 以及 Led 是 sensor 上最常見的硬體，也是最容易使用的 component，只要下正確的指令，產生的結果就可以清楚的在 sensor 上的燈號變化看見。本實驗則是介紹 Timer 這個元件來產生 Counter 的功能。

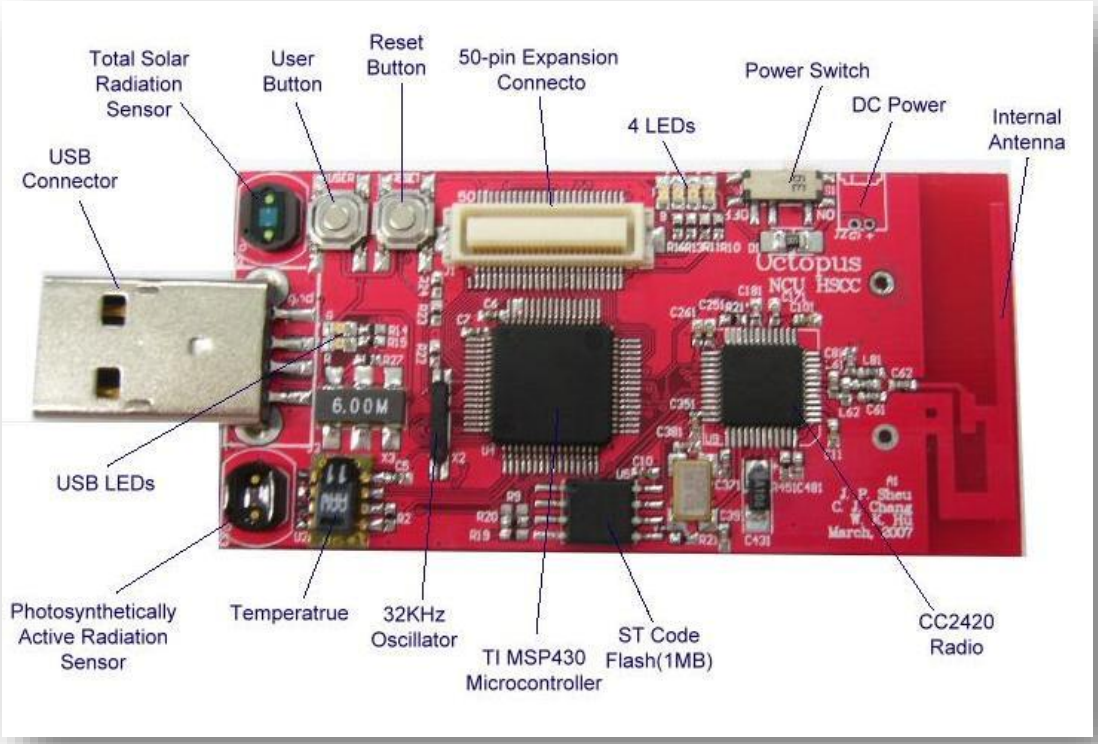
實驗器材

- Octopus II 開發平台
- TinyOS 開發環境請參考前一份實驗教材

實驗過程

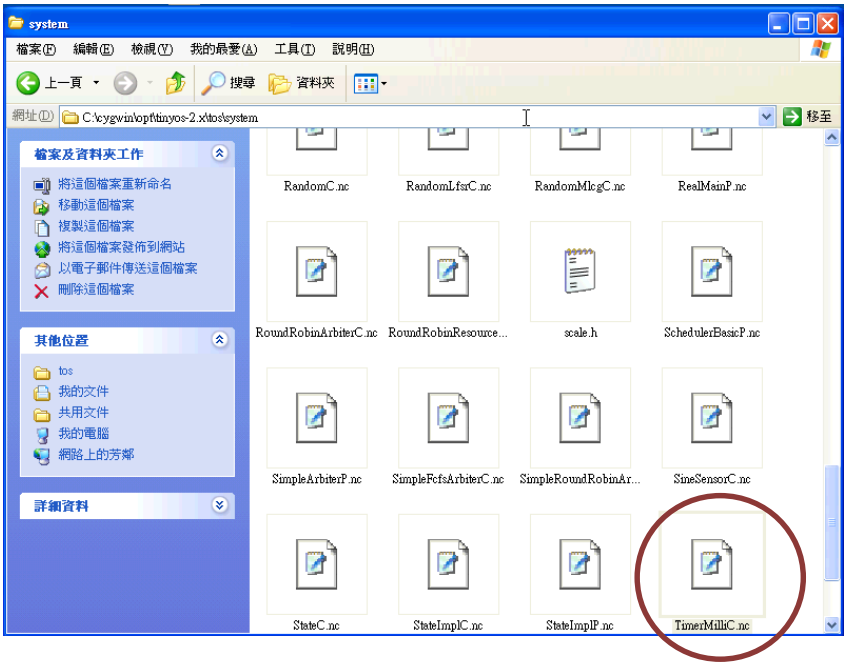
Step 1 瞭解 Octopus II

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。



Step 2 瞭解 Timer 元件

- TimerMilliC 為計時元件，提供了數個命令來控制、重複地間隔計時等功能。其元件的檔案位置在「C:\cygwin\opt\tinyos-2.x\tos\system」，請參考下圖。



- Timer<TMilli>則是由 TimerMilliC 所提供的介面,< >內參數則是定義為計時單位，可以為 milliseconds 或 microseconds 等。
- Timer<TMilli>介面中所提供的指令包含有下列：
  1. startPeriodic(uint32_t dt): 重覆一段 dt 時間後產生 fired()事件
  2. startOneShot(uint32_t dt):只計時一段 dt 的時間即只產生一次 fired()事件
  3. stop():停止計時的動作
- Event:當完成計時動作時將產生的事件，並以下列的程式碼呈現
  event void Timer.fired()

Step 3 建立 hellotimer 目錄

在實驗開始時，請先建立本實驗所需的目錄，可以利用兩種方式 (圖略，請參考實驗一)：

1. 首先執行 Cygwin 程式，並且改變目錄到「/opt/tinyos-2.x/apps」（使用 cd 指令）

2. 接著在 Cygwin 程式中，鍵入 「mkdir hellotimer」，以建立一個新目錄。

3. 接著鍵入指令碼 ls，以查詢目錄是否正確建立。

或

1. 直接使用 Window 滑鼠，雙擊「我的電腦」，並點選到目錄「C:\cygwin\opt\tinyos-2.x\apps」
2. 按滑鼠右鍵以建立一個新目錄，名稱為「hellotimer」

Step 4 hellotimer 程式碼

以下為本實驗的程式碼，總共分成三個部分。主要目的是讓 Octopus 模組亮 Led 燈表示開機成功，另外，則宣告一個 TimerMilliC 的元件，並設定為每一秒計數一次。程式中，共用四個元件，分別是程式本身元件 hellotimerC，系統元件包含 MainC 和 LedsC，分別負責開機和模組上燈號的控制，最後則是 Timer 的控制元件 TimerMilliC。請使用小作家或其他文字編輯軟體，打開一個文件檔，並編輯和存檔下列程式碼到文字檔中，請記得副檔名必須一致。

- 設定檔： hellotimerAppC.nc，設定檔內宣告所有用到的元件（components）及元件間的線性關係。其中，TimerMilliC 是一種 generic 元件，也就是說，可以重複宣告多次，因此需要使用關鍵字 new 做為初始化動作，並且需要為此元件多取一個別名（因為可以宣告多次，辨別之用）。

```
configuration hellotimerAppC{
}
implementation {
    components MainC, hellotimerC, LedsC;     //設定所用到的元件
    components new TimerMilliC as Timer0; //Timer0 為元件別名
            //  以下皆為 Interface 線性關係的定義
    hellotimerC.Boot -> MainC.Boot;
    hellotimerC.Leds -> LedsC.Leds;
    hellotimerC.Ti -> Timer0;             //Ti 為介面的別名，定義在模組檔中
}
```

- 模組檔　helloC.nc，則是實際要撰寫的應用程式碼

```
module hellotimerC {
   uses interface Leds;
   uses interface Boot;
   uses interface Timer<TMilli> as Ti; //介面別名定義為 Ti
}
implementation {
   event void Boot.booted() {
      call Leds.led0On();
      call Ti.startPeriodic( 1000 );   //宣告計時器每一秒響應一次
      //經過開機的程序後，程式接收系統響應的 Boot 事件，並亮第一顆燈號
   }
   event void Timer0.fired()
   {
      call Leds.led0Toggle();     //每一秒 Led0 閃爍
   }
}
```

- 編譯檔 Makefile

```
COMPONENT=hellotimerAppC
include $(MAKERULES)
```

實驗練習

1. 請參考 Timer<TMilli>介面中所提供的指令。
2. 請在上述的程式中，另外加上三個 TimerMilliC 元件，分別控制四個 Led 燈的閃爍，各別以一秒、五秒、十秒和十五秒閃爍。
3. 請改寫第一顆 Led 燈，使用 startOneShot(uint32_t dt)指令，只計時一次。
4. 請另外寫一個變數作為計數器，作為紀錄燈號閃爍的次數。當燈號閃爍超過 50 次，則停止燈號的閃爍。

實驗 三 Serial Port 輸出

實驗目的

本實驗以 Octopus II 為開發平台，並且為入門的第三支程式。讓同學以自己編寫三個主要檔案及加入 Serial Port 的輸出控制，也就是 C 語言常見的 printf 指令，以瞭解 Octopus 的輸出與 PC 之間的連線關係。

熟悉 TinyOS 的人都知道，要在 TinyOS 上除錯是一件艱難的事。為了讓使用者方便可以自行除錯，本實驗介紹 Printf 的函數庫的使用。透過此函數庫，可以讓程式於執行過程中，輸出使用者要的資料到 PC 端的 Serial Port，而 PC 端只要執行特定的 Serial Port 接收軟體即可接收。

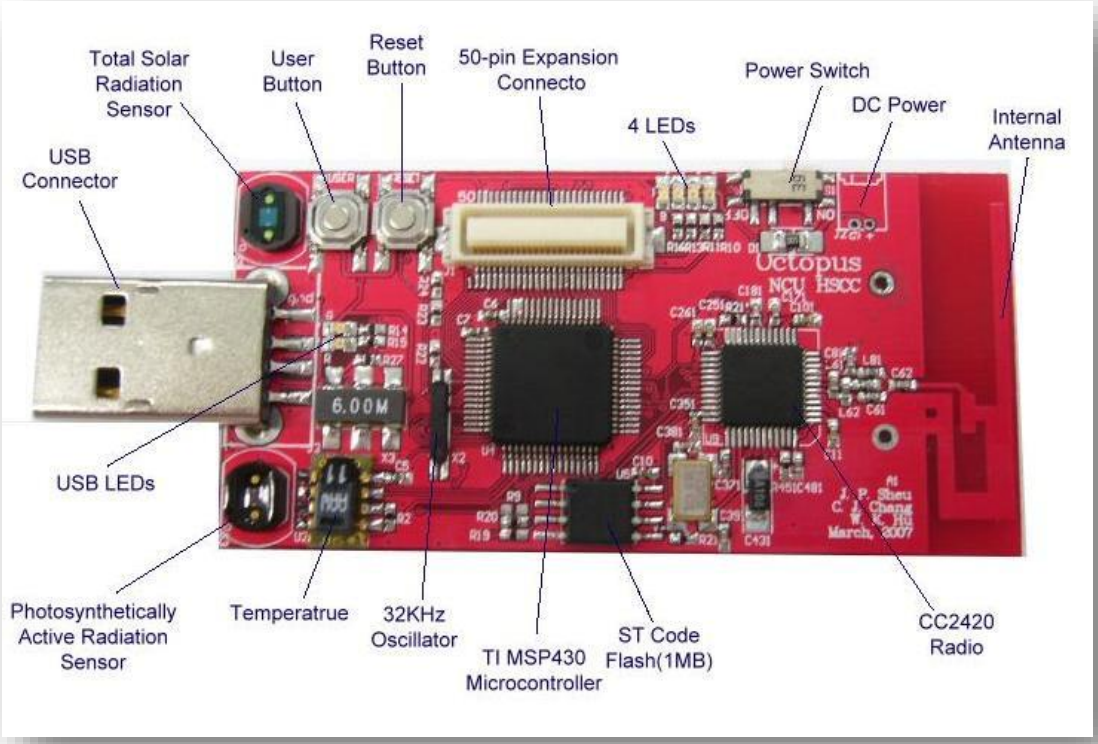
實驗器材

- Octopus II 開發平台
- TinyOS 開發環境請參考前一份實驗教材

實驗過程

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。



Step 2 瞭解 Printf 函數

Printf 函數庫只要在模組程式中加入下列的字串即可使用「#include “printf.h”」，而此函數庫包含的指令函數如下列：

printf()，也就是如同 C 語言中的列印指令

printfflush()則是刷新傳送的緩衝區，一次把累積在緩衝區中的資料送出到 PC 端的 Serial Port

一般來說，printf 所使用的緩衝區是 circular buffer，容量是 250 bytes，當緩衝區滿了或使用者執行「printfflush()」指令，則會自動執行刷新的動作。

Step 3 建立 testprintf 目錄

在實驗開始時，請先建立本實驗所需的目錄，可以利用兩種方式 (圖略，請參考實驗一)：

4. 首先執行 Cygwin 程式，並且改變目錄到「/opt/tinyos-2.x/apps」（使用 cd 指令）

5. 接著在 Cygwin 程式中，鍵入 「mkdir testprintf」，以建立一個新目錄。

6. 接著鍵入指令碼 ls，以查詢目錄是否正確建立。

或

3. 直接使用 Window 滑鼠，雙擊「我的電腦」，並點選到目錄「C:\cygwin\opt\tinyos-2.x\apps」
4. 按滑鼠右鍵以建立一個新目錄，名稱為「testprintf」

Step 4 testprintf 程式碼

以下為本實驗的程式碼，總共分成三個部分。除了讓 Octopus 模組亮 Led 燈表示開機成功，另外，則使用 printf 指令，讓模組在啟動後，會送出招呼的訊息到 PC 端。過程中，請使用小作家或其他文字編輯軟體，打開一個文件檔，並編輯和存檔下列程式碼到文字檔中，請記得副檔名必須一致。

1. TestPrintfAppC 程式

- 請於設定檔 TestPrintfAppC.nc 中加入 printf.h 標頭檔

```
configuration TestPrintfAppC{
}
implementation {
    components MainC, TestPrintfC, LedsC;     //呼叫所用到的元件
    TestPrintfC.Boot -> MainC;
    TestPrintfC.Leds -> LedsC;
}
```

- 請於 testPrintfC.nc 模組檔中加入 printf.h 標頭檔

```
#include "printf.h"
module TestPrintfC {
  uses {
    interface Boot;
    interface Leds;
  }
}
implementation {
  uint8_t dummyVar1 = 123; //宣告三個全域變數
  uint16_t dummyVar2 = 12345;
  uint32_t dummyVar3 = 1234567890;
```

```
  event void Boot.booted() {
    call Leds.led0On();
    printf("Hi I am writing to you from my TinyOS application!!\n");
    printf("Here is a uint8: %u\n", dummyVar1);
    printf("Here is a uint16: %u\n", dummyVar2);
    printf("Here is a uint32: %ld\n", dummyVar3);
    printfflush();                          //刷新緩衝區
  }
}
```

- 編譯檔 Makefile

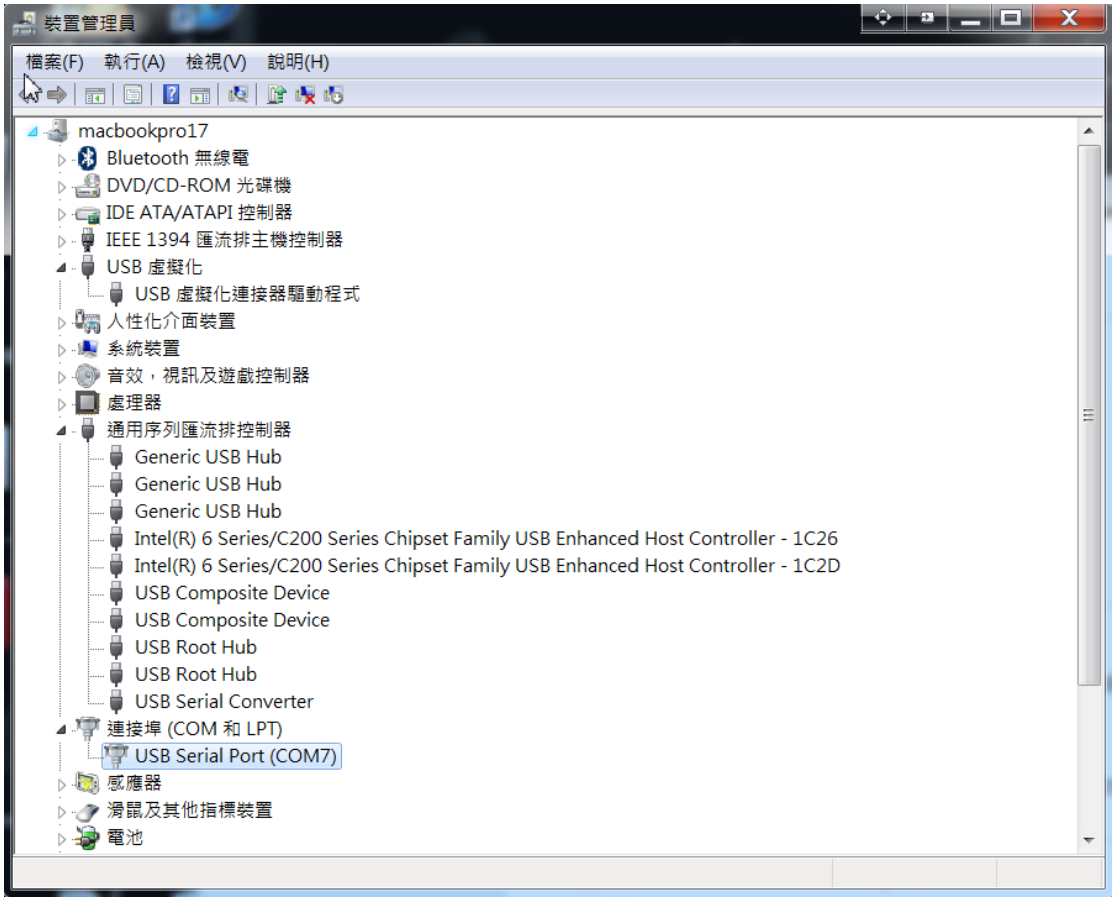
```
COMPONENT=TestPrintfAppC
CFLAGS += -I$(TOSDIR)/lib/printf
include $(MAKERULES)
```

實驗結果

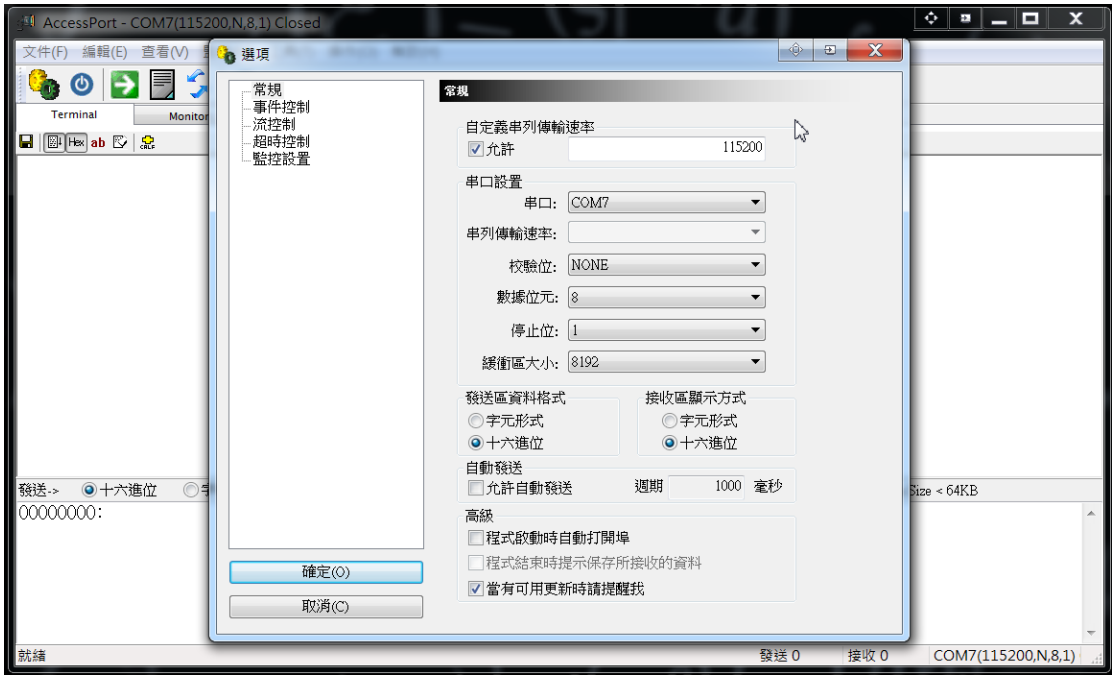
- 以下為實驗的編譯結果

```
/opt/tinyos-2.x/apps/printf
            6134 bytes in ROM
             531 bytes in RAM
msp430-objcopy --output-target=ihex build/octopus/main.exe build/octopus/main.ih
ex
    writing TOS image

XPMUser@VirtualXP-37273 /opt/tinyos-2.x/apps/helloworld
$ cd ..

XPMUser@VirtualXP-37273 /opt/tinyos-2.x/apps
$ cd printf

XPMUser@VirtualXP-37273 /opt/tinyos-2.x/apps/printf
$ ls
Makefile  TestPrintfAppC.nc  TestPrintfC.nc  build

XPMUser@VirtualXP-37273 /opt/tinyos-2.x/apps/printf
$ make octopus
mkdir -p build/octopus
    compiling TestPrintfAppC 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 -I/opt/tinyos-2.x/tos/lib/printf -DIDENT_APPNAME=\"TestPrintfAppC\" -DIDENT
_USERNAME=\"XPMUser\" -DIDENT_HOSTNAME=\"VirtualXP-37273\" -DIDENT_USERHASH=0xc3
df8991L -DIDENT_TIMESTAMP=0x4ea5ed64L -DIDENT_UIDHASH=0x5e843947L  TestPrintfApp
C.nc -lm
    compiled TestPrintfAppC to build/octopus/main.exe
            8020 bytes in ROM
             541 bytes in RAM
msp430-objcopy --output-target=ihex build/octopus/main.exe build/octopus/main.ih
ex
    writing TOS image

XPMUser@VirtualXP-37273 /opt/tinyos-2.x/apps/printf
$ _
```

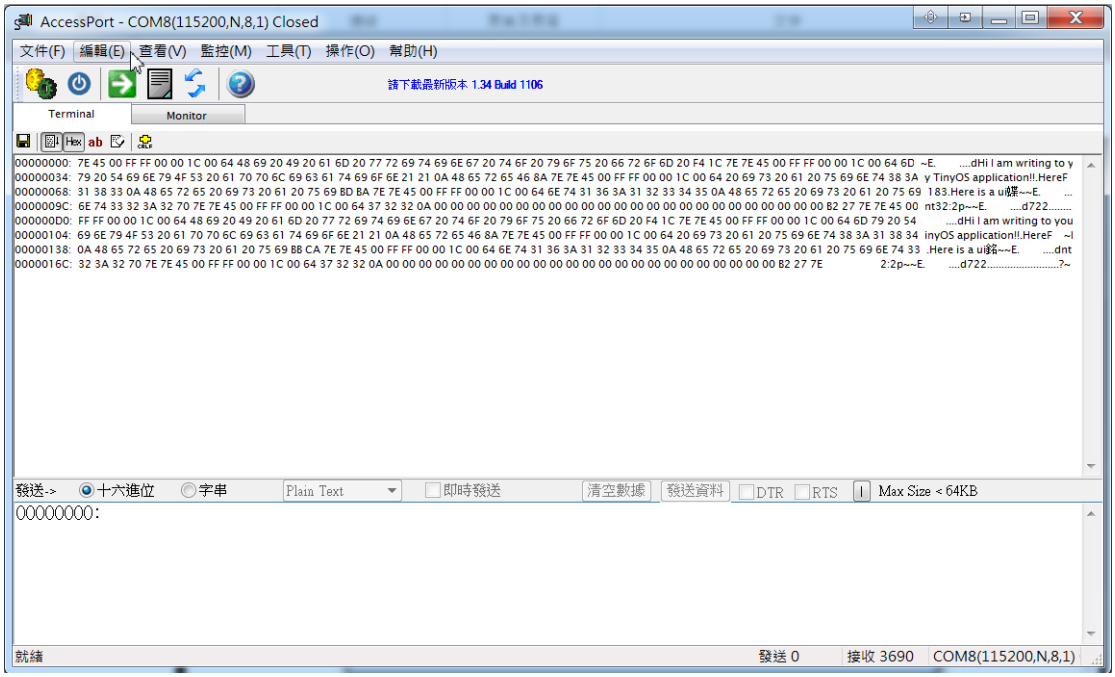
- 檢查 Octopus 的連接 Serial Port（以 USB 模擬 Serial Port）
  請到控制台/系統/裝置管理員，查詢連接埠選項，便可以看到在 PC 上面模擬的 Serial port 編號（Com port，在本範例中為 Com7）



- 推薦的 PC 終端機接收程式（Serial port 接收/監控軟體 AccessPort）
  (XP 內建的超級終端機比較不好用，而 Window7 已經沒有內建此軟體)
  下載位置 http://www.sudt.com/cn/ap/download.htm
  以下為程式設定畫面，請先點選參數配置選項，並修正如下圖：



當設定完畢後，請點選串口開關即可開始監控 Serial Port (Com 7)。



實驗練習

1. 請先設定執行四個燈號，每閃爍一個燈號，便傳送一個代表燈號的文字指令到 PC 端

2. 請設定每隔一段時間（例如三秒）執行的計數元件，每計數一次便傳送一筆訊號到 PC 端