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

太陽能溫度控制風扇轉速器

學生姓名：黃秉福
郭昇昌
鍾育軒
指導老師：葉忠賢 老師

中華民國 102 年 05 月

誌　謝

進入高中職業學校開始學習技藝課程，面對一不同的技能專長領域，除了艱辛，更覺漫長。如今，我們小組幸運地已來到了三年級，回頭俯看這一路上的學習歷程，細數點滴。我想，是我們該將每個階段的感動，留下紀錄的時候了。

此次的專題讓我們認識了很多的零件，也因此讓我們翻閱了許多我們平常不常翻閱過的書籍，讓我們對基本的電子電路元件的使用方法及應用能有一個基本的概念，在生活上也給了我們相當多的幫助。在製作專題過程中，因為平日實習課老師要求我們自己參照課本的電路圖進行 Layou 及銲接，所以繪畫電路圖及焊接對我們來說並不陌生，加上葉忠賢老師的指導，讓我們順利的完成電路，葉老師在做解釋及教學的態度讓我們很容易得抓住要點；以及感謝簡琨祥老師，在報告書面的流暢度及完整度給我們預先的模擬製作，讓我們輕易得完成，在此特別感謝。由此次的專題製作，讓我們了解到平日知識累積的重要，因有老師們的指導及付出讓我們受益良多，才得以使我們在社會上能夠站得住腳，非常感謝。

黃秉福、郭昇昌、鍾育軒　謹上　2013/05

太陽能溫度控制風扇轉速器

摘要

現在氣候不如往常，日夜溫差大，生活的目的不是追求完美嗎?如果我們想要良好的溫度環境，然後去加以控制，那我們就需要透過溫度感測器來做溫度的控制，但是溫度感測器種類很多，而這次我們是利用 DS1821 來進一步溫度感測和分析，最後交由 8051 去進一步判斷，並使用 PWM 方式控制電晶體開關，來控制風扇的轉速。

利用感測器的方法來測量溫度,不只在日常生活中隨處可見(如冷氣、電冰箱...等),所以我們用一些電子零件去加以控制和顯示。

在目前科技發達以及生活便利的現在社會，不論是在日常生活中，或者工商業的發展，甚至是學術之研究。溫度對我們而言，是一項很重要的參數，是不可或缺且與我們息息相關的。在生活上，天氣冷或熱時，我們常用冷暖氣機來控制溫度。

使溫度控制在一定的範圍之內，以達到恆溫之目的，來讓我們覺得舒適。在工業及學術研究上，溫度所扮演的角色能決定產品的好壞、公共安全、實驗數據的可靠性...等。

由此可得知溫度的重要性，因此倘若未來人們能夠更善加的利用溫度、了解溫度，相信將會一直持續的為人類帶來更多的福祉。此專題的目的，讓我們更深刻的了解到溫度感測元件的動作原理，以即認識更多的感測元件；也讓我們對於感測元件電路有更多的認知。因此，經過此專題的製作，應當對於培養我們有關於感測與控制方面的思考模式、設計概念，會有相當大的助益，令我們受益匪淺。

關鍵詞： DS1821 溫度晶體、2 位數 LED 七段顯示器、風扇控制。

目　錄

表目錄

圖目錄

壹、前言

一、製作動機

現在氣候不如往常，日夜溫差大，生活的目的不是追求完美嗎?如果我們想要良好的溫度環境，然後去加以控制，那我們就需要透過溫度感測器來做溫度的控制，但是溫度感測器種類很多，而這次我們是利用DS1821 來進一步溫度感測和分析，最後交由8051 去進一步判斷，並使用PWM方式控制電晶體開關，來控制風扇的轉速。利用感測器的方法來測量溫度，不只在日常生活中隨處可見(如冷氣、電冰箱…等)，所以我們用一些電子零件去加以控制和顯示。

二、製作目的

在目前科技發達以及生活便利的現在社會，不論是在日常生活中，或者工商業的發展，甚至是學術之研究。溫度對我們而言，是一項很重要的參數，是不可或缺且與我們息息相關的。在生活上，天氣冷或熱時，我們常用冷暖氣機來控制溫度。使溫度控制在一定的範圍之內，以達到恆溫之目的，來讓我們覺得舒適。在工業及學術研究上，溫度所扮演的角色能決定產品的好壞、公共安全、實驗數據的可靠性…等。由此可得知溫度的重要性，因此倘若未來人們能夠更善加的利用溫度、了解溫度，相信將會一直持續的為人類帶來更多的福祉。此專題的目的，讓我們更深刻的了解到溫度感測元件的動作原理，以即認識更多的感測元件；也讓我們對於感測元件電路有更多的認知。因此，經過此專題的製作，應當對於培養我們有關於感測與控制方面的思考模式、設計概念，會有相當大的助益，令我們受益匪淺。

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

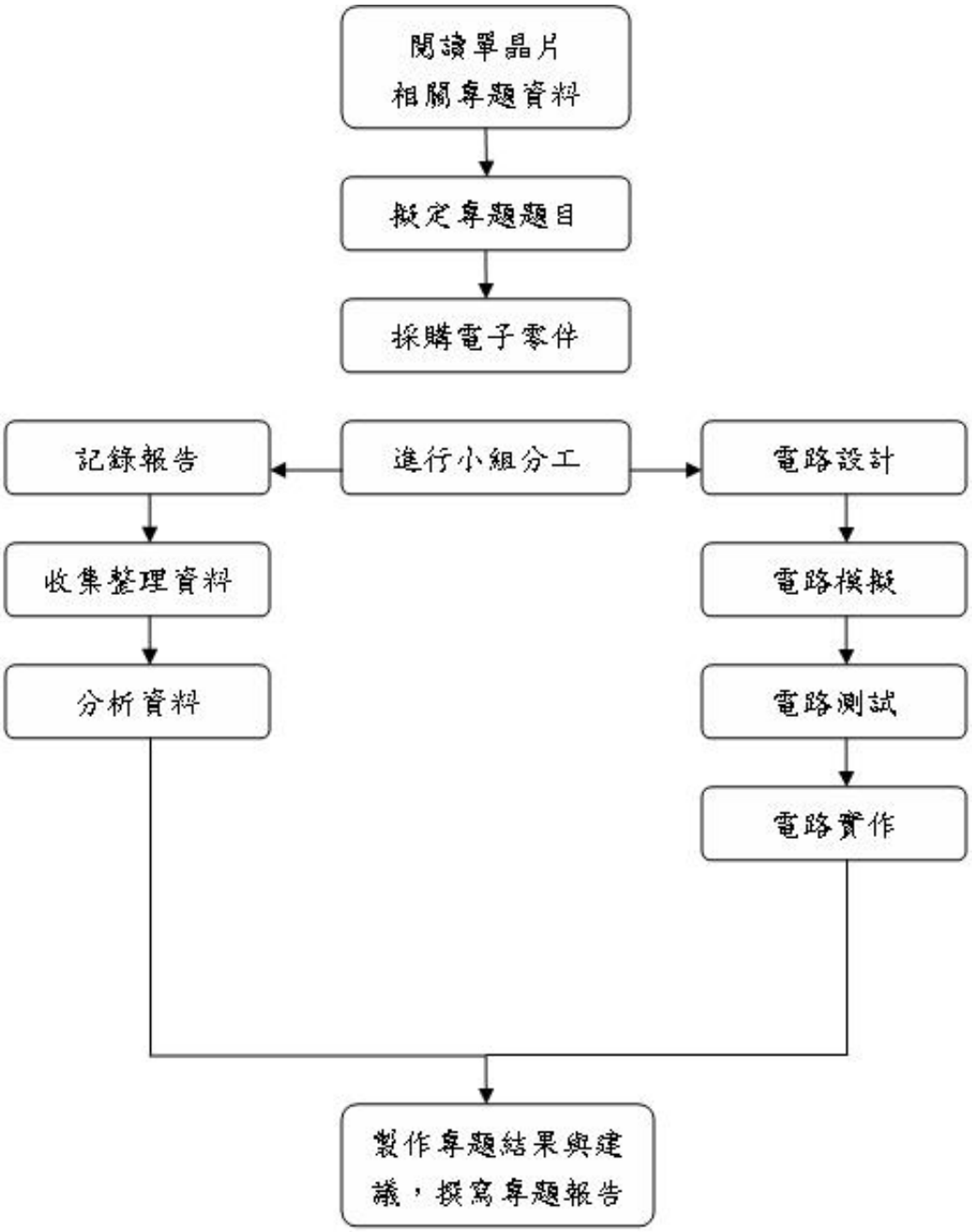


圖 1-3-1 溫度風速轉速器製作流程圖

四、製作預期成效

我們小組雖是第一次進行合作製作專題-風扇的溫度控制，雖擔心可能會無法成功，但有老師的協助及同學們的互相協助，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：

(1) 透過風扇轉速控制可進行自動調節室內的溫度。

(2) 當室溫高於設定溫度時，轉速越快，使室溫下降或上升，下降低於設定溫度時，停止轉速。

(3) 按停止後，停止運轉。

(4) 當風扇的溫度控制完成後，可以透過自己的需求，輸入需求的溫度，將自動調節。

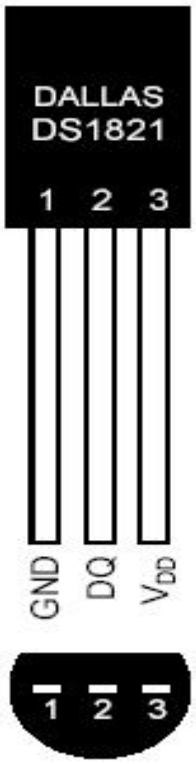
貳、理論探討

一、電子相關零組件

(一)DS1821 溫度感測

DS1821 使用 2.7V 到 5.5V 的工作電壓，使用 DQ Pin 與 MCU（微處理控制器）做通訊介面，加上接地腳共 3 支接腳。MCU 可以只使用一支 I/O 控制腳，連接到 DS1821 DQ 腳，來對 DS1821 做所有的控制，所傳送的命令或資料順序為：

1 整合性晶體,無需外加零件。
2 單接腳,通訊介面,節省 MCU(微處理器)的輸出/入腳使用。
3 偵測溫度範圍為-55 度 C 到+125 度 C。
4 提供 8 位元攝式溫度測量,解析度 1 度。
5 溫度轉換成數位的時間在 1 秒內。



GND：接地
DQ：數位資料輸入/輸出
VDD：工作電壓輸入

圖 2-1-1 DS1821 晶體接腳圖

1 起始（Reset 信號 ＋Presence 信號）。

2 傳送 8 位元命令給 DS1821。
3 依命令的特性，傳送或接收 8 位元資料。
起始：

傳送命令給 DS1821 前，必須對 DQ Pin 做其特定順序的起始動作，此順序動作包含 1 個由 MCU 送出的低電位重置脈衝，然後由 DS1821 送出 1 個稱為“Presence”的低電位脈衝，讓 MCU 知道 DS1821 有存在，並準備接收 MCU 送來的命令。

當 MCU 送出 Reset 脈衝時，要將此 1-wire 的 bus 拉到 LOW 電位，並持續最少 480 us，然後再回到 High 電位，然後讀取 1-wir bus 狀態。在 DS1821 測到 1-wire bus 由 LOW 變成 High 開始，會等待 15 - 60 us，然後將 1-Wire bus 拉成 LOW 電位 60 - 240 us，然後回到 High 電位，此 60 - 240 us 的 LOW 電位信號，就稱為 Presence 脈衝。

讀取資料：

當 8051 起使 1-Wire bus 後，即可對 DS1821 讀取資料，讀取資料前，要傳給 DS1821 一個 8 位元的讀取命令，表示 8051 要讀取某資料，如：讀取溫度高位元組值，或讀取 DS1821 內部狀態值。DS1821 在收到讀取命令後，才會以讀取命令相對應的資料透過 1-Wire bus 傳送給 8051。

寫入資料：

寫入資料到 DS1821 順序也是一樣，在 8051 起始 1-Wire bus 後，先傳給 DS1821 一個 8 位元的寫入命令，然後再傳 8 位元的資料給 DS1821，而當 DS1821 收到第一個寫入命令後，會再接著接收下一個資料；然後依寫入命令的種類，將第二個收到的資料做適當的處理。

起始、讀取資料、寫入資料，這三種動作就是 8051 對 DS1821 資料存取的主要三個動作類型，歸納一下，8051 對 DS1821 讀取資料的步驟順序為：

1. Reset (8051 發出)
2. Presence (DS1821 發出)
3. 讀取命令 (8051 發出)
4. 讀取資料 (DS1821 發出)

Reset：8051 發出約 15-60 us 長的低電位脈衝。
Presence：DS1821 回應一個約 60-240 us 長的低電位脈衝。
讀取命令：8051 送出 8 位元的讀取命令。
讀取資料：DS1821 依讀取命令的種類，傳送回 8 位元的相對應資料。

8051 對 DS1821 寫入資料的步驟為：

Reset：8051 發出約 15-60 us 長的低電位脈衝。

Presence：DS1821 回應一個約 60-240 us 長的低電位脈衝。

寫入命令：8051 送出 8 位元的寫入命令。

寫入資料：8051 送出 8 位元的資料。

寫入 8 位元命令及寫入 8 位元資料到 DS1821 的細部動作在與寫入 8 位元資料到 ds1821，是依低電位時間的長短來表示 1 或 0，而不是看 1-Wirs Pin 是高電壓位還是低電位，說明如下：

寫 1 : 8051 將 1-Wire bus 拉到 LOW 電位，並在 15 us 內要放開 1-Wire bus 回到 HIGH 電位。

寫 0 : 8051 將 1-Wire bus 拉到 LOW 電位，並保持最少 60 us 的時間。

從 DS1821 讀取 8 位元資料：

DS1821 是在 8051 傳送 8 位元讀取命令後，才會傳送 8 位元資料給 8051，所以第一個 8 位讀取命令的動作與傳送資料到 DS1821 是相同的。

在 8051 傳送完 8 位元讀取命令，然後放開 1-Wire bus 1us 後，再拉 1-Wire bus 1us，表示開始要讀取 DS1821，從拉 LOW 開始 15us 時 DS1821 會以 1-wWire bus 的電位來表示 1 和 0，也就是說：HIGH 電位就是 1，LOW 電位就是 0，而不是用低電位的長短來辨別，這與 8051 傳到 DS1821 時的方式不同。

(二)2 位數 LED 七段顯示器

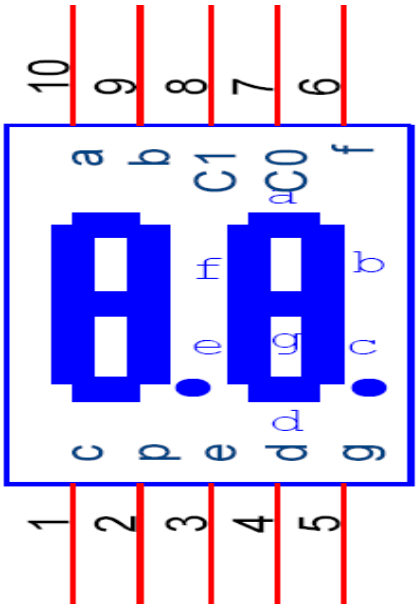


圖 2-1-2 二位數 LED 七段顯示器接腳圖

本專題使用 2 位數 LED 七段共陽極顯示器來顯示溫度值。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。

下表是以個位數為例(C0 接 5V)，要顯示 0~9 時，每個數字所對應的 LED 代號及接腳狀態：

表格內符號"●"表示對應的 LED 要亮 => 接腳接 LOW。

無符號表示對應的 LED 不亮，接腳接 HIGH。

表 2-1-1 LED 顯示接腳

		LED 代號(接腳)							
		a(3)	b(9)	c(8)	d(6)	e(7)	f(4)	g(1)	p(2)
數字	0	●	●	●	●	●	●		
	1		●	●					
	2	●	●		●	●		●	
	3	●	●	●	●			●	
	4		●	●			●	●	
	5	●		●	●		●	●	
	6	●		●	●	●	●	●	
	7	●	●	●					
	8	●	●	●	●	●	●	●	
	9	●	●	●	●		●	●	

(三)按鍵輸入

4 支腳的平頭按鍵開關是常使用易購得的零件，此開關雖然有 4 支腳，但其內部只有一組開關，如下圖所示，開關的第 1、2 腳在開關內部是接通短路的，第 3、4 腳也一樣。

平時第 1、2 腳與第 3、4 腳是開路不導通，當開關按下時，讓第 1、2 腳與第 3、4 腳接通短路，在應用上，通常是將第 1、2 腳，或是第 3、4 腳其中 1 腳接地，而另一組的其中 1 腳接到 8051 的 I/O 控制腳，用 8051 的 I/O 控制腳來讀取開關狀態。

平時按鍵開關未按下時，8051 的 I/O 控制腳無接地，會讀到高電位（High）狀態。當開關按下時，8051 的 I/O 控制腳接到地，會讀到低電位（Low）狀態。

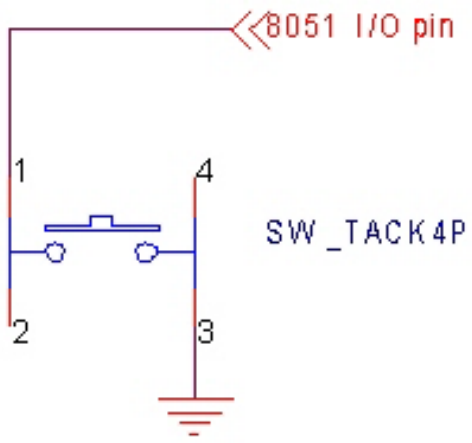


圖 2-1-3 按鍵輸入電路圖

(四)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。

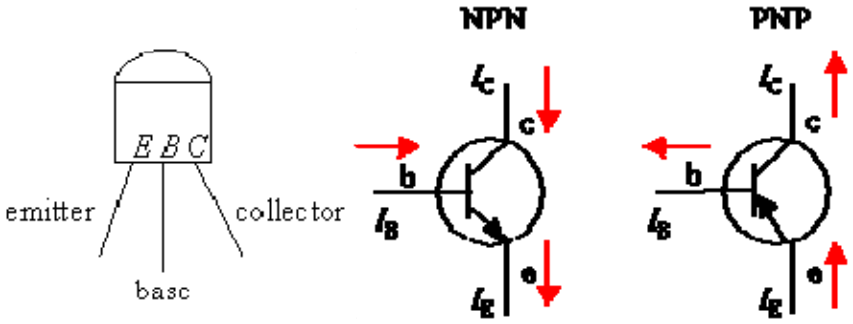


圖 2-1-4 電晶體特性示意圖

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。

NPN 電晶體當當子開關：

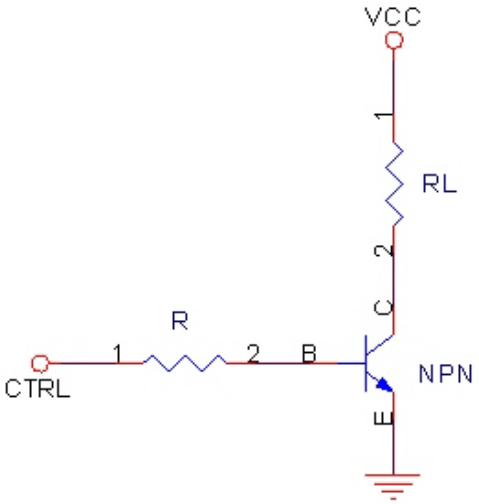


圖 2-1-5 NPN 電晶體當當子開關電路圖

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。

PNP 電晶體當當子開關：

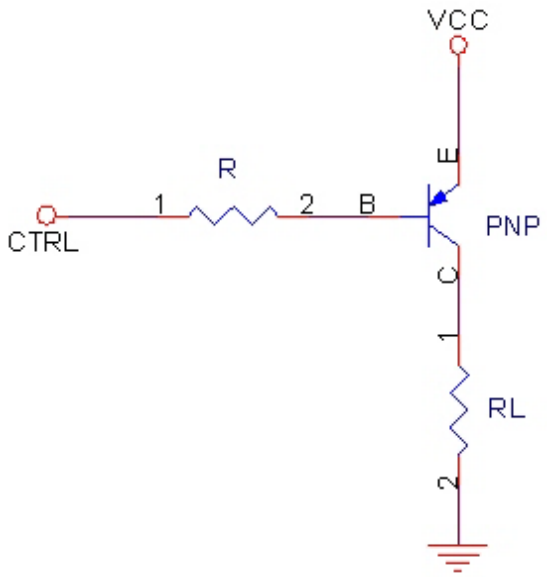


圖 2-1-6 PNP 電晶體當當子開關電路圖

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。

(五)風扇控制

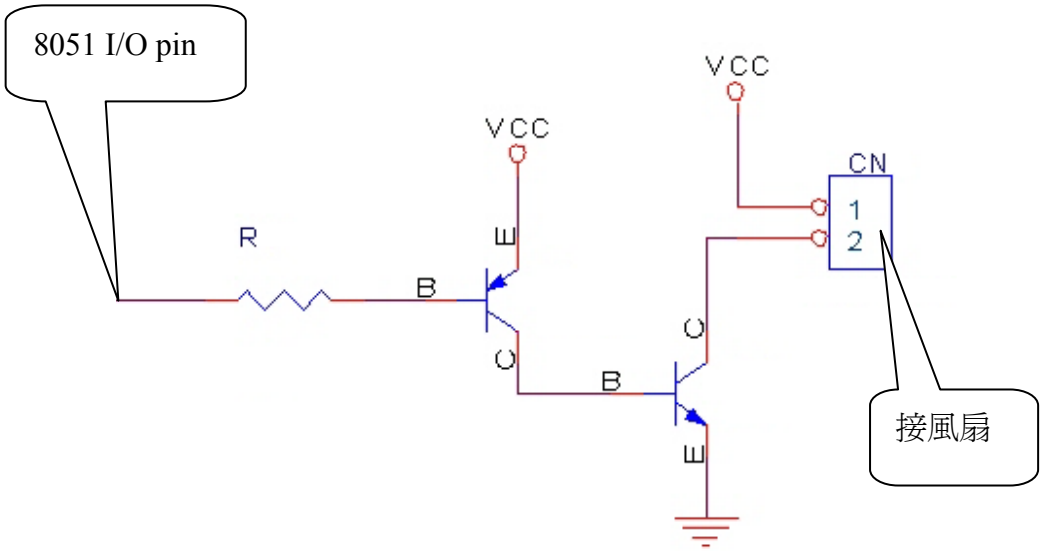


圖 2-1-7 風扇控制接腳圖

方法是以 8051 I/O pin 低電位來驅動風扇，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 高電位使風扇起動，所以設計上採用低電位來驅動風扇較合適。

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，風扇線圈不起動。

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。

(六)PWM 轉速控制

PWM是脈衝寬度調變(Pulse Width Modulation)的縮寫，很多電路都用到PWM的技術，其用途非常廣泛。如應用在高功率轉換效率的switching 電源、馬達Inverter、音響用D 極增幅器、DC-DC Converter、UPS、調光系統等，本專題的轉速控制也使用PWM來控制。

“脈衝寬度調變” 顧名思義是脈衝的寬度可以被調整變化，但脈衝週期是不變的，通常都是用”工作週期”(Duty Cycle)來表示，”工作週期”是指在一個週期內，工作所佔的時間比例，有的用Duty-On、Duty-Off來表示，以下均用Duty-On、Duty-Off此名詞來說明：

Duty-On：電路動作的時間

Duty-Off：電路不動作的時間

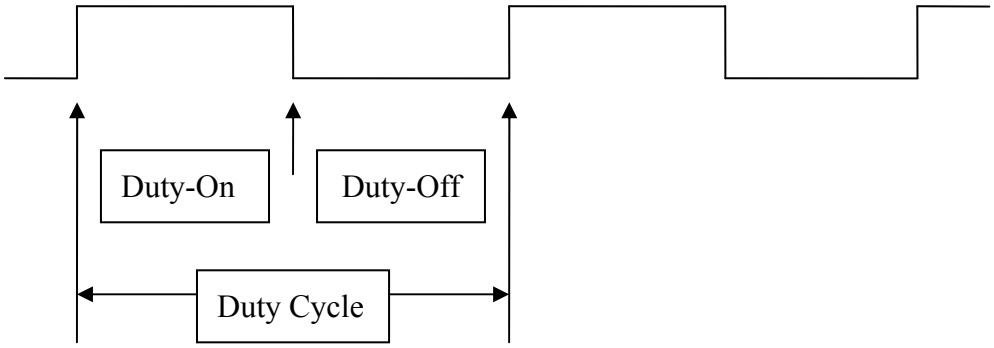


圖 2-1-8 PWM 時序圖(一)

以上時序圖表示在一個 Duty Cycle 中，Duty-On = 50%，Duty-Off = 50%

如果調整脈衝寬度為 Duty-On = 25%，Duty-Off = 75%，波形就像下圖

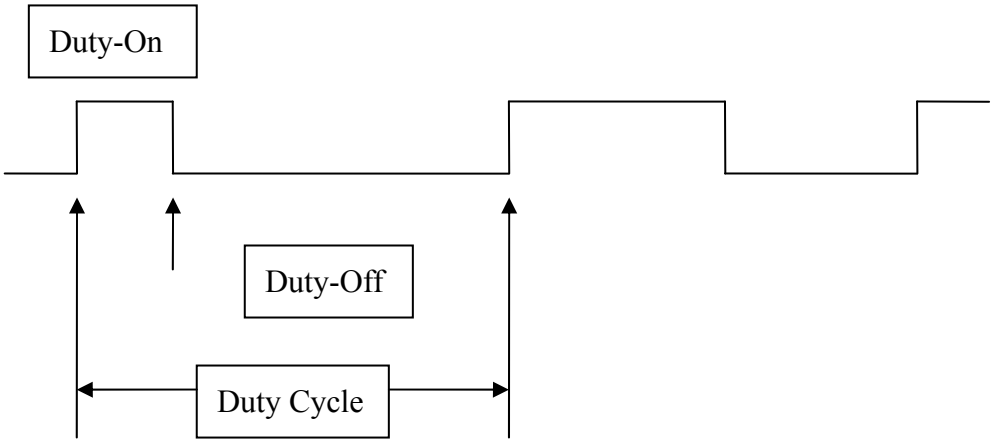


圖 2-1-9 PWM 時序圖(二)

改變脈衝寬度可以控制負載的平均電流，比方Duty-On = Duty Cycle時，表示是負載是全電流在工作，當Duty-On = 50%的Duty Cycle時，負載的平均電流剩一半，當然在 Duty-On = 25%時，負載的平均電流更少。

上面的時序敘述說明是以 Duty-On 是高電位來舉例，但是 Duty-On 不一定就是高電位，要看實際電路設計是高電位驅動，還是低電位驅動，如果是低電位驅動的電路，那時序圖就會變成如下

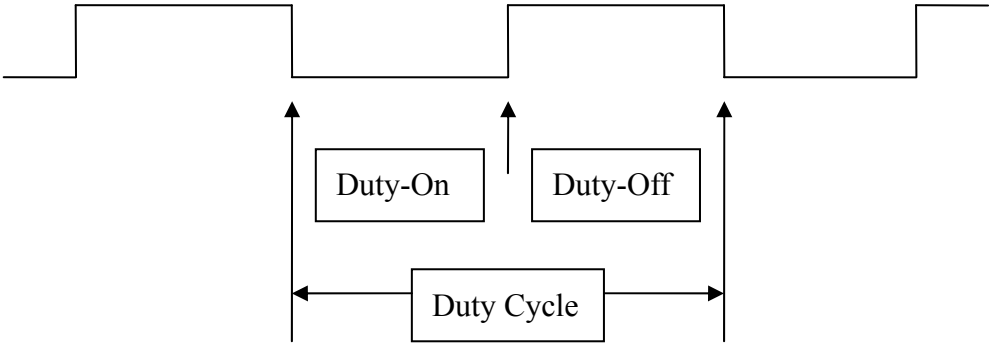


圖 2-1-10 PWM 時序圖(三)

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 高電位使風扇起動，所以設計上採用低電位來驅動風扇較合適。

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，風扇線圈不起動。

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。

當 8051 I/O pin 以 PWM 的方式送出信號時，風扇就會因 Duty-On 的比例不同，而獲得不同的平均電流，Duty-On = 100%時，平均電流最大，風扇會全速轉動，Duty-On 的比例縮小，風扇會因為平均電流變小而轉速降低，但每一種線圈轉動裝置都會有最低的起動電流，所以一般都會設定最低的 Duty-On 值，來確保起動線圈轉動裝置。

(七)8051 的接腳

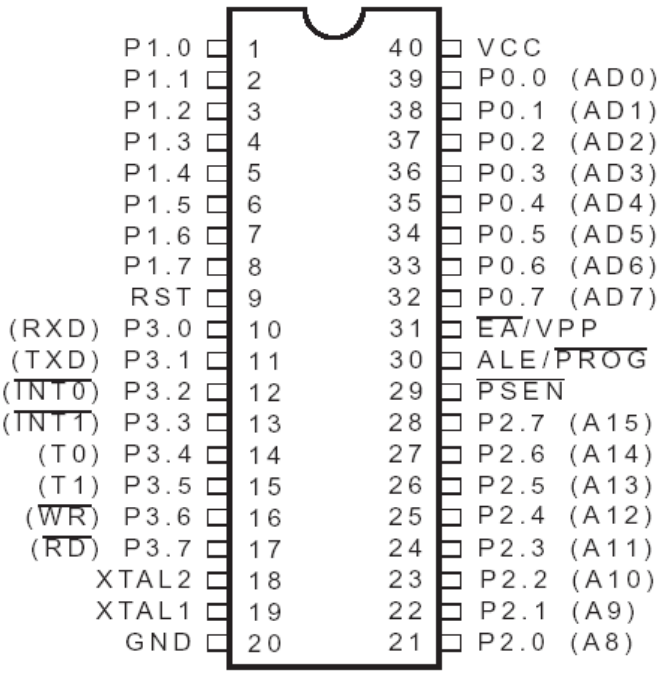


圖 2-1-11 8051 接腳圖

接腳名稱說明

VCC：輸入

供應系統工作電壓。

GND：輸入

接地。

P0.0 ~ P0.7：Port 0，雙向

8 位元具開汲極雙向 I/O 埠，各接腳可個別動態設定為輸入或輸出，當設定為輸出時，可當高阻抗數入腳。

當存取外部記憶體時，Port 0 也具備位址/資料多工切換功能。

有關 Port 0 的詳細結構說明，請詳見『**I/O 埠的結構與運作**』章節。

P1.0 ~ P1.7：Port 1，雙向

8 位元具有內部提升電阻的雙向 I/O 埠。

在 8052，P1.0 和 P1.1 具有雙重功能，P1.0 可以被設定成 Timer 2 的外部計數輸入（P1.0/T2）。而 P1.1 可以被設定成 Timer 2 的觸發（trigger）輸入（P1.1/T2EX）。

有關 Port 1 的詳細結構說明，請詳見『**I/O 埠的結構與運作**』章節。

P2.0 ~ P2.7：Port 2，雙向

8 位元具有內部提升電阻的雙向 I/O 埠。

當存取外部記憶體時，Port 2 也具備位址線輸出功能。

有關 Port 2 的詳細結構說明，請詳見『**I/O 埠的結構與運作**』章節。

P3.0 ~ P3.7：Port 3，雙向

8 位元具有內部提升電阻的雙向 I/O 埠，且每一接腳具有雙重功能，可個別被設定成其他用途如下：

P3.0：RXD = 串列埠輸入

P3.1：TXD = 串列埠輸出

P3.2：/INT0 = 外部中斷輸入 0

P3.3：/INT1 = 外部中斷輸入 1

P3.4：T0 = Timer 0 外部輸入

P3.5：T1 = Timer 1 外部輸入

P3.6：/WR = 外部記憶體寫入激發（write strobe）

P3.7：/RD =外部記憶體讀取激發（read strobe）

有關 Port 3 的詳細結構說明，請詳見『**I/O 埠的結構與運作**』章節。

RST：輸入

系統重置（Reset）輸入，在振盪器有動作的情況下，高電位的時間至少要有兩個機器週期。

ALE/PROG：輸出（ALE）/輸入（/PROG）

ALE：位址線閂鎖致能輸出脈衝，在存取外部記憶體時，用以閂鎖低位元組的位址信號。

PROG：燒錄 FLASH ROM 時的程式脈衝輸入。

在正常情況下，ALE 會發出固定為"振盪器頻率/6"的頻率，此功能可用來提供外部線路的時序或鐘控（clock）的需要。

/PSEN：輸出

外部程式記憶體的讀取激發（strobe）致能，當 CPU 執行於外部程式記憶體時，在每一個機器週期裡，/PSEN 會動作兩次。在存取外部資料記憶體時，此『動作兩次』的情況會被跳過。

/EA/Vpp：輸入

正常操作下為/EA，外部存取致能。當要令 CPU 從外部程式記憶體的位址 0000H 起開始抓取程式碼時，/EA 腳必須保持低電位輸入。

/EA 腳為高電位輸入時，CPU 是操作在內部程式記憶體。
在燒錄作業時，Vpp 是 12 伏燒錄電壓輸入。

XTAL1：輸入

輸入到反向振盪器放大器，並且輸入到內部鐘控（clock）作業電路。
參考『振盪器特性』章節。

XTAL2：輸出

從反向振盪器放大器輸出。
參考『振盪器特性』章節。

(八)太陽能板

太陽能矽晶片主要分為單晶式(Single-crystal silicon)以及多晶式(Multi-crystalline silicon)兩類，單晶式效能較佳，實驗室晶片轉換效率可達 25%以上，一般市售模組大約在 15~16%間；單晶式的實驗晶片轉換效率最大約 18%，一般市售模組則將近 14%。因此同樣發電功率之光電板，多晶式會略大於單晶式。

(九) 蓄電池

太陽能發電系統電力均儲存在蓄電池中，當有使用需求時再從蓄電池供應，這種存電裝置在夜間及陰天時為必要設備。兩種最常見運用在儲存電力的電池為鉛酸(lead-acid)及鹼性(alkaline)電池，其中鹼性電池因為相對於鉛酸電池價格偏高，且有環保處理問題，因此除非有特殊需求否則並不建議使用在太陽能發電系統。

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

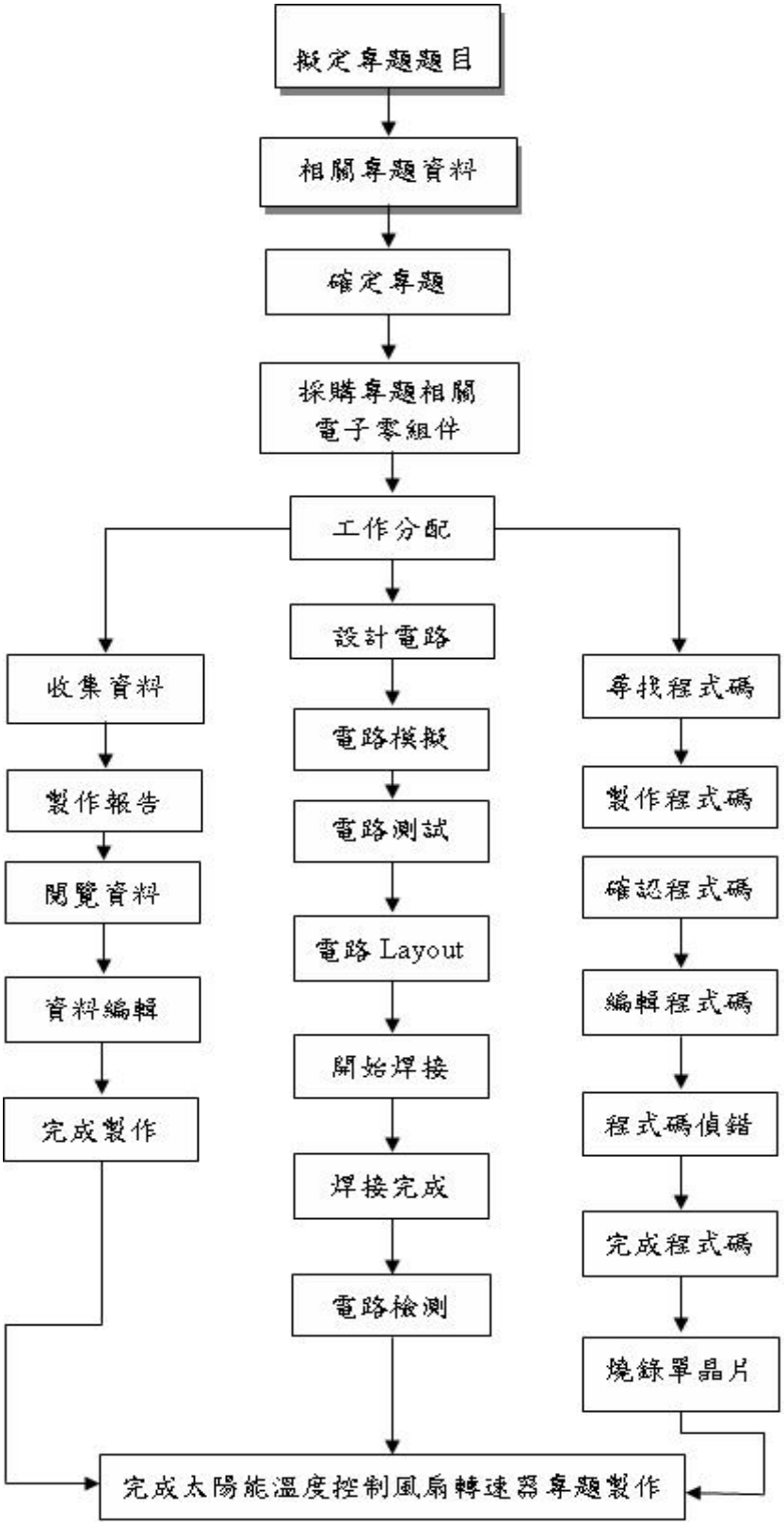


圖 3-2-1 製作方法與步驟

三、專題製作

表 3-3-1 專題製作計畫書

<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">利用單晶片 89C51 製作自動化控制電路</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">太陽能溫度控制風扇轉速器</td></tr>
<tr><td>英文名稱</td><td colspan="2">Temperature control fan speed</td></tr>
<tr><td colspan="2" rowspan="6">專題內容簡述</td><td colspan="2">本篇研究旨在透過單晶片 89C51 的學習，了解單晶</td></tr>
<tr><td colspan="2">片的功能及使用方法，且經由實際製作 PCB 電路的過程</td></tr>
<tr><td colspan="2">中去對單晶片運作有更深入的了解。；故想要藉由設計一</td></tr>
<tr><td colspan="2">單晶片電路，配合組合語言程式去達到將溫度做到自動控</td></tr>
<tr><td colspan="2">制的目的；故現行之小組專題製作的目標即是想透過單晶</td></tr>
<tr><td colspan="2">片的學習。</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>黃秉福</td><td>郭昇昌</td></tr>
<tr><td>鍾育軒</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>

(一) 硬體電路圖:太陽能溫度控制風扇轉速器

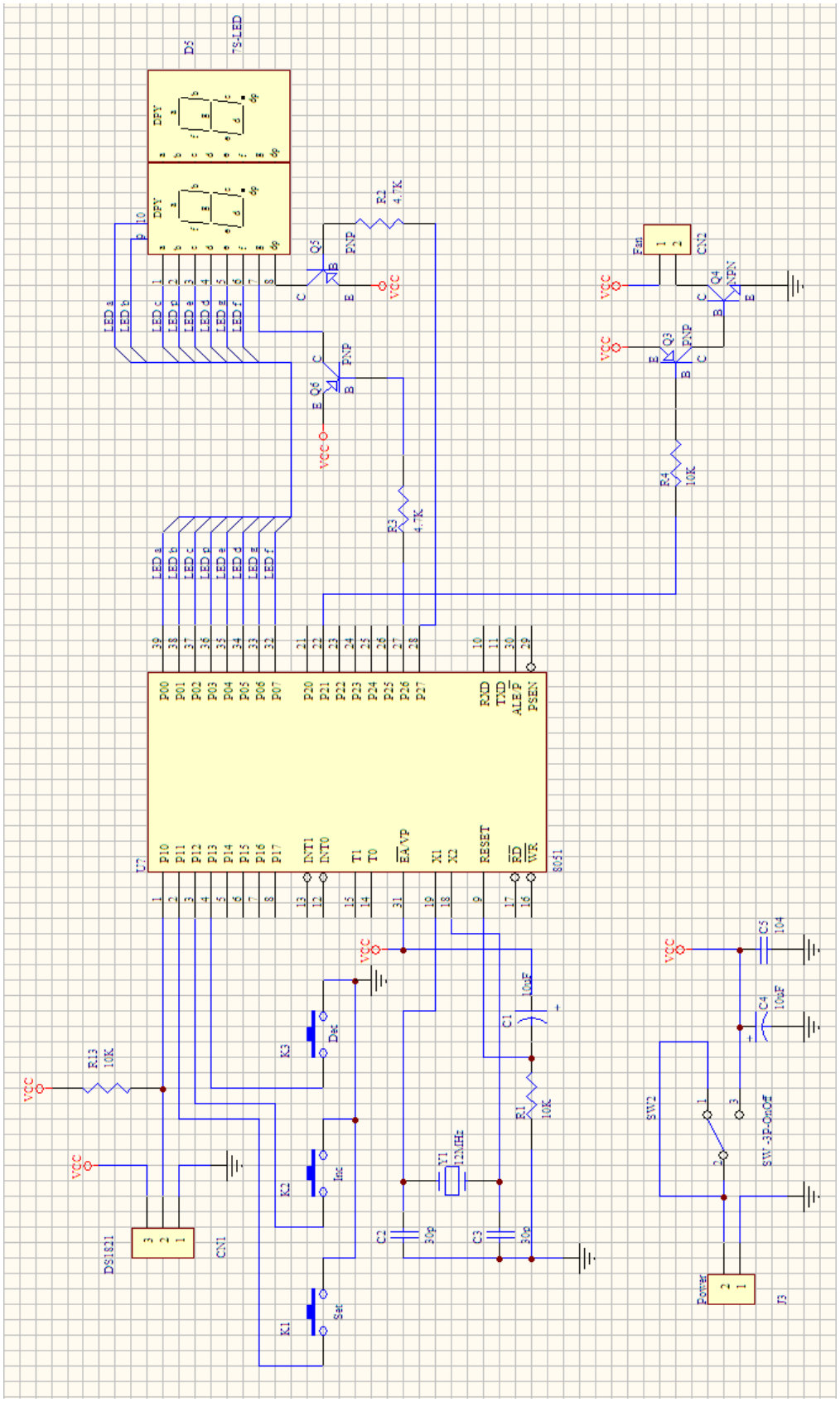


圖 3-3-1 太陽能溫度控制風扇轉速器之完整電路圖

表 3-3-2 太陽能溫度控制風扇轉速器之材料表

材料名稱	規格	單位	數量	備註
單晶片	89C51	個	1	或 89S51
IC 座	40Pin	個	1	
溫度感應晶體	DS1821	顆	1	
圓孔座	1*3	個	1	插 DS1821
電容	10uF	顆	2	電解電容
電容	1uF	顆	1	
電容	30pF	顆	2	陶瓷電容
電晶體	2N4401	顆	1	NPN
電晶體	2N2907	顆	3	PNP
晶體振盪器	12MHz	顆	1	
電阻	4.7K	顆	2	
電阻	10K	顆	3	
風扇	12V	個	1	
七段顯示器	2 位數	個	1	10 隻腳
按鍵開關	2Pin	個	3	
電源開關	3Pin	個	1	
莫士座	2Pin	個	2	
充電電池	6V	個	1	
洞洞板	900 孔	片	1	
太陽能板	4W	片	1	

(三) 小組分工的配置：

育軒負責處理報告，資料的搜尋與匯整，遇到資料不足時可能必須添購書籍或上圖書館尋找，遇到不懂的也可以找老師研究，整個專題完成之後必須整理一份上台用的簡報報告。

秉福負責的部份麵包板的電路製作測試，畫出 Layout 的電路圖，並且焊接在 PCB 板上，在繪製電路圖的過程中遭遇的問題會比較多，須與老師研究修改電路。

昇昌負責撰寫程式、購買零件，必須有效的掌控好元件，以不浪費為原則，初步構想為一次購買 2 套，一組當備用，其餘時間陪育軒去搜尋資料，與整理專題報告。

肆、製作成果

我們小組由決定題目，製作模擬電路、繪製設計電路圖，進而完成焊接製作整個電路；這整個流程，我們小組都用數位相機及相關電腦設備將之紀錄下來，經將這些資料整理過後，我們將之呈現在我們的專題報告之中，如下所示：

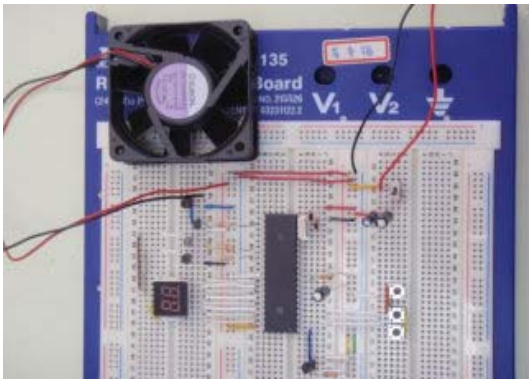


圖 4-1-1 電路測試圖(一)

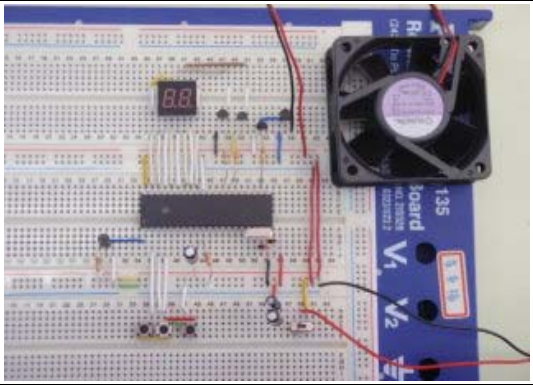


圖 4-1-2 電路測試圖(二)

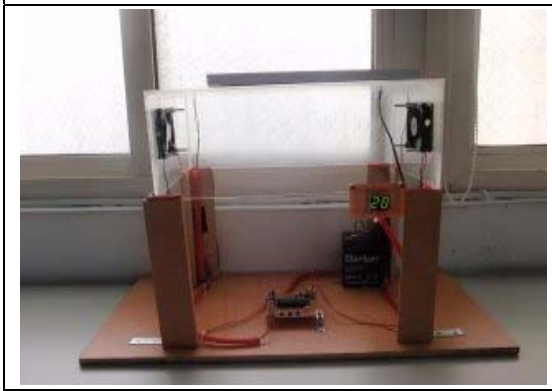


圖 4-1-3 電路版完成圖(一)

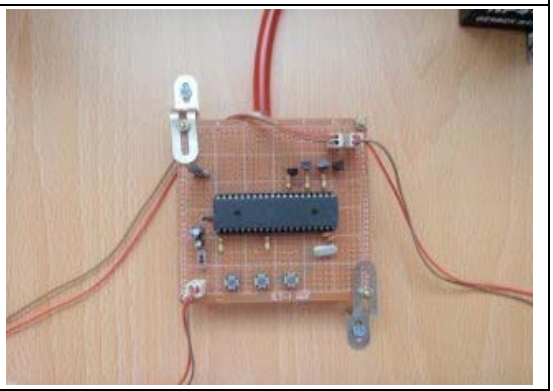


圖 4-1-4 電路版完成圖(二)

圖 4-1-5 二位數溫度顯示器圖

圖 4-1-6 太陽能板圖

至目前為止，我們的電路實作部分，面對我們所遭遇的困難及問題，都已一一克服了，緊接著，由於有牽涉到配線及木工配置的問題，所以待我們將之完成後，會再以實體的方式呈現出來。

伍、結論與建議

本單元我們將針對我們小組對專題製作的整個學習過程，做一最後完整的彙總及記錄，以期作為未來學弟妹們日後學習之參考。

一、結論

本研究達成了用溫度控制電風扇的目的，當溫度到達一定值電風扇將會動，溫度持續上升的話電源也會持續啟動電風扇，同理當溫度降到一定值時電風扇也會，而且當溫度下降到達一定值時電源將會斷路，而我們的電源來自太陽能板的電，所以可達到節約能源的目的。 除了上述功能之外，電路中又可輸入溫度範圍，可以依個人喜好輸入溫度，來決定停止轉動的最低溫度。

透過此次專題製作學習的方式能幫助我們提升對課程的學習、獲得知識的建構及整合，且亦可以幫助我們提升其創造思考的能力，培養我們學習者具備問題解決、研究、反省、團體合作及應用資訊科技等多項能力；小組同學認為專題製作學習為一主動探究的學習，學習中強調學習者必須負起主動探究學習的責任，也鼓勵小組成員分工和合作學習的精神。雖然會遇到不同的困難及問題，但看到自己的成品時，會很有成就感。

整體而言，我們小組同學認為專題製作學習是一有價值的學習方式，因其確實可以增進自己資訊科技的能力及其技能。

二、建議

我們在進行專題製作學習的過程後，提出以下幾點建議：

(一) 學習前清楚的說明：請老師在進行專題製作學習前，能對學生清楚的說明整個專題進行的方式，包括專題報告的格式、課程進度的安排、需要的準備工具以及評量方式等，如都能在事前做好詳細的說明、規範，如此則能避免學生因疑惑而做錯方向。

(二) 在學習過程中給予回饋：同學建議，在專題製作學習研究過程中，老師能否可以在學習的進行過程，給予立即性的回饋，讓學生可以及早發現其缺失，盡早進行改善。

(三) 增長專題製作學習的時間：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，連帶影響了期末完整報告的製作，所以希望老師能增長同學學習時間，讓成果報告的製作能更加完整，避免同學因時間緊迫而草率完成其作品。

參考文獻

1.蔡朝洋，2007，單晶片電腦 8051/8951 原理與應用，台北縣：全華圖書公司。

2.郭庭吉，2008，8051 單晶片微電腦專題製作，台北縣：台科大圖書公司。

3.鄧明發，陳茂璋，2000，微電腦專題製作應用電路，台北市：知行文化公司。

4.朱永昌，2007，8051 單晶片微電腦原理與專題製作(上)，台北縣：台科大圖書公司。

5.林明德，WonDerSun，2008，專題製作-電子電路篇，台北縣：台科大圖書公司。

6.柯南，2001，全能電路設計之 Protel Schematic 99 SE，台北縣：台科大圖書公司。

7.鍾明政，1999，單晶片 8051 原理與實作，台中市：長高企業公司。

附錄 溫度控制風扇轉速器之程式碼

【Main.c】

```
#include <reg51.h>
#include "tick.h"
#include "keypad.h"
#include "led2x1c.h"
#include "ds1821.h"
#include "pwm.h"

#define HIGH_TEMP_VAL   30    //高溫控制預設值
#define BLINK_TIME   MS_300  //閃爍間隔時間
#define KEY_SET  KP_K1    //溫度設定鍵
#define KEY_INC  KP_K2    //遞增鍵
#define KEY_DEC KP_K3    //遞減鍵

typedef enum enumOperatingMode
{
      OP_TempControl,  //溫度控制作業
      OP_FanTriggerDelay,    //風扇起動觸發延遲
      OP_HighTempAdj, //高溫控制值調整作業
      OP_End
}OperatingModeIndex;

void InitDS1821(void);
void StartConvertTemperature(void);
unsigned char ReadTemperature(void);
void ReadTempProc(void);
void ControlProc(void);

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 int tReadTemp;       //溫度轉換程序與讀取溫度值的間隔時間
unsigned int tBlink;//閃爍時間基準
```

```
unsigned int tTriggerDelay;   //風扇起動觸發延遲時間
unsigned char CurrentTemp;  //目前溫度值
unsigned char TempRec;//溫度記錄
unsigned char TempControlHigh;  //高溫控制值
bit fReadTemp;      //溫度轉換讀取辨示旗號
bit fSet;  //溫度設定旗號
bit fBlink;       //閃爍旗號
bit fFan2Run; //第二組風扇起動旗號
bit fFanTrigger;      //風扇起動觸發旗號

main()
{
      InitTick();       //初始系統鐘控計時
      InitKeypad();  //初始按鍵掃瞄
      InitLed2x1c();//初始七段顯示
      InitDS1821(); //初始 DS1821
      InitPWM();    //初始 PWM 控制

      fReadTemp = 0;     //初始溫度轉換讀取辨示旗號
      fSet = 0;  //初始設定模式旗號
      TempRec = 0; //初始溫度記錄
      fFan2Run = 0;//初始第二組風扇起動旗號
      fFanTrigger = 0;     //初始風扇起動觸發旗號
      TempControlHigh = HIGH_TEMP_VAL;    //載入高溫控制預設值
      tReadTemp = GetSystemTick();    //設定溫度轉換與讀取溫度值的間隔時間基
準

      KeyRec = KP_None;     //清除鍵盤碼記錄
      OpMode = OP_TempControl;//初始作業程序

      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
```

```
        ReadTempProc();  //呼叫溫度控制程序
        if(!fSet && (CurrentTemp != TempRec))    //檢查溫度是否有變化
        {
            ControlProc();//風扇控制處理
            TempRec = CurrentTemp;    //記錄溫度
        }

        switch (OpMode)  //依作業碼處理相對作業
        {
            case OP_TempControl:  //溫度控制作業程序
                if(KeyIn == KEY_SET) //檢查是否是設定鍵
                {
                    DspDec(TempControlHigh); //顯示高溫控制值
                    fSet = 1; //設定調整旗號
                    fBlink = 0;    //清除閃爍旗號
                    tBlink = GetSystemTick();   //設定閃爍時間基準
                    OpMode = OP_HighTempAdj;    //下一步 : 溫度調整作
業程序
                }
                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((CurrentTemp - TempControlHigh)/2);  //起動
溫差對應的轉速
                    OpMode = OP_TempControl;//回到溫度控制作業程序
                    OpMode = OpRet; //返回作業程序
                }
                break;

            case OP_HighTempAdj: //溫度調整作業程序
                if(KeyIn == KEY_SET) //檢查是否是設定鍵
                {
                    fSet = 0; //清除調整旗號
                    Display7LEDSw(DISPLAY7LED_ON);    //打開 7 段顯示
                    OpMode = OP_TempControl;//下一步:溫度控制作業
```

```
            }
            else if(KeyIn == KEY_INC)  //檢查是否是遞增鍵
            {
                if(TempControlHigh != 99)   //檢查高溫控制值是否不等於
最大值
                {
                    TempControlHigh++;    //高溫控制值遞增 1
                    DspDec(TempControlHigh);  //顯示高溫控制值
                    ControlProc(); //風扇控制處理
                }
            }
            else if(KeyIn == KEY_DEC) //檢查是否是遞減鍵
            {
                if(TempControlHigh != 20)   //檢查高溫控制值是否不等於
最小值
                {
                    TempControlHigh--;      //高溫控制值遞減 1
                    DspDec(TempControlHigh);  //顯示高溫控制值
                    ControlProc(); //風扇控制處理
                }
            }

            if(fBlink == 0)//檢查閃爍旗號是否為 0
            {
                if((GetSystemTick() - tBlink) >= BLINK_TIME) //檢查閃爍
時間是否到時
                {
                    Display7LEDSw(DISPLAY7LED_OFF);    //關閉 7 段
顯示
                    tBlink = GetSystemTick();    //設定閃爍時間基準
                    fBlink = 1;     //設定閃爍旗號
                }
            }
            else
            {
                if((GetSystemTick() - tBlink) >= BLINK_TIME) //檢查閃爍
時間是否到時
                {
```

```
                    Display7LEDSw(DISPLAY7LED_ON);      //打開 7 段
顯示
                    tBlink = GetSystemTick();    //設定閃爍時間基準
                    fBlink = 0;      //清除閃爍旗號
                }
            }
            break;
        }

        ScanLed2x1c();      //掃瞄七段顯示
    }
}

/*-----------------------------------------------------------------
  溫度控制程序
-----------------------------------------------------------------*/
void ControlProc(void)
{
    //風扇控制
    if(CurrentTemp >= TempControlHigh) //檢查是否到達高溫控制值
    {
        if(fFanTrigger == 0)      //檢查風扇起動觸發旗號是否無設定
        {    //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
            fFanTrigger = 1;    //設定風扇起動觸發旗號
            tTriggerDelay = GetSystemTick() + MS_500;      //設有風扇起動觸發
延遲時間
            OpRet = OpMode; //儲存返回作業碼
            OpMode = OP_FanTriggerDelay; //下一步:風扇起動觸發延遲
        }
        else //風扇在轉動中變動轉速時,直接變動工作週期即可
            DutyByScale(CurrentTemp - TempControlHigh); //起動溫差對應的轉
速
    }
    else
    {
        fFanTrigger = 0;    //清除風扇起動觸發旗號
        DisableDuty();//關閉高溫控制
```

```
    }
}

/*-------------------------------------------------------------
  讀取溫度程序
-------------------------------------------------------------*/
void ReadTempProc(void)
{
    unsigned char temp;

    if(!fReadTemp)     //檢查溫度轉換讀取辨示旗號是否沒設定
    {
        if((GetSystemTick() - tReadTemp) >= MS_200)  //檢查溫度轉換與讀取溫
度值的間隔時間是否到時
        {
            StartConvertTemperature();  //開始溫度轉換程序
            tReadTemp = GetSystemTick();   //設定溫度轉換與讀取溫度值的間
隔時間基準
            fReadTemp = 1;     //設定始溫度轉換讀取辨示旗號
        }
    }
    else
    {    //溫度轉換程序開始後,約 300ms 再讀取溫度值
        if((GetSystemTick() - tReadTemp) >= MS_300)  //檢查溫度轉換與讀取溫
度值的間隔時間是否到時
        {
            temp = ReadTemperature();  //讀取溫度值
            CurrentTemp = temp;    //儲存目前溫度值
            if(!fSet)  //檢查是否不是在溫度設定模式
                DspDec(CurrentTemp); //顯示溫度值
            tReadTemp = GetSystemTick();   //設定溫度轉換與讀取溫度值的間
隔時間基準
            fReadTemp = 0;     //清除始溫度轉換讀取辨示旗號
        }
    }
}
```

【DS1821.c】

```
#include <reg51.h>
#include <INTRINS.H>
#include "ds1821.h"

//DS1821  資料讀取/寫入命令集
#define CMD_StartConvertT 0xee
#define CMD_StopConvertT 0x22
#define CMD_ReadStatus          0xac
#define CMD_WriteStatus         0x0c
#define CMD_ReadTemperature     0xaa
#define CMD_ReadTemperHigh      0xa1
#define CMD_ReadTemperLow       0xa2
#define CMD_WriteTemperHigh     0x01
#define CMD_WriteTemperLow      0x02

sbit DS1821_DQ    =     P1^0;      //1-wire  通訊接腳

void ResetPresence(void);
void DelayUs(unsigned char);
unsigned char SendReceiveByte(unsigned char);
bit SendReceiveBit(bit);

/*=============================================
   初始  DS1821
=============================================*/
void InitDS1821(void)
{
     // 在此加初始設定
     DS1821_DQ = 1;   //DQ  腳輸出高電位
}

/*=============================================
   開始溫度/數位轉換
=============================================*/
void StartConvertTemperature(void)
```

```
{
    ResetPresence();    //起始 1-wire bus
    SendReceiveByte(CMD_StartConvertT);    //送命令給 DS1821
}

/*============================================
  讀取溫度值
  輸出：溫度值
============================================*/
unsigned char ReadTemperature(void)
{
    ResetPresence();    //起始 1-wire bus
    SendReceiveByte(CMD_ReadTemperature);//送命令給 DS1821
    return(SendReceiveByte(0xff));    //讀取溫度值
}

/*============================================
  傳送/接收 8 位元
  輸入：要傳送的位元組
  輸出：接收到的位元組
============================================*/
unsigned char SendReceiveByte(unsigned char dat)
{
    unsigned char cnt;
    bit rbit;

    for(cnt = 0; cnt < 8; cnt++)    //8 位元傳送接收迴路
    {
        rbit = SendReceiveBit(dat & 1);    //LSB(bit0) 先傳
        dat >>= 1;      //右移 1 位元

        if(rbit)
            dat |= 0x80;    //將收到的位元值放到 MSB(bit7)
    }

    return(dat);    //傳回資料
}
```

```
/*==========================================
  傳送/接收 1 位元
  輸入：傳送的位元
  輸出：接收的位元
==========================================*/
bit SendReceiveBit(bit bdat)
{
    unsigned char dly;

    DS1821_DQ = 0;   //DQ 腳輸出低電位
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    DS1821_DQ = bdat;     //傳送 1 位元資料
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    bdat = DS1821_DQ;     //接收 1 位元資料

    for(dly = 0; dly < 12; dly++); //延遲迴路

    DS1821_DQ = 1;   //DQ 腳輸出高電位

    return(bdat);   //傳回接收的位元
}

/*==========================================
  起始 1-wire bus 傳輸功能
  保持 DQ 低電位 480us < RST_low < 960us
==========================================*/
void ResetPresence(void)
{
    // 送出 Reset 脈衝
    DS1821_DQ = 0;   //DQ 腳輸出低電位
```

```
    // 延遲 480-960 us
    DelayUs(70);
    DS1821_DQ = 1;   //DQ 腳輸出高電位
    // 等待 DS1821 回應 PRESENCE 脈衝
    // 延遲 15-60us
    DelayUs(3);
    while(DS1821_DQ);     //等待 DS1821 回應 Low
    while(!DS1821_DQ);    //等待 DS1821 回應 High
}

/*=====================================[]
   input : us delay
=====================================*/
void DelayUs(unsigned char us)
{
    for(;us != 0; us--)   //延遲迴路
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}
```

【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_LEVEL_TOTAL 3     //總段數
#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 =    P2^1;      //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
    {
        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】

```
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) ^ KP_MASK);   //返回按鍵碼
}
```

【Keypad.h】

```
#define KeyPort     P1
typedef enum KeyButtonEnum {
      KP_None= 0x0,
      KP_K1          = 0x2,
      KP_K2          = 0x4,
      KP_K3          = 0x8
} KeyButton;

#define    KP_MASK          (KP_K1 | KP_K2 | KP_K3)

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

【Led2x1c.c】

```
#include <reg51.h>
#include "led2x1c.h"

#define DigiCommPort P0          //顯示資料埠
#define DIGI7LED_ON      0       //打開顯示
#define DIGI7LED_OFF     1       //關閉顯示

sbit  Digi0CtrlPin =P2^6;      //個位數顯示控制腳
sbit  Digi1CtrlPin =P2^7;      //十位數顯示控制腳

bit fLedOut, fDisplay7LED;
unsigned char LedDecode0;   //個位數七段顯示解碼儲存位置
unsigned char LedDecode1;   //十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
    ~0x40,          //a = .x..-....
    ~0x40,          //b = .x..-....
    ~0x40,          //c = .x..-....
    ~0x40,          //d = .x..-....
    ~0x40,          //e = .x..-....
    ~0x40,          //f = .x..-....
    ~0x00           //不顯示
};
```

```
/*---------------------------------------------------
  初始七段顯示
---------------------------------------------------*/
void InitLed2x1c(void)
{
    DigiCommPort = 0xff;  //無顯示輸出
    Digi0CtrlPin = DIGI7LED_OFF;//關閉個位數七段顯示
    Digi1CtrlPin = DIGI7LED_OFF;//關閉十位數七段顯示
    fLedOut = 0;  //起始個位數、十位數顯示切換旗標
    fDisplay7LED = DISPLAY7LED_ON; //起動 7 段顯示
}

/*---------------------------------------------------
  位數顯示開關控制
  sw = 0 : 關閉位數顯示
  sw = 1 : 打開位數顯示
---------------------------------------------------*/
void Display7LEDSw(bit sw)
{
    fDisplay7LED = sw;     //載入設定旗號
}

/*---------------------------------------------------
  顯示數字副程式
---------------------------------------------------*/
void DspDec(unsigned char val)
{
    LedDecode0 = LedDecodeTable[(val % 10)];      //取得個位數七段顯示解碼
    LedDecode1 = LedDecodeTable[(val / 10)]; //取得十位數七段顯示解碼
}

/*---------------------------------------------------
  掃瞄顯示副程式
---------------------------------------------------*/
void ScanLed2x1c(void)
{
    if(!fDisplay7LED) //檢查是否關閉全部的顯示
```

```
    {
        Digi0CtrlPin = DIGI7LED_OFF;  //關閉個位數七段顯示
        Digi1CtrlPin = DIGI7LED_OFF;  //關閉十位數七段顯示
    }
    else if(fLedOut)     //檢查是否顯示十位數
    {    //顯示十位數
        Digi0CtrlPin = DIGI7LED_OFF;//關閉個位數七段顯示
        DigiCommPort = LedDecode1;    //輸出十位數七段顯示解碼值
        Digi1CtrlPin = DIGI7LED_ON;   //打開個位數七段顯示
        fLedOut = !fLedOut;     //反向個位數、十位數顯示切換旗標
    }
    else
    {    //顯示個位數
        Digi1CtrlPin = DIGI7LED_OFF;//關閉十位數七段顯示
        DigiCommPort = LedDecode0;    //輸出個位數七段顯示解碼值
        Digi0CtrlPin = DIGI7LED_ON;   //打開十位數七段顯示
        fLedOut = !fLedOut;     //反向個位數、十位數顯示切換旗標
    }
}
```

【Led2x1c.h】

```
#define DISPLAY7LED_ON 1     /* 打開七段顯示 */
#define DISPLAY7LED_OFF      0    /* 關閉七段顯示 */

void InitLed2x1c(void);
void DspDec(unsigned char);
void ScanLed2x1c(void);
void Display7LEDSw(bit);
```

【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;
      unsigned int RetTick;

/*============================================================
      讀取系統計時時間
============================================================*/
unsigned int GetSystemTick(void)
{
#pragma asm
      /* 抓取系統鐘控值 */
      MOV         A,SystemTick+01H
      MOV         RetTick+01H,A
      MOV         A,SystemTick
      MOV         RetTick,A
      /* 比較系統鐘控值有無變化 */
      MOV         A,SystemTick+01H
      CJNE      A,RetTick+01H,DiffByInterrupt
      MOV         A,SystemTick
      CJNE      A,RetTick,DiffByInterrupt
      /* 系統鐘控值有無變化 */
      JMP         GtRet
```

```
    /* 系統鐘控值有因中斷而變化，重新抓取系統鐘控值 */
DiffByInterrupt:
    MOV     A,SystemTick+01H
    MOV     RetTick+01H,A
    MOV     A,SystemTick
    MOV     RetTick,A
GtRet:
#pragma endasm
    return(RetTick);    /* 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);
```