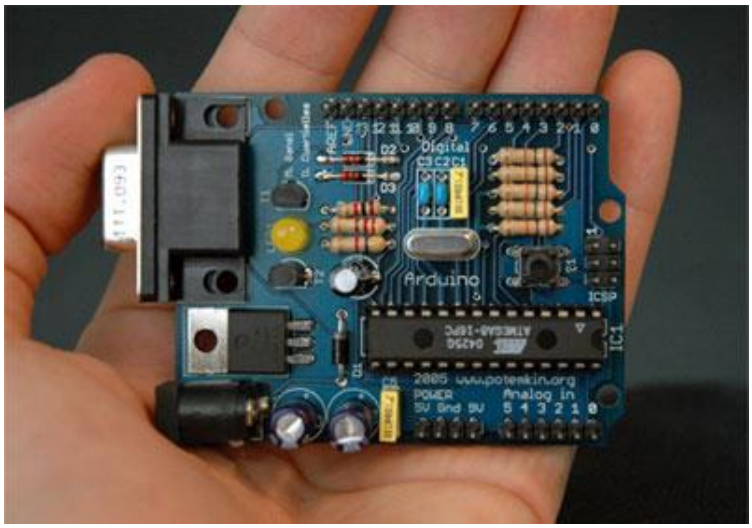
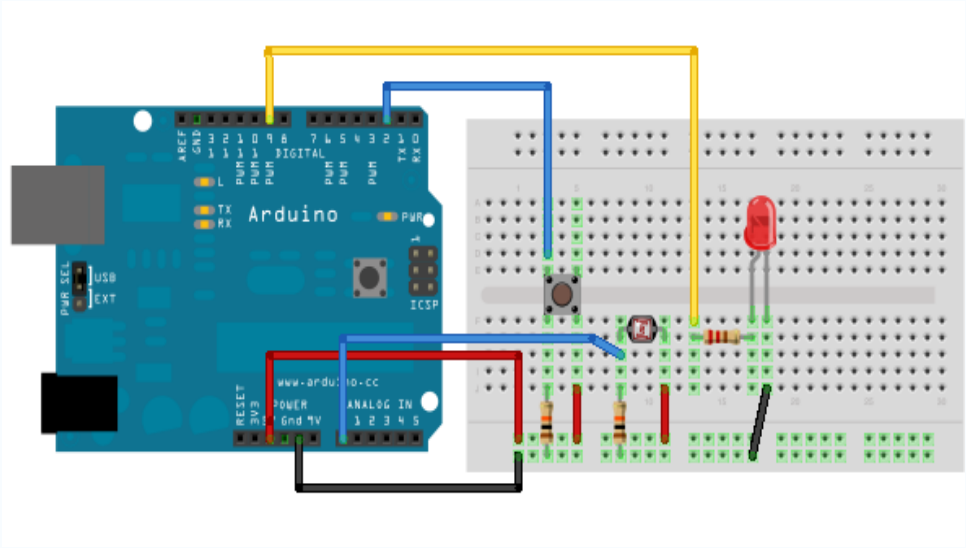
壹、課程說明

單元名稱	程式設計專題：Arduino 程式設計
單元摘要	Arduino 程式設計、微處理機控制板(MicroController board)、自動控制。
設計者	王鼎中(臺北市立建國中學)
學習目標	1. 了解什麼是程式設計 2. 了解 Arduino 程式開發環境。 3. 瞭解變數、運算式及內建函數等程式基本組成要件。 4. 熟習循序、選擇及重複等三種程式設計的基本結構 5. 透過作品展示與分享，展現同學們的創意。
課綱範圍	5.電腦與問題解決➡5.4電腦解題實作➡5.4.1電腦解題工具介紹 5.4.2解題實作
教學節數	8 節（400 分鐘）
先備知識	視窗環境操作、網路資料瀏覽與下載、基本邏輯概念。
評量方法	1. 簡單口頭問答 2. 課堂觀察 3. 作業練習與實作：可於課程進行中，請同學以教材中的範例進行練習，並嘗試完成牛刀小試所提供的例題，以檢視同學對課程內容的吸收程度。 4. 作品觀摩： (1)於每單元課程結束前，請同學以牛刀小試的練習結果進行展示與說明 (2)於課程結束時，安排專題製作的時間並請同學展示專題製作的成果
參考資源	1. 自編教材 2. Arduino 原創者 Massimo Banzi 著，台灣 arduino.tw 林義翔譯，踏進互動世界-使用 Arduino，旗標， 2009 3. Arduino 官方網站 http://www.arduino.cc/ 4. Arduino.TW 樂園 http://arduino.tw/

貳、教學活動計畫（敘述活動細節、教學工具、教學策略等）

教學活動	時間	說　明
一、Arduino 程式設計簡介	1 節	**1-1 什麼是程式設計（programming）** 編寫程式的目的主要是希望透過電腦的協助，來完成某些特定的工作，而完成這些工作通常都有一定的運作指令與程序，程式設計便是透過程式語言，將電腦完成工作所需的指令或步驟記錄下來，如此，程式執行時，電腦便可依據程式中的指令敘述，依序完成所設定的各項運算與動作，藉此完成所要執行的特定工作。 程式語言的種類很多，包含機器語言(Machine Language)、組合語言(Assembly Language)、高階語言(High Level Language)等，其中機器語言最為接近電腦所能理解的語言，而高階語言則最接近人類使用的語言，如一般常見的 C 語言、VB、JAVA 等，近年來，為提升初學者學習程式語言的興趣，並減低程式編寫時所產生的錯誤，許多研究機構研發出視覺化的程式開發工具，如：Alice、Scratch、Lego milestone 等，此外，透過適當的工具及開發環境輔助，學習者也能撰寫程式來控制簡單的電子元件，進而發揮創意創造出有趣的應用，如：Arduino、BASIC Stamp、OOPic、PICAXE 等。 **1-2 什麼是 arduino** Arduino，是一個基於開放原始碼的軟硬體平台，構建於開放原始碼 simple I/O 介面版，並且具有使用類似 Java、C 語言的 Processing/Wiring 開發環境。Arduino 包含一塊微處理機控制板(MicroController board)，以及一個可以將程式寫入到控制板的開發環境，使用者可透過微處理機控制板的連結來接收如：按鈕、開關、感測器(聲音、光線)等傳送進來的訊號，並控制如：LED 燈、喇叭、馬達等週邊電子元件的運作，經由簡單的電路組合，使用者將能透過開發環境所撰寫的程式來控制這些電子元件的運作，進而創造出充滿創意的有趣應用。  Arduino 微處理機控制板 (取自 http://arduino.tw/whatsarduino.html)



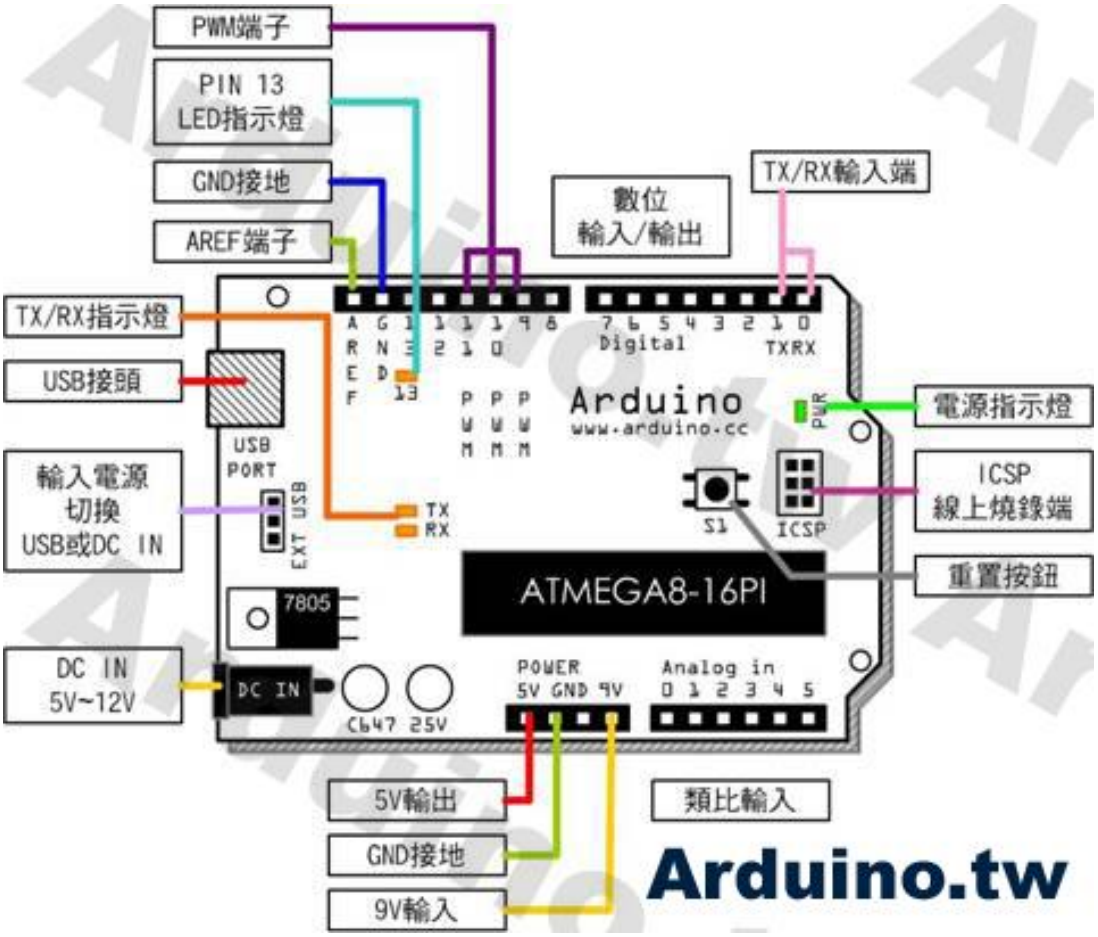
Arduino 的核心開發團隊成員包括：Massimo Banzi，David Cuartielles，Tom Igoe，Gianluca Martino，David Mellis 和 Nicholas Zambetti。據說 Massimo Banzi 之前是義大利 Ivrea 一家高科技設計學校的老師。他的學生們經常抱怨找不到便宜好用的微控制器。2005 年冬天， Massimo Banzi 跟 David Cuartielles 討論了這個問題。David Cuartielles 是一個西班牙籍晶片工程師，當時在這所學校做訪問學者。兩人決定設計自己的電路板，並引入了 Banzi 的學生 David Mellis 為電路板設計編程語言。兩天以後，David Mellis 就寫出了程式碼。又過了三天，電路板就完工了。這塊電路板被命名為 Arduino。幾乎任何人，即使不懂電腦編程，也能用 Arduino 做出很酷的東西，比如對感測器作出回應，閃爍燈光，還能控制馬達。隨後 Banzi，Cuartielles，和 Mellis 把設計圖放到了網上。保持設計的開放原始碼理念，因為版權法可以監管開放原始碼軟體，卻很難用在硬體上，他們決定採用創用 CC (Creative Commons)許可。

創用 CC (Creative Commons)係基於「保留部份權利」（Some Rights Reserved）的概念。以模組化的簡易條件，透過 4 大授權要素(姓名標示、非商業性、禁止改作、相同方式分享)的排列組合，提供了 6 種便利使用的公眾授權條款。創作者可以挑選出最合適自己作品的授權條款，透過簡易的方式自行標示於其作品上，將作品釋出給大眾使用。

在創用 CC 許可下，任何人都被允許生產電路板的複製品，還能重新設計，甚至銷售原設計的複製品。你不需要付版稅，甚至不用取得 Arduino 團隊的許可。然而，如果你重新發佈了引用設計，你必須說明原始 Arduino 團隊的貢獻。如果你調整或改動了電路板，你的最新設計必須使用相同或類似的創用 CC 許可，

以保證新版本的 Arduino 電路板也會一樣的自由和開放。唯一被保留的只有 Arduino 這個名字。它被註冊成了商標。如果有人想用這個名字賣電路板，那他們可能必須付一點商標費用給 Arduino 的核心開發團隊成員。(節錄自維基百科：http://zh.wikipedia.org/wiki/Arduino)

1-3 Arduino 微處理機控制板(MicroController board)



Arduino 微處理機控制板 (取自 http://arduino.tw/whatsarduino.html)

Arduino 微處理機控制板採用 Atmel Atmega8-16PI/PU 單晶片作為系統核心，可透過 USB 接頭與個人電腦連接，作為程式傳輸以及供電之用，透過 DC IN 接頭可外接 5V~12V DC 輸入來供電，提供 Arduino 獨立運作所需之電源，而透過電源輸出接孔則可提供 5V 的電源輸出，用以驅動週邊電子元件。

在控制訊號輸出入方面，Arduino 微處理機控制板提供了數位訊號及類比訊號的輸出入接腳，作為與週邊電子元件傳輸及接收訊號的管道。

- 數位輸出/輸入接腳 (Digital I/O pins)

  位於板子上方的 Digital 區，共有接腳 0-13 等 14 個接腳可供使用，可透過程

式設定各個接腳是用於輸出或是輸入。

- 類比輸入接腳 (analogue In pins)

位於板子下方的 Analog In 區，共有接腳 A0-A5 等 6 個接腳可供使用，用以讀取各種類比輸入裝置(如光源感測器等)的輸入訊號，並將之轉換為 0 至 1023 的數值，供後續程式控制使用。

- 類比輸出接腳 (analogue Out pins)

板子上方的 Digital 區中的 9、10、11 等 3 個接腳，可經由程式的指定，變更形態為類比輸出，可輸出介於 0-255 的類比訊號。

本文中所採用微處理機控制板為 Arduino UNO。

1-4 整合開發環境軟體(IDE，Integrated Development Environment)

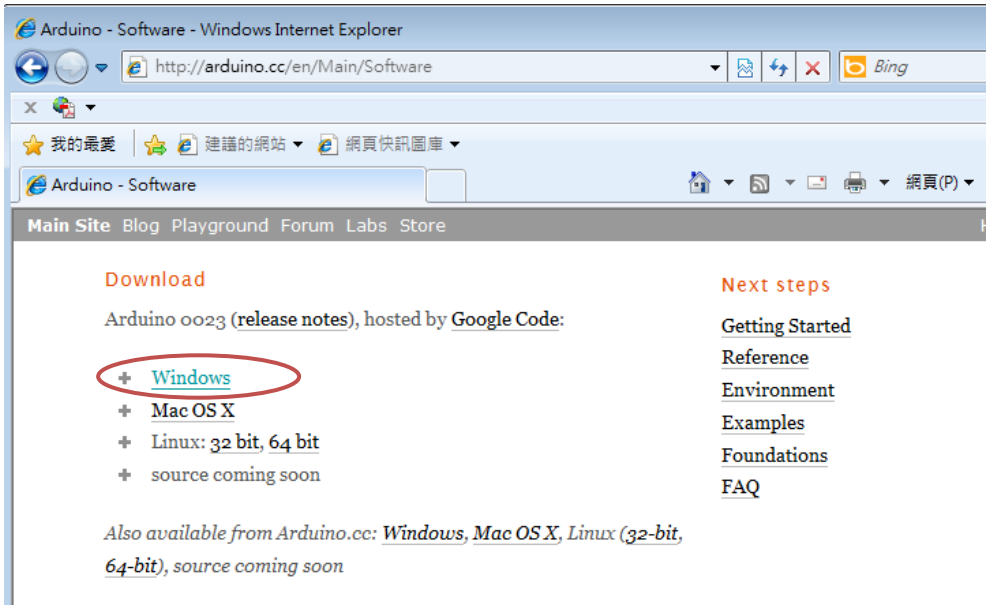
Arduino 的整合開發環境軟體，是用來撰寫要讓 Arduino 微處理機控制板所要執行程式的工具，它所採用的簡單程式語言，是由 Processing 程式語言 (www.processing.org)修改而來，程式語法和常用的 C 語言及 JAVA 語言相近，但相較起來簡單許多，大幅降低初學者的學習障礙。

- 下載 Arduino 整合開發環境軟體

可連結至 Arduino 官方網站免費下載使用，網址為：http://arduino.cc/en/Main/Software ，網站中提供了適用於 Windows、Mac OS X

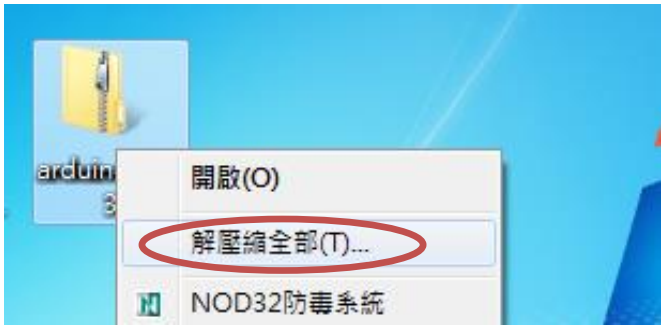
以及 Linux 等不同作業系統的版本可供選擇，本文以 windows 版本為例，目前版本為 Arduino 0023 版。

下載：



解壓縮：

將下載的 arduino-0023.zip 解壓縮到 D:\ arduino-0023\資料夾中(資料夾路徑可依個別需要設定)。



- 利用 USB 線將 Arduino 板與電腦連結，並安裝驅動程式
  用 USB 線將 Arduino 板與電腦連結後，尋找新增硬體精靈將會詢問是否搜尋 Windows Update 網站，請選(不，現在不要)並按(下一步)繼續。接著，點選(從清單或指定位置安裝))並按(下一步)，勾選(搜尋時包括這個位置)，按(瀏覽)鈕並選取 D:\arduino-0023\drivers\資料夾，按(確定)鈕再按(下一步)即會開始進行驅動程式安裝，如此便完成 Arduino 板與個人電腦的連結。

  在裝置管理員視窗的樹狀列表中，點選連接埠(COM 和 LPT)前的+號將之展開，找到 Arduino UNO 裝置，並將後方的 COM 編號記錄下來，如：COM3。

- 開啟 Arduino 整合開發環境軟體，並進行相關設定
  打開 D:\arduino-0023\資料夾，找到 arduino.exe 後，雙擊開始執行軟體後，進行 Arduino 板型號及連接埠的設定。

  設定 Arduino 板型號

  點選 Arduino 整合開發環境軟體上方的 tools 功能表，點選 Board 並將之設定為 Arduino UNO。

  設定連接埠埠號

  點選 Arduino 整合開發環境軟體上方的 tools 功能表，點選 Serial Port 並將之設定為 COM3。

- 測試
  開啟範例程式 Blink

  點選功能表 **File > Examples > 1.Basics > Blink**，載入 **Blink** 範例程式

  將程式上傳至 Arduino 板執行

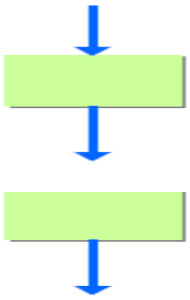
  點選按鈕工具列中的(Upload)，將 Blink 程式上傳至 Arduino 板

  幾秒鐘後將會看到 arduino 板上的 TX 及 RX 燈在閃爍，同時開發環境下方的 status 狀態列的訊息由(Uploading to I/O Board)轉變為(Done Uploading)，即表示上傳程式的動作已經順利完成。

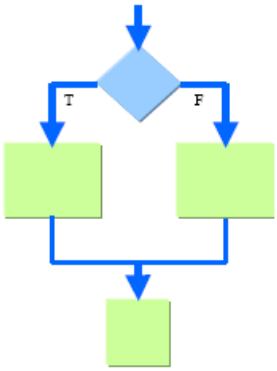
  上傳完成的幾秒鐘後，若在 Arduion 板上的 pin 13 接腳下方的(L)燈，出現橘色燈光閃爍的狀況，恭喜你，你已經正確完成 Arduino 板連結及程式上傳執行的基本設定，接著就可以開始進行 Arduino 程式設計的學習了。

二、循序結構及內建函數	1節	**2-1 Arduino 程式的基本架構** 點選功能表 **File > Examples > 1.Basics > BareMinimum**，載入 **BareMinimum** 範例程式。 仔細觀察程式中的內容會發現，**Arduino** 程式主要由 **setup()**及 **loop()**兩個函數 **(function)**區塊所組成，茲分述如下： **Setup()**函數 語法： **void setup()** **{** **}** 每當 Arduino 的程式(Sketch)開始執行時，Setup()便會被呼叫執行一次，主要用來進行 pin 接腳類型的設定、變數的初始化等，這個函數區塊中的程式只有在程式開始執行時以及程式重新執行(reset)時會被呼叫執行一次。 **Loop()**函數 語法： **void loop()** **{** **}** 在執行完 setup()函數中的所有設定及程式後，系統便會將控制權交給 loop()函數，且自此之後便會重覆不斷的執行撰寫在 loop()函數區塊中的程式，這也是 Arduino 程式的主要執行區塊，在此要特別強調，除非 Arduino 板的電力耗盡，否則，撰寫在 loop()函數區塊中的程式碼將會反覆不斷的被執行，永不停止。 **2-2 程式的基本控制結構** 當程式開始執行時，電腦便會依據程式中的指令敘述以及安排的順序完成指定的動作或工作，指令敘述執行的順序則與程式的基本結構有關，通常，程式語言有三種基本結構，茲分述如下： (1) 循序結構(Sequence) 程式由上而下，第一行指令敘述執行完後接著執行第二行，如此依序一行一行的執行。



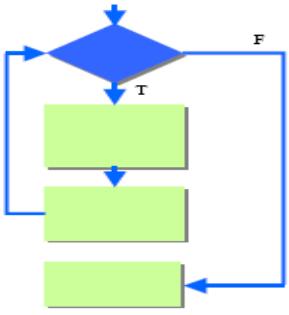
(2) 選擇結構(Selection)

若程式只有循序結構的話，程式只能依據既定的順序依序完成，如此，程式每次都只能作固定的動作處理，大幅限制了程式的執行能力與變化性，為突破此限制，程式語言在程式中加上了條件判斷的機制，讓程式能依據條件判斷的結果分支執行，程式流程進入判斷的菱形符號後，會判斷測試條件是否成立。然後，依據判斷的結果選擇程式的流向。



(3) 重複結構(Iteration)

在程式執行時，常有重複執行某部分程式片段的需要，搭配條件判斷的機制，部分程式片段可重複執行多次，直到某測試條件發生為止，程式重複執行部分即構成迴圈。



2-3 著手撰寫第一個程式

程式任務(教師示範後，請學生依例實作)：

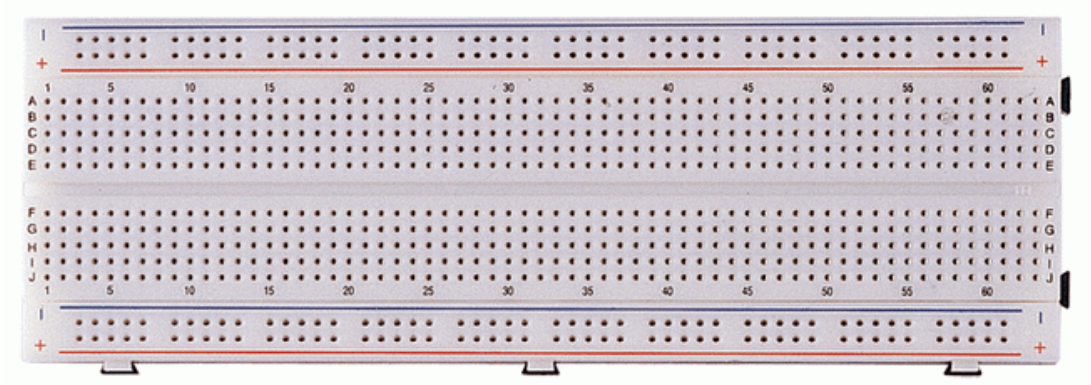
讓 LED 燈閃爍，亦即讓 LED 亮一秒，滅一秒交替進行。

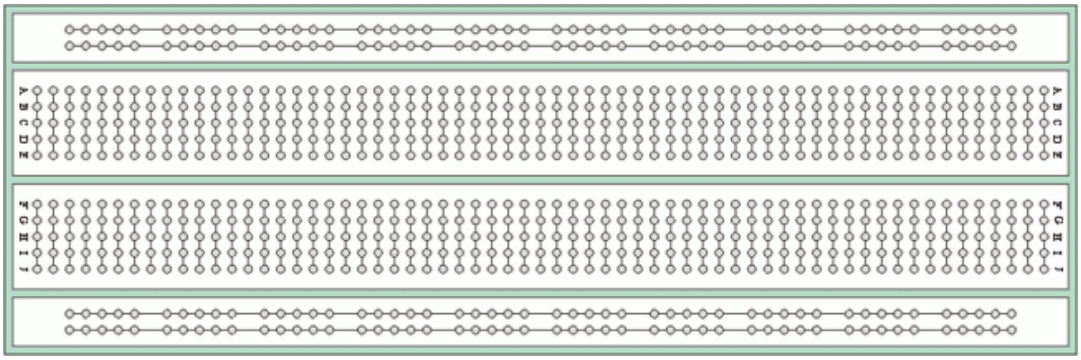
所需硬體

Arduino 板	X 1
麵包板	X 1
LED 燈	X 1
跳接線	X 2

硬體介紹

麵包板





麵包板外觀及內部構造圖

(取自：http://phys.thu.edu.tw/~mengwen/exp-electronics/exp-electronics-pro/exp-electronics-pro-01.htm)

麵包板是搭配及測試電路的重要工具，由上圖可知麵包板內部的電路連通狀態，上下兩排橫條區塊中，電路橫向連通，但兩橫列電路彼此不相連通，中間區塊電路縱向連通，但各縱行電路彼此不相連通。

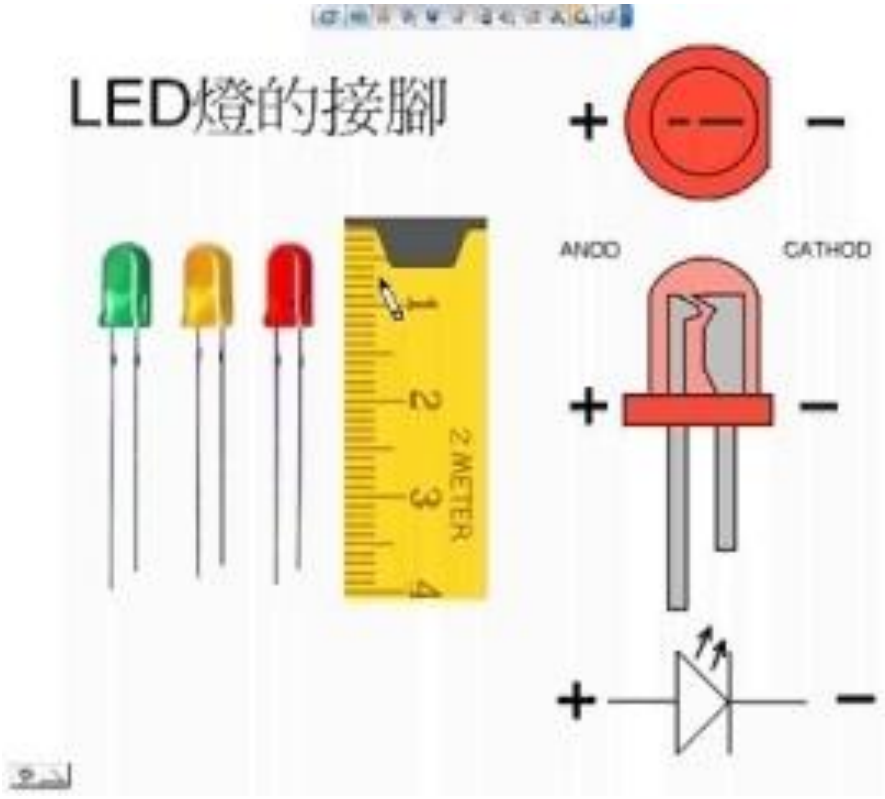
LED (Light Emitting Diode,發光二極體)

LED 在 1950 年代末於實驗室發展出來，1968 年 HP 開始商業化量產，早期

只有單調的暗紅色電子產品指示燈，1992 年 Nichia 突破藍光 LED 技術障礙後，逐漸衍生出多重色彩，亮度也大幅提高，並以顯示器(Display)、表面黏著型(SMD 等各種封裝型態深入生活中各個層面。

LED 是利用電能直接轉化為光能的原理，在半導體內正負極 2 個端子施加電壓，當電流通過，使電子與電洞相結合時，剩餘能量便以光的形式釋放，依其使用的材料的不同，其能階高低使光子能量產生不同波長的光，人眼所能接受到各種顏色的光，其波長介於 400-780nm，在此區間之外則為不可見光，包括紅外光及紫外光(UV)。

LED 只能往同一個方向導通，因此了解 LED 的接腳識別相當重要，LED 的長腳為正，短腳為負，另一種識別方法則由 LED 底面看，削平的一端代表負，保持圓形的一端代表正。



LED 發光二極體 (取自 www.vknow.com.tw/tag/867/LED)

建構電路

1. 將 LED 兩隻接腳分別插在麵包板中間區塊的兩個縱行上。
2. 將 LED 長腳(正)經由麵包板及跳接線的連結，接到 Arduino 板的 Digital 區 pin 9。
3. 將 LED 短腳(負)經由麵包板及跳接線的連結，接到 Arduino 板的 Digital 區 GND 接孔。

撰寫程式及測試

1. 以前一小節的 **BareMinimum** 範例為基本架構開始撰寫
   Examples\basics\BareMinimum
2. 在 setup()區塊中加入

```
void setup()
{
  pinMode( 9 , OUTPUT );
  digitalWrite( 9 , HIGH );
}
```

函數說明：

◆ **pinMode(pin, mode)** 接腳模式設定

其中

pin: 指定要進行模式設定的接腳

mode: 可設定為 INPUT or OUTPUT

因此在這個例子中，`pinMode( 9 , OUTPUT );` 便是將第 9 接腳設定為 OUTPUT 模式，亦即可藉由 pin 9 輸出訊號

◆ **digitalWrite(pin, value)** 輸出數位訊號

其中

pin: 指定要輸出數位訊號的接腳編號

value: 設定要輸出的訊號為 LOW 或 HIGH

因此在這個例子中，`digitalWrite( 9 , HIGH );` 便是由第 9 接腳輸出 HIGH 訊號。

3. 按下 Upload 測試一下，上傳完成後將會發現 LED 燈一直亮著
4. 接著直接在 setup()的第三行加入讓 LED 燈熄滅的程式
   `digitalWrite( 9 , LOW );`
5. 按下 Upload 測試一下，上傳完成後將會發現 LED 燈只閃一下子就滅掉了，這是因為 Arduino 的運作非常快速，在送完一個 HIGH 訊號後，立即再送一個 LOW 訊號，LED 燈也就隨即滅掉，為了要讓 LED 燈亮滅的時間都能維持久一點，可分別在輸出 HIGH 和 LOW 訊號的程式後各加入一個延遲的函數指令 delay()

◆ **delay(ms)** **讓程式暫停一段時間**

其中

ms: 指定要程式暫停多少毫秒

因此可加入 delay(1000); 來讓亮滅各維持一秒鐘。

6. 如此即完成讓 LED 亮一秒，滅一秒的程式，程式碼如下：

```
void setup()
{
  pinMode( 9 , OUTPUT );
  digitalWrite( 9 , HIGH );
  delay(1000);
  digitalWrite( 9 , LOW );
  delay(1000);
}
```

```
void loop()
{
}
```

7. 若要讓 LED 燈一直不斷的重複閃爍的狀況，可將控制明滅的程式區段剪下，貼到 loop()函數區塊中即可，程式碼將改為：

```
void setup()
{
  pinMode( 9 , OUTPUT );
}
```

```
void loop()
{
  digitalWrite( 9 , HIGH );
  delay(1000);
  digitalWrite( 9 , LOW );
  delay(1000);
}
```

在這一小節的例子中可以發現，Arduino 在程式執行時，是依程式碼撰寫的內容，依序由上到下執行，此即為程式結構中的循序結構。而在程式碼中所用到的 pinMode() , digitalWrite(), delay()等函數，都是 Arduino 提供的內建函數，也就是 Arduino 團隊已經將常用的功能寫成內建函數的形式，使用者並不需要知道函

		數中的程式內容，只要依需求選擇適當的函數，並提供該內建函數所需的參數(如：pin、mode 等)，即可完成所需的運作或取得所需的運算結果。 牛刀小試(請學生仿照程式任務的做法嘗試實作) 1. 請嘗試改變程式的順序或提供內建函數不同的參數內容，觀察 LED 燈會有什麼不同的表現。 2. 請加入另一個 LED 燈並發揮創意，嘗試利用兩個 LED 燈做出特殊的創意表現
三、選擇結構	1節	**3-1 什麼是選擇結構** 透過循序結構，我們的程式已能依據程式的設定，循序的執行所要完成的各項動作，但在日常生活以及程式的處理當中，常會遇到需要經過判斷才能決定下一步行動的狀況，例如： 「如果」室內溫度 > 攝氏 28 度 吹冷氣 「否則」 吹電風扇 以上是決定要吹冷氣或吹電風扇的情境，在這個情境當中是以室內溫度的高低來作為判斷的條件，當室內溫度大於攝氏 28 度時，就吹冷氣，否則，就吹電風扇。 在程式的處理過程中，也常有需要透過判斷來決定不同的程式流向的狀況，例如： 「如果」使用者按下按鈕 LED 燈亮 「否則」 LED 燈滅 **3-2 Arduino 的選擇結構** Arduino 的選擇結構語法： `if ( 條件判斷式 )` `{` `  // action A` `}` `else` `{` `  // action B` `}`

說明：

當條件判斷式為真(true)時，程式將執行 action A 區段，反之，則執行 action B 區段，達到讓程式可分支執行的功能。

3-3 條件判斷式

條件判斷式在選擇結構中扮演著非常重要的角色，當條件成立時結果為真(true)，條件不成立時結果為假(false)，為真或為假的結果是靠關係運算式及邏輯運算式來完成，最後，選擇結構便依據條件運算的結果，選擇後續要執行的程式區段。

3-3-1 關係運算式

關係運算式是由運算元與關係運算子所組成，以關係運算子的種類來判斷運算元之間的關係，形成真(True)或假(False)的結果，以供程式判斷之用。

關係運算式的語法結構如下：

A 運算元　關係運算子　B 運算元

例如：　小明的身高　＞　172 公分

在這個例子的關係運算式中，（小明的身高）與（172 公分）分別是兩個待比較的運算元，（＞）則代表要進行的關係運算種類，運算式運算後會產生兩種結果，如果小明實際的身高為 180 公分，則小明的身高＞172 公分的運算結果就會為真(True)，如果小明實際的身高為 160 公分，則小明的身高＞172 公分的運算結果就會為假(False)。

關係運算子包含有下列類型，分別說明如下：

關係運算子	說明
= =	左邊的值***等於***右邊的值
! =	左邊的值***不等於***右邊的值
>	左邊的值***大於***右邊的值
> =	左邊的值***大於或等於***右邊的值
<	左邊的值***小於***右邊的值
<=	左邊的值***小於或等於***右邊的值

3-3-2 邏輯運算式

在日常生活進行判斷時，常會有需要將多個條件放在一起考量的情境，在撰寫程式時，當有兩個以上的條件需要同時考量時，便可用邏輯運算式來組合這些條件。

邏輯運算式的語法結構如下：

A 條件式　邏輯運算子　B 條件式

例如：　小明的身高＞172 公分　AND　小明的體重＜50 公斤

在這個例子的邏輯運算式中，（小明的身高＞172 公分）與（小明的體重＜50 公斤）分別是兩個待連結考量的條件式，（AND）則代表要進行的邏輯運算種類，運算式運算後會產生兩種結果，此例中，如果小明實際的身高為 180 公分，體重為 40 公斤，則（小明的身高＞172 公分）的條件式為真(True)，而（小明的體重＜50 公斤）的條件式亦為真(True)，邏輯運算式（真）AND（真）的結果即為（真），其餘的狀況皆為假(False)。

邏輯運算子包含有下列類型，分別說明如下：

		邏輯運算子		
條件式 A	條件式 B	A And B	A Or B	Not A
True	True	True	True	False
True	False	False	True	
False	True	False	True	True
False	False	False	False	

3-4 以按鈕控制 LED 明滅

程式任務(教師示範後，請學生依例實作)：

當使用者按下按鈕時可以讓 LED 燈亮，放開按鈕後讓 LED 燈滅。

所需硬體

Arduino 板	X 1
麵包板	X 1
LED 燈	X 1
按鈕	X 1
跳接線	X 5
電阻 10K 歐姆	X 1

建構電路

1. 建構 LED 電路(同前例 2-3)
2. 將按鈕放置在麵包板中間部分，將下端的兩隻接腳分別插在麵包板中間下方區塊的兩個縱行上
3. 以跳接線連結按鈕下端**左邊接腳**與 Arduino 板的 **POWER 區的 5V** 接孔
4. 以跳接線接上 10K 歐姆電阻後，連結按鈕下端**右邊接腳**與 Arduino 板的 **POWER 區的 GND** 接孔
5. 以跳接線連結按鈕下端**右邊接腳**與 Arduino 板的 **DIGITAL 區的 pin 2** 接孔

撰寫程式及測試

1. 在 setup()中設定 pin 9 為輸出模式、pin 2 為輸入模式

```
pinMode( 9 , OUTPUT);
pinMode( 2 , INPUT);
```

2. 在 loop()中撰寫選擇結構程式碼

```
if ( digitalRead(2) == HIGH )
{
  digitalWrite( 9, HIGH );
}
else
{
  digitalWrite( 9, LOW );
}
```

說明：

◆ **digitalRead(pin)** 由指定的接腳讀取數位訊號

其中

pin: 為指定要讀取數位訊號的接腳編號

此函數將會回傳數位訊號 HIGH 或 LOW

◆ `digitalRead(2) == HIGH`

此為 if 結構中用以決定程式分支的條件判斷式，條件式中間的==用以判斷式子左右兩邊的值是否相等，也就是判斷自 pin 2 所讀入的數位訊號是否為 HIGH，若是的話就由 pin 9 送出一個 HIGH 訊號讓 LED 亮起來，反之，則由 pin 9 送出一個 LOW 訊號讓 LED 滅掉。

3. 上傳程式並進行測試，完成的程式碼如下：

```
void setup()
{
    pinMode( 9 , OUTPUT);
    pinMode( 2 , INPUT);
}
```

```
void loop()
{
    if ( digitalRead(2) == HIGH )
    {
       digitalWrite( 9, HIGH );
    }
    else
```

```
        {
            digitalWrite( 9, LOW );
        }
    }
```

四、變數及運算式

1節

4-1 什麼是變數

所謂變數即是一塊預先保留的記憶體空間，可在程式執行過程中，將資料或運算的結果儲存起來，供程式後續處理之用，變數，顧名思義，其所含的數值在程式執行過程中，是可以透過儲存變數資料覆蓋變數內原有資料來進行改變的，而為了程式處理需要，變數使用前必須先進行宣告，設定該變數所需的變數名稱、變數型態與初始值等。

4-1-1 變數的宣告

變數在使用前必須先進行宣告，包括設定變數的名稱、設定變數的資料型態，以及設定變數的初始值等，好讓電腦在程式執行時，能在記憶體中保留適當的記憶體空間，並將這塊記憶體空間指定給所設定的變數名稱，如此，在後續的程式中便可以此變數名稱來存取所保留的記憶體空間，作為儲存資料之用。

變數宣告語法：

資料型態 變數名稱 ;

如：int score ;

資料型態 變數名稱 = 初始值 ;

如：int score = 70 ;

4-1-2 變數的使用

變數宣告後即可開始使用，可將資料指定給變數儲存起來供程式後續使用，亦可將儲存在變數中的資料讀取出來應用，茲說明如下：

- 將資料指定給變數儲存起來

  score = 90 ;

  如此便可把 90 儲存在變數 score 中

- 將儲存在變數中的資料讀取出來應用

  grade = score / 10 ;

  此例中會先將變數 score 的值(90)讀出，並將 90 /10 運算後的值(9)指定到另一個變數 grade 中儲存起來。

4-2 以按鈕切換 LED 的狀態

程式任務(教師示範後，請學生依例實作)：

當使用者按下按鈕時可以切換 LED 的狀態，亦即若 LED 目前是亮的狀態，則按下按鈕時，LED 會轉為滅的狀態，反之，若 LED 目前是滅的狀態，則按下按鈕時，LED 會轉為亮的狀態。

所需硬體

同前例 3-4

建構電路

同前例 3-4

撰寫程式及測試

1. 由程式任務的說明中，可知按下按鈕後 LED 燈的明滅，是由目前 LED 燈的狀態來決定的，因此，必須先宣告一個變數 state，將目前 LED 燈的狀態保留下來。

   int state = 0;

   說明：

   變數名稱為 state，變數型態為 int(整數)，同時將內容指定為 0

2. 在 setup()中設定 pin 9 為輸出模式、pin 2 為輸入模式

   pinMode(9 , OUTPUT);

   pinMode(2 , INPUT);

3. 在 loop()中撰寫選擇結構程式碼

```
if ( digitalRead(2) == HIGH )
{
   state = 1 - state;
}

if ( state == 1 )
{
   digitalWrite( 9, HIGH );
}
else
{
   digitalWrite( 9, LOW );
}
```

說明：

1. 在第一個 if 結構中，主要用來判斷**按鈕是否有被按下**，若是的話則進行 state = 1 – state ; 的運算，一開始 state 內容為 0，當按鈕按下時，會進行 1 – 0 的運算得到 1，並將之指定回給 state 儲存起來，此時 state 內容為 1，當按鈕按下時，會進行 1 – 1 的運算得到 0，並將之指定回給 state 儲存起來，則 state 的內容變為 0，因此，經由運算式 state = 1 – state ; 即可達到改變狀態的設定。
2. 在第二個 if 結構中，則依據變數 state 中的值來決定 LED 燈的明或滅，當 state 為 1 時，則讓 LED 燈亮起來，當 state 為 0 時，就讓 LED 燈滅掉

4. 上傳程式並進行測試，完成的程式碼如下：

```
int state = 0;
```

```
void setup()
{
    pinMode( 9 , OUTPUT);
    pinMode( 2 , INPUT);
}
```

```
void loop()
{
    if ( digitalRead(2) == HIGH )
    {
       state = 1 – state ;
    }

    if ( state == 1 )
    {
       digitalWrite( 9, HIGH ) ;
    }
    else
    {
       digitalWrite( 9, LOW ) ;
    }
}
```

5. 特別說明：
   此例中按下按鈕時，程式會有不穩定的狀態，也就是按下按鈕時，不一定會進行正確的切換，其原因是 Arduino 每秒可進行高達 1600 萬次的運算，

所以當按下按鈕的瞬間，Arduino 已經進行的多次的偵測與轉換，因此會產生好像切換不正常的狀態，這樣的狀況可以改寫程式來改善，請嘗試修改看看。

4-3 以 serial monitor 觀察變數的內容變化

在前一節的例子中，我們發現會產生切換不正常的狀態，而由程式碼的內容可知，控制燈的明滅主要是由變數 state 來決定，因此，只要我們能觀察到變數 state 的內容變化情形，即可找出問題的所在，這也是一般程式偵錯時常用的方法。

所需硬體

同 4-2

建構電路

同 4-2

撰寫程式及測試

1. 在 setup()函數中加入

   Serial.begin(9600);

   說明：開啟序列埠(serial port)，並設定傳輸速率為 9600 bps。

2. 在 loop()函數中的第一個 if 結構中加入

   Serial.println(state);

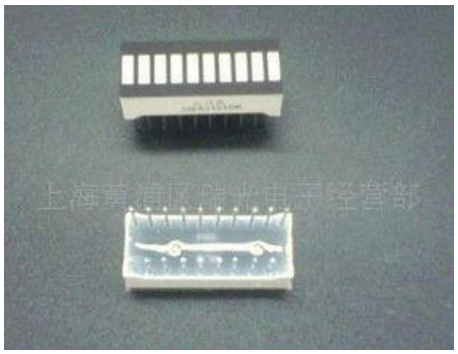
   說明：將變數 state 的內容送到序列埠(serial port)印出來

3. 上傳程式，完成的程式碼如下：

```
int state = 0;
```

```
void setup()
{
    pinMode( 9 , OUTPUT);
    pinMode( 2 , INPUT);
    Serial.begin(9600);
}
```

```
void loop()
{
    if ( digitalRead(2) == HIGH )
    {
       state = 1 – state ;
       Serial.println( state );
```

		`` } if (state == 1) { digitalWrite(9, HIGH) ; } else { digitalWrite(9, LOW) ; } } ``` 4. 在 Tools 功能表中點選 serial monitor，或按 Ctrl+Shift+M 開啟 serial monitor，並進行測試與觀察
五、計數式重複結構 FOR	2 節	**5-1 什麼是重複結構** 所謂重複結構，顧名思義就是可以讓程式反覆執行的結構，有了重複結構的協助，將可使程式更為精簡且讓撰寫好的程式片段可以被重複執行應用，當然我們不能讓程式不斷的反覆執行下去，必須要搭配有讓重複結構停止的條件，而依據這些條件的不同，重複結構大致可分為兩種類型，一為計數式重複結構，另一為條件式重複結構，以下兩個章節將以實例說明這兩種重複結構的用途與編寫方式。 **5-2 以複製程式區段的方式完成重複執行的效果** 程式任務(教師示範後，請學生依例實作)： 讓十段 LED 的第一個燈重複閃爍 5 次。 硬體介紹  十段 LED 燈(取自 http://detail.china.alibaba.com/buyer/offerdetail/690990019.html)

一個十段 LED 燈乃是由 10 個 LED 組合而成，背面上下各有十支接腳，上下各一接腳為所對應的 LED 燈的正極及負極，判斷正極的方式為仔細觀察十段 LED 燈的四個角，其中有一個角為斜角，該角所在的長邊即為正極端。

所需硬體

Arduino 板 X 1
麵包板 X 1
十段 LED 燈 X 1
跳接線 X 2

建構電路

1. 將十段 LED 放置在麵包板中間區塊的上下兩個分區間，並將正負極接腳分別跨接在上下兩個分區上。
2. 將十段 LED 的第一個燈的正極經由麵包板及跳接線的連結，接到 Arduino 板的 Digital 區 pin 9。
3. 將十段 LED 的第一個燈的負極經由麵包板及跳接線的連結，接到 Arduino 板的 Digital 區 GND 接孔。

撰寫程式及測試

1. 參照第二章的例子，完成讓十段 LED 的第一個燈閃爍一次的程式。由於只要閃爍一次，因此請將程式撰寫在 setup()函數中。程式如下：

```
void setup()
{
  pinMode( 9 , OUTPUT );
  digitalWrite( 9 , HIGH );
  delay(1000);
  digitalWrite( 9 , LOW );
  delay(1000);
}
```

```
void loop()
{
}
```

2. 撰寫讓十段 LED 的第一個燈閃爍 5 次的程式
   若要讓十段 LED 的第一個燈重複閃爍數次的動作，你想到該怎麼進行嗎？對了，大家很直覺的都會想到用複製程式碼的方式，請嘗試將前述的程式

區段複製數次即可達成。

```
void setup()
{
  pinMode( 9 , OUTPUT );
  //第一次閃爍
  digitalWrite( 9 , HIGH );
  delay(1000);
  digitalWrite( 9 , LOW );
  delay(1000);
  //第二次閃爍
  digitalWrite( 9 , HIGH );
  delay(1000);
  digitalWrite( 9 , LOW );
  delay(1000);
  //第三次閃爍
  digitalWrite( 9 , HIGH );
  delay(1000);
  digitalWrite( 9 , LOW );
  delay(1000);
  //第四次閃爍
  digitalWrite( 9 , HIGH );
  delay(1000);
  digitalWrite( 9 , LOW );
  delay(1000);
  //第五次閃爍
  digitalWrite( 9 , HIGH );
  delay(1000);
  digitalWrite( 9 , LOW );
  delay(1000);
}
```

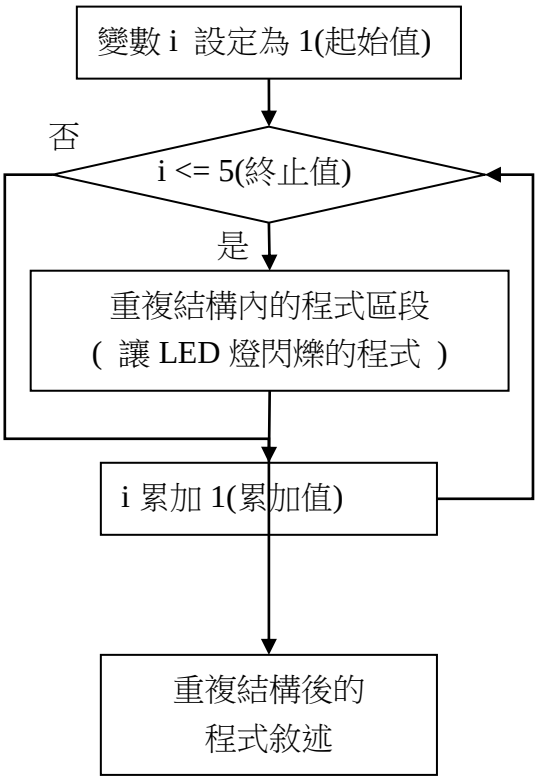
```
void loop()
{
}
```

完成上述程式後，你是否發現程式變得非常冗長，且萬一重複執行的程式區段中有一個動作需要修改，這時就必須將剛剛複製貼上的所有程式區段都一一修改，因此，以複製程式碼來達成程式反覆執行的方式並不可行，較簡潔可行的方法是改以計數式重複結構來設計。

5-3 計數式重複結構 FOR 的基本應用

For 重複結構的主要功能為重複執行一定次數撰寫在{ }中的程式碼，其中包含一個累加變數，用以累加計數次數及終止迴圈，For 迴圈的語法及運作如下：

```
for（初始設定; 繼續執行的條件; 累加運算）
{
    欲重複執行的程式碼
}
```



有了計數式重複結構後，前一節的程式可改寫如下：

```
void setup()
{
   int i ;
   pinMode( 9 , OUTPUT );

   for ( i = 1 ; i <=5 ; i = i + 1)
   {
      digitalWrite( 9 , HIGH );
      delay(1000);
```

```
      digitalWrite( 9 , LOW );
      delay(1000);
    }
  }
```

```
void loop()
{
}
```

為觀察 for 迴圈中變數 i 的改變情形，可以如第四單元介紹的觀察變數的方法，利用 serial monitor 來觀察，改寫後的程式如下：

```
void setup()
{
   int i ;
   pinMode( 9 , OUTPUT );
   Serial.begin(9600);

   for ( i = 1 ; i <=5 ; i = i + 1)
   {
     Serial.println( i );
      digitalWrite( 9 , HIGH );
      delay(1000);
      digitalWrite( 9 , LOW );
      delay(1000);
   }
}
```

```
void loop()
{
}
```

5-4 計數式重複結構 FOR 的進階應用

在上一節的程式中，變數 i 主要用以累加計數次數及判斷是否繼續重複執行之用，但實際上變數 i 還可以運用在 For 重複結構中，創造出更靈活的應用。

程式任務(教師示範後，請學生依例實作)：

讓十段 LED 的所有燈依序閃爍 1 次，即第一個 LED 亮滅、第二個 LED 亮滅…。

所需硬體

Arduino 板　X 1
麵包板　X 1
十段 LED 燈　X 1
跳接線　X 21

建構電路

1. 將十段 LED 放置在麵包板中間區塊的上下兩個分區間，並將正負極接腳分別跨接在上下兩個分區上。
2. 依序將十段 LED 的每一個燈的正極經由麵包板及跳接線的連結，分別接到 Arduino 板的 Digital 區 pin 2、pin 3、pin 4、…、pin 11。
3. 依序將十段 LED 的每一個燈的負極經由跳接線的連結，分別接到麵包板下排相對的藍色橫排上(-)。
4. 利用一條跳接線自麵包板下排的藍色橫排(-)上的任一接孔，連結到 Arduino 板的 Digital 區 GND 接孔。

撰寫程式及測試

1. 此程式以前一節程式為基礎加以修改。
2. 由於要將 pin 2、pin 3、pin 4、…、pin 11 設定為 OUTPUT pin，因此改用 for 迴圈來設定。

```
for ( i = 2 ; i <= 11 ; i = i + 1)
{
      pinMode( i , OUTPUT );
}
```

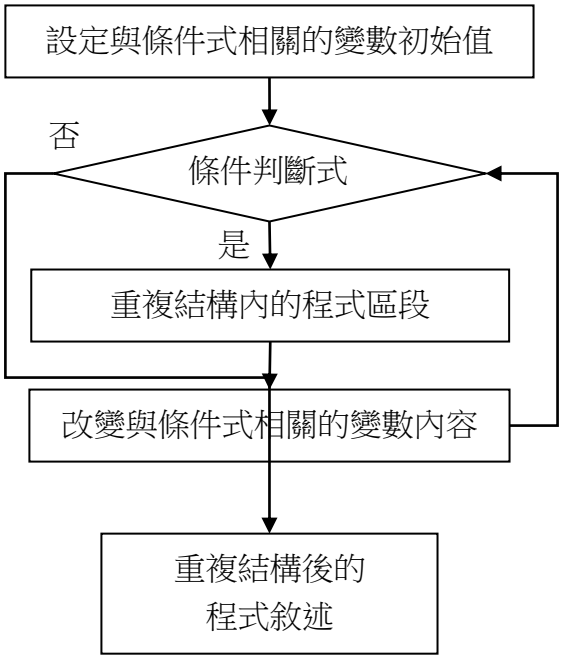
說明：此段程式碼等同於

```
pinMode( 2, OUTPUT );
pinMode( 3 , OUTPUT );
…
pinMode( 11, OUTPUT );
```

3. 將控制 LED 明滅程式中的 pin 編號 9 改為變數 i
4. 上傳程式碼並加以觀察，完成之程式碼如下：

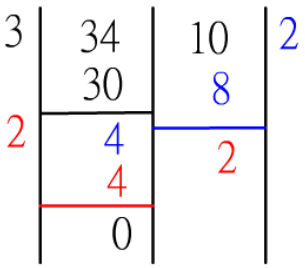
```
void setup()
{
  int i ;
  for ( i = 2 ; i <= 11 ; i = i + 1)
  {
    pinMode( i , OUTPUT );
```

		} Serial.begin(9600); for (i = 2 ; i <= 11 ; i = i + 1) { Serial.println(i); digitalWrite(i , HIGH); delay(1000); digitalWrite(i , LOW); delay(1000); } } void loop() { } 牛刀小試(請學生仿照程式任務的做法嘗試實作)： 請改寫前述程式碼，讓 LED 燈除由 1、2、…、10 依序閃爍外，接著再由 10、9、…、1 依序閃爍回來。
六、條件式重複結構 While	2 節	所謂條件式重複結構，就是依據條件判斷的結果是否為真(True)，來決定重複結構中的程式區塊是否繼續反覆執行。 當程式執行前可以預先得知重複結構所要執行的次數時，可以使用前一章所介紹的計數式迴圈，但若是無法預先得知欲執行的次數，而必須由執行過程中的某些條件來決定時，則可採用條件式重複結構 While。 **6-1 條件式重複結構的語法** while(條件判斷式) { 欲重複執行的程式碼 } 條件式重複迴圈會連續不斷的重複執行大括號{}中所包含的程式碼，直到 while 後方( )中的條件判斷式為假 false，也就是說，當條件為真 True 時，重複迴圈會不斷重複執行，當條件為假 False 時，則跳出迴圈繼續執行後面的程式。



6-2　條件式重複結構 While 的應用

前面章節都是介紹利用 Arduino 控制周邊電子元件，其實 Arduino 也可以進行數學運算，以下便用(以輾轉相除法求兩數的最大公因數)為例，來介紹 While 重複結構的用法，其結果則透過 serial monitor 來顯示。



輾轉相除法 (取自：昌爸工作坊)

由以上輾轉相除法的圖示可以歸納出以下運算步驟：

1.

a		b		q		r
34	/	10	=	3	⋯	4

2.

a		b		q		r
10	/	4	=	2	⋯	2

3.

a		b		q		r
4	/	2	=	2	⋯	0

仔細觀察可以發現，第二輪的被除數為第一輪的除數，第二輪的除數為第一輪的餘數，有了第二輪的被除數與除數，即可運算出第二輪的餘數，並交給第三輪進行運算，以此類推，直至餘數為 0 時結束運算。

程式任務(教師示範後，請學生依例實作)：

以輾轉相除法求兩數的最大公因數，其結果透過 serial monitor 顯示。

所需硬體

Arduino 板　X 1

建構電路

無

撰寫程式及測試

1. 宣告變數 a、b、r，以儲存各輪運算過程中的被除數、除數及餘數

   ```
   int   a, b, r=1 ;
   ```

2. 指定第一輪的被除數為 34，除數為 10

   ```
   a = 34 ;
   b = 10 ;
   ```

3. 計算第一輪的餘數

   ```
   r = a % b ;
   ```

4. 設定第二輪的被除數 a 為第一輪的除數(b)、第二輪的除數 b 為第一輪的餘數(r)

   ```
   a = b ;
   b = r ;
   ```

5. 計算第二輪的餘數

   ```
   r = a % b ;
   ```

6. 設定第三輪的被除數 a 為第二輪的除數(b)、第三輪的除數 b 為第二輪的餘數(r)

```
a = b ;
b = r ;
```

7. 由此我們可以發現步驟 5 及 6 已經重複了步驟 3 及 4 的動作，因此我們可以改用條件式重複結構 while(因我們並不能預先知道要執行幾輪的運算，運算的次數因被除數及除數的不同而異)，而 while 結構中的條件為餘數 r 不等於 0。

```
while( r != 0 )
{
    r = a % b ;
    a = b ;
    b = r ;
}
```

8. 最後將 a 透過 serial monitor 顯示

```
Serial.begin( 9600 ) ;
Serial.println( a ) ;
```

9. 將程式上傳，並透過 serial monitor 觀察執行結果，完成的程式碼如下：

```
void setup()
{
  int   a, b, r=1 ;
  a = 34 ;
  b = 10 ;

  while( r != 0 )
  {
    r = a % b ;
    a = b ;
    b = r ;
  }

  Serial.begin( 9600 ) ;
  Serial.println( a ) ;
}
```

```
void loop()
{
}
```

10. 請嘗試修改被除數及除數的數值，重新執行後觀察結果是否正確。