第六屆旺宏科學獎

成果報告書

參賽編號：SA6-308

作品名稱：彩光魔棒，微機電加速度計的生活應用

姓名：鄭期傑

關鍵字：加油棒、加速度計、MEMS

目錄

1. 研究題目

彩光魔棒，微機電加速度計的生活應用

2. 研究動機

在各種演唱會中，觀眾們都會帶著綠色的螢光棒去捧場，隨著歌曲的節奏左右搖擺，製造氣氛、表達個人的心情。但是我覺得這種使用螢光染料的加油棒，色彩太過單調，又缺乏亮度及閃爍的變化。而且螢光棒雖然有幾種顏色，但是都是像綠色、藍色之類的冷色系，不能是當的表達興奮的心情，因此計劃製作一支能隨著揮舞動作動態變幻顏色及模式的加油棒。

3. 研究目的

使用安置在棒頂端的微機電(Micro-Electro-Mechanical Systems)加速度計(Accelerometer)偵測加油棒的揮動情形(方向、速度、來回頻率等)，經過微控制器(Microcontroller)處理，然後在一組 16 x 4 的高亮度 LED 矩陣上依照揮動的動作顯示各種對應動態圖形(Pattern)。這一組 LED 矩陣由藍、綠、黃、紅四種顏色的 LED 各 16 顆組成。

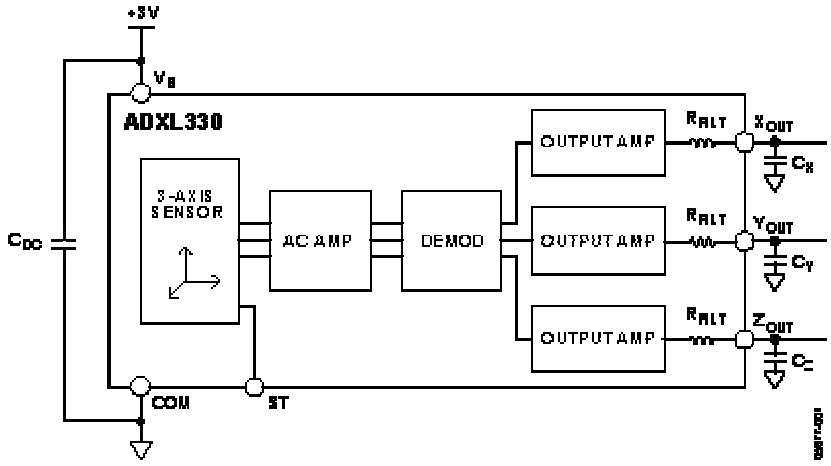
4. 研究過程

彩光魔棒的構思完全於建構一個會隨著揮舞的快慢、動作的方向以及節拍的變化而改變閃燈模式的產品上，因此主要的研究構想如下：1)揮舞動作的偵測，2)揮舞動作的歸類，3)LED 閃爍 Patterns 的構想，4) 硬體製作，5)程式開發。

(1) 揮舞動作的偵測

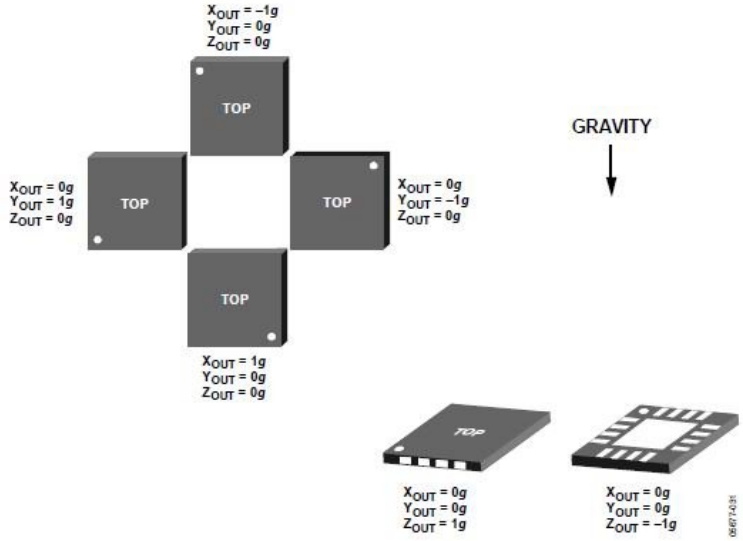
必須使用小型的加速度計，裝在棒子前端，隨時量測棒子在三度空間中的加速度，把數值傳給微控制機，來研判揮舞者實際的動作。

市面上現在有好幾種用來測定加速度的微機電積體電路，例如在任天堂的 Wii 遙控器中使用的是 Analog Device 公司生產的 ADXL330 Accelerometer 積體電路，這是一個以 MEMS(Micro-Electro-Mechanical Systems) 技術製造的



圖一、ADXL330 Accelerometer 功能方塊圖

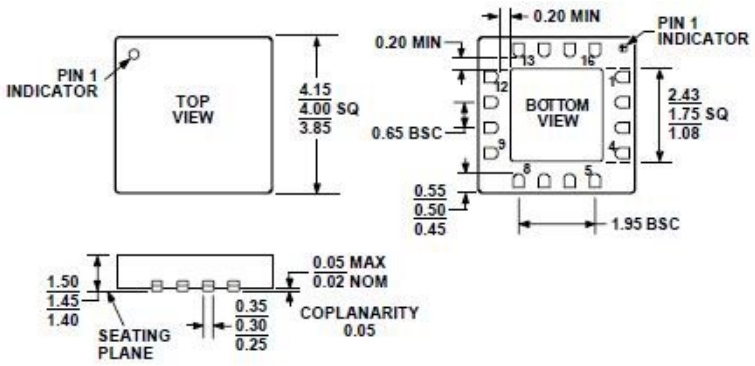
加速度感測器，裏面為奈米尺度的機械式加速度計與量測電路，有 X,Y, Z 三軸輸出。當供電 DC 3V 時, 它的任一軸在 0g 的時候會輸出 1.5V 的電壓, 而測量到的加速度每 +/-1g 就會有對應的 +/-0.3V 電壓變化 (+1g 時輸出 1.8V，-1g 時輸出 1.2V，+2g 時輸出 2.1V，最高在+3g 時可輸出 2.4V)。



圖二、ADXL330 加速度計在各種放置方向時的輸出信號

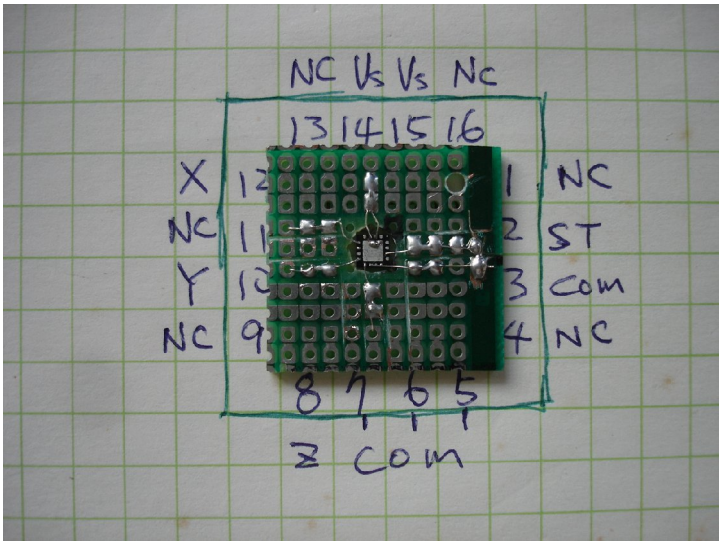
由於買不到 ADXL330 Accelerometer 積體電路的延伸印刷電路版(Breakout Board), 於是在這兩個作品製作的過程中，最困難的、同時也是兩者共通的問題便是將這塊積體電路焊接到電路板上。ADXL330 Accelerometer 積體電路是表面封裝型(SMD), 而且焊點在正下方完全被它自己蓋住, 整塊積體電路只有 4mm * 4mm 的大小卻有 16 個焊點，每一個焊點的尺寸為 0.3mm x 0.3mm，間距也是 0.3mm，極不適合人工錫焊。如果直接在這塊積體電路上用銲錫銲線的話, 幾乎一定會把積體電路本身的數個焊點銲在一起而導致短

路。



圖三、ADXL330 Accelerometer 尺寸圖

但是如果把 ADXL330 Accelerometer 積體電路放在電路板上再用夠細的線和很少的銲錫去銲的話，由於積體電路的焊點會比電路板上的銅箔高，所以很容易因為一個微小的碰撞導致銲點甚至整個積體電路脫落。為了解決這個問題，我預先在電路板上鑽了一個直徑 5.6mm 的小洞後再把 ADXL330 Accelerometer 積體電路崁進去，這樣一來積體電路的銲點和電路板的銅箔就幾乎是在同一個平面上了。把積體電路固定好之後，我先剪一段較長的極細鍍銀銅線，把其中一端銲在電路板上，然後再視銲點到積體電路的腳之間的距離把線剪短到剛好在銲點上。接著我在線端燙上一層薄薄的銲錫，把線跟積體電路上要銲的銲點對好之後再用烙鐵壓一下，完成焊接。至於為什麼先把線銲在電路板上，那是因為積體電路上的銲錫很少，如果先把線銲在積體電路上再拿線的另一端去對銅箔上的銲點的話會很容易被扯下來，不然就是在銲接銅箔端時因為線被加熱而導致線從積體電路上脫落。銲好的成品如下圖，每條線在靠銅箔那邊的都刻意上了很多銲錫作為熱庫，以免在銲另一頭時導致這邊的錫融化。完成後的電路板如下：



圖四、將超小的加速度計 IC 焊到延伸版上

(2) 揮舞動作的歸類

本來是想寫一支程式來進行揮舞動作的歸類, 然後看當時的動作是屬於第幾類就顯示對應的 Pattern。但是經過實際測試後, 發現因為不同人的臂長、揮舞快慢等都有很大差異, 所以目前採用的每一個動作的辨識, 都是用實測數字加上一些寬裕值, 算是程度很有限的歸類。

(3) LED 閃爍 Patterns 的構想

LED 閃爍 Patterns 的構想, 應該算是比較主觀的部份。彩光魔棒有藍、綠、黃、紅四條各 16 粒 LED, 依據色彩學, 藍色及綠色等冷色系的顏色較象徵憂鬱、安靜的情緒；而紅、黃色等暖色系的顏色則令人感受到明快與喜悅的心情。如果再搭配上時快時慢的閃爍節拍, 可以有非常多的變化。

(4) 硬體製作

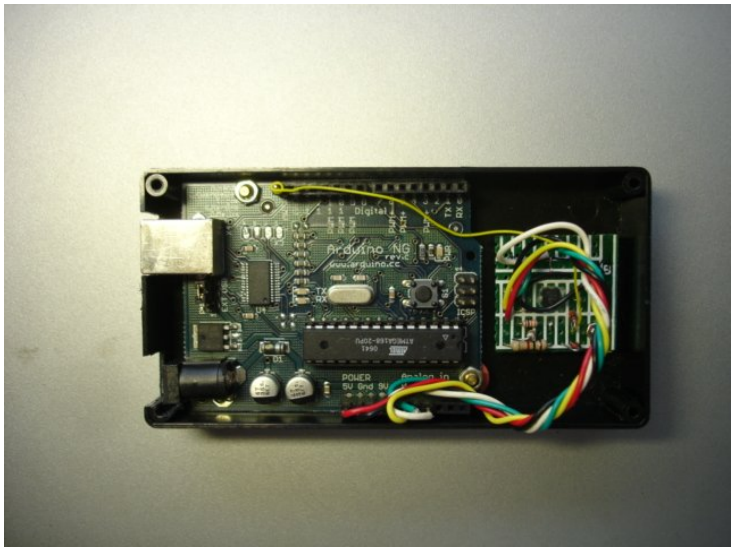
首先就讓加速度計在受重力 1g 的狀態下進行靜態測試, 而這些數值同時也用來當作實驗中用以參考的基準值。因為 x,y,z 三軸的讀數都會隨著 ADXL330 晶片的供電電壓以及模擬／類比轉換(Analog to Digital Converter)的參考電壓變動而改變, 所以採取了兩個方式來確保數據的一致性：1) Analog to Digital Converter 的參考電壓來自 ADXL330 晶片的供電電壓, 兩者連動。2)整個實驗過程中都採用同一個加速度計及微控制機。靜態測試的結果如下(表中的數字是加速度計電壓輸出經過模擬／類比轉換後的值)：

	-1g	每 1g 增減量	0g	每 1g 增減量	+1g
X 軸	509	(125)	634	(130)	764
Y 軸	518	(132)	650	(129)	779
Z 軸	518	(122)	640	(128)	768

在靜止時, 讀數都只有 +/-1、2 的上下浮動, 但是可以看到三個軸的輸出在同樣條件時並不完全一致。查過 Analog Device 公司的規格後, 發現 ADXL330 有 +/-10%的誤差, 所以我們測到的數據符合規格。

以上的測試是將加速度計的 x,y,z 三軸電壓輸出, 接到 Arduino 模組的三支模擬信號輸入腳(analog input pin)後, 以程式讀取三軸的加速度, 再從 USB 介面送到個人電腦。個人電腦上用 Serial port monitor 程式記錄, 為了不同的需要, 這次使用了 NetTerm 及 COM Port Toolkit 兩種程式。

整個研究過程中，包括了各種測試以及最後成品所使用的微控制機都是 Arduino 模組。Arduino 模組是一塊基於開放原始碼的原形平臺(Prototyping Platform)，它的微控制機採用的是 Atmel 公司的 ATMega168，具有 20MHz 的 RISC 核心，大多數指令都可在一個鐘序內完成，也就是說有將近 20MIPs 的計算能力。Arduino 上有 14 個數位輸出/入接點 (Digital I/O pins)、6 個 Analog 輸出/入接點(Analog I/O Pins)及 6 組模擬/數位轉換器(Analog/Digital Converter)，程式語言來自 Wiring(wiring.org.co)，開發環境(IDE)是由 Processing(www.processing.org)移植而來。語言與 C/C++ 相似，支援完整的 C 語言架構及部份的 C++功能，Arduino 可以單獨當作一個微控制機使用，也可以透過序列介面或 USB 與 PC 中的軟體如 Flash, Processing, Max/MSP 等協同工作，創造更多用途。



圖五、USB 介面的 Arduino 模組，右邊小塊的是加速度計

我選用 Arduino 的幾個主要原因是，1) 具有 Analog to Digital Converter，可以用來轉換加速度計 ADXL330 的電壓輸出為數據，2) 程式編修非常容易；3) 運算速度很快； 4) 直接自 USB 取電這點非常方便，當然如果要接上的設備耗電很大時, 還是要用外接電源;5) 有 16K 的 Flash 放程式, 就算 bootloader 佔去 2K 後也還有 14K, 對於不太會精簡程式的初學者來說是一大福音;6) 我想練習製作一些人機互動的游戲，而這正是一個很好的入門練習工具。

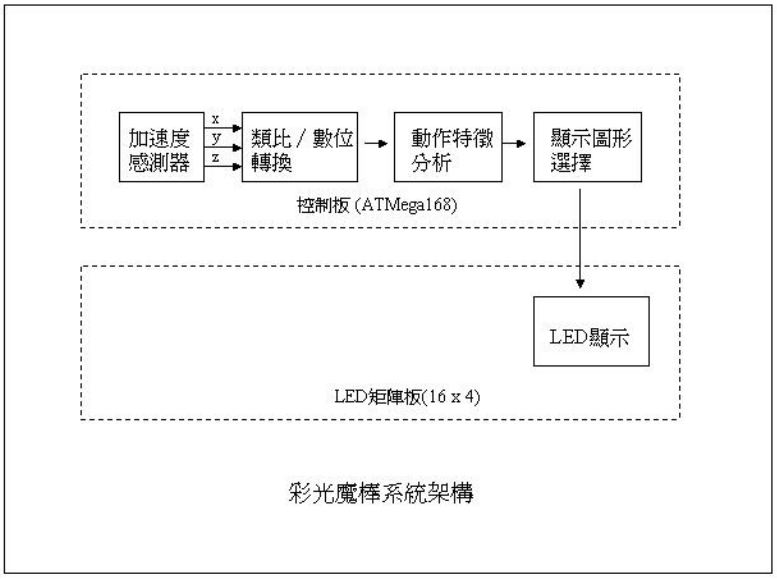
當然，Arduino 也不是完全沒有缺點的。在開發的過程中，我覺得最不方便的就是它沒有模擬器功能來單步運作(single step)及檢視程式參數，偵錯比較難。

決定好了設計原則後，因為採用微控制機，電路部份不會太複雜，所以以功

能取向分為兩部份：1)LED 顯示板，2)控制板。未來有新構想或新的零件出現時，這兩個部份可以單獨更換。

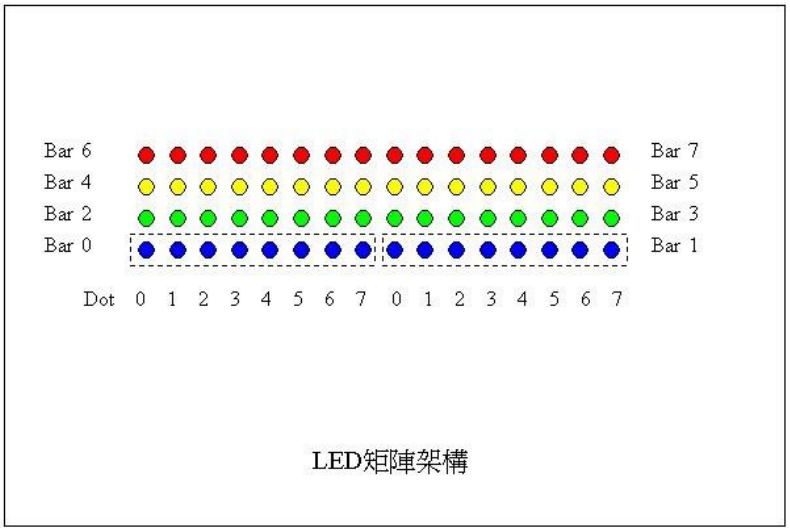
LED 顯示板採用現成的萬用 PCB，跳線焊接成 8 x 8 的矩陣，這是所有電路工作中最耗時費力的部份。控制板部份採用的是 Arduino 模組，它的 Micro Controller 是 Atmel 公司的 ATMega168-20，在測試時候是直接使用 Arduino 模組，但是另外用萬用實驗 PCB 作了一片同樣的板來替換。主要原因是 Arduino 模組比較貴，而且要向美國網購。其實電路很單純，自己作一片大約只需要三分之一的成本，而 Arduino 模組在開發完成後就可以拿去作別的用途。

整個彩光魔棒的架構如下：



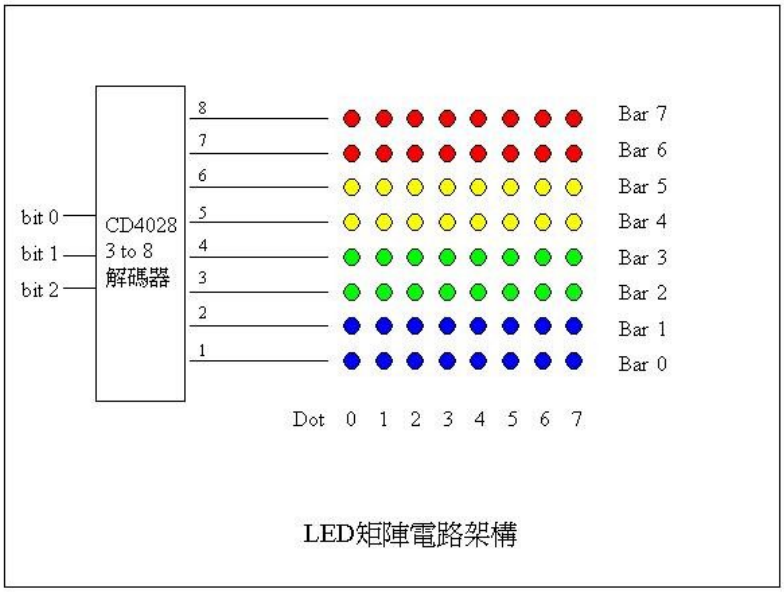
圖六、彩光魔棒系統架構

LED 顯示板的架構如下：



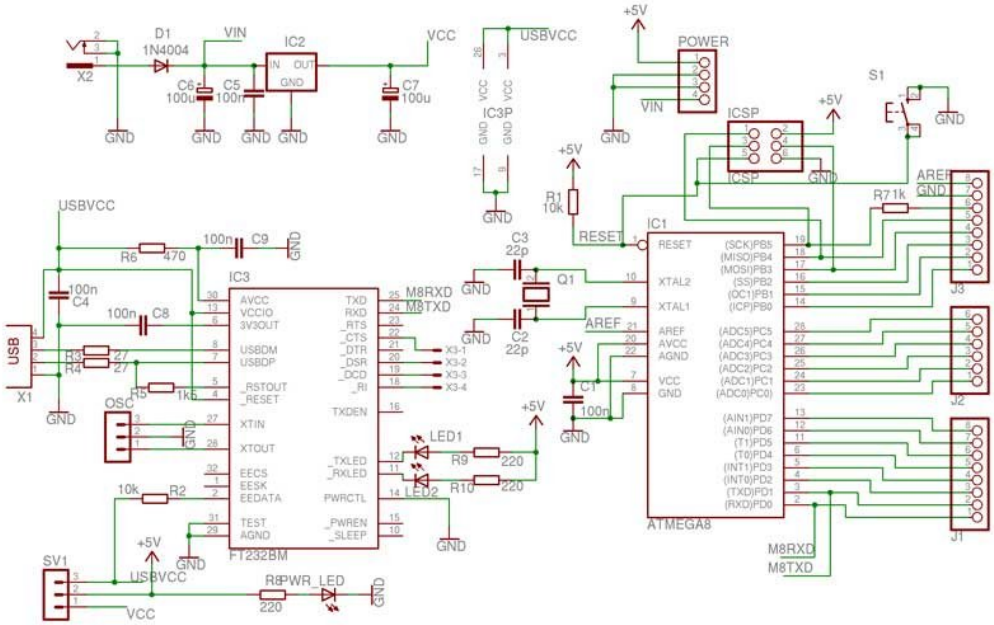
圖七、LED 矩陣排列圖

LED 顯示板上共有 64 粒 LED，實體上排成 16x4 的矩陣(藍、綠、黃、紅各 16 粒)。但是為了在微控制機處使用最少的輸出針(pin)，電路上是連接成 8x8 的矩陣，分為 bar 0-7 以及 dot 0-7，這樣只需 16 支 digital output pins 即可。不過 Arduino 模組也沒有這麼多可用的 digital output pins，想到在 bar 0-7 的掃描中，永遠只有一個是 on 的 (dot 部份同時可能會有 0 到 7 個要亮)，所以在 Arduino 模組這邊就採用 3 支 digital output pins 輸出 0-7 的 binary code，然後在顯示板上用一個 3 to 8 的解碼 IC(CD4028)，用 binary code 控制 8 支輸出針的 1 支為 on，解決了 digital output pins 不夠的問題。



圖八、LED 矩陣的電路架構

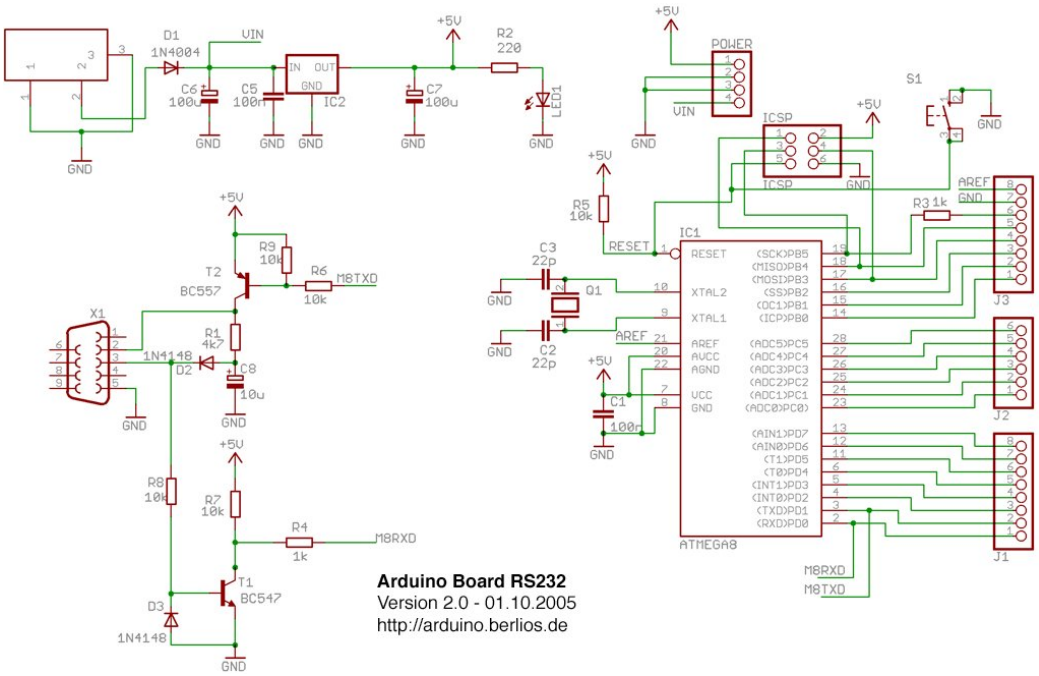
控制板在開發階段用的是買來的 Arduino USB 模組，它的電路如下：



圖九、Arduino USB 模組電路圖(Open Project)

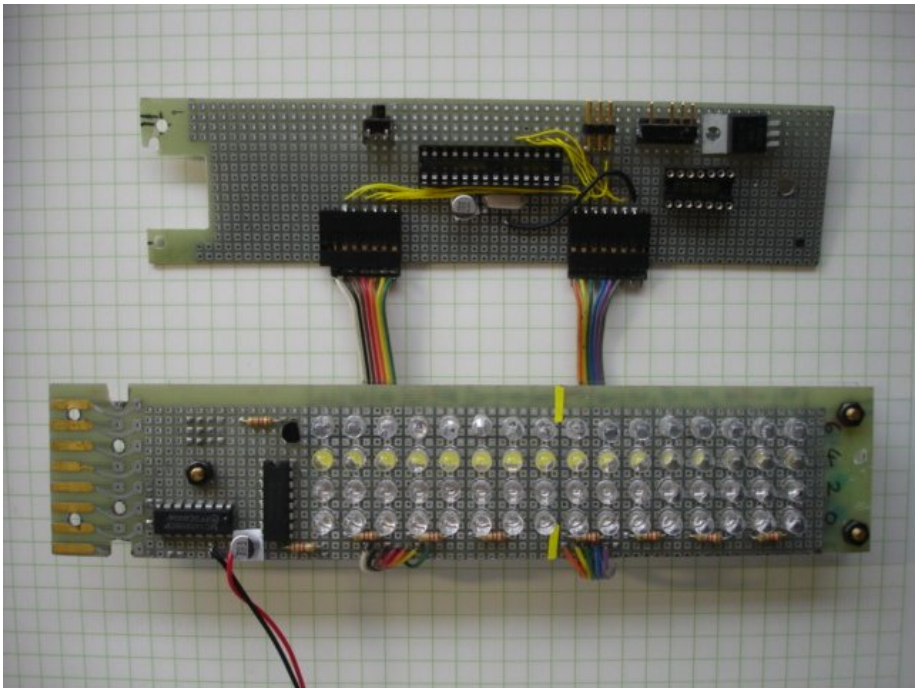
完成的彩光魔棒上，用的是自己依照 Arduino RS232 作的微控制機板，架構

相同，只有原來的 USB port 省略掉，改用 serial port 作 programming 與通訊之用。



圖十、Arduino RS232 模組電路圖(Open Project)

這是完成後的照片，上面是控制板(IC 尚未插上)，下方為 LED 顯示矩陣板，還沒組合起來。



圖十一、上面是控制板(IC 尚未插上)，下方為 LED 顯示矩陣板。

圖十二、組合在一起的照片，6 顆 3 號電池裝在下方握把中。

(5) 程式開發

Arduino 的程式架構分為 5 部份：

```
int x;                          // variable declaration(變數宣告)

void functionname(value);       // function declaration(函數宣告)

void setup( ){                  // initialization(啟動、只執行一遍)
}

void loop( ){                   // execution(持續運轉在這個迴圈中)
}

void functonname(value){        //user defined function(函數宣告)
}
```

在變數宣告、函數宣告後的 setup( ) 指令集，只會在開機後執行一遍，然後就進入 loop( ) 指令集，持續運轉在這個迴圈中，自己宣告的函數放在最後，這樣就是一支完整的程式。

彩光魔棒程式的架構分為四部份編寫與測試：

- 讀加速度計
- 動作分類
- 選用閃爍 Pattern
- LED 掃描顯示

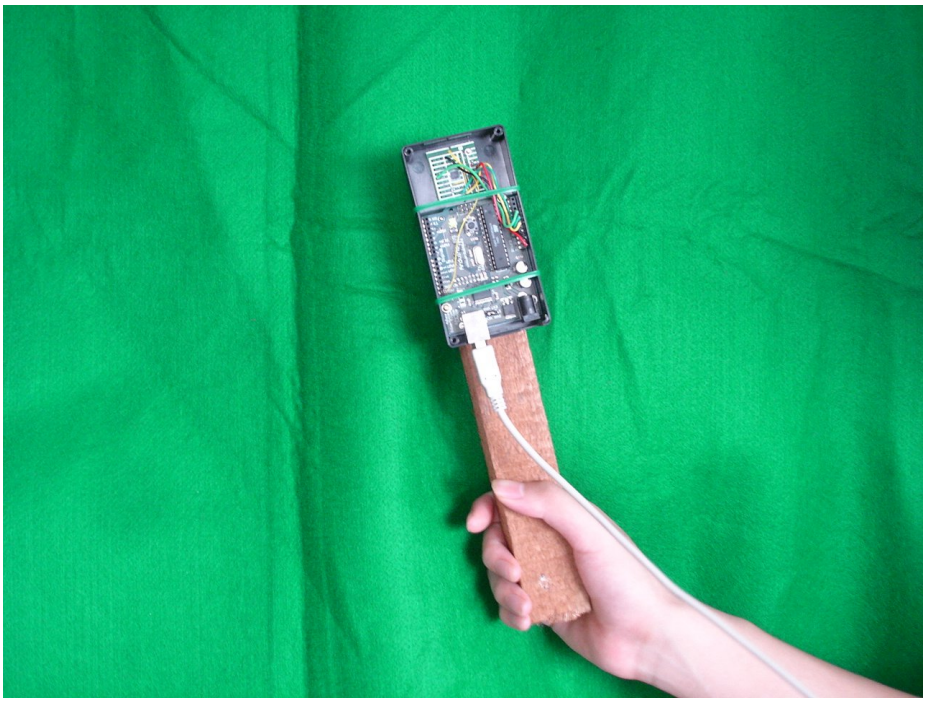
(1) 讀加速度計, 在開始測試加速度計時, 就已經寫好了從 Analog input pin 讀加速度計電壓輸出的程式如下：

```
// read from analog pins 0,1,2 connected to x,y,z output of ADXL330

#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

int potPinX = 0;
int potPinY = 1;
int potPinZ = 2;
int valX = 0;
int valY = 0;
int valZ = 0;
char com = ',';

void setup ()    {
            // set ADC reference voltage to AREF pin
            cbi(ADMUX, REFS1); //clear REFS1 bit (0)
             cbi(ADMUX, REFS0); //clear REFS0 bit (0)
   Serial.begin(9600);
}

void loop () {
   valX = analogRead(potPinX);
   valY = analogRead(potPinY);
   valZ = analogRead(potPinZ);
   Serial.print('(');
   Serial.print(valX);
   Serial.print(com);
   Serial.print(valY);
   Serial.print(com);
   Serial.print(valZ);
   Serial.println(')');
   delay(36);
}
```

這段程式是將 x,y,z 的值轉換後, 從 COM Port 送出, 但原始的值也可在 valX,valY,valZ 中取得。

(2) 動作分類(Motion Classification)

動作分類的程式非常簡單, 要花很多時間思考的是要如何找出想要拿來作為觸發條件的動作的特徵, 並以數字記錄起來。為了在彩光魔棒尚未完成前, 可以同時進行揮舞動作的數位資料的記錄及分析, 所以將 Arduino 模組與加速度計一起貼在一支與彩光魔棒同長(28cm)的木棒的頂端上, 用上面那支程式將揮動木棒時加速度計的讀數傳回 PC, 在 PC 上用 NetTerm 程式記錄收到的資料與時間。每筆記錄之間的時距是 50mS, 用這個方法對幾種揮動模式加以記錄。



圖十三、用來收集揮舞動作數位資料的設備, 全長 28cm 與成品相同

例如最常見的, 將螢光棒高舉過頭、左右慢速來回揮動的動作, 我將其訂名為 Type01 動作, 加速度計傳回的數據如下(只揮了一次)：

	X	Y	Z
1	555	550	665
2	558	549	659
3	559	549	661
4	559	549	665
5	559	548	657
6	556	549	660
7	558	549	665
8	559	548	662
9	561	547	659
10	557	548	666
11	553	549	661
12	552	550	667
13	559	544	659
14	565	546	656
15	557	550	667
16	552	551	667
17	559	549	668
18	553	547	662
19	553	547	668
20	552	548	662
21	554	550	668
22	558	548	669
23	559	548	671
24	557	547	668
25	562	546	672
26	567	543	675
27	571	541	671
28	571	542	666
29	584	541	675
30	589	538	671
31	592	538	672
32	594	539	667
33	601	539	661
34	602	535	650
35	625	541	660
36	660	540	665
37	671	536	648
38	684	539	638
39	698	548	640
40	750	557	636
41	761	565	646
42	754	562	641
43	738	564	630
44	750	571	636
45	766	572	626
46	764	577	640
47	770	577	629
48	768	579	640
49	756	575	637
50	749	570	636
51	733	567	644
52	739	565	633
53	737	562	642
54	729	558	644
55	719	552	642
56	707	548	648
57	708	541	643
58	697	537	650
59	685	535	645
60	676	532	654
61	671	530	650
62	657	534	660
63	630	532	660
64	623	535	669
65	618	536	667
66	615	537	665
67	596	540	665
68	581	543	667
69	572	551	668
70	540	555	657
71	521	563	658
72	515	567	669
73	522	563	669
74	534	559	651
75	540	555	654
76	549	553	660
77	548	555	660
78	551	554	654
79	558	555	656
80	555	551	658
81	561	551	652
82	551	551	654
83	549	552	653
84	547	555	656
85	549	557	659
86	552	556	652
87	556	551	649
88	552	552	656
89	548	554	649
90	550	552	656
91	550	553	650
92	550	554	650
93	550	555	657
94	556	555	651
95	546	554	658
96	555	554	649
97	544	554	652
98	548	552	655
99	555	553	648
100	554	552	652

圖十四、高舉過頭，左右慢速揮動一次時，加速度計傳回的數據

同時也記錄下來彩光魔棒在左右定點靜止時的回傳的值為：
左 = (703,535,645)
右 = (590,526,643)
因為加速度計永遠受到重力的作用，而會在 x,y,z 方向輸出對應的重力分量值，所以要先紀錄一下當彩光魔棒在左右兩點靜止時回傳的值，也就是棒端在那兩點的重力分量。因為考慮到每一個人在握棒與揮動時的姿勢、角度及速度等差異很大，所以對於動作的分類是採用動作前後數劇產生了幾個數字的差值為依據。差值越大，表示揮動的動作快(加速度大)，或是旋轉的角度大(重力分量之改變大)。另外要考慮的是揮動造成的加速度會與當時的重力加速度分量加起來，但是我們並不知道那一瞬間加速度計在空間中與 x,y,z 三軸的夾角，所以不可能把重力分量的值抽取掉。

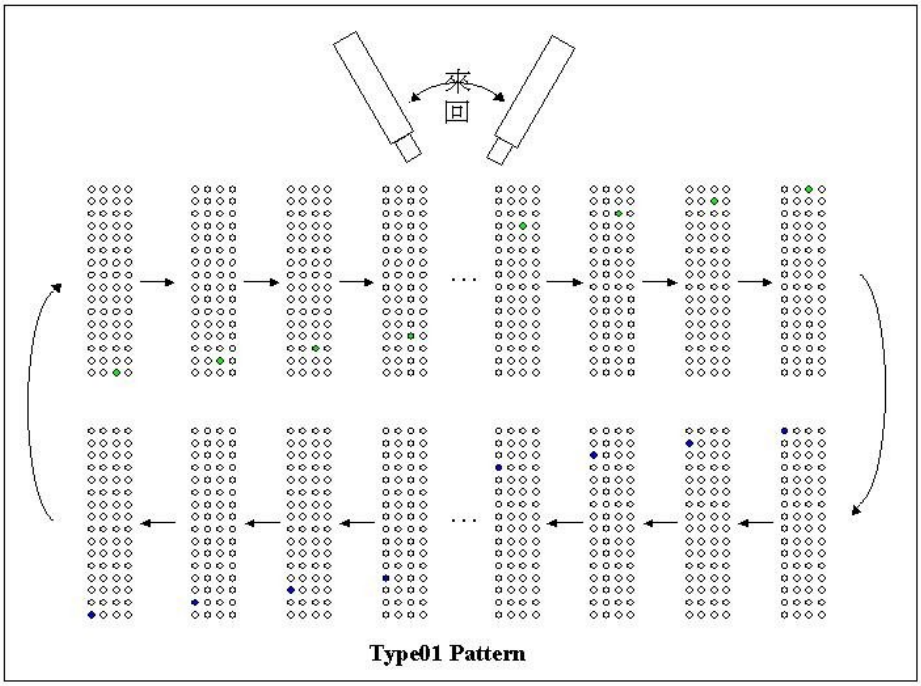
根據這些原則來檢視上面的表格，可以確認從第 1 筆到第 26 筆資料間加速度計在右上方，因為人手不可能握著它完全靜止，所以有點晃動；從第 27 筆到第 47 筆資料間，棒端在向左作曲線運動，因為有 20 筆資料，我們的取樣間距為 50mS，所以自右揮到左為 1 秒；從第 47 筆到第 72 筆資料間，棒頭向右揮動，費時 1.5 秒。

這次總共做了 4 種揮動 Pattern 的記錄與特徵分類，另外 3 種是：type02-木棒頭朝上，向下作 90 度揮動，類似剁肉動作；type03-木棒水平，作垂直上下移動；type04-木棒垂直，作左右水平揮動。

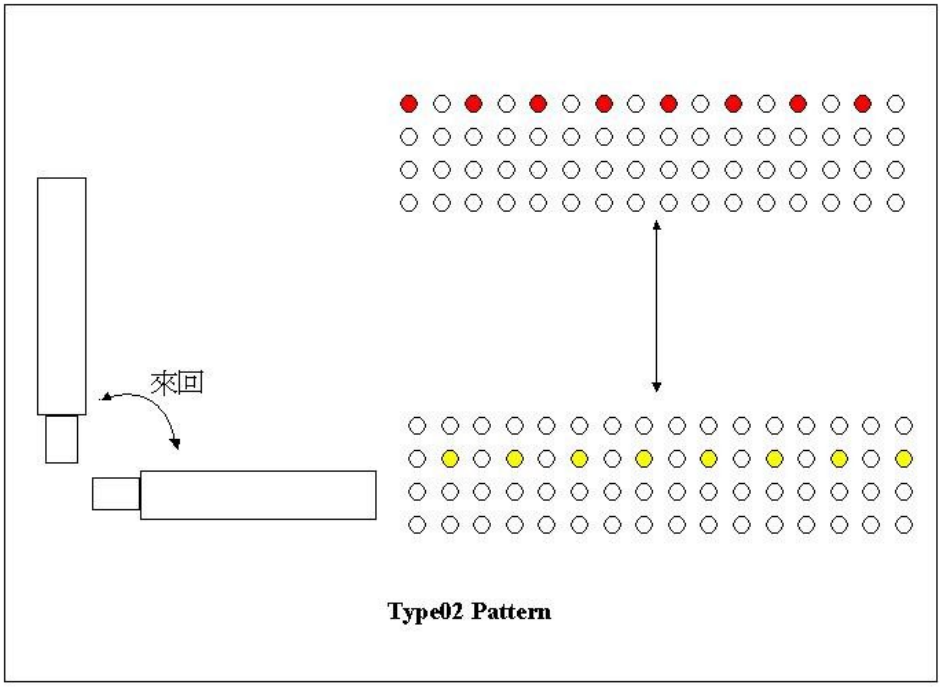
在現在這個剛開始構想的階段，為了使得加速度資料的變化不要太大，刻意選用矩形的握把，而且 LED 只有一面，以迫使使用者只有一種握法。還有一個重要的考慮就是如果要在揮棒動作中(而不是在揮棒動作完成後)就開始變更閃爍 pattern(這是比較合理的設計)，就只能取用該動作的前幾筆資料用以動作分類，否則閃爍 pattern 的變換會與動作的變換脫節。

(3) 設計閃爍 Pattern

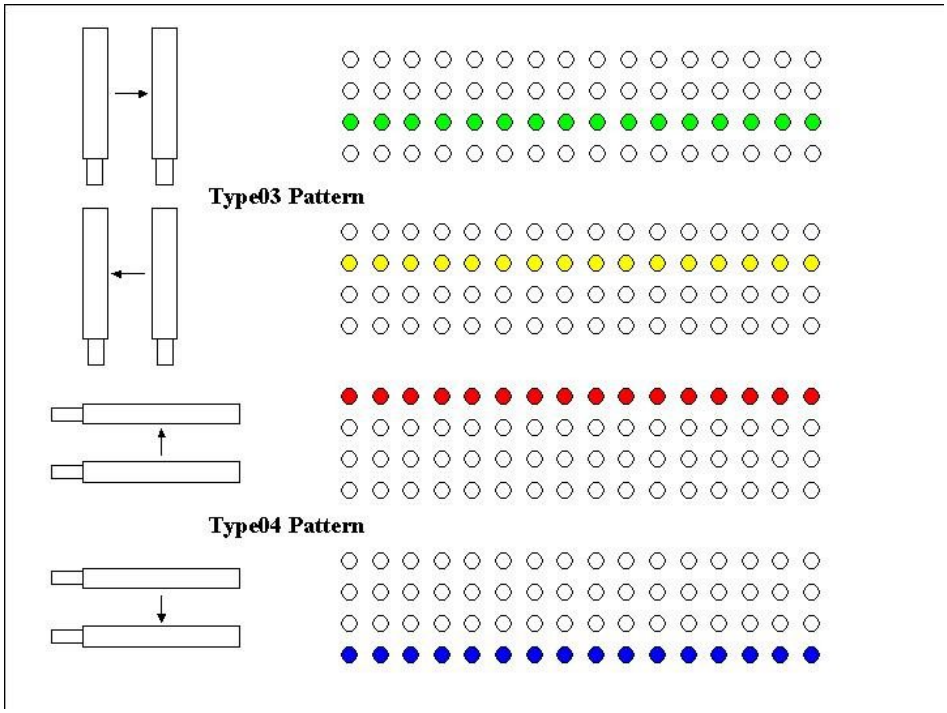
這個研究的性質應該算是一個 proof of concept，所以只設計了 4 套動態 pattern，配合 4 種動作分類，及一個靜止 pattern，供彩光魔棒在靜止時使用。



圖十五、Type 01 揮舞動作及關連 Pattern



圖十六、Type 02 揮舞動作及關連 Pattern



圖十七、Type 03 及 04 揮舞動作及關連 Pattern

為了節省變數空間，顯示 Pattern 盡可能使用 bit shift 來作，避免儲存太多常數，這樣就可以存儲很多顯示 Pattern。

例如：

以下這段程式可以產生一粒紅色的 LED 每 200mS 向左移一格，同時有一粒藍色 LED 每 200mS 向右移一格的效果。

```
        :
        :
unsigned int color[4] = {
   0,0,0,0};    //array for blue, green, yellow, red      <= (4 條 LED 亮燈的初始值)
        :
        :
long previousMillis =0;
long interval = 100;
long previousMillis0 =0;
long interval0 = 200;
long previousMillis1 =0;                                  <= (每條 LED 跳動的時距)
long interval1 = 100;
long previousMillis2 =0;
long interval2 = 200;
long previousMillis3 =0;
long interval3 = 200;
        :
        :
        :

void setup() {
        :
```

```
      :
      :
void loop() {
        :
    if (millis() - previousMillis > interval) {    <= (計時)
      previousMillis = millis();
      color[3] = color[3] << 1;          <= (紅色 LED 向左移一格)
      if (color[3] == 0){                ←(走完一遍？)
         color[3] = 1;                    <= (紅色 LED 在右端亮一粒)
      }
  }
     color[0] = 1;
      if (millis() - previousMillis0 > interval0) {    <= (計時)
      previousMillis0 = millis();
     color[0] = color[0] >> 1;          <= (藍色 LED 向右移一格)
    if (color[0] == 0){                 ←(走完一遍？)
     color[0] = 16384;                  <= (藍色 LED 在左端亮一粒)
    }
     }
```

顯示 pattern 的程式可說是非常容易，pattern 本身的設計才最重要，彩光魔棒的每一個亮點與亮點移動方式，都是由程式產生，所以動態 pattern 設計的彈性可說是無限度。

(4) LED 掃描顯示

LED 掃描顯示完全是用程式進行的，在 LED 矩陣上，依電路分為 8 個 bar，每個 bar 內有 8 個 dot，構成 8x8 的矩陣，所以需要有 bar 0-7 及 dot 0-7 的信號。

最初的設計是對每一個 bar，依序設定 dot 0-7，然後送出那一 bar 的點亮信號，再進入下一 bar，不斷重覆相同的動作。參考了幾本有關 8051 微控制機的書，在講到用掃描式推動 7 節式 LED 數字模組時，程式中都將每一 LED 模組點亮的時距定在 20-40mS，因為人眼的視覺暫留約在 40mS。但是側試過後觀察到 LED 在這個時距下有明顯的閃爍現象，所以就以每次減半的方式來修改掃描的時距。最後的設定是 1mS 是最佳的數值，完全看不出任何閃爍，

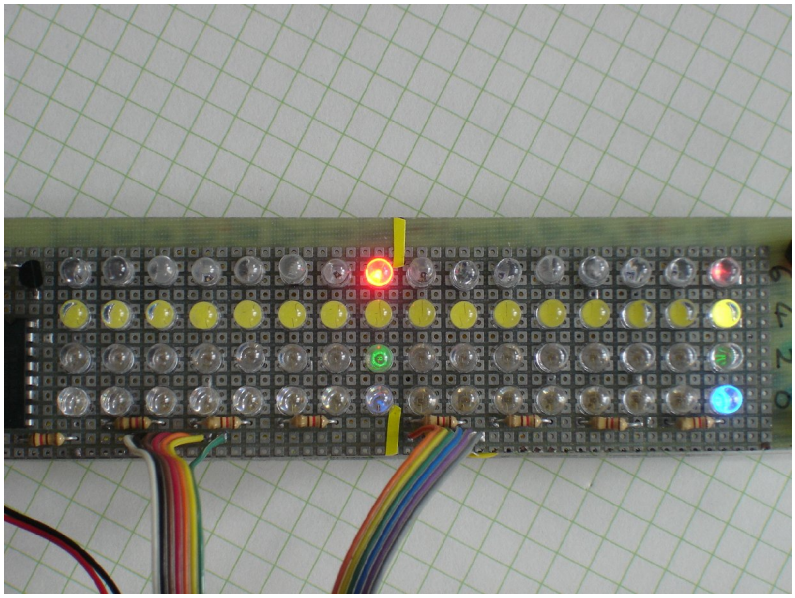
以下是程式中作 LED 掃描的片段：

```
void loop() {
   .
   .
   .

   motionClassify
   cleardigitalports();
```

```
// Turn on dots 0-7 related to bar #
   for (int bar = 0; bar < 8; bar++){          // display start
   int colorIndex = bar >> 1;
   unsigned int colorNow = color[colorIndex];
   if ( bar == ((bar >> 1) << 1)){             // if bar = 0,2,4,6
      for ( int x = 0; x < 8; x++){                  // check dots in bar 0,2,4,6
         if (colorNow & (1 << x)) {
            digitalWrite(pin2 + x , HIGH);
         }
         else
         {
            digitalWrite(pin2 + x, LOW);
         }
      }
   }
   else
   {
      unsigned int colorHigh = colorNow >> 8;
      for ( int x = 0; x < 8; x++){                 // check dots in bar 1,3,5,7
         if (colorHigh & (1 << x)) {
            digitalWrite(pin2 + x , HIGH);
         }
         else
         {
            digitalWrite(pin2 + x, LOW);
         }
      }
   }



// Turn on bar 0 -7
      for (int y = 0; y < 3 ; y++){      // bar scan start
         if (bar & (1 << y)){
            digitalWrite(select0 + y, HIGH);
         }
         else {
            digitalWrite(select0 + y, LOW);
         }
      }                                          // bar scan end

   }                                                    // display end
   delay(delayTime);
}

void cleardigitalports() {                 // clear display start
   int pin;
   for (pin=2; pin <= 9; ++pin) {
      digitalWrite (pin, LOW);
   }
}                                                  // clear display end
```

圖十八、LED 掃描顯示程式，先送出 dot 0-7 的點、滅信號，再啟動所屬的 bar 迴路，那 8 粒 LED 會持續亮 1mS，然後換下一條 bar 上的 LED 點亮，實際上，永遠同時最多只有 8 粒(1 bar)LED 在亮，1mS 後換到下 8 粒(next bar)。

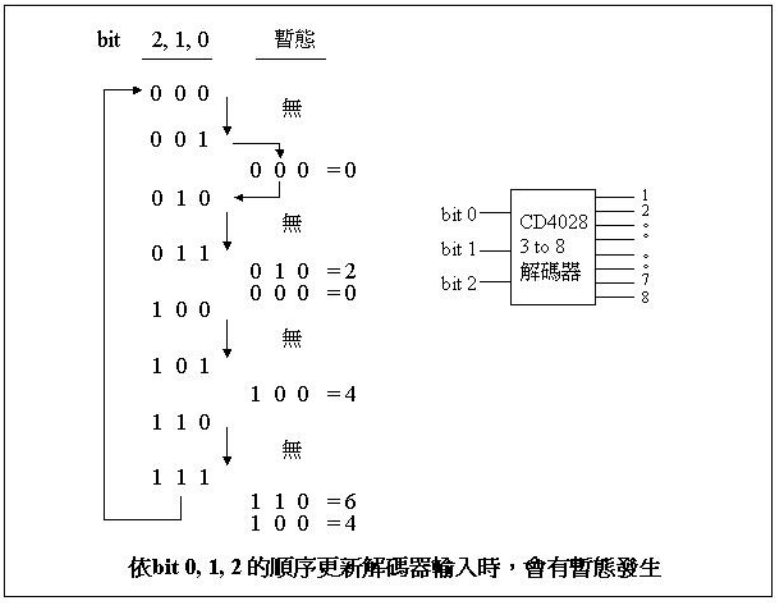
在我所編寫的顯示程式成功運作後，我發現了一個怪異的現象：當 bar 0 上的 dot 0 LED 發亮時，有好幾個其他 bar 上相同 dot 位置的 LED 也會亮，但是亮度低很多。雖然懷疑是程式有問題，用了不少方式重寫，但最多只是亮點換到別處去，問題並沒有改善。



圖十九、當右下角的 bar 0 之 dot 0 點亮時，其它 bar 上相同 dot 位置的 LED 也有一點亮

圖二十、以示波器觀察 bar 0 的信號，在正常脈波後，多了兩個極短的脈波

在用示波器看過 bar 與 dot 的脈衝信號後，觀察到在 bar 0 的線路上，原本應該是 1ms High 及 7mS Low 兩種脈衝信號交互出現，但是在那 7mS Low 狀態的週期中，在固定位置會出現一個短很多的脈衝。雖然很短，但是還是可能會在某個 bar 不該亮的時候，稍稍點亮其中某一 dot 位置的 LED。觀察到這樣的波型後，才想到可能是因為 bar 掃描的 8 條信號線是從一個 3 to 8 解碼 IC 產生，然而這個解碼 IC 的 3 個 binary code 輸入線是循序變動的，並不是同時更動，例如由 b001 進入 b010 時，如果由最低位(LSB)開始更新，在過程中有瞬間是 b000，然後才進入 b010，分析如下：



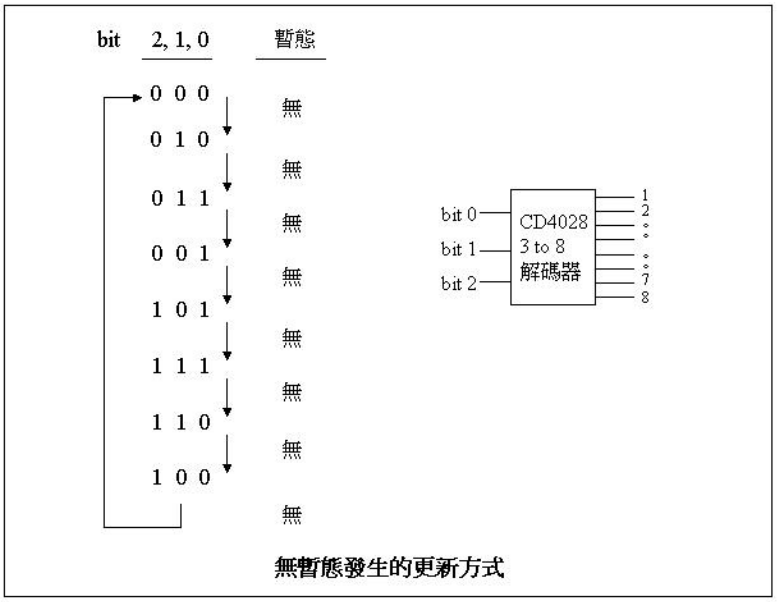
圖二十一、產生暫態的原因分析

因為這三個 bit 的變換是由程式控制，所以想過是否可以從不同的位置開始變換，使它沒有暫態存在？在紙上推演過後發現是不可能的，當時曾考慮過幾個用改變電路解決的方式：

A. 因為所用的 3 to 8 Binary decoder CD4028 實際是一個 4 to 16 的解碼器，現在只用到它的 A,B,C 輸入而未用到的最高位(MSB)輸入，正好可以用在當 3 to 8 解碼時關閉所有的輸出信號。因此可以再用一支 pin 當作 digital output，在每次變換 binary code 時，暫時把解碼輸出關閉，應該就不會誤觸其它的 dot 了。這樣做需要修改少量電路。
B. 在 3 to 8 Binary decoder 前加上一個 Binary Counter IC，從微控制機用兩個 Digital output pin 即可控制，一支 pin 送信號使

Counter 上升, 每 8 個信號後從另一支 pin 送 reset 信號使 Binary Counter 歸零，Binary Counter 的輸出再接到 3 to 8 Binary Decoder 去解為 8 個獨立信號。這個方法不敢確定當 Binary Counter 進位時會不會有同樣的狀況(也就是暫態)發生，但是 Binary Counter 是一個 IC，處理速度應該比微控制機的程式快很多，所以就算在進位時有暫態出現，時間應該也會短很多，不致於造成困擾。這個方式需要修改的電路最多，不確定性也較高。

最後，經過一番紙上推演，想出了一個最簡單的方法。如下圖，三位數的 binary code 在變換時，如果不需依大小順序進行，可以有一個一次只變動一個 bit 的方法。因為每個 bar 的點亮順序並不重要，只要單位時間內點亮的次數相同即可，所以就測試了這個方式，同時將程式改為在變換 bar 掃描時，將所有 dot 關閉，等 bar 信號穩定後再送出 dot 點亮信號。經過這樣的修改，LED 顯示部份可算完成。



圖二十二、進位暫態的解決方式

以下是修改後的程式片段：

```
/* Joyful stick
  This program

  use miawScan
  */
    :
    :
```

```
    :
int valX = 0;            // variable to store the value coming from the sensor
int valY = 0;            // variable to store the value coming from the sensor
int valZ = 0;            // variable to store the value coming from the sensor
    :
unsigned int color[4] = {
   257,257,257,257};    //array for blue, green, yellow, red
    :
int miawScan[8] = {4,0,2,3,1,5,7,6};      ←= (bar  掃描順序)
    :
    :

void setup() {
    :
    :
   motionClassfy
   cleardigitalports();
// Turn on dots 0-7 related to bar #
   for (int seq = 0; seq < 8; seq++){         // display start
   int bar = miawScan[seq];        ← (用改良後的掃描順序)
   int colorIndex = bar >> 1;
   unsigned int colorNow = color[colorIndex];
   if ( bar == ((bar >> 1) << 1)){            // if bar = 0,2,4,6
      for ( int x = 0; x < 8; x++){                  // check dots in bar 0,2,4,6
         if (colorNow & (1 << x)) {
            digitalWrite(pin2 + x , HIGH);
         }
         else
         {
            digitalWrite(pin2 + x, LOW);
         }
      }
   }
```

圖二十三、修改後的 LED 掃描程式片段，對 bar 的跳動順序作了改變，完全無暫態狀況

這個現象的基本原因是 Arduino 模組的 Digital Output Pin 不夠用，如果像一般 8051 微控制機有 16 個以上的 Digital Output Pin 可用的話，bar 掃描與 dot 顯示各使用一個 8 bit 的 Output port，就不會有鬼影的問題。但是因為 Arduino 模組使用的 ATMega168 有速度快、程式空間大、內建類比／數位轉換、體積小、易開發等優點，所以還是決定用 ATMega 來做。為了解決 Digital Output Pin 不夠的問題，想到把永遠是 8 選 1 的 bar 掃描功能設計成用 3 個 bit 的 binary code 驅動，然後在顯示板這邊用一個 3 to 8 解碼器解成 8 條信號線，就只須 8 + 3 共 11 支 Digital output pin，但是這樣做，會因為 binary code 的 3 個 bit 是由程式 1 個 bit 接 1 個 bit 送出，所以在某一個 bit 變換數值時，3 to 8 解碼器的輸入端會短暫出現不是我們期望的 bit 組合，導致那 8 支輸出腳也會短暫出現非預期的信號，而使在其它處於熄燈狀態的 bar 上相對應位置的 dot 微亮。。

5. 結論

經過研究與實際製作，彩光魔棒這個創意是可行的。它有著多彩多姿的顯示模式，而且還會隨著揮舞動作變化，是個又酷又炫的互動娛樂。在演唱會中如果看到心儀的對象，拿著彩光魔棒對著他或她揮一揮，特定的美麗 pattern 就會出現，可以向對方傳達情意。

這個作品製作最因難的是收集揮動資料及分類，因為收集到的資料變化很大，尤其是如果要大量生產時，每一個人的使用習慣(揮動特徵)差異很大。可能這個產品最需要有學習能力，讓新的使用者揮舞一陣子之後，由彩光魔棒學習起來作為分類的依據，應該是最佳的解決方法。

6. 討論及應用

這個題目中遇到的最大問題就是在 LED 顯示控制的部份，雖然花了很多時間嘗試各種解決方式，但是也學到很多經驗，而且只以一顆微控制機加上一顆 3 to 8 的解碼 IC 就完成所有功能，非常簡潔。

因為是試作，為求簡化，LED 只做了一面。如果想要更好看的話，LED 板可以擴充成三面甚至四面，這樣就每一個角度都看得到。

要把揮動資料的分析作到精準又有極大的寬容性看來是矛盾的，參考過一些書上的說明都認為要用特殊的軟體如 Fuzzy Logic，Neural Network 等來分析與配對，因為時間關係，這些研究只好留到進大學後再做。

構思系統時已經考慮到擴充性，所以把揮舞動作與顯示 pattern 的選擇完全分開，動作分析的最終輸出是一個兩位數的分類號碼，代表某型動作，這個數字交給顯示模組後，顯示模組依這個數字取用顯示 Pattern。

當初構想時，還有一些其他設想，但是為了避免「貪多嚼不爛」- 把研究範圍設得太大而失控，所以暫時排除在外。最主要的一個構想是加上無線模組，把揮舞動作傳輸到附近的個人電腦，這樣就創造出更多的應用，DJ 可以揮舞著它控制音樂的播放、演唱會主持人用它啟動舞臺特效、或當作電玩的手把。也可讓會場的彩光魔棒相互連線，其中一支作為主控，由一人控制全場的彩光魔棒同步閃爍，也是很有趣的應用。

也可以再加上喇叭，在出現閃爍 pattern 的同時，有相配的聲音，適合在棒球賽之類環境使用，高亮度 LED 的光夠亮，在室外也很明顯。

7. 參考資料

[1]**ATMega48/88/168** Automotive data sheet(revision F), http://www.atmel.com/dyn/resources/prod_documents/doc7530.pdf, updated 09/07

[2]**ADXL330 Small, Low Power, 3-Axis ±3g iMEMS Accelerometer**, **http://www.analog.com/en/prod/0%2C2877%2CADXL330%2C00.html**,

[3]**Arduino**, **http://www.arduino.cc/**

[4]林振漢, **8051 單晶片實作 - 使用 C 語言**, 博碩文化, 2005

[5]謝澄漢、徐發義、許佳興, **8051 C 語言實作寶典**, 宏友圖書, 2005

[6]Dan O'Sullivan and Tom Igoe, **Physical Computing: Sensing and Controlling the Physical World with Computers**, Course Technology PTR, May 2004