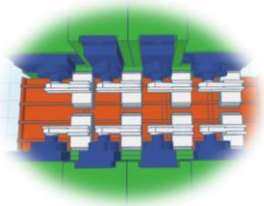
自動彈奏烏克麗麗

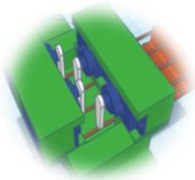
林承曄, 蔣幼齡
中國文化大學光電物理系

摘要
某次在樂器店，看到不需真人彈奏也能演奏曲目的自動鋼琴，讓我非常感興趣。因為自己會彈奏烏克麗麗，加上有使用8051微處理機與設計機構的能力，了解運作原理後，我運用擁有的知識，做出了一部自動彈奏烏克麗麗的機器。

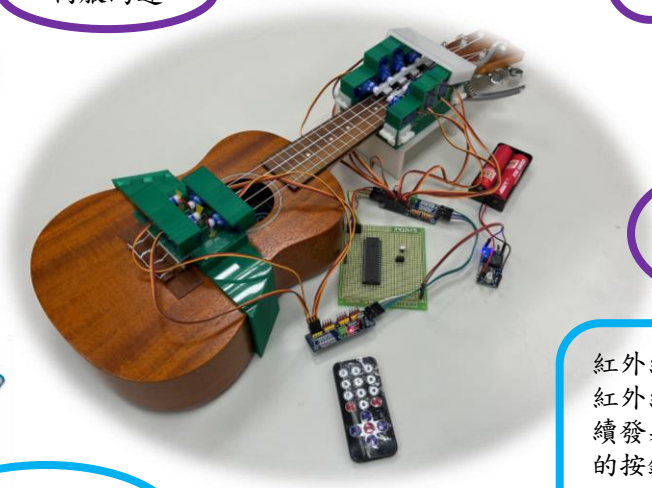
伺服馬達



壓弦馬達(伺服馬達)一邊四個，兩側共八個(0~7)，放在烏克麗麗弦的前端，可以彈奏前四格的音階，加上空弦總共有20個音(見表一)，目前可以彈奏C到Ċ#。



四個撥弦馬達連接撥片，負責撥弦。



紅外線接收器

發射 LED 940nm

紅外線發射器及接受器組成紅外線遙控模組。發射器持續發射38kHz的載波，遙控器的按鍵被按下時，送出一組8位元代碼及一組反代碼，解碼後可確認哪個按鍵被按下。

馬達驅動板 PCA9685

作品目前內建三個曲目—小星星、野玫瑰和Do-Re-Mi之歌。
本專題作品精采的部分有:

- 控制12顆馬達在同一時間正確運作。
- 壓弦機構：模擬人的手指壓弦，需要很精確的控制每個壓弦的位置。
- 馬達架設計：配合烏克麗麗尺寸與外型和馬達的工作特性，需要精確設計。

3	2	1	0	空	馬達編號 / 弦編號
B	$A^{\#}$	A	$G^{\#}$	G	一
E	$D^{\#}$	D	$C^{\#}$	C	二
$G^{\#}$	G	$F^{\#}$	F	E	三
$\dot{C}^{\#}$	$\dot{C}$	B	$A^{\#}$	A	四
7	6	5	4	空	弦編號 / 馬達編號

表一、壓弦動作表

參考資料
1. 伺服馬達介紹 | LinkIt smart 7688 guide (gitbooks.io)
https://yuanyouyuan.gitbooks.io/linkit-smart-7688-guide/content/chap3/servo.html
2. PCA9685簡介-莓亞科技-官網PCA9685簡介(meiyagroup.com.tw)
https://www.meiyagroup.com.tw/pca9685%e7%b0%a1%e4%bb%8b/

111 學年度第 2 學期

專題討論(一)

自動彈奏烏克麗麗

系級：光電物理三

學號：A9214553

姓名：林承曄

指導老師：蔣幼齡 教授

中華民國 112 年 六 月

光電物理學系 三 年級學生 林承曄 學號 A9214553

修習 專題討論（一） 所完成之論文

自動彈奏烏克麗麗

內容與格式經論文指導老師審查，認為符合繳交標準。

指導老師：

蔣 幼 齡

中華民國 112 年 六 月

目錄

一、 中文摘要

某次在樂器店，看到不需真人彈奏也能演奏曲目的自動鋼琴，讓我非常感興趣。因為自己會彈奏烏克麗麗，加上有使用 8051 微處理機與設計機構的能力，了解運作原理後，我運用擁有的知識，做出了一部自動彈奏烏克麗麗的機器。

自動彈奏烏克麗麗除了觀賞性十足外，還有助於表演和教學。自動彈奏烏克麗麗可以提供即時的伴奏功能，允許使用者在演奏過程中享受和弦和節奏的支援。這使得演奏更具豐富和有趣，同時為用戶提供更大的創作空間；自動彈奏烏克麗麗不需要使用者具備音樂技巧或彈奏經驗。任何人都可以輕鬆享受彈奏烏克麗麗的樂趣，而不必花費大量時間學習和練習。

二、 英文摘要

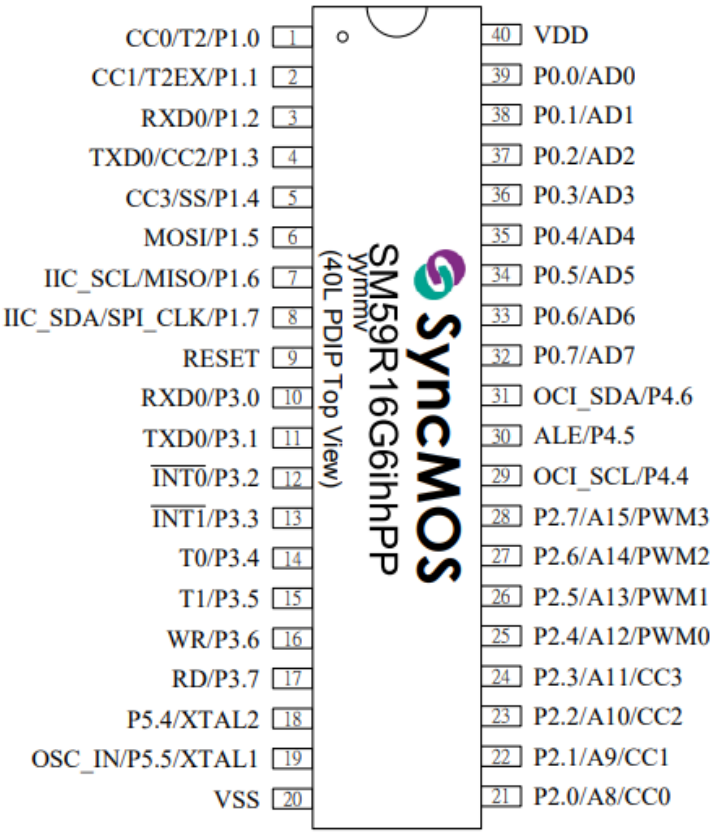
One day, at a musical instrument store, I came across a player piano that can play music without a human performer. It piqued my interest greatly. I can play the ukulele. Plus, I am able to make use of 8051 microcontrollers and design mechanisms. After realizing operational principles, I utilize my existing knowledge and successfully create the machine that can automatically play the ukulele.

In addition to ornamental, an automatic ukulele player benefits from both performance and teaching purposes. It provides instant accompaniment function and allows users to enjoy chord and rhythm support while playing. This enhances the richness and fun of the performance while providing users with greater creative freedom. Besides, the automatic ukulele player eliminates the need for users to possess musical skills or playing experiences. Anyone can effortlessly enjoy the pleasure of playing the ukulele without spending a large amount of time on learning and practicing.

三、 8051 單晶片介紹

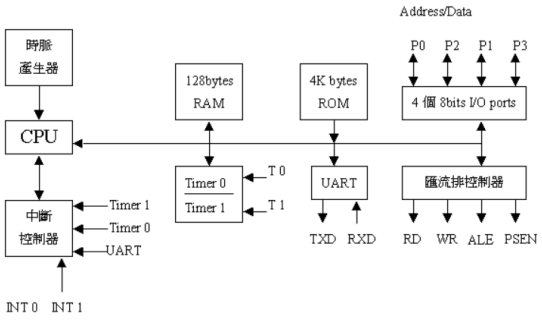
本專題使用的 8051 微控制器為新貌國際科技公司的 1T 單晶片 SM59R16G6，具有以下特性：

- 8 位元處理器架構：具有 8 位元處理器核心。
- 儲存結構：具有 1KB 的 RAM 和 64KB 的 ROM。
- 外部存儲擴展：可以通過外部存儲器芯片擴展其存儲容量，例如使用外部 RAM 或 EEPROM。
- I/O 引腳：具有四組 8 位元可配置的 I/O 引腳，可以用於連接各種外部設備和感測器。
- 定時器/計數器：具有三個定時器/計數器，可用於計時、生成精確的時間延遲和執行週期性任務。
- 串行通信：支援多種串行通信接口，例如 UART（通用異步接收發送器）和 SPI（串行外設接口）。
- 中斷系統：具有十個中斷向量，可以檢測和處理外部事件和中斷請求。



圖一、SM59R16G6 腳位圖

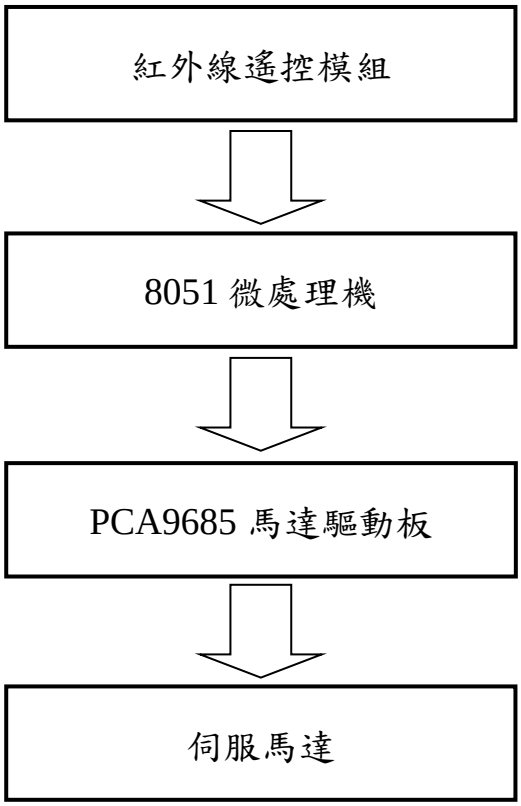
圖二、8051 單晶片



圖三、8051 內部結構

四、工作流程

自動彈奏烏克麗麗的工作流程如圖四，利用遙控器選擇內建曲目傳訊給 8051，8051 依所選曲目的曲譜彈奏烏克麗麗。彈奏的方式是 8051 控制 8 個伺服馬達壓弦、4 個伺服馬達撥弦來完成。8051 無法提供足夠的電流控制伺服馬達，我利用了 PCA9685 馬達控制板，一片 PCA9685 可以控制 16 個馬達，為了更好區分壓弦與撥弦的工作,也為了未來更容易增加壓弦的馬達,所以分別使用兩片 PCA9685。



圖四、工作流程

五、元件介紹

(1)伺服馬達(TowerPro SG90)

作品使用的伺服馬達 SG90(圖五)有電源線、地線及訊號三條線，可通過對訊號線傳遞不同時寬的脈衝訊號來控制馬達運轉的位置，控制時通常會送出數個相同時寬的脈衝波以確保馬達能到達準確的位置。是故單晶片會利用對伺服馬達傳送 PWM (Pluse Width Modulation) 訊號來控制馬達轉動的角度。

高脈衝時間(ms)	轉動角度(°)
1.00	0
1.25	45
1.50	90
1.75	135
2.00	180

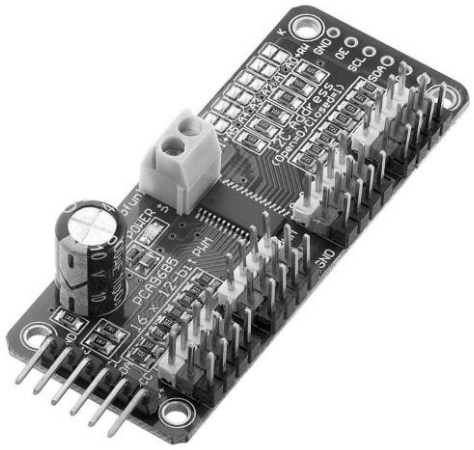
表一、高脈衝時間與角度對照表

圖五、伺服馬達

表一是傳送的高電平脈衝時間與伺服馬達角度對照，使用時須先送 1ms 的脈衝波確認 0°的位置。通常會以 50Hz 或 60Hz 的 PWM 脈波改變佔空比傳送訊號。以傳送 50Hz(週期 20ms)PWM 波為例，高電平脈衝時間 1.5 毫秒(90°)，就是將 duty cycle 調至 1.5/20 = 1.5×0.05 = 7.5%，馬達就會轉動並固定在 90°的位置；若使用 60Hz 的 PWM 波，同樣要轉到 90°，duty cycle 應調至 1.5*0.06 = 9%。

(2)PCA9685 馬達驅動板

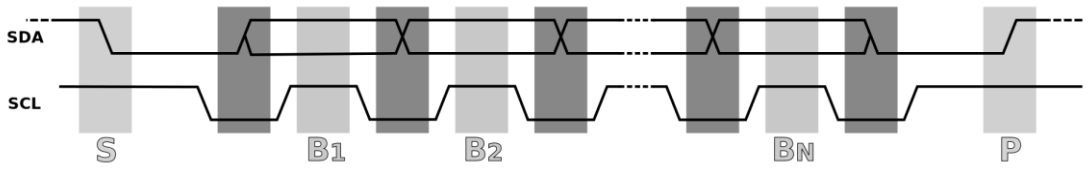
PCA9685(圖六)是採用 I^2C 通訊介面的 16 路 PWM 產生器，解析度為 12bits(4096)，開關頻率為 24 Hz 到 1526 Hz，16 路的頻率無法單獨設置，但可以調整各路的 duty cycle，很適合控制多軸小型伺服馬達。PCA9685 包含一個帶有內置時鐘的 I^2C 控制 PWM 驅動器，每次設置皆會保持最後一次所設置的參數運作，無需連續發送訊號控制。使用 PCA9685 同時可以解決 8051 的 PWM 震盪頻率太高的困擾。



圖六、PCA9685 馬達驅動板

I^2C Bus(Inter-Integrated Circuit Bus) 積體匯流排電路在 1982 年由荷蘭飛利浦半導體公司(Philips Semiconductor)所開發，是一種串列通訊匯流排。主要是為了讓微控制器或 CPU 以較少的接腳數連接眾多的低速週邊裝置之用。圖七是 I^2C 通訊協定的電路訊號規則，只使用兩條雙向汲極開路(Open Drain)線，其中一條線為傳輸資料的串列

資料線(SDA)，另一條線是啟動或停止傳輸以及傳送時鐘序列的串列時脈(SCL)線，允許相當大的工作電壓範圍，但典型的電壓準位為+3.3V 或+5v。



圖七、I^2C 通訊協定

通訊協定重點整理如下：

1. SCL 為 Low 時(圖七深色)，SDA 可以改變資料，以準備傳下一筆資料。
2. SCL 為 High 時(圖七 B_1 到 B_N)，SDA 必需保持訊號穩定，不可以改變，以方便對方讀取(栓鎖)資料。
3. SCL 為 High 時，如果 SDA 有變動則視為特殊狀況：Start(啟始，SDA 由 High 轉為 Low)或 Stop(結束,SDA 由 Low 轉為 High)。如圖七 S 跟 P 的位置。

(3)紅外線遙控模組(HX1838)

紅外線模組(如圖八)是由紅外線發射器及接受器組成，以紅外線發光 LED 發射波長 940nm 的紅外線、38kHz 的載波頻率來傳送信號，在每個按鍵被按下發送編號碼時，會送出一組 8 位元代碼及一組 8 位元的反代碼，當兩組代碼相互以 16 進制為互補時，才為有效代碼。

1	0XBA45	0	0XE619
2	0XB946	*	0XC916
3	0XB847	#	0XE20D
4	0XBB44	↑	0XE718
5	0XBF40	↓	0XAD52
6	0XBC43	→	0XA55A
7	0XF807	←	0XC708
8	0XCA15	OK	0XE31C
9	0XE609		

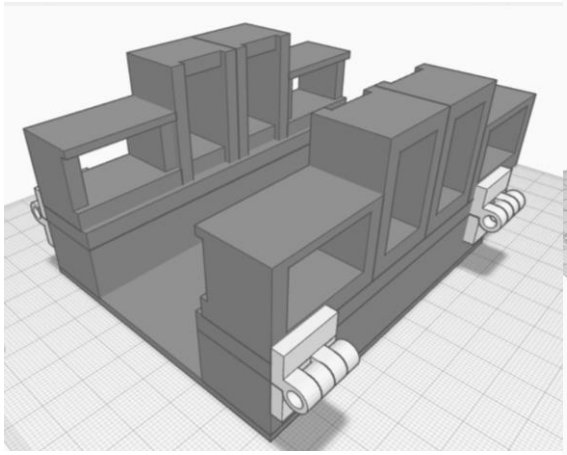
表二、各按鈕編碼對照表

圖八、紅外線遙控模組

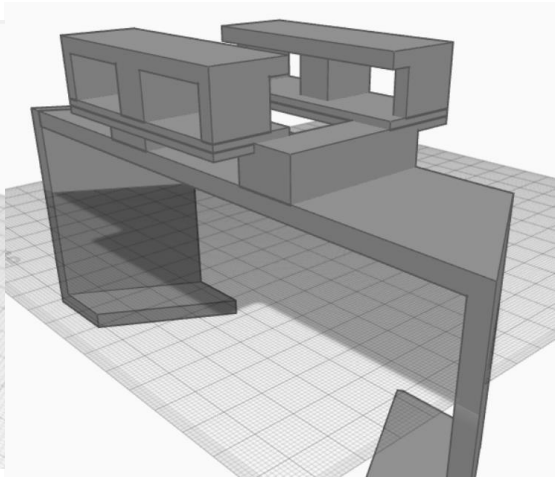
六、機構製作

本專題的機構（撥弦和壓弦）都是由 Tinkcad 網站繪製，再由 3D 列印機（型號：XYZprinting da Vinci 1.0 Pro 3-in-1）列印，並加上伺服馬達、吉他撥片和止滑墊等零件做組裝。

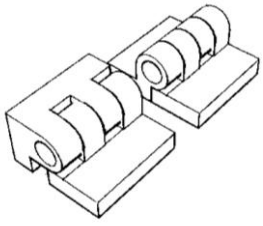
我使用了 PLA 材質列印馬達架（圖九為上方壓弦，圖十為下方撥弦）、門軸（圖十一）和壓弦支架（圖十二），用 SpiderFlex TPE 材質列印軟墊（圖十三），Flex 在常溫下具有高伸長率、高回彈率、低壓縮永久變形率、及低脆化溫度等特性，擁有接近人類手指的彈性和摩擦力。



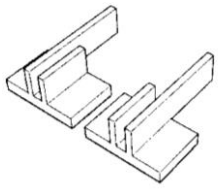
圖九、壓弦馬達架



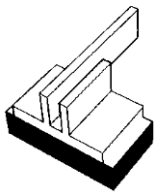
圖十、撥弦馬達架



圖十一、門軸



圖十二、壓弦支架



圖十三、軟墊

七、壓弦動作

壓弦的動作如表三所示，其中 **0**～**7** 是馬達編號，**空**代表空弦馬達不需下壓，一~四是弦的編號，*A*～*G* 是彈奏的音，一個馬達同時控制兩條弦。如果要彈奏 *G* 的音，可以有兩種選擇：

1. 馬達不按壓，第一條弦彈奏
2. **6** 號馬達按壓，第三條弦彈奏

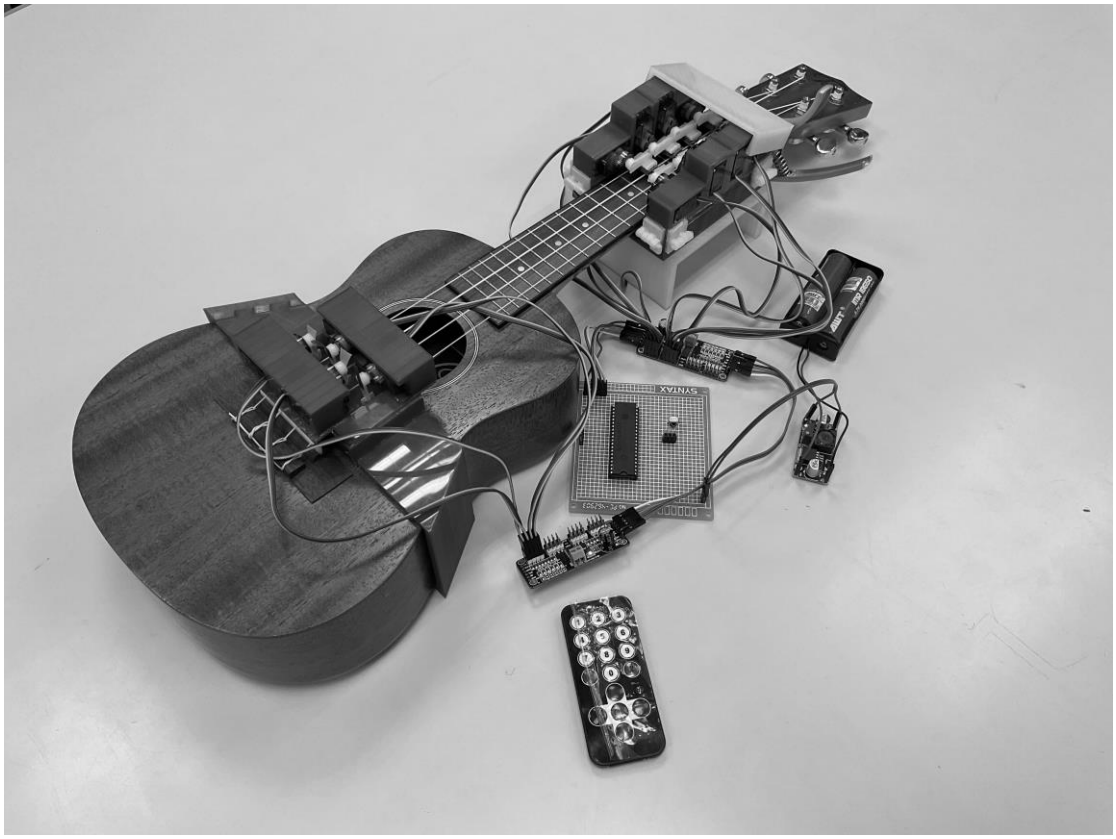
3	**2**	**1**	**0**	**空**	
B	$A^{\#}$	*A*	$G^{\#}$	*G*	一
E	$D^{\#}$	*D*	$C^{\#}$	*C*	二
$G^{\#}$	*G*	$F^{\#}$	*F*	*E*	三
$\dot{C}^{\#}$	$\dot{C}$	*B*	$A^{\#}$	*A*	四
7	**6**	**5**	**4**	**空**	

表三、壓弦動作表

八、成品

壓弦馬達一邊四個，兩側共八個，放在烏克麗麗弦的前端，可以彈奏前四格的音階，加上空弦總共有 20 個音，目前可以彈奏 Do 到高音 Do#，作品內建三個曲目，我就挑在這些音韻間的小星星、野玫瑰和 Do-Re-Mi 之歌，如果想要有更寬的音韻必須在琴上增加壓弦馬達。

本專題最難的除了控制 12 顆馬達在同一時間正確的運作，還有機構的設計，為了模擬人的手指壓弦，需要很精確的控制每個壓弦的位置，3D 列印也是重新設計過很多版本才有現在的成品，最後的成品如圖十四。



圖十四、成品

九、參考資料

1. PCA9685|恩智浦半導體，網址：

   https://www.nxp.com/products/power-management/lighting-driver-and-controller-ics/led-controllers/16-channel-12-bit-pwm-fm-plus-ic-bus-led-controller:PCA9685

2. I2C bus 簡介 (Inter-Integrated Circuit Bus) @ 傑克! 真是太神奇了! :: 痞客邦 :: (pixnet.net)，網址：

   https://magicjackting.pixnet.net/blog/post/173061691

3. 第一章單晶片微電腦簡介 (archive.org)，網址：

   https://web.archive.org/web/20080309140012/http://elearning.stut.edu.tw/mechelec/ch1.htm

4. 伺服馬達介紹 | LinkIt smart 7688 guide (gitbooks.io)，網址：

   https://yuanyouyuan.gitbooks.io/linkit-smart-7688-guide/content/chap3/servo.html

5. PCA9685 簡介-莓亞科技-官網 PCA9685 簡介 (meiyagroup.com.tw)，網址：

   https://www.meiyagroup.com.tw/pca9685%e7%b0%a1%e4%bb%8b/

十、工作日誌

時 間	地 點	內　　容
112/02/12	大義 221	題目確定
112/02/25	大義 221	完成流程圖，設計機體架構
112/03/11	大義 221	3D 列印原型
112/03/18	大義 221	學習 I^2C 和 PCA9685 的使用方式
112/03/25	大義 221	重新列印機構
112/04/08	大義 221	完成壓弦與撥弦程式
112/04/22	大義 221	編寫紅外線遙控模組
112/04/29	大義 221	完成音階動作流程
112/05/06	大義 221	撰寫曲目
112/05/13	大義 221	曲目調整

附錄一 :程式碼

主程式

```
#include "SM59R16G6.h"
#include "pca9685_for_i2c.h"
#include "IR_remote.h"
#include "performing.h"

#define board1    0x40
#define board2    0x41
#define pin0      0x06
#define pin1      0x0a
#define pin2      0x0e
#define pin3      0x12
#define pin4      0x16
#define pin5      0x1a
#define pin6      0x1e
#define pin7      0x22
#define pin12     0x36
#define pin13     0x3A
#define pin14     0x3E
#define pin15     0x42
unsigned char pitch,length=0;
main()
{
      unsigned char first=0,second=0,third=0,fourth=0;
      unsigned char i=0;
      unsigned char state=0;
      const unsigned char      song1[]={
            'C',00,'E',00,'G',00,'A',80,
            'C',20,'C',20,'G',20,'G',20,'A',20,'A',20,'G',40,
            'F',20,'F',20,'E',20,'E',20,'D',20,'D',20,'C',40,
            'G',20,'G',20,'F',20,'F',20,'E',20,'E',20,'D',40,
            'G',20,'G',20,'F',20,'F',20,'E',20,'E',20,'D',40,
            'C',20,'C',20,'G',20,'G',20,'A',20,'A',20,'G',40,
            'F',20,'F',20,'E',20,'E',20,'D',20,'D',20,'C',40,
            0,0};
```

```
const unsigned char      song2[]={
        'C',00,'E',00,'G',00,'A',80,
        'E',20,'E',20,'E',20,'G',20,'G',10,'F',10,'F',10,'E',10,'D',40,
        'D',20,'D',20,'E',20,'F',20,'G',40,'Q',40,
        'E',20,'E',20,'E',20,'G',20,'G',10,'O',10,'F',10,'E',10,'D',40,
        'G',20,'G',20,'A',30,'G',10,'O',10,'G',10,'A',10,'B',10,'G',40,
        'G',10,'B',10,'A',10,'G',10,'O',10,'E',10,'N',10,'E',10,'Q',30,'O',10,'G',40,
        'D',20,'D',20,'E',20,'F',20,'G',20,'A',10,'B',10,'Q',40,
        //'A',20,'Q',20,'F',20,'A',20,'C',20,'D',10,'E',10,'C',40,
        'A',20,'Q',20,'F',20,'A',20,'C',20,'E',10,'D',10,'C',40,
        0,0};
const unsigned char      song3[]={
        'C',00,'E',00,'G',00,'A',80,
        'C',30,'D',10,'E',30,'C',10,'E',20,'C',20,'E',40,
        'D',30,'E',10,'F',10,'F',10,'E',10,'D',10,'F',80,
        'E',30,'F',10,'G',30,'E',10,'G',20,'E',20,'G',40,
        'F',30,'G',10,'A',10,'A',10,'G',10,'F',10,'A',80,
        'G',30,'C',10,'D',10,'E',10,'F',10,'G',10,'A',80,
        'A',30,'D',10,'E',10,'O',10,'G',10,'A',10,'B',80,
        'B',30,'E',10,'O',10,'P',10,'A',10,'B',10,'Q',80,
        'Q',10,'B',10,'A',20,'F',20,'B',20,'G',20,'Q',80,
        0,0};
const unsigned char      song4[]={
        'C',00,'E',00,'G',00,'A',80,
        'C',40,'M',40,'D',40,'N',40,'E',40,'F',40,'O',40,'G',40,'P',40,'A',40,'L',40,'B',40,'Q',40,'R',40,
        0,0};
ini_remote();
TMOD=0x19;
ET1=1;
TH1=(65536-18510)/256;
TL1=(65536-18510)%256;
//TR1=1;
EA=1;

begin();
PWMfreaq_set();
reset();
```

```
while(1)
{
	switch(state)
	{
		case 0:

			while(key_edge==0);
			state=1;

			//key_edge = 0x46;
			//state = 1;

			break;

		case 1:
			switch(key_edge)
			{
				case 0x45://1
					pitch=song1[i++];
					length=song1[i++];
					break;

				case 0x46://2
					pitch=song2[i++];
					length=song2[i++];
					break;

				case 0x47://3
					pitch=song3[i++];
					length=song3[i++];
					break;

				case 0x44://4
					pitch=song4[i++];
					length=song4[i++];
					break;

				default:
```

```
                ;
}

if (pitch==0 && length==0)
{
        state=0;
        key_edge=0;
        i=0;
}
else
{
        switch(pitch)
        {
                case 'M'://CD
                        setPWM(board1,pin0,0x00,0x00,0x3b,0x01); //30
                        break;

                case 'D':
                        setPWM(board1,pin1,0x00,0x00,0x2c,0x01); //26
                        break;

                case 'N'://DE
                        setPWM(board1,pin2,0x00,0x00,0x16,0x01); //20
                        break;

                case 'F'://Fa
                        setPWM(board1,pin4,0x00,0x00,0x3d,0x02); //100
                        break;

                case 'O'://FG
                        setPWM(board1,pin5,0x00,0x00,0xce,0x01); //70
                        break;

                case 'P'://GA
                        setPWM(board1,pin7,0x00,0x00,0xce,0x01); //70
                        break;

                case 'L'://AB
```

```
                setPWM(board1,pin4,0x00,0x00,0x23,0x02); //93
                break;

        case 'B'://Si
                setPWM(board1,pin5,0x00,0x00,0xdd,0x01); //74
                break;

        case 'Q'://Do.
                setPWM(board1,pin6,0x00,0x00,0x18,0x02); //90
                break;

        case 'R'://CCD
                setPWM(board1,pin7,0x00,0x00,0xbc,0x01); //65
                break;

        default:
                break;
}

switch(pitch)
{
        case 'G':
                first=string(first,pin15);
                break;

        case 'C': case 'M': case 'D': case 'N':
                second=string(second,pin13);
                break;

        case 'E': case 'F': case 'O': case 'P':
                third=string(third,pin14);
                break;

        case 'A': case 'L': case 'B': case 'Q': case 'R':
                fourth=string(fourth,pin12);
                break;

        default:
```

```
                    break;
            }
            TR1=1;
            while (length);
            TR1=0;
            state=2;
        }
        break;

case 2:
        switch(pitch)
        {
            case 'M':
                setPWM(board1,pin0,0x00,0x00,0xa9,0x01); //60
                break;

            case 'D':
                setPWM(board1,pin1,0x00,0x00,0x85,0x01); //50
                break;

            case 'N':
                setPWM(board1,pin2,0x00,0x00,0x72,0x01); //45
                break;

            case 'F'://Fa
                setPWM(board1,pin4,0x00,0x00,0xa9,0x01); //60
                break;

            case 'O':
                setPWM(board1,pin5,0x00,0x00,0x72,0x01); //45
                break;

            case 'P':
                setPWM(board1,pin7,0x00,0x00,0x72,0x01); //45
                break;

            case 'L':
                setPWM(board1,pin4,0x00,0x00,0xa9,0x01); //60
```

```
                    break;

                case 'B'://Si
                    setPWM(board1,pin5,0x00,0x00,0x72,0x01); //45
                    break;

                case 'Q'://Do.
                    setPWM(board1,pin6,0x00,0x00,0x72,0x01); //45
                    break;

                case 'R':
                    setPWM(board1,pin7,0x00,0x00,0x72,0x01); //45
                    break;

                default:
                    break;
            }
            state=1;
            break;
        }
    }
}
        //pitch=song1[i++];
        //length=song1[i++];
        //while(key_edge==0);

void T1_int(void) interrupt 3
{
    TH0=(65536-18510)/256;
    TL0=(65536-18510)%256;
    if(length)length--;
    //P0_0=~P0_0;
}
```

副程式：pca9685_for_i2c.c

```
#include "SM59R16G6.h"

sbit scl = P0^6;
sbit sda = P0^7;

void smelldelay(unsigned int delay_cnt) //for 22.1MHz OPT:4 ~(2.33+delay_cnt*1.5)uS
{
	while (delay_cnt>0)delay_cnt--;
}

void start()//start
{
	scl = 1;
	smelldelay(5);	//bus free time between stop and start >4.7us
	sda = 1;
	smelldelay(5);	//setup times (r)START condition >4.7us
	sda = 0;
	smelldelay(4);	//hold time (r)START condition >4us
}

void send_8bits(unsigned char byte_data)//write
{
	unsigned char bit_ptr=0x80;
	while(bit_ptr)
	{
		scl = 0;
		sda =(byte_data & bit_ptr);
		bit_ptr>>=1;
		smelldelay(5);
		scl = 1;
		while(scl==0);	//in case of slave time strench
		smelldelay(3);
	}
	scl = 0;	//finish bit data transfer
	sda = 1;	 //release sda line for ACK
}
```

```
char read_8bits(void)
{
	unsigned char bit_ptr=0x80;
	unsigned char dt=0;
	P1_5 =0;
	P1_5 =1;
	sda = 1;
	while(bit_ptr)
	{
		scl = 1;
		if(sda)
		{
			dt|=bit_ptr;
		}
		scl = 0;
		bit_ptr>>=1;
		scl =0;
	}
	P1_5 =0;
	return dt;
}

char pulling_ack(void)//ACK
{
	sda = 1;			//assure sda line is released
	smelldelay(4);			//3.4us for valid data
	if(sda == 0)
	{
		scl = 1;
		while(scl==0);	//in case clock stretching
		smelldelay(4);
		scl = 0;
		return 0;
	}
	else
	{
		scl = 1;
```

```
            while(scl==0);    //in case clock stretching
            smelldelay(4);
            scl = 0;
            return 1;
        }
}
void return_nack(void)
{
        sda = 1;
        scl = 1;
        scl = 0;
}

void stop(void)//stop
{
        sda = 0;
        scl = 1;
        smelldelay(5);
        sda = 1;
}

void pca9685_write_command(char add, char rag ,char dat)
{
        add=add*2;
        start();
        send_8bits(add);
        pulling_ack();
        send_8bits(rag);
        pulling_ack();
        send_8bits(dat);
        pulling_ack();
        stop();
}

void setPWM(char add, char rag ,char LEDn_ON_L,char LEDn_ON_H,char LEDn_OFF_L,char
LEDn_OFF_H)
{
        add=add*2;
```

```
	start();
	send_8bits(add);
	pulling_ack();
	send_8bits(rag);
	pulling_ack();
	send_8bits(LEDn_ON_L);
	pulling_ack();
	send_8bits(LEDn_ON_H);
	pulling_ack();
	send_8bits(LEDn_OFF_L);
	pulling_ack();
	send_8bits(LEDn_OFF_H);
	pulling_ack();
	stop();
}

char pca9685_read_command( char add )
{
	char dt;
	add=add*2+1;
	start();
	send_8bits(add);
	pulling_ack();
	dt=read_8bits();
	return_nack();
	stop();
	return dt;
}

void blank(char add , char rag )
{
	add=add*2;
	start();
	send_8bits(add);
	pulling_ack();
	send_8bits(rag);
	pulling_ack();
	stop();
```

```
}

void begin (void)
{
	pca9685_write_command(0x40,0x00,0x80);
	pca9685_write_command(0x41,0x00,0x80);
	blank(0x40,0x00);
	blank(0x41,0x00);
	pca9685_read_command(0x40);
	pca9685_read_command(0x41);
	pca9685_write_command(0x40,0x00,0x10);
	pca9685_write_command(0x41,0x00,0x10);
	pca9685_write_command(0x40,0xFE,0x05);
	pca9685_write_command(0x41,0xFE,0x05);
	pca9685_write_command(0x40,0x00,0x00);
	pca9685_write_command(0x41,0x00,0x00);
	pca9685_write_command(0X40,0X00,0XA0);
	pca9685_write_command(0X41,0X00,0XA0);
}

void PWMfreaq_set(void)
{
	blank(0x40,0x00);
	blank(0x41,0x00);
	pca9685_read_command(0x40);
	pca9685_read_command(0x41);
	pca9685_write_command(0x40,0x00,0x30);
	pca9685_write_command(0x41,0x00,0x30);
	pca9685_write_command(0x40,0xFE,0x4B);
	pca9685_write_command(0x41,0xFE,0x4B);
	pca9685_write_command(0x40,0x00,0x20);
	pca9685_write_command(0x41,0x00,0x20);
	pca9685_write_command(0x40,0x00,0xA0);
	pca9685_write_command(0x41,0x00,0xA0);
}
```

副程式：pca9685_for_i2c.h

```
extern void setPWM(char add, char rag ,char LEDn_ON_L,char LEDn_ON_H,
                                char LEDn_OFF_L,char LEDn_OFF_H);
extern void begin(void);
extern void PWMfreaq_set(void);
```

副程式：IR_remote.c

```
//#############################################################################
//#        IR_remote decoder                                                  #
//#        using INT0 and TIMER0 to capture the high time for decoding        #
//#        sensor output connect to INT0                                      #
//#############################################################################

//pin P3_2

#include "SM59R16G6.h"
//#include "Debug_Pin.h"
//define state mode

#define        IDLE            0
#define        BYTE0           1
#define        BYTE1           2
#define        BYTE2           3
#define        BYTE3           4
#define        KEY_HOLD        5

#define        NEW_FRAME_LOW           (4500*0.9)/4.34
#define        NEW_FRAME_HIGH          (4500*1.1)/4.34
#define        SAME_KEY_LOW            (2200*0.9)/4.34
#define        SAME_KEY_HIGH           (2200*1.1)/4.34
#define        ONE_LOW                 (1690*0.9)/4.34
#define        ONE_HIGH                (1690*1.1)/4.34
#define        ZERO_LOW                (560*0.82)/4.34
#define        ZERO_HIGH               (560*1.1)/4.34
#define        HEADER_LOW              (4500*0.9)/4.34
#define        HEADER_HIGH             (4500*1.1)/4.34
#define        KEYHOLD_PERIOD          (98190*1.2)/4.34
#define        KEYHOLD_FRAME_LOW       (2250*0.9)/4.34
#define        KEYHOLD_FRAME_HIGH      (2250*1.1)/4.34

#define        CLEAR_TIMER0            TL0=0;TH0=0
#define        SET_KEYHOLD_TIMER       TH0=(65535-KEYHOLD_PERIOD)/256;\
```

```
TL0=256-(KEYHOLD_PERIOD-((KEYHOLD_PERIOD/256)*265));

unsigned char           data0;
unsigned char           data1;
unsigned char           data2;
unsigned char           data3;
unsigned char           key_edge=0;
unsigned char           key_level=0;
char        state=0;
char        cnt=0;

    void ini_remote(void)
    {   TMOD&=0xF0;         //set timer0 16 bits timer
        TMOD|=0x09;
        PFCON&=0xFC;        //set timer0 clk using OSC/96
        PFCON|=0x02;        //OSC 22.1184M (4.34us/clk;284ms period)
        CLEAR_TIMER0;
        ET0=1;              //enable timer0 inturrupt

        IT0=1;              //set EXT0 falling edge trigger
        EX0=1;              //enable EXT0 interrupt
        EA=1;               //enable all interrupt
    }


    void ir_decode(void) interrupt d_INT0_Vector using 3
    {   int timer;
        timer=(int)TH0*256+TL0;         //collect timer data for process
        CLEAR_TIMER0;                   //clear timer for next capture
        TR0=1;                          //arm timer0 capture action

        switch(state){                  //wait for legit header
            case IDLE:
                if(timer>NEW_FRAME_LOW && timer<NEW_FRAME_HIGH)
                {   data0=0;            //prepare to receive bit string data
                    cnt=8;
                    state=BYTE0;
                }
```

```
        break;

case BYTE0:
        data0>>=1;
        if(timer>ONE_LOW && timer<ONE_HIGH)data0|=0x80;
        else if(timer>ZERO_LOW && timer<ZERO_HIGH);
        else
        {       state=IDLE;
                //TOOGLE_TEST_PIN1;TOOGLE_TEST_PIN1;//##########
                break;
        }
        if(--cnt==0)
        {       data1=0;
                cnt=8;
                state=BYTE1;
        }
        break;

case BYTE1:
        data1>>=1;
        if(timer>ONE_LOW && timer<ONE_HIGH)data1|=0x80;
        else if(timer>ZERO_LOW && timer<ZERO_HIGH);
        else
        {       state=IDLE;
                //TOOGLE_TEST_PIN1;TOOGLE_TEST_PIN1;//##########
                break;
        }
        if(--cnt==0)
        {       data2=0;
                cnt=8;
                state=BYTE2;
        }
        break;

case BYTE2:
        data2>>=1;
        if(timer>ONE_LOW && timer<ONE_HIGH)data2|=0x80;
        else if(timer>ZERO_LOW && timer<ZERO_HIGH);
```

```
        else
        {   state=IDLE;
            //TOOGLE_TEST_PIN1;TOOGLE_TEST_PIN1;//##########
            break;
        }
        if(--cnt==0)
        {   data3=0;
            cnt=8;
            state=BYTE3;
        }
        break;

case BYTE3:
        data3>>=1;
        if(timer>ONE_LOW && timer<ONE_HIGH)data3|=0x80;
        else if(timer>ZERO_LOW && timer<ZERO_HIGH);
        else
        {   state=IDLE;
            //TOOGLE_TEST_PIN1;TOOGLE_TEST_PIN1;//##########
            break;
        }
        if(--cnt==0)
        {   if((data0==0)) //&& ((data0+data1)==0xFF) && ((data2+data3)==0xFF))
            {   key_edge=data2;
                key_level=data2;
                SET_KEYHOLD_TIMER;
                state=KEY_HOLD;
            }
            else state=IDLE;
        }
        break;

case KEY_HOLD:
        if(timer>KEYHOLD_FRAME_LOW && timer<KEYHOLD_FRAME_HIGH)
        {   SET_KEYHOLD_TIMER;
        }
        else state=IDLE;
        break;
```

```
        }
}


void timer0_overflow(void) interrupt d_T0_Vector using 2
{     //TOOGLE_TEST_PIN2;
      TR0=0;
      CLEAR_TIMER0;
      key_level=0;
}
```

副程式：IR_remote.h

```
extern void ini_remote(void);
extern unsigned char     key_edge;       /*nonzero hex data represent key edge data; clear after action */
extern unsigned char     key_level;      /*nonzero hex data represent key pressed and hold */
```

副程式：performing.c

```
#include "SM59R16G6.h"
#include "pca9685_for_i2c.h"

#define board2     0x41
#define pin12      0x36
#define pin13      0x3A
#define pin14      0x3E
#define pin15      0x42

unsigned char string(unsigned char x,unsigned char y)
{
        if(x==0)
        {
       setPWM(board2,y,0x00,0x00,0x72,0x01); //45
                return 1;
        }
        else
        {
                setPWM(board2,y,0x00,0x00,0x3b,0x01); //30
                return 0;
        }
}
```

副程式：performing.h

```
unsigned char string(unsigned char x,unsigned char y);
```