EduCake 的入門介紹和 Digital 腳位功能的使用

一、 規格介紹與 IO 腳位

在日常生活中，各式各樣的電動玩具、電器、遙控、或工廠裏生產線的控制、各種機器的控制，都存在著單晶片。但對於一般非電子相關科系的人來說，學習使用單晶片有著很大的困難，常常需要熟悉各種電子零件的功能、電路的架構、複雜的低階程式撰寫等等，得要不少時間的工夫養成，初學者常常接觸後不久就放棄了。

這狀況終於在數年前獲得較好的解決方式，由義大利工程師所開發出 Arduino 開放式架構單晶開發平台，依舊屬於單晶片，但不同的是 Arduino 所內含與設計的開發平台非常適合初學者使用，不論是語法精簡、介面和善、或是中文書的輔助、還是網路資源眾多、相關零組件或功能模組價格便宜、易取得等皆是所具備之多項優點，更重要的是電路和資源的完全開放，促使這幾年間廣泛的、大量的被運用在教學和專題上使用。

由台灣廠商獨立開發設計、自行製造的 86Duino 系列控制板基於這樣的條件下問世了。這個由台灣廠商獨立研發製造的 86Duino 系列卻擁有如下的驚人規格:

- 300MHz 的 32-位元架構 x86 平台 CPU 作核心
- 內建 128MB DDR3 高速記憶體
- 有 PC 等級的區域網路 LAN 介面
- USB 2.0 * 2
- Micro-SD
- Open-Source Hardware

- Support DOS, Windows, Linux
- 提供 Arduino-完全相容的開發 IDE 介面，這一項，使得 Arduino 的使用者不用再花費時間學習新的東西，直接拿來用，原有 Arduino IDE 所附帶的數十個範例程式, 全部都不用修改就可在 86Duino 上執行；許多 Arduino 自帶的函式庫, 在 86Duino 上也都有支援。

86Duino 系列裡面的 EduCake 教育版本，直接和麵包板結合，從外觀清爽明義，搭配噴沙金屬外殼及白色的麵包板，十分有質感，是一個非常適合用來實作和教學用的微電腦控制板。

圖 1-EduCake 立體圖

圖 2-EduCake 背面

圖 3-EduCake 正面

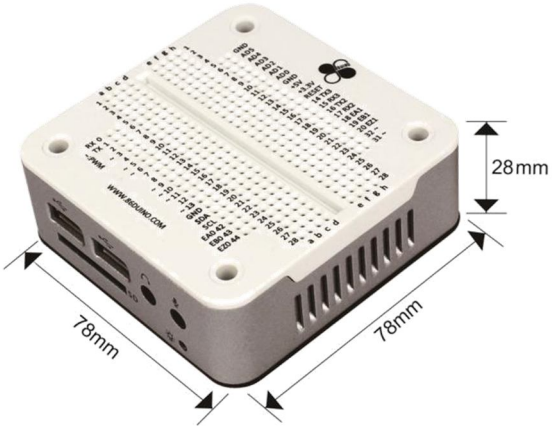

圖 4-EduCake 尺寸圖

板子上可以清楚看到 Digital 0~13、Analog0~5、GND、5V、3.3V、RX/TX 等 Arduino Leonardo 完全一模一樣的腳位(功能和程式寫法也都相同，一個字都不用改)。

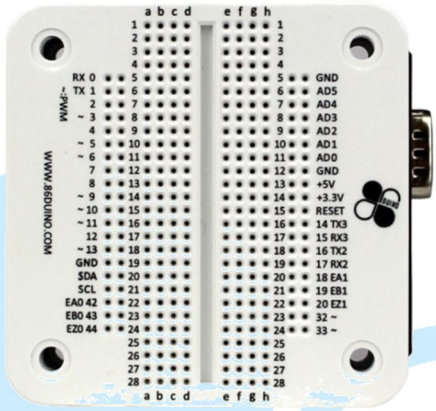

圖 5- EduCake 上所設置的腳位圖

主要腳位和接頭介紹如下:

1. Digital0~20, 31,32, 42~44，共 26 個 digital 可用來作為一般數位 IO 的使用，電流量最大 16mA，都有限流保護，防止不當使用的故障，讓初學者更可放心大膽嘗試各種應用功能。

2. Analog 0~5 做為類比輸入使用。

3. TTL 共有三組，分別是 RX/TX、RX2/TX2、RX3/TX3，對於各種通訊應用非常方便

4. 腳位旁邊有標示"~" 符號者為支援 PWM 的腳位，此部分與 Arduino 相同，但 86Duino 所出的 EduCake 多了 Digital 13、31、32 等三個腳位也是支援 PWM。

5. 板上多了 I²C 專用的 SCL/SDA 腳位，不用像 Arduino 328 要使用 I²C 就必須犧牲 Analog 4、5 兩腳，方便許多。

6. EA0~1、EB0~1、EZ0~1 是專門 ENCODER 給 Motion Control 用，此功能 Arduino 可就得要另外加控制板才可作到。

7. 5V 的腳位是直接 By Pass、3.3V 最大輸出 800mA

想必原先 Arduino 玩家看了之後也會覺得很親切，不止操作上相同，還額外增加許多功能，尚未使用過的玩家也可快速上手；在外觀尺寸上大小適宜，剛好一隻手可掌握；而金屬外殼將相關電路包裹住，兼具保護及美觀之功能。

二、 開發介面

說到 86Duino 開發介面即可十足可以感受到 DMP 開發團隊的用心，畫面除了顏色之外，與 Arduino 標準 IDE 介面一模一樣，操作方式也當然相同了，真是使用者一大福音!!以下是正式版本的 IDE 畫面，是不是感覺很熟悉呢？

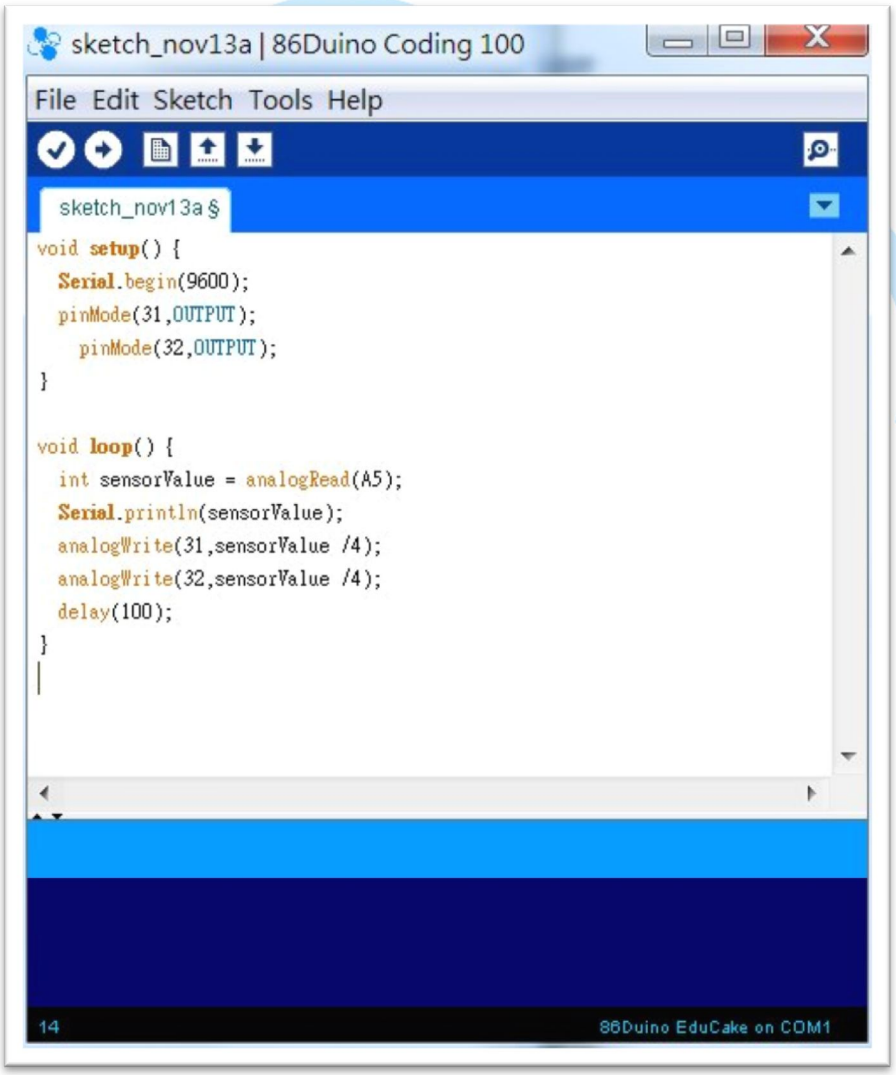

圖 6-開發介面與 Arduino 的介面十分相似

再來，看到 File-> Example 裡面的預設範例都差不多

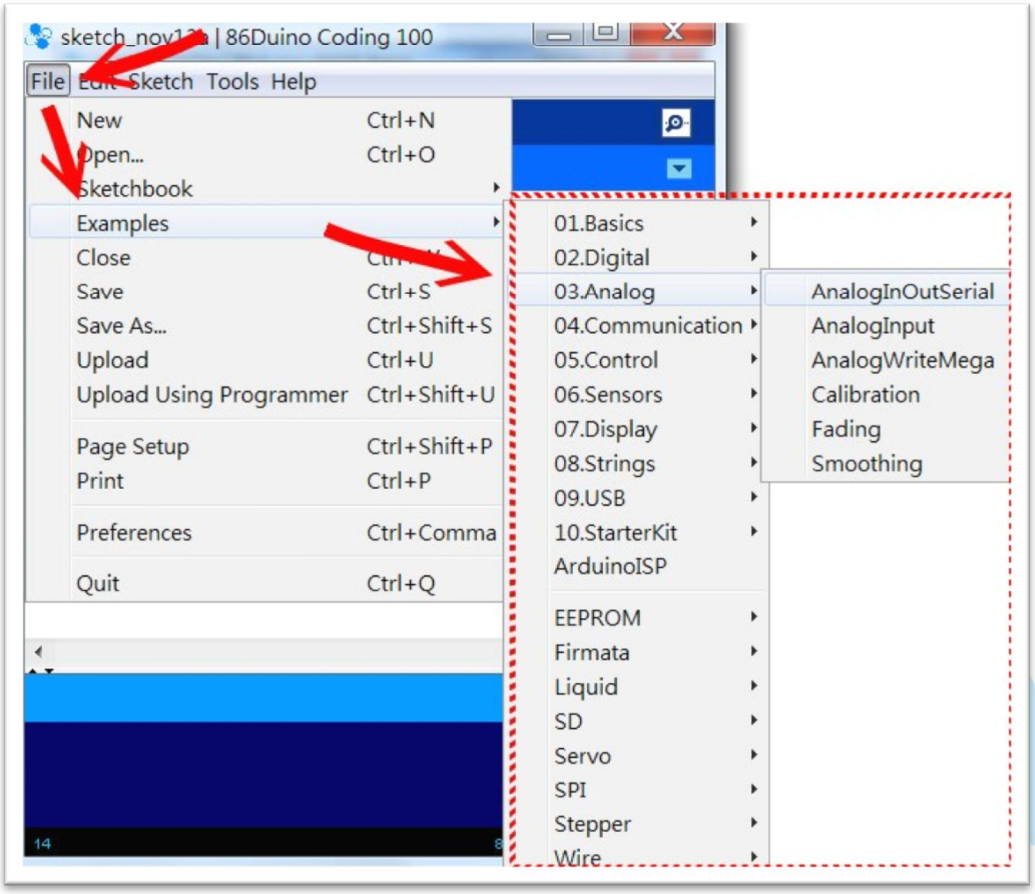

圖 7-開發介面操作模式也很相近

若是需要相關的指令參考，http://www.86Duino.com 網站中都有詳細介紹，基本上就是 htpp://Arduino.cc 官方網站裡面的 reference 相同的東西，若是已經看過的玩家可以直接跳過。當然，很多時候寫程式都是需要參考，有很多的基本應用範例、接線圖等等可以採用，另外也可以當作指令速查字典使用，很方便。

不過 86Duino 開發 IDE 的檔案目錄結構和 Arduino 的結構是有落差的，但對我們目前開發運用沒有任何阻礙，若是有興趣想要自己改函式庫的讀者可能得要先重新熟悉一下，這部分後面會有更深入介紹。

程式寫好以後要放到板子上的流程都相同，唯一要注意的是板子必須是 86Duino EduCake 這個選項，不要弄錯了（如下圖）。

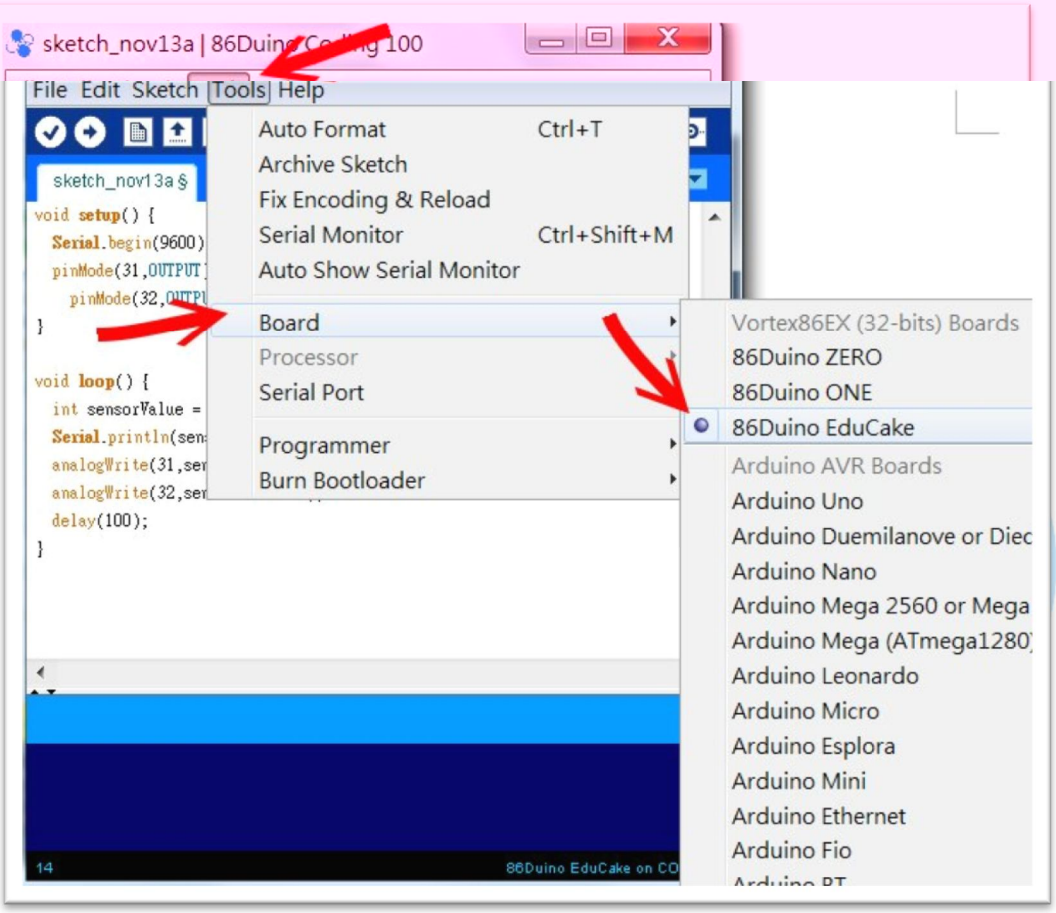

圖 8-開發介面 板子名稱請選 86Duino EduCake

板子選擇好以後，SerialPort 的選擇也要正確，筆者的號碼是 COM1。COM 的號碼可以隨時在控制台裡面的系統-->裝置管理員作任意修改。請注意這個畫面右下方的文字: 86Duino EduCake on COM11，意思是 IDE 介面抓到板子是在 COM11，但我已經修改成 COM1，這部分使用前須要再三確認正確才行。

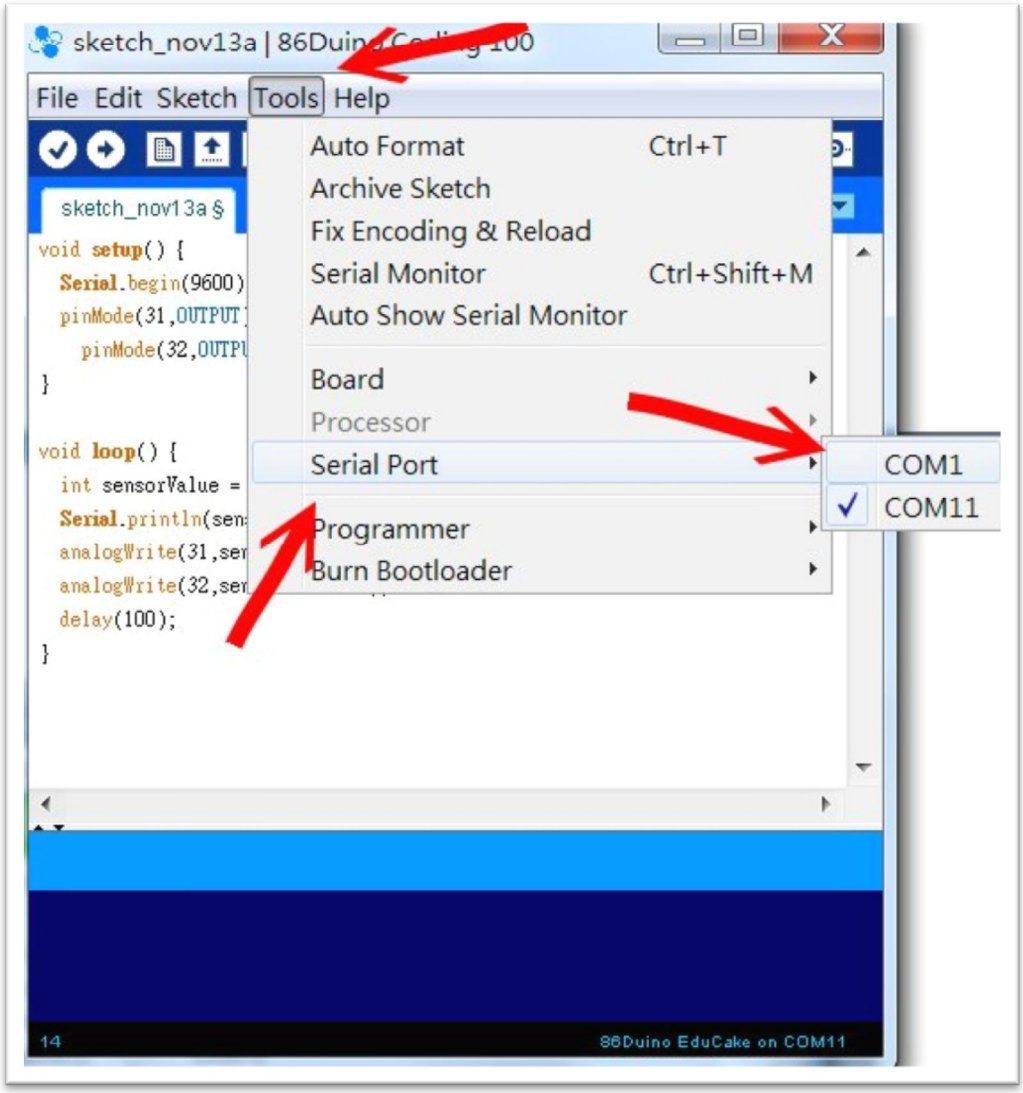

圖 9-開發介面 COM11 設置

選好 COM 以後，直接點下這個向右箭頭就可以開始上傳程式到 EduCake 板子上，開始跑囉。至於左方那個打勾符號是 Verify 程式的功能，筆者幾乎從來不用的，主要是，反正寫好上傳的時候就會執行檢查、編譯、上傳的連續動作，有沒有錯也都會知道，不需要額外作這個動作。

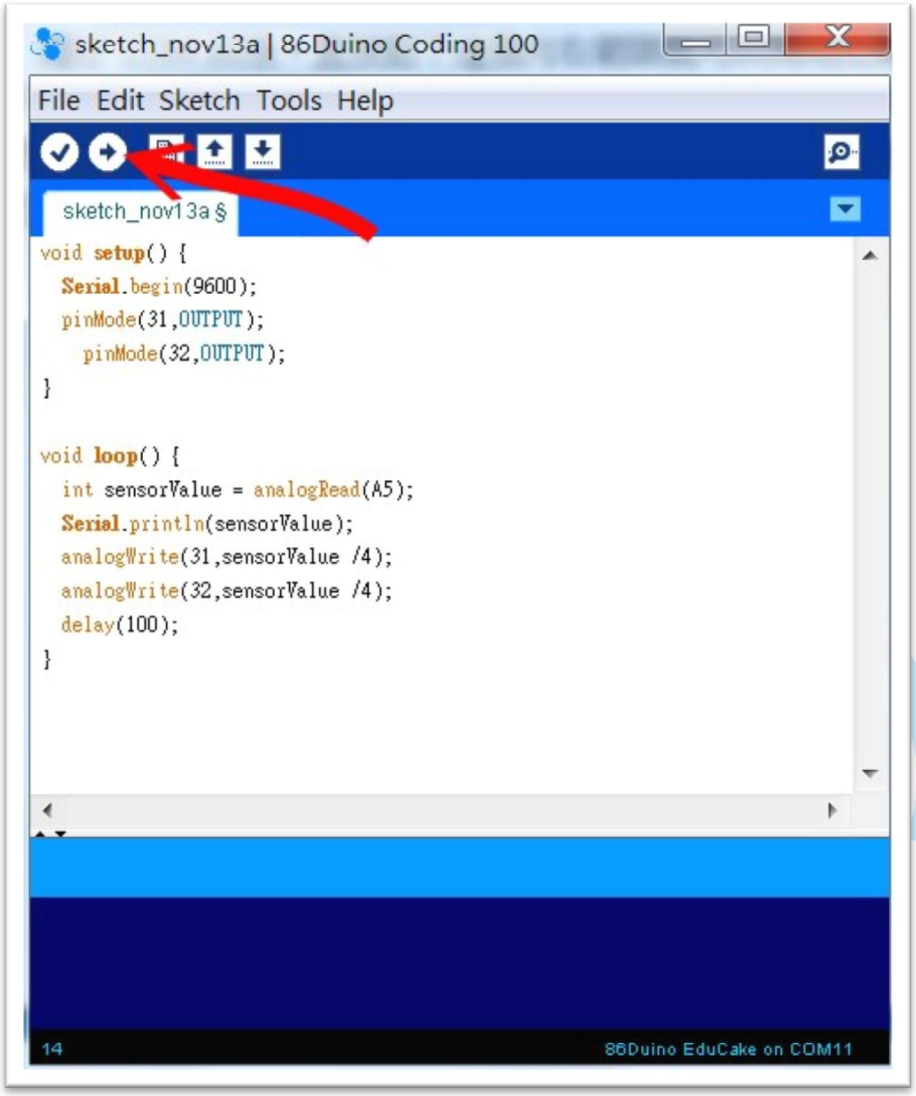

圖 10-開發介面 開始寫程式了！

三、 第一個程式

先來實作一個最簡單的程式試試看。首先，準備一個 LED 和一個電阻，像下圖這樣把線路接好。電阻不用也沒關係，因為板子的腳位電流很小，即使電壓超過電阻需求也不會燒掉的，只是加電阻來保護 LED 是使用各種電路的好習慣，最好還是繼續維持。

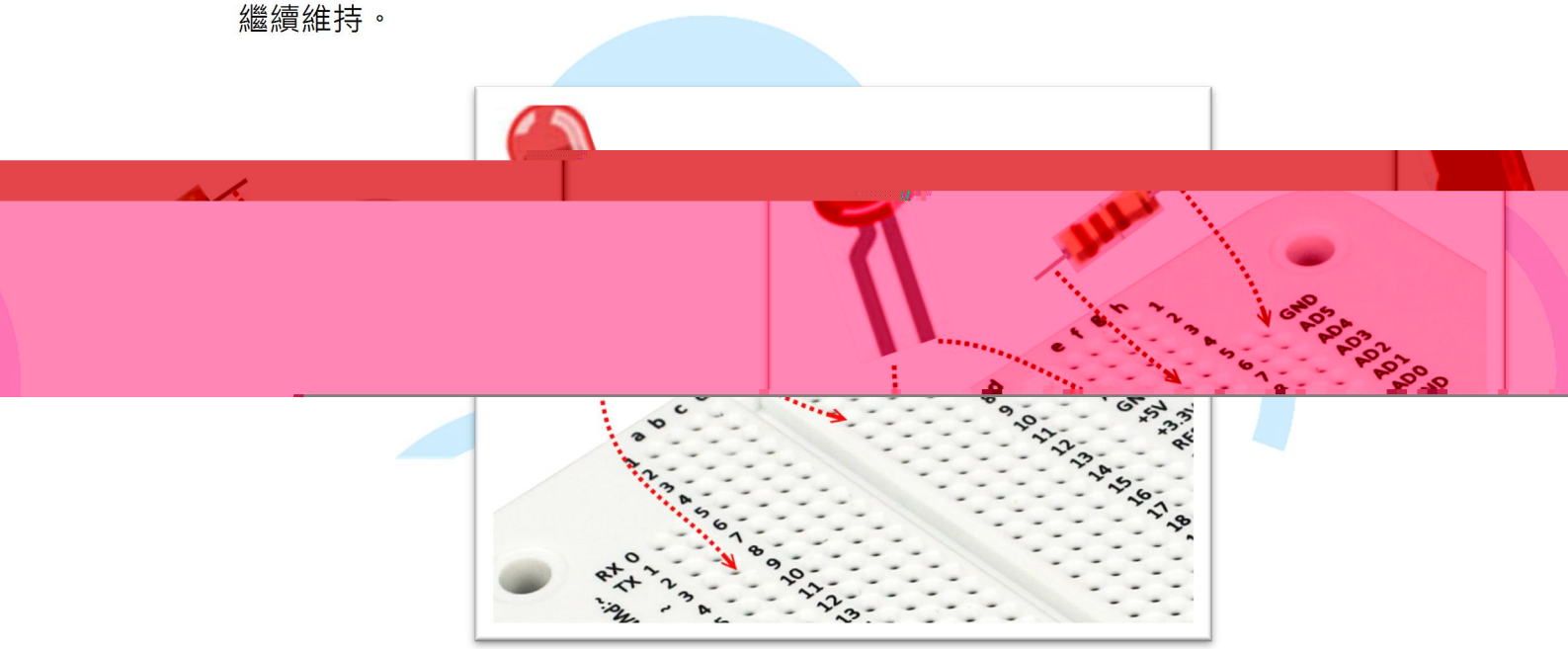

圖 11-將準備好的 LED 及電阻 接上 EduCake

請注意 LED 的正極接在 digital 腳位 3；然後開啟 IDE 介面，輸入以下程式:

```
void setup()
{
  pinMode(3, OUTPUT); //  設定腳位 3 為輸出模式
}
void loop()
{
  digitalWrite(3, HIGH); //  設定腳位 3 HIGH，LED 燈會亮起來
  delay(1000);
  digitalWrite(3,LOW); //  設定腳位 3   LOW，LED 燈會熄滅
  delay(1000);
    }
```

上傳程式後就可以看到 LED 燈間隔一秒鐘的時間在一閃一滅，循環不斷，這個範例是否和你寫過的 Arduino 程式一模一樣呢，簡單吧。後續還可以修改這個程式，使用 Digital 腳位的 PWM 功能，搭配 analogWrite 指令就可以控制燈光的慢慢變亮或是慢慢變暗，或是加入稍微的亂數，來製造簡單的燭光搖曳的效果，您會發現，光是一個 LED 燈都可以玩出無數變化呢。

到目前為止，使用方式跟 Arduino 是完全一模一樣的，程序也很簡單，對於有經驗或是第一次玩這個的讀者來說，應該能完全的上手了才對。

四、　第二個程式

第一個程式會動以後，我們就可以好好的來試驗這塊板子的功能了。把第一個程式延伸，多使用幾個腳位來做 LED 的效果，不過因為考量麵包板的空間安排，此次使用八個 Digital 腳位來接 LED，接腳圖如下:

首先，接好八顆 LED 和電阻

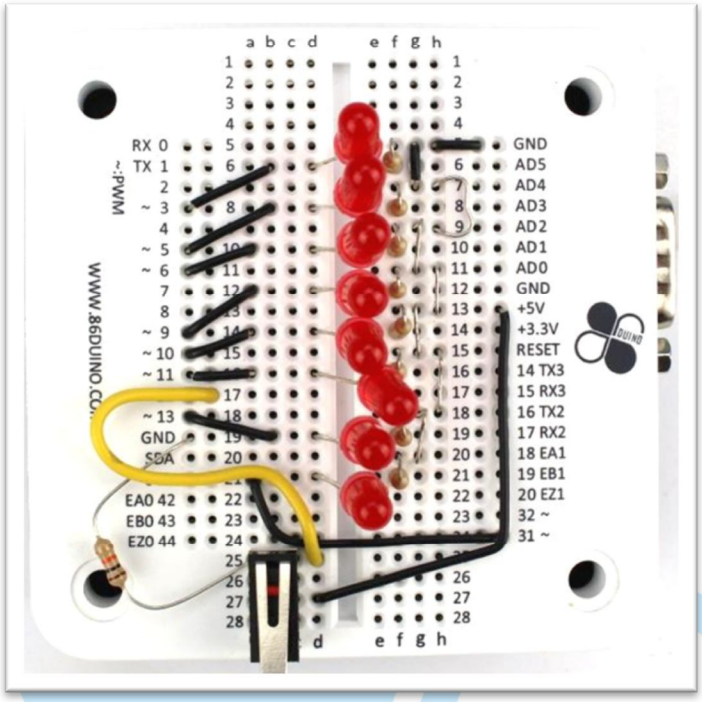

圖 12-接好 8 個 LED 及電阻，背面圖

換個方向看看

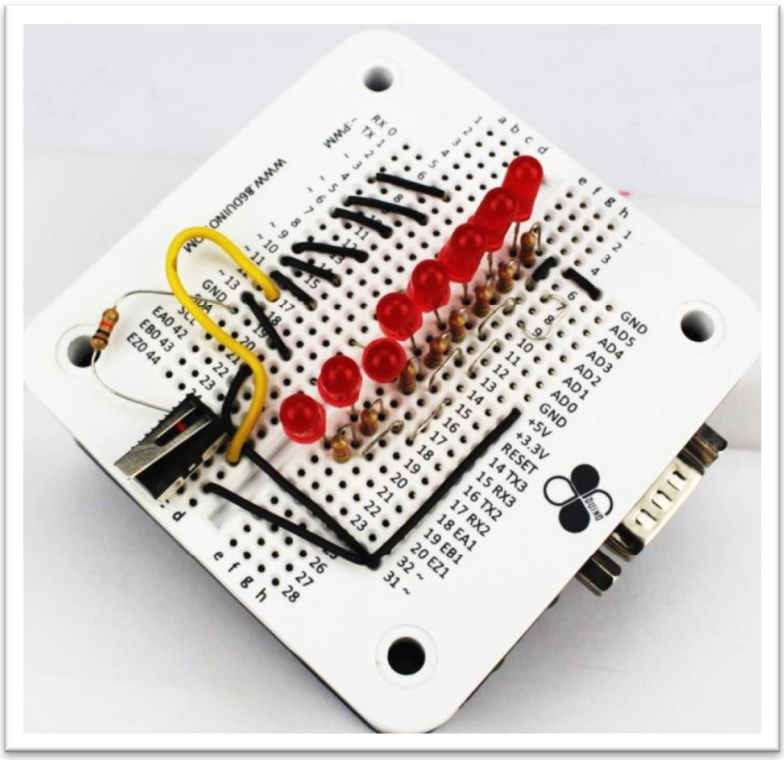

圖 13-接好 8 個 LED 及電阻，換各角度看一下

考量到後面需要用到按鈕來作別的事情，也把按鈕線路先接好

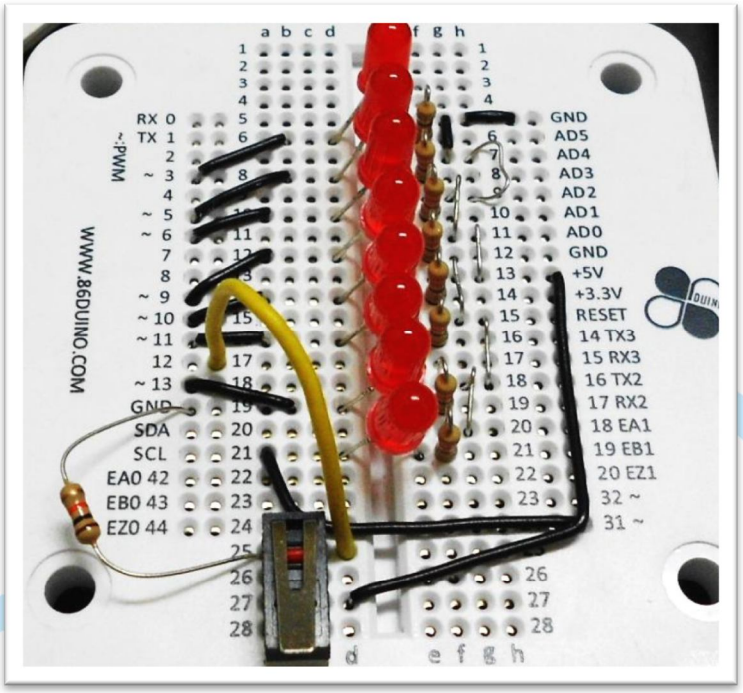

圖 14-將其他線路也一併接好

接好電路以後，即可利用這個電路來實做很多種不同的東西，先來做個可以類似跑馬燈的程式，讓八個 LED 燈連續不斷的循環亮起，程式碼如下:

```
// 動用這八個 Digital 腳位
// 這裡故意選擇 PWM~符號的腳位，這樣後面還可以延伸很多的應用
// 而不用一直改電路，很方便教學工作的進行
int led[]={
  3,5,6,9,10,11,13,31}; // 把八個腳位的號碼先存到陣列裡面, 方便存取
int pos =0;
void setup() {
  for(int a=0;a<8;a++) // 先設定這八個腳位為輸出模式
    pinMode(led[a],OUTPUT);
}

void loop() {
  digitalWrite(led[pos],LOW); // 把前一個燈熄滅
  pos=(pos+1)%8; //計算下一個要亮的燈位置, 以便輪流亮
  digitalWrite(led[pos],HIGH); // 點亮下一個燈
  delay(200); // 延遲一下，這個數字調的越大越慢，越小越快
    }
```

這樣就可以看到八顆燈輪流的一顆一顆亮下去，不斷的循環

```
    pos=(pos+1)%8;
```

其中那行

```
pos ++;
    if (pos>=8) pos =0;
```

其實是以下程式碼一樣的東西，要如何寫就看個人習慣了

最後那行 delay(200)的指令，也可以利用變數去變化他，來做出本來跑很快，然後漸漸變慢的效果等等。

五、 第三個程式

前面的程式能夠動以後，就可以開始做更多的變化，接下來換利用 COM 的通訊功能來跟電腦進行溝通。電腦端也需要對應的應用程式，這樣就可通過這個介面從電腦端來控制 EduCake。這個應用必須分兩邊寫程式。

一邊是在 EduCake 上面寫以下這段程式:

```
int led[]={
  3,5,6,9,10,11,13,31};
int pos =0;
void setup() {
  Serial.begin(9600); // 這行主要是設定和電腦的 COM 通訊的速度
// 通訊速度一般用 9600 / 19200/38400/115200 等等
// 必須兩個裝置之間要一樣
  for(int a=0;a<8;a++)
    pinMode(led[a],OUTPUT);
}

void loop() {
  char ch;
  if (Serial.available()) // 檢查電腦端是否有訊息來
  {
    ch=Serial.read(); // 若有訊息, 一次讀取一個 byte
    if (ch>='1' && ch<='8') // 只處理數字 1~8 的訊息
      digitalWrite(led[ch-49],HIGH);// 點亮對應的燈
  }
  delay(200);
}
```

其中

```
      digitalWrite(led[ch-49],HIGH);
```

ch-49 代表的意思是電腦裡面的每一個字元(不管中文、數字、英文或是任何其他國家的文字)都有對應的代碼，數字 1~8 的 ASCII 對應號碼分別是 49~57，即可利用這個簡單的減去 49 的計算，轉換成 0~7，正好對應到上面設定的 Digital 腳位的那個陣列內容

好了以後，就可以利用 IDE 介面裡面的 COM 監視器(Serial Monitor)來和電腦通訊，輸入 1~8 來控制對應的燈號亮起來，畫面如下

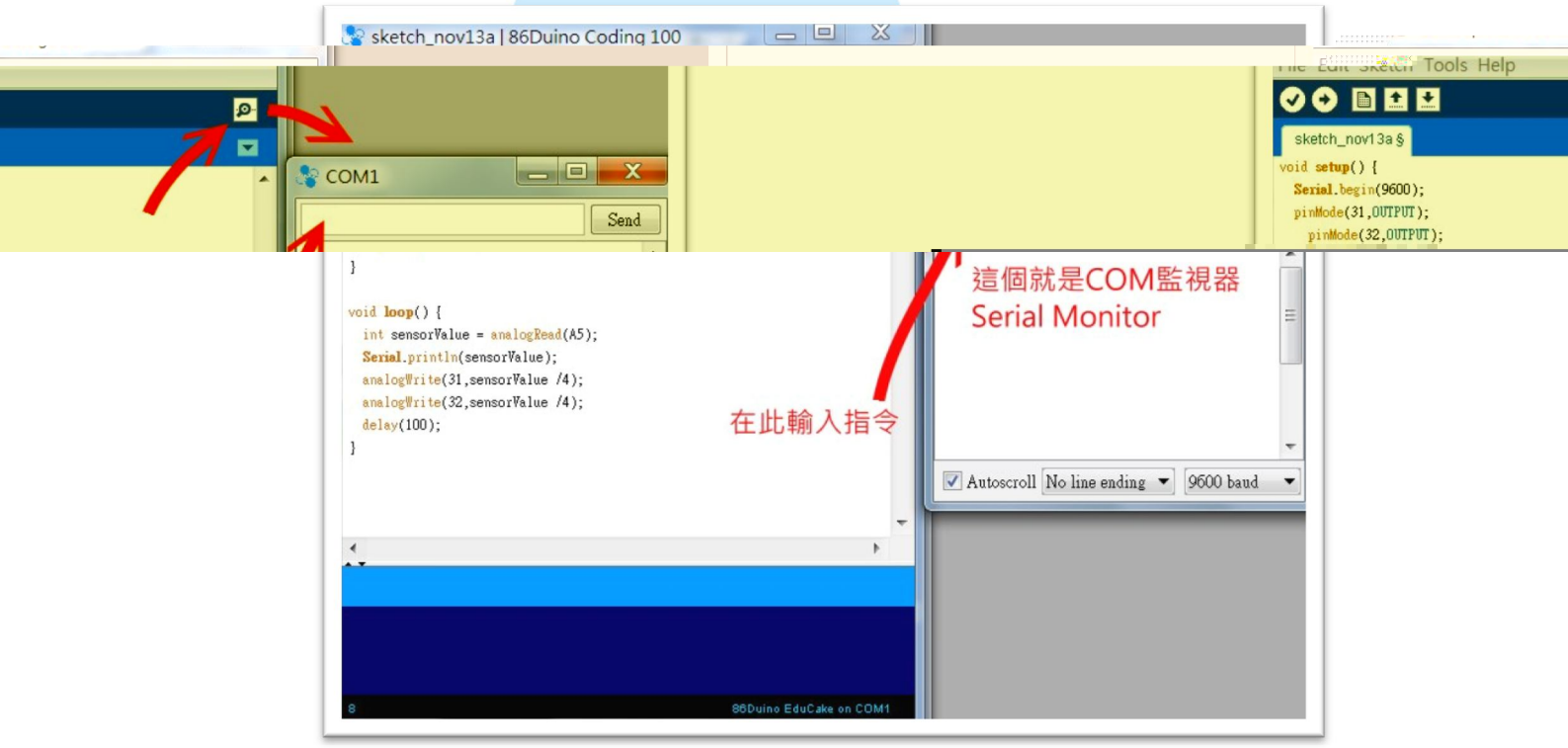

圖 15-輸入指令

可在實心箭頭處輸入 1～8 的數字直接控制對應的 LED 燈亮起來。這個畫面很常被使用在程式開發過程中，輸出執行過程的訊息，用來作邏輯的偵錯還滿方便的；而且也常用在通訊功能的測試部分。

不過要注意會占用 COM，所以若像後面會使用 VB/C#來寫 UI 畫面去控制 EduCake 的時候，要注意這個 Serial Monitor 就不能開，不然程式會有衝突。所以若是這種情況，UI 執行前要記得關閉這個畫面，否則有些嚴重的情況下，電腦得重開機才能重新讓 UI 抓到 COM

也可以用任何能寫 COM 通訊的程式來和他通訊，我們這裡使用 VS2008 版本(2005~2013 版本都可以使用不會有錯誤)的 VB.net/C#來試試看，畫面佈置如下

圖 16-LED 控制

首先，設定好 COM Port 和 9600 的傳輸速率，按下連線按鈕，就可以和 EduCake 作溝通。

這部分因為屬於視窗程式，不在本文章討論範圍，就請自行觀看附件的 VB 程式了。若要有更多變化，也可以結合滑鼠的滾輪來作往前滾動，LED 就依序亮起；往後滾動 LED 就依序熄滅的應用。或是結合可變電阻來轉動，電阻越大就量越多。

後面學會 Analog 的腳位功能後，還能搭配來測試電池的電量，電力越強就亮起越多的燈...等等，應用的領域就端看您的想像能力到哪裡囉。

有些人需要 C#的版本，可以到網路上搜尋"VB2C#"這樣的關鍵字，就可以找到一些轉換的網頁，程式碼很簡單，轉換不會有什麼問題的。後續我們在專案中還會結合手機的版本來做更多的應用。

不過這個程式有點瑕疵，燈號亮起來就不會熄滅了，這樣變成若是按到已經按過的號碼，也不曉得到底是有沒有送出訊息來控制燈號。這問題很好解決，可以把

```
        digitalWrite(led[ch-49],HIGH);
```

修改成這樣

```
{
  digitalWrite(led[ch-49],HIGH);
  delay(1000);
  digitalWrite(led[ch-49],LOW);
    }
```

讓他亮一秒後就變暗，這樣的就可以讓整個程式很好的運作了。有人會問，那幹嘛不一次寫好?

我們必須了解到，學習的過程中，不可能直接有完美答案的，很多時候是一個功能先要求有答案，然後再慢慢的加入修正去讓他穩定運作，加入新的功能讓他慢慢的完善，這整個測試的過程也才能體會到真正的程式和控制的樂趣哩。而且更重要的是，往往一直修改的過程中，會一直體會到不一樣的新寫法，對功力的進步也是有幫助的。

六、 第四個程式

瞭解了前面的基本功能後，就可以來寫更完整實用的功能。不過因為這個章節討論 Digital，這裡就來作一個類似電動玩具機台上的那種猜數字燈，一開始 LED 會往某個方向一路亮過去又亮回來，速度越來越慢，慢到某種程度的時候，會隨機停在某一個 LED 發亮。這樣可以用來製作像是電子骰子、吃角子老虎之類的應用，並且和電腦 UI 作結合，用來控制他產生骰子號碼 1~8 號(因為有八個 LED 燈，當然讀者也可以自己改成對應普通骰子的六個燈)，Edu Cake 端的程式碼如下:

```
int led[]={
  3,5,6,9,10,11,13,31};

// 因為要製作類似貪食蛇那種效果的燈號移動方式，
// 所以這裡使用上三個變數用來控制燈號的發亮位置和方向
int nowPos =2; // 開頭發亮的燈
int midPos=1; // 中間發亮的燈
int lastPos=0; //尾部發亮的燈
int dir=1; // 燈號發亮移動方向，1:小到大, -1:大到小

int spd = 20; // 移動速度

void setup() {
  Serial.begin(9600);
  for(int a=0;a<8;a++)
    pinMode(led[a],OUTPUT);
  randomSeed(analogRead(0)); // 一開始先來初始化亂數種子
}

void loop() {
  if (Serial.available())
      {
```

```
        char ch=Serial.read();
          if (ch=='1') //  接收到 UI 端的命令就重新執行
          {
            spd =20;
            // nowPos =2;
            midPos=nowPos-1;
            lastPos=nowPos-2;
            dir =1;
          }
        }
        if (spd<220) //  速度越來越慢，設定 220 的時候停止移動，產生亂數結
果
        {
          digitalWrite(led[nowPos],HIGH); //  開頭位置最亮
          if (midPos<8 && midPos>=0)
            analogWrite(led[midPos],40); //  中間位置稍暗
          if (lastPos<8 && lastPos>=0)
            analogWrite(led[lastPos],15); //尾巴位置最暗

          delay(spd);

          spd +=5;
          //  速度越來越慢，設定 220 的時候停止移動，產生亂數結果
          if (spd>=220 )
          {
            if (midPos<8 && midPos>=0) //  以下這些把整排燈都熄滅
              digitalWrite(led[midPos],LOW);
            if (lastPos<8 && lastPos>=0)
              digitalWrite(led[lastPos],LOW);
            digitalWrite(led[nowPos],LOW);

            digitalWrite(led[random(0, 8)],HIGH);//產生亂數位置，把該位置亮
起來
```

```
    spd =1000;
    }
        if (lastPos<8 && lastPos>=0)
          digitalWrite(led[lastPos],LOW); //先熄滅尾巴位置，準備往前移動
        lastPos=midPos;
        midPos=nowPos;
        nowPos+=dir;
        if (nowPos>7)
        {
          nowPos=7;
          midPos=8;
          lastPos=9;
          dir=-1;
        }
        else   if (nowPos <0)
        {
          nowPos=0;
          midPos=-1;
          lastPos=-2;
          dir=1;
        }
      }

      delay(spd/3);
        }
```

最後那行

```
        delay(spd/3);
```

是依照 spd 變數來作 delay，這部分可以自行調整速度，或是改成使用 UI 來控制速度的變化，哪一種作法比較有趣就看讀者自行發揮了。

在來就是，如果沒有寫 UI 或是不想用 Serial Monitor 來控制，想要直接按按鈕來控制他重新跑一次，則可利用前面已經接好的按鈕(在 Digital 12)，在執行的過程中檢查該按鈕狀態，若有按下就重新執行。修改後的程式碼如下(重複的部分就用。。。省略了):

```
 。。。
void setup() {
   。。。
   pinMode(12,INPUT);   // 多這行，先設定 12 腳位為輸入
}

void loop() {
   int bb;
   bb=digitalRead(12); //  讀取 Pin 12 的狀態
   Serial.println(bb); //  順便印出來作除錯和其他用途
   if (bb==1) // 1 代表被按下
     run_again(); //呼叫重新執行的函數

   。。。
}

void run_again() //  在主 LOOP 外面寫這一段，程式重新執行的設定用途
{
   spd =20;
   midPos=nowPos-1;
   lastPos=nowPos-2;
   dir =1;
     }
```