模組化進階 Arduino 教學實驗平台

實驗手冊

華亨科技股份有限公司

目錄

實驗 2-1: 8 顆 LED 亮滅控制

目的：利用 Arduino MEGA2560 開發板上的數位輸出入(DIO)腳，來進行 Arduino 控制 LED 亮滅的電路與程式設計方法。

功能：利用 Arduino MEGA2560 開發板上的 8 隻數位輸出入腳，分別為第 2 到第 9 隻數位接腳，接上 74LS244(74AC244)的 8 個緩衝閘，完成 8 顆 LED 亮滅控制，而一開始要用 Arduino MEGA2560 開發板上的 A15 數位輸出入腳輸出 LOW(低電位)來制能 74LS244，才能驅動 LED 亮滅。本實驗有四個程式來控制 LED 亮滅，分別為控制一顆和 8 顆 LED 亮滅各一秒的兩個基礎程式，還有讓使 8 顆 LED 向左和向右逐一點亮的兩個陣列查表程式，使用迴圈一直不斷的重覆循環就會看到 LED 固定亮滅閃爍和向左、向右移動的效果。本電路利用外接電源加上限流電阻，來驅動 LED，當 Arduino MEGA2560 開發板接腳設定為 LOW(低電位)時產生順偏，便得 LED 點亮。

原理：LED 二極體點亮需 10~20mA，因 74LS244(74AC244)的 8 個緩衝閘的汲取電流比輸出電流大，所以設計用 74LS244(74AC244)的 8 個緩衝閘來做低電位驅動可使 LED 較亮，且 Arduino MEGA2560 開發板上的 8 隻數位輸出入腳也只要輸出 LOW 便得 LED 點亮 LED，輸出 HIGH 來熄滅 LED。LED 導通電壓大約 2V~2.5V，設計取 2V，導通電流 10mA，電阻設計為 R1=(5-2)V/10mA=300Ω。

電路：Arduino MEGA2560 開發板上的 A15 數位輸出入腳接上 74LS244 的 $1\overline{G}$ 和 $2\overline{G}$，當輸出 LOW(低電位)來制能 74LS244，而 Arduino MEGA2560 開發板接腳第 2 到第 9 隻數位接腳，接上 74LS244(74AC244)的 8 個緩衝閘的 8 個輸入(1A1~4 和 2A1~4)，74LS244(74AC244)的 8 個緩衝閘輸出(1Y1~4 和 2Y1~4)接上 8 顆 LED，8 顆 LED 再接上 8 顆 300Ω的限流電阻後接到 5V 電源上如圖 2-1-1。

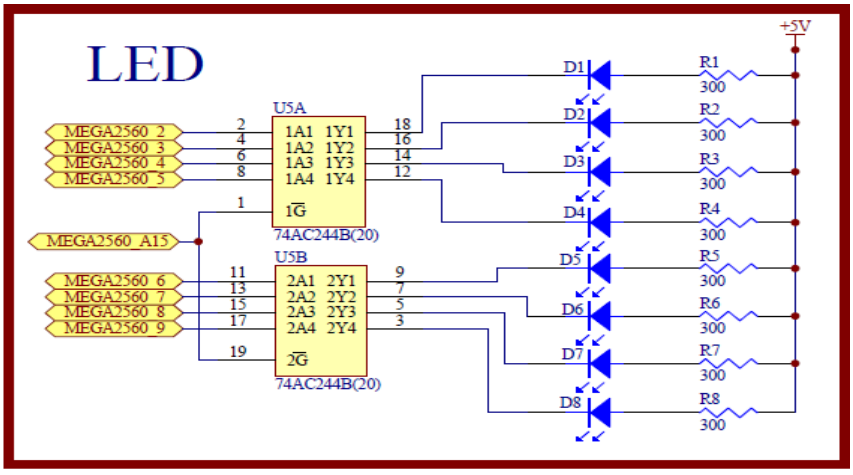


圖 2-1-1　LED 亮滅控制電路

元件：

編號	元件項目	數量	元件名稱
1	74AC244	1	8 個緩衝閘晶片
2	D1~D8	8	單色 LED
3	R1~R8	8	300 Ω電阻

程式：本實驗共有四個程式來控制 LED 亮滅，分別為控制一顆 LED 亮滅各一秒的 2-1-1 Flash 程式和 8 顆 LED 亮滅各一秒的 2-1-2 All_Flash 程式，還有讓使 8 顆 LED 向左逐一點亮的 2-1-3Shift_Left 程式和向右逐一點亮的 2-1-4 Shift_Right 程式，使用迴圈一直不斷的重覆循環就會看到 LED 固定亮滅閃爍和向左、向右移動的效果。

2-1-1 Flash 程式:

Flash

行號	程式敘述	註解
1	int led = 2;	// 定義 LED 接腳
2	void setup(){	// 只會執行一次的程式初始設定函式
3	pinMode(led,OUTPUT);	// 規劃 LED 腳為輸出模式
4	pinMode(A15,OUTPUT);	// 規劃 A15 腳為輸出模式
5	digitalWrite(A15, LOW);	// A15 輸出 LOW，制能 74AC244
6	}	// 結束 setup()函式
7	void loop(){	// 永遠周而復始的主控制函式
8	digitalWrite(led, HIGH);	// led 輸出 HIGH，LED 熄滅
9	delay(1000);	// 呼叫延遲函式等 1000 毫秒=1 秒
10	digitalWrite(led, LOW);	// led 輸出 LOW，LED 點亮
11	delay(1000);	// 呼叫延遲函式等 1000 毫秒=1 秒
12	}	// 結束 loop()函式

2-1-2 All_Flash 程式:

All_Flash

行號	程式敘述	註解
1	int BASE = 2;	// 定義第一顆 LED 接的 I/O 腳
2	int NUM = 8;	// 定義 LED 的總數
3	int index = 0;	// 定義計數指標，由 0 開始
4	int D = 1000;	// 定義延遲的時間
5	void setup(){	// 只會執行一次的程式初始設定函式
6	for (int i = BASE; i < BASE + NUM; i ++) {	// 使用 for 迴圈規劃 8 顆 LED 腳
7	pinMode(i,OUTPUT); }	// 規劃 8 顆 LED 腳為輸出模式
8	pinMode(A15,OUTPUT);	// 規劃 A15 腳為輸出模式
9	digitalWrite(A15, LOW);	// A15 輸出 LOW，制能 74AC244
10	}	// 結束 setup()函式
11	void loop(){	// 永遠周而復始的主控制函式
12	for (int i = BASE; i < BASE + NUM; i ++) {	// 使用 for 迴圈設定輸出 8 顆 LED 電壓
13	digitalWrite(i, LOW); }	// led 輸出 LOW，8 顆 LED 點亮
14	delay(D);	// 呼叫延遲函式等 D=1000 毫秒=1 秒
15	for (int i = BASE; i < BASE + NUM; i ++) {	// 使用 for 迴圈設定輸出 8 顆 LED 電壓
16	digitalWrite(i, HIGH); }	// led 輸出 HIGH，8 顆 LED 熄滅

```
17      delay(1000);                    // 呼叫延遲函式等 D=1000 毫秒=1 秒
18  }                                   // 結束 loop()函式
```

2-1-3 Shift_Left 程式:

Shift_Left

```
行號  程式敘述                                  註解
1   const int led[]={2,3,4,5,6,7,8,9};      // 使用查表陣列，定義 8 顆 LED 左移接腳順序編號
2   int i;                                  // 定義陣列計數索引
3   int j=0;                                // 定義陣列計數索引，由 0 開始
4   void setup(){                           // 只會執行一次的程式初始設定函式
5   for (int i=0; i<8; i++) {               // 使用 for 迴圈規劃 8 顆 LED 腳
6      pinMode(led[i],OUTPUT); }            // 規劃 8 顆 LED 腳為輸出模式
7      pinMode(A15,OUTPUT);                 // 規劃 A15 腳為輸出模式
8      digitalWrite(A15, LOW);              // A15 輸出 LOW，制能 74AC244
9   }                                       // 結束 setup()函式
10  void loop(){                            // 永遠周而復始的主控制函式
11  for (int i=0; i<8; i++) {               // 使用 for 迴圈設定輸出 8 顆 LED 電壓
12     digitalWrite(i, HIGH);   }           // led 輸出 HIGH，8 顆 LED 熄滅
13     digitalWrite(led[j], LOW);           // led[j]輸出 LOW，點亮陣列中第一個 LED
14     delay(500);                          // 呼叫延遲函式等 D=500 毫秒=0.5 秒
15     j++;                                 // 將 j 加 1，逐一點亮陣列中的 LED
16     if(j==8)                             // 假如點亮到陣列最後一顆 LED
17     j=0;                                 // 再重頭開始點亮陣列中第一個 LED
18  }                                       // 結束 loop()函式
```

2-1-4 Shift_Right 程式:

Shift_Right

```
行號  程式敘述                                  註解
1   const int led[]={9,8,7,6,5,4,3,2};      // 使用查表陣列，定義 8 顆 LED 左移接腳順序編號
2   int i;                                  // 定義陣列計數索引
3   int j=0;                                // 定義陣列計數索引，由 0 開始
4   void setup(){                           // 只會執行一次的程式初始設定函式
5   for (int i=0; i<8; i++) {               // 使用 for 迴圈規劃 8 顆 LED 腳
6      pinMode(led[i],OUTPUT); }            // 規劃 8 顆 LED 腳為輸出模式
7      pinMode(A15,OUTPUT);                 // 規劃 A15 腳為輸出模式
8      digitalWrite(A15, LOW);              // A15 輸出 LOW，制能 74AC244
9   }                                       // 結束 setup()函式
10  void loop(){                            // 永遠周而復始的主控制函式
11  for (int i=0; i<8; i++) {               // 使用 for 迴圈設定輸出 8 顆 LED 電壓
12     digitalWrite(i, HIGH);   }           // led 輸出 HIGH，8 顆 LED 熄滅
13     digitalWrite(led[j], LOW);           // led[j]輸出 LOW，點亮陣列中第一個 LED
14     delay(500);                          // 呼叫延遲函式等 D=500 毫秒=0.5 秒
15     j++;                                 // 將 j 加 1，逐一點亮陣列中的 LED
16     if(j==8)                             // 假如點亮到陣列最後一顆 LED
17     j=0;                                 // 再重頭開始點亮陣列中第一個 LED
18  }                                       // 結束 loop()函式
```

練習：

一、 利用 8 個 LED 設計來回左右位移的霹靂燈。

實驗 2-2: 繼電器驅動控制

目的：利用 Arduino 數位輸出控制透過繼電器間接控制家電，了解如何使用微處理機，來達成控制大電流的電器產品。

功能：利用 Arduino MEGA2560 第 A4 隻數位接腳送出訊號來控制繼電器 ON、OFF 動作，繼電器用 5V 驅動，用單刀雙擲(SPDT)開關的繼電器，可提供 AC 電源的控制開關，當 Arduino 第 A4 隻數位接腳送出 HIGH 訊號繼電器 ON，送出 LOW 訊號繼電器 OFF，並使用 delay(1000)副程式來完成 1 秒的延遲，做為 ON->OFF->ON 的切換時間，並可聽聽繼電器切換的聲音。

原理：繼電器可做為控制家用交流電源的機械開關，一般主要缺點是切換速度慢，大約在 1 秒左右，無法做快速控制，另一缺點是有切換次數的限制，所以要使用的繼電器要查看它的使用次數，一般多會選用 10 萬次到 100 萬次以上，繼電器還要注意是它的電感端驅動電壓與電流，本實驗用+5V 驅動，繼電器就要選用+5V 驅動，繼電器驅動電流大約在 30mA~100mA，電流會大於 Arduino 發展板所提供的電流，所以加 NPN 電晶體來驅動，要使用二極體做為電感電流的放電路徑，防止電晶體由於電感放電而受損。繼電器有兩端，一端為共同接點(COM)，另一端有常關(NC)和常開(NO)兩個接點，NC 為繼電器不需動作平常即和 COM 導通，NO 是當繼電器動作後才會和 COM 導通，使用 SPDT 繼電器，可用來切換 110V 交流市電的電路設計，但實驗因 110V 有觸電危險，所以用三用電表切換到量短路的檔位，用三用電表短路會叫和開路不叫的功能，來得知技電器的開關動作。

電路：繼電器驅動電路接線如圖 2-2-1。繼電器在切換控制時，其電感會有逆向電流所以電感端用二極體並聯來吸收，繼電器的驅動使用電晶體的共射極電路，可將 Arduino 發展板數位輸出所提供的電流放大來驅動繼電器，實驗時用三用電表先量測 CN2 的 1(COM)和 2(NC)是否短路，來證實繼電器不動作，將 A4 輸出 HIGH 使繼電器電感通電動作時，再用三用電表先量測 CN2 的 1 和 3(NO)是否短路，來證實繼電器已經作動。

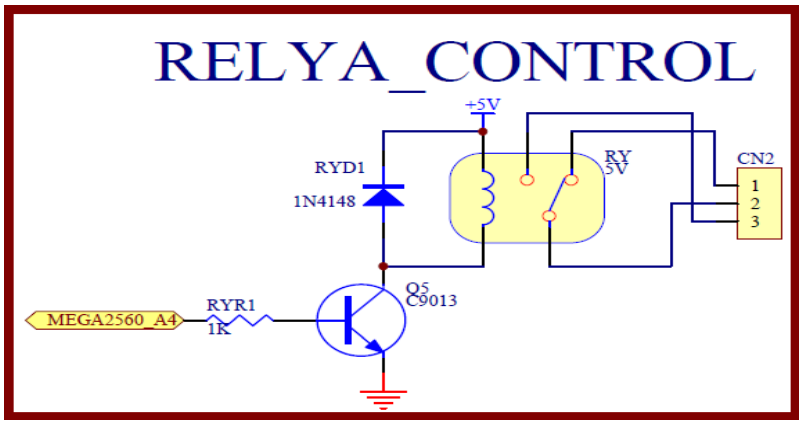


圖2-2-1　繼電器驅動電路

元件：

編號	元件項目	數量	元件名稱
1	Q5	1	NPN 電晶體(C9013)
2	RYD1	1	二極體(1N4148)
3	RYR1	1	1K Ω電阻
4	RY5V	1	單刀雙擲繼電器
5	CN2	1	三個輸出接頭

程式：本實驗 2-2-1 Relay 程式控制繼電器 ON->OFF->ON 的切換時間各一秒。

2-2-1 Relay 程式:

Relay

行號	程式敘述	註解
1	int RY5V = A4;	// 定義繼電器控制接腳
2	void setup(){	// 只會執行一次的程式初始設定函式
3	pinMode(RY5V,OUTPUT);	// 規劃 LED 腳為輸出模式
4	digitalWrite(RY5V, LOW);	// RY5V 輸出 LOW，繼電器不動作
5	}	// 結束 setup()函式
6	void loop(){	// 永遠周而復始的主控制函式
7	digitalWrite(RY5V, HIGH);	// RY5V 輸出 HIGH，繼電器作動
8	delay(1000);	// 呼叫延遲函式等 1000 毫秒=1 秒
9	digitalWrite(RY5V, LOW);	// RY5V 輸出 LOW，繼電器不動作
10	delay(1000);	// 呼叫延遲函式等 1000 毫秒=1 秒
11	}	// 結束 loop()函式

練習：

一、 設計定時關掉的開關，在程式執行後繼電器作動 1 分鐘後關閉?

實驗 2-3：蜂鳴器實習

目的：瞭解蜂鳴器的工作原理及使用 Arduino 程式控制方法。

功能：本實習使用 Arduino MEGA2560 控制蜂鳴器發出聲音。程式(一)中控制蜂鳴器間隔1秒發出嗶聲。程式(二)中使用可變電阻控制蜂鳴器嗶聲的頻率。

原理：蜂鳴器一般可分為壓電式及電磁式的二大類，壓電式蜂鳴器是以壓電陶瓷的壓電效應，來帶動金屬片的振動而發出聲音，所以壓電式蜂鳴器是以方波來驅動；而電磁式的蜂鳴器則是利用電磁的原理，當通電時會將金屬振動膜吸下，不通電時會依振動膜的彈力彈回，所以電磁式的蜂鳴器是 1/2 方波驅動。就工作電壓來區分，電磁式的蜂鳴器，從 1.5 到 24V，壓電式的從 3V 到 220V 都是可行的，但一般壓電的還是建議有 9V 以上的電壓，較能獲得較大的聲音。若以消耗電流來看，電磁式的依電壓的不同，從幾十到上百毫安培都有，壓電式的就比較省電，幾毫安培就可以正常的動作，但是在蜂鳴器啟動時，瞬間需消耗約三倍的電流。驅動方式可分為有源式與無源式蜂鳴器，有源式蜂鳴器只要接上直流電(DC)即可發聲,因為已內建了驅動線路在蜂鳴器中了。本實習使用 Arduino MEGA2560 控制有源式蜂鳴器(如圖 2-3-1)發出聲音。

圖 2-3-1　有源式蜂鳴器實體圖

電路：蜂鳴器與 Arduino MEGA2560 開發板的接線如圖 2-3-2 所示，圖中 Arduino MEGA2560 開發板上的第 A9 支類比接腳對應連接到蜂鳴器的正端接腳。程式(一)中利用 Arduino A9 類比輸出接腳送出週期 2 秒的 1/2 方波驅動蜂鳴器使其風出嗶聲。程式(二)中使用 Arduino A1 類比輸入腳讀取可變電阻的分壓值，再經數位化為 10 位元的值(0~1023)作為送出 1/2 方波的週期值，進而改變蜂鳴器發出嗶聲的頻率，可變電阻電路如圖 2-3-3 所示。

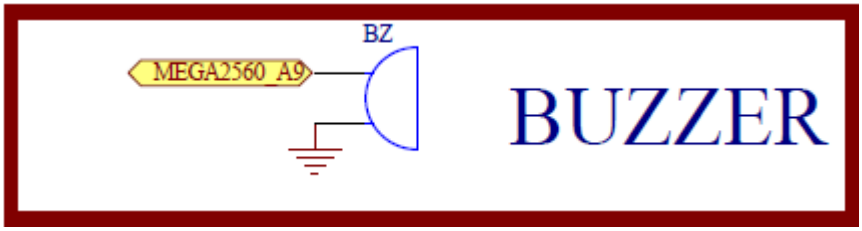


圖 2-3-2 Arduino MEGA2560 對應蜂鳴器的實體接腳電路圖

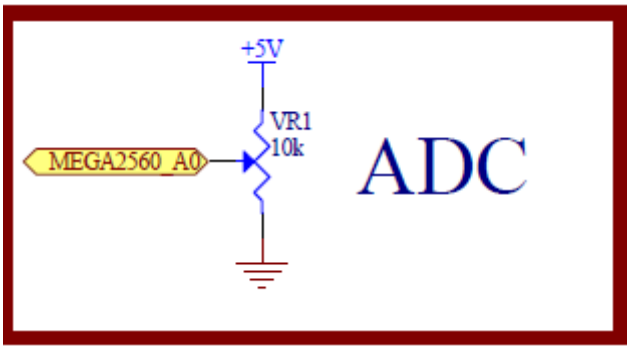


圖 2-3-3 4x4 鍵盤電路

元件：本實習所需元件如表 2-3-1 所示。

表 2-3-1 元件表

編號	元件項目	數量	元件名稱
1	Arduino MEGA2560	1	Arduino 開發板
2	BUZZER	1	5V 電磁式有源式蜂鳴器
3	VR1	1	10kΩ 可變電阻

程式（一）：基本蜂鳴器發聲

P2-3-1

行號	程式敘述	註解
1	int BUZZER = A9;	// 定義 BUZZER 接腳在類比 IO 第 9 支
2	void setup(){	// 只會執行一次的程式初始函式
3	pinMode(BUZZER, OUTPUT);	//規劃 BUZZER 腳為輸出模式
4	}	//結束 setup()函式
5	void loop(){	// 永遠周而復始的主控制函式
6	digitalWrite(BUZZER, HIGH);	// BUZZER 輸出 HIGH，BUZZER 發聲
7	delay(1000);;	//呼叫延遲函式等待 1000 毫秒
8	digitalWrite(BUZZER, LOW);	// BUZZER 輸出 LOW，BUZZER 發聲停止
9	delay(1000);;	//呼叫延遲函式等待 1000 毫秒
10	}	// 結束 loop()函式

程式（二）： 可變電阻控制蜂鳴器嗶聲的頻率

P2-3-2

行號	程式敘述	註解
1	int BUZZER = A9;	// 定義 BUZZER 接腳在類比 IO 第 9 支
2	int VR_pin =A0:	// 定義 VR_pin 接腳在類比 IO 第 0 支
3	int VR_value = 0;	// VR_value = analogRead(VR_pin) 0~1023;
4	void setup(){	// 只會執行一次的程式初始函式
5	pinMode(BUZZER, OUTPUT);	//規劃 BUZZER 腳為輸出模式
6	pinMode(VR_pin, INPUT);	//規劃腳位 VR_pin 為輸入腳
7	}	// 結束 setup()函式
8	void loop(){	// 永遠周而復始的主控制函式
9	VR_value = analogRead(VR_pin);	//讀取腳位 VR_pin 可變電阻的類比訊號放 //入 VR_value 變數中
10	digitalWrite(BUZZER, HIGH);	// BUZZER 輸出 HIGH，BUZZER 發聲
11	delay(VR_value);	//呼叫延遲函式等待 VR_value 毫秒
12	digitalWrite(BUZZER, LOW);	// BUZZER 輸出 LOW，BUZZER 發聲停止
13	delay(VR_value);	//呼叫延遲函式等待 VR_value 毫秒
14	}	// 結束 loop()函式

練習

二、 試設計模擬防空警報的聲響。

實驗 2-4: 四位七段顯示器

目的：了解 Arduino 使用掃描四顆七段顯示器來顯示四位數字，並用計數累加程式來顯示 0000~9999，進一步熟悉七段顯示器陣列查表程式設計的變化性。

功能：首先使用 Arduino MEGA2560 第 2~9 隻數位接腳完成控制一個七段顯示器的八顆 LED(七個段和一個小數點)如圖 2-4-1 所示，可使它重覆顯示 0~9 數字，間隔時間為 1 秒鐘，本實驗使用共陽極七段顯示器加上 8 個 300 毆姆限流電阻，在 Arduino 數位輸出低電位時點亮該段 LED 燈，並一直重覆執行。接著使用四顆的七段顯示器構成四位數七段顯示器顯示數字 0000~9999，由 Arduino MEGA2560 第 2~9 隻數位接腳完成控制四位數的七段顯示器顯示 0~9 數字，並用 Arduino MEGA2560 第 30~31 隻數位接腳驅動 3x8 解碼器來完成四位數字掃描來顯示四位數字，每個七段顯示器的數字停留 4ms，每次顯示四位數字的時間為 300mS(0.3 秒)時間，程式用計數累加 1 來順序產生 0000~9999 共 10000 個四位數字，輸出到四個共陽極七段顯示器，在 Arduino 數位輸出低電位，共陽極為高電位時，點亮該段 LED 燈，順序掃描四個數字，如此循環下去從 0000 顯示到 9999，並一直重複顯示。

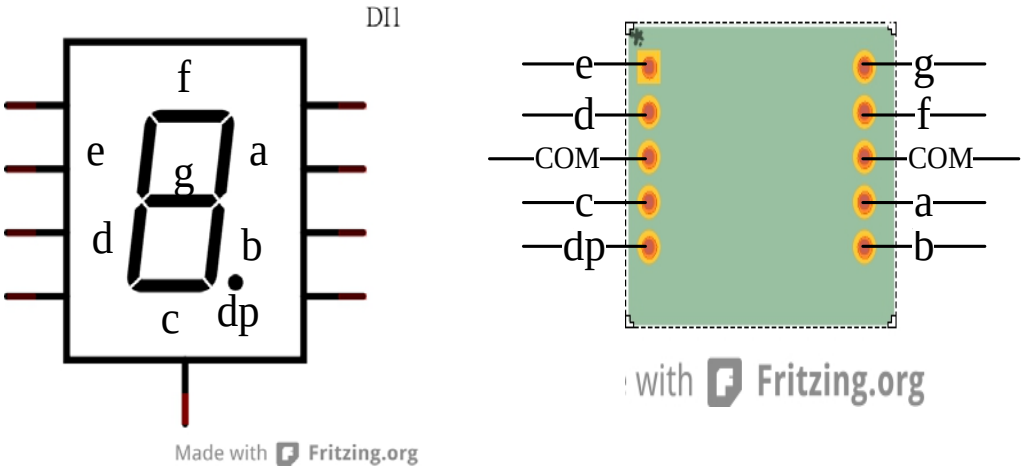


圖 2-4-1 七段顯示器實體與接腳對照圖

原理：要顯示四位數字的七段顯示器，使用掃描可讓接腳使用數減少，但要用視覺暫留效應，更新畫面速率為每秒要超過 60 次，就不會產生閃爍，每次掃描四位數字要小於 1/60 秒，即約 16.7ms，所以本實驗設計一個七段顯示器只停留 4ms，掃描四個七段顯示器要 16ms，也就是更新畫面速率為每秒約 60 次，每次顯示四位數要顯示停留 0.3 秒，再加 1 後顯示下一個數字，即能顯示四位數字 0000~9999。

電路：Arduino MEGA2560 開發板上的第 2~9 隻數位接腳藉由 74AC244 緩衝器和八個

300 歐姆來連接四個七段顯示器的七個段和一個小數點，七個段分別為 a, b, c, d, e, f, g 而小數點為 dp，而四個七段顯示器的共陽極接到 PNP 電晶體的集極，電晶體的射極接到電源 5V，電晶體的基極接到 74LS138 解碼器的輸出，Arduino MEGA2560 的第 30~31 隻數位接腳產生 0~3 的編碼輸入到 74LS138 解碼器來控制四個七段顯示器的共陽極來掃描顯示四位數 0000~9999 的電路，如圖 2-4-2 所示。

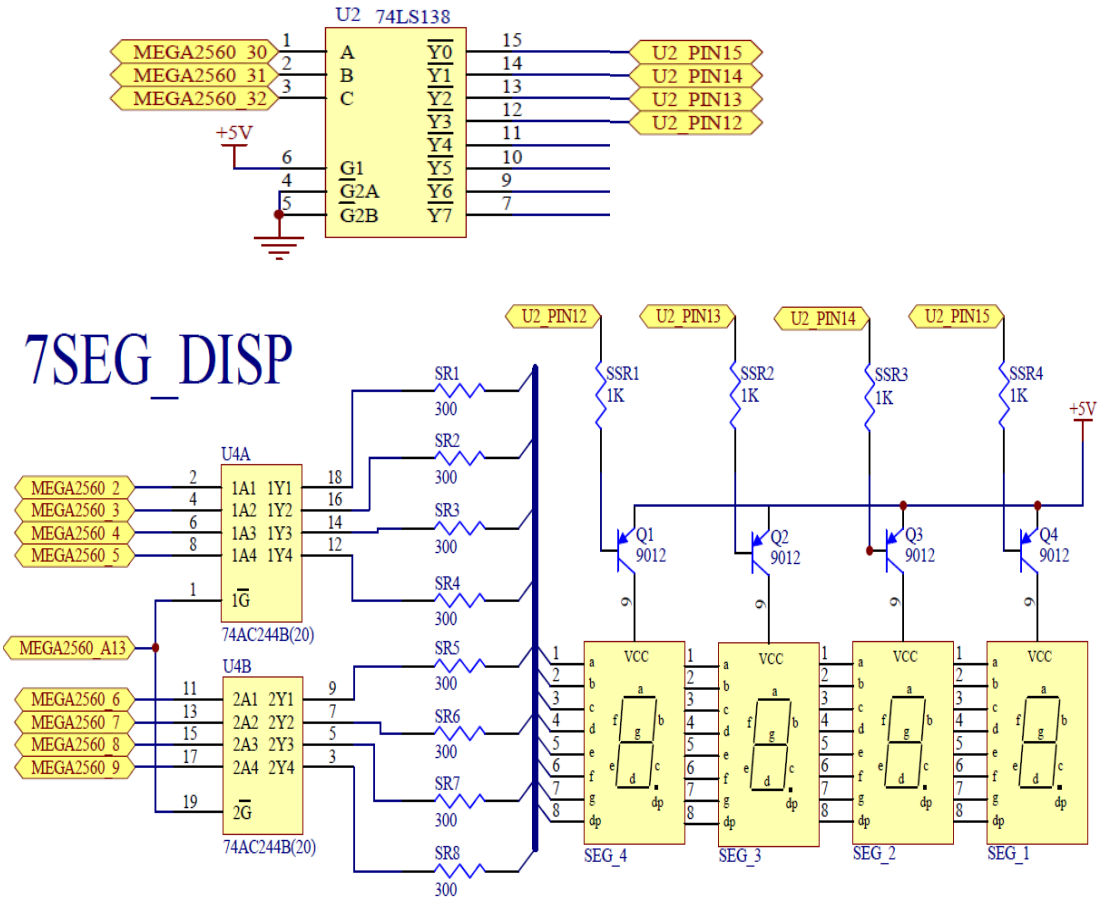


圖2-4-2　四位七段顯示器電路

元件：

編號	元件項目	數量	元件名稱
1	74AC244	1	8 個緩衝閘晶片
2	SEG1~SEG84	4	七段顯示器
3	Q1~Q4	4	PNP 電晶體 9012
4	U2	1	3x8 解碼器 74LS138
5	R1~R8	8	300 Ω電阻

編號	元件項目	數量	元件名稱
6	SSR1~SSR4	4	1k Ω電阻

程式：本實驗共有三個程式來顯示七段顯示器，分別為控制一顆七段顯示器來顯示 0~9 數字各一秒的 2-4-1 Segment_7_LED 程式，還有使用四顆的七段顯示器構成四位數七段顯示器顯示數字 0000~9999 的 2-4-2 For_Digit_SegmentV1 和 2-4-3 For_Digit_SegmentV2 程式，其中 2-4-2 程式使用設定 Arduino MEGA2560 第 2~9 隻數位接腳的方式來顯示數字，因每個數字的七段 LED 均要設定數位輸出為高電位或低電位，不能用迴圈方式來設定，所以程式較大，而 2-4-3 程式將顯示數字的七段 LED 用陣列的查表方式設定，可用 FOR 迴圈來讓七段顯示器顯示 0~9 數字，所以程式較小且較容易理解。顯示四顆的七段顯示器的 2-4-2 和 2-4-3 程式在掃描上是一樣的，由設定要顯示四位數字的個位數、十位數、百位數或千位數分別對應 3x8 編碼器 74LS138 的輸入數字 0(000)、1(001)、2(010)和 3(011)，一直不斷的重覆循環就會看到七段顯示器顯示數字 0000~9999。

2-4-1 Segment_7_LED 程式:

p2-4-1 Segment_7_LED

行號	程式敘述	註解
1	//Arduino pin: 2,3,4,5,6,7,8	// 定義七段 LED 控制接腳
2	byte seven_seg_digits[10][7] = { { 0,0,0,0,0,0,1 },	// 數字 = 0 ，二為陣列 seven_seg_digits 中 0 表示 LED 點亮， 1 表示 LED 熄滅
3	{ 1,0,0,1,1,1,1 },	// 數字 = 1
4	{ 0,0,1,0,0,1,0 },	// 數字 = 2
5	{ 0,0,0,0,1,1,0 },	// 數字 = 3
6	{ 1,0,0,1,1,0,0 },	// 數字 = 4
7	{ 0,1,0,0,1,0,0 },	// 數字 = 5
8	{ 0,1,0,0,0,0,0 },	// 數字 = 6
9	{ 0,0,0,1,1,1,1 },	// 數字 = 7
10	{ 0,0,0,0,0,0,0 },	// 數字 = 8
11	{ 0,0,0,1,1,0,0 },	// 數字 = 9
12	};	// 陣列結束
13	void setup(){	// 只會執行一次的程式初始設定函式
14	pinMode(2, OUTPUT);	// 規劃第 2 腳為輸出模式
15	pinMode(3, OUTPUT);	// 規劃第 3 腳為輸出模式
16	pinMode(4, OUTPUT);	// 規劃第 4 腳為輸出模式
17	pinMode(5, OUTPUT);	// 規劃第 5 腳為輸出模式
18	pinMode(6, OUTPUT);	// 規劃第 6 腳為輸出模式
19	pinMode(7, OUTPUT);	// 規劃第 7 腳為輸出模式
20	pinMode(8, OUTPUT);	// 規劃第 8 腳為輸出模式
21	pinMode(9, OUTPUT);	// 規劃第 9 腳為輸出模式
22	writeDot(1);	// 規劃不顯示小數點

23	pinMode(A13,OUTPUT);	// 規劃 A13 腳為輸出模式
24	digitalWrite(A13, LOW);	// A13 輸出 LOW，制能 74AC244
25	pinMode(30,OUTPUT); pinMode(31,OUTPUT); pinMode(32,OUTPUT);	// 規劃 30~31 腳為輸出模式
26	digitalWrite(32,LOW); digitalWrite(31,LOW); digitalWrite(30,LOW);	// 第 30~31 腳輸出 LOW，使 74LS138 解碼到 0，掃瞄到個位數來顯示個位數字
27	}	// 結束 setup()函式
28	void writeDot(byte dot) {	// 規劃設定小數點顯示副程式
29	digitalWrite(9, dot);	// dot=0 表示顯示小數點，1 不顯示小數點
30	}	// 結束設定小數點顯示副程式
31	void sevenSegWrite(byte digit) {	// 七段顯示器顯示數字副程式
32	byte pin = 2;	// 陣列設定從第 2 腳開始
33	for (byte segCount = 0; segCount < 7; ++segCount) {	// 使用 for 迴圈設定輸出 7 段顯示器 LED 的輸出電壓
34	digitalWrite(pin, seven_seg_digits[digit][segCount]);	// 由陣列 seven_seg_digits 設定輸出 7 段顯示器的數字
35	++pin;	// segCount 從 0 累加到 6 數位輸出腳位累加到第 8 腳，使七段顯示器顯示用 digit 設定的數字
36	}	// for 迴圈結束
37	}	// 七段顯示器顯示數字副程式結束
38	void loop() {	// 永遠周而復始的主控制函式
39	for (byte count = 0; count < 10; count++) {	// 使用 for 迴圈設定輸出 7 段顯示器數字由 0 到 9 顯示
40	delay(1000);	// 呼叫延遲函式等 1000 毫秒=1 秒
41	sevenSegWrite(count);	// 呼叫七段顯示器顯示數字副程式
42	}	// 結束 loop()函式

2-4-2 For_Digit_SegmentV1 程式:

p2-4-2 For_Digit_SegmentV1

行號	程式敘述	註解
1	#define CA1 30	// 定義 3x8 解碼器 74LS138 的 輸入 A 所接的 I/O 腳
2	#define CA2 31	// 定義 3x8 解碼器 74LS138 的 輸入 B 所接的 I/O 腳
3	#define CA3 32	// 定義 3x8 解碼器 74LS138 的 輸入 C 所接的 I/O 腳
4	int delay_time = 4;	// 定義每個七段顯示器顯示的時間 (延遲時間), 預設 4ms
5	void setup(){	// 只會執行一次的程式初始設定函式
6	pinMode(2, OUTPUT);	// 規劃第 2 腳為輸出模式
7	pinMode(3, OUTPUT);	// 規劃第 3 腳為輸出模式
8	pinMode(4, OUTPUT);	// 規劃第 4 腳為輸出模式
9	pinMode(5, OUTPUT);	// 規劃第 5 腳為輸出模式
10	pinMode(6, OUTPUT);	// 規劃第 6 腳為輸出模式
11	pinMode(7, OUTPUT);	// 規劃第 7 腳為輸出模式
12	pinMode(8, OUTPUT);	// 規劃第 8 腳為輸出模式
13	pinMode(9, OUTPUT);	// 規劃第 9 腳為輸出模式
14	digitalWrite(9, HIGH);	// 設定第 9 腳為 HIGH，使 LED 熄

```
                                                    減，不顯示小數點式
15  pinMode(CA1, OUTPUT);                          // 規劃 CA1 第 30 腳為輸出模式
16  pinMode(CA2, OUTPUT);                          // 規劃 CA2 第 31 腳為輸出模式
17  pinMode(CA3, OUTPUT);                          // 規劃 CA3 第 32 腳為輸出模式
18  pinMode(A13,OUTPUT);                           // 規劃 A13 腳為輸出模式
19  digitalWrite(A13, LOW);                        //  A13 輸出 LOW，制能 74AC244
20  }                                              // 結束 setup()函式
21  void loop() {                                  // 永遠周而復始的主控制函式
22  for (unsigned int number = 0; number < 10000;  // FOR 迴圈從 0000 計數到 9999
    number++) {
23      unsigned long startTime = millis();        // 定義 startTime 為經過時間(單位為
                                                    毫秒)
24  for (unsigned long elapsed=0; elapsed < 300;   // 用 FOR 迴圈多工掃描，輪流點亮
    elapsed = millis() - startTime) {               個、十、百、以及千位數的七段顯
                                                    示器，每一四位數字停留 elapsed
                                                    時間共 300mS
25      pickDigit(1);                              // 呼叫 pickDigit()，1 表示顯示個位
                                                    數
26      pickNumber(number%10);                     // 從四位數 number 除 10 的餘數得到
                                                    個位數字
27      delay(delay_time);                         // 呼叫延遲函式等 delay_time= 4 毫
                                                    秒
28      pickDigit(2);                              // 呼叫 pickDigit()，2 表示顯示十位
                                                    數
29      pickNumber((number/10)%10);                // 從四位數 number 除 10 後，再除 10
                                                    的餘數得到十位數字
30      delay(delay_time);                         // 呼叫延遲函式等 delay_time= 4 毫
                                                    秒
31      pickDigit(3);                              // 呼叫 pickDigit()，3 表示顯示百位
                                                    數
32      pickNumber((number/100)%10);               // 從四位數 number 除 100 後，再除
                                                    10 的餘數得到百位數字
33      delay(delay_time);                         // 呼叫延遲函式等 delay_time= 4 毫
                                                    秒
34      pickDigit(4);                              // 呼叫 pickDigit()，4 表示顯示千位
                                                    數
35      pickNumber(number/1000)%10);               // 從四位數 number 除 1000 後，再除
                                                    10 的餘數得到千位數字
36      delay(delay_time);                         // 呼叫延遲函式等 delay_time= 4 毫
                                                    秒
37          }                                      // FOR 迴圈多工掃描結束
38      }                                          // FOR 迴圈從 0000 計數到 9999 結
                                                    束
39  }                                              // 結束 loop()函式
40  void pickDigit(int x) {                        // 選擇顯示的位數副程式 (4:千、3:
                                                    百、2:十、或 1:個位數)
41      digitalWrite(CA1,HIGH);                    // 設定 CA1~3 為 HIGH，即第 30~31
        digitalWrite(CA2,HIGH);                     腳輸出 HIGH，使 74LS138 解碼到
        digitalWrite(CA3,HIGH);                     3，掃描到千位數來顯示千位數字

42    switch(x) {                                  // 使用 switch 選擇顯示的位數
43      case 1:                                    // 1:個位數，數位輸出 000 使 74LS138
    digitalWrite(CA1,LOW); digitalWrite(CA2,LOW);   解碼到 0，掃描到個位數來顯示個
                                                    位數字
```

```
digitalWrite(CA3,LOW); break;
44      case 2:                                                 // 2:十位數,數位輸出 001 使 74LS138 解碼到 1，掃描到十位數來顯示十位數字
digitalWrite(CA1,HIGH); digitalWrite(CA2,LOW);
digitalWrite(CA3,LOW); break;
45      case 3:                                                 // 3:百位數,數位輸出 010 使 74LS138 解碼到 2，掃描到百位數來顯示百位數字
digitalWrite(CA1,LOW); digitalWrite(CA2,HIGH);
digitalWrite(CA3,LOW); break;
46      case 4:                                                 // 4:千位數,數位輸出 011 使 74LS138 解碼到 3，掃描到千位數來顯示千位數字
digitalWrite(CA1,HIGH); digitalWrite(CA2,HIGH);
digitalWrite(CA3,LOW); break;
47    }                                                         // 結束 switch 選擇式
48  }                                                           // 結束選擇顯示的位數副程式
49  void pickNumber(int x) {                                    // 顯示數字 0~9 的副程 式
50  switch(x) {                                                 // 使用 switch 選擇顯示的數字
51                  case 1: one(); break;                       // 1:呼叫 one()副程式顯示數字 1
52                  case 2: two(); break;                       // 2:呼叫 two()副程式顯示數字 2
53                  case 3: three(); break;                     // 3:呼叫 three()副程式顯示數字 3
54                  case 4: four(); break;                      // 4:呼叫 four()副程式顯示數字 4
55                  case 5: five(); break;                      // 5:呼叫 five()副程式顯示數字 5
56                  case 6: six(); break;                       // 6:呼叫 six()副程式顯示數字 6
57                  case 7: seven(); break;                     // 7:呼叫 seven()副程式顯示數字 7
58                  case 8: eight(); break;                     // 8:呼叫 eight()副程式顯示數字 8
59                  case 9: nine(); break;                      // 9:呼叫 nine()副程式顯示數字 9
60                  default: zero();break;                      // 內定:呼叫 zero()副程式顯示數字 0
61    }                                                         // 結束 switch 選擇式
62  }                                                           // 結束顯示數字 0~9 的副程 式
63  void one() {                                                // 顯示數字 '1' 的副程式
64  digitalWrite(2, HIGH);                                      // 設定第 2 腳為 HIGH，使 LED 熄滅，不顯示
65  digitalWrite(3, LOW);                                       // 設定第 3 腳為 LOW,使 LED 點亮,顯示
66  digitalWrite(4, LOW);                                       // 設定第 4 腳為 LOW,使 LED 點亮,顯示
67  digitalWrite(5, HIGH);                                      // 設定第 5 腳為 HIGH，使 LED 熄滅，不顯示
68  digitalWrite(6, HIGH);                                      // 設定第 6 腳為 HIGH，使 LED 熄滅，不顯示
69  digitalWrite(7, HIGH);                                      // 設定第 7 腳為 HIGH，使 LED 熄滅，不顯示
70  digitalWrite(8, HIGH);                                      // 設定第 8 腳為 HIGH，使 LED 熄滅，不顯示
71  }                                                           // 結束顯示數字 '1' 的副程式
72  void two() {                                                // 顯示數字 '2' 的副程式
73  digitalWrite(2, LOW);                                       // 設定第 2 腳為 LOW,使 LED 點亮,顯示
74  digitalWrite(3, LOW);                                       // 設定第 3 腳為 LOW,使 LED 點亮,顯示
75  digitalWrite(4, HIGH);                                      // 設定第 4 腳為 HIGH，使 LED 熄滅，不顯示
```

```
76  digitalWrite(5, LOW);          // 設定第 5 腳為 LOW，使 LED 點亮，顯示
77  digitalWrite(6, LOW);          // 設定第 6 腳為 LOW，使 LED 點亮，顯示
78  digitalWrite(7, HIGH);         // 設定第 7 腳為 HIGH，使 LED 熄滅，不顯示
79  digitalWrite(8, LOW);          // 設定第 8 腳為 LOW，使 LED 點亮，顯示
80  }                              // 結束顯示數字 '2' 的副程式
81  void three() {                 // 顯示數字 '3' 的副程式
82  digitalWrite(2, LOW);          // 設定第 2 腳為 LOW，使 LED 點亮，顯示
83  digitalWrite(3, LOW);          // 設定第 3 腳為 LOW，使 LED 點亮，顯示
84  digitalWrite(4, LOW);          // 設定第 4 腳為 LOW，使 LED 點亮，顯示
85  digitalWrite(5, LOW);          // 設定第 5 腳為 LOW，使 LED 點亮，顯示
86  digitalWrite(6, HIGH);         // 設定第 6 腳為 HIGH，使 LED 熄滅，不顯示
87  digitalWrite(7, HIGH);         // 設定第 7 腳為 HIGH，使 LED 熄滅，不顯示
88  digitalWrite(8, LOW);          // 設定第 8 腳為 LOW，使 LED 點亮，顯示
89  }                              // 結束顯示數字 '3' 的副程式
90  void four() {                  // 顯示數字 '4' 的副程式
91  digitalWrite(2, HIGH);         // 設定第 2 腳為 HIGH，使 LED 熄滅，不顯示
92  digitalWrite(3, LOW);          // 設定第 3 腳為 LOW，使 LED 點亮，顯示
93  digitalWrite(4, LOW);          // 設定第 4 腳為 LOW，使 LED 點亮，顯示
94  digitalWrite(5, HIGH);         // 設定第 5 腳為 HIGH，使 LED 熄滅，不顯示
95  digitalWrite(6, HIGH);         // 設定第 6 腳為 HIGH，使 LED 熄滅，不顯示
96  digitalWrite(7, LOW);          // 設定第 7 腳為 LOW，使 LED 點亮，顯示
97  digitalWrite(8, LOW);          // 設定第 8 腳為 LOW，使 LED 點亮，顯示
98  }                              // 結束顯示數字 '4' 的副程式
99  void five() {                  // 顯示數字 '5' 的副程式
100 digitalWrite(2, LOW);          // 設定第 2 腳為 LOW，使 LED 點亮，顯示
101 digitalWrite(3, HIGH);         // 設定第 3 腳為 HIGH，使 LED 熄滅，不顯示
102 digitalWrite(4, LOW);          // 設定第 4 腳為 LOW，使 LED 點亮，顯示
103 digitalWrite(5, LOW);          // 設定第 5 腳為 LOW，使 LED 點亮，顯示
104 digitalWrite(6, HIGH);         // 設定第 6 腳為 HIGH，使 LED 熄滅，不顯示
105 digitalWrite(7, LOW);          // 設定第 7 腳為 LOW，使 LED 點亮，顯示
106 digitalWrite(8, LOW);          // 設定第 8 腳為 LOW，使 LED 點亮，
```

```
                                        顯示
107 }                                   // 結束顯示數字 '5' 的副程式
108 void six() {                         // 顯示數字 '6' 的副程式
109 digitalWrite(2, LOW);                // 設定第 2 腳為 LOW，使 LED 點亮，
                                        顯示
110 digitalWrite(3, HIGH);               // 設定第 3 腳為 HIGH，使 LED 熄
                                        滅，不顯示
111 digitalWrite(4, LOW);                // 設定第 4 腳為 LOW，使 LED 點亮，
                                        顯示
112 digitalWrite(5, LOW);                // 設定第 5 腳為 LOW，使 LED 點亮，
                                        顯示
113 digitalWrite(6, LOW);                // 設定第 6 腳為 LOW，使 LED 點亮，
                                        顯示
114 digitalWrite(7, LOW);                // 設定第 7 腳為 LOW，使 LED 點亮，
                                        顯示
115 digitalWrite(8, LOW);                // 設定第 8 腳為 LOW，使 LED 點亮，
                                        顯示
116 }                                   // 結束顯示數字 '6' 的副程式
117 void seven() {                       // 顯示數字 '7' 的副程式
118 digitalWrite(2, LOW);                // 設定第 2 腳為 LOW，使 LED 點亮，
                                        顯示
119 digitalWrite(3, LOW);                // 設定第 3 腳為 LOW，使 LED 點亮，
                                        顯示
120 digitalWrite(4, LOW);                // 設定第 4 腳為 LOW，使 LED 點亮，
                                        顯示
121 digitalWrite(5, HIGH);               // 設定第 5 腳為 HIGH，使 LED 熄
                                        滅，不顯示
122 digitalWrite(6, HIGH);               // 設定第 6 腳為 HIGH，使 LED 熄
                                        滅，不顯示
123 digitalWrite(7, HIGH);               // 設定第 7 腳為 HIGH，使 LED 熄
                                        滅，不顯示
124 digitalWrite(8, HIGH);               // 設定第 8 腳為 HIGH，使 LED 熄
                                        滅，不顯示
125 }                                   // 結束顯示數字 '7' 的副程式
126 void eight() {                       // 顯示數字 '8' 的副程式
127 digitalWrite(2, LOW);                // 設定第 2 腳為 LOW，使 LED 點亮，
                                        顯示
128 digitalWrite(3, LOW);                // 設定第 3 腳為 LOW，使 LED 點亮，
                                        顯示
129 digitalWrite(4, LOW);                // 設定第 4 腳為 LOW，使 LED 點亮，
                                        顯示
130 digitalWrite(5, LOW);                // 設定第 5 腳為 LOW，使 LED 點亮，
                                        顯示
131 digitalWrite(6, LOW);                // 設定第 6 腳為 LOW，使 LED 點亮，
                                        顯示
132 digitalWrite(7, LOW);                // 設定第 7 腳為 LOW，使 LED 點亮，
                                        顯示
133 digitalWrite(8, LOW);                // 設定第 8 腳為 LOW，使 LED 點亮，
                                        顯示
134 }                                   // 結束顯示數字 '8' 的副程式
135 void nine() {                        // 顯示數字 '9' 的副程式
136 digitalWrite(2, LOW);                // 設定第 2 腳為 LOW，使 LED 點亮，
                                        顯示
137 digitalWrite(3, LOW);                // 設定第 3 腳為 LOW，使 LED 點亮，
                                        顯示
```

138	digitalWrite(4, LOW);	// 設定第 4 腳為 LOW，使 LED 點亮，顯示
139	digitalWrite(5, LOW);	// 設定第 5 腳為 LOW，使 LED 點亮，顯示
140	digitalWrite(6, HIGH);	// 設定第 6 腳為 HIGH，使 LED 熄滅，不顯示
141	digitalWrite(7, LOW);	// 設定第 7 腳為 LOW，使 LED 點亮，顯示
142	digitalWrite(8, LOW);	// 設定第 8 腳為 LOW，使 LED 點亮，顯示
143	}	// 結束顯示數字 '9' 的副程式
144	void six() {	// 顯示數字 '0' 的副程式
145	digitalWrite(2, LOW);	// 設定第 2 腳為 LOW，使 LED 點亮，顯示
146	digitalWrite(3, LOW);	// 設定第 3 腳為 LOW，使 LED 點亮，顯示
147	digitalWrite(4, LOW);	// 設定第 4 腳為 LOW，使 LED 點亮，顯示
148	digitalWrite(5, LOW);	// 設定第 5 腳為 LOW，使 LED 點亮，顯示
149	digitalWrite(6, LOW);	// 設定第 6 腳為 LOW，使 LED 點亮，顯示
150	digitalWrite(7, LOW);	// 設定第 7 腳為 LOW，使 LED 點亮，顯示
151	digitalWrite(8, HIGH);	// 設定第 8 腳為 HIGH，使 LED 熄滅，不顯示
152	}	// 結束顯示數字 '0' 的副程式

2-4-3 For_Digit_SegmentV2 程式:

p2-4-3 For_Digit_SegmentV2

行號	程式敘述	註解
1	#define CA1 30	// 定義 3x8 解碼器 74LS138 的 輸入 A 所接的 I/O 腳
2	#define CA2 31	// 定義 3x8 解碼器 74LS138 的 輸入 B 所接的 I/O 腳
3	#define CA3 32	// 定義 3x8 解碼器 74LS138 的 輸入 C 所接的 I/O 腳
4	byte segs[7] = { 2, 3, 4, 5, 6, 7, 8 };	// 定義七個節段的腳位，將 A, B, C, D, E, F, G 依序放入陣列
5	byte seven_seg_digits[10][7] = { { 0,0,0,0,0,0,1 },	// 數字 = 0 ，二維陣列 seven_seg_digits 中 0 表示 LED 點亮， 1 表示 LED 熄滅
6	{ 1,0,0,1,1,1,1 },	// 數字 = 1
7	{ 0,0,1,0,0,1,0 },	// 數字 = 2
8	{ 0,0,0,0,1,1,0 },	// 數字 = 3
9	{ 1,0,0,1,1,0,0 },	// 數字 = 4
10	{ 0,1,0,0,1,0,0 },	// 數字 = 5
11	{ 0,1,0,0,0,0,0 },	// 數字 = 6
12	{ 0,0,0,1,1,1,1 },	// 數字 = 7
13	{ 0,0,0,0,0,0,0 },	// 數字 = 8
14	{ 0,0,0,1,1,0,0 },	// 數字 = 9
15	};	// 陣列結束
16	int delay_time = 4;	// 定義每個七段顯示器顯示的時間

```
                                                    (延遲時間), 預設 4ms
17                  { 1,0,0,1,1,0,0 },              // 數字 = 4
18                  { 0,1,0,0,1,0,0 },              // 數字 = 5
19                  { 0,1,0,0,0,0,0 },              // 數字 = 6
20                  { 0,0,0,1,1,1,1 },              // 數字 = 7
21                  { 0,0,0,0,0,0,0 },              // 數字 = 8
22                  { 0,0,0,1,1,0,0 },              // 數字 = 9
23                      };                          // 陣列結束
24  void setup(){                                   // 只會執行一次的程式初始設定函式
25  pinMode(2, OUTPUT);                             // 規劃第 2 腳為輸出模式
26  pinMode(3, OUTPUT);                             // 規劃第 3 腳為輸出模式
27  pinMode(4, OUTPUT);                             // 規劃第 4 腳為輸出模式
28  pinMode(5, OUTPUT);                             // 規劃第 5 腳為輸出模式
29  pinMode(6, OUTPUT);                             // 規劃第 6 腳為輸出模式
30  pinMode(7, OUTPUT);                             // 規劃第 7 腳為輸出模式
31  pinMode(8, OUTPUT);                             // 規劃第 8 腳為輸出模式
32  pinMode(9, OUTPUT);                             // 規劃第 9 腳為輸出模式
33  digitalWrite(9, HIGH);                          // 設定第 9 腳為 HIGH，使 LED 熄滅，不顯示小數點式
34  pinMode(CA1, OUTPUT);                           // 規劃 CA1 第 30 腳為輸出模式
35  pinMode(CA2, OUTPUT);                           // 規劃 CA2 第 31 腳為輸出模式
36  pinMode(CA3, OUTPUT);                           // 規劃 CA3 第 32 腳為輸出模式
37  pinMode(A13,OUTPUT);                            // 規劃 A13 腳為輸出模式
38  digitalWrite(A13, LOW);                         //  A13 輸出 LOW，制能 74AC244
39  }                                               // 結束 setup()函式
40  void loop() {                                   // 永遠周而復始的主控制函式
41  for (unsigned int number = 0; number < 10000;   // FOR 迴圈從 0000 計數到 9999
    number++) {
42      unsigned long startTime = millis();         // 定義 startTime 為經過時間(單位為毫秒)
43  for (unsigned long elapsed=0; elapsed < 300;    // 用 FOR 迴圈多工掃描，輪流點亮個、十、百、以及千位數的七段顯示器，每一四位數字停留 elapsed 時間共 300mS
    elapsed = millis() - startTime) {
44      lightDigit1(number%10);                     // 從四位數 number 除 10 的餘數得到個位數字，並呼叫 lightDigit1()來顯示個位數
45      delay(delay_time);                          // 呼叫延遲函式等 delay_time= 4 毫秒
46      lightDigit1(number%10);                     // 從四位數 number 除 10 的餘數得到個位數字，並呼叫 lightDigit1()來顯示個位數
47      delay(delay_time);                          // 呼叫延遲函式等 delay_time= 4 毫秒
48      lightDigit2((number/10)%10);                // 從四位數 number 除 10 後，再除 10 的餘數得到十位數字，並呼叫 lightDigit2()來顯示十位數
49      delay(delay_time);                          // 呼叫延遲函式等 delay_time= 4 毫秒
50      lightDigit3((number/100)%10);               // 從四位數 number 除 100 後，再除 10 的餘數得到百位數字，並呼叫 lightDigit3()來顯示百位數
51      delay(delay_time);                          // 呼叫延遲函式等 delay_time= 4 毫秒
```

```
52      lightDigit4((number/1000)%10);                  // 從四位數 number 除 1000 後，再除 10 的餘數得到千位數字，並呼叫 lightDigit4()來顯示千位數
53      delay(delay_time);                              // 呼叫延遲函式等 delay_time= 4 毫秒
54          }                                           // FOR 迴圈多工掃描結束
55      }                                               // FOR 迴圈從 0000 計數到 9999 結束
56  }                                                   // 結束 loop()函式
57  void pickDigit(int x) {                             // 選擇顯示的位數副程式 (4:千、3:百、2:十、或 1:個位數)
58      digitalWrite(CA1,HIGH);                         // 設定 CA1~3 為 HIGH，即第 30~31 腳輸出 HIGH，使 74LS138 解碼到 3，掃描到千位數來顯示千位數字
        digitalWrite(CA2,HIGH);
        digitalWrite(CA3,HIGH);
59      switch(x) {                                     // 使用 switch 選擇顯示的位數
60          case 1:                                     // 1:個位數，數位輸出 000 使 74LS138 解碼到 0，掃描到個位數來顯示個位數字
        digitalWrite(CA1,LOW); digitalWrite(CA2,LOW);
        digitalWrite(CA3,LOW); break;
61          case 2:                                     // 2:十位數，數位輸出 001 使 74LS138 解碼到 1，掃描到十位數來顯示十位數字
        digitalWrite(CA1,HIGH); digitalWrite(CA2,LOW);
        digitalWrite(CA3,LOW); break;
62          case 3:                                     // 3:百位數，數位輸出 010 使 74LS138 解碼到 2，掃描到百位數來顯示百位數字
        digitalWrite(CA1,LOW); digitalWrite(CA2,HIGH);
        digitalWrite(CA3,LOW); break;
63          case 4:                                     // 4:千位數，數位輸出 011 使 74LS138 解碼到 3，掃描到千位數來顯示千位數字
        digitalWrite(CA1,HIGH); digitalWrite(CA2,HIGH);
        digitalWrite(CA3,LOW); break;
64      }                                               // 結束 switch 選擇式
65  }                                                   // 結束選擇顯示的位數副程式
66      void lightDigit1(byte number) {                 // 顯示個位數字副程式
67      pickDigit(1);                                   // 呼叫 pickDigit()，1 表示顯示個位數
68      lightSegments(number);                          // 呼叫函式 lightSegments()使七段顯示器顯示數字
69          }                                           // 結束顯示個位數字副程式
70      void lightDigit2(byte number) {                 // 顯示十位數字副程式
71      pickDigit(2);                                   // 呼叫 pickDigit()，2 表示顯示十位數
72      lightSegments(number);                          // 呼叫函式 lightSegments()使七段顯示器顯示數字
73          }                                           // 結束顯示十位數字副程式
74      void lightDigit3(byte number) {                 // 顯示百位數字副程式
75      pickDigit(3);                                   // 呼叫 pickDigit()，3 表示顯示百位數
76      lightSegments(number);                          // 呼叫函式 lightSegments()使七段顯示器顯示數字
77          }                                           // 結束顯示百位數字副程式
```

```
78  void lightDigit4(byte number) {                  // 顯示千位數字副程式
79  pickDigit(4);                                    // 呼叫 pickDigit()，4 表示顯示千位數
80  lightSegments(number);                           // 呼叫函式 lightSegments()使七段顯示器顯示數字
81      }                                            // 結束顯示千位數字副程式
82  void lightSegments(byte number) {                // 使七段顯示器顯示數字的副程式
83  for (int i = 0; i < 7; i++) {                    // 使用 FOR 迴圈依序驅動七段和小數點 LED 的亮滅
84  digitalWrite(segs[i],
    seven_seg_digits[number][i]);                    // 使用 seven_seg_digits 矩陣值來設定數位輸出，用來使七段顯示器顯示數字
85      }                                            // 結束 FOR 迴圈
86  }                                                // 結束使七段顯示器顯示數字的副程式
```

練習：

一、 利用查表法完成一個七段顯示器顯示 0~F。

二、 利用四個七段顯示器完成 99 分 59 秒的計時碼表，以左邊兩個七段顯示器顯示 00~99 和右邊兩個七段顯示器顯示 00~59 來完成。

實驗 2-5：RGB LED 混光實習

目的：瞭解 RGB LED 混光的工作原理及使用 Arduino 程式控制方法。

功能：本實習使用 Arduino MEGA2560 控制 RGB LED 混光達到變換顏色的目的。程式(一)中使用 PWM 控制 RGB LED 三個燈依序由最亮到最暗漸層顯示。程式(二)中使用亂數函式指令進行 PWM 控制自動混光而變色。

原理：RGB(紅綠藍)LED 封裝的方式有插件式(DIP) (如圖 2-5-1)與表面黏著型(SMD)兩種(如圖 2-5-2)，外型看起來就像普通的 LED 一樣，但是和一般 LED 不同的是 RGB LED 封裝內，有三個 LED，一個紅色，一個綠色的，一個藍色的。透過 PWM 方法控制各個 LED 的亮度，就可以混合出幾乎任何你想要的顏色。

插件式 RGB LED 共有 4 支引腳，可分為共陽與共陰兩種，常見的共用腳是第二引腳，也是最長的那支引腳；而常見表面黏著型 RGB LED 共有 6 支引腳，本實習即使用表面黏著型 RGB LED 為控制的元件。使用 RGB LED 需注意每個 LED 需要串聯限流電阻（約 100 ~330Ω）以防止太大的電流流過而燒毀。

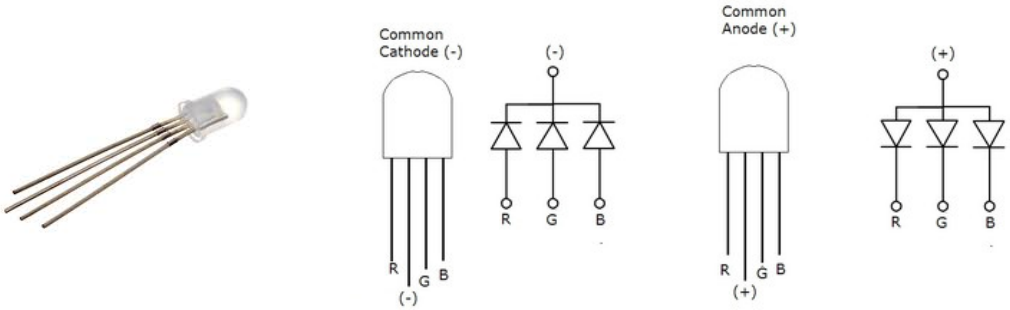


圖 2-5-1　插件式 RGB LED 實體圖與內部電路圖

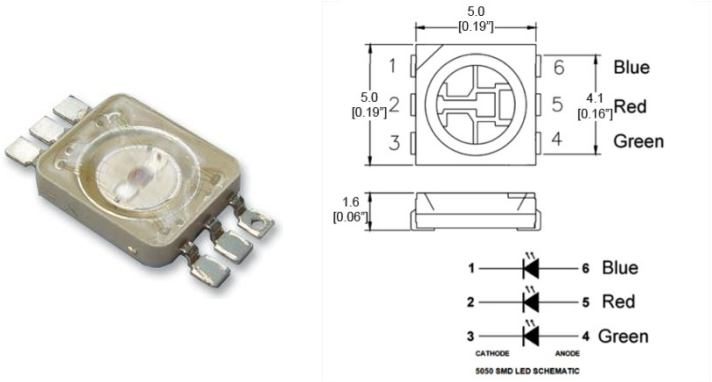


圖 2-5-2　表面黏著型 RGB LED 實體圖與內部電路圖

電路： RGB LED 與 Arduino MEGA2560 開發板的接線如圖 2-5-3 所示，圖中 Arduino MEGA2560 開發板上的第 45、46 與 47 支數位接腳對應連接到表面黏著型 RGB LED 陰極端的 RGB 接腳上。程式(一)中使用 PWM 控制 RGB LED 三個燈依序由最亮到最暗漸層顯示。程式(二)中使用使用亂數函式指令進行 PWM 控制，同時控制三個 LED 燈產生不同亮度進一步達到混光而變色。

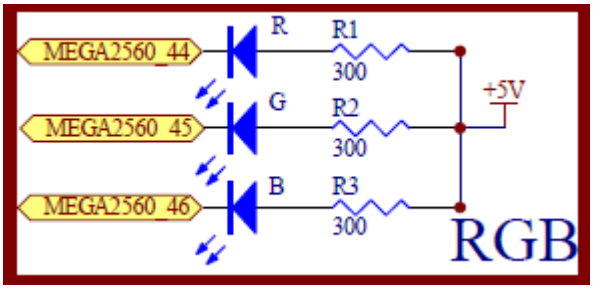


圖 2-5-3 Arduino MEGA2560 對應 RGB LED 的實體接腳電路圖

元件：本實習所需元件如表 2-5-1 所示。

表 2-5-1 元件表

編號	元件項目	數量	元件名稱
1	Arduino MEGA2560	1	Arduino 開發板
2	RGB LED	1	表面黏著型 RGB LED

程式（一）：基本 RGB LED 漸層顯示

P2-5-1

行號	程式敘述	註解
1	int redPin = 44;	// 定義 Red LED 接腳在數位 IO 第 44 支
2	int grnPin= 45;	// 定義 Green LED 接腳在數位 IO 第 45 支
3	int bluPin= 46;	// 定義 Blue LED 接腳在數位 IO 第 45 支
4	int speed = 10;	// 定義漸層亮度變化的時間為 10 毫秒
5	void setup(){	// 只會執行一次的程式初始函式
6	pinMode(redPin, OUTPUT);	// 規劃 redPin 腳為輸出模式
7	pinMode(grnPin, OUTPUT);	// 規劃 grnPin 腳為輸出模式
8	pinMode(bluPin, OUTPUT);	// 規劃 bluPin 腳為輸出模式
9	}	// 結束 setup()函式
10	void loop(){	// 永遠周而復始的主控制函式
11	analogWrite(redPin, 255);	// Red LED OFF
12	analogWrite(grnPin, 255);	// Green LED OFF
13	analogWrite(bluPin, 255);	// Blue LED OFF
14	for (int i = 0; i <255 ; i++) {	// 使用 PWM 控制 Red LED 由最亮到最暗
15	analogWrite(redPin, i);	// 漸層顯示
16	delay(speed);	// 呼叫延遲函式等待 speed 毫秒
17	}	// 結束 for 迴圈
18	delay(100);;	// 呼叫延遲函式等待 100 毫秒
19	for (int i = 0; i <255 ; i++) {	// 使用 PWM 控制 Green LED 由最亮到最暗
20	analogWrite(grnPin, i);	// 漸層顯示
21	delay(speed);	// 呼叫延遲函式等待 speed 毫秒
22	}	// 結束 for 迴圈
23	delay(100);;	// 呼叫延遲函式等待 100 毫秒
24	for (int i = 0; i <255 ; i++) {	// 使用 PWM 控制 Blue LED 由最亮到最暗
25	analogWrite(bluPin, i);	// 漸層顯示
26	delay(speed);	// 呼叫延遲函式等待 speed 毫秒
27	}	// 結束 for 迴圈
28	delay(100);;	// 呼叫延遲函式等待 100 毫秒
29	}	// 結束 loop()函式

程式（二）：RGB LED 混光顯示

P2-5-2

行號	程式敘述	註解
1	int redPin = 44;	// 定義 Red LED 接腳在數位 IO 第 44 支
2	int grnPin= 45;	// 定義 Green LED 接腳在數位 IO 第 45 支
3	int bluPin= 46;	// 定義 Blue LED 接腳在數位 IO 第 45 支
4	int speed = 10;	// 定義漸層亮度變化的時間為 10 毫秒
5	int r_value = 0;	// 宣告全域變數 r_value
6	int g_value = 0;	// 宣告全域變數 g_value
7	int b_value = 0;	// 宣告全域變數 b_value
8	void setup(){	// 只會執行一次的程式初始函式
9	pinMode(redPin, OUTPUT);	// 規劃 redPin 腳為輸出模式
10	pinMode(grnPin, OUTPUT);	// 規劃 grnPin 腳為輸出模式
11	pinMode(bluPin, OUTPUT);	// 規劃 bluPin 腳為輸出模式
12	analogWrite(redPin, 255);	// Red LED OFF
13	analogWrite(grnPin, 255);	// Green LED OFF
14	analogWrite(bluPin, 255);	// Blue LED OFF
15	}	// 結束 setup()函式
16	void loop(){	// 永遠周而復始的主控制函式
17	r_value = random(0, 255);	// 亂數產生 r_value
18	g_value = random(0, 255);	// 亂數產生 g_value
19	b_value = random(0, 255);	// 亂數產生 b_value
20	analogWrite(redPin, r_value);	// PWM 控制輸出 Red LED 亮度值
21	analogWrite(redPin, g_value);	// PWM 控制輸出 Green LED 亮度值
22	analogWrite(redPin, b_value);	// PWM 控制輸出 Blue LED 亮度值
23	delay(speed);	// 呼叫延遲函式等待 speed 毫秒
24	}	// 結束 loop()函式

練習：

一、試設計旋轉可變電阻而可以改變 RGB LED 顏色顯示。

實驗 2-6：8x8 點矩陣顯示器實驗

目的： 瞭解 8×8 點矩陣顯示器的字形設計與控制方法

功能： 當電源 ON 時，在顯示器上顯示英文字母〝A〞

原理： 8x8 點矩陣顯示器其內部結構如圖 2-6-1 所示，同一列中所有 LED 的陽(P)極連接在一起再接到外部接腳，而同一行中所有 LED 的陰(N)極也接在一起再接到外部接腳。因此我們要點亮某一個 LED 時，只要提供正電壓給對應的列，再將其 N 極所在的行接地，那麼 LED 便處於順向偏壓狀態，就能發亮了。因此，我們可以藉著控制顯示器上 LED 的亮與滅，使其顯示出數字或符號。如圖 2-6-2 為英文字母〝A〞在 8x8 點矩陣顯示器上的顯示造形，其中黑色部分表示點亮的 LED。因此要顯示某一個造形時，必須先將代表這個造形的資料碼找出來，然後將其放入程式中，再利用掃描的方式將其取出顯示。

本實驗是以 Pin 2~9 經 74AC244B 控制 8×8 點矩陣顯示器的 R0~R7，Pin A14 接到 74AC244B 的三態控制腳，Pin 2 控制 R0、Pin 3 控制 R1、…、Pin 9 控制 R7；Pin 30~32 經 74LS138 解碼控制 8×8 點矩陣顯示器的 C0~C7。因此資料碼由 Pin 2~9 輸出，而掃描碼則由 Pin 30~32 輸出。代表英文字母〝A〞之資料碼如圖 2-6-3 所示。圖 2-6-4 為 74LS244 接腳及邏輯電路圖，圖 2-6-5 為 74LS138 接腳圖，圖 2-6-6 為 74LS138 真值表。

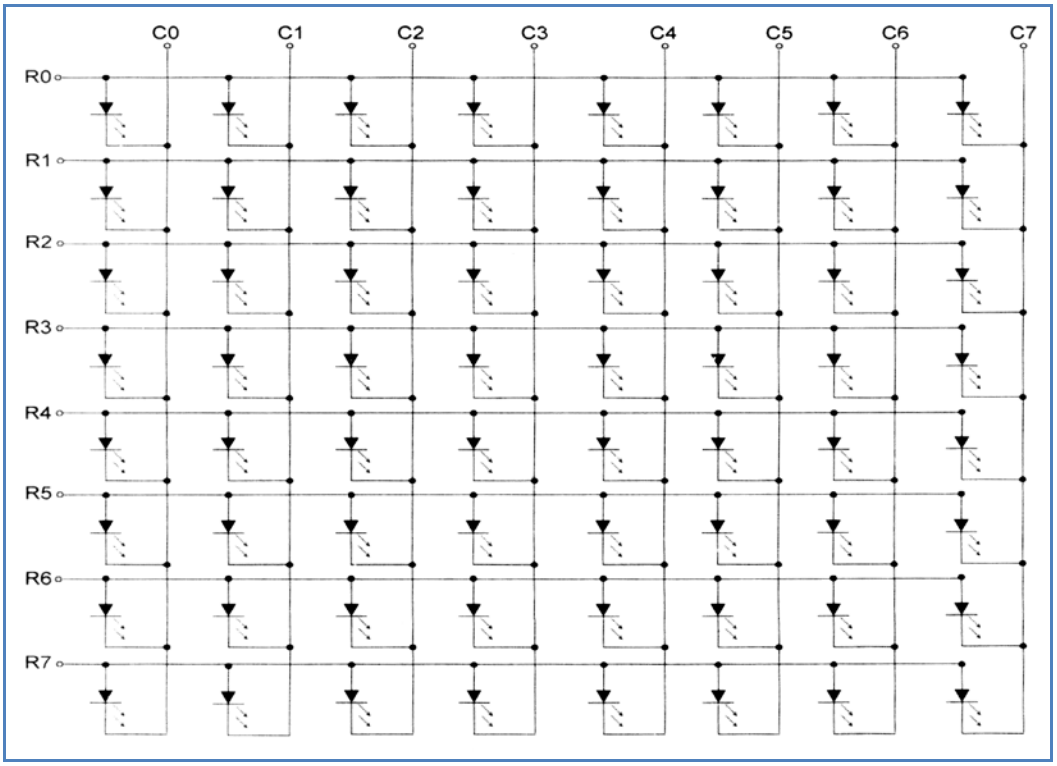


圖 2-6-1、8×8 點矩陣顯示器的內部結構

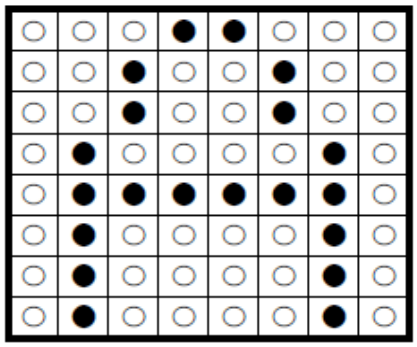


圖 2-6-2、英文字母〝A〞之造形

行 / 列	C0	C1	C2	C3	C4	C5	C6	C7
R0	0	0	0	1	1	0	0	0
R1	0	0	1	0	0	1	0	0
R2	0	0	1	0	0	1	0	0
R3	0	1	0	0	0	0	1	0
R4	0	1	0	0	0	0	1	0
R5	0	1	1	1	1	1	1	0
R6	0	1	0	0	0	0	1	0
R7	0	1	0	0	0	0	1	0

圖 2-6-3、〝A〞之資料碼

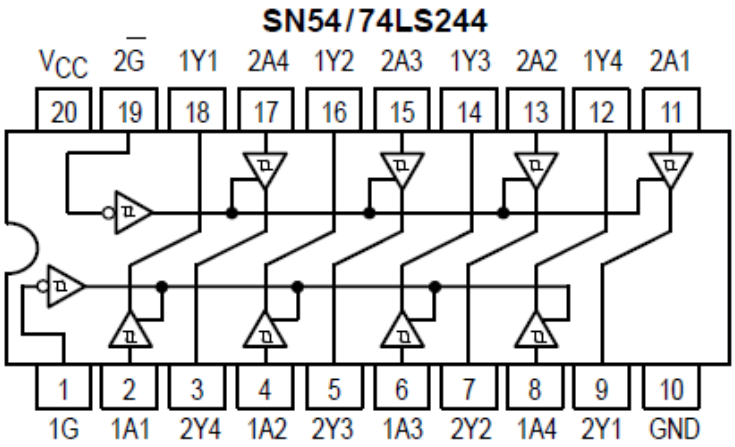


圖 2-6-4、74LS244 接腳及邏輯電路圖

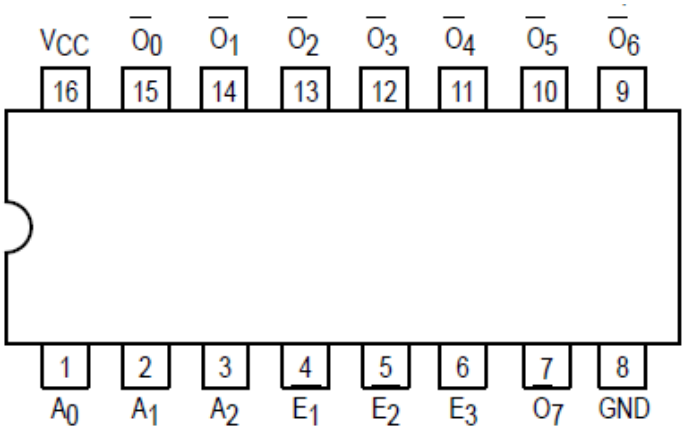


圖 2-6-5、74LS138 接腳圖

TRUTH TABLE

INPUTS						OUTPUTS							
$\overline{E_1}$	$\overline{E_2}$	E_3	A_0	A_1	A_2	$\overline{O_0}$	$\overline{O_1}$	$\overline{O_2}$	$\overline{O_3}$	$\overline{O_4}$	$\overline{O_5}$	$\overline{O_6}$	$\overline{O_7}$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

圖 2-6-6、74LS138 真值表

電路： 74LS138 與 Arduino MEGA2560 開發板的接線如表 2-6-1 所示，8x8 點矩陣經過 74AC244B 與 Arduino MEGA2560 開發板的接線如表 2-6-2 所示，電路如圖 2-6-7 所示。

表 2-6-1　74LS138 與 Arduino MEGA2560 接線

	74LS138	MEGA2560
1	A	D30
2	B	D31
3	C	D32

表 2-6-2　LED 經過 74AC244B 與 Arduino MEGA2560 接線

	8x8 點矩陣	MEGA2560
1	R0	D2
2	R1	D3

3	R2	D4
4	R3	D5
5	R4	D6
6	R5	D7
7	R6	D8
8	R7	D9

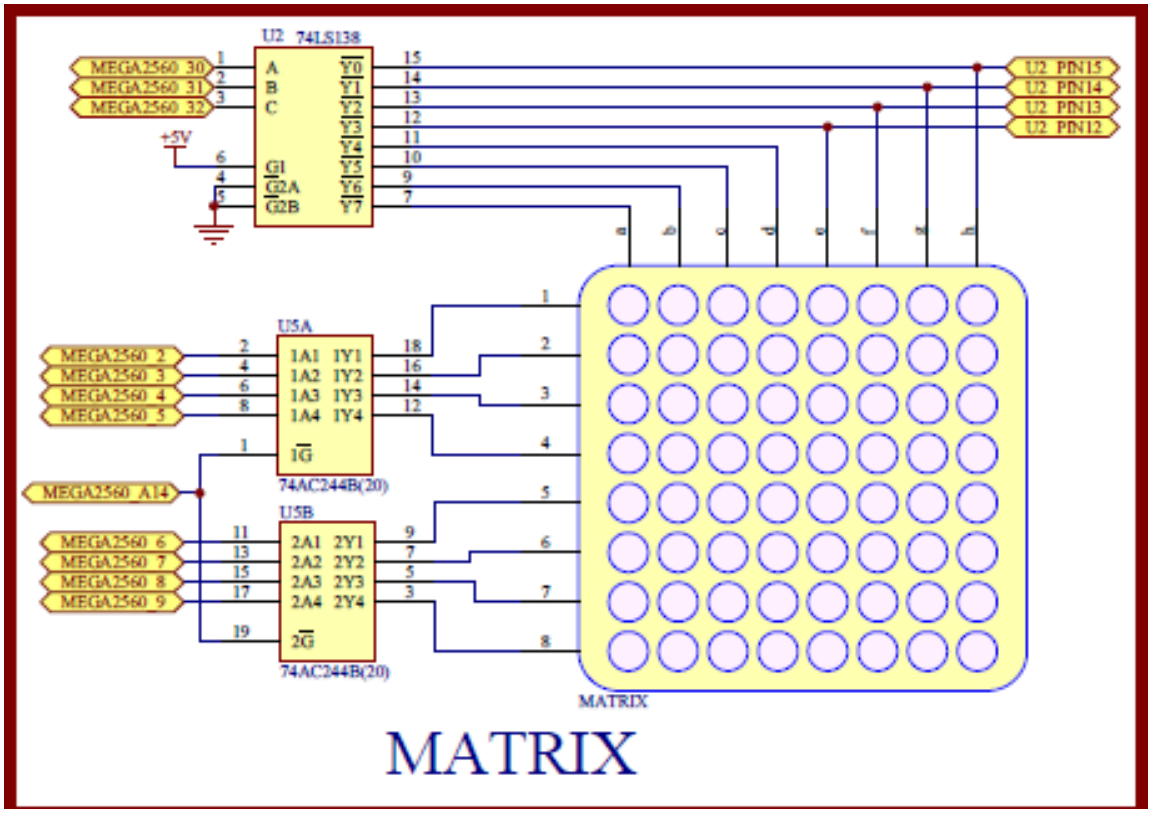


圖 2-6-7　8X8 點矩陣顯示電路

元件：

編號	元件項目	數量	元件名稱
1	MEGA2560	1	Arduino 開發板
2	MATRIX	1	8X8 點矩陣 LED
3	U2	1	74LS138
4	U5	1	74AC244B

程式：

行號	程式敘述	註解
1	#define Ls138_A 30	// 定義 74138 A 腳位
2	#define Ls138_B 31	// 定義 74138 B 腳位
3	#define Ls138_C 32	// 定義 74138 C 腳位
4	#define R0 2	// 定義 8x8 點矩陣 LED ROW0 連接腳位
5	#define R1 3	// 定義 8x8 點矩陣 LED ROW1 連接腳位
6	#define R2 4	// 定義 8x8 點矩陣 LED ROW2 連接腳位
7	#define R3 5	// 定義 8x8 點矩陣 LED ROW3 連接腳位
8	#define R4 6	// 定義 8x8 點矩陣 LED ROW4 連接腳位
9	#define R5 7	// 定義 8x8 點矩陣 LED ROW5 連接腳位
10	#define R6 8	// 定義 8x8 點矩陣 LED ROW6 連接腳位
11	#define R7 9	// 定義 8x8 點矩陣 LED ROW7 連接腳位
12	#define row_size 8	// 定義 Matrix row 大小
13	#define col_size 8	// 定義 Matrix colum 小
14	#define delay_time 300	// 延遲時間
15		//字的數目
16	#define data_A {0, 0, 0, 1, 1, 0, 0, 0,\ 0, 0, 1, 0, 0, 1, 0, 0,\ 0, 0, 1, 0, 0, 1, 0, 0,\ 0, 1, 0, 0, 0, 0, 1, 0,\ 0, 1, 1, 1, 1, 1, 1, 0,\ 0, 1, 0, 0, 0, 0, 1, 0,\ 0, 1, 0, 0, 0, 0, 1, 0,\ 0, 1, 0, 0, 0, 0, 1, 0}	//顯示 A 的資料
17	boolean word_array[4][row_size][col_size] = {data_A};	// 所有要顯示字的 array
18	boolean led[row_size][col_size] ;	// LED Matrix
19	byte row_pin[row_size]={R0, R1, R2, R3, R4, R5, R6, R7};	// 水平方向 led
20	void setup(){	// 初始設定
21	pinMode(Ls138_A, OUTPUT);	//設定 PIN Ls138_A 為 OUTPUT
22	pinMode(Ls138_B, OUTPUT);	//設定 PIN Ls138_B 為 OUTPUT
23	pinMode(Ls138_C, OUTPUT);	//設定 PIN Ls138_C 為 OUTPUT
24	pinMode(R0, OUTPUT);	//設定 R0 為 OUTPUT

```
25      pinMode(R1, OUTPUT);                                  //設定 R1 為 OUTPUT
26      pinMode(R2, OUTPUT);                                  //設定 R2 為 OUTPUT
27      pinMode(R3, OUTPUT);                                  //設定 R3 為 OUTPUT
28      pinMode(R4, OUTPUT);                                  //設定 R4 為 OUTPUT
29      pinMode(R5, OUTPUT);                                  //設定 R5 為 OUTPUT
30      pinMode(R6, OUTPUT);                                  //設定 R6 為 OUTPUT
31      pinMode(R7, OUTPUT);                                  //設定 R7 為 OUTPUT
32      clear_led();                                          //將 8x8 點矩陣 LED 全滅
33      pinMode(A14, OUTPUT);                                 //設定 A14 為 OUTPUT
34      digitalWrite(A14, LOW) ;                              //A14 為 LOW，使 74AC244 動作狀
                                                              態
35  }                                                         //結束 setup 函式
36
37  void loop(){                                              // 永遠周而復始的主控制函式
38      word_to_led(0);                                       // 將第一個 word 讀入 led
39      display_led(led, 1000);                               //顯示 led[][]內字組
40  }                                                         //結束 loop()函式
41
42  void word_to_led(int n){                                  // 將第 n 個 word 讀入 led
43      for (int i = 0; i < row_size; i++)
44          for (int j = 0; j < col_size; j++)
45              led[i][j] = word_array[n][i][j];
46  }                                                         //結束 word_to_led ()
47
48  void clear_led(){                                         // 將 8x8 點矩陣 LED 全滅函式
49      for (int i = 0; i < row_size; i++)
50          digitalWrite(row_pin[i], LOW);
51  }                                                         //結束 clear_led ()函式
52
53  void display_led(byte led[row_size][col_size], int        //顯示 led[][]內字組函式
    continue_time) {
54      for (int k = 0; k < continue_time; k++){
55          for (int i = 0; i < col_size; i++){
56              clear_led();
57              low_74138pin(i);
58              for (int j = 0; j < row_size; j++)
59                if (led[j][i] == 1)
60                    digitalWrite(row_pin[j], HIGH);
61          }
62      }
```

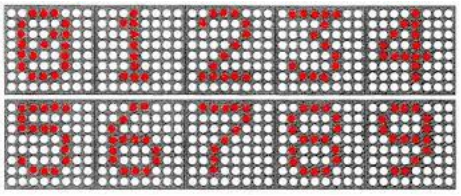
```
63    }                                                    //結束 display_led ()函式
64
65    void low_74138pin(int num){                          // 8x8 點矩陣 LED 掃描函式
66      switch (num){
67        case 0:                                          //掃描第 0 行(C0)
68          digitalWrite(Ls138_A, LOW);
69          digitalWrite(Ls138_B, LOW);
70          digitalWrite(Ls138_C, LOW);
71          break;
72        case 1:                                          //掃描第 1 行(C1)
73          digitalWrite(Ls138_A, HIGH);
74          digitalWrite(Ls138_B, LOW);
75          digitalWrite(Ls138_C, LOW);
76          break;
77        case 2:                                          //掃描第 2 行(C2)
78          digitalWrite(Ls138_A, LOW);
79          digitalWrite(Ls138_B, HIGH);
80          digitalWrite(Ls138_C, LOW);
81          break;
82        case 3:                                          //掃描第 3 行(C3)
83          digitalWrite(Ls138_A, HIGH);
84          digitalWrite(Ls138_B, HIGH);
85          digitalWrite(Ls138_C, LOW);
86          break;
87        case 4:                                          //掃描第 4 行(C4)
88          digitalWrite(Ls138_A, LOW);
89          digitalWrite(Ls138_B, LOW);
90          digitalWrite(Ls138_C, HIGH);
91          break;
92        case 5:                                          //掃描第 5 行(C5)
93          digitalWrite(Ls138_A, HIGH);
94          digitalWrite(Ls138_B, LOW);
95          digitalWrite(Ls138_C, HIGH);
96          break;
97        case 6:                                          //掃描第 6 行(C6)
98          digitalWrite(Ls138_A, LOW);
99          digitalWrite(Ls138_B, HIGH);
100         digitalWrite(Ls138_C, HIGH);
101         break;
102       case 7:                                          //掃描第 7 行(C7)
```

```
103             digitalWrite(Ls138_A, HIGH);
104             digitalWrite(Ls138_B, HIGH);
105             digitalWrite(Ls138_C, HIGH);
106             break;
107         }                                   //結束 switch
108     }                                       //結束 low_74138pin ()函式
```

練習：

一、 請寫出 74LS138 的真值表及接腳圖?

二、 請寫出 0 到 9 的資料碼，如下圖所示?



三、 撰寫一個程式顯示 0~9?

實驗 2-7：4 位指撥開關實驗

目的： 利用 LED 亮滅反應指撥開關（DIP switch）的狀態，來了解輸入及輸出合併使用方法及程式的撰寫技巧。

功能： 使用 Arduino MEGA2560 讀取指撥開關的值並以對應之 LED 顯示，ON：亮、OFF：滅。

原理： 由指撥開關的數量，可分為 2P、4P、5P、8P...等，2P 是指撥開關內部有獨立兩個開關，4P 是有 4 個獨立開關。圖 2-7-1 為 4P 的指撥開關實體圖，通常會在指撥開關上標示「ON」或記號，若將開關撥到「ON」端時，則接點為接通(on)，撥到另一端時，則接點為開路(off)。

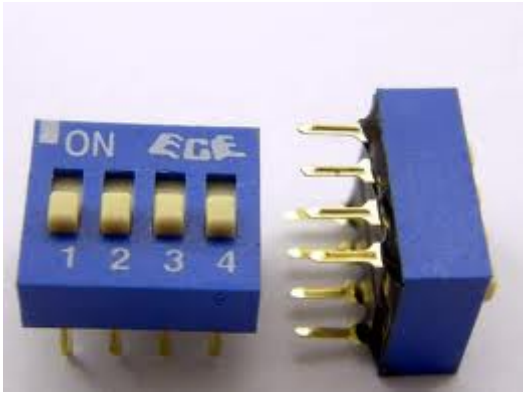


圖 2-7-1、為 4P 的指撥開關實體圖

電路： 4P 指撥開關與 Arduino MEGA2560 開發板的接線如表 2-7-1 所示，LED 經過 74AC244B 與 Arduino MEGA2560 開發板的接線如表 2-7-2 所示，電路如圖 2-7-2、圖 2-7-3 所示。

表 2-7-1　4P 指撥開關與 Arduino MEGA2560 接線

	指撥開關	MEGA2560
1	DIP_SW1	D37
2	DIP_SW2	D38
3	DIP_SW3	D39
4	DIP_SW4	D40

表 2-7-2　LED 經過 74AC244B 與 Arduino MEGA2560 接線

	LED	MEGA2560
1	D1	D2

2	D2	D3
3	D3	D4
4	D4	D5

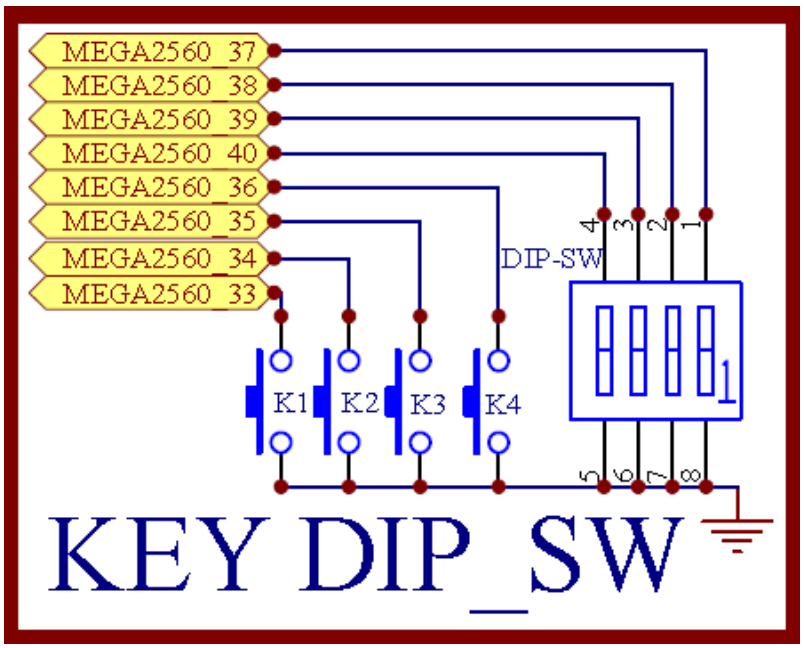


圖 2-7-2　4P 指撥開關電路

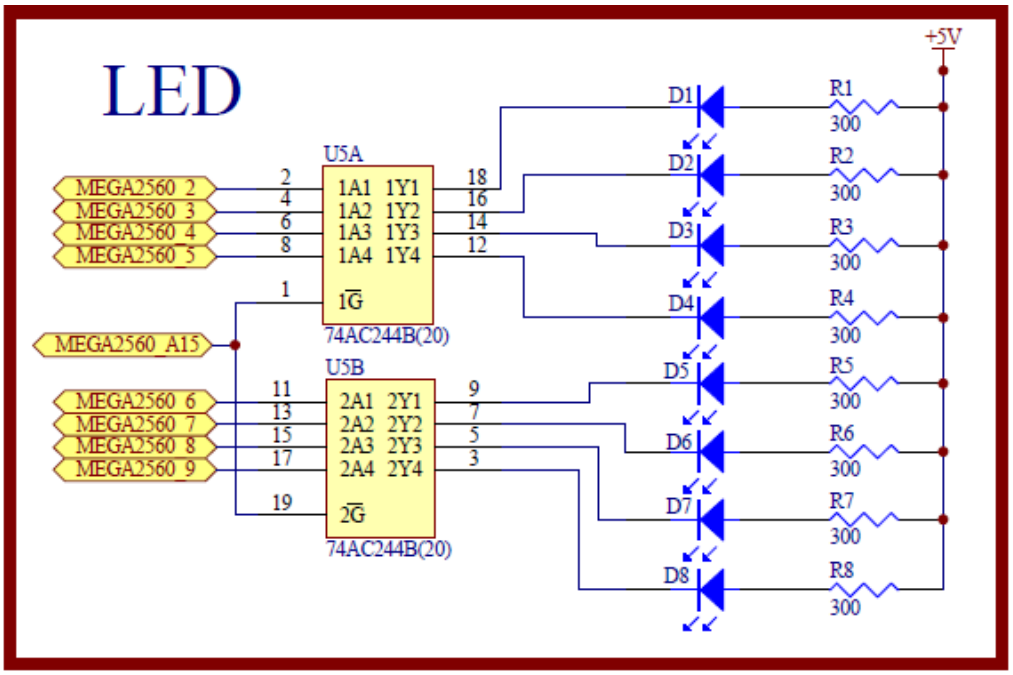


圖 2-7-3　LED 顯示電路

元件：

編號	元件項目	數量	元件名稱
1	MEGA2560	1	Arduino 開發板
2	DIP_SW	1	4P
3	D1~D4	4	LED
4	U5	1	74AC244B
5	R1~R4	4	300Ω

程式：

行號	程式敘述	註解
1	int BASE = 2;	//第一顆 LED 的 I/O 接腳
2	i nt NUM = 4;	//LED 及指撥開關的總數
3	int BASE1 = 37;	//第一個指撥開關的 I/O 接腳
4	int DIPSW1state;	// 定義指撥開關讀取變數
5	int DIPSW2state;	
6	int DIPSW3state;	
7	int DIPSW4state;	
8	void setup(){	// 只會執行一次的程式初始式數
9	for (int i = BASE; i < BASE + NUM; i ++) {	//設定 2~5 接腳為輸出
10	pinMode(i, OUTPUT);	
11	}	
12	for (int i = BASE1; i < BASE1 + NUM; i ++) {	//設定 37~40 接腳為輸入
13	pinMode(i, INPUT_PULLUP);	
14	}	
15	pinMode(A15, OUTPUT);	//A15 控制 74AC244
16	digitalWrite(A15, LOW);	//74AC244 三態緩衝器為 ON
17	}	// 結束 setup()函式
18	void loop(){	// 永遠周而復始的主控制函式
19	DIPSW1state = digitalRead(37);	//讀取第一個指撥開關的狀態
20	digitalWrite(2, DIPSW1state);	//將第一個指撥開關的狀態，輸出到對應的 LED
21	DIPSW2state = digitalRead(38); /	/讀取第二個指撥開關的狀態
22	digitalWrite(3, DIPSW2state);	//將第二個指撥開關的狀態，輸出到對應的 LED
23	DIPSW3state = digitalRead(39);	//讀取第三個指撥開關的狀態
24	digitalWrite(4, DIPSW3state);	//將第三個指撥開關的狀態，輸出到對應的 LED

25	DIPSW4state = digitalRead(40); /	/讀取第四個指撥開關的狀態
26	digitalWrite(5, DIPSW4state);	//將第四個指撥開關的狀態，輸出到對應的 LED
27	}	//結束 loop()

練習：

一、 試使用指撥開關 ON/OFF 狀態直接反應在對應的 LED 上，當指撥開關於 ON 時，對應之 LED 會作閃爍動作，反之當指撥開關於 OFF 時，對應之 LED 會滅掉。

二、 試使用指撥開關 ON/OFF 狀態，以十六進位反應在對應的 LED 上。

實驗 2-8：4 位按鈕開關實驗

目的： 利用 LED 亮滅反應按鈕開關（push button switch）的狀態，來了解輸入及輸出合併使用方法及程式的撰寫技巧。

功能： 使用 Arduino MEGA2560 讀取按鈕開關的值並以對應之 LED 顯示，ON：亮、OFF：滅。

原理： 本實驗使用的按鈕開關如圖 2-8-1 所示，未壓下開關時，其接點為開路(off) ，若將按鈕開關壓下時，開關的狀態為接通(on)。

圖 2-8-1、按鈕開關實體圖

電路： 按鈕開關與 Arduino MEGA2560 開發板的接線如表 2-8-1 所示，LED 經過 74AC244B 與 Arduino MEGA2560 開發板的接線如表 2-8-2 所示，電路如圖 2-8-2、圖 2-8-3 所示。

表 2-8-1　按鈕開關與 Arduino MEGA2560 接線

	按鈕開關	MEGA2560
1	K1	D33
2	K2	D34
3	K3	D35
4	K4	D36

表 2-8-2　LED 經過 74AC244B 與 Arduino MEGA2560 接線

	LED	MEGA2560
1	D5	D6
2	D6	D7
3	D7	D8
4	D8	D9

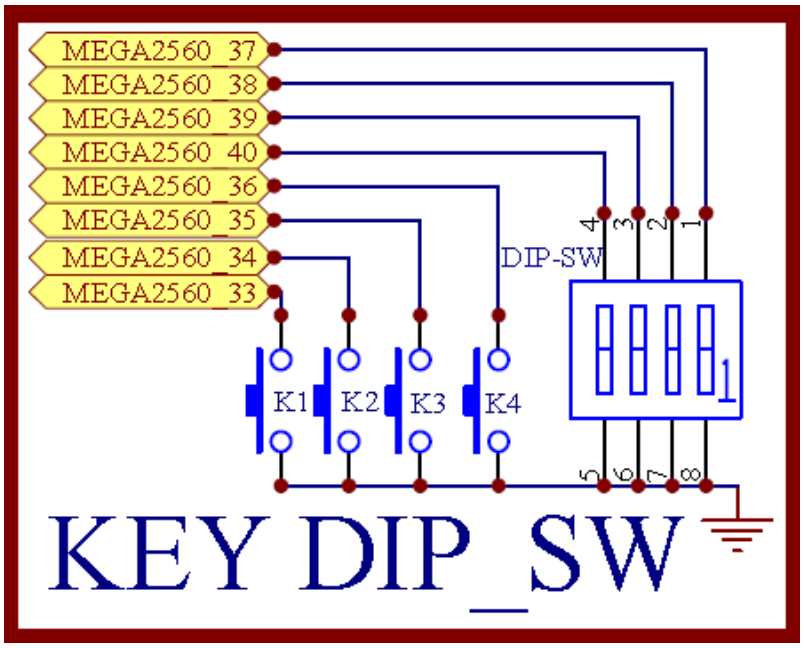


圖 2-7-2　按鈕開關電路

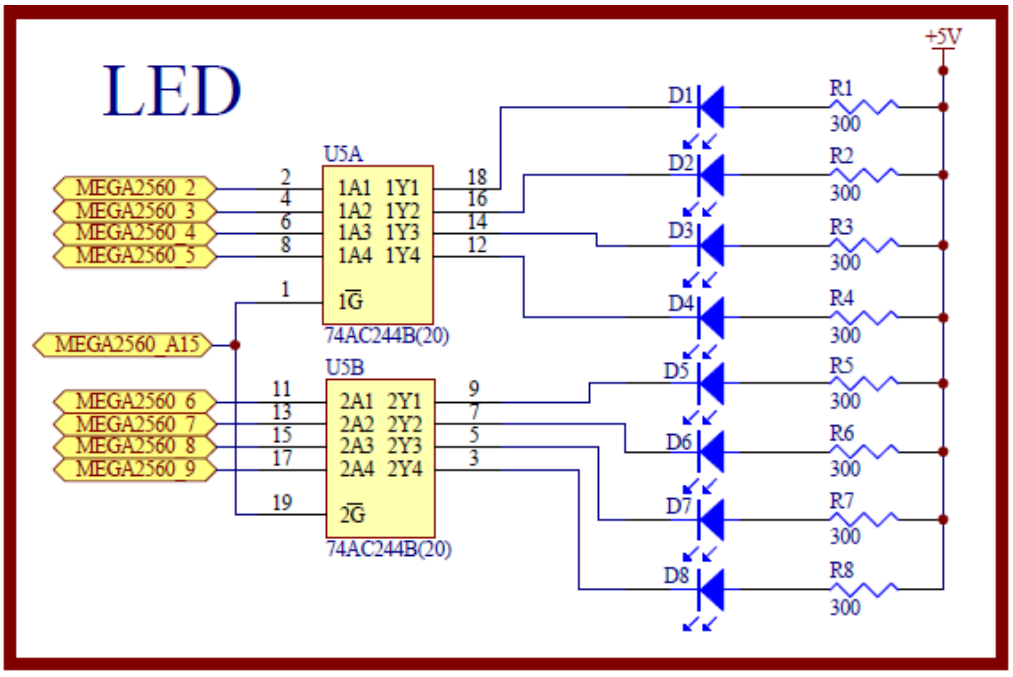


圖 2-7-3　LED 顯示電路

元件：

編號	元件項目	數量	元件名稱
1	MEGA2560	1	Arduino 開發板
2	K1~K2	4	按鈕開關
3	D5~D8	4	LED

編號	元件項目	數量	元件名稱
4	U5	1	74AC244B
5	R5~R8	4	300Ω

程式：

行號	程式敘述	註解
1	int BASE = 6;	//第一顆 LED 的 I/O 接腳
2	i nt NUM = 4;	//LED 及按鈕開關的總數
3	int BASE1 = 33;	//第一個按鈕開關的 I/O 接腳
4	int DIPSW1state;	// 定義按鈕開關讀取變數
5	int DIPSW2state;	
6	int DIPSW3state;	
7	int DIPSW4state;	
8	void setup(){	// 只會執行一次的程式初始式數
9	for (int i = BASE; i < BASE + NUM; i ++) {	//設定 6~9 接腳為輸出
10	pinMode(i, OUTPUT);	
11	}	
12	for (int i = BASE1; i < BASE1 + NUM; i ++) {	//設定 33~36 接腳為輸入
13	pinMode(i, INPUT_PULLUP);	
14	}	
15	pinMode(A15, OUTPUT);	
16	digitalWrite(A15, LOW);	
17	}	// 結束 setup()函式
18	void loop(){	// 永遠周而復始的主控制函式
19	K1state = digitalRead(33);	//讀取第一個按鈕開關的狀態
20	digitalWrite(9, K1state);	//將第一個按鈕開關的狀態，輸出到對應的 LED
21	K2state = digitalRead(34);	/讀取第二個按鈕開關的狀態
22	digitalWrite(8, K2state);	//將第二個按鈕開關的狀態，輸出到對應的 LED
23	K3state = digitalRead(35);	//讀取第三個按鈕開關的狀態
24	digitalWrite(7, K3state);	//將第三個按鈕開關的狀態，輸出到對應的 LED
25	K4state = digitalRead(36);	/讀取第四個按鈕開關的狀態
26	digitalWrite(6, K4state);	//將第四個按鈕開關的狀態，輸出到對應的 LED
27	}	//結束 loop()

練習：

一、 模擬汽機車之方向燈、煞車燈及緊急燈控制。

a. 試使用按鈕開關 K1 模擬左轉燈控制，D8 閃爍。

b. 試使用按鈕開關 K2 模擬右轉燈控制，D5 閃爍。

c. 試使用按鈕開關 K3 模擬煞車燈控制，D5~D8 全亮。

d. 試使用按鈕開關 K4 模擬緊急燈控制，D5~D8 閃爍。

實驗 2-9：4×4 鍵盤實習

目的： 瞭解 4×4 鍵盤的工作原理及使用 Arduino 程式控制方法。

功能： 本實習使用 Arduino MEGA2560 讀取 4×4 鍵盤的按鍵值並顯示序列埠觀測視窗。

原理： 4×4 矩陣鍵盤採用四行四列的組合，一般提供給微控制器作為輸入狀態的裝置。每個按鍵的下方一端連接到一個行，另一端連接到一個列，如圖 2-9-1 所示。一般市售 4×4 鍵盤及其對應接腳圖顯示在圖圖 2-9-2。

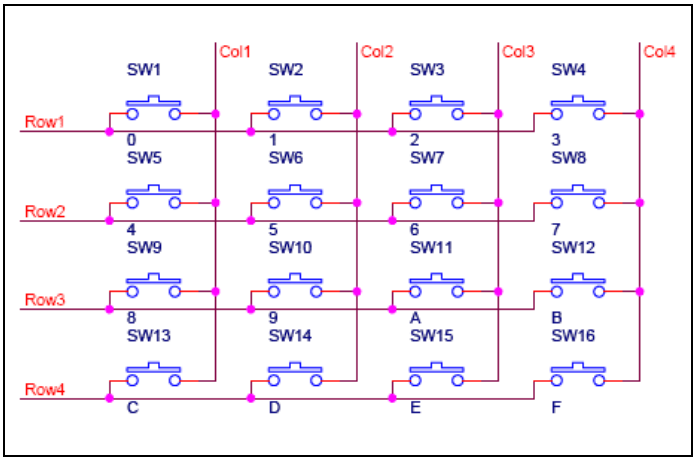


圖 2-9-1　4×4 鍵盤內部電路

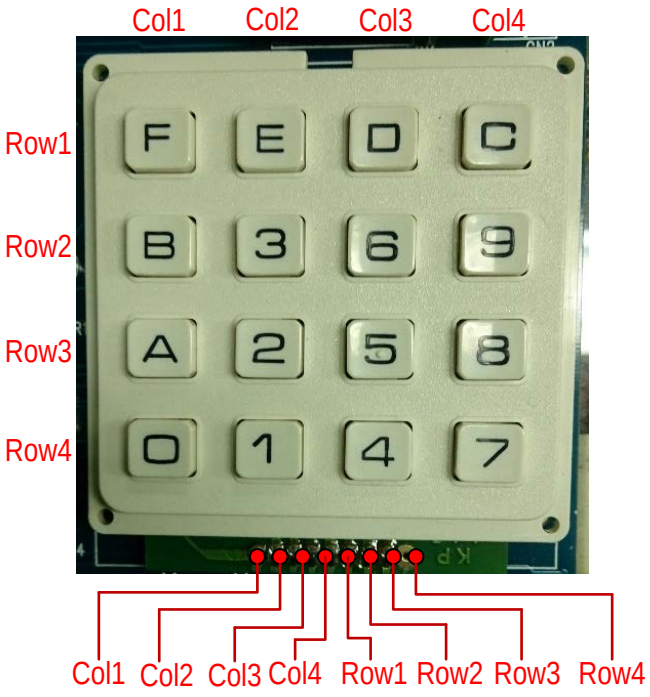


圖 2-9-2　市售 4×4 鍵盤接腳圖

要判斷按鍵有沒被按下，就對某一行(Column)的線路送出電壓，然後檢查每一列(Row)的線路看看是否有導通，假如有導過的話，就代表該行與該列交集的按鍵有被按下，接著換到下一行(Column)，然後依序檢查每一列的線路，照著這個步驟做，就可以對 Keypad 上所有按鍵的狀態整個檢查一遍。這個方法叫做鍵盤掃瞄(Keypad scan)。

Keypad 鍵盤掃瞄程序必須控制輸出訊號、判斷輸入訊號，還要處理按鍵彈跳(debounce)等問題，所以 Keypad 鍵盤掃瞄程序是有一點繁瑣的。本實習使用 Arduino 官方網站提供的 keypad 函式庫把鍵盤掃瞄程序變簡單了，只要安裝 Keypad 函式庫，就可以很輕鬆地讀取鍵盤的輸入，而且 Keypad 函式庫支援 3×4, 4×4 等各種型式的鍵盤。

電路： 4×4 鍵盤與 Arduino MEGA2560 開發板的接線如表 2-9-1 所示，整體電路如圖 2-9-3 所示。

表 2-9-1　4×4 鍵盤與 Arduino MEGA2560 接線

	4×4 鍵盤	MEGA2560
8	Col1	D29
7	Col2	D28
6	Col3	D27
5	Col4	D26
4	Row1	D25
3	Row2	D24
2	Row3	D23
1	Row4	D22

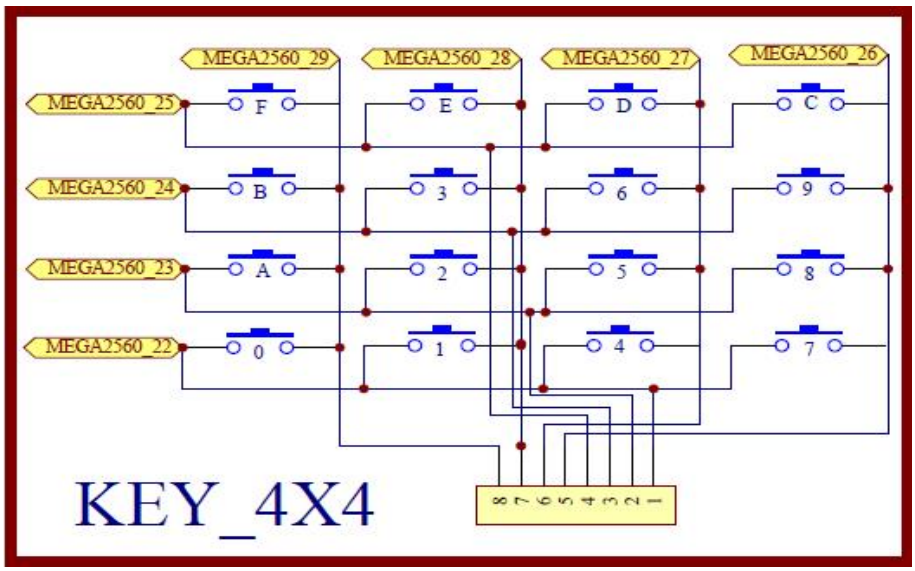


圖 2-9-3　4×4 鍵盤電路

元件：本實習所需元件如表 2-9-2 所示。

表 2-9-2 元件表

編號	元件項目	數量	元件名稱
1	MEGA2560	1	Arduino 開發板
2	4×4 Keypad	1	4×4 鍵盤

程式：此實習需從 Arduino 官方網站下載 keypad 函式庫，把檔案解壓縮放到 Arduino 的 Libraries 資料夾內。

P2-9-1

行號	程式敘述	註解
1	#include <Keypad.h>	// 加入 Keypad 函式庫
2	const byte ROWS = 4;	// 設定鍵盤為 4 列
3	const byte COLS = 4;	// 設定鍵盤為 4 行
4	char hexaKeys[ROWS][COLS]={ {'F','E','D','C'}, {'B','3','6','9'}, {'A','2','5','8'}, {'0','1','4','7'} };	// 定義 Keypad 的按鍵
5	byte rowPins[ROWS] ={25, 24, 23, 22};	// 定義 Arduino 的接腳接到 Keypad 的 Row1, Row2, Row3, Row4
6	byte colPins[COLS] ={29, 28, 27, 26};	// 定義 Arduino 的接腳接到 Keypad 的 Col1, Col2, Col3, Col4
7	Keypad customKeypad= Keypad(makeKeymap(hexaKeys),rowPins,colPins, ROWS, COLS);	// 建立 Keypad 物件
8	void setup(){	// 只會執行一次的程式初始式數
9	Serial.begin(9600);	// 將串列埠通訊鮑率設為 9600bps
10	}	// 結束 setup()函式
11	void loop(){	// 永遠周而復始的主控制函式
12	char customKey = customKeypad.getKey();	// 讀取鍵盤的輸入
13	if (customKey){	// 判別有按鍵按下時
14	Serial.println(customKey);	// 按鍵輸出至序列埠觀測視窗
15	}	// 結束 if
16	}	/結束 loop()函式/

練習：

利用 4×4 鍵盤與 4 個 LED 設計將按鍵的數值轉換成由 LED 顯示。

實驗 2-10：可變電阻之類比/數位轉換實驗

目的： 瞭解 MEGA 2560 的類比/數位轉換(ADC)的設計與控制方法

功能： 利用可變電阻的調整變化，將類比電壓轉換成相對應數位值，0V ~ 5V 轉換成相對應 0 ~ 1023。

原理： 電路如圖 2-10-1 所示，MEGA 2560 的 A0 接到可變電阻的中間接點，A0 接受 0~5V 間的電壓輸入，利用 MEGA 2560 的 A/D 轉換器，轉換成相對應 0 ~ 1023 間之值。程式 1：是利用 Arduino 的序列埠監控視窗，觀察 A/D 轉換的狀況。程式 2：是利用 A/D 轉換後的數位值，作為 LED 的亮滅的時間。

電路： 可變電阻與 Arduino MEGA2560 開發板的接線如表 2-10-1、圖 2-10-1 所示。

表 2-10-1　可變電阻與 Arduino MEGA2560 接線

	可變電阻	MEGA2560
1	VR1	A0

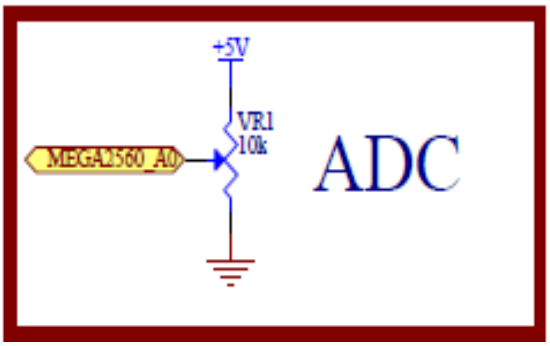


圖 2-10-1　可變電阻類比轉數位電路

元件：

編號	元件項目	數量	元件名稱
1	MEGA2560	1	Arduino 開發板
2	VR1	1	10 k
3	LED	1	LED

程式：1

行號	程式敘述	註解
1	int VR_Pin = A0;	// 選擇可變電阻輸入腳位 A0
2	int VR_Value = 0;	// 定義可變電阻轉換成數位值之變數
3		
4	void setup() {	// setup()函式開始
5	Serial.begin(9600);	// 定義序列埠監控視窗傳遞速率為 9600
6	}	// setup()函式結束
7		
8	void loop() {	// loop()函式開始
9	VR_Value = analogRead(VR_Pin);	// 讀取可變電阻的數位值
10	Serial.print("VR_Value = ");	//傳送到序列埠監控視窗
11	Serial.println(VR_Value);	//傳送到序列埠監控視窗
12	delay(10);	// loop ()函式結束
	}	

程式：2

行號	程式敘述	註解
1	int VR_Pin = A0;	// 選擇可變電阻輸入腳位 A0
2	int VR_Value = 0;	// 定義可變電阻轉換成數位值之變數
3	int ledPin = 13;	// 選擇 LED 輸出腳位 D13
4	void setup() {	// setup()函式開始
5	pinMode(ledPin, OUTPUT);	// 定義 ledPin 為輸出
6	}	
7		
8	void loop() {	// loop()函式開始
9	VR_Value = analogRead(VR_Pin);	// 讀取可變電阻的數位值
10	digitalWrite(ledPin, HIGH);	//LED 亮
11	delay(VR_Value);	//延時 VR_Value 毫秒
12	digitalWrite(ledPin, LOW);	//LED 滅
	delay(VR_Value);	//延時 VR_Value 毫秒
	}	// setup()函式結束

練習：

一、 請利用可變電阻控制 LED 燈光亮度，達到調光的目的。

實驗 2-11：雙軸搖桿模組實驗

目的： 瞭解雙軸搖桿模組的結構與控制方法

功能： 利用雙軸搖桿控制兩個可變電阻作 x 軸和 y 軸的調整變化。

原理： 雙軸搖桿模組的實體如圖 2-11-1 所示，內含一個按鈕開關、搖桿及兩個雙向的 10K 可變電阻器，可變電阻器的中間抽頭電阻值會隨著搖桿移動而隨著變化。本模組使用 5V 供電，搖桿在零點(0,0)狀態下 x,y 電壓分別為 2.5V 左右，當隨搖桿移動，電壓值隨著變化，最大到 5V；最小為 0V。雙軸搖桿模組提供兩個實驗，程式 1：是利用 Arduino 的序列埠監控視窗，觀察搖桿移動及相對應的數位值的狀況。程式 2：是結合 LCD 顯示轉換後的數位值。

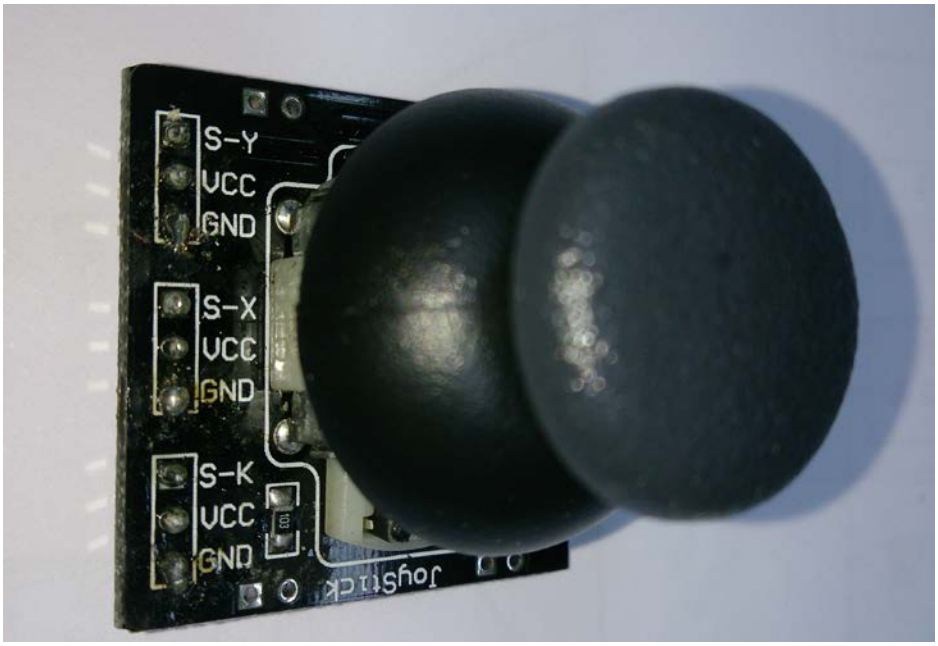


圖 2-11-1　雙軸搖桿模組的實體

電路： 雙軸搖桿模組與 Arduino MEGA2560 開發板的接線如表 2-11-2 所示。

表 2-11-2　雙軸搖桿模組與 Arduino MEGA2560 接線

	雙軸搖桿模組	MEGA2560
1	x 軸	A1
2	y 軸	A2
3	按鈕開關	A3

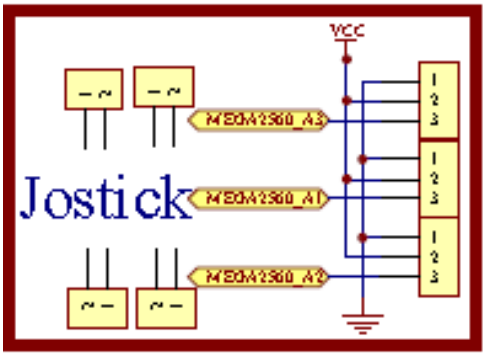


圖 2-11-2　雙軸搖桿模電路

元件：

編號	元件項目	數量	元件名稱
1	MEGA2560	1	Arduino 開發板
2	雙軸搖桿模組	1	雙軸搖桿模組

程式：1

行號	程式敘述	註解
1	int joyPinX = A1;	//定義 x 軸接腳
2	int joyPinY = A2;	//定義 y 軸接腳
3	int SW = A3;	//定義按鈕開關接腳
4	int value = 0;	//
5	int xzero = 0;	
6	int yzero = 0;	
7	int SWstate = 0;	
8	void setup() {	//setup()函式開始
9	Serial.begin(9600);	// 定義序列埠監控視窗傳遞速率為 9600
10	pinMode(SW, INPUT_PULLUP);	//設定 SW 為提升輸入模式
11	yzero = analogRead(joyPinY);	// 讀取(0,0)之 y 軸值
12	xzero = analogRead(joyPinX);	// 讀取(0,0)之 x 軸值
13	}	// setup()函式結束
14		
15	void loop() {	// loop()函式開始
16	int x,y,value;	//定義變數
17	value = analogRead(joyPinX);	// x 軸值讀取
18	x=value-xzero;	//減去(0,0)之 x 值，得到 x 軸之值

```
19        value = analogRead(joyPinY);                    // y 軸值讀取
20         y=value-yzero;                                 //減去(0,0)之 y 值，得到 y 軸之值
21        SWstate = digitalRead(SW);                      //  按鍵值讀取
22         Serial.print("X = ");                          //傳送"X = "到序列埠監控視窗
23         Serial.print(x);                               //傳送 x 值到序列埠監控視窗
24         Serial.print("    Y = ");                      //傳送"Y = "到序列埠監控視窗
25         Serial.println(y);                             //傳送 y 值到序列埠監控視窗
26         Serial.print("SW = ");                        //傳送"SW = "到序列埠監控視窗
27         Serial.println(SWstate);                       //傳送按鈕開關到序列埠監控視窗
28         delay(20);                                     //延時 20 毫秒
29    }                                                   // loop ()函式結束
```

程式：2

行號	程式敘述	註解
1	#include <LiquidCrystal.h>	
2	int joyPinX = A1;	//定義 x 軸接腳
3	int joyPinY = A2;	//定義 y 軸接腳
4	int SW = A3;	//定義按鈕開關接腳
5	int value = 0;	
6	int xzero = 0;	
7	int yzero = 0;	
8	int SWstate = 0;	
9	LiquidCrystal lcd(49, 48, 47, 43, 42, 41);	//定義 LCD 與 MEGA2560 接腳
10		
11	void setup() {	// setup()函式開始
12	pinMode(SW, INPUT_PULLUP);	//設定 SW 為提升輸入模式
13	yzero = analogRead(joyPinY);	// 讀取(0,0)之 y 軸值
14	xzero = analogRead(joyPinX);	// 讀取(0,0)之 x 軸值
15	lcd.begin(16, 2);	// 設定 LCD 為 2 列 16 格
16	}	// setup()函式結束
17		
18	void loop() {	// loop()函式開始
19	int x,y,value;	//宣告變數
20	lcd.setCursor(0, 0);	//設定 LCD 顯示位址第 0 列第 0 格
21	lcd.print("X= Y= ");	//顯示"X= Y= "
22	lcd.setCursor(0, 1);	//設定 LCD 顯示位址第 1 列第 0 格
23	lcd.print("SW= ");	//顯示"SW= "

```
24      value = analogRead(joyPinX);                    // x 軸值讀取
25      x=value-xzero;                                  //減去(0,0)之 x 值，得到 x 軸之值
26      value = analogRead(joyPinY);                    // y 軸值讀取
27      y=value-yzero;                                  //減去(0,0)之 y 值，得到 y 軸之值
28      SWstate = digitalRead(SW);                      //按鈕開關狀態讀取
29      lcd.setCursor(2, 0);                            //設定 LCD 顯示位址第 0 列第 2 格
30       lcd.print(x);                                  //顯示 x 值
31      lcd.setCursor(10, 0);                           //設定 LCD 顯示位址第 0 列第 10
                                                            格
32      lcd.print(y);                                   //顯示 y 值
33      lcd.setCursor(3, 1);                            //設定 LCD 顯示位址第 1 列第 3 格
34      lcd.print(SWstate);                             //顯示按鈕開關值
35      delay(30);                                      //延時 30  毫秒
36    }                                                 // setup()函式結束
```

練習：

一、 請利用雙軸搖桿模組控制兩個 LED(水平，垂直)燈光亮度，和一個 LED ON/OFF。

實驗 2-12：LCD 液晶模組電路功能實習

目的：瞭解 2X16 型 LCD 顯示模組的工作原理及使用 Arduino 程式控制方法。

功能：本實習使用 Arduino MEGA2560 控制 2X16 文數字型 LCD 顯示模組做為 Arduino 開發板的輸出顯示介面。程式(一)中在 LCD 的第一行顯示“How are you?”，第二行顯示“LCD Testing!”，並讓螢幕間隔 0.5 秒做顯示與不顯示功能展示。程式(二)中讀取 4×4 鍵盤的按鍵值並由 LCD 顯示按鍵值。

原理：LCD（Liquid Crystal Display）內部有字元產生器，可以從 Arduino 接收欲顯示的字元碼（ASCII CODE），並將其存入到顯示資料 RAM 中，而由 LCD 的控制器來控制及顯示。圖 2-12-1 為 LCD 顯示模組實體圖，其介面信號一般共有十四條信號線，每一個信號線的意義及用途請參閱表 2-12-1。程式中使用 Arduino IDE 內建 LCD 函式庫來控制 LCD 上游標顯示的位置並顯示字串文字與數值。本實習使用 Arduino IDE 內建<LiquidCrystal.h>函式庫把 LCD 顯示控制變簡單了，此函式庫可定義 Byte 或 Nibble(4-bits)存取模式，如果以 lcd(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)來建構 LCD 物件則 Arduino 是使用 Byte 存取模式，亦即需使用到 10 支 IO 腳位來輸出指令或資料，這會佔用較多 IO 腳位。如果改以 lcd(rs, enable, d4, d5, d6, d7)來建構 LCD 物件則 Arduino 是使用 Nibble 存取模式，亦即只需使用到 6 支 IO 腳位，Arduino Uno 會將指令與資料分成 2 個 Nibble 來輸出，這可以省下 4 支 IO 腳位轉用於其他 IO 控制之用。

圖 2-12-1　LCD 顯示模組實體圖

表 2-12-1　LCD 1602 顯示模組 16 支腳位功能說明

腳位編號	腳位名稱	輸入／輸出	腳位說明
1	V_{SS}	I	接地(0V)
2	V_{DD}	I	電源(5V)
3	V_O	I	調整顯示器的明暗度對比(0-5V),可接一顆 1k 電阻，或利用可變電阻調整適當的螢幕亮度對比
4	RS	I	LCD 內部暫存器的選擇線， 當 RS=1：D0 – D7 當作資料解釋 當 RS=0：D0 – D7 當作指令解釋

腳位編號	腳位名稱	輸入／輸出	腳位說明
5	$R/\overline{W}$	I	讀／寫信號： 1：從 LCD 讀取資料 0：寫資料到 LCD 因為很少從 LCD 這端讀取資料，所以可將此腳位接地以節省 I/O 腳位。
6	E	I	LCD 致能信號
7	DB0	I/O	此四位元使用在 8 位元資料傳輸
8	DB1	I/O	
9	DB2	I/O	
10	DB3	I/O	
11	DB4	I/O	此四位元被使用在 4 位元或 8 位元資料傳輸 在讀取旗號時，此位元 7 亦可以當 BF 旗號
12	DB5	I/O	
13	DB6	I/O	
14	DB7	I/O	

電路： LCD 與 Arduino MEGA2560 開發板的接線如圖 2-12-2 所示，使用 6 支 IO 腳位接到 LCD 顯示模組的接腳。圖中 Arduino MEGA2560 開發板上的第(49, 48, 47, 43, 42, 41)支數位接腳對應連接到 LCD 顯示模組的第(4, 6, 11, 12, 13, 14)支接腳，Arduino 透過 lcd(rs, enable, d4, d5, d6, d7)函式來定義兩者間的對應關係。LCD 顯示模組的第 1 支接腳和第 5 支$R/\overline{W}$接腳一起接電源地端，以設定 LCD 顯示器只有寫入功能。電源(+5V)接到 LCD 顯示模組的第 2 支接腳，第 3 支接腳則經一顆 10k 可變電阻接到地端，藉由改變可變電阻值得到可變的分壓以得到適當的螢幕亮度對比。程式(一)中會在 LCD 顯示字串。程式(二)中將讀取 4x4 鍵盤的按鍵值並由 LCD 顯示按鍵值，4x4 鍵盤電路如圖 2-12-3 所示。

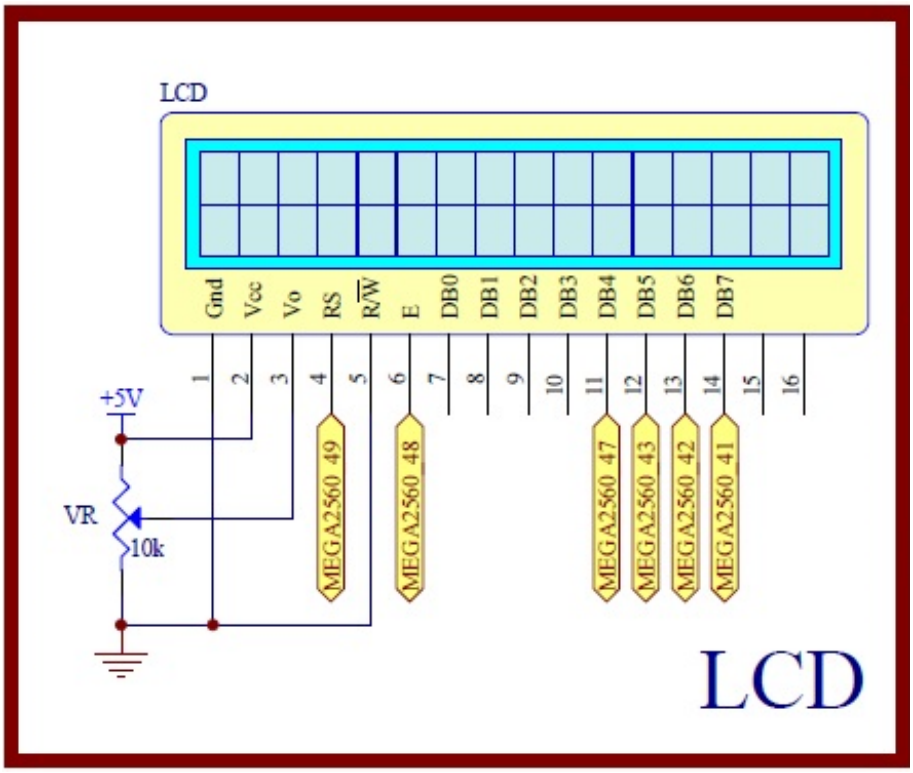


圖 2-12-2 Arduino MEGA2560 對應 LCD 顯示模組的實體接腳電路圖

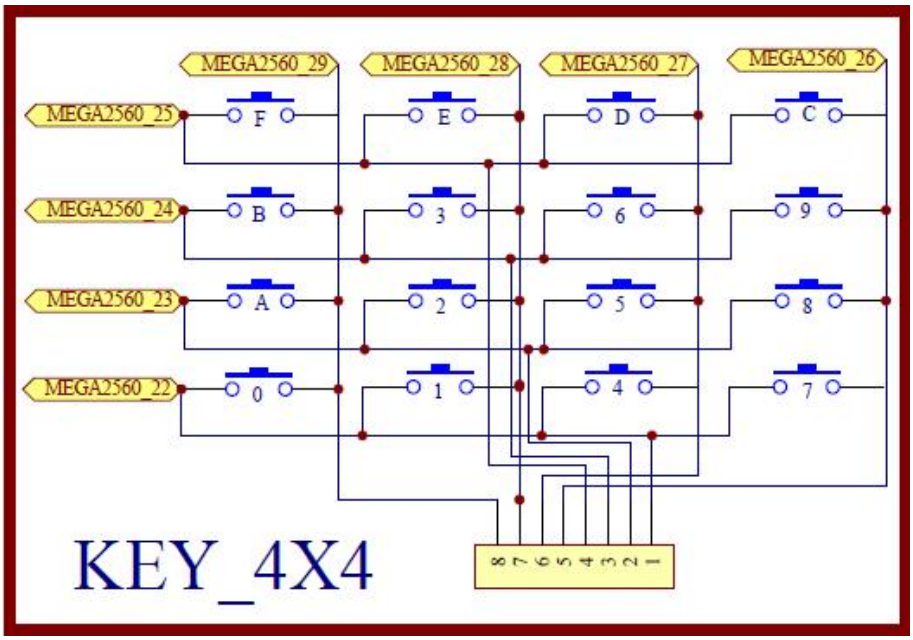


圖 2-12-3 4x4 鍵盤電路

元件：本實習所需元件如表 2-12-2 所示。

表 2-12-2 元件表

編號	元件項目	數量	元件名稱
1	Arduino MEGA2560	1	Arduino 開發板

編號	元件項目	數量	元件名稱
2	LCD	1	16×2 文字型 LCD 顯示器
3	VR	1	10kΩ 可變電阻
4	4×4 Keypad	1	4×4 鍵盤

程式（一）：LCD 字串顯示

P2-12-1

行號	程式敘述	註解
1	#include <LiquidCrystal.h>	//加入 LCD 顯示模組的驅動函式庫
2	LiquidCrystal lcd(49, 48, 47, 43, 42, 41);	// 定義 LCD 物件對應 Arduino 的腳位 // LiquidCrystal(rs, enable, d4, d5, d6, d7)
3	void setup(){	// 只會執行一次的程式初始函式
4	lcd.begin(16,2);	// 初始化 LCD 物件的格式為 16 字、2 行
5	lcd.clear();	// 清除 LCD 螢幕
6	lcd.setCursor(0,0);	// 游標設到 LCD 第 1 字、第 1 行
7	lcd.print("How are you?");	// LCD 顯示出字串" How are you ?"
8	lcd.setCursor(0,1);	// 游標設到 LCD 第 1 字、第 2 行
9	lcd.print("LCD Testing!");	// LCD 顯示出字串"LCD Testing!"
10	}	// 結束 setup()函式
11	void loop(){	// 永遠周而復始的主控制函式
12	lcd.noDisplay();	// 關閉螢幕
13	delay(500);	// 延遲 500msec
14	lcd.display();	// 打開螢幕
15	delay(500);	// 延遲 500msec
16	}	// 結束 loop()函式

程式（二）： LCD 顯示 4x4 鍵盤按鍵值

P2-12-2

行號	程式敘述	註解
1	#include <LiquidCrystal.h>	//加入 LCD 顯示模組的驅動函式庫
2	#include <Keypad.h>	// 加入 Keypad 函式庫
3	const byte ROWS = 4;	// 設定鍵盤為 4 列
4	const byte COLS = 4;	// 設定鍵盤為 4 行
5	char hexaKeys[ROWS][COLS]={ {'F','E','D','C'}, {'B','3','6','9'}, {'A','2','5','8'}, {'0','1','4','7'} };	// 定義 Keypad 的按鍵

```
6   byte rowPins[ROWS] ={25, 24, 23, 22};                    // 定義 Arduino 的接腳接到 Keypad 的 Row1, Row2, Row3, Row4
7   byte colPins[COLS] ={29, 28, 27, 26};                    // 定義 Arduino 的接腳接到 Keypad 的 Col1, Col2, Col3, Col4
8   LiquidCrystal lcd(49, 48, 47, 43, 42, 41);               // 定義 LCD 物件對應 Arduino 的腳位
                                                             // LiquidCrystal(rs, enable, d4, d5, d6, d7)
9   Keypad customKeypad=                                     // 建立 Keypad 物件
        Keypad(makeKeymap(hexaKeys),rowPins,colPins,
                                ROWS, COLS );
10  void setup(){                                            // 只會執行一次的程式初始函式
11      lcd.begin(16,2);                                     // 初始化 LCD 物件的格式為 16 字、2 行
12      lcd.print("KEY IN:    ");                            // LCD 顯示出字串" KEY IN:"
13  }                                                        // 結束 setup()函式
14  void loop(){                                             // 永遠周而復始的主控制函式
15    lcd.setCursor(7, 0);                                   // 游標設到 LCD 第 8 字、第 1 行
16    char customKey = customKeypad.getKey();                // 讀取鍵盤的輸入
17     if (customKey){                                       // 判別有按鍵按下時
18         lcd.print(customKey);                             // 按鍵輸出至 LCM 顯示
19       }                                                   // 結束 if
18  }                                                        // 結束 loop()函式
```

練習

三、 利用 4×4 鍵盤設計個位數的計算機，並將結果由 LCM 顯示。

實驗 2-13：藍牙 BT 無線通訊功能實習

目的：瞭解 BC04-B 藍牙模組的工作原理及使用 Arduino 程式控制如何通訊。

功能：本實習使用 Arduino MEGA2560 控制 HC-02 藍牙模組做為無線通訊介面。程式(一)中會利用 Android 手機或平板 App 透過藍牙遙控主板 LED 作亮滅動作。程式(二)中讀取可變電阻分壓類比值然後透過藍牙傳給 Android 手機或平板 App 做顯示。

原理：藍牙是由 Ericsson、NOKIA、IBM、Toshiba 及 Intel 等廠商共同所定義及發起的無線傳輸技術標準，是目前在消費電子通訊的應用上最普及的無線通訊的標準。藍牙技術與紅外線無線傳輸技術(IrDA)相似，皆為短距離的無線傳輸。但是紅外線無線傳輸裝置在進行資料傳輸時需將兩傳輸裝置對準，而藍牙為「點」傳輸技術，在進行傳輸時，資料從發射點以球狀向四面八方進行傳輸，故在應用性及方便性上而言，藍牙技術是比紅外線無線傳輸方便多了。

藍牙使用跳頻展頻(Frequency Hopping Spread Spectrum，FHSS)技術運作於 2402～2480MHz 頻段上，並採特定跳頻方式同步地在 79 個頻寬為 1MHz 的頻道上傳送訊號。藍牙的理論資料速率最低為 1 Mbps，最高可達 24 Mbps。Class 1、2、3 為藍牙發射功率的標準化等級，功率越小，所能通訊的距離越近。Class 1 所發射出來的微波功率約為 10mW，通訊距離約為 60~100 公尺；至於 Class 2 發射出來的功率不能超過 1mW，通訊距離為 10~20 公尺；目前也有 Class 3 的等級，通訊距離約為 3~5 公尺，不過因為距離過近，所以 Class 3 沒有成為理想的省電傳輸裝置。發射功率越小，耗電量越低，所以目前的隨身型裝置(智慧型手機、平板電腦、大部分筆記型電腦)若有配備藍牙者，都是只配備 Class 2 等級的通訊器。

常見的支援 SPP（Serial Port Profile，序列埠規範）的藍牙模組有：

HC-05：主／從（master/slave）一體型藍牙模組，主從可指令切換，指令豐富齊全，供電電壓 3.3V~3.6V。出廠預設通常是「從端」模式，但是能自行透過 AT 命令修改。

HC-06：主機或從機模式，出廠前就設定好，不能更改；指令少於 HC-05，使用簡單，供電電壓 3.3V~3.6V。市面上販售的通常是「從端」模式。

HC-02：只能用來做從機使用，供電電壓 2.0V~3.6V（支持 AT 指令修改 MAC 地址，支持 AT 指令恢復默認參數）。

通常主機是用來搜索從機設備，不能被其他設備搜索；從機是用來被搜的設備，不能主動搜索其他設備。當主/從設備連上以後就相當於使用一組串列線，這個時候就不分主從，也就是串列傳輸模式。

本實驗使用如圖 2-13-1 所示 HC-02 藍牙模組是專為智慧無線資料傳輸而打造，採用英國 CSR 公司 BlueCore4-Ext 晶片，遵循 V2.1+EDR 藍牙規範，屬於 Class 2 等級的通訊器，主要用於短距離的資料無線傳輸領域。本模組支援 UART, USB, SPI, PCM, SPDIF 等介面，並支援 SPP 藍牙串列埠協定，具有成本低、體積小、功耗低、收發靈敏性高等優點，只需配備少許的週邊元件就能實現其強大功能。基本上要使用藍牙模組作為資料的無線傳輸介面，並不需要懂得太多藍牙技術，只要遵循模組的接腳定義與設定程序，並接到 MEGA2560 板上，即可像串列通訊般的使用它。

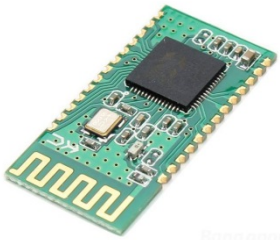


圖 2-13-1　HC-02 藍牙模組實體圖

電路： HC-02 藍牙模組與 Arduino MEGA2560 開發板的的實體接圖如圖 2-13-2 所示，圖 2-13-3 為實體接腳電路圖，圖中 Arduino MEGA2560 開發板上的第 19 支與 18 支數位接腳對應連接到 HC-02 藍牙模組 Tx 與 Rx 接腳。Arduino MEGA2560 開發板在串列傳輸上已有專用接腳供使用，其中軟體的函式 Serial 使用接腳 0 (Rx)及 1 (Tx)，函式 Serial 1 使用接腳 19 (Rx)及 18 (Tx)，函式 Serial 2 使用接腳 17(Rx)及 16(Tx)，函式 Serial 3 使用接腳 15(Rx)及 14(Tx). 這些接腳可用於接收（Rx）和發送（Tx）TTL 串列資料。0 和 1 號接腳也連接到 ATmega2560 USB-to-TTL 串列轉換晶片的對應接腳上。程式(一)中會利用 Android 手機或平板 App 透過藍牙遙控主板 LED D1(圖 2-12-4 LED 電路)作亮滅動作，首先開啟手機或平板的藍牙裝置，找到 HC-02 藍牙裝置後再與本開發板做藍牙配對，配對密碼為[1234]。接著安裝 BT_Remote_LED.apk (顯示畫面如圖 2-13-6 所示)到 Android 手機或平板，然後打開 App 即可進行本實習功能。程式(二)中 Arduino 將讀取可變電阻分壓類比值(圖 2-13-5 所示)，轉換成 0~1024 區間值，然後透過藍牙傳給 Android 手機或平板 App 做顯示，所以本實習

需先在 Android 手機或平板安裝 BT_Receive_Data.apk 後(顯示畫面如圖 2-13-7 所示)才可進行本實習功能。

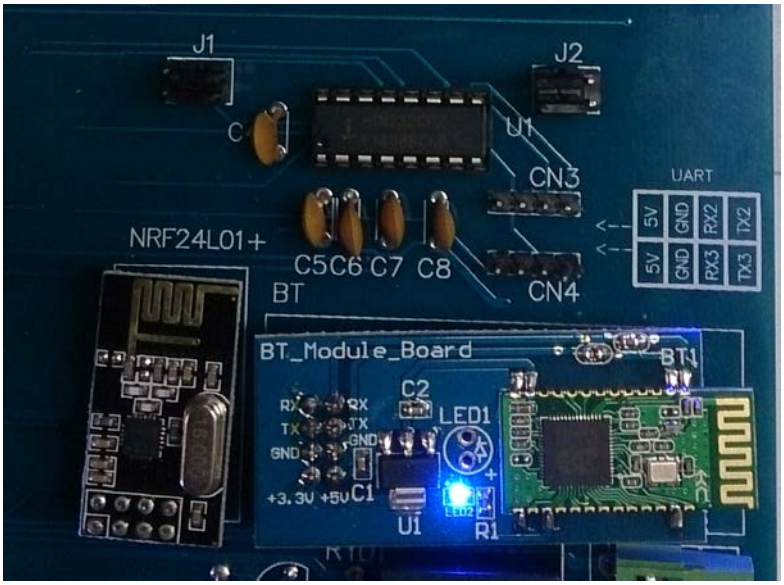


圖 2-13-2 Arduino MEGA2560 與 HC-02 藍牙模組的實體接圖

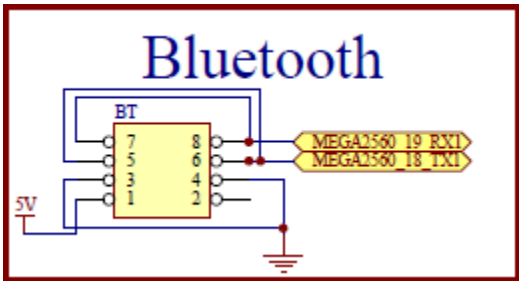


圖 2-13-3 Arduino MEGA2560 對應 HC-02 藍牙模組的實體接腳電路圖

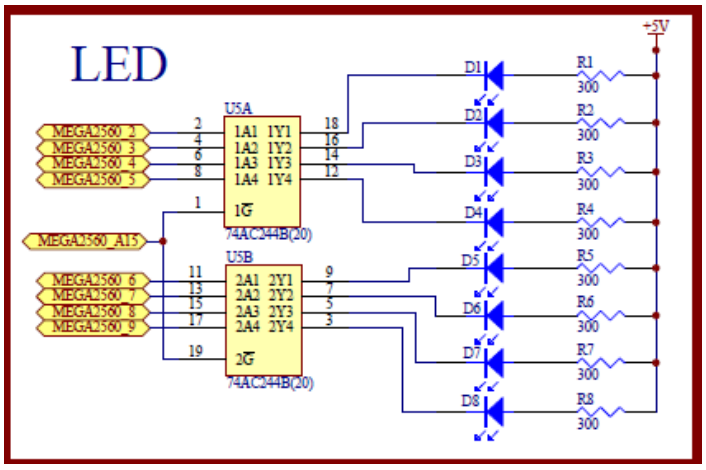


圖 2-13-4 Arduino MEGA2560 對應 LED 的實體接腳電路圖

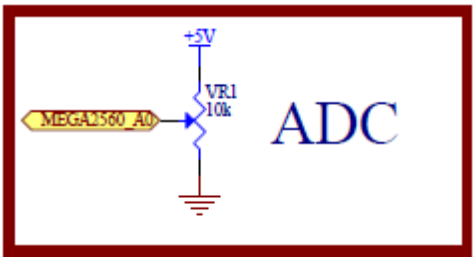


圖 2-13-5 Arduino MEGA2560 對應可變電阻類比轉數位的實體接腳電路圖

元件：本實習所需元件如表 2-13-1 所示。

表 2-13-1 元件表

編號	元件項目	數量	元件名稱
1	Arduino MEGA2560	1	Arduino 開發板
2	HC-02	1	藍牙模組
3	74AC244	1	8 個緩衝閘晶片
4	D1	1	單色 LED
5	R1	1	300 Ω電阻
6	VR1	1	10KΩ可變電阻

圖 2-13-6 BT_Remote_LED.apk

圖 2-13-7 BT_Receive_Data.apk

程式（一）：藍牙遙控 LED 亮滅

P2-13-1

行號	程式敘述	註解
1	#define LED 2	定義 LED 接腳
2	void setup() {	只會執行一次的程式初始設定函式
3	Serial.begin(9600);	定義序列埠監控視窗傳遞速率為 9600
4	Serial1.begin(9600);	定義藍牙傳遞速率為 9600
5	pinMode(A15,OUTPUT);	規劃 A15 腳為輸出模式
6	digitalWrite(A15, LOW);	A15 輸出 LOW，致能 74AC244
7	pinMode(LED, OUTPUT);	規劃 LED 腳為輸出模式
8	}	setup()函式結束
9	void loop() {	loop()函式開始
10	char cmmd[20];	宣告字元變數
11	int charSize;	宣告整數變數
12	if ((charSize=(Serial1.available()))>0){	從藍牙端讀取是否有資料
13	Serial.print("input size = ");	若有資料則輸出幾個字元數到觀測視窗
14	Serial.println(charSize);	
15	for (int i=0; i<charSize; i++){	從藍牙端讀取字元存放到 cmmd 陣列
16	Serial.print(cmmd[i]= char(Serial1.read()));	並輸出到觀測視窗
17	Serial.print("\n");	輸出換行
18	}	for loop 結束
19	}	if 結束
20	switch (cmmd[0]) {	判別讀取第一個字元
21	case 'a':	如果是'a'
22	digitalWrite(LED,LOW);	LED ON
23	break;	離開 switch 結構
24	case 'b':	如果是'b'
25	digitalWrite(LED,HIGH);	LED OFF
26	break;	離開 switch 結構
27	}	switch 結束
28	}	loop()函式結束

程式（二）：藍牙接收可變電阻量測值

P2-13-2

行號	程式敘述	註解
1	#define VR_Pin A0	定義可變電阻輸入腳位 A0
2	void setup(){	只會執行一次的程式初始設定函式
3	Serial.begin(9600);	定義序列埠監控視窗傳遞速率為 9600
4	Serial1.begin(9600);	定義藍牙傳遞速率為 9600
5	}	setup()函式結束
6	void loop (){	loop()函式開始
7	char Rece_char;	宣告字元變數
8	byte Data[3];	宣告位元組變數
9	int VR_Value;	宣告整數變數
10	VR_Value=analogRead(VR_Pin);	讀取可變電阻的類比值
11	Rece_char=Serial1.read();	從藍牙端讀取字元存放到 Rece_char
12	Data[0]='R';	存放'R'作為 App 端辨識接收數值的開始
13	Data[1]=VR_Value/256;	將高位元組取出置入 Data[1]
14	Data[2]=VR_Value%256;	將低位元組取出置入 Data[2]
15	Serial.println(VR_Value);	讀取值輸出到觀測視窗
16	if (Rece_char == 'T'){	判別藍牙端讀取字元為'T'表示 App 已準備好接收資料
17	for(int j=0;j<3;j++)	透過藍牙傳送資料到 App
18	Serial1.write(Data[j]);	
19	Rece_char=' ';	將 Rece_char 清除為空字元,準備下一次的傳送
20	}	if 結束
21	delay(100);	延遲 100ms
22	}	loop()函式結束

練習

二、 設計一系統可以利用手機或平板藍牙遙控 LED 亮度的明暗變化。

實驗 2-14: RF 模組(2.4G RF 模組)

目的：使用 Arduino MEGA2560 開發板上的 2.4G RF 模組做收發無線電訊號，並使用序列埠監控視窗觀看無線傳輸資料。

功能：使用 Arduino MEGA2560 的 2.4G RF 模組做收發無線電訊號的串列碼，分別由發射與接收電腦上的序列埠監控視窗觀看傳輸的串列碼，並藉由觀看發射與接收電腦上的序列埠監控視窗來比對無線傳輸資料。

原理：2.4G RF 模組由 Nordic Semiconductor 開發的 nRF24L01 無線收發晶片所設計完成，可發射與接收 2.4G Hz 的無線訊號，圖 2-14-1 為 2.4G RF 模組的佈線圖(Layout)和輸出入接腳，本實驗要有兩套實驗器材和兩台電腦，每一套實驗器材均包括 Arduino MEGA2560 和 2.4G RF 模組，並連接上電腦，分別做成發射無線電和接收無線電，發射無線電的實驗器材稱為客戶端(client)，可藉由連接電腦觀看發射的無線電資料；接收無線電的實驗器材稱為伺服端(server)，可藉由連接電腦觀看接收的無線電資料，因程式用無線迴圈一直發射和接收無線電，就能比較 client 端和 server 端的電腦顯示資料是否一致，來證明無線收發的資料是否正確。

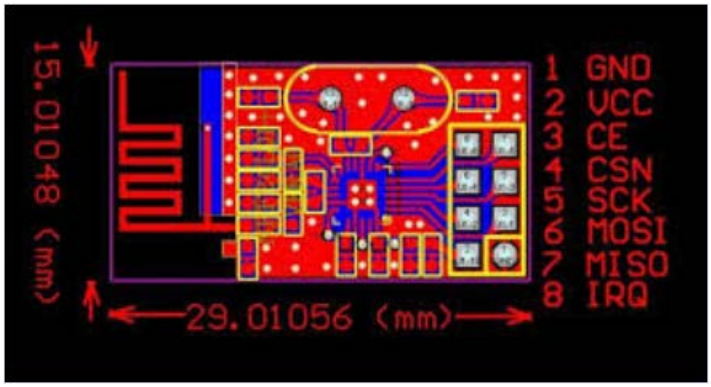


圖 2-14-1　2.4G RF 模組的佈線圖(Layout)和輸出入接腳

電路：Arduino MEGA2560 開發板上的數位接腳規劃 2.4G RF 模組，由表 2-14-1 來接線，就可用範例程式來使 2.4G RF 模組做發射與接收 2.4G Hz 的無線訊號，電路圖如圖 2-14-2 所示。

表 2-14-1　2.4G RF 模組與 Arduino MEGA2560 接線

	2.4G RF 模組	MEGA2560
8	IRQ	MEGA2560_50
6	MOSI	MEGA2560_52
5	SCK	MEGA2560_51

4	CSN	MEGA2560_A12
3	CE	MEGA2560_A11

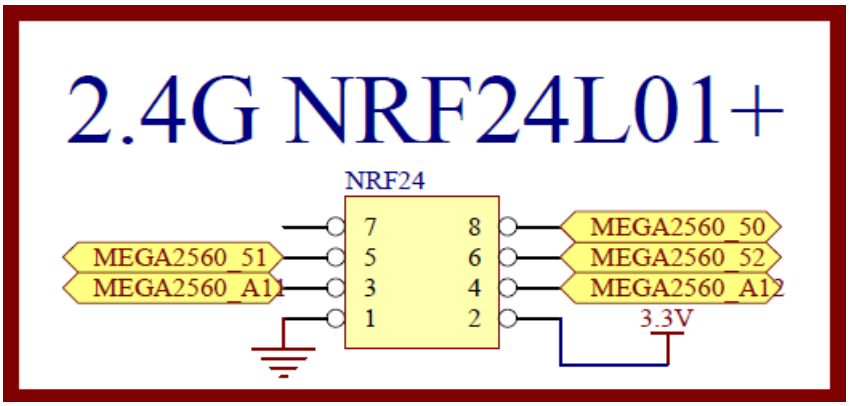


圖 2-14-2　Arduino MEGA2560 和 2.4G RF 模組電路

元件：

編號	元件項目	數量	元件名稱
1	NRF24	1	2.4G RF 模組

程式：本實驗共有兩個程式來完成 2.4GHz 的無線電發射與接收，分別為將無線電發射的 p2-14-1 ping_client_103 程式，藉由連接電腦可觀看發射的無線電資料；和將無線電接收的 p2-14-2 ping_server_103 程式，藉由連接電腦可觀看接收的無線電資料。

p2-14-1 ping_client_103 程式:

p2-14-1 ping_client_103

行號	程式敘述	註解
1	#include <SPI.h>	// 將 SPI.h 包含到程式，可驅動 SPI 介面
2	#include <Mirf.h>	// 將 Mirf.h 包含到程式，可驅動 rf 晶片
3	#include <nRF24L01.h>	// 將 nRF24L01.h 包含到程式，可驅動 nRF24L01 晶片
4	#include <MirfHardwareSpiDriver.h>	// 將 MirfHardwareSpiDriver.h 包含到程式，可驅動 SPI 介面
5	void setup(){	// 只會執行一次的程式初始設定函式
6	Serial.begin(9600);	// 將串列埠通訊鮑率設為 9600bps
7	Mirf.spi = &MirfHardwareSpi;	// 設定 nRF24L01 晶片的 SPI 硬體驅動
8	Mirf.init();	// 設定 nRF24L01 晶片的 SPI 接腳

9	Mirf.setRADDR((byte *)"clie1");	// 規劃發射的 nRF24L01 晶片的暫存器位址為"clie1"
10	Mirf.payload = sizeof(unsigned long);	// 將 nRF24L01 晶片的傳輸資料長度設定為 long(8bytes)
11	Mirf.config();	// 規劃 nRF24L01 晶片可開始發射
12	Serial.println("Beginning ... ");	// 序列埠監控視窗顯示 Beginning ...
13	}	// 結束 setup()函式
14	void loop(){	// 永遠周而復始的主控制函式
15	unsigned long time = millis();	// nRF24L01 晶片的無線傳輸資料為 millis()(經過時間(mS))，且資料型態為 long
16	byte data[Mirf.payload];	// 設定傳輸資料將 long 的資料存入 data[]陣列共 8bytes
17	Mirf.setTADDR((byte *)"serv1");	// 規劃接收的 nRF24L01 晶片的暫存器位址為" serv1"
18	Mirf.send((byte *)&time);	// nRF24L01 晶片無線傳輸資料為經過時間(mS)
19	while(Mirf.isSending()){ }	// 等到 nRF24L01 晶片無線傳輸資料完成
20	Serial.println("Finished sending");	// 序列埠監控視窗顯示 Finished sending
21	delay(10);	// 等待 10mS
22	while(!Mirf.dataReady()){	// 等待到 nRF24L01 晶片已經接收好資料
23	if ((millis() - time) > 1000) {	// 假如經過時間超過 1000mS(1 秒)
24	Serial.println("Timeout on response from server!");	// 序列埠監控視窗顯示 Timeout on response from server!(表示 server 未接收好資料)
25	return;	// 跳出 while 迴圈，不再等待
26	}	// 結束假如
27	}	// 結束 while 迴圈
28	Mirf.getData(data);	// 讀取 nRF24L01 晶片無線傳輸資料，並存放在 data[]陣列
29	for (int i;i<Mirf.payload;i++)	// 用 for 迴圈來顯示 data[]陣列的無線傳輸資料
30	{ Serial.print(data[i]);	// 序列埠監控視窗顯示 data[]陣列的無線傳輸資料
31	Serial.print("\n");	// 序列埠監控視窗顯示跳下一行
32	}	// 結束 for 迴圈
33	Serial.print("\nPing: ");	// 序列埠監控視窗顯示跳下一行後，再顯示 Ping:
34	Serial.println((millis() - time));	// 序列埠監控視窗顯示傳輸資料經過時間
35	delay(1000);	// 延時 1000mS，即延時 1 秒
36	}	// 結束 loop()函式

p2-14-2 ping_server_103 程式:

p2-14-1 ping_server_103

行號	程式敘述	註解
1	#include <SPI.h>	// 將 SPI.h 包含到程式，可驅動 SPI 介面
2	#include <Mirf.h>	// 將 Mirf.h 包含到程式，可驅動 rf 晶片

```
3   #include <nRF24L01.h>                          // 將 nRF24L01.h 包含到程式，可驅動 nRF24L01 晶片
4   #include <MirfHardwareSpiDriver.h>             // 將 MirfHardwareSpiDriver.h 包含到程式，可驅動 SPI 介面
5   void setup(){                                  // 只會執行一次的程式初始設定函式
6   Serial.begin(9600);                            // 將串列埠通訊鮑率設為 9600bps
7   Mirf.spi = &MirfHardwareSpi;                   // 設定 nRF24L01 晶片的 SPI 硬體驅動
8   Mirf.init();                                   // 設定 nRF24L01 晶片的 SPI 接腳
9   Mirf.setRADDR((byte *)"serv1");                // 規劃接收的 nRF24L01 晶片的暫存器位址為"serv1"
10        Mirf.payload = sizeof(unsigned long);    // 將 nRF24L01 晶片的傳輸資料長度設定為 long(8bytes)
11  Mirf.config();                                 // 規劃 nRF24L01 晶片可開始接收
12     Serial.println("Listening... ");            // 序列埠監控視窗顯示 Listening...
13  }                                              // 結束 setup()函式
14  void loop(){                                   // 永遠周而復始的主控制函式
15        byte data[Mirf.payload];                 // 設定傳輸資料將 long 的資料存入 data[]陣列共 8bytes
16     if(!Mirf.isSending() && Mirf.dataReady()){  // 假如 nRF24L01 晶片沒有傳送資料並且已經接收好資料
17        Serial.println("Got packet");            // 序列埠監控視窗顯示 Got packet
18  Mirf.getData(data);                            // 讀取 nRF24L01 晶片無線傳輸資料，並存放在 data[]陣列
19     for (int i;i<Mirf.payload;i++)              // 用 for 迴圈來顯示 data[]陣列的無線傳輸資料
20  { Serial.print(data[i]);                       // 序列埠監控視窗顯示 data[]陣列的無線傳輸資料
21  Serial.print("\n");                           // 序列埠監控視窗顯示跳下一行
22  }                                              //  結束 for 迴圈
23  Mirf.setTADDR((byte *)"clie1");                // 規劃接收的 nRF24L01 晶片的暫存器位址為" clie1"
24          Mirf.send(data);                       //  nRF24L01 晶片回傳到 client 資料為 data[]陣列
25  Serial.println("Reply sent.");                 // 序列埠監控視窗顯示 Reply sent.
26          }                                      // 結束假如
27  }                                              // 結束 loop()函式
```

練習：

一、 使用 client 端的 ARDUINO 發展板上的 4x4 鍵盤輸入數字並由 2.4G RF 模組發射此數字，接著由 server 端的 ARDUINO 發展板上的 2.4G RF 模組接收此數字後，顯示在 server 端的 LCD 上來觀看比對。

實驗 3-1: ADC VR 輸入電路功能

目的：利用 Arduino MEGA2560 開發板上的類比數位轉換器(ADC)的類比輸入讀到類比電壓值，並轉換為整數讀值由電腦序列埠監控視窗觀看。

功能：利用 Arduino 提供類比輸入接腳 A0 接腳來讀到類比電壓值，類比輸入接腳內部為類比數位轉換器(ADC)，可將類比電壓值轉換為 10 位元數位值，範圍為 0-1023 對應類比電壓 0-5V，可用工具->序列埠監控視窗觀看類比值。

原理：類比電壓值可用三端點的可變電阻來產生，電阻值為 10KΩ，可變電阻上下兩端接電源和地端，可變電阻中間端就能使用轉動可變電阻來產生 0-5V 的電壓，將電阻中間端直接接到 Arduino MEGA2560 提供類比輸入接腳編號 A0，即能用 Arduino 內部的 ADC 轉換電壓值(0-5V)成數位讀值(0-1023)。

電路：Arduino MEGA2560 開發板上的 A0 接到可變電阻的中間抽頭，可變電阻上下兩端接電源和地端，如此轉動可變電阻，就能讓 A0 量測到 0-5V 的類比電壓如圖 3-1-1 所示。

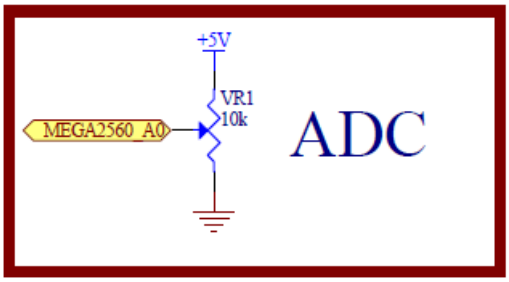


圖 3-1-1　ADC VR 輸入電路

元件：

編號	元件項目	數量	元件名稱
1	VR1	1	10k 可變電阻

程式：本實驗 Analoginserial 程式由 Arduino MEGA2560 提供類比輸入接腳編號 A0，即能用 Arduino 內部的 ADC 轉換電壓值(0-5V)成數位讀值(0-1023)，再用工具->序列埠監控視窗觀看類比值。

3-1-1 Analoginserial 程式:

Analoginserial

行號	程式敘述	註解
1	const int analogInPin = A0;	// 定義類比輸入接腳
2	int sensorValue = 0;	// 定義儲存類比轉換的數位值
3	void setup(){	// 只會執行一次的程式初始設定函式
4	Serial.begin(9600);	// 規劃序列埠的傳輸率 9600
5	}	// 結束 setup()函式
6	void loop(){	// 永遠周而復始的主控制函式

```
7         sensorValue = analogRead(analogInPin);    // 讀取類比轉換的數位值儲存在
                                                       sensorValue
8       Serial.print("sensor = " );                 // 序列埠監控視窗顯示 sensor =
9       Serial.print(sensorValue);                  // 序列埠監控視窗顯示 sensorValue
10      Serial.print("\n" );                        // 序列埠監控視窗顯示跳行
11      delay(100);                                 // 呼叫延遲函式等 100 毫秒=0.1 秒
12    }                                             // 結束 loop()函式
```

練習：

一、 修改程式使序列埠監控視窗顯示電壓值，並使用三用電表量測 ARDUINO 發展板的 A0 接腳的電壓值比較，看是否有誤差，若有誤差，說明誤差造成的原因。

實驗 3-2: PWM 類比輸出

目的：利用 Arduino MEGA2560 開發板上的 PWM 輸出接腳輸出 PWM 訊號控制 LED 亮度的電路與程式設計方法。

功能：利用 Arduino MEGA2560 開發板上輸出 PWM 接腳編號第 2、3、4、5、6、7、8 和 9 隻接腳，首先說明 PWM 為脈波寬度調變，由脈波寬度佔用脈波總週期的比例來換算成等值的類比電壓如圖 3-2-1 所示，使用 analogWrite(接腳編號, PWM 數值)函數來設定有 PWM 輸出接腳的 PWM 輸出數值，PWM 數值為 8 位元範圍為 0-255 對應類比電壓 0-5V，本實驗的數位 PWM 輸出為第 2、3、4、5、6、7、8 和 9 隻接腳來完成 LED 亮度控制，為了使 LED 有較高的亮度，本電路利用外接電源加上限流電阻，並加上 8 個邏輯閘緩衝器來驅動 LED，當 Arduino 數位接腳設定為 LOW(低電位)時產生順偏便得 LED 亮，所以 PWM 數值與亮度相反，即 PWM 數值 255 表示 LED 不亮，從 PWM 數值慢慢減少，LED 會慢慢變亮，當 PWM 數值為 0 時 LED 最亮。本實驗有兩個程式來控制 LED 亮滅，分別為控制 1 顆和 8 顆 LED 逐漸亮滅的兩個基礎程式，兩個程式的 LED 設定均是先 PWM 數值從 0 遞增到 255，控制 LED 亮度從亮到暗，再從 255 遞減，LED 亮度會漸亮，每個 PWM 數值遞增和遞減值都是 5，每個 PWM 數值的 LED 亮度停留 20mS，如此循環下去。

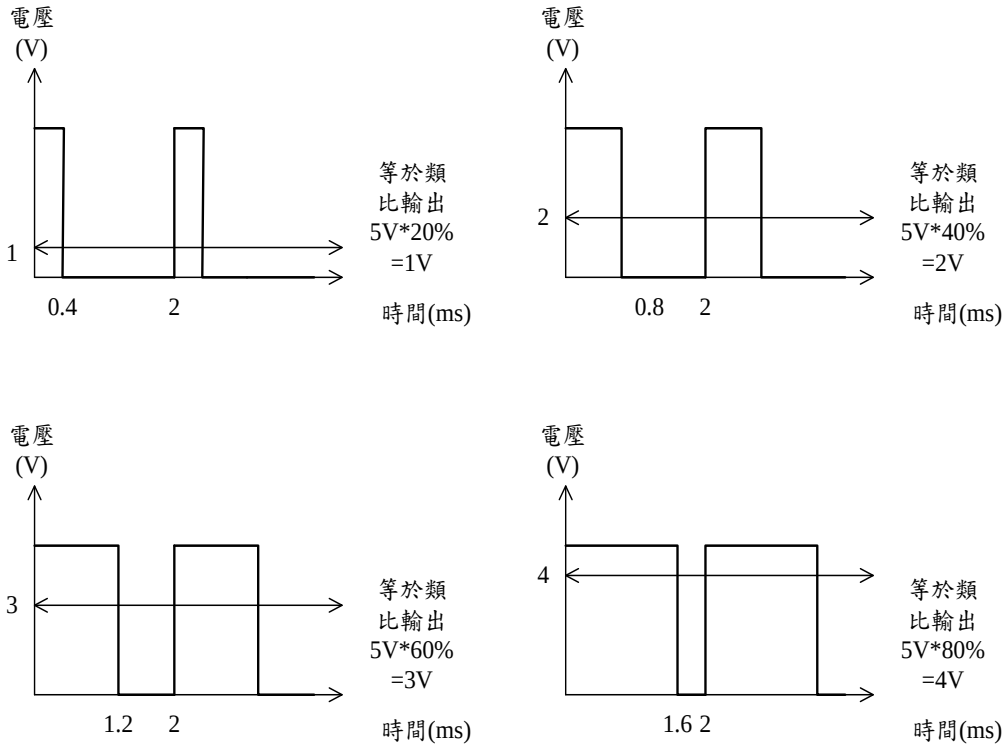


圖 3-2-1 脈波寬度與類比電壓對照圖

原理：Arduino PWM 輸出可用示波器觀察，為一個寬度變化，頻率固定在約 500Hz

的方波訊號，因電路圖 LED 亮度與 PWM 輸出訊號的寬度成反相，PWM 設定數值越大，PWM 寬度越寬，LED 越暗，反之 PWM 設定數值越小，PWM 寬度越窄，LED 越亮。

電路：Arduino MEGA2560 開發板上的 A15 數位輸出入腳接上 74LS244 的$1\overline{G}$和$2\overline{G}$，當輸出 LOW(低電位)來制能 74LS244，而 Arduino MEGA2560 開發板接腳第 2 到第 9 隻數位接腳，接上 74LS244(74AC244)的 8 個緩衝閘的 8 個輸入(1A1~4 和 2A1~4)，74LS244(74AC244)的 8 個緩衝閘輸出(1Y1~4 和 2Y1~4)接上 8 顆 LED，8 顆 LED 再接上 8 顆 300Ω的限流電阻後接到 5V 電源上如圖 3-2-2 和實驗 2-1 的圖 2-1-1 一樣。

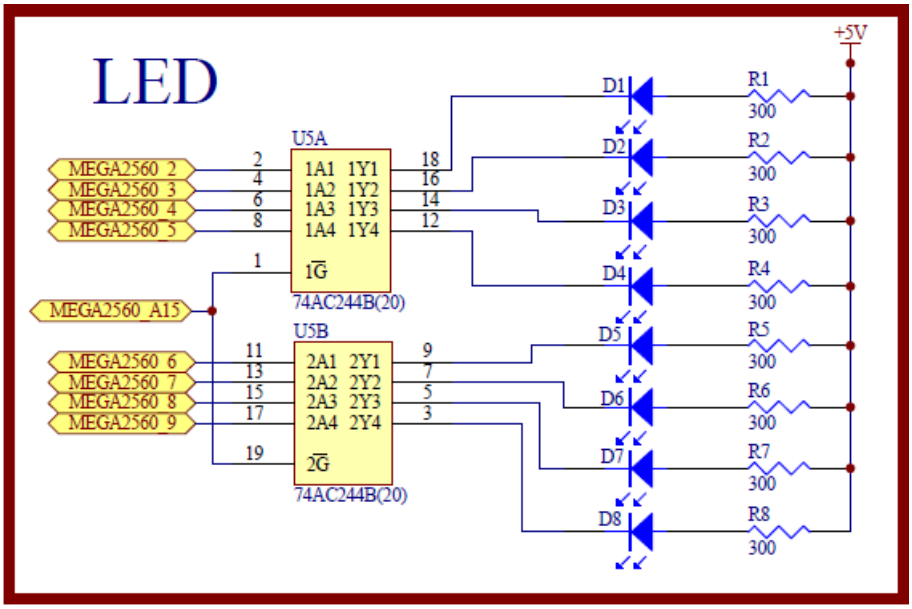


圖 3-2-2　LED 的 PWM 控制電路

元件：

編號	元件項目	數量	元件名稱
1	74AC244	1	8 個緩衝閘晶片
2	D1~D8	8	單色 LED
3	R1~R8	8	300 Ω電阻

程式：本實驗共有兩個程式來控制 LED 亮滅，分別為控制一顆 LED 逐漸亮滅的 p3-2-1 BreathingLed 程式和 8 顆 LED 逐漸亮滅的 p3-2-2 All_BreathingLed 程式，兩個程式的 LED 設定均是先 PWM 數值從 0 遞增到 255，控制 LED 亮度從亮到暗，再從 255 遞減，LED 亮度會漸亮，每個 PWM 數值遞增和遞減值都是 5，每個 PWM 數值的 LED 亮度停留 20mS，如此循環下去。

p3-1-1 BreathingLed 程式:

p3-2-1 BreathingLed

行號	程式敘述	註解
1	int brightness = 0;	// 定義開始 LED 亮度 0 為最亮
2	int fadeAmount = 5;	// 定義每次遞增或遞減的數位值
3	int delayDuration = 20;	// 定義每次控制 LED 亮度的 PWM 值維持 20mS
4	void setup(){	// 只會執行一次的程式初始設定函式
5	pinMode(led,OUTPUT);	// 規劃 LED 腳為輸出模式
6	pinMode(A15,OUTPUT);	// 規劃 A15 腳為輸出模式
7	digitalWrite(A15, LOW);	// A15 輸出 LOW，制能 74AC244
8	}	// 結束 setup()函式
9	void loop(){	// 永遠周而復始的主控制函式
10	analogWrite(2, brightness);	// 設定 Arduino 第二接腳的 LED 輸出 PWM 值 brightness，用 brightness 控制 LED 亮度
11	brightness = brightness + fadeAmount;	// PWM 值 brightness 每次遞增或遞減值為 fadeAmount
12	if (brightness <= 0 || brightness >= 255){	// 當 PWM 值 brightness 小於等於 0 或大於等於 255，即超過範圍時改變設定
13	fadeAmount = -fadeAmount ; }	// 將遞增改遞減、遞減改遞增
14	delay(delayDuration);	// 呼叫延遲函式等 20 毫秒，使控制 LED 亮度的 PWM 值維持 20mS
15	}	// 結束 loop()函式

p3-2-2 All_BreathingLed 程式:

p3-2-2 All_BreathingLed

行號	程式敘述	註解
1	int BASE = 2;	// 定義第一顆 LED 接的 I/O 腳
2	int NUM = 8;	// 定義 LED 的總數
3	int brightness = 0;	// 定義開始 LED 亮度 0 為最亮
4	int fadeAmount = 5;	// 定義每次遞增或遞減的數位值
5	int delayDuration = 20;	// 定義每次控制 LED 亮度的 PWM 值維持 20mS
6	void setup(){	// 只會執行一次的程式初始設定函式
7	for (int i = BASE; i < BASE + NUM; i ++) {	// 使用 for 迴圈規劃 8 顆 LED 腳
8	pinMode(i,OUTPUT); }	// 規劃 8 顆 LED 腳為輸出模式
9	pinMode(A15,OUTPUT);	// 規劃 A15 腳為輸出模式
10	digitalWrite(A15, LOW);	// A15 輸出 LOW，制能 74AC244
11	}	// 結束 setup()函式
12	void loop(){	// 永遠周而復始的主控制函式
13	for (int i = BASE; i < BASE + NUM; i ++) {	// 使用 for 迴圈設定輸出 8 顆 LED 電壓
14	analogWrite(i, brightness); }	// 8 顆 LED 輸出用 PWM 值 brightness 設定亮度
15	brightness = brightness + fadeAmount;	// PWM 值 brightness 每次遞增或遞減值為 fadeAmount
16	if (brightness <= 0 || brightness >= 255){	// 當 PWM 值 brightness 小於等於 0 或大於等於 255，即超過範圍時改變設定
17	fadeAmount = -fadeAmount ; }	// 將遞增改遞減、遞減改遞增
18	delay(delayDuration);	// 呼叫延遲函式等 20 毫秒，使控制 LED 亮度的 PWM 值維持 20mS
19	}	// 結束 loop()函式

練習：

一、 使用示波器量測 ARDUINO 發展板的第 2 隻接腳的 PWM 的波形，並和 Tools->Serial Monitor 觀看的 PWM 數值比較並繪出 PWM 數值所對應的示波器波形。

二、 請使用 8 顆 LED 用 ARDUINO 發展板的 8 隻 PWM 接腳的波形，來產生有拖曳效果的霹靂燈。

實驗 3-3：亮度控制實驗

目的： 瞭解亮度控制的設計與控制方法

功能： 利用光電二極體受不同光度照射後，會產生不同的光電流，並將光電流的變化轉換成電壓的變化，同時經 MEGA2560 之 ADC(類比轉數位電路)轉換成相對應數位值。

原理：表 3-3-1 為光電二極體(HW3P-1)之特性，實驗電路如圖 3-3-1 所示，MEGA 2560 的 A6 接到電路 Vo，A6 接受光電二極體的光電流變化，所轉換的電壓，同時利用 MEGA 2560 的 A/D 轉換器，轉換成相對應數位值。

程式 1：是利用 Arduino 的序列埠監控視窗，觀察 A/D 轉換的狀況。

程式 2：是利用光電二極體電路，作為 LED 的亮度控制。

表 3-3-1　光電二極體-HW3P-1 特性　(資料來源：HW3P-1 手冊)

參數	符號	測試條件	最小	典型	最大	單位
集電極光電流	I_C	V_{CE}=5V, $E_{v=}$ 10 Lux, ($E_{e=}$1Mw/cm^2) ※2	17		19	微安培
集電極暗電流	I_{CEO}	V_{CE}=5V, $E_{e=}$ 0 ※2			10	納安
集電極-發射級飽和壓降	$V_{CE(sat)}$	I_C=20mA,I_B=100uA			2.0	伏
峰值波長	λp			850		納米
光譜靈敏度	Δλ			450~1050		納米
半角度	Δθ			±60		度
反應時間（上升）	t_r	V_{CC}=5V, I_C=1mA		15		微秒
反應時間（下降）	T_f	R_L=1K		15		微秒

※2 E_v ,E_e are illuminance irradiant by CIE standard light source A(tungsten lamp)at 2856K。

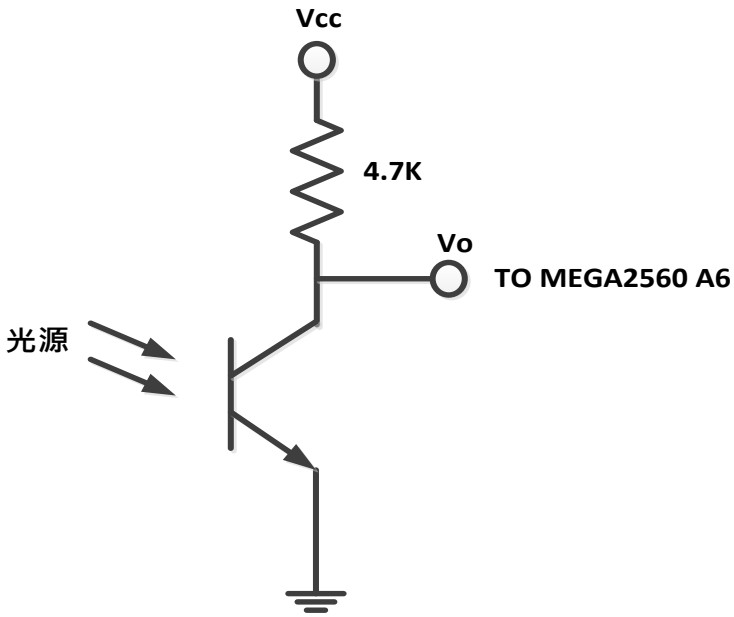


圖 3-3-1　光電二極體電路

元件：

編號	元件項目	數量	元件名稱
1	MEGA2560	1	Arduino 開發板
2	電阻	1	4.7k
3	光電二極體	1	HW3P-1

程式：1

行號	程式敘述	註解
1	int Pin = A6;	// 選擇光電二極體輸入腳位 A0
2	Int Value = 0;	// 定義光電二極體光度轉換成數位值之變數
3		
4	void setup() {	// setup()函式開始
5	Serial.begin(9600);	// 定義序列埠監控視窗傳遞速率為 9600
6	}	// setup()函式結束
7		
8	void loop() {	// loop()函式開始
9	Value = analogRead(Pin);	// 讀取光電二極體光度的數位值
10	Serial.print("Value = ");	//傳送到序列埠監控視窗

11	Serial.println(Value);	//傳送到序列埠監控視窗
12	delay(10);	// loop ()函式結束
	}	

程式：2

行號	程式敘述	註解
1	int Pin = A6;	// 選擇光電二極體輸入腳位 A0
2	int Value = 0;	// 定義光電二極體轉光度換成數位值之變數
3	int ledPin = 3;	// 選擇 LED 輸出腳位 D3
4	void setup() {	// setup()函式開始
5	pinMode(ledPin, OUTPUT);	// 定義 ledPin 為輸出
6	pinMode(A15, OUTPUT);	
7	digitalWrite(A15, LOW);	
8	}	// setup()函式結束
9	void loop() {	// loop()函式開始
10	Value = analogRead(Pin);	//讀取光電二極體光度的數位值
11	analogWrite(ledPin, Value/4);	//LED 亮度控制
12	}	// setup()函式結束

練習：

一、 請利用圖 3-3-1 光電二極體電路，模擬路燈自動開啟、關閉控制。

實驗 3-4：直流馬達控制實驗

目的：瞭解直流馬達的工作原理及程式控制方法

功能：本實驗使用直流馬達驅動 IC(L298N)控制馬達正反轉及轉速。程式(一)中的動作為正轉 1 分鐘、停止 0.5 分鐘，再反轉 1 分鐘，重複執行。程式(二)中是以 PWM(Pulse Width Modulation)的方式控制直流馬達的轉速，其動作為由停止慢慢加速到最快，再由最快減速到停止。

原理：使用 MCU 去控制大電流之負載都會使用到電流放大電路，主要原因是一般 MCU 的電流輸出大約只有 20mA，Arduino 也是，甚至目前講求低功耗的 MCU 只有 8mA 或更少，因此需要由兩個電晶體組成的電路「達靈頓電路」來做電流放大，「TIP12X」系列即為達靈頓電路 IC，但這類電晶體只能單一方向控制直流馬達轉動，如果要改變轉動方向，就必須要能改變電流流向之驅動電路，這時就要用到所謂「H Bridge」也就是俗稱的「H 橋」電路，H 橋就是由四個電晶體組成，如圖 3-4-1 所示，由 MCU 輸入到基極(Base)的電位決定電晶體集極(Collector)與射極(Emitter)是否導通，當 MCU 輸出高電位於 Q1、Q4 的基極時 Q1 和 Q4 會導通，電流從左側流進直流馬達，假設此時直流馬達的轉動為正轉。反之，當 Q2 和 Q3 導通時，電流右側流進直流馬達，直流馬達的轉動會為反轉。市面上有將 H 橋電路封裝的 IC，本實驗是使用市售直流馬達控制模組，如圖 3-4-2 所示，其直流馬達驅動 IC 為 L298N，如圖 3-4-3 所示，其電路結構是將兩個 H 橋電路封裝成一個 IC 的產品。L298N 直流馬達控制模組電路圖，如圖 3-4-4 所示。L298N 直流馬達控制模組有兩個 H 橋電路，可控制兩個直流馬達，控制真值表如表 3-4-1、3-4-2 所示。

注意：的是 MCU 的供電與馬達的供電建議要分開，才不會發生電流不穩定造成 MCU 當機的情況，要記得將 MCU 的電源接地腳與馬達電源的接地腳共接！

註：如果你的直流馬達和驅動是合在一起，如圖 3-4-6 所示，其接線請參照表 3-4-5 所示。

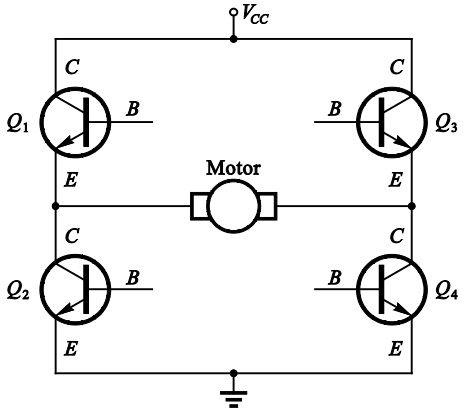


圖 3-4-1、H 橋電路

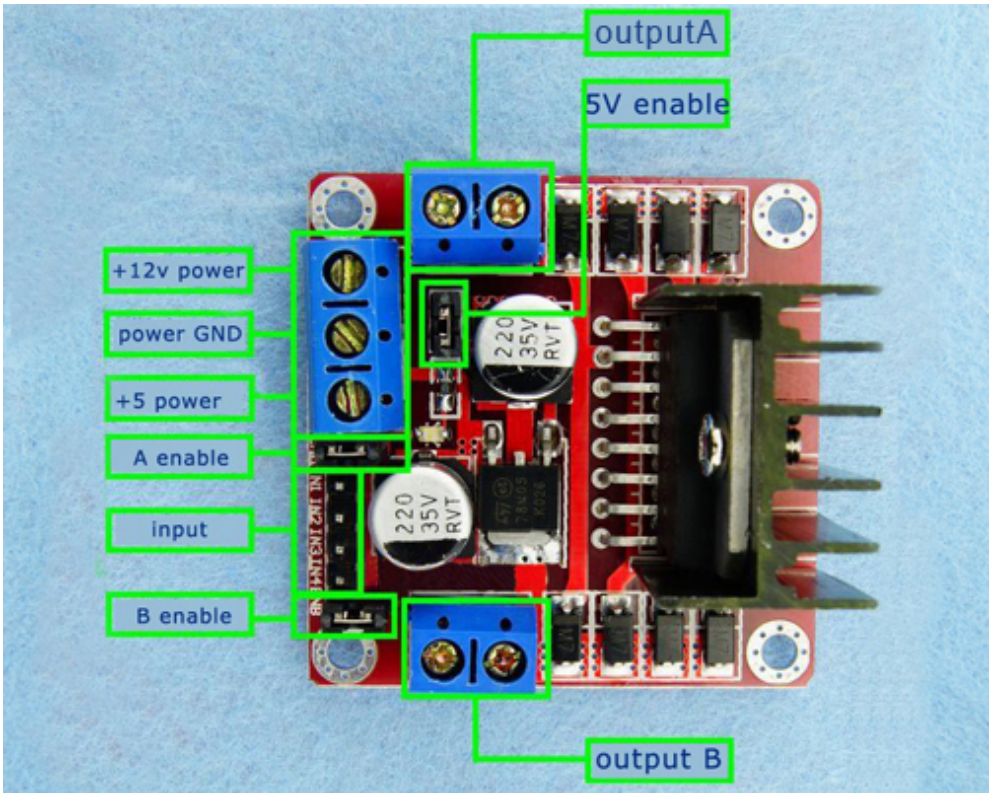


圖 3-4-2、L298N 模組(資料來源：www.buyic.com.tw)

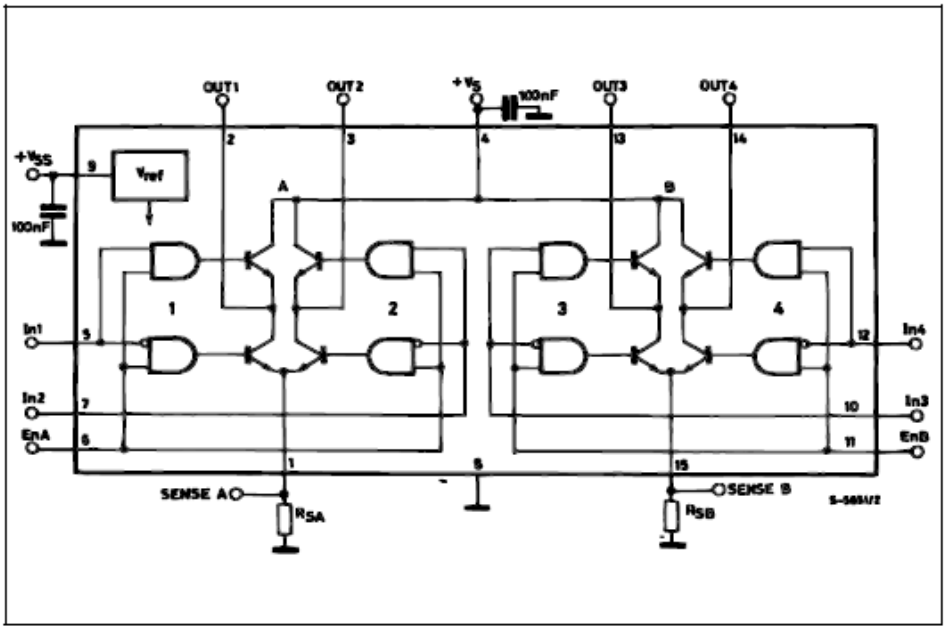


圖 3-4-3、L298N 電路結構圖(資料來源：STMicroelectronics 資料手冊)

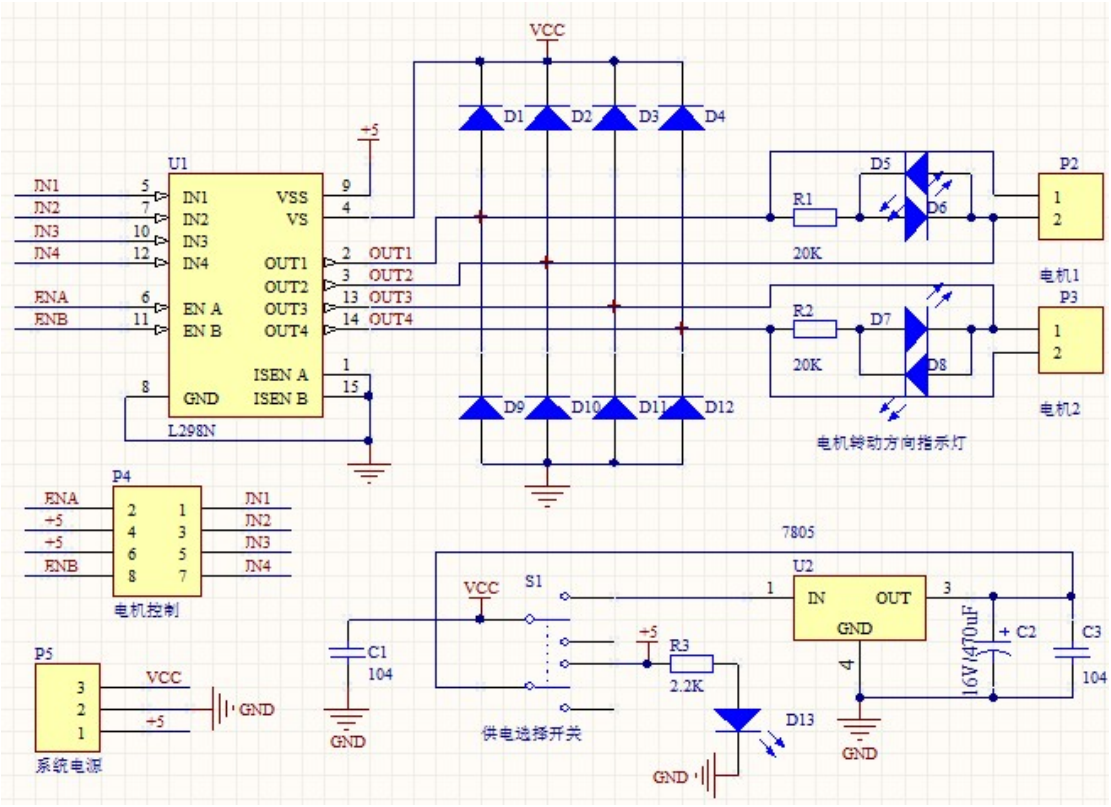


圖 3-4-4、L298N 模組電路圖(資料來源：www.buyic.com.tw)

表 3-4-1、直流馬達兩端接 OUT1、OUT2

IN1	IN2	直流馬達動作
0	0	停止
1	0	正轉
0	1	反轉
1	1	停止

表 3-4-2、直流馬達兩端接 OUT3、OUT4

IN3	IN4	直流馬達動作
0	0	停止
1	0	正轉
0	1	反轉
1	1	停止

電路圖：直流馬達，L298N 模組與 Arduino MEGA2560 開發板的接線如表 3-4-3、3-4-5 所示。

表 3-4-3　L298N 模組與 Arduino MEGA2560 接線

	L298N 模組	MEGA2560
1	IN1	D5
2	IN2	D6

表 3-4-4　L298N 模組與直流馬達接線

	L298N 模組	直流馬達
1	OUT1	+
2	OUT2	-

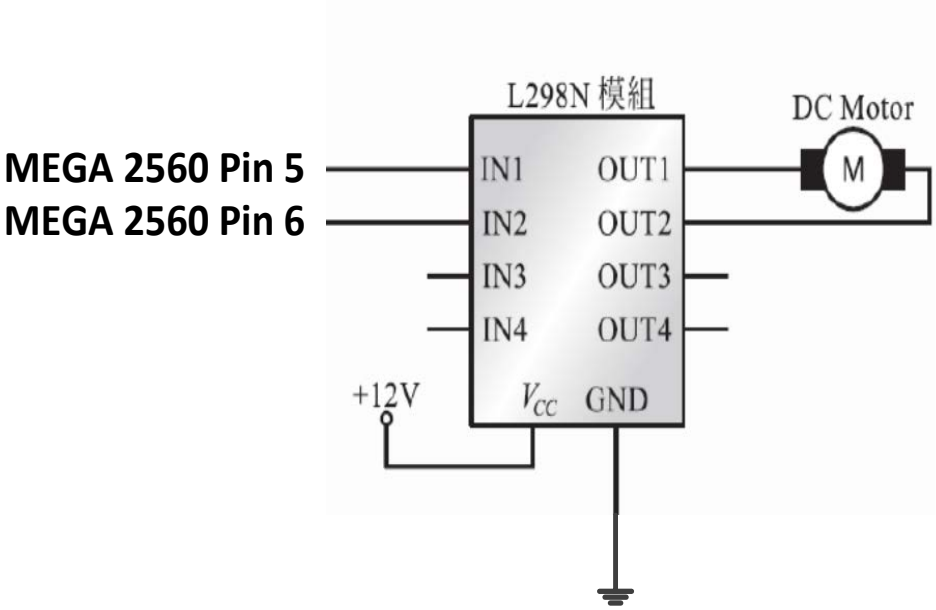


圖 3-4-5、直流馬達控制電路圖

圖 3-4-6 直流馬達和驅動合一

表 3-4-5 馬達控制模組與 MEGA2560 接線

	馬達控制模組	MEGA2560
1	Vcc	Vcc
2	Gnd	Gnd
3	1A	2
4	1B	3

5. 主要元件：

編號	元件項目	數量	元件名稱
1	Arduino 開發板	1	MEGA 2560
2	直流馬達	1	直流馬達(12V)
3	L298N 模組	1	L298N 模組

6. 程式設計(一)：正轉/反轉控制

```
/*   Input for motorA:
     IN1        IN2                Action
     LOW        LOW                Motor Stop
     HIGH       LOW                Motor moves forward
     LOW        HIGH               Motor moves backward
```

```
    HIGH        HIGH                    Motor Stop   */
1   const int motorIn1 = 5;             //設定直流馬達控制腳位為 Pin 5
2   const int motorIn2 = 6;             //設定直流馬達控制腳位為 Pin 6
3   const int DELAY = 1000;             //設定延時時間
4   void setup() {                      //setup()開始
5     pinMode(motorIn1, OUTPUT);        //設定腳位 Pin 5 為輸出
6     pinMode(motorIn2, OUTPUT);        //設定腳位 Pin 6 為輸出
7   }                                   //setup()結束
8
9   void loop(){                        // loop()開始
10    forward();                        //直流馬達正轉
11    delay(DELAY);
12    motorstop();                      //直流馬達停止
13    delay(500);
14    backward();                       //直流馬達反轉
15    delay(DELAY);
16    motorstop();                      //直流馬達停止
17    delay(500);
18  }                                   // loop()結束
19  void motorstop(){                   //直流馬達停止副程式
20    digitalWrite(motorIn1, LOW);      //Pin 5 = LOW
21    digitalWrite(motorIn2, LOW);      //Pin 6 = HIGH
22  }
23  void forward(){                     //直流馬達正轉副程式
24    digitalWrite(motorIn1, HIGH);     //Pin 5 = HIGH
25    digitalWrite(motorIn2, LOW);      //Pin 6 = LOW
26  }
27  void backward(){                    //直流馬達反轉副程式
28    digitalWrite(motorIn1, LOW);      //Pin 5 = LOW
29    digitalWrite(motorIn2, HIGH);     //Pin 6 = HIGH
30  }
```

程式設計(二)：轉速控制

```
/*    Input for motorA:
      IN1        IN2                Action
      LOW        LOW                Motor Stop
      HIGH       LOW                Motor moves forward
      LOW        HIGH               Motor moves backward
      HIGH       HIGH               Motor Stop   */
1     const int motorIn1 = 5;       //設定直流馬達控制腳位為 Pin 5
2     const int motorIn2 = 6;       //設定直流馬達控制腳位為 Pin 6
3     const int tt = 100;           //設定延時時間
4     void setup() {                //setup()開始
5       pinMode(motorIn1, OUTPUT);  //設定腳位 Pin 5 為輸出
6       pinMode(motorIn2, OUTPUT);  //設定腳位 Pin 6 為輸出
7     }                             //setup()結束
8
9     void loop(){                  // loop()開始
10      int i ;
11      digitalWrite(motorIn2, LOW);
12      for(i=100;i<=255;i+=10) {   //直流馬達加速
13        analogWrite(motorIn1,i);
14        delay(tt);
15      }
16      for(i=255;i>=100;i-=10) {   //直流馬達減速
17        analogWrite(motorIn1,i);
18        delay(tt);
19      }
20    }                             // loop()結束
```

7.練習

1. 請接二個直流馬達，撰寫一個正轉一個反轉之程式?
2. 請撰寫一個反轉之加速及減速程式?
3. 請撰寫利用可變電阻當輸入，控制直流馬達之轉速?

實驗 3-5：步進馬達控制實驗

目的： 瞭解步進馬達的工作原理及程式控制方法

功能： 本實驗分別以 1 相、2 相、1-2 相控制方法，完成步進馬達正反轉控制及轉速控制。

原理： 步進馬達的基本構造可分為定子與轉子，當電流流過定子時，其產生的磁場推動轉子，使轉子轉動。一般小型步進馬達多為四相式，可分為四相 5 線、四相 6 線及四相 8 線式，如圖 3-5-1 所示。四相是指定子上有四組相對線圈，稱為 A、B、$\overline{A}$ 及 $\overline{B}$，各提供 90^{o} 的相位差，其接線如圖 3-5-2 所示。若步進馬達為單極磁式，則每接收一個脈衝訊號，就會走一步即為轉動一個角度，稱為步進角，通常為 1.8° 或 0.9° 等

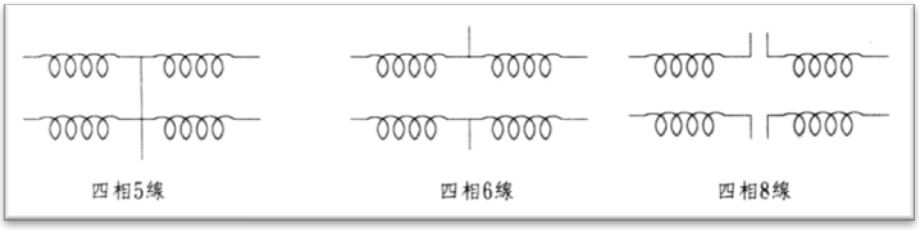


圖 3-5-1、步進馬達線圈接線種類

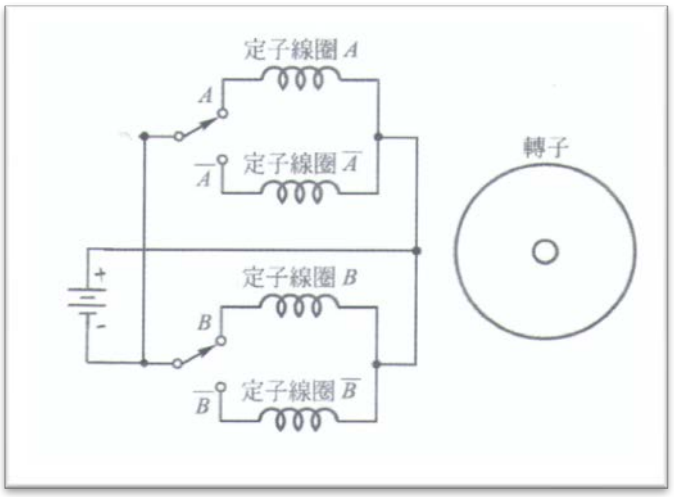


圖 3-5-2、步進馬達接線圖

四相式步進馬達定子線圈的激磁方式，會影響其轉動的角度及方向。其激磁的方式可以分成三種方式分別為一相激磁，二相激磁和一、二相激磁，分別如下描述：

A. 一相激磁：當脈衝訊號輸入後，四組線圈相位中只有一組相位激磁，即電流只通過其中一組線圈，每次激磁可轉動一個步進角，一相激磁如表 3-5-1 所

示。當由 STEP 1 至 STEP 8 方向作激磁，步進馬達會順時針轉動，反之由 STEP 8 至 STEP 1 方向作激磁，步進馬達會逆時針轉動。此種激磁方式會有轉動時力矩小、振動大和易失步等缺點。

表 3-5-1、一相激磁順序

	A	B	$\overline{A}$	$\overline{B}$
STEP 1	1	0	0	0
STEP 2	0	1	0	0
STEP 3	0	0	1	0
STEP 4	0	0	0	1
STEP 5	1	0	0	0
STEP 6	0	1	0	0
STEP 7	0	0	1	0
STEP 8	0	0	0	1

A. 二相激磁：當脈衝訊號輸入後，有二組相位激磁，即電流通過二組線圈，每次激磁可轉動一個步進角，其激磁表如表 3-5-2 所示。當由 STEP 1 至 STEP 8 方向作激磁，步進馬達會順時針轉動，反之由 STEP 8 至 STEP 1 方向作激磁，步進馬達會逆時針轉動。此種激磁方式會有轉動時力矩較大、振動小和不易失步。

表 3-5-2、二相激磁順序

	A	B	$\overline{A}$	$\overline{B}$
STEP 1	1	1	0	0
STEP 2	0	1	1	0
STEP 3	0	0	1	1
STEP 4	1	0	0	1
STEP 5	1	1	0	0
STEP 6	0	1	1	0
STEP 7	0	0	1	1
STEP 8	1	0	0	1

B. 一、二相激磁：將上述二種激磁合併交互激磁，每次激磁可轉動半個步進角，其激磁表如表 3-5-3 所示。當由 STEP 1 至 STEP 8 方向作激磁，步進馬達

會順時針轉動，反之由 STEP 8 至 STEP 1 方向作激磁，步進馬達會逆時針轉動。此種激磁方式會有轉動時較平滑，且振動的程度較低。

表 3-5-3、一、二相激磁順序

	A	B	$\overline{A}$	$\overline{B}$
STEP 1	1	0	0	0
STEP 2	1	1	0	0
STEP 3	0	1	0	0
STEP 4	0	1	1	0
STEP 5	0	0	1	0
STEP 6	0	0	1	1
STEP 7	0	0	0	1
STEP 8	1	0	0	1

電路： 步進馬達，L298N 模組與 Arduino MEGA2560 開發板的接線如表 3-5-4、3-5-5 及圖 3-5-3 所示。

注意：如果使用步進馬達是 28BYJ-48-5V(如圖 3-5-4 所示)，步進馬達驅動模組(如圖 3-5-5 所示)，其接線的如表 3-5-6 所示，實驗程式請參閱 p3_5_1F_v3.ino，p3_5_2F_v3.ino，p3_5_12F_v3.ino。

表 3-5-4　L298N 模組與 Arduino MEGA2560 接線

	L298N 模組	MEGA2560 接腳
1	IN1	0
2	IN2	1
3	IN3	2
4	IN4	3

表 3-5-5　L298N 模組與步進馬達接線

	L298N 模組	步進馬達
1	OUT1	A
2	OUT2	B

3	OUT3	$\overline{A}$
4	OUT4	$\overline{B}$

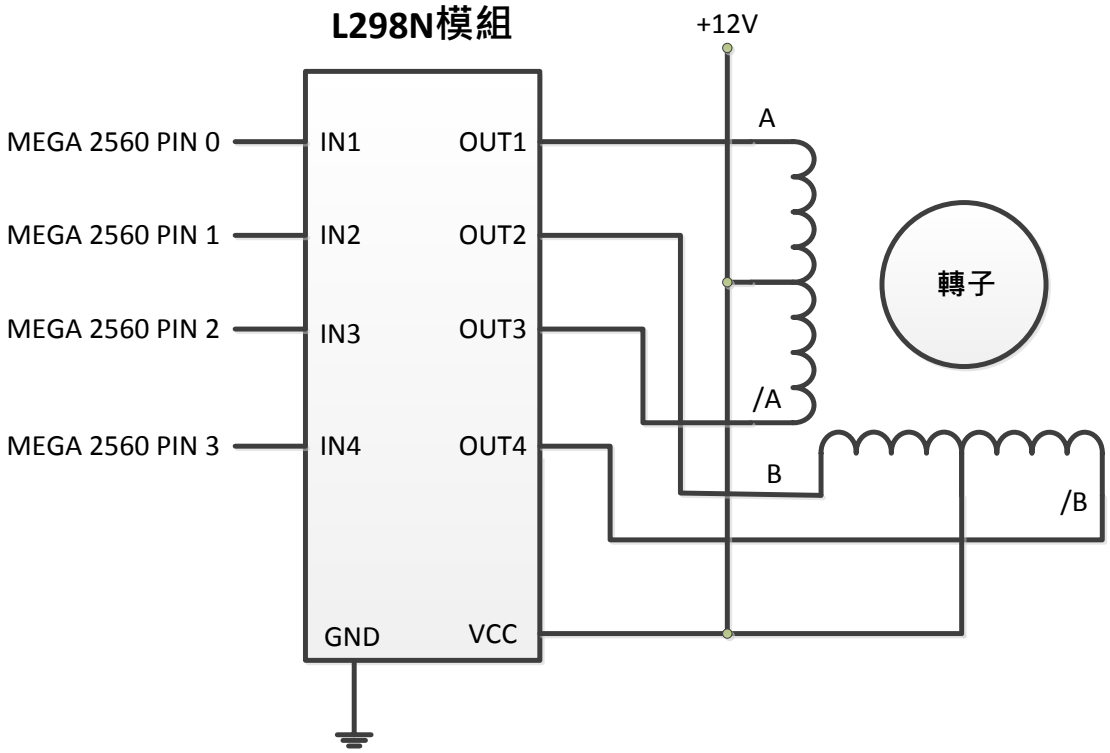


圖 3-5-3　步進馬達控制電路

元件：

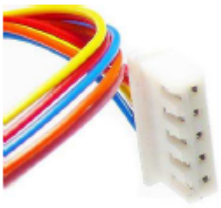
編號	元件項目	數量	元件名稱
1	MEGA2560	1	Arduino 開發板
2	步進馬達	1	步進馬達(12V)
3	L298N 模組	1	L298N 模組

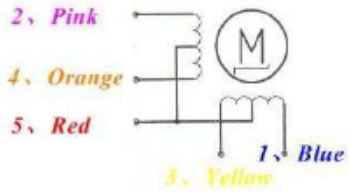
Kiatronics
electronic design and manufacture

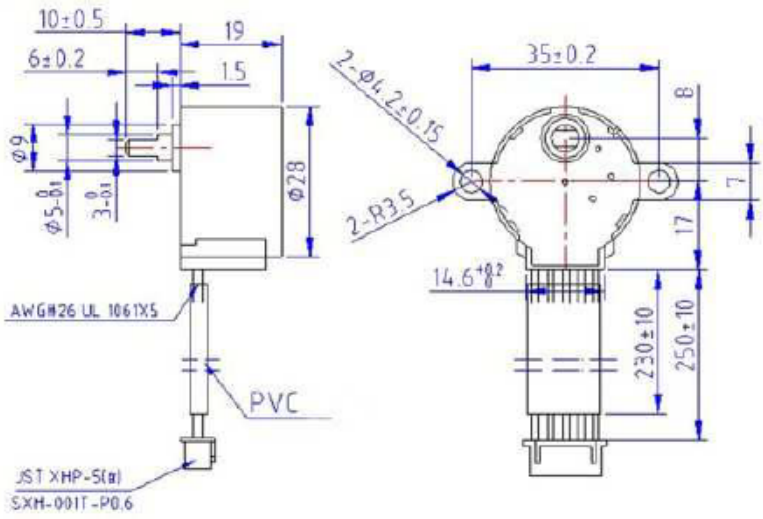
28BYJ-48 – 5V Stepper Motor

The 28BYJ-48 is a small stepper motor suitable for a large range of applications.

Rated voltage :	5VDC
Number of Phase	4
Speed Variation Ratio	1/64
Stride Angle	5.625°/64
Frequency	100Hz
DC resistance	50Ω±7%(25℃)
Idle In-traction Frequency	> 600Hz
Idle Out-traction Frequency	> 1000Hz
In-traction Torque	>34.3mN.m(120Hz)
Self-positioning Torque	>34.3mN.m
Friction torque	600-1200 gf.cm
Pull in torque	300 gf.cm
Insulated resistance	>10MΩ(500V)
Insulated electricity power	600VAC/1mA/1s
Insulation grade	A
Rise in Temperature	<40K(120Hz)
Noise	<35dB(120Hz,No load,10cm)
Model	28BYJ-48 – 5V







P.O. Box 8231 Cherrywood Tauranga New Zealand Phone: ++64 7 578 7739 Fax: ++64 7 578 7749 E-mail: enquiry@kiatronics.com
Website: www.kiatronics.com

圖 3-5-4 28BYJ-48-5V Step Motor

圖 3-5-5、步進馬達驅動模組

表 3-5-6、MEGA2560 步進馬達驅動模組與步進馬達接線

	MEGA2560 接腳	驅動模組	步進馬達
1	2	IN1	A
2	3	IN 2	B
3	4	IN 3	$\overline{A}$
4	5	IN 4	$\overline{B}$

程式設計(一)：一相激磁順時針轉動

```
1  #define A   0                          //A 相線圈接到 Arduino Pin 0
2  #define B   1                          //B 相線圈接到 Arduino Pin 1
3  #define A_BAR 2                        //Ā 相線圈接到 Arduino Pin 2
4  #define B_BAR 3                        //B̄ 相線圈接到 Arduino Pin 3
5  #define tt 20                          //轉速設定
6  unsigned                               //一相順時針激磁設定
   run1_F[4]={A,B,A_BAR,B_BAR};
7
8  void stop1(){
9     digitalWrite(A,LOW);                //A 相線圈不激磁
10    digitalWrite(B,LOW);                //B 相線圈不激磁
11    digitalWrite(A_BAR,LOW);            //Ā 相線圈不激磁
12    digitalWrite(B_BAR,LOW);            //B̄ 相線圈不激磁
13 }
14
15                                        void setup() {
16                                           int i ;
17                                           for(i=0;i<=3;i++)          //設定 Pin 0,1,2,3 為輸出
18                                              pinMode(i,OUTPUT);
19                                        }
20
21                                        void loop() {
22                                           int i ;
23                                           for(i=0;i<=3;i++){         //依 A，B，Ā，B̄ 一相激磁
24                                              stop1();
25                                              digitalWrite(run1_F[i],HIGH);
26                                              delay(tt);
```

```
27                                          }
28                                      }
```

程式設計(二)：二相激磁順時針轉動

```
1   #define A   0                           //A 相線圈接到 Arduino Pin 0
2   #define B   1                           //B 相線圈接到 Arduino Pin 1
3   #define A_BAR 2                         //Ā 相線圈接到 Arduino Pin 2
4   #define B_BAR 3                         //B̄ 相線圈接到 Arduino Pin 3
5   #define tt 20                           //轉速設定
6   unsigned run2_F[4][2]={{A,B} ,{B,       //二相順時針激磁設定
    A_BAR},{A_BAR,B_BAR},{B_BAR,A}};
7
8   void stop1(){
9     digitalWrite(A,LOW);                  //A 相線圈不激磁
10    digitalWrite(B,LOW);                  //B 相線圈不激磁
11    digitalWrite(A_BAR,LOW);              //Ā 相線圈不激磁
12    digitalWrite(B_BAR,LOW);              //B̄ 相線圈不激磁
13  }
14
15  void setup() {
16    int i ;
17    for(i=0;i<=3;i++)                     //設定 Pin 0,1,2,3 為輸出
18      pinMode(i,OUTPUT);
19  }
20
21  void loop() {
22    int i,j ;
23    for(i=0;i<=3;i++){                    //依 A，B， Ā ， B̄ 二相激磁
24     stop1();
25     for(j=0;j<=1;j++)
26       digitalWrite(run2_F[i][j],HIGH);
27    delay(tt);
28    }
29  }
```

程式設計(三)：一、二相激磁順時針轉動

```
1   #define A   0                                   //A 相線圈接到 Arduino Pin 0
2   #define B   1                                   //B 相線圈接到 Arduino Pin 1
3   #define A_BAR 2                                 //Ā 相線圈接到 Arduino Pin 2
4   #define B_BAR 3                                 //B̄ 相線圈接到 Arduino Pin 3
5   #define tt 20                                   //轉速設定
6   unsigned run1_F[4]={A,B,A_BAR,B_BAR};           //一相順時針激磁設定
7   unsigned run2_F[4][2]={{A,B} ,{B,               //二相順時針激磁設定
    A_BAR},{A_BAR,B_BAR},{B_BAR,A}};
8
9   void stop1(){
10     digitalWrite(A,LOW);                         //A 相線圈不激磁
11     digitalWrite(B,LOW);                         //B 相線圈不激磁
12     digitalWrite(A_BAR,LOW);                     //Ā 相線圈不激磁
13    digitalWrite(B_BAR,LOW);                      //B̄ 相線圈不激磁
14  }
15
16  void setup() {
17     int i ;
18     for(i=0;i<=3;i++)                            //設定 Pin 0,1,2,3 為輸出
19        pinMode(i,OUTPUT);
20  }
21
22  void loop() {
23     int i,j ;
24     for(i=0;i<=3;i++){                           //依 A，B，Ā ，B̄ 一、二相激磁
25       stop1();
26       digitalWrite(run1_F[i],HIGH);              //一相激磁
27       delay(tt);
28       stop1();
29       for(j=0;j<=1;j++)                          //二相激磁
30          digitalWrite(run2_F[i][j],HIGH);
31       delay(tt);
32     }
33  }
```

程式設計(四)：使用 Stepper()函數

```
1   #define A   0                    //A 相線圈接到 Arduino Pin 0
2   #define B   1                    //B 相線圈接到 Arduino Pin 1
3   #define A_BAR 2                  //Ā 相線圈接到 Arduino Pin 2
4   #define B_BAR 3                  //B̄ 相線圈接到 Arduino Pin 3
5   #define tt 20                    //轉速設定
6   #include <Stepper.h>             //引入 Stepper.h 檔
7   Stepper stepper(200, A,          //馬達轉一圈為 200 步 (1.8 deg)，定義
    A_BAR, B, B_BAR);                //0, 1, 2, 3 為輸出腳位
8
9   void setup(){
10      stepper.setSpeed(20);        // 將馬達的速度設定成 20 RPM
11  }
12
13  void loop(){
14      stepper.step(200);           //順時針 1 圈
15      stepper.step(-200);          //逆時針 1 圈
16  }
```

練習：

一、 請撰寫一個一相激磁逆時針轉動之程式?

二、 請撰寫一個二相激磁逆時針轉動之程式?

三、 請撰寫一個一、二相激磁逆時針轉動之程式?

實驗 3-6：超音波感測器實習

目的：瞭解超音波測距感測器的工作原理及使用 Arduino 程式控制方法。

功能：使用 Arduino MEGA2560 讀取超音波測距感測器測量到的障礙物距離值並顯示在 LCD 與序列埠觀測視窗上。

原理：超音波(Ultrasonic 或稱 Ultrasound)定義為超過人類耳朵能夠聽到的聲音，一般來說聲音頻率超過 16,000 Hz 就開始認定為超音波。而超音波感測器是由超音波發射器、接收器和控制電路所組成。當它被觸發的時候，會發射一連串 40 kHz 的聲波並且從離它最近的物體接收回音。如圖 3-6-1 所示，超音波測量距離的方法，將超音波發射器與接收器擺在同一方向，藉由發射出去遇到障礙物反射到接收器的音波所經歷的時間來作距離遠近的計算。

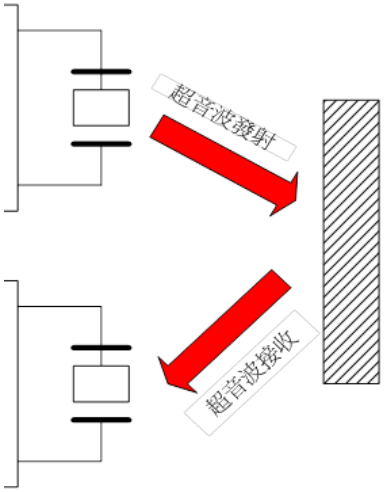


圖 3-6-1　超音波測量距離原理

聲音在空氣中的傳播速度大約是每秒 340 公尺，傳播速度會受溫度影響，溫度愈高，傳播速度愈快。假設超音波音速以每秒 340 公尺計算，可知聲音傳播 1 公分所需的時間為

$$Ts=1/(340*100)=2.94\times10^{-5}\ sec/cm=29.4\ \mu s/cm \fallingdotseq 29\ \mu s/cm$$

由於超音波從發射到返迴是兩段距離，因此在計算時必須將所經歷的時間 Td 微秒結果除以 2 才是正確的經歷的時間。所以物體距離的計算為

Distance=(Td /2)/Ts=(Td)/(2*29) =(Td)/(58) 公分

或

Distance=(Td /2)/Ts=(Td)/(2*29*2.54) ≒(Td)/(148) 英吋

超音波感測器主要應用在機器人或自走車避障、物體測距等。本實習使用 HC-SR04 超音波測距感測器(圖 3-6-2)，它可以探測的距離為 2cm-400cm，精度為 0.3 cm，感應角度為 15 度。

圖 3-6-2　HC-SR04 超音波測距感測器

電路： LCD 與 Arduino MEGA2560 開發板的接線如圖 3-6-3 所示，使用 6 支 IO 腳位接到 LCD 顯示模組的接腳。圖中 Arduino MEGA2560 開發板上的第(49, 48, 47, 43, 42, 41)支數位接腳對應連接到 LCD 顯示模組的第(4, 6, 11, 12, 13, 14)支接腳，Arduino 透過 lcd(rs, enable, d4, d5, d6, d7)函式來定義兩者間的對應關係。LCD 顯示模組的第 1 支接腳和第 5 支接腳一起接電源地端，以設定 LCD 顯示器只有寫入功能。電源(+5V)接到 LCD 顯示模組的第 2 支接腳，第 3 支接腳則經一顆 10k 可變電阻接到地端，藉由改變可變電阻值得到可變的分壓以得到適當的螢幕亮度對比。接著超音波測距感測器 HC-SR04 與 Arduino MEGA2560 開發板的接線方式很簡單，總共只有 4 支接腳，整體電路如圖 3-6-4 所示。圖中 Arduino MEGA2560 開發板上的數位 IO 第 12 支接腳接到 HC-SR04 Trig 接腳當輸出發射端，Arduino MEGA2560 開發板上的數位 IO 第 13 支接腳接到 HC-SR04 Echo 接腳當輸入接收端。程式中會在 LCD 與序列埠觀測視窗顯示超音波測距感測器測量到的障礙物距離值並以公分顯示。

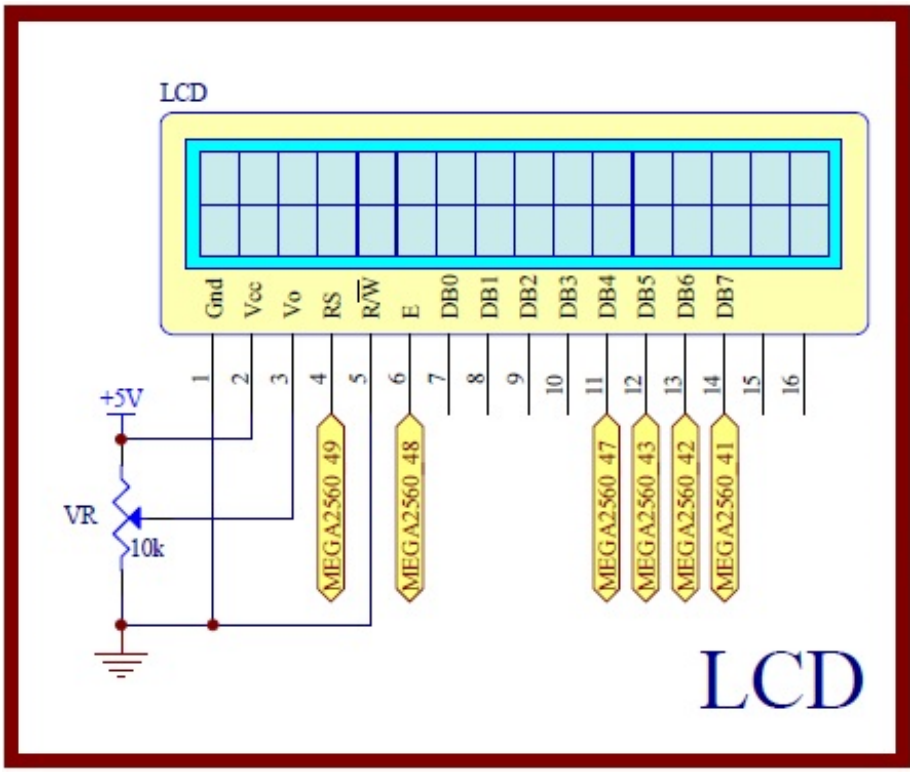


圖 3-6-3 Arduino MEGA2560 對應 LCD 顯示模組的實體接腳電路圖

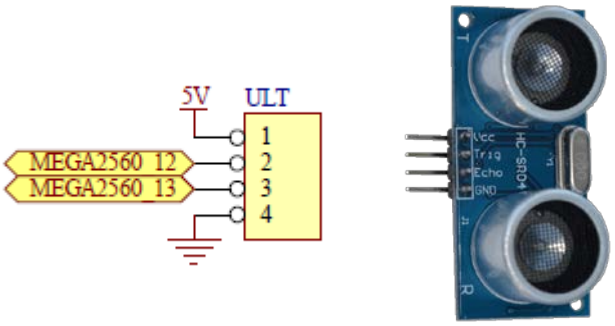


圖 3-6-4 電路

元件：本實習所需元件如表 3-6-1 所示。

表 3-6-1 元件表

編號	元件項目	數量	元件名稱
1	Arduino MEGA2560	1	Arduino 開發板
2	LCD	1	16×2 文字型 LCD 顯示器
3	VR	1	10kΩ 可變電阻
4	HC-SR04 模組	1	超音波測距感測器

程式： LCD 顯示超音波感測器測距值，並以公分顯示

P3_6_1

行號	程式敘述	註解
1	#define TrigPin 12	//定義 Trig 腳接在 D12
2	#define EchoPin 13	//定義 Echo 腳接在 D13
3	#include <LiquidCrystal.h>	//加入 LCD 顯示模組的驅動函式庫
4	LiquidCrystal lcd(49, 48, 47, 43, 42, 41);	// 定義 LCD 物件對應 Arduino 的腳位 // LiquidCrystal(rs, enable, d4, d5, d6, d7)
5	long ping() {	// 計算超音波往返時間的函式開始
6	digitalWrite(TrigPin, LOW);	// TrigPin 腳送出低電位
7	delayMicroseconds(12);	// 延遲 12 微秒
8	digitalWrite(TrigPin, HIGH);	// TrigPin 腳送出高電位
9	delayMicroseconds(10);	// 延遲 10 微秒
10	digitalWrite(TrigPin, LOW);	// TrigPin 腳送出低電位
11	return pulseIn(EchoPin, HIGH);	// 回傳測得往返時間，單位微秒
12	}	// 計算超音波往返時間函式結束
13	void setup(){	// 只會執行一次的程式初始函式
14	pinMode(TrigPin, OUTPUT);	// 設定 TrigPin 接腳為輸出
15	pinMode(EchoPin, INPUT);	// 設定 EchoPin 接腳為輸入
16	Serial.begin(9600);	// 將串列埠通訊鮑率設為 9600bps
17	lcd.begin(16, 2);	// 定義 LCD 顯示器為 16x2 文字型
18	lcd.print("Distance: cm");	// 輸出至 LCD 顯示
19	}	// 結束 setup()函式
20	void loop(){	// 永遠周而復始的主控制函式
21	long duration, cm;	// 變數宣告為長整數
22	String result = "Distance: ";	// 變數宣告為字串
23	duration = ping();	// 取得超音波發射到接收的往返時間
24	cm = duration / 58;	// 換算成距離(公分)
25	result += cm;	// 字串相加
26	result += " (cm).";	// 字串相加
27	Serial.println(result);	// 測得距離輸出至序列埠觀測視窗
28	lcd.setCursor(0, 0);	// 設定 LCD 顯示器的座標在第一行第一字
29	lcd.print("Distance: cm");	// LCD 顯示器顯示"Distance: cm"

```
30      lcd.setCursor(9, 0);                    // 設定 LCD 顯示器的座標在第一行第十字
31      lcd.print(cm);                          // 測得距離輸出至 LCD 顯示器顯示
32    delay( 500 );                             // 延遲 500 微秒
33    }                                         // 主控制函式結束
```

練習

一、 請將範例程式改成以英吋顯示超音波測障距離。

二、 請使用超音波測距感測器設計一個用手的距離遠近來調整蜂鳴器頻率的系統，亦即距離越近，蜂鳴器音頻越高，反之越低。

實驗 3-7: 紅外線感測器(接收器)

目的：使用 Arduino MEGA2560 開發板上的紅外線接收器來接收不同廠商的紅外線遙控器，並完成紅外線遙控繼電器的開關功能。

功能： 使用 Arduino MEGA2560 的紅外線接收器來讀取不同廠商的紅外線遙控器的串列碼，由序列埠監控視窗觀看解碼值，並經程式判斷為正確的編碼後，控制繼電器的開或關。

原理:不同廠商的紅外線遙控器使用 38kHz 的載波來發射不同的紅外線訊號串列碼，可由紅外線接收器來讀取不同廠商的紅外線遙控器的串列碼，使用 Ken Shirriff 的紅外線接收函式庫，可對不同廠商的遙控器解碼，由序列埠監控視窗觀看解碼值，並經程式判斷為正確的編碼後，控制繼電器的開或關。

電路：Arduino MEGA2560 開發板上的第 11 接腳接到紅外線接收器的訊號輸出，就能讓第 11 接腳獲取紅外線遙控器的串列碼如圖 3-7-1 所示。經程式判斷為正確的編碼後，由 Arduino MEGA2560 第 A4 隻數位接腳送出訊號來控制繼電器 ON、OFF 動作如圖 3-7-2 所示，詳細說明請參考實驗 2-2。

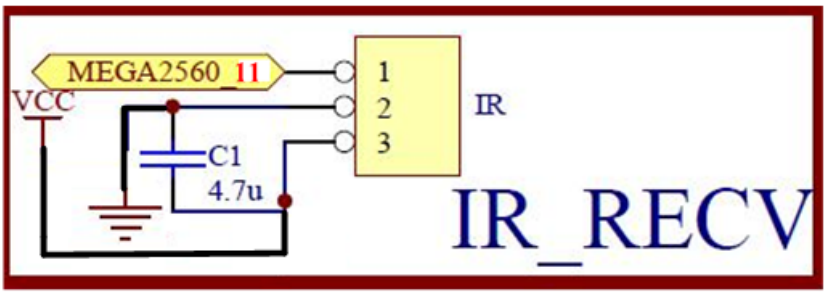


圖 3-7-1　紅外線接收電路

元件：

編號	元件項目	數量	元件名稱
1	VR1	1	10k 可變電阻

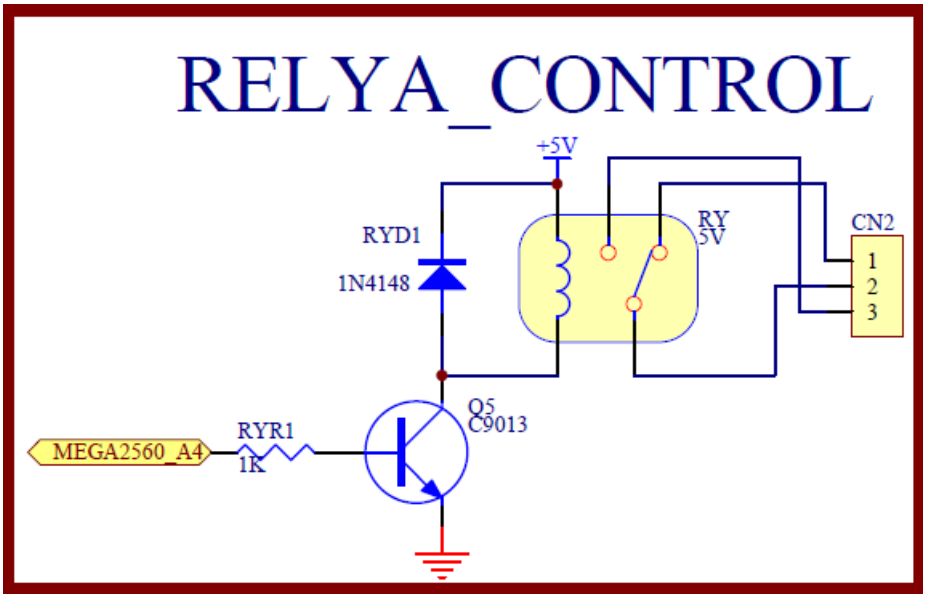


圖3-7-2　繼電器驅動電路

元件：

編號	元件項目	數量	元件名稱
1	Q5	1	NPN 電晶體(C9013)
2	RYD1	1	二極體(1N4148)
3	RYR1	1	1K Ω電阻
4	RY5V	1	單刀雙擲繼電器
5	CN2	1	三個輸出接頭

程式：本實驗共有兩個程式來完成紅外線遙控與接收，分別為只將紅外線接收的解碼由序列埠監控視窗觀看的 p3-7-1 IRrevDump 程式和除了紅外線接收的解碼由序列埠監控視窗觀看，並經程式判斷為正確的編碼後，控制繼電器的開或關的 p3-7-2 IRrelay 程式。

p3-7-1 IRrevDump 程式:

p3-7-1 IRrevDump

行號	程式敘述	註解
1	#include <IRremote.h>	// 將 IRremote.h 包含到程式
2	int RECV_PIN = 11;	// 定義 Arduino 第 11 接腳為紅外線接收腳
3	IRrecv irrecv(RECV_PIN);	// 紅外線接收副程式包含到程式
4	decode_results results;	// 定義解碼結果的變數為 results
5	void setup(){	// 只會執行一次的程式初始設定函式

```
6   Serial.begin(9600);                                  // 將串列埠通訊鮑率設為 9600bps
7   irrecv.enableIRIn();                                 // 開始接收紅外線串列碼
8   }                                                    // 結束 setup()函式
9   void dump(decode_results *results) {                 // 讀取接收的紅外線串列碼結果的副程式
10      int count = results->rawlen;                     // 定義紅外線串列碼的長度變數為 rawlen
11  if (results->decode_type == UNKNOWN) {               // 假如解碼結果的廠商未知時
12    Serial.print("Unknown encoding: ");    }           // 序列埠監控視窗顯示 Unknown encoding:
13  else if (results->decode_type == NEC) {              // 其他假如解碼結果的廠商為 NEC 時
14        Serial.print("Decoded NEC: ");}                // 序列埠監控視窗顯示 Decoded NEC:
15  else if (results->decode_type == SONY) {             // 其他假如解碼結果的廠商為 SONY 時
16        Serial.print("Decoded SONY: ");}               // 序列埠監控視窗顯示 Decoded SONY:
17  else if (results->decode_type == RC5) {              // 其他假如解碼結果的廠商為 RC5 時
18        Serial.print("Decoded RC5: ");}                // 序列埠監控視窗顯示 Decoded RC5:
19  else if (results->decode_type == RC6) {              // 其他假如解碼結果的廠商為 RC6 時
20        Serial.print("Decoded RC6: ");}                // 序列埠監控視窗顯示 Decoded RC6:
21  else if (results->decode_type ==                     // 其他假如解碼結果的廠商為
    PANASONIC) {                                           PANASONIC 時
22  Serial.print("Decoded PANASONIC -                   // 序列埠監控視窗顯示 Decoded
    Address: ");                                           PANASONIC - Address:
23  Serial.print(results->panasonicAddress,HEX);         // 序列埠監控視窗顯示讀取 Address 結
                                                           果，並用 16 進位表示
24          Serial.print(" Value: ");}                   // 序列埠監控視窗顯示 Value:
25  Serial.print(results->value, HEX);                   // 序列埠監控視窗顯示讀取 value 結果，並
                                                           用 16 進位表示
26  Serial.print(" (");                                  // 序列埠監控視窗顯示(
27  Serial.print(results->bits, DEC);                    // 序列埠監控視窗顯示讀取 bits 數的結
                                                           果，並用 10 進位表示
28  Serial.println(" bits)");                            // 序列埠監控視窗顯示 bits)
29  Serial.print("Raw (");                               // 序列埠監控視窗顯示 Raw (
30   Serial.print(count, DEC);                           // 序列埠監控視窗顯示讀取串列碼長度的
                                                           結果，並用 10 進位表示
31   Serial.print("): ");                                // 序列埠監控視窗顯示):
32   for (int i = 0; i < count; i++) {                   //  for 迴圈來將紅外線接收結果寫出
33   if ((i % 2) == 1) {                                 // 假如 i 除 2 餘數為 1 時，即 i 為奇數時
34   Serial.print( results->rawbuf[i]                    // 序列埠監控視窗顯示讀取串列碼第奇數
    *USECPERTICK, DEC);}                                   個的結果，並用 10 進位表示
35   else {                                              // 其他 i 為偶數時
36   Serial.print( results->rawbuf[i]                    // 序列埠監控視窗顯示讀取串列碼第偶數
    *USECPERTICK, DEC);    }                               個的結果，並用 10 進位表示
37   Serial.print("    "); }                             // 序列埠監控視窗顯示空白
38   Serial.print("    ");                               // 序列埠監控視窗顯示空白
39   }                                                   // 結束讀取接收的紅外線串列碼結果的副
                                                           程式
40   void loop(){                                        // 永遠周而復始的主控制函式
41   if (irrecv.decode(&results)) {                      // 假如紅外線接收器接收到串列碼
42      Serial.println(results.value, HEX);              // 序列埠監控視窗顯示讀取 value 結果，並
                                                           用 16 進位表示
43    dump(&results);                                    // 呼叫讀取接收的紅外線串列碼結果的副
                                                           程式
44      irrecv.resume();}                                // 呼叫可回到接收下一筆紅外線串列碼結
                                                           果狀態的副程式
45   }                                                   // 結束 loop()函式
```

p3-7-2 IRrelay 程式:

p3-7-2 IRrelay

行號	程式敘述	註解
1	#include <IRremote.h>	// 將 IRremote.h 包含到程式
2	int RECV_PIN = 11;	// 定義 Arduino 第 11 接腳為紅外線接收腳
3	int RELAY_PIN=A4;	// 定義 Arduino 第 A4 接腳為繼電器控制腳
4	IRrecv irrecv(RECV_PIN);	// 紅外線接收副程式包含到程式
5	decode_results results;	// 定義解碼結果的變數為 results
6	void dump(decode_results *results) {	// 讀取接收的紅外線串列碼結果的副程式
7	int count = results->rawlen;	// 定義紅外線串列碼的長度變數為 rawlen
8	if (results->decode_type == UNKNOWN) {	// 假如解碼結果的廠商未知時
9	Serial.print("Could not decode message "); }	// 序列埠監控視窗顯示 Could not decode message
10	else {	// 其他符合已知廠商編碼
11	if (results->decode_type == NEC) {	// 假如解碼結果的廠商為 NEC 時
12	Serial.print("Decoded NEC: ");}	// 序列埠監控視窗顯示 Decoded NEC:
13	else if (results->decode_type == SONY) {	// 其他假如解碼結果的廠商為 SONY 時
14	Serial.print("Decoded SONY: ");}	// 序列埠監控視窗顯示 Decoded SONY:
15	else if (results->decode_type == RC5) {	// 其他假如解碼結果的廠商為 RC5 時
16	Serial.print("Decoded RC5: ");}	// 序列埠監控視窗顯示 Decoded RC5:
17	else if (results->decode_type == RC6) {	// 其他假如解碼結果的廠商為 RC6 時
18	Serial.print("Decoded RC6: ");}	// 序列埠監控視窗顯示 Decoded RC6:
19	Serial.print(results->value, HEX);	// 序列埠監控視窗顯示讀取 value 結果，並用 16 進位表示
20	Serial.print(" (");	// 序列埠監控視窗顯示(
21	Serial.print(results->bits, DEC);	// 序列埠監控視窗顯示讀取 bits 數的結果，並用 10 進位表示
22	Serial.println(" bits)");	// 序列埠監控視窗顯示 bits)
23	Serial.print("Raw (");	// 序列埠監控視窗顯示 Raw (
24	Serial.print(count, DEC);	// 序列埠監控視窗顯示讀取串列碼長度的結果，並用 10 進位表示
25	Serial.print("): ");	// 序列埠監控視窗顯示):
26	for (int i = 0; i < count; i++) {	// for 迴圈來將紅外線接收結果寫出
27	if ((i % 2) == 1) {	// 假如 i 除 2 餘數為 1 時，即 i 為奇數時
28	Serial.print(results->rawbuf[i] *USECPERTICK, DEC);}	// 序列埠監控視窗顯示讀取串列碼第奇數個的結果，並用 10 進位表示
29	else {	// 其他 i 為偶數時
30	Serial.print(results->rawbuf[i] *USECPERTICK, DEC); }	// 序列埠監控視窗顯示讀取串列碼第偶數個的結果，並用 10 進位表示
31	Serial.print(" "); }	// 序列埠監控視窗顯示空白
32	Serial.print(" ");	// 序列埠監控視窗顯示空白
33	}	// 結束讀取接收的紅外線串列碼結果的副程式
22	void setup(){	// 只會執行一次的程式初始設定函式
23	pinMode(RELAY_PIN,OUTPUT);	// 規劃 Arduino 控制繼電器接腳為輸出模式
24	pinMode(13,OUTPUT);	// 規劃 Arduino 第 13 接腳 LED 為輸出模式
37	Serial.begin(9600);	// 將串列埠通訊鮑率設為 9600bps
38	irrecv.enableIRIn();	// 開始接收紅外線串列碼
39	}	// 結束 setup()函式
40	int on = 0;	// 設定繼電器 on 一開始為 0
41	unsigned long last = millis();	// 設定 last 為經過時間變數(毫秒)
42	void loop(){	// 永遠周而復始的主控制函式
43	if (irrecv.decode(&results)) {	// 假如紅外線接收器接收到串列碼
44	if (millis() - last > 250) {	// 呼叫 millis()得到經過時間，假如經過時間大於 250ms 時

```
45      on = !on;                                   // 將繼電器控制反相，使繼電器原本為 on 變 off，原本為 off 變 on
46    digitalWrite(RELAY_PIN, on ? HIGH : LOW);     // 繼電器控制腳輸出 HIGH 為 on，輸出 LOW 為 off
47      digitalWrite(13, on ? HIGH : LOW);          // LED 控制腳輸出 HIGH 為 on，輸出 LOW 為 off
48     dump(&results);                              // 呼叫讀取接收的紅外線串列碼結果的副程式
49    }                                             // 結束假如經過時間超過 250ms
50      last = millis();                            // 呼叫 millis()讀取經過時間存在 last
51      irrecv.resume();}                           // 呼叫可回到接收下一筆紅外線串列碼結果狀態的副程式
52    }                                             // 結束 loop()函式
```

練習：

一、 使用示波器量測 ARDUINO 發展板的第 11 隻接腳的紅外線接收串列碼的波形，並和 Tools->Serial Monitor 觀看的值比對。

實驗 3-8：溫濕度感測模組實習

目的：瞭解 DHT11 數位溫濕度感測器讀取溫度和濕度的工作原理及使用 Arduino 程式控制方法。

功能：使用 Arduino MEGA2560 讀取 DHT11 數位溫濕度感測器感測到的溫度和濕度值並顯示在 LCD 與序列埠觀測視窗上。

原理：DHT-11 模組是一款含有已校準數位信號輸出的溫濕度複合感測器，如圖 3-8-1 所示，它使用專用的數位模組資料擷取技術和溫濕度傳感技術，確保產品具有極高的可靠性與卓越的長期穩定性。DHT-11 模組包括一個電阻式感濕元件和一個 NTC（負溫度係數熱敏電阻）測溫元件，並與一個高性能 8 位元單晶片相連接，可以將所量測到的溫、濕度資料轉換成為數位訊號，再由 data 腳位將資料送出。

DHT-11 模組接線的方法十分簡單，VCC 腳接 3~5.5V，GND 腳接地，DATA 腳接上要輸入的 Pin 腳。因為數位訊號要接高 DATA 腳電位，所以把 DATA 腳外接 5K 歐姆提升電阻至 Vcc，如圖 3-8-2 所示。

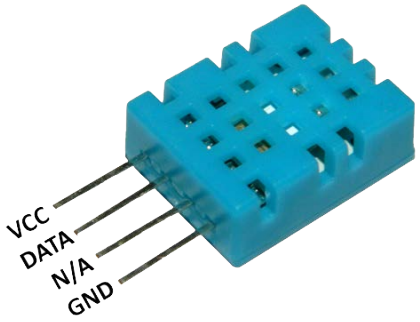


圖 3-8-1 DHT11 數位溫濕度感測器

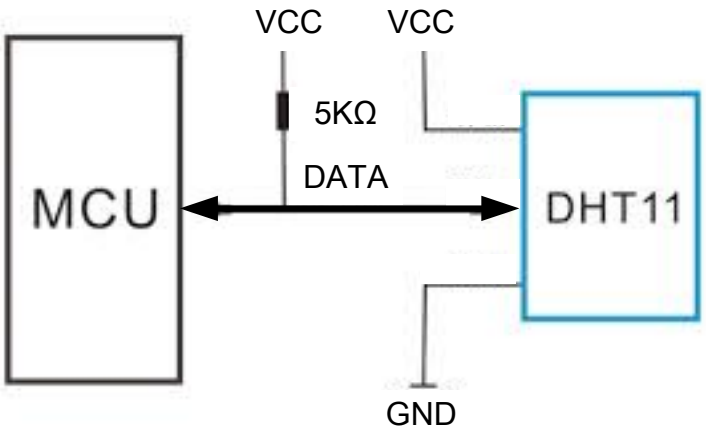


圖 3-8-2 DHT11 模組接線電路圖

DHT-11 模組其規格如下：

1、濕度測量範圍：20~90%RH;

2、濕度測量精度：±5%RH;

3、溫度測量範圍：0~50℃

4、溫度測量精度：±2℃

5、電源供應範圍： 3~5.5V

6、取樣時間不可低於 1 秒

7、量測腳接線長度不可超過 20 公尺

電路： LCD 與 Arduino MEGA2560 開發板的接線如圖 3-8-3 所示，使用 6 支 IO 腳位接到 LCD 顯示模組的接腳。圖中 Arduino MEGA2560 開發板上的第(49, 48, 47, 43, 42, 41)支數位接腳對應連接到 LCD 顯示模組的第(4, 6, 11, 12, 13, 14)支接腳，Arduino 透過 lcd(rs, enable, d4, d5, d6, d7)函式來定義兩者間的對應關係。LCD 顯示模組的第 1 支接腳和第 5 支接腳一起接電源地端，以設定 LCD 顯示器只有寫入功能。電源(+5V)接到 LCD 顯示模組的第 2 支接腳，第 3 支接腳則經一顆 10k 可變電阻接到地端，藉由改變可變電阻值得到可變的分壓以得到適當的螢幕亮度對比。接著 Arduino MEGA2560 開發板與數位溫濕度感測器 DHT11 的接線方式很簡單，總共只有 1 支接腳 A20 接到 DH11 模組 DATA 接腳。整體電路如圖 3-8-3 所示，圖中 DHT11 已模組化，只需接到 Arduino MEGA2560 開發板上的 CN1 介面接腳即可。程式中會在 LCD 與序列埠觀測視窗顯示 DHT11 感測器感測到溫度和濕度值。

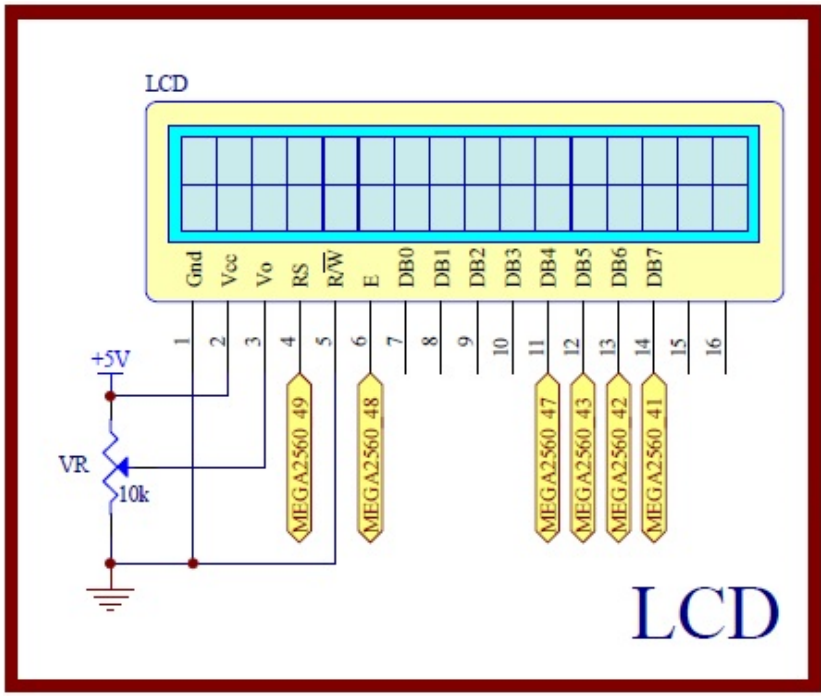


圖 3-8-2 Arduino MEGA2560 對應 LCD 顯示模組的實體接腳電路圖

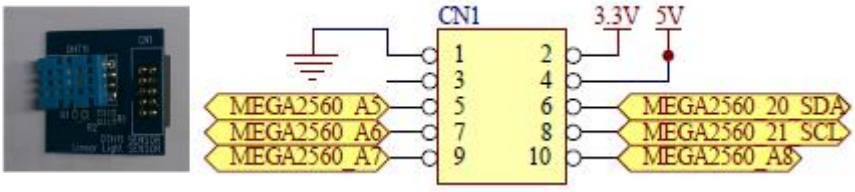


圖 3-8-3 Arduino MEGA2560 對應 DHT11 模組的實體接腳電路圖

元件：本實習所需元件如表 3-8-1 所示。

表 3-8-1 元件表

編號	元件項目	數量	元件名稱
1	Arduino MEGA2560	1	Arduino 開發板
2	LCD	1	16×2 文字型 LCD 顯示器
3	VR	1	10kΩ 可變電阻
4	DHT11 模組	1	數位溫濕度感測器

程式： LCD 顯示 DHT11 感測器感測溫濕度值，並顯示在序列埠觀測視窗上

P3_8_1

行號	程式敘述	註解
1	#include <dht11.h>	//加入 DHT11 模組的驅動函式庫
2	#include <LiquidCrystal.h>	//加入 LCD 顯示模組的驅動函式庫
3	#define DHT11PIN 20	//定義訊號要從 Pin A20 進來
4	dht11 DHT11;	//宣告 dht11 是 DHT11 類別
5	LiquidCrystal lcd(49, 48, 47, 43, 42, 41);	//定義 LCD 物件對應 Arduino 的腳位
6	void setup() {	//只會執行一次的程式初始式數
7	Serial.begin(9600);	//將串列埠通訊鮑率設為 9600bps
8	Serial.println("DHT11 TEST PROGRAM ");	//字串"DHT11 TEST PROGRAM "輸出至序列埠觀測視窗並換行
9	Serial.print("LIBRARY VERSION: ");	//字串"LIBRARY VERSION: "輸出至序列埠觀測視窗
10	Serial.println(DHT11LIB_VERSION);	//DHT11 模組的驅動函式庫版本輸出至序列埠觀測視窗並換行
11	Serial.println();	//序列埠觀測視窗換行
12	lcd.begin(16, 2);	//定義 LCD 顯示器為 16x2 文字型
13	lcd.clear();	//LCD 顯示器螢幕清除
14	}	//結束 setup()函式

```
15  void loop(){                                          //永遠周而復始的主控制函式
16    DHT11.read(DHT11PIN);                               //自 DHT11 的 DATA 腳讀取資料
17    Serial.print("Humidity (%): ");                     //字串"Humidity (%): "輸出至序列埠觀測視窗
18    Serial.println((float)DHT11.humidity, 2);           //濕度數值顯示到小數值兩位並輸出至序列埠觀測視窗且換行
19    lcd.setCursor(0, 0);                                //設定 LCD 顯示器的座標在第一行第一字
20    lcd.print("Humi:          % ");                    //LCD 顯示器顯示"Humi:     % "
21    lcd.setCursor(5, 0);                                //設定 LCD 顯示器的座標在第一行第六字
22    lcd.print((float)DHT11.humidity, 2);                //濕度數值顯示到小數值兩位並輸出至 LCD 顯示器
23    Serial.print("Temperature (oC): ");                 //字串"Temperature (oC): "輸出至序列埠觀測視窗
24    Serial.println((float)DHT11.temperature,2);         //溫度數值顯示到小數值兩位並輸出至序列埠觀測視窗且換行
25    lcd.setCursor(0, 1);                                //設定 LCD 顯示器的座標在第二行第一字
26    lcd.print("Temp:          oC");                     //LCD 顯示器顯示"Temp:         oC"
27    lcd.setCursor(5, 1);                                //設定 LCD 顯示器的座標在第一行第六字
28    lcd.print((float)DHT11.temperature, 2);             //溫度數值顯示到小數值兩位並輸出至 LCD 顯示器
29    delay(1000);                                        //每 1000ms 更新一次
30  }                                                     ///主控制函式結束
```

練習

一、 請將範例程式改成以華氏顯示溫度值。

實驗 3-9：Zigbee 模組通訊實習

目的：瞭解 Zigbee 模組的工作原理及使用 Arduino 程式控制如何通訊。

功能：本實習使用 Arduino MEGA2560 控制 ZigBee 模組做為無線通訊介面。程式(一)中會利用 PC 端透過 ZigBee 模組與主板 ZigBee 模組互傳字串並顯示。程式(二)中讀取可變電阻分壓類比值然後透過主板 ZigBee 模組傳給 PC 端做顯示。

原理：目前市面上有許多彼此不相容的無線網路技術，如 Wi-Fi、ZigBee、藍芽、超寬頻(UWB)，以及「近場通訊」(Near Field Communications)，這些標準雖然有類似的用途，但實際的使用情境又不盡相同。例如，Wi-Fi 有不錯的有效範圍及傳輸率，可以使用在大部份的無線網路場合，但是在可攜式裝置上，它的耗電量可能相當大。藍芽的耗電量相當小，一般認為適合作為手機等一類裝置的網路裝置。ZigBee 由於傳輸速率和距離的表現都沒有上述兩者好，被定位在收發少量的資料量的應用。

ZigBee 一詞源於蜜蜂，蜜蜂透過 ZigZag 字形舞蹈與同伴通信傳遞花與蜜的位置、方向、距離等訊息，因而藉此做為這短距無線通訊新技術的命名，主要是由 IEEE 802.15.4 小組與 ZigBee Alliance 兩個組織，分別制訂硬體與軟體標準。工作頻率為 868MHz （歐洲）、915MHz(美國)或 2.4GHz。

ZigBee 是一種短距離無線通訊標準，具有低成本、低耗電、雙向傳輸、高可靠度及感應網路功能等特性，容易整合個人無線數位環境並應用於多樣的產品，其「監控」角色高於「通訊」功能，朝著開放的方向發展制訂標準規範。一般 ZigBee 技術應用是藉由在各種物品上置入一個低耗電以及低成本的 ZigBee 無線模組來達到存取資訊以及進行控制的技術。有關 ZigBee 的優點包括：

(1) 彈性的傳輸距離：從數十公尺到數百公尺都可以簡單應用。
(2) 多節點傳輸：ZigBee 的網路特性提供您多節點以及系統化的快速應用。
(3) 長效電源：使用一對 AA 電池達到長達數年的工作時間。
(4) 低成本：因為 IEEE 802.15.4 低複雜性架構。
(5) 優秀的抗干擾能力：融合 CSMA/CA 以及 DSSS 的抗干擾技術。

本實驗使用ZigBee 模組（如圖3-9-1 所示）是專為智慧無線資料 ZigBee 傳輸而打造，其規格說明如下：

●輸入電壓範圍：DC 5-12V
□●溫度使用範圍：-400℃~850℃
□●通訊介面： RS232

- UART 鮑率：9600bps, 19200bps，38400bps，115200bps(預設)

- 使用無線頻率：2.4GHz
- 使用的通訊協定：ZigBee2007 /PRO
- 傳輸距離：無障礙可視環境下400公尺
- 發射時最大耗電流：34mA
- 接收時最大耗電流：25mA
- 接收靈敏度：-96dBm
- 使用晶片：CC2530F256，具有256KB FLASH

圖3-9-2 說明ZigBee 模組的硬體組成與功能按鍵作用。本實習要使用 ZigBee 模組作為資料的無線傳輸介面，基本上需要一對 ZigBee 模組，一組為ZigBee 協調者 (Coordinator)，負責網路初始化與網路的控制；另一組是 ZigBee 末端裝置 (End device)，透過協調者或路由器加入網路。在使用上並不需要懂太多ZigBee 技術，只要遵循模組的設定程序，並接到MEGA2560 主板上，即可像串列通訊般的使用它。

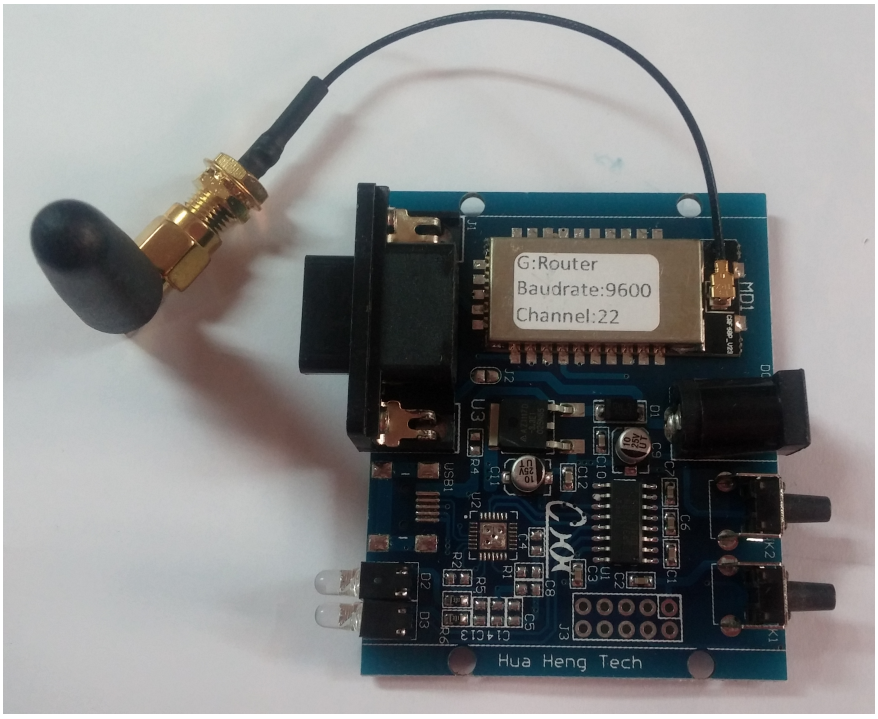


圖 3-9-1 DRF1601 ZigBee 模組實體圖

圖 3-9-2　DRF1601 ZigBee 模組功能說明

電路： ZigBee模組與Arduino MEGA2560開發板的實體接圖如圖3-9-3所示，圖 3-9-4 為實體接腳電路圖，圖中 Arduino MEGA2560 開發板上 RS232 串列埠 DB9_2 接到一組末端裝置 ZigBee 模組（標籤為黃色）上，另一組 ZigBee 模組為協調者（標籤為紅色）接到PC 端主機背後串列埠com1 上，接著將兩組ZigBee 模組接上DC 5V 電源後就會自行連線，連線成功後PC 端ZigBee 模組（協調者）收到資料 R4 上 LED 會快速閃滅，皆下來就可由終端機測試程式測試ZigBee 通訊。Arduino MEGA2560 開發板在串列傳輸上已有專用接腳供使用，其中軟體的函式 Serial 使用接腳0 (Rx)及1 (Tx)，函式 Serial 1使用接腳19 (Rx)及18 (Tx)，函式Serial 2 使用接腳17(Rx)及16(Tx)，函式Serial 3 使用接腳15(Rx)及14(Tx). 這些接腳可用於接收（Rx）和發送（Tx）TTL 串列資料。0 和 1 號接腳也連接到 ATmega2560 USB-to-TTL 串列轉換晶片的對應接腳上。本實習使用開發板上 RS232 串列埠DB9_2，所以必須使用，所以程式中需使用函式Serial 2 作為串列通訊介面。在PC 端必須先開啟工具資料夾內串列埠偵錯助手V2.2.exe 軟體，顯示畫面如圖 3-9-9。程式(一)中所示程式中 Arduino 端會不斷傳送"Hua Heng Arduino V3"給PC 端，此外利用PC 終端機測試程式輸入字串透過Zigbee 無線通訊到主板LCD(圖3-9-5 LCD 電路)作顯示，同時讓主板上8個LED(圖3-9-6 LCD 電路)亮滅一次，蜂鳴器(圖3-9-7 LCD 電路)”嗶”一聲。程式(二) Arduino 將讀取可變電阻分壓類比值(圖3-9-8 所示)，轉換成0~1024 區間值，然後透過主板ZigBee 模組傳給PC 端做顯示。

圖 3-9-3 Arduino MEGA2560 與 ZigBee 模組的實體接圖

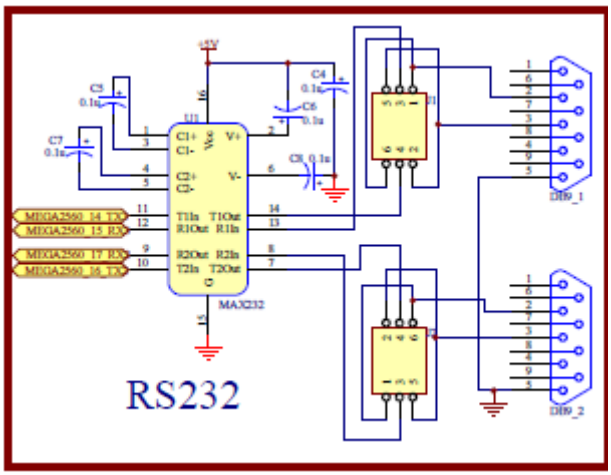


圖3-9-4 Arduino MEGA2560 對應ZigBee 模組的實體接腳電路圖

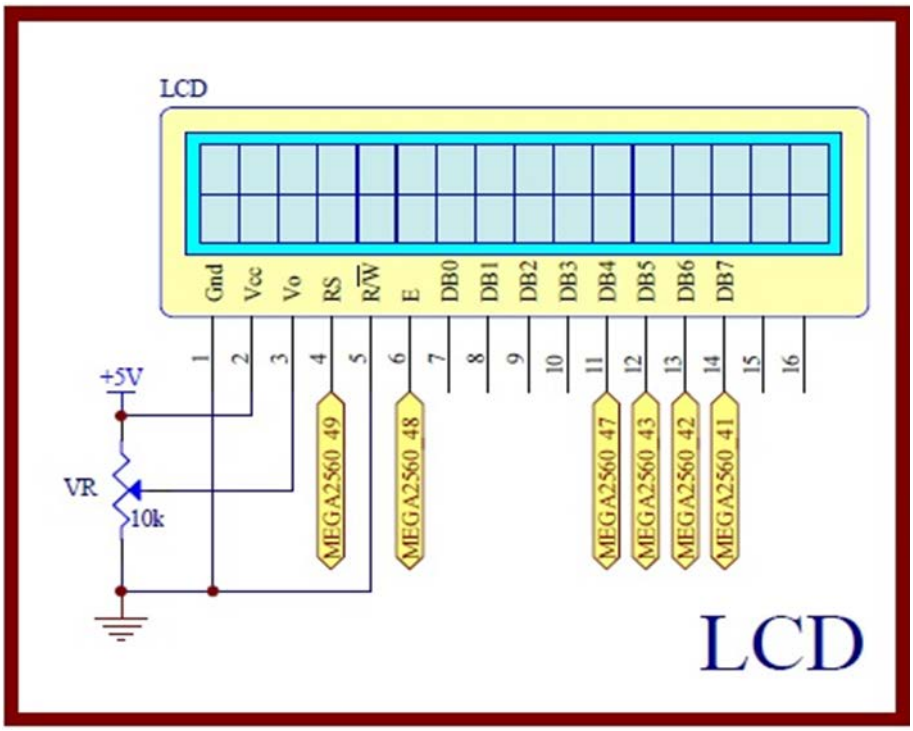


圖 3-9-5 Arduino MEGA2560 對應 LCD 顯示模組的實體接腳電路圖

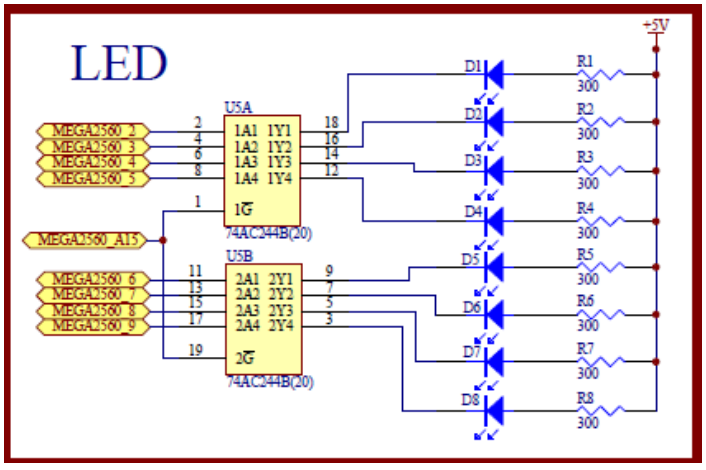


圖 3-9-6 Arduino MEGA2560 對應 LED 的實體接腳電路圖

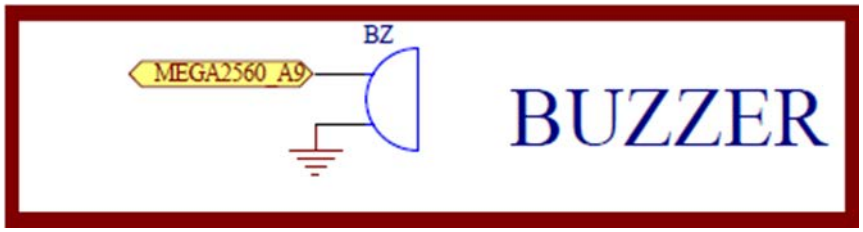


圖 3-9-7 Arduino MEGA2560 對應可變電阻類比轉數位的實體接腳電路圖

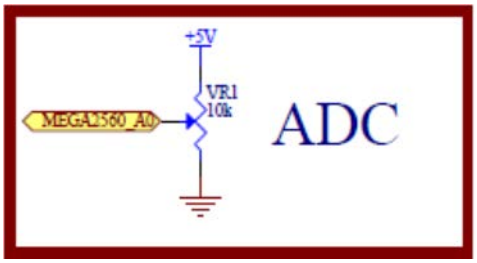


圖 3-9-8 Arduino MEGA2560 對應可變電阻類比轉數位的實體接腳電路圖

鮑率改為 115200

圖 3-9-9 串列埠偵錯助手軟體操作畫面

元件：本實習所需元件如表 3-9-1 所示。

表 3-9-1 元件表

編號	元件項目	數量	元件名稱
1	Arduino MEGA2560	1	Arduino 開發板
2	RF1601	2	ZigBee 模組
3	LCD	1	16×2 文字型 LCD 顯示器
4	VR	1	10kΩ 可變電阻
5	BUZZER	1	5V 電磁式有源式蜂鳴器
6	74AC244	1	8 個緩衝閘晶片
7	D1~D8	8	單色 LED
8	R1	8	300 Ω電阻
9	VR1	1	10KΩ可變電阻

程式（一）：PC 與 Arduino 之 ZigBee 通訊

P3-9-1

行號	程式敘述	註解
1	#include <LiquidCrystal.h>	加入 LCD 顯示模組的驅動函式庫
2	LiquidCrystal lcd(49, 48, 47, 43, 42, 41);	定義 LCD 物件對應 Arduino 的腳位 lcd(RS,E,D4,D5,D6,D7)
3	const int buzzer = A9;	定義 BUZZER 接腳在類比 IO 第 9 支
4	void setup() {	只會執行一次的程式初始設定函式
5	Serial.begin(9600);	定義序列埠監控視窗傳遞速率為 9600
6	Serial2.begin(115200);	定義DB9_2 傳遞速率為115200
7	pinMode(A15,OUTPUT);	規劃 A15 腳為輸出模式
8	digitalWrite(A15, LOW);	A15 輸出 LOW，致能 74AC244
9	pinMode(buzzer,OUTPUT);	規劃 buzzer 腳為輸出模式
10	for (int i = 2; i < 10; i ++) {	for 迴圈 23456789
11	pinMode(i, OUTPUT);	輸出 LOW 訊號到 23456789 腳
12	digitalWrite(i, HIGH);	輸出 HIGH 訊號到 23456789 腳,熄滅 LED
13	}	for loop 結束
14	lcd.begin(16, 2);	初始化 LCD 物件的格式為 16 字、2 行
15	lcd.clear();	清除 LCD 螢幕
16	lcd.setCursor(0,0);	游標設到 LCD 第 1 字、第 1 行
17	lcd.print("=Arduino ZigBee=");	LCD 顯示出字串"=Arduino RFID="
18	}	setup()函式結束
19	void loop(){	loop()函式開始
20	int ZIGBEE_sb;	宣告 ZIGBEE_sb 為整數
21	int ZIGBEE_serInIndx=0;	宣告整數變數
22	char ZIGBEE_char;	宣告 ZIGBEE_char 為字元
23	char ZIGBEE_TX_DATA[25]="Hua Heng Arduino V3 \n";	ZIGBEE 傳送暫存陣列
24	Serial2.write((byte*) ZIGBEE_TX_DATA, 25);	傳送字串資料到 PC 端
25	if(Serial2.available()>0) {	判斷從 PC 端讀取是否有資料
26	lcd.setCursor(0, 1);	游標設到 LCD 第 1 字、第 2 行
27	lcd.print(" ");	LCD 字串清空
28	lcd.setCursor(0, 1);	游標設到 LCD 第 1 字、第 2 行

29	while (Serial2.available()){	持續讀取字串,直到 0 為無資料
30	ZIGBEE_sb = Serial2.read();	讀取資料
31	ZIGBEE_char=ZIGBEE_sb;	改成字元
32	lcd.print(ZIGBEE_char);	LCD 輸出
33	ZIGBEE_serInIndx++;	字數累加
34	if (ZIGBEE_serInIndx>15){	判斷字數大於 15
35	lcd.setCursor(0, 1);	游標設到 LCD 第 1 字、第 2 行
36	}	if 結束
37	}	for loop 結束
38	for (int i = 2; i < 10; i ++) {	for 迴圈 23456789
39	digitalWrite(i, LOW);	輸出 LOW 訊號到 23456789 腳,點亮 LED
40	}	for loop 結束
41	digitalWrite(buzzer, HIGH);	輸出 HIGH 訊號,蜂鳴器響
42	delay(100);	延遲 100ms
43	digitalWrite(buzzer, LOW);	輸出 LOW 訊號,蜂鳴器滅
44	for (int i = 2; i < 10; i ++) {	for 迴圈 23456789
45	digitalWrite(i, HIGH);	輸出 HIGH 訊號到 23456789 腳,熄滅 LED
46	}	for loop 結束
47	}	if 結束
48	delay(300);	延遲 300ms，更新頻率
49	}	loop()函式結束

程式（二）：PC 端接收可變電阻量測值

P3-9-2

行號	程式敘述	註解
1	#define VR_Pin　A0	定義可變電阻輸入腳位 A0
2	void setup(){	只會執行一次的程式初始設定函式
3	Serial2.begin(115200);	//定義DB9_2 傳遞速率為115200
4	}	setup()函式結束
5	void loop (){	loop()函式開始
6	int VR_Value;	宣告整數變數
7	VR_Value=analogRead(VR_Pin);	讀取可變電阻的類比值
8	Serial2.println(VR_Value);	讀取值輸出到 PC 端
9	delay(500);	延遲 500ms，更新頻率
10	}	loop()函式結束

練習

三、 設計一系統可以感測超音波距離，然後透過 ZigBee 無線傳輸介面傳送到 PC 端顯示。

實驗 3-10：GY-80 模組感測器功能實習

目的：瞭解 GY-80 模組的工作原理及使用 Arduino 程式控制方法。

功能：本實習使用 Arduino MEGA2560 各別讀取 GY-80 模組內四種感測器的值並顯示在 LCM 上。程式(一)使用 Arduino MEGA2560 藉由 I2C 介面讀取三軸加速度感測器 ADXL345 的 G 值並顯示在 LCD 上。程式(二)使用 Arduino MEGA2560 藉由 I2C 介面讀取三軸陀螺儀 L3G4200D 的角速度值與溫度並顯示在 LCD 上。程式(三) 使用 Arduino MEGA2560 藉由 I2C 介面讀取三軸電子羅盤 HMC5883L 的磁感應數值與計算出指的方向並顯示在 LCM 上。程式(四)使用 Arduino MEGA2560 藉由 I2C 介面讀取 BMP085 模組的氣壓與溫度感測器的感測數值並顯示在 LCM 上。

原理：GY-80 模組感測器整合四顆感測器而成，包含重力加速度計(ADXL345)、陀螺儀(L3G4200D)、電子羅盤(HMC5883L)及氣壓計(BMP085)，如圖 3-10-1 所示，模組接腳主要有提供 I2C 介面兩支接腳 SCL 及 SDA，剩下是狀態腳和中斷腳。現就四種感測器分別介紹如下：

ADXL345 模組：

ADXL345 模組是一款三軸重力加速度計感測器(G-Sensor)，其原理即是偵測這三維空間的變動，亦即偵測 G 力的大小及動作方向（如圖 3-10-2 所示）。在靜止平放的狀態下，Z 軸正方向朝上，重力角速度方向朝下，而此時感測器的 Z 軸讀數卻是“正數”，會有 1G 的地心引力會在 Z 軸，隨著改變裝置的同時 G 值會在不同軸向做變化，就可以知道姿態的變換。ADXL345 模組是一款超低功耗 3 軸加速度感測器，分辨率可達 13 位，測量範圍達±16G，可使用 SPI(3 線或 4 線)或 I2C 介面讀取 G 值，三軸 G 值輸出數據為 16 位二進制格式。加速度感測器始終能測量到重力加速度，所以在靜止狀態下，根據重力加速度在 x、y、z 三軸的分量，可以測得感測器當前的傾斜角度。所以 ADXL345 常被用作傾斜感測器來使用，非常適合移動設備應用，還可以測量運動或衝擊導致的動態加速度。

L3G4200D 模組：

L3G4200D 模組是一款三軸陀螺儀感測器，可以分別計算出物體在 Roll、Pitch 及 Yaw 的角速度數值（如圖 3-10-3 所示），根據物體的旋轉方向，可以讀出正/負的數值。比方說，平放在桌面上，Z 軸正方向朝上，根據右手法則，將模組逆時針旋轉的時候，Z 軸角速度讀數為正數，若順時針旋轉，讀數為負數。此模組特性包括：(1)可選擇三種解析範圍(250/500/2000 dps)。(2)具有 I2C/SPI 數位輸出介面。(3) 角速度量測值是 16 位元數值資料輸出。(4)內建 8-bits 溫度

感測器，可以單獨輸出，並作為溫度補償。(5)具有相當大範圍的電壓輸入(2.4 V ~ 3.6 V)。(6)相容於低電壓的 IO (1.8 V)。(7)內建 power-down 及 sleep 模式。(8)高抗撞擊能力。

HMC5883L 模組：

HMC5883L 模組是一款三軸電子羅盤感測器，俗稱電子指南針，用來測量周圍的磁感應強度，一般用途主要測量物體靜止時候的地磁方向，使用上要注意測量地磁時候容易受到周圍磁場影響。此模組特性包括：(1)具有 I2CI 數位輸出介面，設計使用非常方便。(2)尺寸小: 3x3x0.9mm LCC 封裝，適合大規模量產使用。(3)精度高，內置 12 位 A/D,OFFSET, SET/RESET 電路，不會出現磁飽和現象，不會有累加誤差。(4)具有自動校準程序，簡化使用步驟，終端產品使用非常方便。(5)內置自測試電路，方便量產測試，無需增加額外昂貴的測試設備。(6)功耗低：供電電壓 1.8V, 功耗睡眠模式耗電流約 2.5uA 測量模式耗電流約 0.6mA。

BMP085 模組：

BMP085 模組是一款高精度、超低能耗的氣壓與溫度感測器，可以應用在移動設備中。此模組特性包括：(1)具有相當大範圍的電壓輸入(1.62 V ~ 3.6V)。(2)壓力量測值是 16~19 位元數值資料輸出，可讀取的壓力值為 300~1100 百帕(hPa)。(3)精確度可達到 0.03 百帕。(4)內建 16-bits 溫度感測器，可以單獨輸出，並作為溫度補償。(5)具有 I2CI 數位輸出介面，設計使用非常方便，通迅速度最快可達 3.4MHz。(6)耗電極低，正常模式只有 5μA。

在使用 BMP085 氣壓與溫度感測器時，必須先做數據的校正與補償，由於海拔高度和大氣壓強的關係受溫度的影響，因此需要用溫度值對氣壓值進行校正與補償。BMP085 模組的 E2PROM 中有原廠自帶的 11 個校準參數，每一個傳感器的校準參數都不同。在第一次讀取氣壓和溫度值之前，必須先讀取 E2PROM 中的校準參數，再從指定的暫存器中將未經校正的溫度和氣壓值讀取出來，然後採用 BMP085 數據手冊提供的校正算法對溫度值和氣壓值進行補償。該算法中，需要根據所設置的過取樣參數 OSS(Oversampling Setting)的值來選擇 BMP085 的工作模式，OSS 的值決定了測量精度和轉換時間。

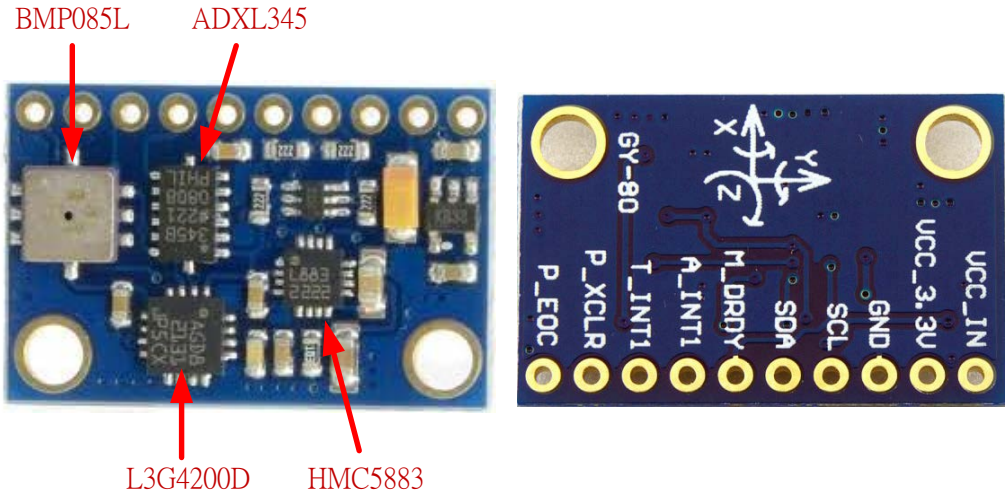


圖 3-10-1　GY-80 模組實體圖

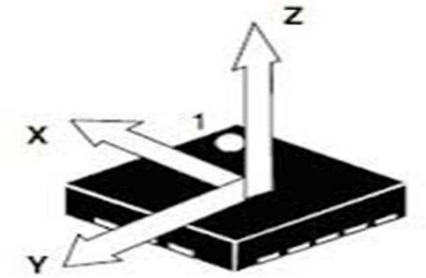


圖 3-10-2　三軸加速度值座標示意圖

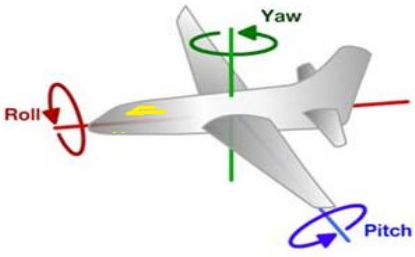


圖 3-10-3　三軸角速度值座標示意圖

電路： GY-80 模組與 Arduino MEGA2560 開發板的實體接圖如圖 3-10-4 所示，圖 3-10-5 為實體接腳電路圖，圖中 Arduino MEGA2560 開發板上的第 20 支(SDA)與 21 支(SCL)數位接腳對應連接到 GY-80 模組 SDA 與 SCL 接腳。程式(一)中會使用 Arduino MEGA2560 藉由 I2C 介面控制 GY-80 讀取 ADXL345 晶片三軸 G 值並用 Serial.print 顯示出來及顯示在 LCM 上(圖 3-10-6)。另外提供 processing 程式可以 3D 展示三軸 G 值變化情況。使用

方法是先開啟 processing 軟體後打開書上所附的 P3_10_1_processing.pde 程式，將 setup()函式內第二行指令 sp = new Serial(this, "COM17", 9600); 中"COM17"改成與串列埠觀測視窗同一 com 編號即可。執行展示畫面如圖 3-10-7 所示。程式(二) 中會使用 Arduino MEGA2560 藉由 I2C 介面控制 GY-80 讀取 L3G4200D 晶片三軸 Roll、Pitch 及 Yaw 的角速度數值與溫度並用 Serial.print 顯示出來及顯示在 LCM 上。程式(三)中會使用 Arduino MEGA2560 藉由 I2C 介面控制 GY-80 讀取 HMC5883L 晶片三軸磁感應數值與計算出電子指南針指的方向並用 Serial.print 顯示出來及顯示在 LCM 上。程式(四)中會使用 Arduino MEGA2560 藉由 I2C 介面控制 GY-80 讀取 BMP085 晶片三溫度&大氣壓力，並換算成高度，並用 Serial.print 顯示出來及顯示在 LCM 上。程式上必須先執行 bmp085_Calibration 指令進行 BMP085 氣壓感測器的 E2PROM 中有原廠自帶的 11 個校準參數，接著再執行 bmp085GetTemperature 指令會先計算出 b5 這個變數，然後 b5 又會被用在 bmp085GetPressure 指令中校正壓力、高度，所以 bmp085GetTemperature 指令必須在 bmp085GetPressure 指令之前先執行。

圖 3-10-4 Arduino MEGA2560 對 GY-80 模組的實體接圖

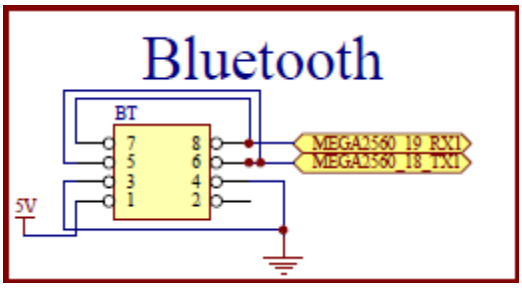


圖 3-10-5 Arduino MEGA2560 對 GY-80 模組的實體接腳電路圖

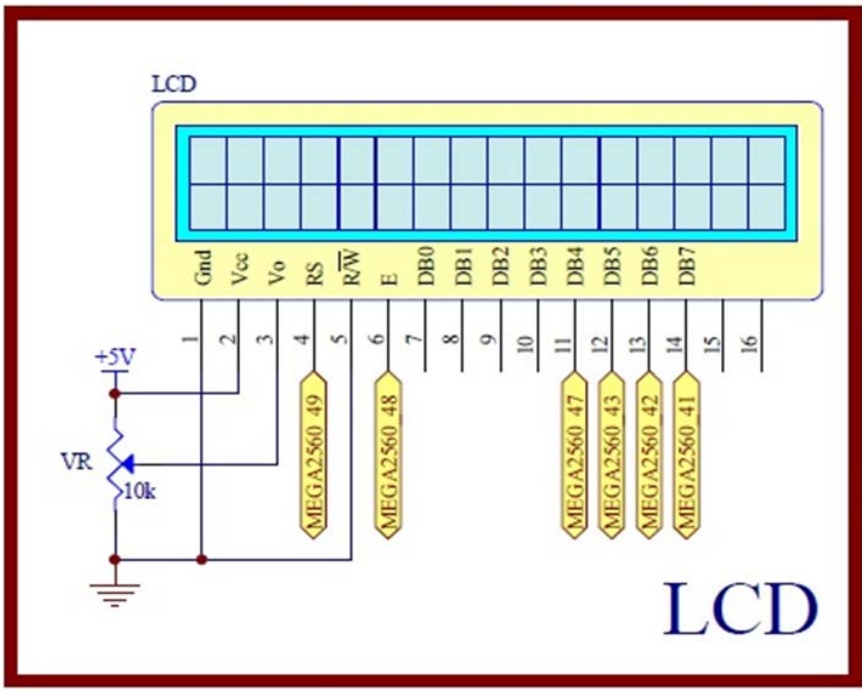


圖 3-10-6 Arduino MEGA2560 對應 LCD 顯示模組的實體接腳電路圖

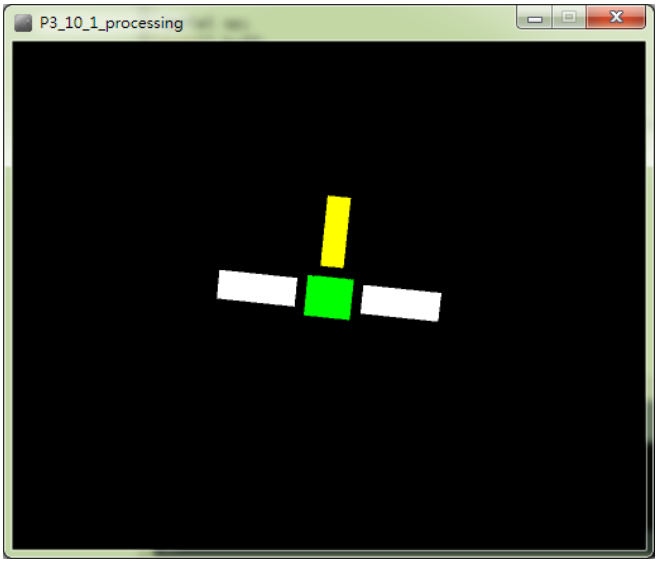


圖 3-10-7 使用 processing 展示 ADXL345 三軸 G 值之 3D 圖

元件：本實習所需元件如表 3-10-1 所示。

表 3-10-1 元件表

編號	元件項目	數量	元件名稱
1	Arduino MEGA2560	1	Arduino 開發板
2	GY-80	1	9 軸+1 感測器模組
3	LCD	1	16×2 文字型 LCD 顯示器
4	VR	2	10kΩ 可變電阻

程式（一）：讀取 ADXL345 三軸 G 值

P3-10-1

行號	程式敘述	註解
1	#include <LiquidCrystal.h>	加入 LCD 顯示器驅動函式庫
2	#include <Wire.h>	加入 I2C 通訊介面函式庫
3	#define ADXAddress 0x53	ADXL345 的 I2C 地址
4	#define Register_16G 0x31	資料格式設定之暫存器位址
5	#define Register_Power 0x2D	相關電源設定之暫存器位址
6	#define Register_X0 0x32	X 軸低序位元組資料暫存器位址
7	#define Register_X1 0x33	X 軸高序位元組資料暫存器位址
8	#define Register_Y0 0x34	Y 軸低序位元組資料暫存器位址
9	#define Register_Y1 0x35	Y 軸高序位元組資料暫存器位址
10	#define Register_Z0 0x36	Z 軸低序位元組資料暫存器位址
11	#define Register_Z1 0x37	Z 軸高序位元組資料暫存器位址
12	LiquidCrystal lcd(49, 48, 47, 43, 42, 41);	初始化 LCD，建立物件 lcd，指定腳位。 順序是 RS、Enable、D4、D5、D6、D7， 要配合前面的接線順序
13	int Xout,Yout,Zout;	全域變數宣告成整數
14	float Xg,Yg,Zg;	全域變數宣告成浮點數
15	void setup(){	只會執行一次的程式初始函式
16	Serial.begin(9600);	將串列埠通訊鮑率設為 9600bps
17	lcd.begin(16, 2);	定義 LCD 顯示器為 16x2 文字型
18	Wire.begin();	初始化 I2C
19	setReg(ADXAddress,Register_16G,0x0B);	測量範圍,正負 16g，13 位元模式
20	setReg(ADXAddress, Register_Power,0x08);	選擇電源模式為測量模式
21	}	setup()函式結束
22	void loop(){	永遠周而復始的主控制函式
23	byte MSB, LSB;	區域變數宣告成位元組
24	LSB = getData(ADXAddress,Register_X0);	取得 X 軸低位元資料
25	MSB = getData(ADXAddress,Register_X1);	取得 X 軸高位元資料

26	Xout = ((MSB << 8) | LSB) ;	高位元資料往左移 8 位元，組合成 16 位元數值
27	Xg=Xout/256.00;	把 X 軸 G 值輸出結果轉換為重力加速度 G,精確到小數點後 2 位
28	LSB = getData(ADXAddress,Register_Y0);	取得 Y 軸低位元資料
29	MSB = getData(ADXAddress,Register_Y1);	取得 Y 軸高位元資料
30	Yout = ((MSB << 8) | LSB) ;	高位元資料往左移 8 位元，組合成 16 位元數值
31	Yg=Yout/256.00;	把 Y 軸 G 值輸出結果轉換為重力加速度 G,精確到小數點後 2 位
32	LSB = getData(ADXAddress,Register_Z0);	取得 Z 軸低位元資料
33	MSB = getData(ADXAddress,Register_Z1);	取得 Y 軸高位元資料
34	Zout = ((MSB << 8) | LSB) ;	高位元資料往左移 8 位元，組合成 16 位元數值
35	Zg=Zout/256.00;	把 Z 軸 G 值輸出結果轉換為重力加速度 G,精確到小數點後 2 位
36	Serial.print(Xout);	Xout 輸出至序列埠觀測視窗
37	Serial.print(" ");	空格輸出至序列埠觀測視窗
38	Serial.print(Yout);	Yout 輸出至序列埠觀測視窗
39	Serial.print(" ");	空格輸出至序列埠觀測視窗
40	Serial.print(Zout);	Zout 輸出至序列埠觀測視窗
41	Serial.print("\n ");	序列埠觀測視游標跳到下一行
42	lcd.setCursor(0, 0);	將游標移動第 1 欄、第 1 列
43	lcd.clear();	清除 LCD 螢幕
44	lcd.print("Xg=");	使 LCM 顯示文字"Xg="
45	lcd.print(Xg);	使 LCM 顯示 Xg 值
46	lcd.setCursor(8, 0);	將游標移動第 9 欄、第 1 列
47	lcd.print("Yg=");	使 LCM 顯示文字"Yg="
48	lcd.print(Yg);	使 LCM 顯示 Yg 值
49	lcd.setCursor(0, 1);	將游標移動第 1 欄、第 2 列
50	lcd.print("Zg=");	使 LCM 顯示文字"Zg="
51	lcd.print(Zg);	使 LCM 顯示 Zg 值
52	delay(30);	延遲 0.03 秒，更新頻率
53	}	loop 函式結束

60	void setReg(int device_address, unsigned char reg_address, unsigned char data){	寫入暫存器函式開始
61	Wire.beginTransmission(device_address);	開始指定位址的裝置傳輸
62	Wire.write(reg_address);	送出暫存器位址
63	Wire.write(data);	寫入資料到暫存器
64	Wire.endTransmission();	結束傳輸
65	}	函式結束
66	int getData(int device_address, unsigned char reg_address){	讀取暫存器裡的資料函式開始
67	Wire.beginTransmission(device_address);	開始指定位址的裝置傳輸
68	Wire.write(reg_address);	送出暫存器位址
69	Wire.endTransmission();	結束傳輸
70	Wire.requestFrom(device_address,1);	請求讀取
71	if(Wire.available()<=1){	判別接收的資料數量小於等於 1
72	return Wire.read();	回傳一位元組資料
73	}	if 結束
74	}	函式結束

程式（二）：讀取 L3G4200D 三軸角速度值與溫度

P3-10-2

行號	程式敘述	註解
1	#include <LiquidCrystal.h>	加入 LCD 顯示器驅動函式庫
2	#include <Wire.h>	加入 I2C 通訊介面函式庫
3	#define L3G4200D_Address 0x69	L3G4200D 的 I2C 位址
4	#define CTRL_REG1 0x20	L3G4200D 的 CTRL_REG1 位址
5	#define CTRL_REG2 0x21	L3G4200D 的 CTRL_REG2 位址
6	#define CTRL_REG3	L3G4200D 的 CTRL_REG3 位址

	0x22	
7	#define CTRL_REG4 0x23	L3G4200D 的 CTRL_REG4 位址
8	#define CTRL_REG5 0x24	L3G4200D 的 CTRL_REG5 位址
9	#define Register_TEMPERATURE 0x26	溫度位元組資料暫存器位址
10	#define Register_X0 0x28	X 軸低序位元組資料暫存器位址
11	#define Register_X1 0x29	X 軸高序位元組資料暫存器位址
12	#define Register_Y0 0x2A	Y 軸低序位元組資料暫存器位址
13	#define Register_Y1 0x2B	Y 軸高序位元組資料暫存器位址
14	#define Register_Z0 0x2C	Z 軸低序位元組資料暫存器位址
15	#define Register_Z1 0x2D	Z 軸高序位元組資料暫存器位址
16	LiquidCrystal lcd(49, 48, 47, 43, 42, 41);	初始化 LCD，建立物件 lcd，指定腳位。 順序是 RS、Enable、D4、D5、D6、D7， 要配合前面的接線順序
17	int T, X, Y, Z;	全域變數宣告成整數
18	void setup(){	只會執行一次的程式初始函式
19	Wire.begin();	初始化 I2C
20	Serial.begin(9600);	將串列埠通訊鮑率設為 9600bps
21	lcd.begin(16, 2);	定義 LCD 顯示器為 16x2 文字型
22	setReg(L3G4200D_Address,CTRL_REG1, 0x0F);	三軸致能,power down 禁能
23	setReg(L3G4200D_Address,CTRL_REG3, 0x08);	致能 data ready
24	setReg(L3G4200D_Address,CTRL_REG4, 0x80);	設定角速度解析度為 500 度
25	delay(100);	延遲 100ms
26	}	setup()函式結束
27	void loop(){	永遠周而復始的主控制函式
28	getGyroValues();	取得測量值
29	Serial.print(" Roll(X):");	Roll(X):輸出至序列埠觀測視窗

30	Serial.print(X);	X 輸出至序列埠觀測視窗
31	Serial.print(" Pitch(Y):");	Pitch(Y):輸出至序列埠觀測視窗
32	Serial.print(Y);	Y 輸出至序列埠觀測視窗
33	Serial.print(" Yaw(Z):");	Yaw(Z):輸出至序列埠觀測視窗
34	Serial.print(Z);	Z 輸出至序列埠觀測視窗
35	Serial.print(" Degrees C:");	Degrees C:輸出至序列埠觀測視窗
36	Serial.println(T);	T 輸出至序列埠觀測視窗
37	lcd.setCursor(0, 0);	將游標移動第 1 欄、第 1 列
38	lcd.clear();	清除 LCD 螢幕
39	lcd.print("X=");	使 LCM 顯示文字"X="
40	lcd.print(X);	使 LCM 顯示 Xg 值
41	lcd.setCursor(8, 0);	將游標移動第 9 欄、第 1 列
42	lcd.print("Y=");	使 LCM 顯示文字"Y="
43	lcd.print(Y);	使 LCM 顯示 Y 值
44	lcd.setCursor(0, 1);	將游標移動第 1 欄、第 2 列
45	lcd.print("Z=");	使 LCM 顯示文字"Z="
46	lcd.print(Z);	使 LCM 顯示 Z 值
47	lcd.setCursor(8, 1);	將游標移動第 8 欄、第 2 列
48	lcd.print("T=");	使 LCM 顯示文字"T="
49	lcd.print(T);	使 LCM 顯示 T 值
50	delay(300);	延遲 0.03 秒，更新頻率
51	}	loop 函式結束
52	void getGyroValues () {	取得測量值函式
53	byte MSB, LSB;	區域變數宣告成位元組
54	T = getData(L3G4200D_Address,Register_TEMPERATURE);	取得溫度位元資料
55	LSB = getData(L3G4200D_Address,Register_X0);	取得 X 軸低位元資料
56	MSB = getData(L3G4200D_Address,Register_X1);	取得 X 軸高位元資料
57	X = ((MSB << 8) | LSB) ;	高位元資料往左移 8 位元，組合成 16 位元數值
58	LSB =	取得 Y 軸低位元資料

	getData(L3G4200D_Address,Register_Y0);	
59	MSB = getData(L3G4200D_Address,Register_Y1);	取得 Y 軸高位元資料
60	Y = ((MSB << 8) | LSB);	高位元資料往左移 8 位元，組合成 16 位元數值
61	LSB = getData(L3G4200D_Address,Register_Z0);	取得 Z 軸低位元資料
62	MSB = getData(L3G4200D_Address,Register_Z1);	取得 Z 軸高位元資料
63	Z = ((MSB << 8) | LSB) ;	高位元資料往左移 8 位元，組合成 16 位元數值
64	}	函式結束
65	void setReg(int device_address, unsigned char reg_address, unsigned char data){	寫入暫存器函式
66	Wire.beginTransmission(device_address);	開始指定位址的裝置傳輸
67	Wire.write(reg_address);	送出暫存器位址
68	Wire.write(data);	寫入資料到暫存器
69	Wire.endTransmission();	結束傳輸
70	}	函式結束
71	int getData(int device_address, unsigned char reg_address){	取得暫存器裡的資料函式
72	Wire.beginTransmission(device_address);	開始指定位址的裝置傳輸
73	Wire.write(reg_address);	送出暫存器位址
74	Wire.endTransmission();	結束傳輸
75	Wire.requestFrom(device_address,1);	請求讀取
76	if(Wire.available()<=1){	判別接收的資料數量小於等於 1
77	return Wire.read();	回傳一位元組資料
78	}	if 結束

79	}	函式結束

程式（三）：讀取 HMC5883L 三軸磁感應值與計算出電子指南針指的方向

P3-10-3

行號	程式敘述	註解
1	#include <LiquidCrystal.h>	加入 LCD 顯示器驅動函式庫
2	#include <Wire.h>	加入 I2C 通訊介面函式庫
3	#include <math.h>	加入數學運算函式庫
4	#define HMC5883_Address 0x1E	HMC5883 的 I2C 位址
5	#define Register_Mode 0x02	模式設定之暫存器位址
6	#define Continuous_mode 0x00	工作模式為連續測量模式指令
7	#define CTRL_REG_B 0x01	HMC5883 的的 CTRL_REG_B 位址
8	#define CTRL_REG_B_Data 0x80	HMC5883 測量值範圍-4 高斯~+4 高斯
9	#define Register_X0 0x04	X 軸低序位元組資料暫存器位址
10	#define Register_X1 0x03	X 軸高序位元組資料暫存器位址
11	#define Register_Y0 0x08	Y 軸低序位元組資料暫存器位址
12	#define Register_Y1 0x07	Y 軸高序位元組資料暫存器位址
13	#define Register_Z0 0x06	Z 軸低序位元組資料暫存器位址
14	#define Register_Z1 0x05	Z 軸高序位元組資料暫存器位址
15	LiquidCrystal lcd(49, 48, 47, 43, 42, 41);	初始化 LCD，建立物件 lcd，指定腳位。 順序是 RS、Enable、D4、D5、D6、D7， 要配合前面的接線順序
16	int X, Y, Z;	全域變數宣告成整數
17	void setup(){	只會執行一次的程式初始函式
18	Serial.begin(9600);	將串列埠通訊鮑率設為 9600bps
19	lcd.begin(16, 2);	定義 LCD 顯示器為 16x2 文字型
20	Wire.begin();	初始化 I2C
21	setReg(HMC5883_Address,Register_Mode,Continuous_mode);	選擇測量模式為連續測量模式
22	setReg(HMC5883_Address, CTRL_REG_B,CTRL_REG_B_Data);	測量範圍,正負 4 高斯
23	}	setup()函式結束
24	void loop() {	永遠周而復始的主控制函式

25	double angle;	區域變數宣告成倍精數
26	getValues();	取得測量值
27	Serial.print(" X:");	" X:"輸出至序列埠觀測視窗
28	Serial.print(X);	X 輸出至序列埠觀測視窗
29	Serial.print(" Y:");	" Y:"輸出至序列埠觀測視窗
30	Serial.print(Y);	Y 輸出至序列埠觀測視窗
31	Serial.print(" Z:");	" Z:"輸出至序列埠觀測視窗
32	Serial.println(Z);	Z 輸出至序列埠觀測視窗
33	lcd.setCursor(0, 0);	將游標移動第 1 欄、第 1 列
34	lcd.clear();	清除 LCD 螢幕
35	lcd.print("X=");	使 LCM 顯示文字"X="
36	lcd.print(X);	使 LCM 顯示 Xg 值
37	lcd.setCursor(8, 0);	將游標移動第 9 欄、第 1 列
38	lcd.print("Y=");	使 LCM 顯示文字"Y="
39	lcd.print(Y);	使 LCM 顯示 Y 值
40	lcd.setCursor(0, 1);	將游標移動第 1 欄、第 2 列
41	lcd.print("Z=");	使 LCM 顯示文字"Z="
42	lcd.print(Z);	使 LCM 顯示 Z 值
43	lcd.setCursor(8, 1);	將游標移動第 8 欄、第 2 列
44	angle=atan2((double)Y,(double)X)*(180/3.14159265)+180;	計算指南針角度
45	Serial.print("You are heading: ");	" You are heading: "輸出至序列埠觀測視窗
46	if((angle < 22.5) || (angle > 337.5)){	判斷角度範圍
47	Serial.println("South");	" You are heading: "輸出至序列埠觀測視窗
48	lcd.print("South");	使 LCM 顯示文字"South"
49	}	
50	else if((angle > 22.5) && (angle < 67.5)){	判斷角度範圍
51	Serial.println("South-West");	" You are heading: "輸出至序列埠觀測視窗
52	lcd.print("SW");	使 LCM 顯示文字"SW"
53	}	
54	else if((angle > 67.5) && (angle < 112.5)){	判斷角度範圍
55	Serial.println("West");	" You are heading: "輸出至序列埠觀測視窗

56	lcd.print("West");	使 LCM 顯示文字"West"
57	}	
58	else if((angle > 112.5) && (angle < 157.5)){	判斷角度範圍
59	Serial.println("North-West");	" You are heading: "輸出至序列埠觀測視窗
60	lcd.print("NW");	使 LCM 顯示文字"NW"
61	}	
62	if((angle > 157.5) && (angle < 202.5)){	判斷角度範圍
63	Serial.println("North");	" You are heading: "輸出至序列埠觀測視窗
64	lcd.print("North");	使 LCM 顯示文字"North"
65	}	
66	if((angle > 202.5) && (angle < 247.5)){	判斷角度範圍
67	Serial.println("NorthEast");	" You are heading: "輸出至序列埠觀測視窗
68	lcd.print("NE");	使 LCM 顯示文字"NE"
69	}	
70	if((angle > 247.5) && (angle < 292.5)){	判斷角度範圍
71	Serial.print("East");	" You are heading: "輸出至序列埠觀測視窗
72	lcd.print("East");	使 LCM 顯示文字"East"
73	}	
74	if((angle > 292.5) && (angle < 337.5)){	判斷角度範圍
75	Serial.println("SouthEast");	" You are heading: "輸出至序列埠觀測視窗
76	lcd.print("SouthEast");	使 LCM 顯示文字"SouthEast"
77	}	
78	delay(500);	延遲 0.05 秒，更新頻率
79	}	
80	void getValues () {	取得測量值函式
81	byte MSB, LSB;	區域變數宣告成位元組
82	LSB = getData(HMC5883_Address,Register_X0);	取得 X 軸低位元資料
83	MSB =	取得 X 軸高位元資料

	getData(HMC5883_Address,Register_X1);	
84	X = ((MSB << 8) | LSB) ;	高位元資料往左移 8 位元，組合成 16 位元數值
85	LSB = getData(HMC5883_Address,Register_Y0);	取得 Y 軸低位元資料
86	MSB = getData(HMC5883_Address,Register_Y1);	取得 Y 軸高位元資料
87	Y = ((MSB << 8) | LSB);	高位元資料往左移 8 位元，組合成 16 位元數值
88	LSB = getData(HMC5883_Address,Register_Z0);	取得 Z 軸低位元資料
89	MSB = getData(HMC5883_Address,Register_Z1);	取得 Z 軸高位元資料
90	Z = ((MSB << 8) | LSB) ;	高位元資料往左移 8 位元，組合成 16 位元數值
91	}	函式結束
92	void setReg(int device_address, unsigned char reg_address, unsigned char data){	寫入暫存器函式
93	Wire.beginTransmission(device_address);	開始指定位址的裝置傳輸
94	Wire.write(reg_address);	送出暫存器位址
95	Wire.write(data);	寫入資料到暫存器
96	Wire.endTransmission();	結束傳輸
97	}	函式結束
98	int getData(int device_address, unsigned char reg_address){	取得暫存器裡的資料函式
99	Wire.beginTransmission(device_address);	開始指定位址的裝置傳輸
100	Wire.write(reg_address);	送出暫存器位址
101	Wire.endTransmission();	結束傳輸
102		請求讀取

	Wire.requestFrom(device_address,1);	
103	if(Wire.available()<=1){	判別接收的資料數量小於等於 1
104	return Wire.read();	回傳一位元組資料
105	}	if 結束
106	}	函式結束

程式（四）：讀取 BMP085 氣壓值與溫度

行號	程式敘述	註解
1	#include <LiquidCrystal.h>	加入 LCD 顯示器驅動函式庫
2	#include <Wire.h>	加入 I2C 通訊介面函式庫
3	#define BMP085_ADDRESS 0x77	BMP085 的 I2C 位址
4	#define Register_CMD 0xF4	指令暫存器位址
5	#define TEMPERATURE_CMD 0x2E	溫度讀取指令
6	#define Register_D0 0xF8	XLSB 資料暫存器位址
7	#define Register_D1 0xF7	LSB 資料暫存器位址
8	#define Register_D2 0xF6	MSB 資料暫存器位址
9	const unsigned char OSS = 0;	工作模式選擇低功耗工作模式
10	int ac1;	全域變數宣告成整數，定義校準值
11	int ac2;	全域變數宣告成整數，定義校準值
12	int ac3;	全域變數宣告成整數，定義校準值
13	unsigned int ac4;	全域變數宣告成無符號位元整數，定義校準值
14	unsigned int ac5;	全域變數宣告成無符號位元整數，定義校準值
15	unsigned int ac6;	全域變數宣告成無符號位元整數，定義校準值
16	int b1;	全域變數宣告成整數，定義校準值
17	int b2;	全域變數宣告成整數，定義校準值
18	int mb;	全域變數宣告成整數，定義校準值
19	int mc;	全域變數宣告成整數，定義校準值
20	int md;	全域變數宣告成整數，定義校準值

21	long b5;	全域變數宣告成長整數，定義校準值
22	LiquidCrystal lcd(49, 48, 47, 43, 42, 41);	初始化 LCD，建立物件 lcd，指定腳位。 順序是 RS、Enable、D4、D5、D6、D7， 要配合前面的接線順序
23	void setup(){	只會執行一次的程式初始函式
24	Wire.begin();	初始化 I2C
25	Serial.begin(9600);	將串列埠通訊鮑率設為 9600bps
26	lcd.begin(16, 2);	定義 LCD 顯示器為 16x2 文字型
27	bmp085_Calibration();	校準參數
28	}	setup()函式結束
29	void loop(){	永遠周而復始的主控制函式
30	float temperature = bmp085GetTemperature(bmp085ReadUT());	先取出溫度的量測值(攝氏)
31	float pressure = bmp085GetPressure(bmp085ReadUP());	再叫出氣壓的量測值(帕斯卡)
32	float atm = pressure / 101325;	將氣壓帕斯卡(pa)換算成氣壓單位(atm)->1atm = 101325 Pa
33	float altitude = calcAltitude(pressure);	最後再取出高度的的量測值(公尺)
34	Serial.print(" Temperature: ");	Temperature:輸出至序列埠觀測視窗
35	Serial.print(temperature, 2);	顯示小數位下兩位
36	Serial.println("deg C");	deg C 輸出至序列埠觀測視窗
37	Serial.print(" Pressure: ");	Pressure:輸出至序列埠觀測視窗
38	Serial.print(pressure, 0);	0 代表顯示整數
39	Serial.println (" Pa");	Pa 輸出至序列埠觀測視窗
40	Serial.print(" Standard Atmosphere: ");	Standard Atmosphere:輸出至序列埠觀測視窗
41	Serial.print(atm, 4);	顯示小數位下四位
42	Serial.println (" atm");	atm 輸出至序列埠觀測視窗
43	Serial.print(" Altitude: ");	Altitude:輸出至序列埠觀測視窗
44	Serial.print(altitude, 2);	顯示小數位下兩位
45	Serial.println (" M");	M 輸出至序列埠觀測視窗
46	Serial.println ();	換行
47	lcd.setCursor(0, 0);	將游標移動第 1 欄、第 1 列
48	lcd.clear();	清除 LCD 螢幕

49	lcd.print("T=");	使 LCM 顯示文字"T="
50	lcd.print(temperature,2);	使 LCM 顯示 temperature 值
51	lcd.setCursor(8, 0);	將游標移動第 9 欄、第 1 列
52	lcd.print("Pa=");	使 LCM 顯示文字"Pa="
53	lcd.print(pressure,2);	使 LCM 顯示 pressure 值顯示小數位下兩位
54	lcd.setCursor(0, 1);	將游標移動第 1 欄、第 2 列
55	lcd.print("A=");	使 LCM 顯示文字"A="
56	lcd.print(atm,2);	使 LCM 顯示 atm 值顯示小數位下兩位
57	lcd.setCursor(8, 1);	將游標移動第 8 欄、第 2 列
58	lcd.print("H=");	使 LCM 顯示文字"H="
59	lcd.print(altitude);	使 LCM 顯示 altitude 值
60	delay(1000);	延遲 1 秒，更新頻率
61	}	loop 函式結束
62	void bmp085_Calibration() {	校準溫度與壓力的函式
63	ac1 = bmp085ReadInt(0xAA);	
64	ac2 = bmp085ReadInt(0xAC);	
65	ac3 = bmp085ReadInt(0xAE);	
66	ac4 = bmp085ReadInt(0xB0);	
67	ac5 = bmp085ReadInt(0xB2);	
68	ac6 = bmp085ReadInt(0xB4);	
69	b1 = bmp085ReadInt(0xB6);	
70	b2 = bmp085ReadInt(0xB8);	
71	mb = bmp085ReadInt(0xBA);	
72	mc = bmp085ReadInt(0xBC);	
73	md = bmp085ReadInt(0xBE);	
74	}	
75	float bmp085GetTemperature(unsigned int ut){	計算溫度函式
76	long x1, x2;	
77	x1 = (((long)ut - (long)ac6)*(long)ac5) >> 15;	
78	x2 = ((long)mc << 11)/(x1 + md);	
79	b5 = x1 + x2;	
80	float temp = ((b5 + 8)>>4);	

81	temp = temp /10;	
82	return temp;	
83	}	
84	long bmp085GetPressure(unsigned long up){	計算氣壓函式
85	long x1, x2, x3, b3, b6, p;	
86	unsigned long b4, b7;	
87	b6 = b5 - 4000;	
88	x1 = (b2 * (b6 * b6)>>12)>>11;	
89	x2 = (ac2 * b6)>>11;	
90	x3 = x1 + x2;	
91	b3 = (((((long)ac1)*4 + x3)<<OSS) + 2)>>2;	
92	x1 = (ac3 * b6)>>13;	
93	x2 = (b1 * ((b6 * b6)>>12))>>16;	
94	x3 = ((x1 + x2) + 2)>>2;	
95	b4 = (ac4 * (unsigned long)(x3 + 32768))>>15;	
96	b7 = ((unsigned long)(up - b3) * (50000>>OSS));	
97	if (b7 < 0x80000000)	
98	p = (b7<<1)/b4;	
99	else	
100	p = (b7/b4)<<1;	
101	x1 = (p>>8) * (p>>8);	
102	x1 = (x1 * 3038)>>16;	
103	x2 = (-7357 * p)>>16;	
104	p += (x1 + x2 + 3791)>>4;	
105	long temp = p;	
106	return temp;	
107	}	
108	unsigned int bmp085ReadUT() {	讀取未補償溫度值函式
109	unsigned int ut;	區域變數宣告成無符號位元整數
110	setReg(BMP085_ADDRESS,Register_	下溫度讀取指令

	CMD, TEMPERATURE_CMD);	
111	delay(5);	延遲 5ms
112	ut = bmp085ReadInt(Register_D2);	取得溫度
113	return ut;	回傳溫度值
114	}	函式結束
115	unsigned long bmp085ReadUP(){	讀取未補償氣壓值函式
116	byte MSB, LSB, XLSB;	區域變數宣告成位元組
117	unsigned long up = 0;	區域變數宣告成無符號位元長整數
118	setReg(BMP085_ADDRESS,Register_CMD, (0x34 + (OSS<<6)));	下氣壓讀取指令 0x34+(OSS<<6)
119	delay(2 + (3<<OSS));	根據 OSS 等待轉換
120	MSB = getData(BMP085_ADDRESS,Register_D2);	取得氣壓 MSB 資料
121	LSB = getData(BMP085_ADDRESS,Register_D1);	取得氣壓 LSB 資料
122	XLSB = getData(BMP085_ADDRESS,Register_D0);	取得氣壓 XLSB 資料
123	up = (((unsigned long) MSB << 16) | ((unsigned long) MSB << 8) | (unsigned long) XLSB) >> (8-OSS);	組合成無符號位元長整數數值
124	return up;	回傳氣壓值
125	}	函式結束
126	float calcAltitude(float pressure){	計算高度函式
127	float A = pressure/101325;	
128	float B = 1/ 5.25588;	
129	float C = pow(A,B);	
130	C = 1 - C;	
131	C = C / 0.0000225577;	
132	return C;	
133	}	
134	void setReg(int device_address,	取得暫存器裡的位元組資料函式

	unsigned char reg_address, unsigned char data){	
135	Wire.beginTransmission(device_address);	開始指定位址的裝置傳輸
136	Wire.write(reg_address);	送出暫存器位址
137	Wire.write(data);	寫入資料到暫存器
138	Wire.endTransmission();	結束傳輸
139	}	函式結束
140	byte getData(int device_address, unsigned char reg_address){	取得暫存器裡的位元組資料函式
141	Wire.beginTransmission(device_address);	開始指定位址的裝置傳輸
142	Wire.write(reg_address);	送出暫存器位址
143	Wire.endTransmission();	結束傳輸
144	Wire.requestFrom(device_address,1);	請求讀取
145	if(Wire.available()<=1){	判別接收的資料數量小於等於 1
146	return Wire.read();	回傳一位元組資料
147	}	if 結束
148	}	函式結束
149		
150	int bmp085ReadInt(unsigned char address) {	取得暫存器裡的整數資料函式
151	byte MSB, LSB;	
152	MSB =getData(BMP085_ADDRESS, address);	
153	LSB =getData(BMP085_ADDRESS, address+1);	
154	return (int) MSB<<8 | LSB;	
155	}	

練習

四、 透過網路搜尋 ADXL345 函式庫，使用 ADXL345 函式庫讀取三軸加速度感測器 ADXL345 的 G 值並顯示在 LCD 上。

五、 透過網路搜尋 L3G4200D 函式庫，使用 L3G4200D 函式庫讀取的角速度值與溫度並顯示在 LCD 上。

六、 透過網路搜尋 HMC5883L 函式庫，使用 HMC5883L 函式庫讀取的磁感應數值與計算出指的方向並顯示在 LCM 上。

七、 透過網路搜尋 BMP085 函式庫，使用 BMP085 函式庫讀取的氣壓與溫度感測器的感測數值並顯示在 LCM 上。

實驗 3-11：太陽能電池模組實習

目的：瞭解太陽能電池模組的工作原理及使用 Arduino 程式如何控制。

功能：本實習使用 Arduino MEGA2560 如何偵測太陽能電池模組輸出電壓。程式中使用 Arduino MEGA2560 類比輸入接腳來檢測太陽能模組輸出的電壓值並由 LCM 做顯示。

原理：太陽光電是利用太陽能電池元件直接將太陽光能轉換成電力的方式。太陽能電池是利用自然界中的矽元素，製成 P 型及 N 型半導體作正負極，這兩種半導體吸收太陽能後即可產生電位差而產生電流，因此所謂「太陽能電池」其實是一種發電系統，而非儲電系統。

太陽能模組板可以視電池模組所需要的輸出能量需求，採串聯或並聯形式進行整合與連接，搭配外框與保護玻璃的設計進行組合，即可建置不同功率輸出的太陽能電池模組。太陽能所產生的電為直流電，必須透過轉換器，變成交流電，才可以供電器使用。有些太陽能發電裝設蓄電功能，可以將轉換的電能儲存起來，晚上或沒有太陽的時候仍然有電可用。

本實驗使用型號為 ZBM_SENSOR_SolarEnergy 太陽能模組(如圖 3-11-1 所示)。為了方便學習，本實驗僅以小面積太陽能板作為實驗項目，使用 Arduino MEGA2560 類比輸入接腳來偵測太陽能模組輸出的電壓。

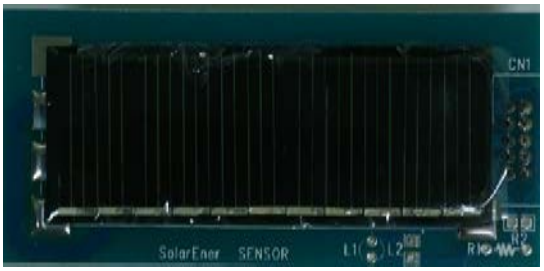


圖 3-11-1　型號為 ZBM_SENSOR_SolarEnergy 太陽能模組實體圖

電路：太陽能模組與 Arduino MEGA2560 開發板的實體接圖如圖 3-11-2 所示，圖 3-11-3 為實體接腳電路圖，主要使用主板中 CN1 連接埠內的 A5 類比輸入接腳連接到太陽能模組輸出電壓接腳。程式中會利用 A5 類比輸入接腳偵測太陽能模組輸出類比電壓值轉成 10 位元(0~1024)數值 V_i，再利用以下公式(1)轉成 0~5V 數值 V_o，並由 LCM 顯示。當太陽能模組板接收的光越強，輸出顯示的電壓就越大，反之就越小！

$$V_o = (V_i \times 5)/1024 \qquad (1)$$

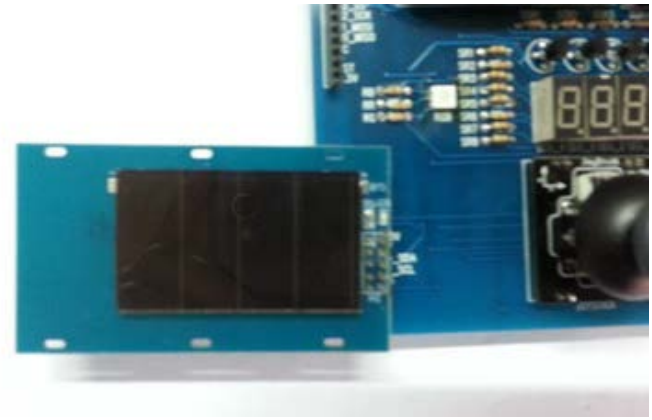


圖 3-11-2 Arduino MEGA2560 對太陽能模組的實體接圖

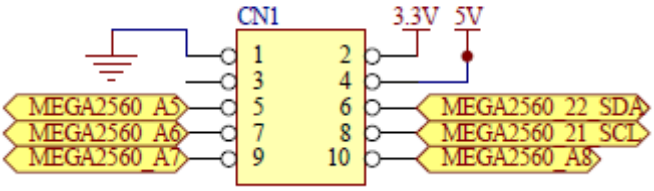


圖 3-11-3 Arduino MEGA2560 對太陽能模組的實體接腳電路圖

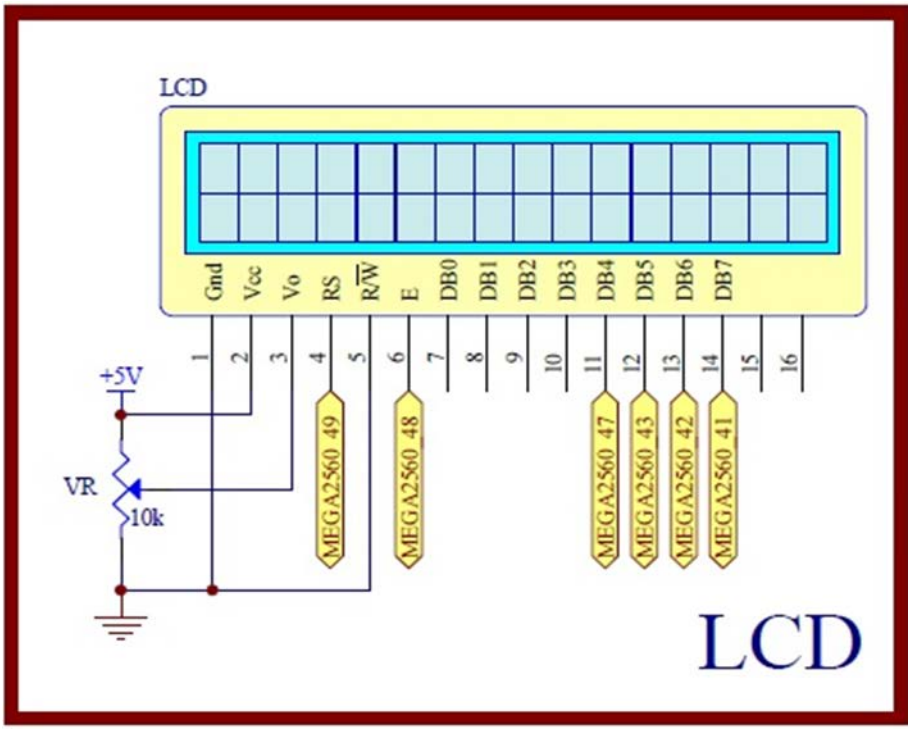


圖 3-11-4 Arduino MEGA2560 對應 LCD 顯示模組的實體接腳電路圖

元件：本實習所需元件如表 3-11-1 所示。

表 3-11-1 元件表

編號	元件項目	數量	元件名稱
1	Arduino MEGA2560	1	Arduino 開發板
2	ZBM_SENSOR_SolarEnergy	1	太陽能模組
3	LCD	1	16×2 文字型 LCD 顯示器
4	VR	1	10kΩ 可變電阻

程式：藍牙遙控 LED 亮滅

P3-11-1

行號	程式敘述	註解
1	#include <LiquidCrystal.h>	加入 LCD 顯示模組的驅動函式庫
2	#define Solar_Pin A5	定義太陽能輸出電壓接腳在類比 IO 第 5 支
3	LiquidCrystal lcd(49, 48, 47, 43, 42, 41);	定義 LCD 物件對應 Arduino 的腳位 lcd(RS,E,D4,D5,D6,D7)
4	float V;	宣告全域變數 V 為浮點數
5	void setup() {	只會執行一次的程式初始設定函式
6	Serial.begin(9600);	設定序列埠監控視窗傳遞速率為 9600
7	lcd.begin(16, 2);	初始化 LCD 物件的格式為 16 字、2 行
8	lcd.print("PV= V");	LCD 顯示出字串"PV= V"
9	}	setup()函式結束
10	void loop() {	loop()函式開始
11	int sensorValue = analogRead(Solar_Pin);	讀進太陽能輸出類比電壓轉成 10 位元數位值
12	V=(float)sensorValue*5/1024;	轉換成 0~5V 電壓值
13	Serial.print("PV = ");	觀測視窗顯示出字串"PV = "
14	Serial.print(V);	觀測視窗顯示 V 值
15	Serial.println(" V");	觀測視窗顯示出字串" V"
16	lcd.setCursor(4, 0);	游標設到 LCD 第 5 字、第 1 行
17	lcd.print(V);	LCD 顯示出 V 值
18	delay(100);	延時 0.1 秒，更新頻率
19	}	loop()函式結束

練習

一、 設計一系統，當太陽能模組輸出電壓大於 4V 時，將 LED 打亮，低於 4V 時 LED 關閉。

實驗 3-12：RFID 讀寫器通訊實驗

目的：瞭解 RFID 無線射頻辨識系統的工作原理及使用 Arduino 程式控制如何通訊。

功能：本實習使用 Arduino MEGA2560 控制 RFID 讀卡機讀取 RFID 卡片控制。程式(一)中使用 Arduino MEGA2560 控制 RFID 讀卡機讀取 RFID 卡片內 ID 碼並顯示在 LCM 上。程式(二)中讀取 RFID 卡片內 ID 碼後進一步作門禁控制。

原理：RFID（Radio Frequency Identification）無線射頻辨識系統基本架構圖如圖 3-12-1 所示，基本上包括電子標籤（Tag）、讀卡機（RF Reader）及應用程式資料庫電腦系統。是一種新興的辨識技術，是一種以 RF 無線電波辨識物件的自動辨識技術，主要是利用讀卡機發射 RF 能量來讀取植入或貼附在物件上的電子標籤，以進行無線資料辨識及存取的工作。所以 RFID 系統只要將 RFID 電子標籤貼在商品上，商家便可藉由一台讀取機，立即清點商品數量與流向。RFID 辨識技術的應用層面相當廣泛，日常生活中便可接觸到，例如：捷運的悠遊卡、社區門禁管制系統的感應卡、或者賣場、書店門口的防竊系統等等。

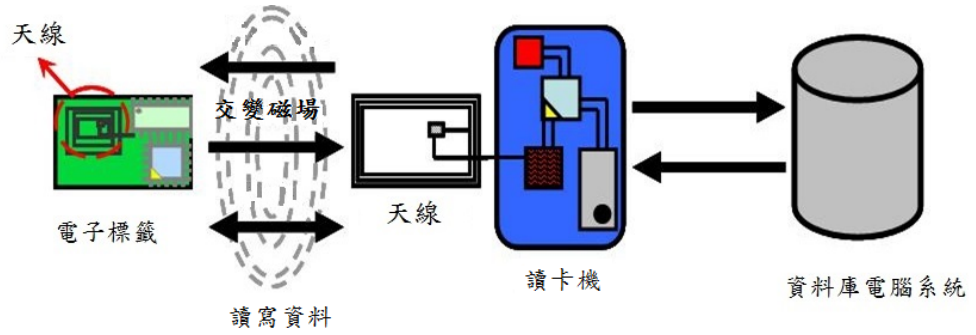


圖 3-12-1 RFID 系統基本架構圖

本實驗使用 RFID-RC522 模組(圖 3-12-2 所示)，MF RC522 是應用於 13.56MHz 非接觸式通信中高集成度的讀寫卡晶片，是 NXP 公司推出的一款低電壓、低成本、體積小的非接觸式讀寫卡晶片，是智慧型儀器表和可攜式手持設備研發的較好選擇。它與主機間通信採用 SPI 模式，有利於減少連線，縮小 PCB 板體積，降低成本。支援的卡類型有 mifare1 S50、mifare1 S70、mifare UltraLight、mifare Pro 及 mifare Desfire。

RFID-RC522 模組使用 13.56MHz 的無線電來讀取靠近 RFID 天線 5 公分內電子標籤的資料，每張卡片存放 4 Bytes 的 ID 資料及 1Byte CRC 校驗位元組，因每一個卡片的資料都是唯一無法修改，好像機械鑰匙的刻痕，所以可用於當成擁有此張 RFID 卡片的使用者之辨識碼。

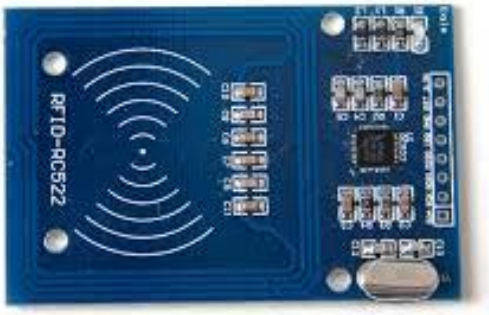


圖 3-12-2 RFID-RC522 模組

電路：RFID-RC522 模組與 Arduino MEGA2560 開發板的實體接圖如圖 3-12-3 所示，圖 3-12-4 為實體接腳電路圖使用 SPI 通訊界面，圖中 Arduino MEGA2560 開發板上的第(53, 52, 51, 50, 5)支數位接腳對應連接到 SPI 介面的(SS, SCK, MOSI, MISO, RST)接腳，Arduino 透過引入<SPI.h>函式庫函式把 SPI 介面控制變簡單了。程式(一)中會使用 Arduino MEGA2560 藉由 SPI 介面控制 RFID-RC522 讀卡機讀取 RFID 卡片內 ID 碼並顯示在 LCM 上，同時讓蜂鳴器產生短暫的嗶一聲。程式(二)中會使用 Arduino MEGA2560 藉由 SPI 介面控制 RFID-RC522 讀卡機讀取 RFID 卡片內 ID 碼並顯示在 LCM 上，同時比對程式內建的 ID 碼資料是否存在，若有則觸發繼電器打開並將 8 個 LED 打亮，表示身分比對正確。若沒有則讓蜂鳴器重複產生警報聲，表示身分比對錯誤，達到門禁的控制功能。

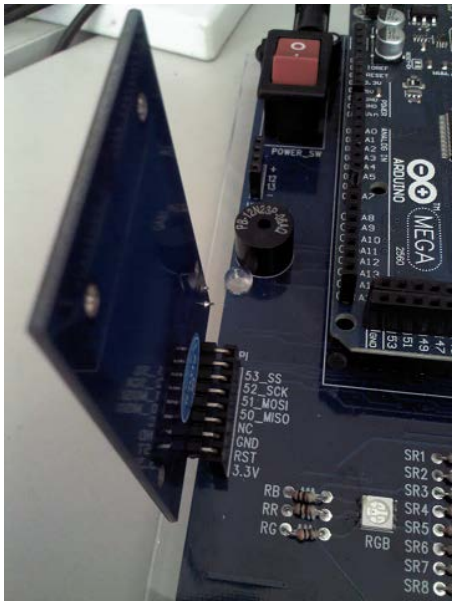


圖 3-12-3 Arduino MEGA2560 對 RFID-RC522 模組的實體接圖

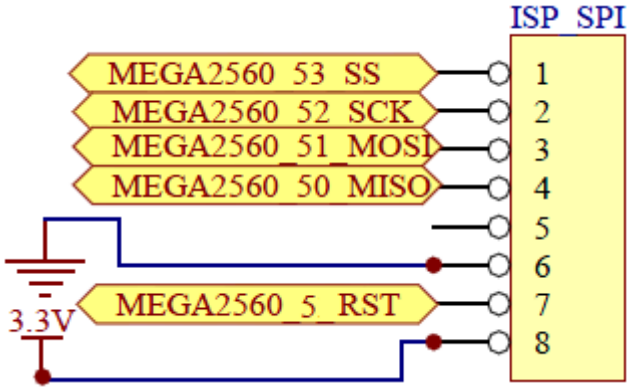


圖 3-12-4 Arduino MEGA2560 對 RFID-RC522 模組的實體接腳電路圖

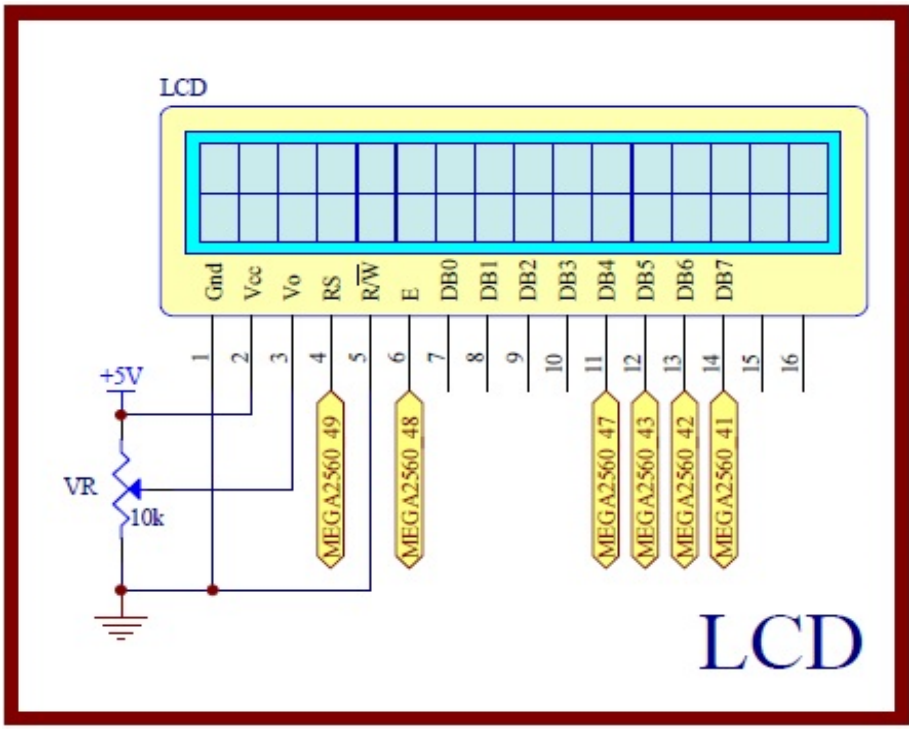


圖 3-12-5 Arduino MEGA2560 對應 LCD 顯示模組的實體接腳電路圖

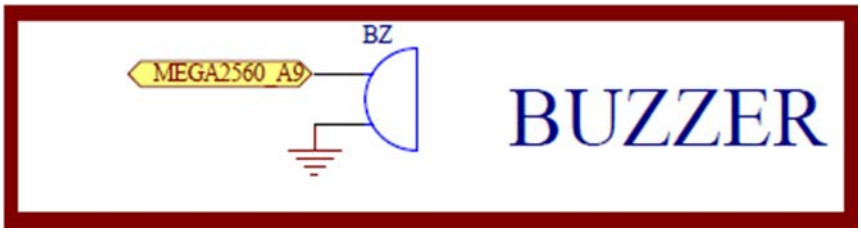


圖 3-12-6 Arduino MEGA2560 對應蜂鳴器的實體接腳電路圖

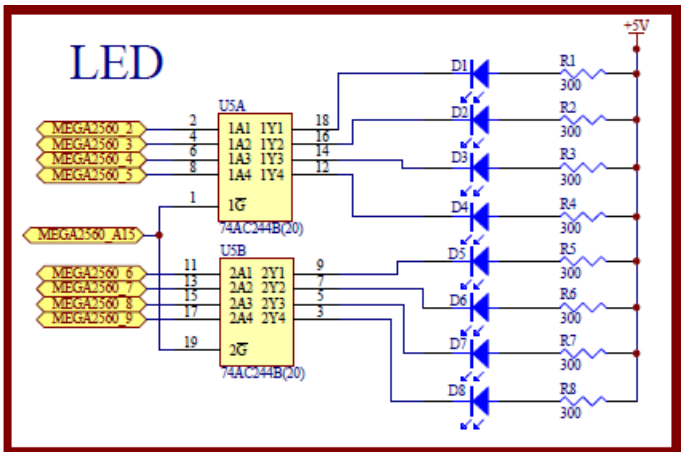


圖 3-12-7 Arduino MEGA2560 對應 LED 的實體接腳電路圖

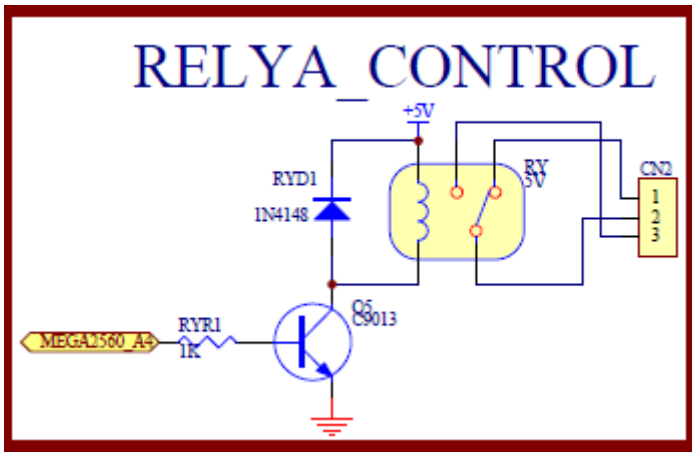


圖 3-12-8 Arduino MEGA2560 對應繼電器的實體接腳電路圖

元件：本實習所需元件如表 3-12-1 所示。

表 3-12-1 元件表

編號	元件項目	數量	元件名稱
1	Arduino MEGA2560	1	Arduino 開發板
2	RFID-RC522 模組	1	RFID 模組
3	RFID 卡	1	電子標籤
4	LCD	1	16×2 文字型 LCD 顯示器
5	VR	2	10kΩ 可變電阻
6	BUZZER	1	5V 電磁式有源式蜂鳴器
7	74AC244	1	8 個緩衝閘晶片
8	D1	1	單色 LED
9	R1	1	300 Ω電阻
10	RELAY	1	繼電器

程式（一）：讀取 RFID 卡的 ID 值

P3-12-1

行號	程式敘述	註解
1	#include <LiquidCrystal.h>	加入 LCD 顯示模組的驅動函式庫
2	#include <SPI.h>	加入 SPI 介面驅動函式庫
3	#include <RFID.h>	加入 RFID 模組的驅動函式庫
4	const int buzzer = A9;	定義 BUZZER 接腳在類比 IO 第 9 支
5	RFID rfid(53,5);	D53--讀卡機 SS 接腳、D5--讀卡機 RST 接腳

6	LiquidCrystal lcd(49, 48, 47, 43, 42, 41);	定義 LCD 物件對應 Arduino 的腳位 lcd(RS,E,D4,D5,D6,D7);
7	void setup() {	只會執行一次的程式初始設定函式
8	pinMode(buzzer,OUTPUT);	規劃 buzzer 腳為輸出模式
9	SPI.begin();	初始化 SPI 腳位
10	rfid.init();	初始化 RFID
11	lcd.begin(16, 2);	初始化 LCD 物件的格式為 16 字、2 行
12	lcd.clear();	清除 LCD 螢幕
13	lcd.setCursor(0,0);	游標設到 LCD 第 1 字、第 1 行
14	lcd.print("==Arduino RFID==");	LCD 顯示出字串"==Arduino RFID=="
15	lcd.setCursor(0, 1);	游標設到 LCD 第 1 字、第 2 行
16	lcd.print("Swipe Your Card!");	LCD 顯示出字串"Swipe Your Card!"
17	}	setup()函式結束
18	void loop(){	loop()函式開始
19	unsigned char ii;	宣告 ii 為 unsigned char 變數
20	if (rfid.isCard()) {	判斷是否有電子標籤卡片存在
21	if (rfid.readCardSerial()){	讀取電子標籤卡片的資料
22	lcd.setCursor(0, 1);	游標設到 LCD 第 1 字、第 2 行
23	lcd.print("ID:");	LCD 顯示出字串"ID:"
24	for(ii=0;ii<5;ii++){	第 0~3 個 byte:卡片 ID，第 4 個 byte:CRC 校驗位元
25	if(rfid.serNum[ii] <= 0x0f){	讀出的值小於等於 0x0f
26	lcd.print("0");	則補一個'0'，由 LCD 顯示出字
27	lcd.print(rfid.serNum[ii] ,HEX);	LCD 顯示出讀出的值
28	}	if 結束
29	else{	否則
30	lcd.print(rfid.serNum[ii] ,HEX);	LCD 顯示出讀出的值讀出的值
31	}	else 結束
32	}	if 結束
33	lcd.print(" ");	LCD 第 2 行清除
34	digitalWrite(buzzer,HIGH);	buzzer 輸出 HIGH，buzzer 發聲
35	delay(200);	延時 0.2 秒
36	digitalWrite(buzzer,LOW);	buzzer 輸出 LOW，buzzer 發聲停止
37	delay(1000);	延時 1 秒

38	lcd.setCursor(0, 1);	游標設到 LCD 第 1 字、第 2 行
39	lcd.print("Swipe Your Card!");	LCD 顯示出字串"Swipe Your Card!"
40	}	if 結束
41	}	if 結束
42	rfid.halt();	命令電子標籤卡片進入休眠狀態
43	delay(500);	延時 0.5 秒
44	}	loop()函式結束

程式（二）：RFID 門禁控制

P3-12-2

行號	程式敘述	註解
1	#include <LiquidCrystal.h>	加入 LCD 顯示模組的驅動函式庫
2	#include <SPI.h>	加入 SPI 介面驅動函式庫
3	#include <RFID.h>	加入 RFID 模組的驅動函式庫
4	#define buzzer A9	定義 BUZZER 接腳在類比 IO 第 9 支
5	#define relay A4	定義 relay 接腳在類比 IO 第 4 支
7	RFID rfid(53,5);	D53--讀卡機 SS 接腳、D5--讀卡機 RST 接腳
8	LiquidCrystal lcd(49, 48, 47, 43, 42, 41);	定義 LCD 物件對應 Arduino 的腳位 lcd(RS,E,D4,D5,D6,D7);
9	int cards[][5] = {	宣告全域變數，紀錄 RFID 電子標籤 ID 碼
10	{0x20,0x9E,0x0F,0x55,0xE4},	card 1 電子標籤 ID 碼
11	{0xCE,0xC3,0x8F,0xA9,0x2B}	card 2 電子標籤 ID 碼
12	};	
13	int BASE = 2;	宣告全域變數，第一顆 LED 接的 I/O 腳編號
14	int NUM = 8;	宣告全域變數，LED 的總數
15	bool access = true;	宣告全域變數，access 為布林變數
17	void setup() {	只會執行一次的程式初始設定函式
18	Serial.begin(9600);	定義序列埠監控視窗傳遞速率為 9600
19	SPI.begin();	初始化 SPI 腳位
20	rfid.init();	初始化 RFID

21	pinMode(buzzer,OUTPUT);	規劃 buzzer 腳為輸出模式
22	pinMode(relay, OUTPUT);	規劃 relay 腳為輸出模式
23	digitalWrite(relay, LOW);	設定 relay 不觸發
24	for (int i = BASE; i < BASE + NUM; i ++){	
25	pinMode(i, OUTPUT);	設定 LED I/O 腳為輸出模式
26	}	
27	pinMode(A15, OUTPUT);	規劃 A15 腳為輸出模式
28	digitalWrite(A15, LOW);	A15 輸出 LOW，致能 74AC244
29	lcd.begin(16, 2);	初始化 LCD 物件的格式為 16 字、2 行
30	lcd.clear();	清除 LCD 螢幕
31	lcd.setCursor(0,0);	游標設到 LCD 第 1 字、第 1 行
32	lcd.print("==Arduino RFID==");	LCD 顯示出字串"==Arduino RFID=="
33	lcd.setCursor(0, 1);	游標設到 LCD 第 1 字、第 2 行
34	lcd.print("Swipe Your Card!");	LCD 顯示出字串"Swipe Your Card!"
35	}	setup()函式結束
37	void loop(){	loop()函式開始
38	int i,j;	變數宣告 i,j 為整數
39	if (rfid.isCard()) {	判斷是否有電子標籤卡片存在
40	if (rfid.readCardSerial()){	讀取電子標籤卡片的資料
41	lcd.setCursor(0, 1);	游標設到 LCD 第 1 字、第 2 行
42	lcd.print(" ");	LCD 第 2 行清除
43	lcd.setCursor(0, 1);	游標設到 LCD 第 1 字、第 2 行
44	lcd.print("ID:");	LCD 顯示出字串"ID:"
45	Serial.print("ID:");	觀測視窗顯示出字串"ID:"
46	for(i=0; i<5; i++){	第 0~3 個 byte:卡片 ID，第 4 個 byte:CRC 校驗位元
47	if(rfid.serNum[i] <= 0x0f){	讀出的值小於等於 0x0f
48	lcd.print("0");	則補一個'0'，由 LCD 顯示出字
49	lcd.print(rfid.serNum[i] ,HEX);	LCD 顯示出讀出的值
50	Serial.print("0");	觀測視窗顯示出字串"0"
51	Serial.print(rfid.serNum[i] ,HEX);	觀測視窗顯示出讀出的值
52	}	if 結束

53	else{	否則
54	lcd.print(rfid.serNum[i] ,HEX);	LCD 顯示出讀出的值讀出的值
55	Serial.print(rfid.serNum[i] ,HEX);	觀測視窗顯示出讀出的值
56	}	else 結束
57	}	if 結束
58	for(i = 0; i < sizeof(cards); i++){	
59	for(j = 0; j < sizeof(rfid.serNum); j++){	ID 碼 5 個 Bytes 資料比對
60	if(rfid.serNum[j] != cards[i][j]) {	每一個 Byte 比對是否正確
61	access = false;	若沒有，access 設 false
62	break;	離開 for loop
63	}	if 結束
64	else {	每一個 Byte 比對正確
65	access = true;	access 設 true
66	}	else 結束
67	}	for loop 結束
68	if(access) break;	每一個 Byte 比對正確，離開 for loop
69	}	for loop 結束
70	}	if 結束
71	if(access){	ID 碼 5 個 Bytes 資料比對結果成功
72	lcd.setCursor(0,0);	游標設到 LCD 第 1 字、第 1 行
73	lcd.print("Welcome! ");	LCD 顯示出字串"Welcome! "
74	Serial.println("\nWelcome!");	觀測視窗顯示字串"Welcome!"，並換行
75	for (i = BASE; i < BASE + NUM; i ++) {	
76	digitalWrite(i, LOW);	設定 8 個 LED 打亮
77	}	
78	digitalWrite(relay, HIGH);	設定繼電器打開
79	}	if 結束
80	else {	ID 碼 5 個 Bytes 資料比對結果失敗
81	lcd.setCursor(0,0);	游標設到 LCD 第 1 字、第 1 行
82	lcd.print("No Entry! ");	LCD 顯示出字串"No Entry! "
83	Serial.println("No Entry!");	觀測視窗顯示字串"No Entry!" ，並換行

84	digitalWrite(relay, LOW);	設定繼電器關閉
85	}	else 結束
86	}	if 結束
88	if(!access) {	ID 碼 5 個 Bytes 資料比對結果失敗
89	digitalWrite(buzzer, HIGH);	打開蜂鳴器
90	delay(500);	延時 0.5 秒
91	digitalWrite(buzzer, LOW);	關閉蜂鳴器
92	delay(1000);	延時 1 秒
93	}	if 結束
94	else {	ID 碼 5 個 Bytes 資料比對結果成功
95	delay(2000);	延時 2 秒
96	digitalWrite(buzzer, LOW);	關閉蜂鳴器
97	digitalWrite(relay, LOW);	設定繼電器關閉
98	for (i = BASE; i < BASE + NUM; i ++) {	
99	digitalWrite(i, HIGH);	設定 8 個 LED 關閉
100	}	
101	}	else 結束
102	lcd.clear();	清除 LCD 螢幕
103	lcd.setCursor(0,0);	游標設到 LCD 第 1 字、第 1 行
104	lcd.print("==Arduino RFID==");	LCD 顯示出字串"==Arduino RFID=="
105	lcd.setCursor(0, 1);	游標設到 LCD 第 1 字、第 2 行
106	lcd.print("Swipe Your Card!");	LCD 顯示出字串"Swipe Your Card!"
107	rfid.halt();	命令電子標籤卡片進入休眠狀態
108	delay(500);	延時 0.5 秒
109	}	loop()函式結束

練習

一、利用 Arduino 發展 RFID 讀卡機，可以寫入並讀出 RFID 卡片的資料。