高雄縣高英高級工商職業學校
Kao Ying Industrial Commercial Vocational High School

專題製作報告

無線遙控風扇轉速控制器

學生姓名：黃　信　昌
劉　名　軒
曾　聖　豪
指導老師：葉　忠　賢　老師

中　華　民　國　102　年　05　月

誌　謝

首先，要特別感謝指導老師葉忠賢老師，在專題寫作過程中，從資料蒐集與分析、甚至書寫修辭，盡心盡力的指導，逐字逐句、嚴謹的審閱修改，付出極大的辛勞，使我們獲益良多；在忠賢老師適時的鼓勵之下，專題終能如期完成。

也感謝組員們的分工合作及資料收集，在學習期間的互相砥礪及陪伴與彼此加油打氣。最後，也要感謝在專題製作期間曾經幫助過我們的所有老師，正因為大家的同心協力，才能使我們的專題製作更加完整，也能更順利的完成，在此本組特別的感謝，僅致上最高的敬意及謝意。

在專題如期完成的這一刻，心裡真的很感謝幫助過我們的人。回想當初開始作專題的初步時，心中充滿著艱辛與喜悅，研究者把艱辛藏在心裡，將喜悅分享給關心我們的親、師、好友分享。

黃信昌、劉名軒、曾聖豪　謹上　2013/05

無線遙控風扇轉速控制器

摘要

現在科技日新月異，有許多日常用品不斷的改良，使用品使用起來更環保更方便，改良一項產品的動機大多是要改善使用不便之處，因此，人們的惰性使然的需求，儼然是造成科技發展的主要因素之一 。此專題對開觀採取不同方式的控制應用和製作，朝向安全、方便、等方向前進，或許有許多人有過我們這種點子，但大都只是想想罷了，卻沒有人去真正實現，這種看似簡單的點子，其中卻蘊含著許多令我們意想不到且有待克服的問題。

風扇在人類的日常生活中扮演著一位重要的角色，雖然現在的科技已經發明出許多可以代替電風扇的產品，像是空調，冷氣機，等等.....但是基於成本的高低及能源的消耗性，更重要的一點就是冷氣，空調也是造成全球暖化的元凶之一，所以電風扇在市面上所佔的比例還是比較高，他既不會很花成本也不會很耗能源。常有過這樣的經驗：想調整一風扇的風量，就得趨前去調整，靠近調好感覺剛好的風量，退回到位置後，又感覺吹到的風量不對了，反覆來回幾次才能調好適當的風量，如果能在位置上透過無線的控制調整，直接調整到合適的風量，感覺就方便多了。

關鍵詞：無線控制、風扇、8051。

目　錄

表目錄

圖目錄

壹、前言

一、製作動機

常有過這樣的經驗：想調整一風扇的風量，就得趨前去調整，靠近調好感覺剛好的風量，退回到位置後，又感覺吹到的風量不對了，反覆來回幾次才能調好適當的風量，如果能在位置上透過無線的控制調整，直接調整到合適的風量，感覺就方便多了。

二、製作目的

現在科技日新月異，有許多日常用品不斷的改良，使用品使用起來更環保更方便，改良一項產品的動機大多是要改善使用不便之處，因此，人們的惰性使然的需求，儼然是造成科技發展的主要因素之一 。

借由專題製作，就在校所學與自己學習所知，進而應用於專題的實際製作上，考驗本組組員的團隊合作與實際應用能力。過程中，應該也會得到之前未理解的專業知識，如無線控制與轉速控制，增加深入探討的精神，將來畢業後投身產業界做預先的暖身練習。

在增加專業知識方面，希望能增加邏輯信號的相關狀況，在以後參與產業的產品設計時，有較佳的領悟能力。

三、製作架構

(一)專題製作流程

我們小組成員確定後，即開始進行報告資料整理、選購相關專業書籍來參考及詢問相關的專業任課教師，經小組一再地問題討論及溝通後，訂下了此次專題製作的題目。題目確定後，我們小組便開始構思如何去完成風扇的溫度控制的電路，首先，畫出電路圖與焊接的 Layout 電路圖，反覆確認無誤後，便在麵包板上進行模擬，待測試完成即開始進行焊接工作；在整個專題應用過程中，如發現錯誤，即會與相關教師進行討論，想辦法如何去補救，且了解程式是否能夠運用自如。電路零組件部份，則會要多買一份當備用零件，假如一次就可成功則算多買；如不行，需要用到第二份時，就少買一些可以重複使用的零件，藉此可控管專題製作成本。

(二) 專題製作流程圖

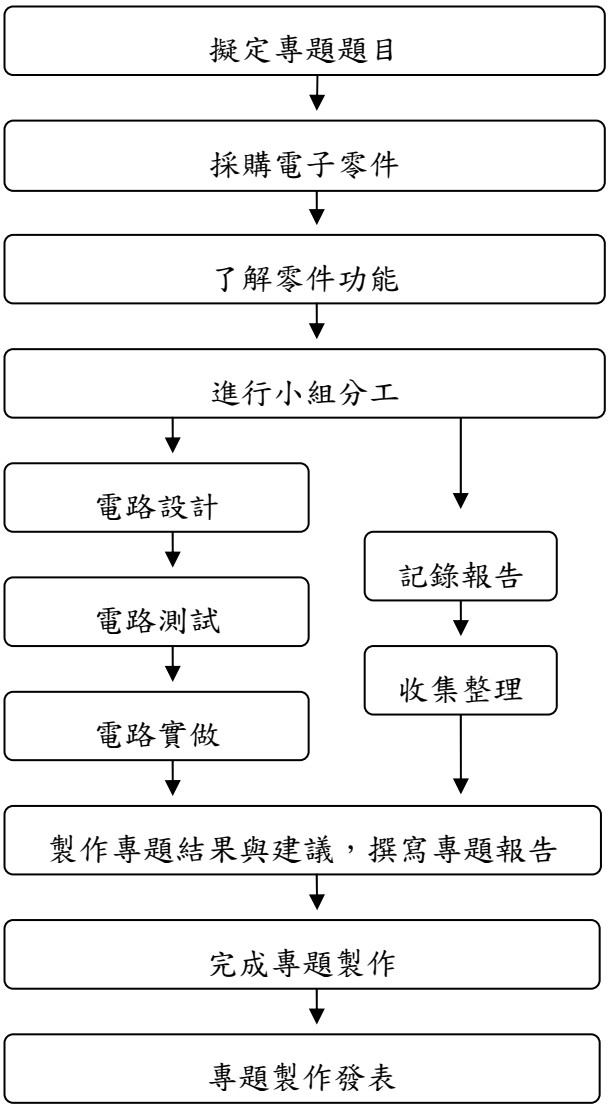

圖 1-3-1 製作流程圖

四、製作預期成效

我我們小組雖然是第一次進行合作製作專題-無線遙控風扇轉速控制，雖然擔心可能會無法成功，但有老師的協助及同學的互相協助，及辛苦製作的過程，亦希望我們的辛苦能獲得回饋及代價；為此，我們小組將專題製作的成效經討論後，定義為：

(一).透過組合語言編寫，讓風扇可以進行段數運轉

(二).經由遙控控制可讓風扇順利運轉

(三).七段顯示器可顯示風扇轉速 1~20 段

貳、理論探討

一、電子相關零組件

(一) 8051 的外部接腳及內部方塊圖

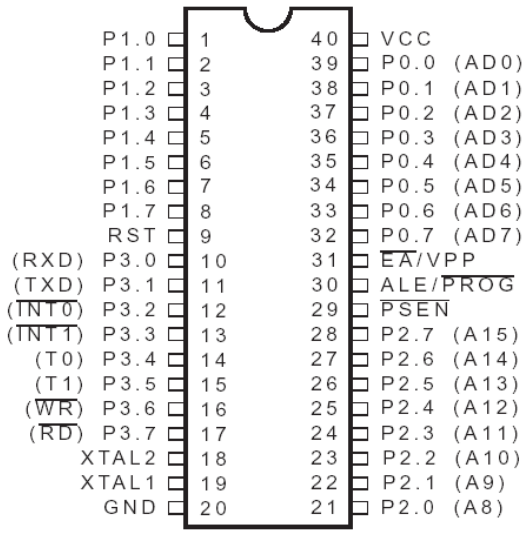

圖 2-1-1 8051 的外部接腳

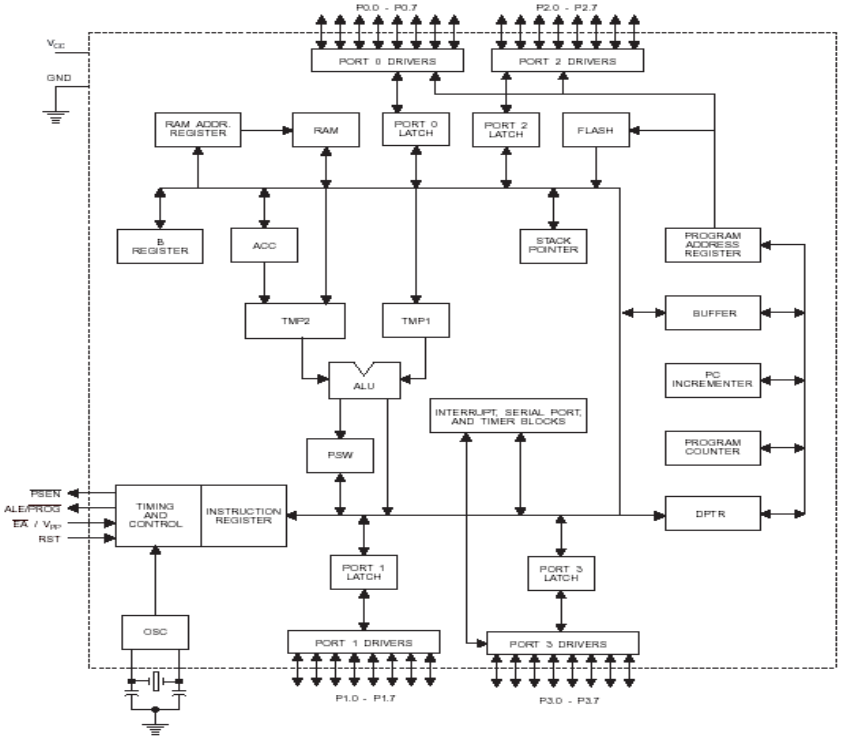

圖 2-1-2 內部方塊圖

(二)電晶體開關

「電晶體(transistor)」是一種半導體元件，也是最被使用於電子開關的電子零件，它由三個 N 與 P 型半導體材料所構成，外形上有三個接腳，分別是射極(emitter)、基極(base)、與集極(collector)，有 NPN 與 PNP 兩種基本類型，功能差別在於電流方向，下圖是小功率電晶體的外觀與表示符號。

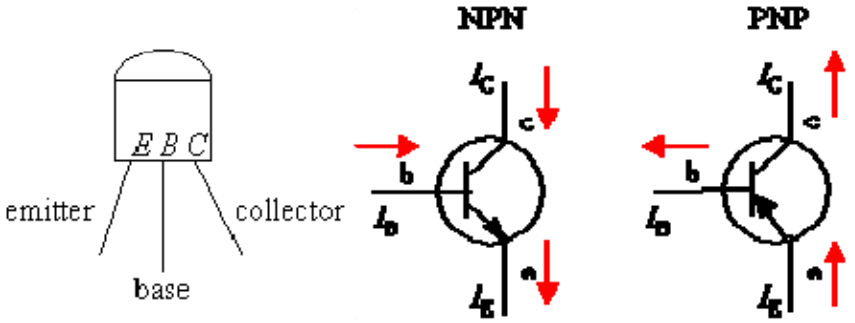

圖 2-1-3 小功率電晶體

電晶體最主要的功能是放大電流訊號，當基極到射極之間有微量電流導通時，會觸發集極到射極之間的大電流。以下分別對利用 NPN 與 PNP 電晶體常應用到的電子開關電路做說明。

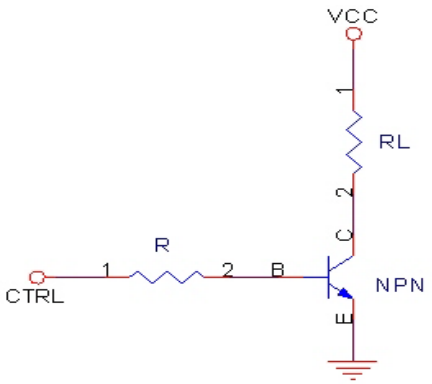

圖 2-1-4 NPN 電晶體開關

當 CTRL=LOW 時，B-E 極不導通，C-E 極亦不導通，電晶體為 OFF 狀態，RL 無電流通過。

當 CTRL=HIGH 時，B-E 極順向導通，C-E 極亦導通，電晶體為 ON 狀態，RL 有電流通過。

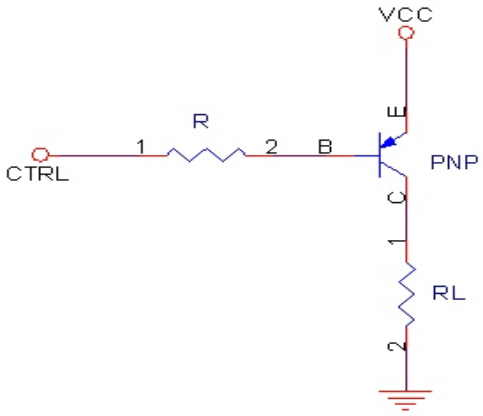

圖 2-1-5 PNP 電晶體開關

當 CTRL=HIGH 時，E-B 極不導通，E-C 極亦不導通，電晶體為 OFF 狀態，RL 無電流通過。

當 CTRL=LOW 時，E-B 極順向導通，E-C 極亦導通，電晶體為 ON 狀態，RL 有電流通過。

由上面 NPN 與 PNP 電晶體電子開關的特性比較，可看出：

CTRL = HIGHT 時，NPN 電晶體為 ON，而 PNP 電晶體為 OFF。

CTRL = LOW 時，NPN 電晶體為 OFF，而 PNP 電晶體為 ON。

所以在實際電路應用設計時，可依 CTRL 控制腳的特性，來選擇用 NPN 還是 PNP 電晶體來當電子開關。

(三)振盪器特性

XTAL1 及 XTAL2 是輸入和輸出，分別接到內部的反向放大器，來組合成為一個內建的（on-chip）振盪器。

不管是晶體或是陶瓷的振盪器都可使用，以 12MHz 振盪器為例，搭配的電容 C1、C2 的大小約為：

晶體振盪器：30pF ± 10

陶瓷振盪器：40pF ± 10

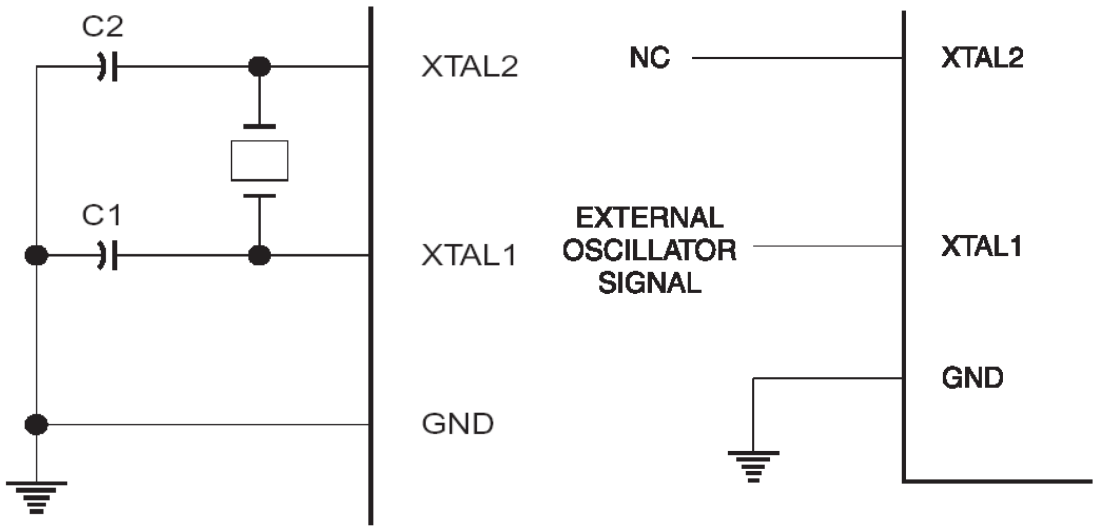

圖 2-1-6 振盪器

鐘控（clock）也可以選擇由外部振盪源由 XTAL1 輸入，此時 XTAL2 應保持不連接的狀態。

二、電源降壓電路

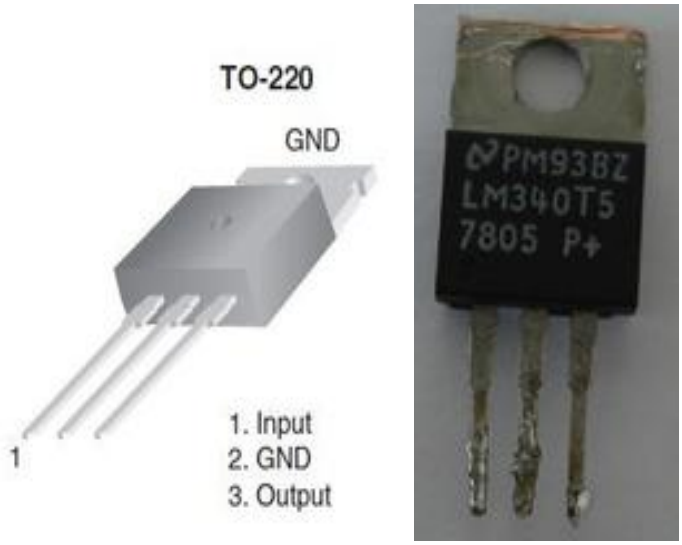

圖 2-2-1 穩壓 IC 7805

下圖是本專題的降壓電路

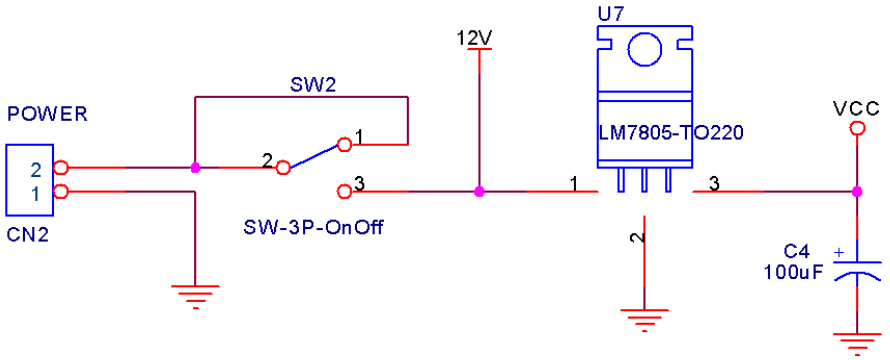

圖 2-2-2 降壓電路

12V 由 POWER 端輸入，經開關 SW 接進 7805 的輸入，輸出端接 100uF 的電容，以減少電源的雜訊。

三、2 位數 LED 七段顯示器

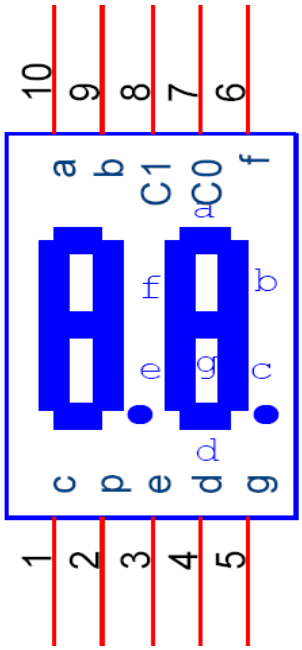

圖 2-3-1 2 位數七段顯示器

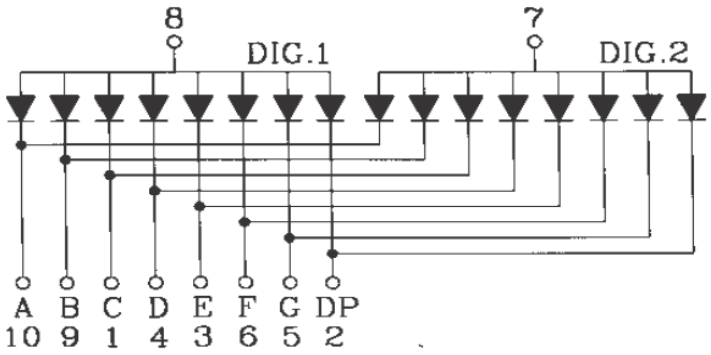

圖 2-3-2 等效電路

本專題使用 2 位數 LED 七段共陽極顯示器來顯示段數值。如等效電路所示，LED 第 7、8 腳(C0、C1)分別控制個位數及十位數的顯示，一次只能顯示一位數，此類型的 LED，都是需要利用掃瞄方式，來逐一快速切換顯示，利用視覺暫留的特性，讓人看起來好像同時在顯示一樣。

以個位數七段 LED 為例，當第 7 腳接電源，其他 LED 負極的接腳接地時，相對應的 LED 即發亮。

四、風扇控制

下圖是本專題驅動風扇的驅動電路：

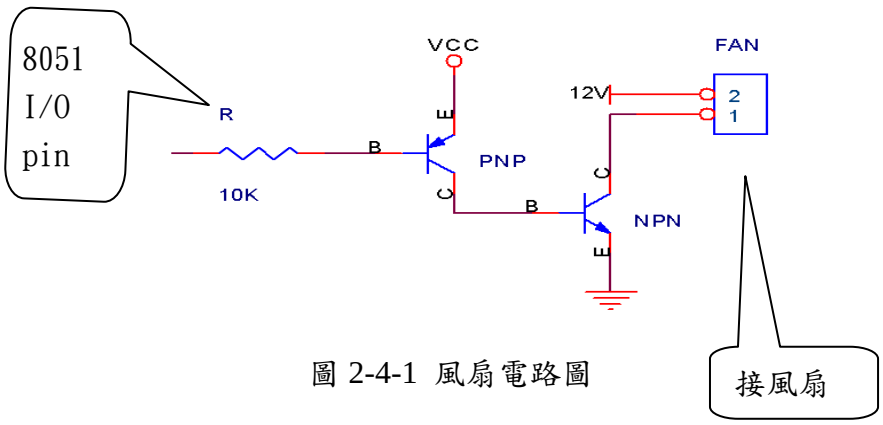

圖 2-4-1 風扇電路圖

方法是以 8051 I/O pin 低電位來驅動繼電器，因為在 8051 起動重置時，I/O pin 都是高電位，為避免在 8051 重置時的 I/O 高電位使繼電器起動，所以設計上採用低電位來驅動繼電器較合適。

當 8051 I/O pin 為高電位時，經電阻 R 到 PNP 電晶體 B 極，PNP 電晶體 E-B 不導通，PNP 電晶體的 E-C 極亦不導通，如同開路，NPN 的 B 極無電壓，NPN 電晶體不導通，繼電器線圈不起動。

當 8051 I/O pin 為低電位時，PNP 電晶體 E-B 極得到順向偏壓(大於 0.7V

的順向偏壓)，PNP 電晶體 E-C 極導通，使得 VCC 電壓進入 NPN 電晶體的 B 極，NPN 電晶體的 B-E 極得到順向偏壓，使 C-E 極導通，風扇的一端得以接地使風扇運轉。

五、紅外線傳送接收

本專題使用 PT2248、PT2249 來設計紅外線的編碼傳送與接收解碼功能，PT2248 是編碼傳送器，PT2249 是接收解碼器，兩者是搭配使用的。

PT2248 編碼傳送器是 16 pin 的 IC，接腳功能如下：

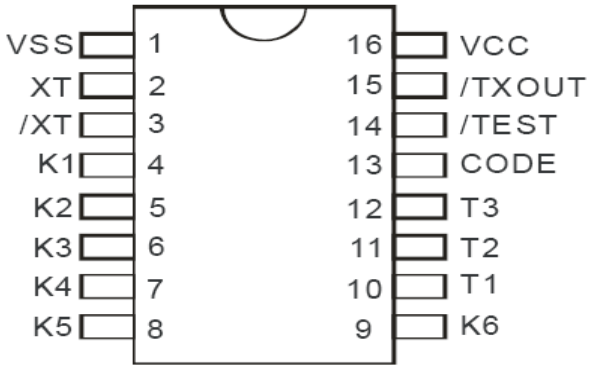

圖 2-5-1 IC PT2248 接腳

VSS：電源接地
XT、/XT：接振盪器
K1 ~ K6：按鍵輸入
T1 ~ T3：掃瞄鍵碼輸出
CODE：傳送與接收之間的匹配碼接腳
/TEST：傳送鍵碼測試腳，通常是空接
/TX OUT：傳送訊號輸出腳
VCC：電源輸入

下圖是 PT2248 的振盪器輸入電路，使用 455KHz 的振盪器可使載波傳送訊號固定在 38KHz，38KHz 是常見的紅外線載波頻率，這關係到接收端的紅外線接收元件，當然本專題採用 38KHz 的紅外線接收器。

圖 2-5-2 PT2248 振盪器輸入電路

下圖是 PT2248 的按鍵輸入電路，當按鍵按下時，PT2248 內部即會將按鍵狀態，編成固定排列的碼，然後自動由/TXOUT 腳，以前面提到的 38KHz 載波頻率送出去。

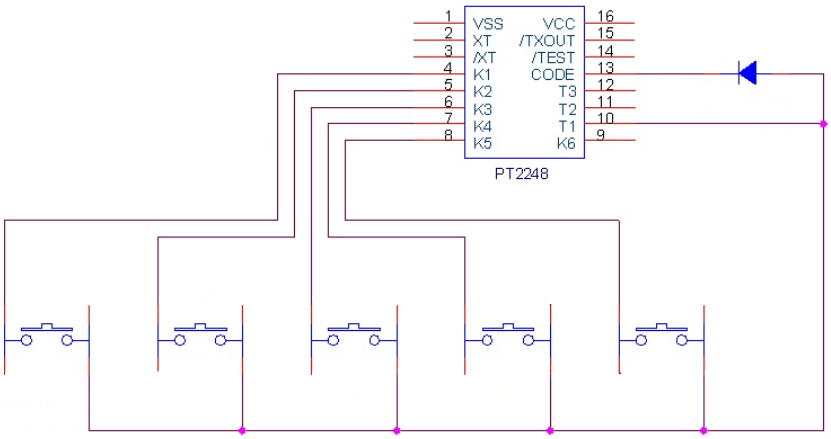

圖 2-5-3 PT2248 按鍵輸入電路

下圖是紅外線接收電路，本專題使用 38KHz 的 3pin 紅外線接收器，紅外線接收器收到 38KHz 的紅外線載波時，會從 OUT pin 輸出，經由 PNP 電晶體放大，由電晶體的 C 極送到 PT2249 的 RXIN pin，做解碼動作。

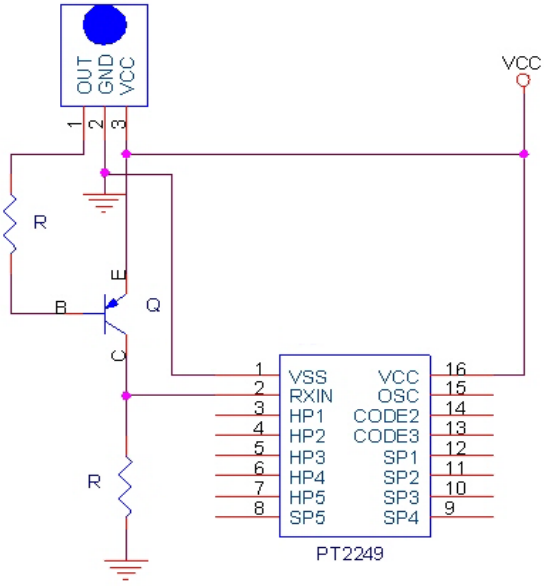

圖 2-5-4 紅外線接收電路

參、專題製作

一、設備及材料

表 3-1-1 專題製作使用儀器(軟體)設備一覽表

儀器（軟體） 設 備 名 稱	應 用 說 明
個人電腦	專題報告、電路圖製作及進行專題成品電路測
數位相機	拍攝小組合作過程、專題功能使用及紀錄整個專題製作流程
雷射印表機	列印專題資料、圖片及專題報告成果
三用電錶	測量零件有無損壞及專題電路板各信號之量
IC 萬用燒錄器	利用燒錄器將程式燒錄至 89C51 單晶片中
電源供應器	提供專題成品所需之電源
Microsoft Office Word	專題報告、製作過程的撰寫
Microsoft Office Power Point	進行口頭報告、製作及專題成品報告呈現
Keil-C	晶片組合語言程式之編輯、燒錄軟體
Protel 99SE	繪畫專題電路之線路圖

二、製作方法與步驟

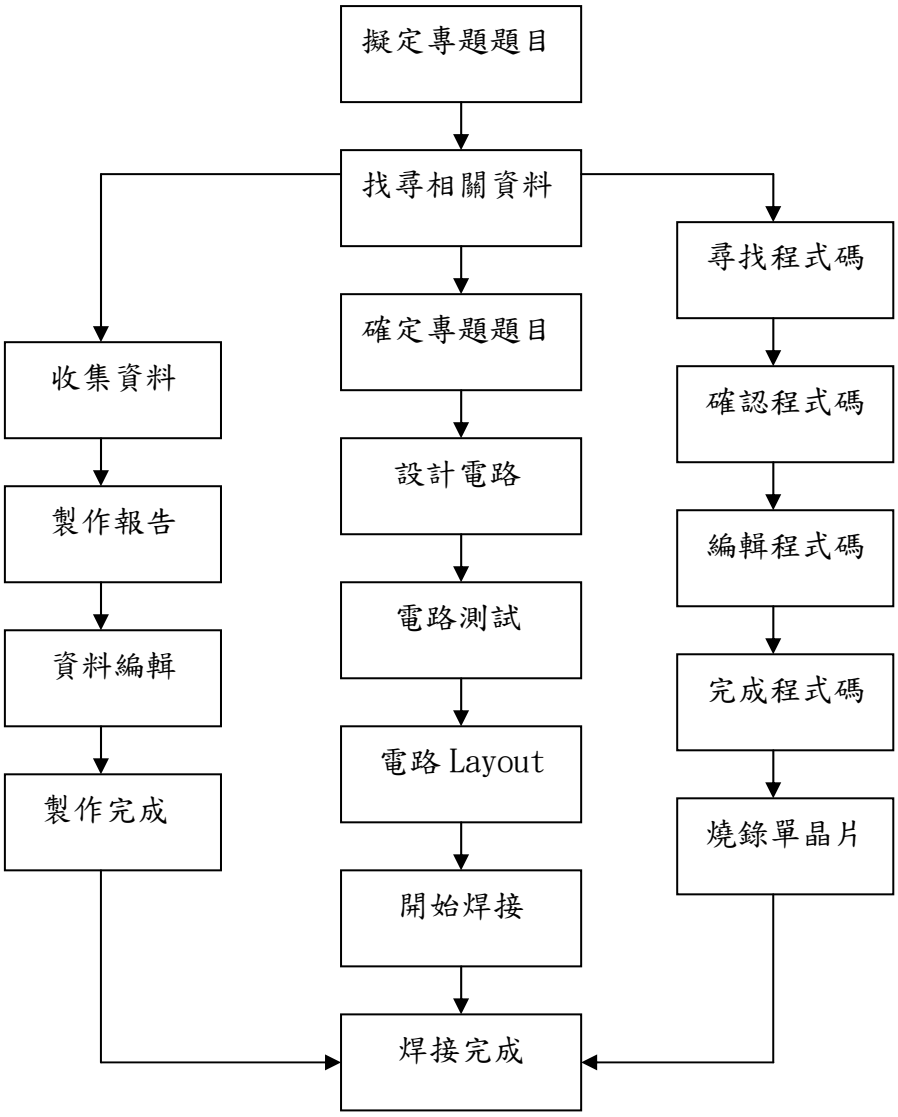

圖 3-2-1 流程圖

三、專題製作

表 3-3-1 專題製作計劃書

<table>
<tr><td colspan="2">專題型別</td><td colspan="2">個人型專題　　團隊型專題</td></tr>
<tr><td colspan="2">專題性質</td><td colspan="2">單晶片控制研究</td></tr>
<tr><td colspan="2">科別／年級</td><td colspan="2">資訊 科 三 年級</td></tr>
<tr><td rowspan="2">專題名稱</td><td>中文名稱</td><td colspan="2">無線遙控風扇轉速控制器</td></tr>
<tr><td>英文名稱</td><td colspan="2">Wireless remote control fan warble speed controller</td></tr>
<tr><td colspan="2" rowspan="7">專題內容簡述</td><td colspan="2">常有過這樣的經驗：想調整一風扇的風量，就得趨前去調</td></tr>
<tr><td colspan="2">整，靠近調好感覺剛好的風量，退回到位置後，又感覺吹</td></tr>
<tr><td colspan="2">到的風量不對了，反覆來回幾次才能調好適當的風量，如</td></tr>
<tr><td colspan="2">果能在位置上透過無線的控制調整，直接調整到合適的風</td></tr>
<tr><td colspan="2">量，感覺就方便多了。</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">指導老師姓名</td><td colspan="2">葉忠賢 老師</td></tr>
<tr><td colspan="2" rowspan="2">參與同學姓名</td><td>黃信昌</td><td>劉名軒</td></tr>
<tr><td>曾聖豪</td><td></td></tr>
<tr><td colspan="2">專題執行日期</td><td colspan="2">101 年 9 月 1 日至 102 年 5 月 31 日</td></tr>
</table>

(1) 無線遙控風扇的功能與設定

無線遙控風扇， 用 8051 控制 2 位數的 LED 顯示器，能讓遙控器控制時清楚的知道當前的段數

(2) 硬體電路圖：無線遙控風扇轉速控制

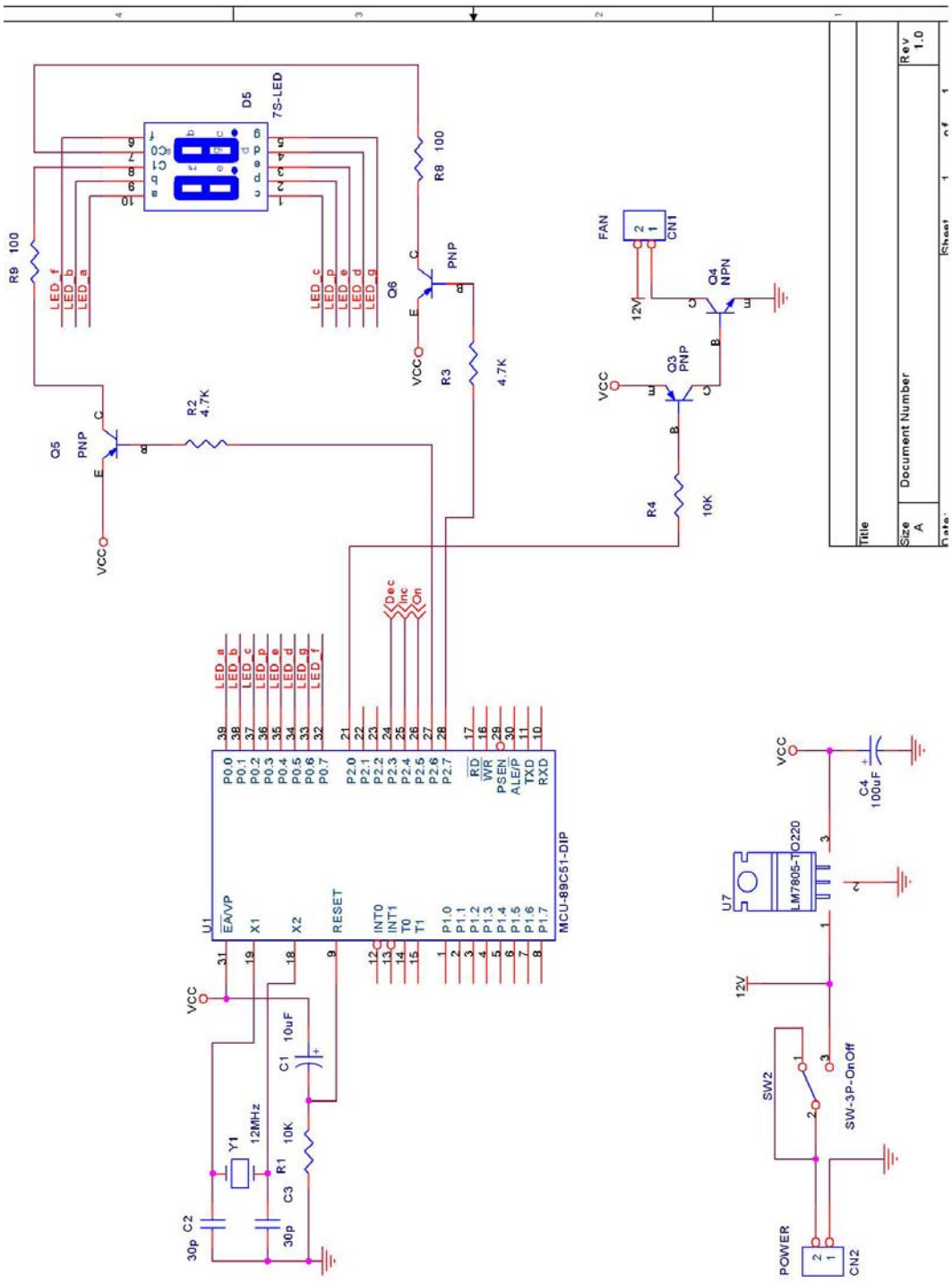

圖 3-3-1 無線遙控風扇轉速控制器完整電路圖

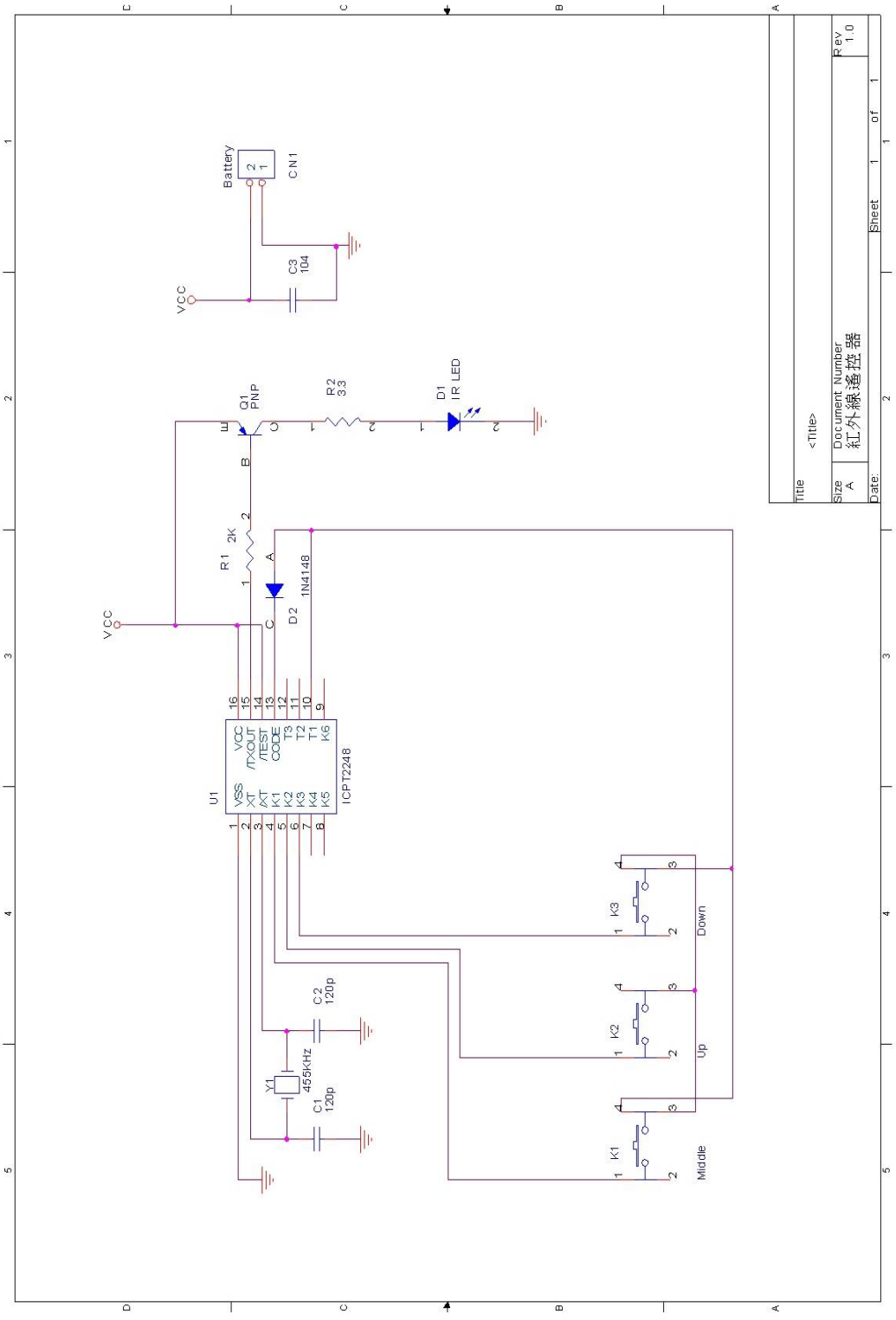

圖 3-3-2 遙控器完整電路圖

表 3-3-2 無線風扇轉速零件表

零件	名稱	數量
電晶體	PT2248	1
電容	120pF	2
電容	0.1uF	1
電阻	2K Ω	1
電阻	3.3Ω	1
電晶體	PNP	1
電晶體	1N4148	1
LED	IR 發射	1
按鈕開關		3
發射器	AT89C51	1
紅外線接收器		1
陶瓷電容	1000pF	1
陶瓷電容	30pF	2
電解質電容	10uF	1
電解質電容	100uF	1
電晶體	NPN	1
電晶體	PNP	4
七段顯示器	2 位數	1
風扇	12V	1
莫士座	2 PIN	1
單晶片	89C51	1

(3) 小組分工配置：

劉名軒負責硬體部分，電路設計、以及協助曾聖豪製作簡報部份

曾聖豪負責製作簡報及收集資料傳給黃信昌做參考文獻，而其餘時間協助劉名軒製作焊接在PCB板上。

黃信昌除了負責統整曾聖豪傳過來的資料外還要紀錄製作專題之過程，經過討論之後，再將資料統合。

至於零件就由我們三人一起去相關的零件商品街尋找。程式碼在一起討論，在請老師指導。簡報報告部分，由三人輪流報告。

肆、製作成果

我們小組決定專題，繪製設計電路圖，進而完成焊接整個電路；製作過程中，我們都由數位相機及相關電腦記錄下來，經過資料統整後，我們將呈現在專題報告中。 如下圖所示：

圖 4-1 專題製作報告設計

圖 4-2 專題製作焊接

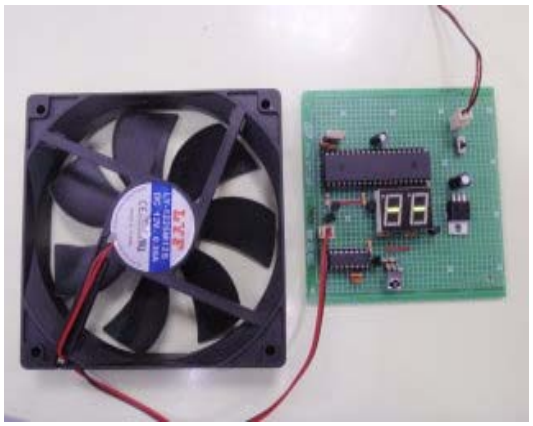

圖 4-3 電路成品圖(一)

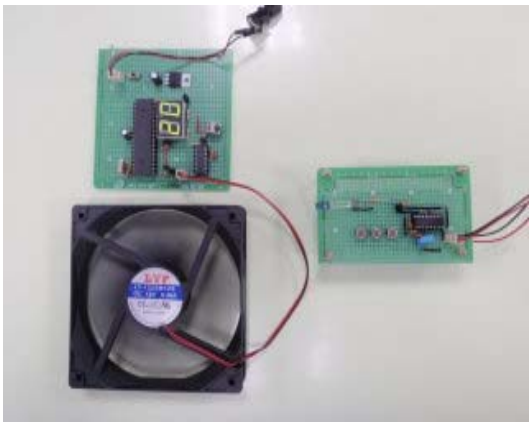

圖 4-4 電路成品圖(二)

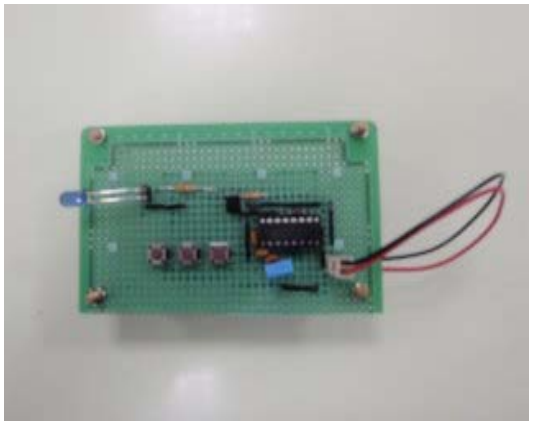

圖 4-5 電路成品圖(三)

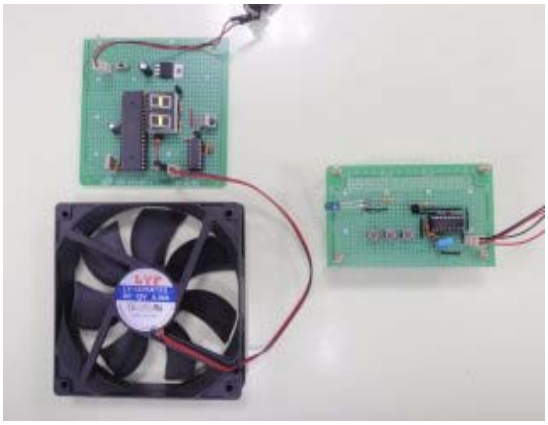

圖 4-6 電路成品圖(四)

伍、結論與建議

本單元是針對我們小組對製作專題的整的學習過程，做最後的統整與結論。讓日後的學弟妹們學習與參考。

一、結論

紅外線遙控，現在還是很普遍的電器用產品，我們親手作完這個專題，讓我們知道用紅外線遙控是很方便的一種東西，雖然在做的過程中有點辛苦，但是完成了這個作品一切都是直得的，讓我們小組一起分工合作這是我們專題最大的收穫。

本研究達成了用無線控制電風扇的目的，當速度到達 1 時電風扇將會動，段數持續上升的話電風扇的速度也會加快，同理當速度下降時電風扇也會變慢，而且當速度暫停時風扇將會停止，而我們的風量可依自己的喜好自由調整。

二、建議

(一) 製作專題的過程中，學到了很多在課堂上不會遇到的困難，如團隊的合作，以及解決問題的能力；再來就是做事要小心，如果團隊沒辦法配合分工來製作這個作品的話，那要完成這個專題就會很困難。

(二)雖然在此階段已成就我們的想法，但在未來我們要想的是如何將這項科技作品做的更加完善更加符合社會的需求。所以我們在結束此專題後，希望組員們依舊是在這方面上不斷學習，也期待專題製作對組員們來說不是結束，而是另一個新的起程。

(三)進行專題活動的學習，每個階段皆需完成一個學習報告，而單元學習的時間太少，希望能夠有充足的時間，能夠把作品做的更完整一點，跟老師的互動也會更佳。

參考文獻

1. 徐椿樑、陳輔賢，2004，8051/8951 理論與實務應用，台北縣：台科大圖書公司
2. 作郭庭吉，8051 單晶片微電腦專題製作，台北縣：台科大圖書公司
3. 作楊明豐，8051 單晶片設計實務：組合語言版，台北縣：台科大圖書公司
4. 鄭美珠、李鴻鵬，8051 單晶片C語言程式設計－使用Keil Cx51，台北縣：台科大圖書公司
5. 王廖明，電晶體電路，全華科技圖書股份有限公司

附錄一 無線遙控風扇轉速控制器之程式碼

【Main.c】

```
#include <reg51.h>
#include "tick.h"
#include "keypad.h"
#include "led7seg.h"
#include "pwm.h"

#define KEY_ON_OFF  KP_K3   //開關鍵
#define KEY_INC  KP_K2   //遞增鍵
#define KEY_DEC KP_K1   //遞減鍵

typedef enum
{
    OP_WaitKey,  //等待按鍵
    OP_FanTriggerDelay,    //風扇起動觸發延遲
    OP_End
};

void DisplayLevel(unsigned char);

unsigned char OpMode; //作業程序碼
unsigned char OpRet;    //返回作業程序碼
unsigned char KeyIn;    //儲存目前按鈕狀態
unsigned char KeyRec;  //記錄前次按鈕狀態
unsigned char FanSpeedLevel;     //轉速段數
unsigned int tTriggerDelay;   //風扇起動觸發延遲時間
bit fFanOn;    //風扇開啟旗號

main()
{
    InitTick();      //初始系統鐘控計時
    InitKeypad();  //初始按鍵掃瞄
    InitLed7Seg();//初始七段顯示
```

```
    InitPWM();    //初始 PWM 控制

    FanSpeedLevel = 1;      //初始段數
    fFanOn = 0;   //初始為風扇關閉狀態

    KeyRec = KP_None;     //清除鍵盤碼記錄
    DisplayLevel(FanSpeedLevel);     //顯示段數值
    OpMode = OP_WaitKey;      //初始為等待按鍵

    while(1)
    {
        KeyIn = GetKeyScanCode(); //讀取按鍵狀態
        if(KeyIn == KeyRec)     //檢查按鍵狀態更新
            KeyIn = KP_None; //狀態無改變, 設為無效碼
        else
            KeyRec = KeyIn;   //記錄按鍵狀態

        switch (OpMode)   //依作業碼處理相對作業
        {
            case OP_WaitKey: //等待按鍵
                if(KeyIn == KEY_ON_OFF) //檢查是否是設定鍵
                {
                    if(fFanOn == 0)
                    {     //風扇由靜止變慢轉時,需先用較大些的工作週期來
啟動
                        PerformDuty(25000);    //風扇預起動觸發
                        tTriggerDelay = GetSystemTick() + MS_500;      //設有
風扇起動觸發延遲時間
                        fFanOn = 1;   //設定開啟旗號
                        DisplayLevel(FanSpeedLevel);     //顯示段數值
                        OpRet = OpMode; //儲存返回作業碼
                        OpMode = OP_FanTriggerDelay; //下一步:風扇起動
觸發延遲
                    }
                    else
                    {     //風扇關閉
                        DisableDuty();//關閉風扇
                        fFanOn = 0;   //清除開啟旗號
```

```
                DisplayLevel(FanSpeedLevel);    //顯示段數值
            }
        }
        else if((fFanOn == 1) && (KeyIn == KEY_INC))      //開啟狀態
且按遞增鍵
        {
            if(FanSpeedLevel != PWM_LEVEL_TOTAL)    //段數是否
不是最大段
            {
                FanSpeedLevel++; //段數遞增
                DisplayLevel(FanSpeedLevel);    //顯示段數值
                DutyByScale(FanSpeedLevel);    //起動溫差對應的轉
速
            }
        }
        else if((fFanOn == 1) && (KeyIn == KEY_DEC))     //開啟狀態
且按遞減鍵
        {
            if(FanSpeedLevel != 1)  //段數是否不是 1
            {
                FanSpeedLevel--;   //段數遞減
                DisplayLevel(FanSpeedLevel);    //顯示段數值
                DutyByScale(FanSpeedLevel);    //起動溫差對應的轉
速
            }
        }
        break;

    case OP_FanTriggerDelay:    //風扇起動觸發延遲
        if(GetSystemTick() == tTriggerDelay)  //檢查風扇起動觸發延遲
時間是否到時
        {
            DutyByScale(FanSpeedLevel);    //起動溫差對應的轉速
            OpMode = OP_WaitKey;      //延遲後回到等待按鍵
            OpMode = OpRet; //返回作業程序
        }
        break;
    }
```

```
            ScanLed7Seg();      //掃瞄七段顯示
        }
}

/*-------------------------------------
   顯示段數
-------------------------------------*/
void DisplayLevel(unsigned char val)
{
        if(fFanOn == 0)      //是否關閉
        {
            DspSymbol(0,SYM_DASH); //顯示  '-'
            DspSymbol(1,SYM_DASH); //顯示  '-'
        }
        else
            DspLocDecDigi(val,2,0,ZERO_ON);    //顯示段數值
}
```

【PWM.c】

```
#define PWM_C

#include <reg51.h>
#include "system.h"
#include "pwm.h"

#define COUNT_US      1           /*計時器遞增 1 計數的 1 timer clocks = 1 us*/
#define DUTY_ACTIVE             0     //週期工作期的信號
#define DUTY_INACTIVE   1     //週期非工作期的信號

/*-------------------------------------------------
   PWM 參數定義
-------------------------------------------------*/
#define    PWM_CYCLE      30000     // PWM 總週期 (us)
#define    DUTY_MIN   10000     // 工作週期最短時間 (us)
#define DUTY_ISR_MIN      22    //中斷的消耗時間 (us)
#define    DUTY_STEP ((PWM_CYCLE - DUTY_MIN - DUTY_ISR_MIN) /
PWM_LEVEL_TOTAL)      //每段的時間差值

sbit PWMOutputPin =    P2^0;      //PWM 輸出控制接腳

bit    fDutySignal;   //工作週期旗號
unsigned int tDutyOn;    //工作週期時間
unsigned int tDutyOff;   //非工作週期時間

/*-------------------------------------------------
 PWM 產生計時器中斷常式
-------------------------------------------------*/
void timer1 (void) interrupt 3 using 2
{
      TR1 = 0; //停止計時

      fDutySignal = !fDutySignal;  //工作週期旗號反向
      PWMOutputPin = fDutySignal;     //送出信號到控制腳
      if (fDutySignal == DUTY_ACTIVE) //檢查工作週期旗號是否是非工作週期
```

```
    {
        TL1 = tDutyOn;     //載入工作週期時間的低位元組值
        TH1 = tDutyOn >> 8;    //載入工作週期時間的高位元組值
    }
    else //工作週期旗號為非工作週期
    {
        TL1 = tDutyOff;    //載入非工作週期時間的低位元組值
        TH1 = tDutyOff >> 8;   //載入非工作週期時間的高位元組值
    }

    TR1 = 1; //起動計時
}

/*-----------------------------------------------
  以段數調整工作週期
  輸入 : 段數值
-----------------------------------------------*/
void DutyByScale(unsigned char scale)
{
    unsigned int duty_high;

    if(scale >= PWM_LEVEL_TOTAL)    //檢查輸入段數值是否達到超過總段數
值
        FullDuty();     //全工作週期輸出
    else if(scale == 0)
        DisableDuty();//關閉
    else
    {
        duty_high = DUTY_STEP * scale;//計算段數對應時間
        duty_high += DUTY_MIN;  //加上最小工作週期時間
        PerformDuty(duty_high);     //輸出執行工作週期
    }
}

/*-------------------------------------------------------
  輸出執行工作週期
-------------------------------------------------------*/
void PerformDuty(unsigned int duty)
```

```
{
    tDutyOn = 65536 - (duty * COUNT_US);   //計算工作週期時間
    tDutyOff = 65536 - ((PWM_CYCLE - duty) * COUNT_US);   //計算非工作週
期時間
    TR1 = 0; //停止計時
    TF1 = 0; //清除計時溢位旗號
    ET1 = 1; //計時中斷致能
    TL1 = tDutyOn;   //載入工作週期時間的低位元組值
    TH1 = tDutyOn >> 8;   //載入工作週期時間的高位元組值
    fDutySignal = DUTY_ACTIVE;  //設定工作週期旗號為工作週期
    PWMOutputPin = fDutySignal;   //送出信號到控制腳
    TR1 = 1; //起動計時
}

/*-------------------------------------------------
  全工作週期輸出
-------------------------------------------------*/
void FullDuty(void)
{
    fDutySignal = DUTY_ACTIVE;  //設定工作週期旗號為工作週期
    PWMOutputPin = fDutySignal;   //送出信號到控制腳
    TR1 = 0; //停止計時
    TF1 = 0; //清除計時溢位旗號
    ET1 = 0; //計時中斷除能
}

/*-------------------------------------------------
  無工作週期
-------------------------------------------------*/
void DisableDuty(void)
{
    fDutySignal = DUTY_INACTIVE;    //設定工作週期旗號為非工作週期
    PWMOutputPin = fDutySignal;   //送出信號到控制腳
    TR1 = 0; //停止計時
    TF1 = 0; //清除計時溢位旗號
    ET1 = 0; //計時中斷除能
}
```

```
/*------------------------------------------------
   初始 PWM 控制
------------------------------------------------*/
void InitPWM(void)
{
      fDutySignal = DUTY_INACTIVE;      //設定工作週期旗號為非工作週期
      PWMOutputPin = fDutySignal;    //送出信號到控制腳
      EA = 1;   //所有中斷開關致能
      TMOD &= 0xf;      //清除計時模式控制位元
      TMOD |= 0x10;     //  設定為 16 位元計時
```

```
      //PT1 = 1;       //PWM  中斷設定為最高優先權
}
```

【PWM.h】

```
#define    PWM_LEVEL_TOTAL 20    //總段數
void InitPWM(void);
void DutyByScale(unsigned char);
void PerformDuty(unsigned int);
void FullDuty(void);
void DisableDuty(void);
```

【Keypad.c】

```
#define    KEYPAD_C

#include <reg51.h>
#include "keypad.h"

#define TIME_Bounce   1     //防彈跳延遲

typedef enum KeyPhaseEnum {
      KB_Read,            //讀取階段
      KB_Debounce         //防彈跳階段
} KeyPhaseType;

      unsigned char  KeyPhase;      //按鍵處理階段碼
      unsigned char  KeyCode;       //按鍵碼
      unsigned char        KeyRecord;    //按鍵碼記錄
      unsigned char  BounceTime;  //彈跳過濾計數

/*-------------------------
  初始掃瞄參數
-------------------------*/
void InitKeypad(void)
{
      KeyPort |= KP_MASK;  //設定輸入腳
      KeyPhase = KB_Read;   //讀取階段
      KeyCode = KP_None;    //無效碼
      KeyRecord = KP_None;//無效碼
      BounceTime = 0;    //初始彈跳過濾計數
}

/*-------------------------
  讀取按鍵碼
-------------------------*/
unsigned char GetKeyScanCode(void)
{
      return (KeyCode);  //返回按鍵碼
```

```
}

/*-------------------------
  掃描按鍵處理副程式
-------------------------*/
void ScanKeypad(void)
{
    switch ( KeyPhase )
    {
        case KB_Read:      //讀取階段
            if ((KeyCode = GetKey()) != KeyRecord)    //讀取並檢查狀態是否改
變
            {    //按鍵狀態改變
                KeyRecord = KeyCode; //記錄狀態
                BounceTime = TIME_Bounce;     //設定彈跳過濾計數值
                KeyPhase = KB_Debounce;  //設定下個階段為過濾彈跳階段
            }
            break;

        case KB_Debounce:      //防彈跳階段
            if (--BounceTime == 0)  //彈跳過濾計數減 1,並檢查是否為 0
            {    //彈跳過濾計數到時,表示按鈕狀態已穩定
                KeyPhase = KB_Read;  //設定下個階段為讀取階段
            }
            break;
    }
}

/*-------------------------
  取得按鍵掃瞄碼
-------------------------*/
unsigned char GetKey(void)
{
    return (KeyPort & KP_MASK);    //返回按鍵碼
}
```

【Keypad.h】

```
#define KeyPort     P2
typedef enum KeyButtonEnum {
      KP_None= 0x0,
      KP_K1          = 0x8,
      KP_K2          = 0x10,
      KP_K3          = 0x20
} KeyButton;

#define    KP_MASK          (KP_K1 | KP_K2 | KP_K3)

extern void InitKeypad(void);
extern void ScanKeypad(void);
unsigned char GetKey(void);
unsigned char GetKeyScanCode(void);
```

【Led7Seg.c】

```
#include <reg51.h>
#include "led7seg.h"

#define DigiDataPort P0//七段顯示器顯示碼資料埠
#define DIGI_NUM      2     //位數數量
#define DIGI_ON       0     //位數顯示
#define DIGI_OFF 1     //位數不顯示
#define DIGI_DOT0x08

sbit Digi0CtrlPin = P2^7;      //位數 0 控制腳
sbit Digi1CtrlPin = P2^6;      //位數 1 控制腳

unsigned char DigiScanIndex;      //位數掃瞄索引值
unsigned char LedDecode[DIGI_NUM];     //位數七段顯示解碼儲存位置

code char LedDecodeTable[] = {
                    //7-段 LED 顯示字型表
                    //       fgde-pcba
    ~0xb7,          //0 = x.xx-.xxx
    ~0x06,          //1 = ....-.xx.
    ~0x73,          //2 = .xxx-..xx
    ~0x67,          //3 = .xx.-.xxx
    ~0xc6,          //4 = xx..-.xx.
    ~0xe5,          //5 = xxx.-.x.x
    ~0xf5,          //6 = xxxx-.x.x
    ~0x07,          //7 = ....-.xxx
    ~0xf7,          //8 = xxxx-.xxx
    ~0xe7,          //9 = xxx.-.xxx
    ~0xd7,          //a = xx.x-.xxx
    ~0xe4,          //b = xxx.-.x..
    ~0xb1,          //c = x.xx-...x
    ~0x6e,          //d = .xx.-.xx.
    ~0x8f,          //e = x...-.xxx
    ~0x8b,          //f = x...-...x
    ~0x40,          //10= .x..-....
```

```
    ~0x00,          //11=  不顯示
};

/*-------------------------------------------
  初始七段顯示器介面及相關控制參數
-------------------------------------------*/
void InitLed7Seg(void)
{
    unsigned char cnt;

    AllDigiOff();  //關閉所有位數顯示
    for(cnt = 0;cnt != DIGI_NUM; cnt++)
        DigiDot(cnt,DOT_OFF);      //小數點不顯示
    DigiScanIndex = 0; //初始位數掃瞄索引值
}

/*-----------------------------------------------
  顯示指定位數值到指定的位置
  輸入 : val =  數值
          digi =  位數
          loc =  位置
          fhdig =  高位數零顯示旗號
-----------------------------------------------*/
void DspLocDecDigi(unsigned char val,unsigned char digi,unsigned char loc,bit
fhdig)
{
    unsigned char hdig;
    unsigned char eten;
    unsigned char tcnt;
    unsigned char dot;

    while(digi != 0)     //位數處理迴圈
    {
        eten = 1;  //10 的倍數基值
        for(tcnt = 0; tcnt < digi-1; tcnt++)  //位數迴圈
            eten *= 10;     //累計 10 的倍數值

        hdig = val/eten;      //計算高位數值
```

```
        //LedDecode[digi + loc - 1] &= DIGI_DOT; //保留小數點燈號位元
        dot = LedDecode[digi + loc - 1] & DIGI_DOT;   //保留小數點燈號位元

        if(fhdig || (hdig != 0) || (digi == 1))//檢查高位數零顯示或高位數值不為 0
或第 1 位數
        {    //顯示此位數
            LedDecode[digi + loc - 1] = LedDecodeTable[hdig];   //位數顯示解碼
            fhdig = 1;//設定最高位數顯示旗號
        }
        else
            LedDecode[digi + loc - 1] = LedDecodeTable[SYM_OFF]; //位數顯示
不顯示

        //放回小數點狀態
        LedDecode[digi + loc - 1] &= ~DIGI_DOT; //清除小數點
        LedDecode[digi + loc - 1] |= dot;  //放回小數點狀態

        val -= (hdig*eten);  //減去已顯示的值
        digi--;    //位數數量遞減
    }
}

/*-----------------------------------------------
  顯示指定位數值到指定的位置
  輸入 : val =  數值
          digi =  位數
          loc =  位置
          fhdig =  高位數零顯示旗號
-----------------------------------------------*/
#if 0
void DspLocHexByte(unsigned char val,unsigned char loc)
{
    unsigned char dot;
    dot = LedDecode[loc + 1] & DIGI_DOT;    //保留小數點燈號位元
    LedDecode[loc + 1] = LedDecodeTable[val>>4]; //高位數解碼
    //放回小數點狀態
    LedDecode[loc + 1] &= ~DIGI_DOT;  //清除小數點
    LedDecode[loc + 1] |= dot;   //放回小數點狀態
```

```
    dot = LedDecode[loc] & DIGI_DOT;   //保留小數點燈號位元
    LedDecode[loc] = LedDecodeTable[val&0xf];    //低位數解碼
    //放回小數點狀態
    LedDecode[loc] &= ~DIGI_DOT; //清除小數點
    LedDecode[loc] |= dot;  //放回小數點狀態
}
#endif

/*-------------------------------------------------
  顯示單位數副程式
-------------------------------------------------*/
void DspSymbol(unsigned char loc,unsigned char sym)
{
    unsigned char dot;
    dot = LedDecode[loc] & DIGI_DOT;   //保留小數點燈號位元

    LedDecode[loc] = LedDecodeTable[sym];   //取得單位數七段顯示解碼

    //放回小數點狀態
    LedDecode[loc] &= ~DIGI_DOT; //清除小數點
    LedDecode[loc] |= dot;  //放回小數點狀態
}

/*-------------------------------------------------
  指定位數的小數點亮滅控制
-------------------------------------------------*/
void DigiDot(unsigned char digi_index,bit fdot)
{
    if (fdot == DOT_ON)    //檢查點燈號設定旗號是否設定
        LedDecode[digi_index] &= ~DIGI_DOT;    //點燈號亮
    else
        LedDecode[digi_index] |= DIGI_DOT; //點燈號滅
}

/*-------------------------------------------------
  六位數七段 LED 顯示掃瞄
-------------------------------------------------*/
```

```
void ScanLed7Seg(void)
{
    DigiDataPort = 0xff;     //無顯示輸出
    Digi0CtrlPin = DIGI_OFF;   //位數 0 關閉
    Digi1CtrlPin = DIGI_OFF;   //位數 1 關閉

    DigiDataPort = LedDecode[DigiScanIndex];//送出數字顯示碼
    if (DigiScanIndex == 0)
        Digi0CtrlPin = DIGI_ON;    //打開位數 0 顯示
    else if(DigiScanIndex == 1)
        Digi1CtrlPin = DIGI_ON;    //打開位數 1 顯示

    DigiScanIndex++;  //位數掃瞄索引值遞增
    if (DigiScanIndex == DIGI_NUM)     //檢查位數掃瞄索引值是否到達位置值
        DigiScanIndex = 0;//重置位數掃瞄索引值
}

/*--------------------------------------
  關閉所有位數顯示
 --------------------------------------*/
void AllDigiOff(void)
{
    unsigned char cnt;

    for(cnt = 0;cnt != DIGI_NUM; cnt++)
        DspSymbol(cnt,SYM_OFF); //不顯示
}

/*--------------------------------------
  不顯示指定位數
 --------------------------------------*/
 #if 0
void InvisibleDigi(unsigned char dnum,unsigned char loc)
{
    while(dnum != 0)   //位數迴圈
        LedDecode[dnum-- + loc - 1] |= LedDecodeTable[SYM_OFF];  //位數顯示
解碼
}
```

```
#endif
```

【Led7Seg.h】

```
#define ZERO_OFF      0    //高位數零不顯示
#define ZERO_ON       1    //高位數零顯示
#define DOT_OFF       0    //小數點亮
#define DOT_ON        1    //小數點不亮

#define SYM_DASH    0x10
#define SYM_OFF     0x11

void InitLed7Seg(void);
void ScanLed7Seg(void);
void DigiDot(unsigned char,bit);
void DspLocDecDigi(unsigned char,unsigned char,unsigned char,bit fhdig);
void DspLocHexByte(unsigned char,unsigned char);
void DspSymbol(unsigned char,unsigned char);
void AllDigiOff(void);   //關閉所有位數顯示
void InvisibleDigi(unsigned char,unsigned char); //不顯示指定位數
```

【Tick.c】

```
/*-----------------------------------------------
   系統鐘控計時中斷服務常式
-----------------------------------------------*/
#include <reg51.h>
#include "system.h"
#include "tick.h"
#include "keypad.h"

#define TICK_INTERRUPT_PERIOD_CNT
      (((XTAL*TICK_INTERRUPT_PERIOD_MS)/1000)/12)
#define MICRO_ADJUST     22   //鐘控計時準確度微調,值減少則調慢
#define    TICK_PERIOD
      ((65536-TICK_INTERRUPT_PERIOD_CNT)+MICRO_ADJUST)

unsigned int SystemTick;      //系統計時值
unsigned int StableTick; //檢查計時值穩定

/*============================================================
      讀取系統計時時間
============================================================*/
unsigned int GetSystemTick(void)
{
      do
      {
            StableTick = SystemTick;      //讀取鐘控計時值
      }while(StableTick != SystemTick);      //若無改變則跳出
      return(StableTick); /*  傳回系統鐘控值  */
}

/*-----------------------------------------
   初始系統鐘控計時
-----------------------------------------*/
void InitTick(void)
{
      SystemTick = 0;     //清除系統鐘控計時值
```

```
    TMOD &= 0xf0;   /* 清除計時模式控制位元 */
    TMOD |= 0x1;/* 設定 16 位元計時 */
    TR0 = 0; /* 停止計時 */
    TF0 = 0; /* 清除計時溢位旗號 */
    TH0 = TICK_PERIOD >> 8; /* 載入系統鐘控計時值高位元組 */
    TL0 = (unsigned char)TICK_PERIOD; /* 載入系統鐘控計時值低位元組 */
    PT0 = 1; //系統鐘控中斷為最高優先權
    TR0 = 1; /* 開始計時 */
    ET0 = 1; /* 致能計時器 0 中斷 */
    EA = 1;   /* 致能中斷開關 */
}

/*----------------------------------------
  系統鐘控計時中斷服務常式
----------------------------------------*/
void timer0 (void) interrupt 1 using 1
{
    TR0 = 0; //停止計時
    TH0 = TICK_PERIOD >> 8; //重載高位元組
    TL0 = (unsigned char)TICK_PERIOD; //重載低位元組
    TR0 = 1; //開始計時

    SystemTick++;      //遞增系統計時時間
    ScanKeypad();//掃瞄按鍵信號
}
```

【Tick.h】

```
#define TICK_INTERRUPT_PERIOD_MS  10

#define MS_100     (100/TICK_INTERRUPT_PERIOD_MS)
#define MS_200     (2 * MS_100)
#define MS_300     (3 * MS_100)
#define MS_400     (4 * MS_100)
#define MS_500     (5 * MS_100)
#define MS_600     (6 * MS_100)
#define MS_700     (7 * MS_700)
#define SEC_1      (10 * MS_100)

void InitTick(void);
    unsigned int GetSystemTick(void);
```