國立高雄師範大學

微控制器期末專題

熱敏電阻暨溫度感測器設計

班級:電機三

學號:410773019

姓名:許曼楨

指導教授:黃裕峰 教授

一.設計動機:

上個月，高雄天氣非常炎熱，但我的家人為了要省錢，所以就選擇不吹冷氣，沒想到，居然就中暑了!以前，我總以為中暑不是什麼大不了的事，因為每次我中暑，頂多就是拉拉肚子而已。不曉得是否因為是我還很年輕，身強體壯，所以總覺得中暑只是小 case，因而每次看到電視新聞播報:有老人因為熱衰竭死亡，我都覺得很誇張!直到這次看到家中的長輩因為中暑而苦不堪言，並且需要休養數日，當時我就在想是否能利用 Arduino 製作簡易溫度計，這樣一來，我們既不用花錢買溫度計，又可以將其放在家中顯眼處，隨時提醒長輩:現在室溫有多高，該吹冷氣還是要吹!

二.設計理念:

還記得之前上課老師曾提到熱敏電阻，所以這次我決定利用熱敏電阻，來作為我測量溫度的儀器，並且以七段顯示器來作為顯示儀器，最後再以 LED 燈作為警示燈。

三.使用器材:

1.Mega 2560*1

2.麵包板*1

3.公-公杜邦線(數條)

4.熱敏電阻(5kΩ)*1

5. 共陰極 7 段顯示器(4 位數)*1

6. LED 燈*2(紅、綠)

7. 5kΩ電阻*1

8. 330Ω電阻*10

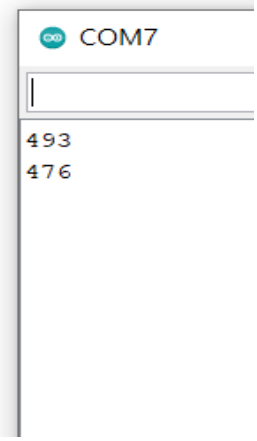
四. 器材介紹:

(一)熱敏電組

熱敏電阻主要可分為兩類，一類為 NTC(負溫度係數)，另一類為 PTC(正溫度係數)。負溫度係數(NTC)，顧名思義，即當溫度升高，電阻阻值反而會下降，此種熱敏電阻適合電流限制器。反之，正溫度係數(PTC)，即當溫度升高，電阻阻值亦會隨之增高，其適合用來作為保險絲。

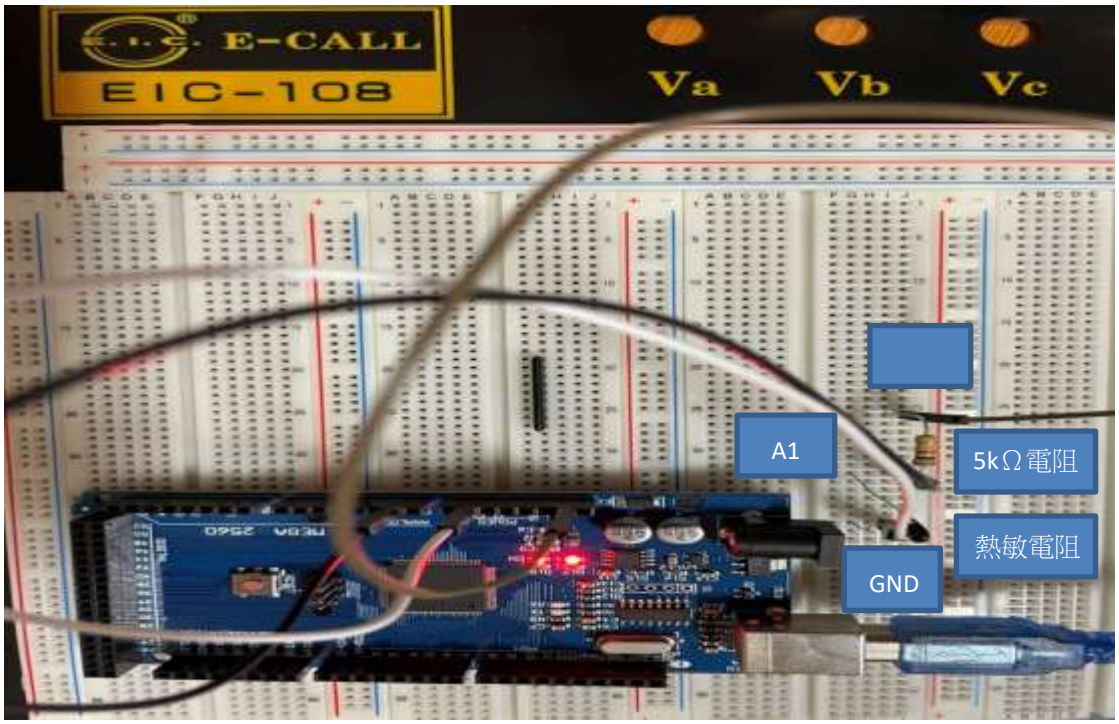
當我在購買熱敏電阻時，店家並未標示其為正溫度係數之熱敏電阻，抑或是負溫度係數之熱敏電阻，所以我用 Arduino 寫了一個程式，並以序列埠監控視窗顯示其輸出。

```
int a;
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
a=analogRead(A1);
Serial.println(a);
delay(3000);
}

void loop() {
  // put your main code here, to run repeatedly:

}
```



▲ 測試熱敏電阻為 NTC 抑或是 PTC 之程式

從此圖片，我們可發現:當我未加熱前，輸出為 493，但當我拿吹風機加熱後，輸出反而下降到了 476，也就是說，當溫度升高，輸出會下降，所以結論為:我買到的熱敏電阻為 NTC 熱敏電阻。



▲ 測試熱敏電阻為 NTC 抑或是 PTC 之實際電路

(二)七段顯示器

七段顯示器亦分為兩種，一種為共陽極，另一種為共陰極。共陽極是採用低電位驅動，而共陰極則採用高電位驅動。而這次我所採用的是共陰極四位數七段顯示器，也就是說，四位數內部的

a, b, c, d, e, f, g, dp 都接在一起，再共用一個接點拉出，而每一位數的共陰極個別輸出，千位數為 com1，百位數為 com2，十位數為 com3，個位數為 com4。

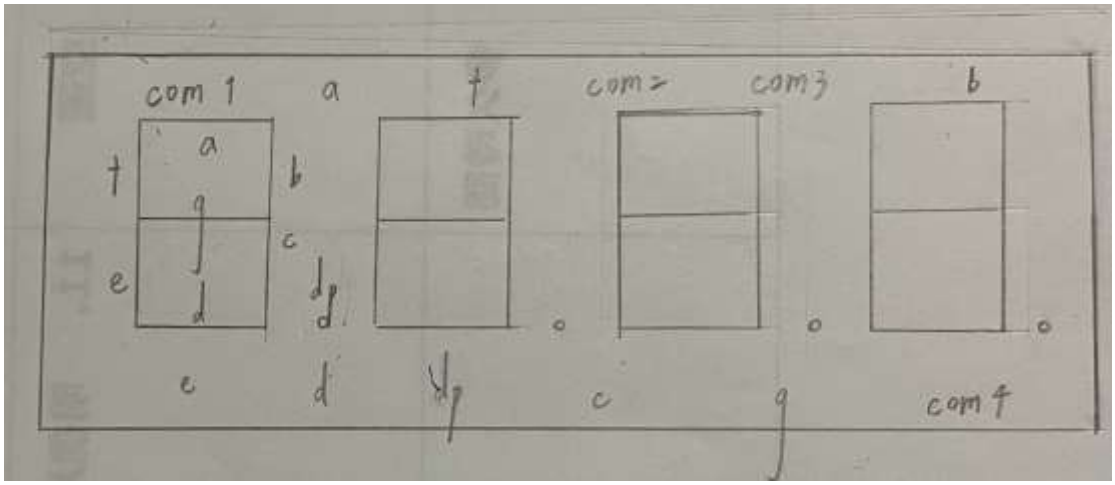
顯示燈號	千 com1(PC3)	百 Com2(PC2)	拾 com3(PC1)	個 com4(PC0)	值
千位	0	1	1	1	0x7
百位	1	0	1	1	0xb
十位	1	1	0	1	0xd
個位	1	1	1	0	0xe

▲ com1、com2、com3、com4 電位表

數字	dp	g	f	e	d	c	b	a	十六進制數值
0	0	0	1	1	1	1	1	1	0x3F
1	0	0	0	0	0	1	1	0	0x6
2	0	1	0	1	1	0	1	1	0x5B
3	0	1	0	0	1	1	1	1	0x4F
4	0	1	1	0	0	1	1	0	0x66
5	0	1	1	0	1	1	0	1	0x6D
6	0	1	1	1	1	1	0	1	0x7D
7	0	0	0	0	0	1	1	1	0x7
8	0	1	1	1	1	1	1	1	0x7F
9	0	1	1	0	1	1	1	1	0x6F

▲ 數字 0~9 的七段顯示器十六進制數值表

而四位數的七段顯示器是利用人類視覺暫留的特性，快速輪流掃描輸出(頻率大於 24Hz)，所以我們才能看到四位數同時顯示，但事實上，並不是同時顯示，只是我們感覺不出來!



▲ 此為七段顯示器腳位圖

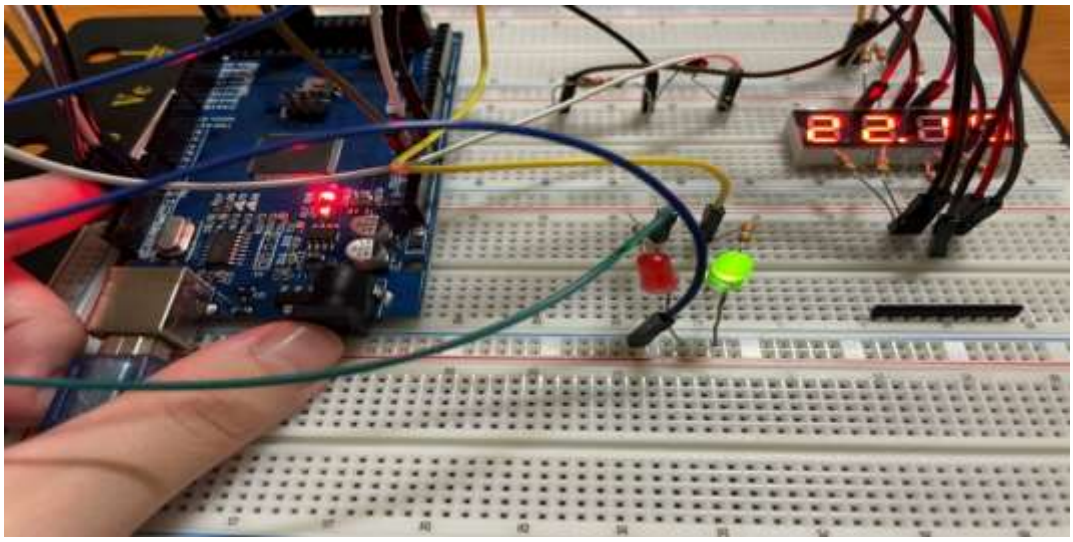
五、程式設計

final

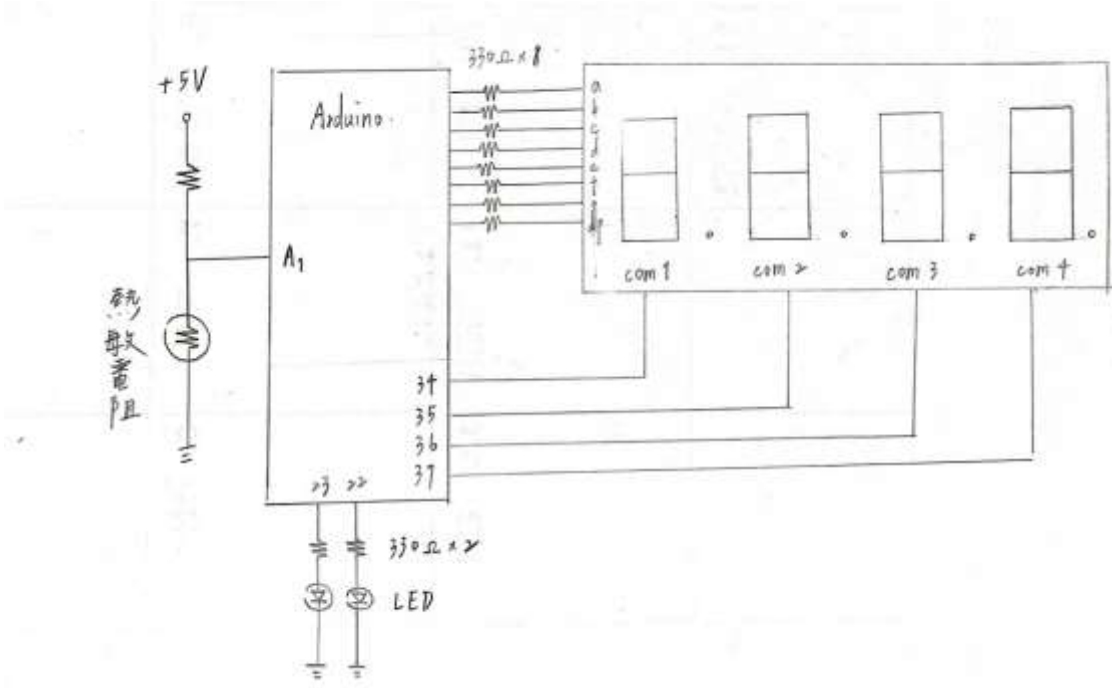
```
float c,d,Rt,e,g;  //規定資料儲存空間
float a[]={0x7,0xb,0xd,0xe};  //七段顯示器位址
float b[]={0x3f,0x6,0x5b,0x4f,0x66,0x6d,0x7c,0x7,0x7f,0x67,0xbf,0x86,0xdb,0xcf,0xe6,0xed,0xfd,0x87,0xff,0xe7}; //七段顯示器資料
int f[4];  //規定資料儲存空間(陣列)
unsigned long t1,t2;  //規定時間資料儲存空間

void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);  //啟用序列埠
pinMode(22,OUTPUT);  //單隻腳位指派腳位功能
pinMode(23,OUTPUT);  //單隻腳位指派腳位功能
}

void loop() {
  // put your main code here, to run repeatedly:
DDRB=0xFF;  //指派PB為輸出
DDRC=0xFF;  //指派PC為輸出
c=analogRead(A1);  //從A1腳位讀取電壓，c從0到1023
```

```
//將A1腳位輸出的電壓，轉換為實際凱式溫度
d=(c/1023)*5;  //得到實際電壓值(實際電壓值只會在0~5之間，所以不能直接使用上面輸出的電壓值)
Rt=d*5/(5-d);  //利用分壓比得到熱敏電阻的電阻值
g=1/298.15+log(Rt/5)/3600;  //利用NTC熱敏電阻公式計算實際凱式溫度(此處3600為熱敏指數)
e=1/g-273.15+0.5;  //利用NTC熱敏電阻公式計算實際凱式溫度(0.5為誤差)
Serial.println(e);  //在序列埠中顯示實際凱式溫度

t1=millis();  //讀取程式執行時間
f[0]=e/10;    //取出凱式溫度的十位數值
f[1]=(e-f[0]*10)/1;  //取出凱式溫度的個位數值
f[2]=((e*10)-f[0]*100-f[1]*10)/1;  //取出凱式溫度的十分位數值
f[3]=((e*100)-f[0]*1000-f[1]*100-f[2]*10)/1;  //取出凱式溫度的百分位數值

//使程式一直掃描，如此一來，七段顯示器才能一直顯示數字
do{
//在位址千位中放入凱式溫度的十位數

PORTC=a[0];  //位址
PORTB=b[f[0]];  //資料
delay(1);  //延遲一秒

//在位址百位中放入凱式溫度的個位數
PORTC=a[1];  //位址
PORTB=b[f[1]+10];  //資料(這裡多加了10是因為要顯示出小數點)
delay(1);  //延遲一秒

//在位址十位中放入凱式溫度的十分位數
PORTC=a[2];  //位址
PORTB=b[f[2]];  //資料
delay(1);  //延遲一秒

//在位址個位中放入凱式溫度的百分位數
PORTC=a[3];  //位址
PORTB=b[f[3]];  //資料
delay(1);  //延遲一秒

PORTA=0;  //指派PB為輸出
if(e>=30)
digitalWrite(22,HIGH);  //如果凱式溫度大於30度就亮紅燈
else
digitalWrite(23,HIGH);  //如果凱式溫度小於30度就亮綠燈

t2=millis();  //讀取程式執行時間
}while((t2-t1)<5000);  //當時間差小於5，會一直執行上述程式
}
```

六、實際電路

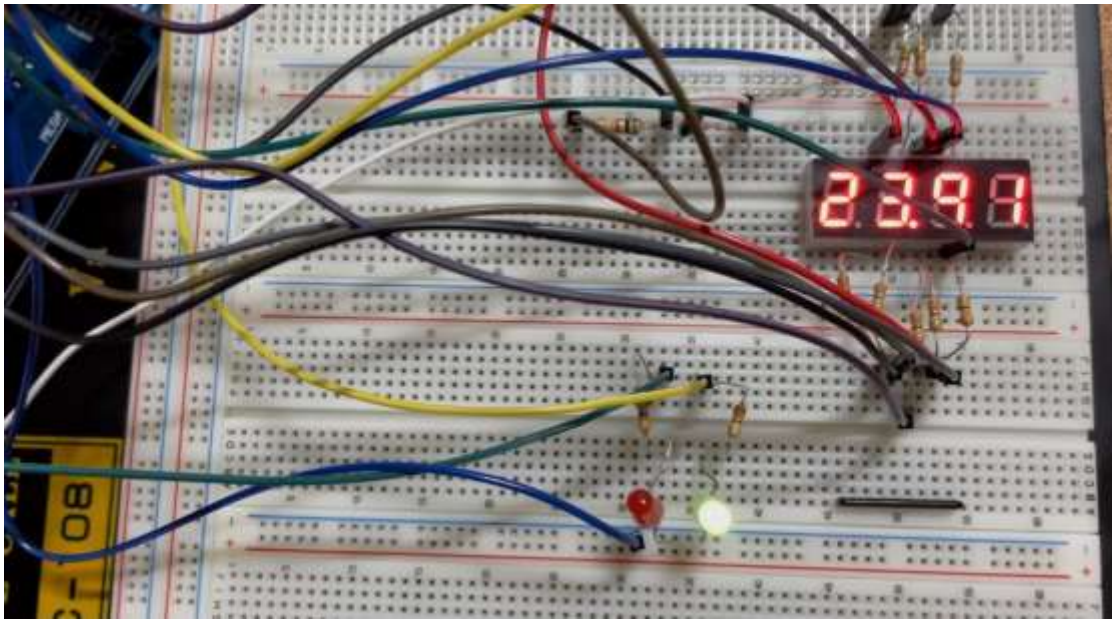


從這張圖我們可以看到:我的設計主要分成三部分，一部分是中上的溫度測量，一部分是最右邊的七段顯示器，而最後一部分則是以 LED 作為警示燈，但由於接線較為複雜，以下，我將以手繪的方式呈現電路。



七、實驗結果及說明

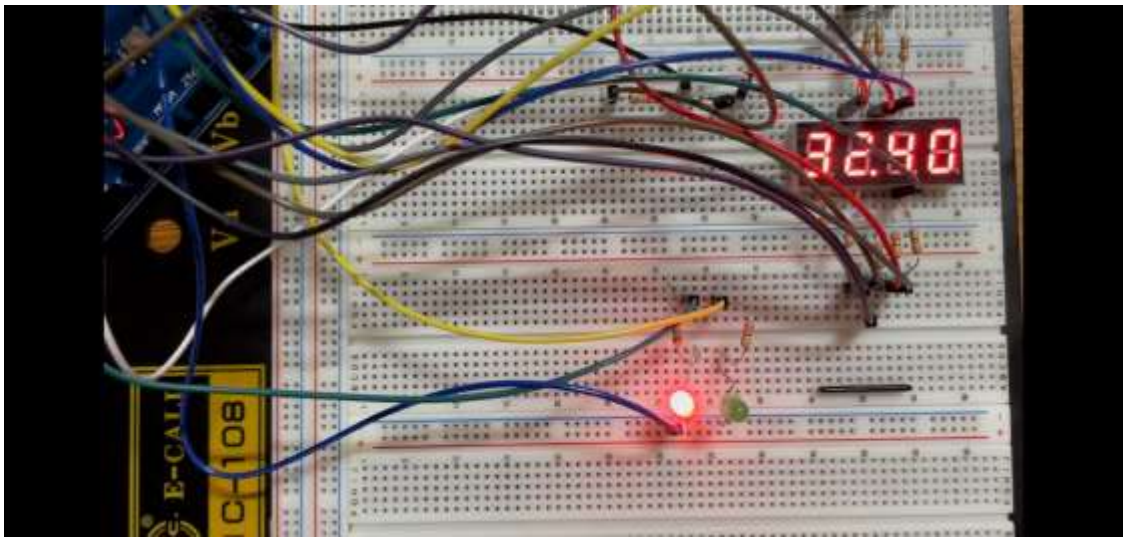
首先，在室溫下(未開電風等)，我測量到的溫度，如影片一及下圖所示，幾乎都是接近 24°C(影片中測量到的多個數據皆為 23.91°C)，而量測當下室外的溫度大約為27°C，當天的溫度則為23~29°C，我認為，這樣的可信度還是有的，因為室外本來就會比室內還高溫，而且溫差並不是太大。而因為最近都在下雨，天氣涼爽，所以警示燈當然是亮起了綠燈，也就是說這樣的溫度不太對人體有害。



▲ 測量室溫

▼ 手機顯示量測當下室外的溫度

接著，因為最近都在下雨，所以溫度難以達到 30°C 以上，但是為了測試當溫度高達 30°C 以上時，警示燈還是會正常亮起紅燈，所以我利用吹風機加熱到 30°C 以上，如影片二及下圖所示，紅燈亮起了，這樣正如我所預期的:我們可以及時發現天氣非常炎熱，必須要打開冷氣，以防中暑!但因為我這次是利用吹風機加熱，所以當關閉吹風機後，溫度會快速下降。最後降到 30°C 以下後，紅燈會切換至綠燈，如同影片二所呈現的。



▲ 利用吹風機使室溫上升到 32.40°C(警示燈亮起)

八、心得

一開始,覺得只要有想法之後,要自己做一個專題其實並沒有很困難,但真正實踐時，我才知道，要把我的想法做成我想要的東西，其實並不是件簡單的事情，從深入了解熱敏電阻及七段顯示器的原理，到如何修改程式使之符合我的要求,這些其實都是花費了我極大的心力及

時間，但是真的學到了許多東西，而且還做出了我所想的東西，這其實讓我小有成就感。雖然我這次使用的元件大部分都是我們未曾接觸過的，但其實在課本中，我們曾接觸過一些相關知識，這讓我非常想實際使用這樣的元件!而以下我想分享一些關於我遇到的困難與解法。

首先是熱敏電阻，一直以來，我都以為 A1 腳位讀出來的就是溫度值了，但當我在測試我買到的是 NTC 還是 PTC 時，序列埠顯示出來的居然是高達 400 多的數值，這當然不會是溫度!若這是溫度，那我們應該已經熱死了吧!接著，我就在想:既然不是溫度那麼就是阻值了吧，後來研究了資料後，才知道讀到的輸出必須先除以 204，而且除完後的數值並不是阻值，而是電壓，接著，還必須透過分壓比才能算出熱敏電阻的阻值，當然，算出的阻值是會隨溫度變化而改變，再透過算出的阻值我們才能求得實際溫度，而且，公式還包含了熱敏指數，在我們購買熱敏電阻時，店家並不會標示，而是要我們自行研究 datasheet，去找出此型號熱敏電阻的熱敏指數為何。

然後關於七段顯示器的數值顯示我也是研究了許久，後來才發現原來是用十六進位數值顯示。可是問題又來了，我們要如何透過 a, b, c, d, e, f, g, dp，湊成我前面所打的表格的輸出數值呢?後來才發現要把 dp, g, f, e 分成一組，而剩下的 d, c, b, a 分成一組，兜成十六

進位的輸出，但是我所看到的資料中，都是只有 0~9 的輸出，並沒有包含小數點的輸出，所以我自己研究出了如何同時輸出數字及小數點，其輸出表格如下所示:

數字	dp	g	f	e	d	c	b	a	十六進制數值
0	1	0	1	1	1	1	1	1	0xBF
1	1	0	0	0	0	1	1	0	0x86
2	1	1	0	1	1	0	1	1	0xDB
3	1	1	0	0	1	1	1	1	0xCF
4	1	1	1	0	0	1	1	0	0xE6
5	1	1	1	0	1	1	0	1	0xED
6	1	1	1	1	1	1	0	1	0xFD
7	1	0	0	0	0	1	1	1	0x87
8	1	1	1	1	1	1	1	1	0xFF
9	1	1	1	0	1	1	1	1	0xEF

▲ 包含小數點的數字 0~9 的七段顯示器十六進制數值表

而且我們取出的數字只會是 0~9，所以如果我們要顯示小數點的話，就必須將陣列中的 index 加 10。

而雖然我們可以透過視覺暫留，看到 4 位數同時顯示，但這也代表，

很快的，這些數字就會消失，這也讓我頭痛非常久，後來我才發現:如果使用 do、while 就可以讓數字持續顯示。

透過這次的專題，讓我發現:其實生活中的許多小東西，我們都可以藉由 Arduino 實做出來，如此一來，我們便可以不必花錢去購買一些我們並不是很想購買但卻必須使用的物品。

九、參考資料

1.全民自造與程式設計：使用 Arduino

（泉勝出版有限公司）

2.熱敏電阻(維基百科)

https://zh.wikipedia.org/zh-tw/%E7%83%AD%E6%95%8F%E7%94%B5%E9%98%BB