產學合作案結案報告書

華電子 104 產學字 003 號

紅外線追蹤系統
Infrared Tracking System

甲方：崧皓企業有限公司

乙方：中華學校財團法人中華科技大學電子系

計劃主持人：毛大喜

目錄

表次

圖次

摘　要

有感於探測器對人類生命起源以及人類未來發展的重要地位，本文研究紅外線遙控原理，探討如何設計一輛方便使用於各個領域的遙控自走車。

本論文是以紅外線的發射與接收，採用馬達控制模型車進行遙控和追蹤。整個系統分成三部分製作:第一部分為硬體電路設計：硬體驅動與電路、紅外線感應器、馬達與車子架構。第二部分則是軟體設計：設計流程圖、8051 程式設計。第三部分則是軟硬體結合測試：驗證紅外線線追蹤系統功能，將軟體與硬體結合測試及各種操作。

關鍵字：紅外線、遙控、追蹤。

Abstract

Responding to probe the origins of life and the importance of human development in the future of mankind, we study the infrared remote control principle, to explore how to design an easy to use remote control in all areas of self-propelled vehicles.

This paper is based on the infrared transmitter and receiver, the use of motor control model car remote control and tracking. Making the whole system is divided into three parts: The first part is a hard circuit design: the hardware drivers and circuits, infrared sensors, and motor vehicle architecture. The second part is the software design: Design flowchart 8051 programming. The third part is a combination of hardware and software testing: validation function infrared line tracking system will combine software and hardware testing and various operations.

Keywords: infrared, remote control, tracking.

一、研究緣起

在工程領域中，探測器始終扮演著相當重要的角色。探測器能在不適於人類活動的地方工作，如：地形崎嶇的峭壁或坑谷、陰暗且無氧氣存在的海底、地心引力遠小於地球甚或毫無重力的外太空等環境。隨著科技愈益發達，探測器所具備的功能也愈見強大。有感於探測器對人類生命起源以及人類未來發展的重要地位，利用產學案的機會，研究紅外線遙控原理，探討如何設計一輛方便使用於各個領域的遙控自走車。

如何設計一具性能優越的探測器，其控制原理即為最基本的環節所在。因此設計一輛紅外線遙控自走車以作為探討、設計、製作與整合等為此產學計畫的動機。

二、研究方法及過程

本文主要是在實現 8051 應用的運作方式，以及如何控制車子自動行走和手動操作。本文分別研究了直流馬達與8051單晶片IC、紅外線發射器與紅外線接收模組等軟硬體架構，並用組合語言來實作8051單晶片設計，同時配合8051單晶片驅動直流馬達的系統，來製作紅外線追蹤遙控車[1]。

採用8051方便性並配合馬達特性，及紅外線傳輸和接收模組，來完成紅外線追蹤遙控車，同時完成夠切換手動和自動追蹤的功能。本文是以遙控方式自動搜尋紅外線發射器，以追蹤車子且經過切換之後可以改為手動控制車子。自動追蹤判定監測範圍距離是否有信號，有信號則往信號方向前進，無信號則停止。手動則以人工搜尋範圍操作車子前進方向。

設計紅外線追蹤遙控就是可以在某些狹小的地方讓機器自由進出，在發射器接近的地方可以啟動自動開啟某些東西，例如回家之後家裡自動監測到發射器而自動啟動電燈開關，也可以把發射器放在某些地方，當被移動之後接收器消失信號提醒人們注意[2,3]。

三、重要發現

系統硬體簡介：

1. 電路是由 89C51 單晶片為控制核心，具有低功耗；系統則由 89C51 內部中斷提供。
2. 紅外線接收電路及紅外線發射器之移動使用者。
3. 直流馬達電路則由 L293 IC 提供驅動電路

4. PT2248 是一個 COMS 紅外線遙控發射器，它有 18 種功能和 75 個指令可以使用，單點或連續按鍵皆可，可以運用於電視遙控、鐵捲門遙控等等。

5.PT2248 特性：低消耗功率、需較少外接元件、寬電源供應 2.2~5V、RC 共振當作震盪步率來源[4-6]

四、主要建議事項

現在的科技進步很快，產品數位化自動化已經是一個趨勢，電子產品就是讓人們生活更加方便、便利，以節省時間，現在自動化的東西在這社會上已經非常普遍的存在。

本文設計紅外線追蹤系統時，是參考了網路上別人研究的一些資料做為初步的構想和設計，因為紅外線可以控制許多東西，經過分析與探討，最後決定用紅外線控制車子來做為教學相長的計畫。

本論文設計並實作出紅外線追蹤系統。本文採用以二元化、去除雜訊、連結區塊、驗證、追蹤的順序來進行偵測，本系統經實驗得到的良好的操控性，而加入了 tracking 的機制之後更可以達到自動化功能。但是在某些情況，如障礙物造成紅外線的遮蔽、過多的雜訊導致效果未臻完美，希望將來系統可以導入藍芽無線機制，使得追蹤能力更加提升。

第一章、緒論

1.1研究動機

在工程領域中，探測器始終扮演著相當重要的角色。探測器能在不適於人類活動的地方工作，如：地形崎嶇的峭壁或坑谷、陰暗且無氧氣存在的海底、地心引力遠小於地球甚或毫無重力的外太空等環境。隨著科技愈益發達，探測器所具備的功能也愈見強大。如何設計一具性能優越的探測器，其控制原理即為最基本的環節所在。因此本計畫設計一輛紅外線遙控自走車以作為探討、設計、製作與整合等工作。

1969年7月20日，美國太空人阿姆斯壯登陸月球，在月球表面踏出了人類的一大步。同時，探測器-月球探勘者號，證實了月球極區有冰存在，又發現月球上具有常與水分子並存的氫氣，在在顯示月球上有水的存在，這使的人類移民月球甚或其他行星的可能性大為提高。之所以能夠獲得這些寶貴的資訊，沒有配備電腦的探測器，貢獻實在不可忽略。時間推進到1997年，搭載更新技術的探測器惠更斯號(Huygens)發射升空，並於2004年降落在土星的衛星泰坦（Titan）上，讓人們更進一步了解地球生命演化的奧秘。

當今超過40位的頂尖科學家共同預測50年後的未來世界，其中，美國航太總署的太空科學專家麥凱更大膽預測，藉由精密的探測器，未來50年可能在火星古老的永凍土發現冰封的外星生命。

有感於探測器對人類生命起源以及人類未來發展的重要地位，遂研究最基本的紅外線遙控原理，探討如何設計一輛方便使用於各個領域的遙控自走車。

1.2 研究方法概述與研究目的

在機器人的世界中，包含了機械、自動化、光學、電子、通訊、資訊軟體、系統安全還有創意的特性等相關學問。伴隨著人類文明的發展，許多工作都是由電腦和機器來控制，人力漸漸地被取代。因此利用現代化科技改善以保障每個人的生活需求，這都是科技一日千里的原因。

在本研究計畫當中，共分成兩大部分，一個為控制電路的部分，另一個為紅外線接收電路部分，這兩個部分的結合構成一個基本的架構，因此如何掌握控制這兩大電路是這計畫的一大關鍵。有了基本架構的認識後，便可以對其加以擴充以及增加其完備性。

第二章、相關文獻探討

本計畫係以實作為主，相關參考文獻皆以技術資料如8051微控器IC，外線發射及接收電路及馬達控制電路。主要參考文獻如下:

1. 黃宏彥、余文俊、楊國輝編著，感測器原理與應用電路實習，高立圖書有限公司，1999。

2.蔡朝陽，電工實習（四），全華科技圖書股份有限公司，中華民國八十三年五月。

3.鄧錦城編著，8051單晶片實作寶典，益眾資訊有限公司，2000。

4. M. Nilsson, D. Binnie, and A. Armitage, “Pedestrian Detection using Low-resolution Thermal Imager Versus Visual Imager,” *PREP 2004*, pp.156-157, UK, 2004

上述文獻資料對本計畫執行甚有助益。

第三章、系統架構

本系統由下述部分所組成:

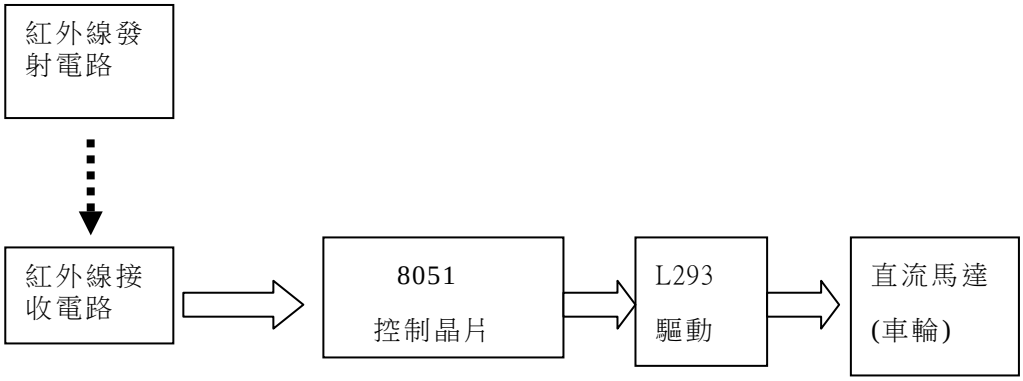

圖1 紅外線遙控追蹤系統

系統硬體簡介：

1. 電路是由 89C51 單晶片為控制核心，具有低功耗；系統則由 89C51 內部中斷提供。(圖二單晶片基本電路)
2. 紅外線接收電路及紅外線發射器之移動使用者。
3. 直流馬達電路則由 L293 IC 提供驅動電路(圖三 馬達驅動電路)

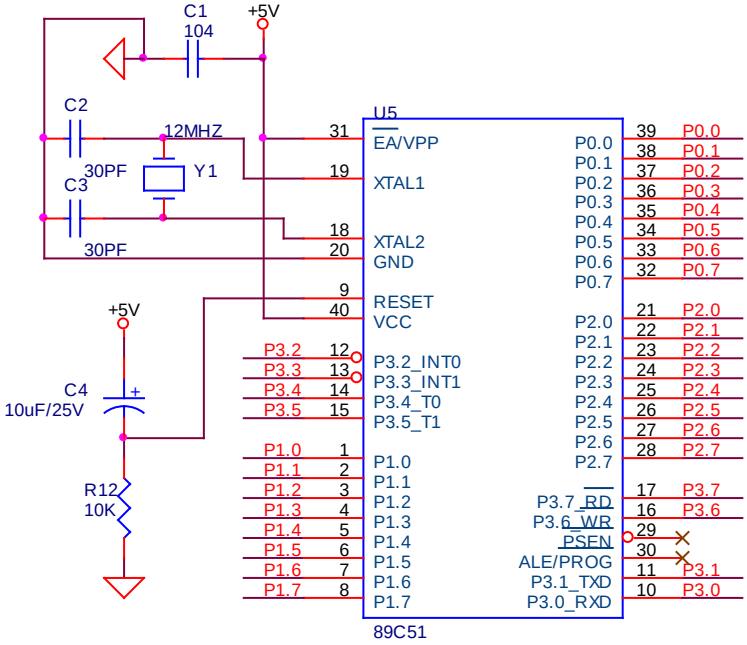

圖 2 單晶片基本電路

使用馬達控制IC L293D

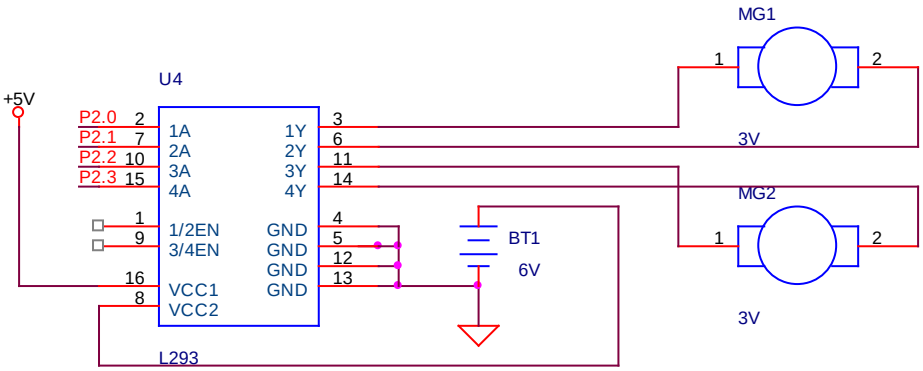

圖 3 馬達驅動電路

1A 接到 8051 之 P2.0 腳

2A 接到 8051 之 P2.1 腳

3A 接到 8051 之 P2.2 腳

4A 接到 8051 之 P2.3 腳

控制車子前進、右轉、左轉、停止的方法如下表所示

執行< 前進 >對策，即左前進,右前進			
左輪前進	SETB P2.0	右輪前進	SETB P2.2
	CLR P2.1		CLR P2.3

執行< 右轉 >對策，即左前進,右停止			
左輪前進	CLR P2.0	右輪停止	SETB P2.2
	SETB P2.1		CLR P2.3

執行< 左轉 >對策，即左停止,右前進			
左輪停止	SETB P2.0	右輪前進	CLR P2.2
	CLR P2.1		SETB P2.3

執行< 停止 >對策，即左停止,右停止			
左輪停止	CLR P2.0	右輪停止	CLR P2.2
	CLR P2.1		CLR P2.3

紅外線發射系統 使用的是 PT2248

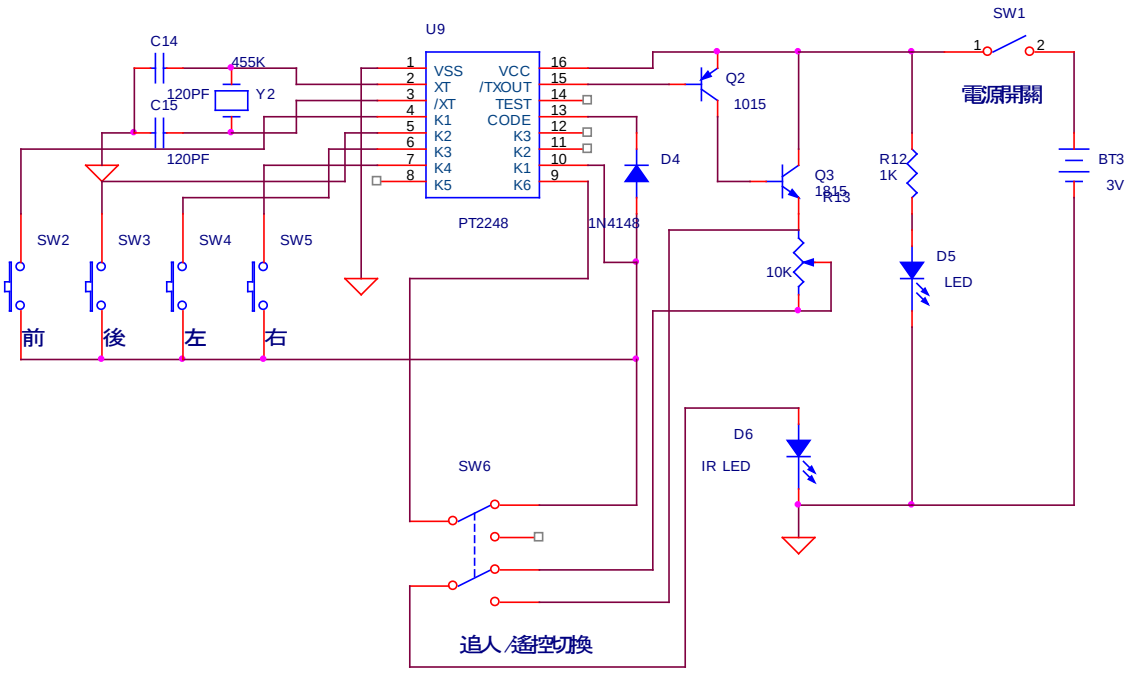

圖 4 紅外線發射系統

pt2248 是一個 COMS 紅外線遙控發射器，它有 18 種功能和 75 個指令可以使用，單點或連續按鍵皆可，可以運用於電視遙控、鐵捲門遙控等等。PT2248 特性：低消耗功率、需較少外接元件、寬電源供應 2.2~5V、RC 共振當作震盪步率來源[4-6]

紅外線接收系統 使用 PT2249

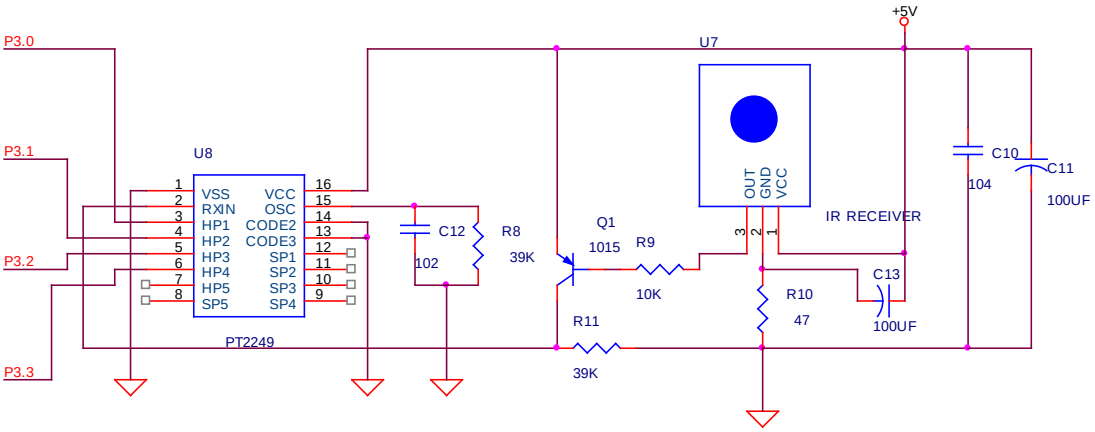

圖 5 紅外線接收電路

第四章、紅外線追蹤之 8051 程式設計流程

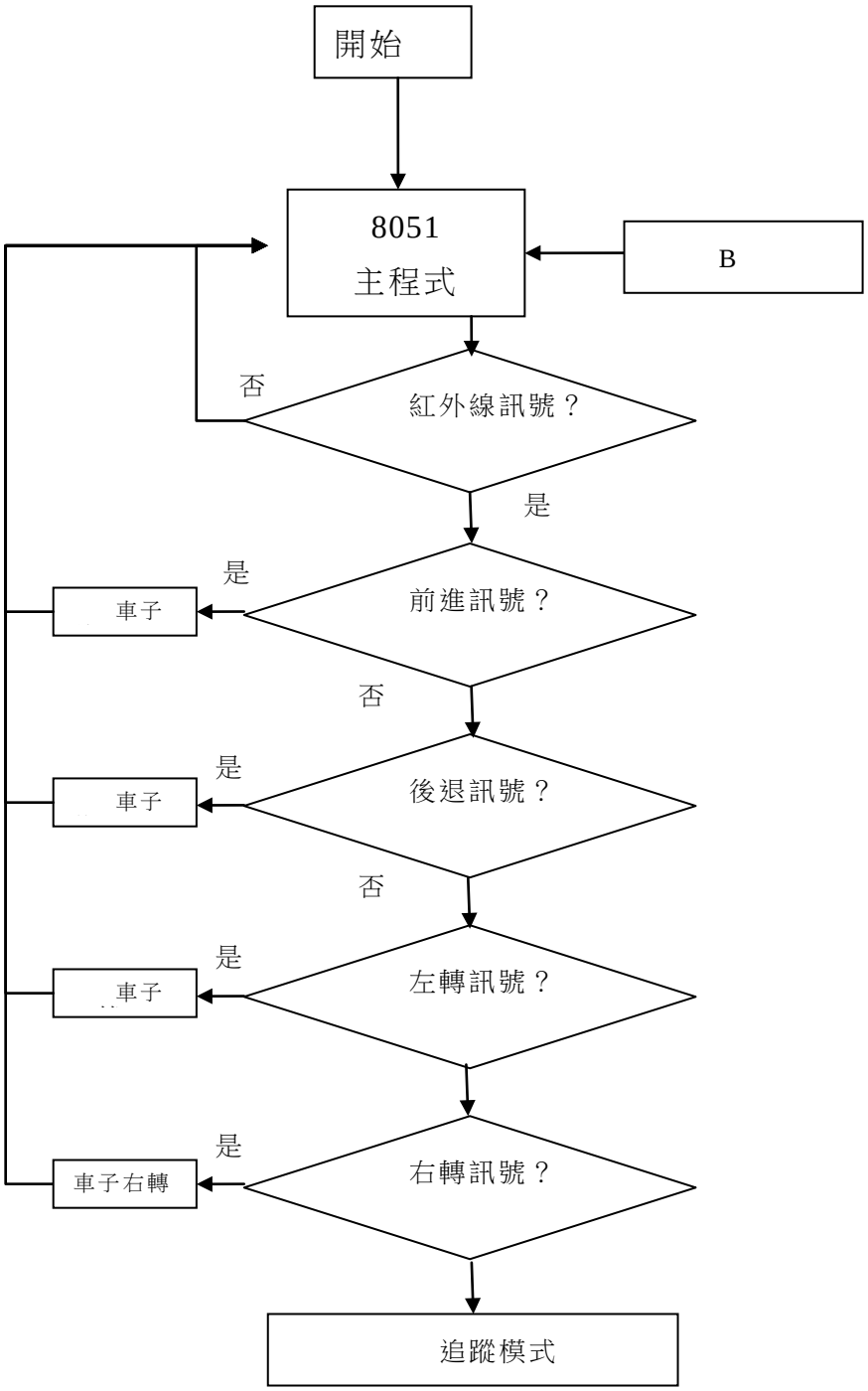

圖6 程式設計流程

第五章、實作與測試

一、硬體

表 1 零件表

項目	數量	編號	規格	
1	1	BT1	6V	電池
2	1	BT2	9V	電池
3	3	MG1,MG2,BT3	3V	馬達、電池
4	5	C1,C2,C3,C11,C13	100UF	電解電容
5	1	C4	220uF/16V	電解電容
6	3	C5,C6,C10	104	陶瓷電容
7	2	C7,C8	30PF	陶瓷電容
8	1	C9	10uF/25V	電解電容
9	1	C12	102	陶瓷電容
10	2	C14,C15	120PF	陶瓷電容
11	4	D1,D2,D3,D5	LED	
12	1	D4	1N4148	二極體
13	1	D6	IR LED	紅外線發射 LED
14	2	Q2,Q1	1015	電晶體 PNP
15	1	Q3	1815	電晶體 NPN
16	4	R1,R2,R3,R12	1K	電阻
17	4	R4,R5,R6,R10	47	電阻
18	3	R7,R9,R13	10K	電阻
19	2	R11,R8	39K	電阻
20	1	SW1	電源開關	
21	1	SW2	前	
22	1	SW3	後	
23	1	SW4	左	
24	1	SW5	右	
25	1	SW6	追人/遙控切換	
26	4	U1,U2,U3,U7	IR RECEIVER	紅外線接收模組
27	1	U4	L293	馬達驅動 IC
28	1	U5	78M05	5V 穩壓 IC
29	1	U6	89C51	
30	1	U8	PT2249	紅外線解碼 IC
31	1	U9	PT2248	紅外線編碼 IC

下圖是本文驅動繼電器的驅動電路：

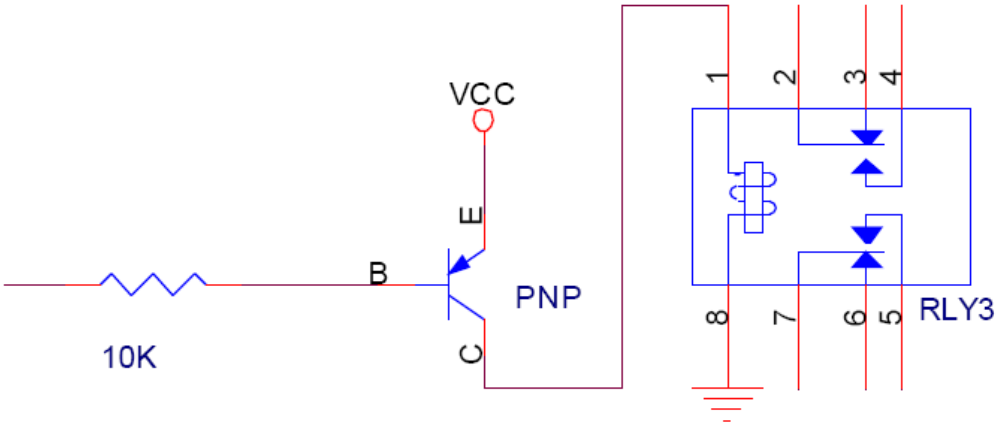

圖 7 繼電器的驅動電路

當 8051 I/O pin 為高電位時，PNP 電晶體的 E - B 極無順向偏壓，PNP 電晶體的 E - C 極不導通，如同開路，繼電器線圈不起動。

當 8051 I/O pin 為低電位時，PNP 電晶體的 E - B 極得到順向偏壓，PNP 電晶體的 E - C 極導通，VCC 被送到繼電器線圈一端，而繼電器線圈另一端接地，形成供電迴路，因而起動繼電器線圈。

馬達正反轉控制電路

一般直流馬達的兩端是沒有分正負端的，當馬達的兩端電壓反接時，馬達軸心會變成反方向旋轉。下面圖示是利用繼電器來控制直流馬達正反轉的電路圖示意圖解。圖 8 中 SW1 為開路，繼電器未充磁，故繼電器電門接點保持向上位置，所以馬達順向旋轉；圖 9 中將 SW1 電門壓下，繼電器作動使電門移至向下的位置，馬達則逆向旋轉。

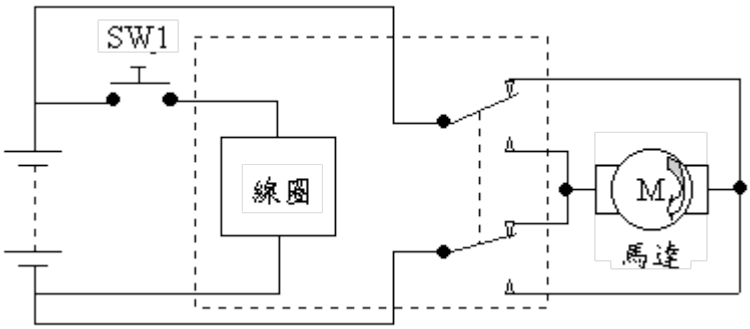

圖 8 馬達順向旋轉

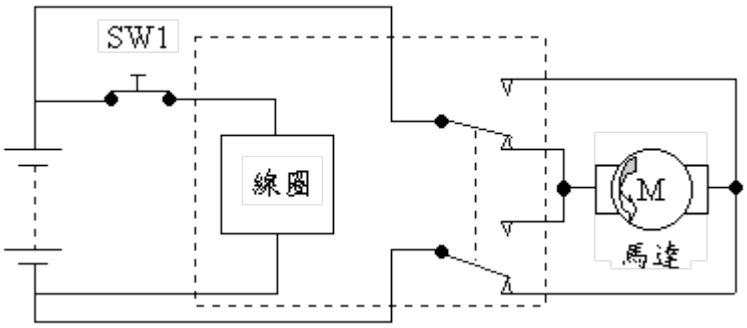

圖 9 馬達逆向旋轉

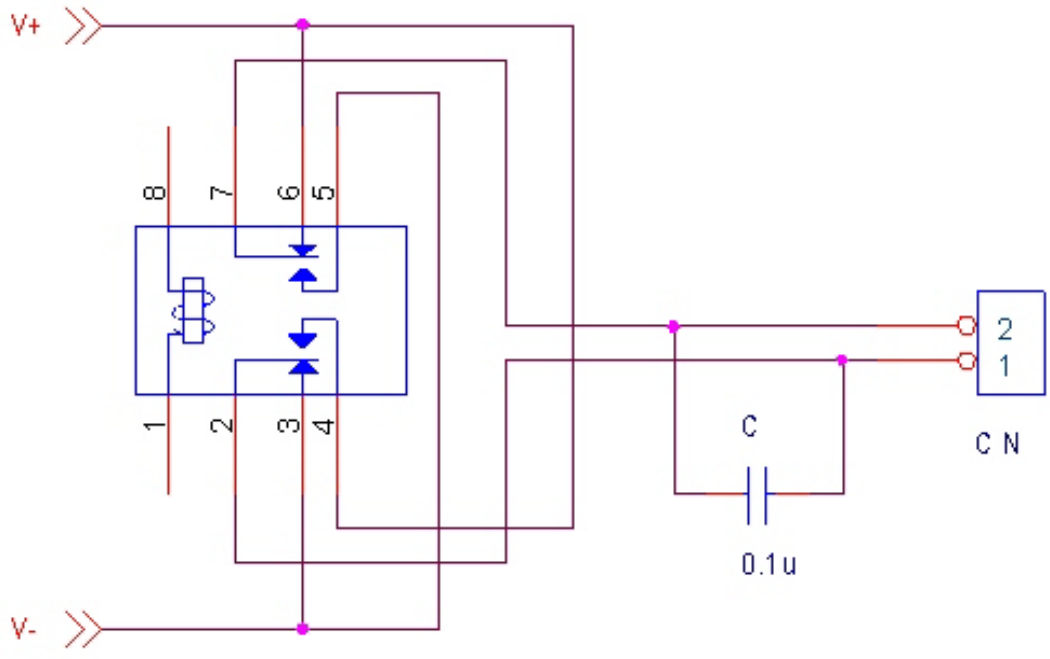

圖 10 直流馬達正反轉的應用電路

當繼電器線圈起動時，V+經繼電器 pin#4 接 pin#2 再到 CN pin#1，而 V-經繼電器 pin#5V+與 V-是來自供應馬達轉動的直流電源，V+是正極，V-是負極。電容 C 是用於消除電源雜訊，CN 是連接座，用於連接外部的直流馬達。

前面已說明過繼電器線圈的激磁起動方式，在此不贅述。當繼電器線圈未起動時，V+經繼電器 pin#6 接 pin#7 再到 CN pin#2，而 V-經繼電器 pin#3 接 pin#2 到 CN pin#1，外接馬達轉動，假設是正轉（此時繼電器 pin#4 與 pin#2 開路，pin#5 與 pin#7 開路）。

接 pin#7 到 CN pin#2，外接馬達得到相反的電壓，使用軸心反向轉動（此時繼電器 pin#4 與 pin#2 短路，pin#5 與 pin#7 短路）。

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：

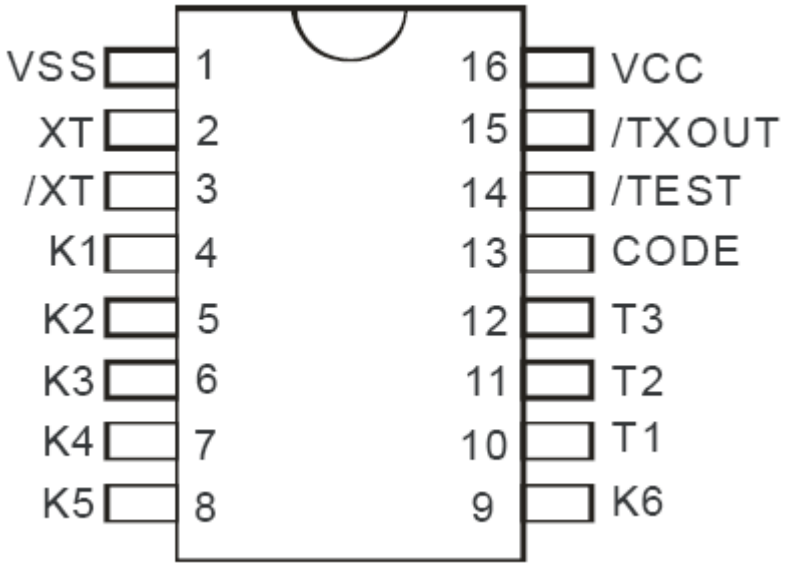

圖 11 PT2248 編碼傳送器

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。

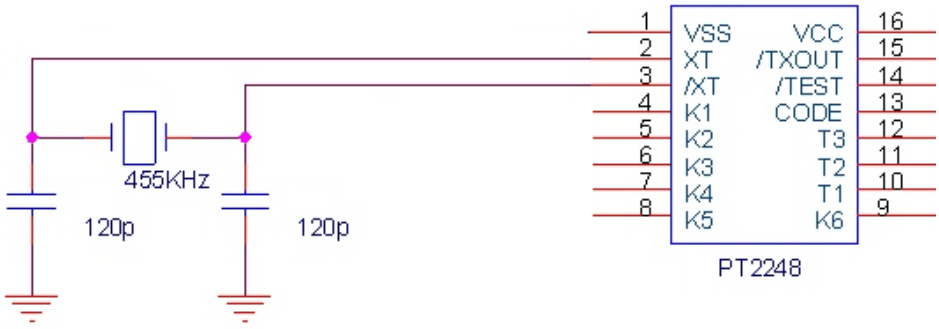

圖 12 PT2248 的振盪器輸入電路

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。

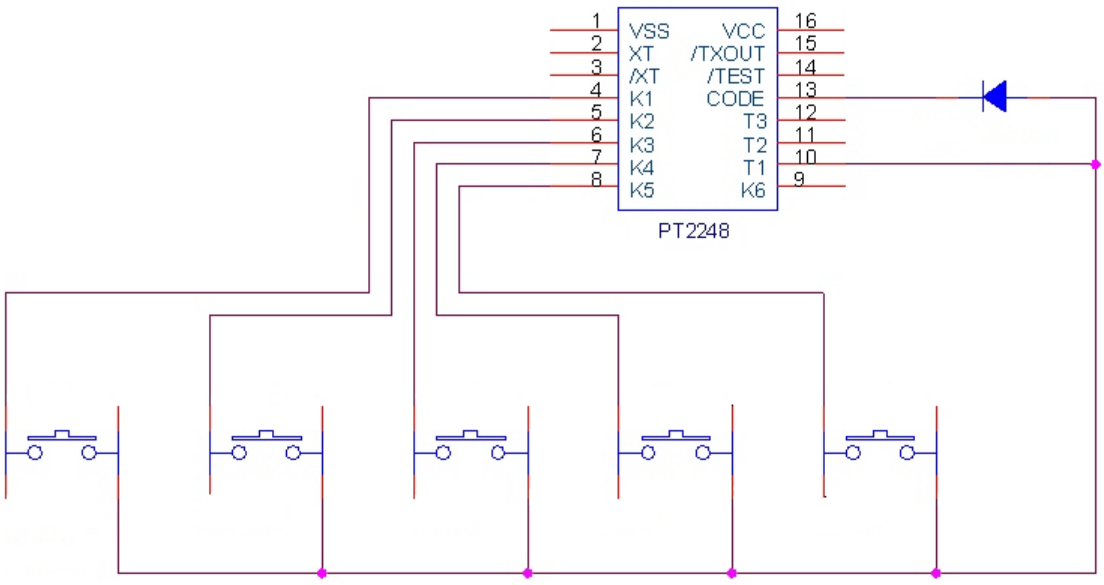

圖 13 PT2248 的按鍵輸入電路

下圖是 PT2248 的紅外線載波傳送電路，當/TXOUT 為高電位時，經 R 送到 PNP 電晶體的 B 極，由於 PNP 電晶體的 E 極、B 極電位相同，電晶體的 E 極、C 極不導通，IR LED（紅外線發射 LED）不動作。當/TXOUT 為低電位時，PNP 電晶體的 E 極、B 極形成順向偏壓，令電晶體動作，即 E 極、C 極導通，使紅外線發射 LED 得到工作電壓而動作。與紅外線發射 LED 串聯的電阻 R 是限流電阻，用以保護紅外線發射 LED。

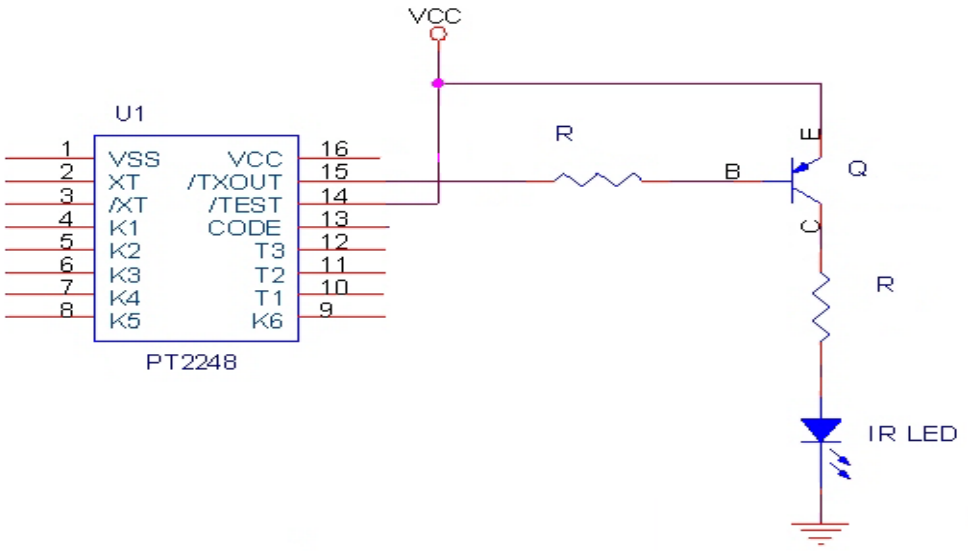

圖 14 PT2248 的紅外線載波傳送電路

PT2249 接收解碼器是 16 pin 的 IC，接腳功能如下：

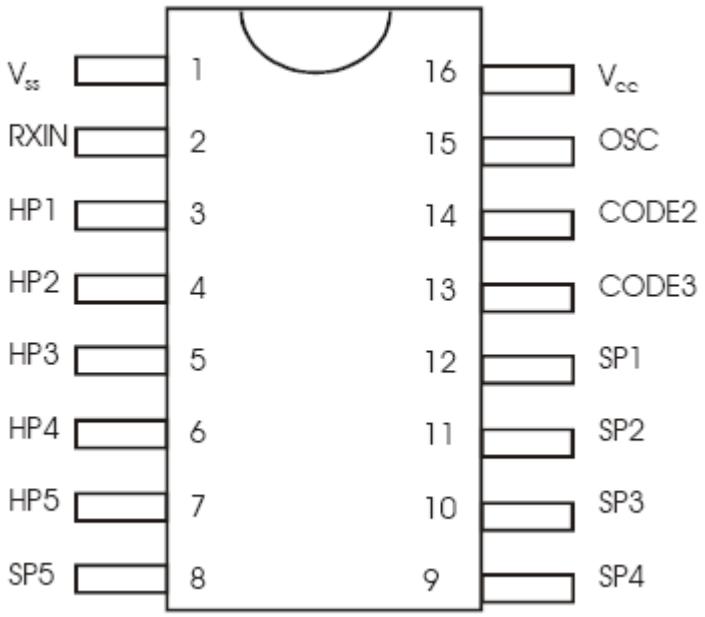

圖 15 PT2249 接收解碼器

Vss：電源接地

RXIN：載波輸入

HP1 ~ HP5：保持脈衝（Hold Pulse）輸出

SP1 ~ SP5：單脈衝（Single Pulse）輸出

CODE2 ~ 3：

OSC：振盪輸入，以電容與電阻並接到地

Vcc：電源輸入

PT2249 的振盪電路非常簡單，在 OSC pin 接 39K 電阻並接 1000p 的電容到地，即可讓 IC 內部振盪電路工作。

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。

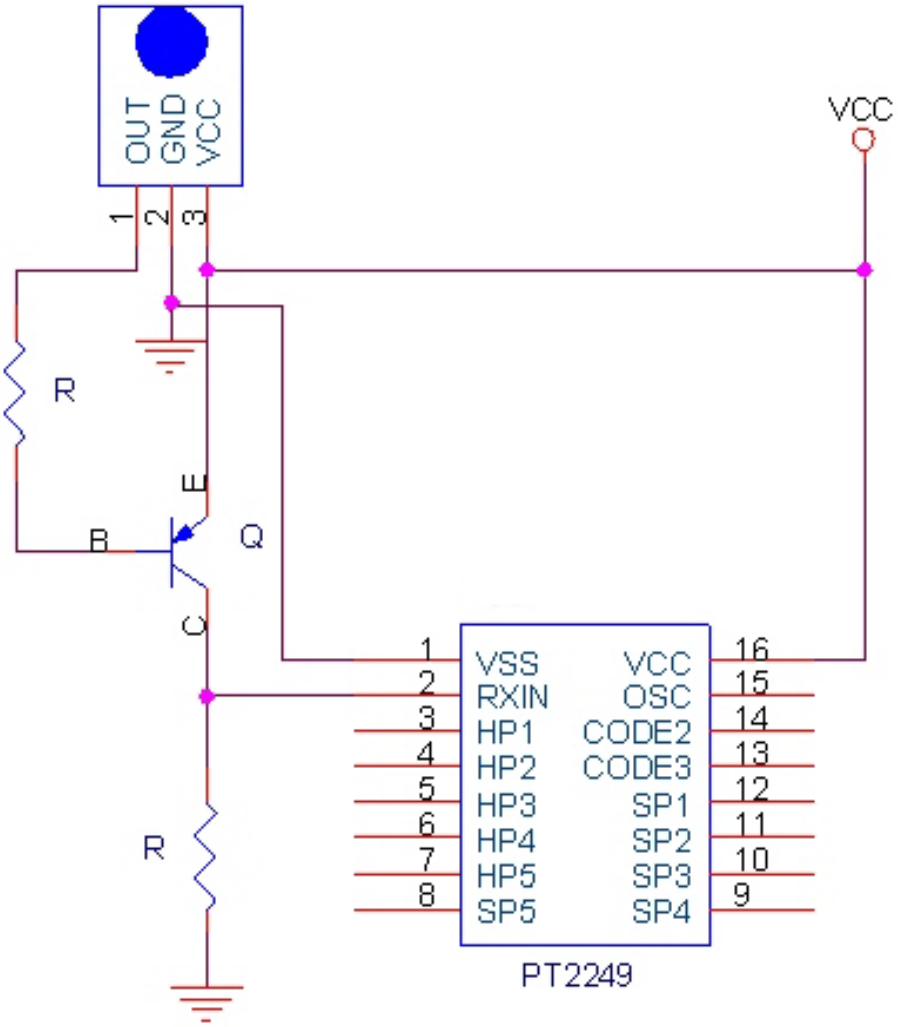

圖 16 紅外線接收電路

PT2249 解碼後，會由 HP1 ~ 5 及 SP1 ~ 4 輸出，輸出結果依配對的 PT2248 按鍵接法而定,本專題在 PT2248 端使用 5 個按鍵,固由 PT2249 的 HP1 ~ 5 輸出給 8051 來讀取。

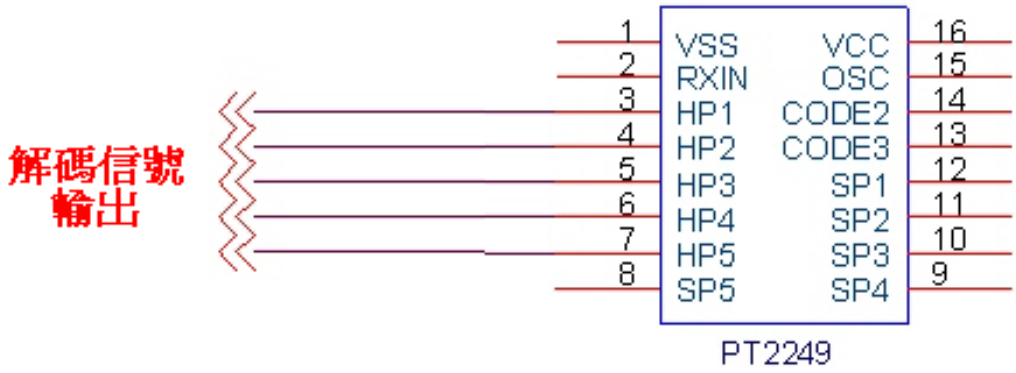

圖 17 紅外線接收解碼電路

硬體方塊圖

遙控器

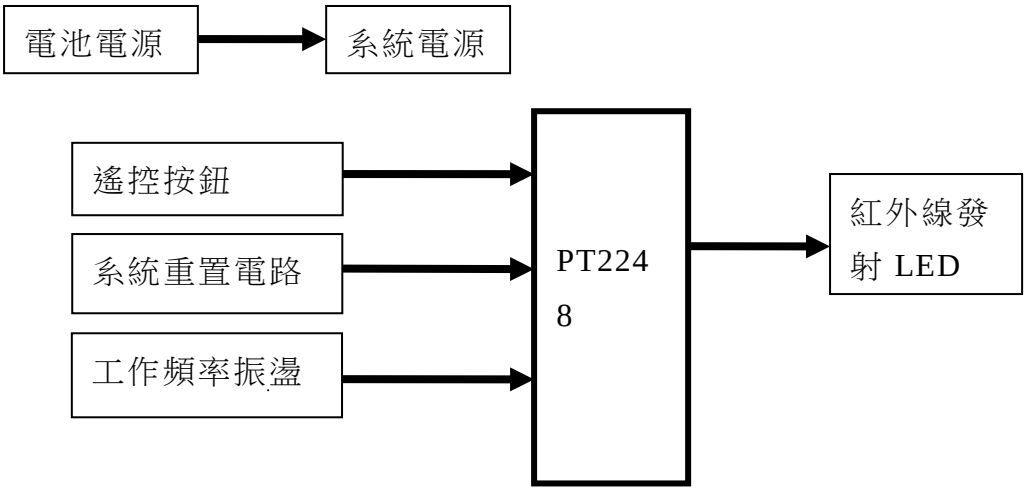

圖 18 遙控器硬體方塊圖

紅外線接收電路

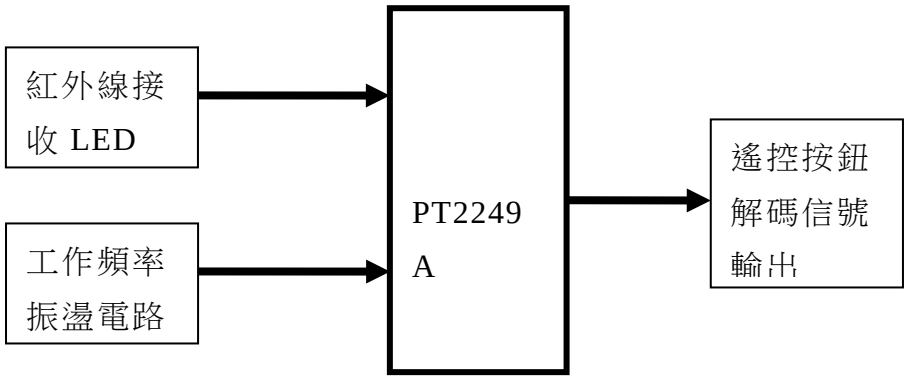

圖 19 紅外線接收電路硬體方塊圖

車體控制板控制電路

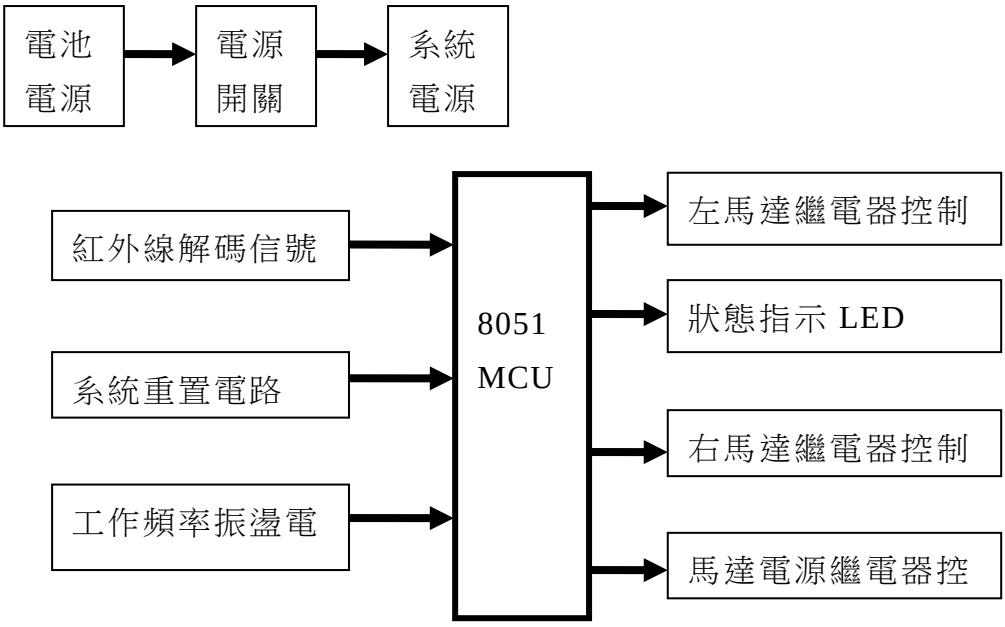

圖 20 車體控制板控制電路

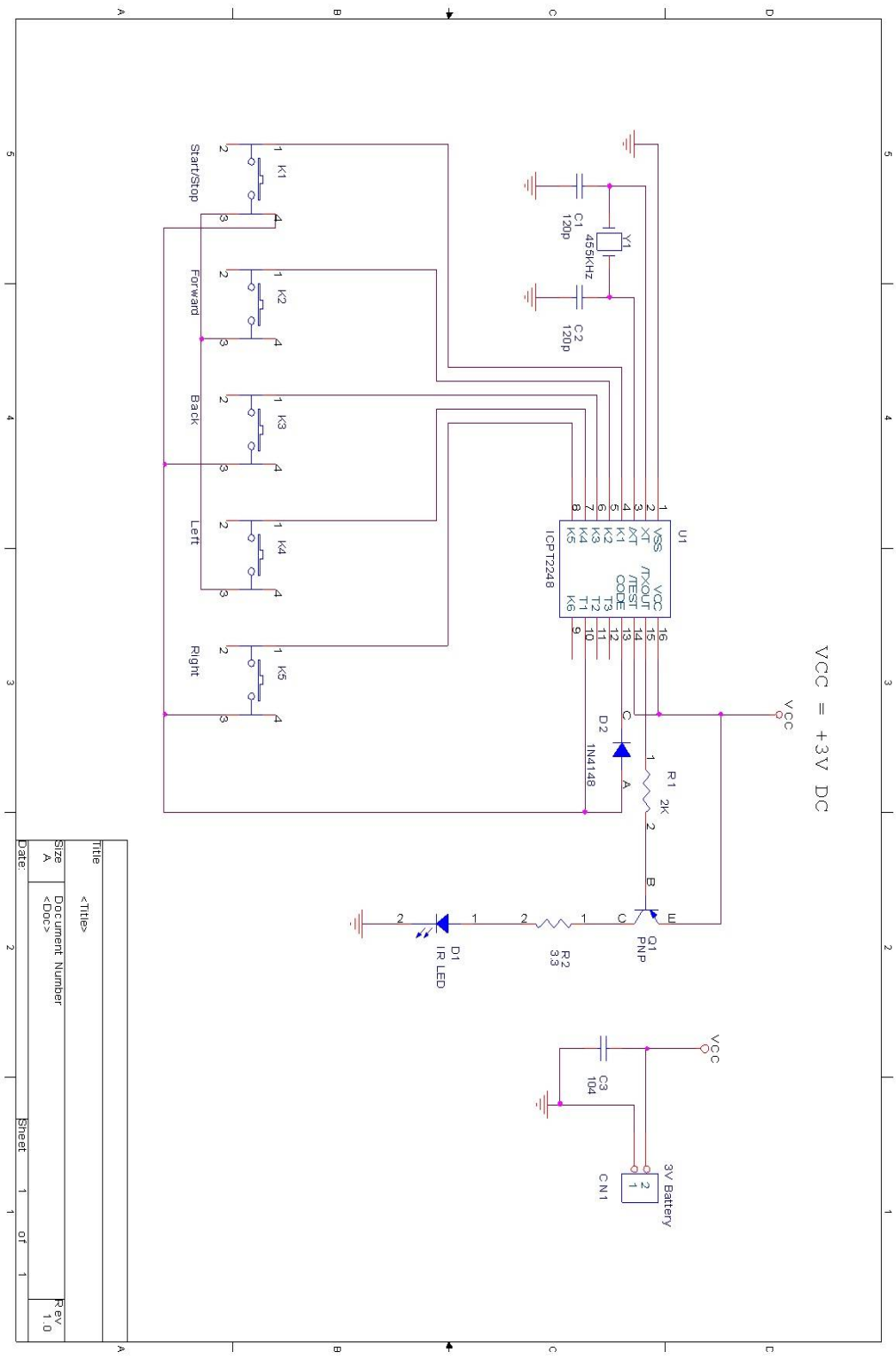
Start/Stop
Forward
Back
Left
Right
K1
K2
K3
K4
K5
C1
120p
C2
120p
Y1
455KHz
U1
ICPT2248
VSS
XT
/XT
/TEST
CODE
VCC
K1
K2
K3
K4
K5
T3
T2
T1
K6
D2
IN4148
R1
2K
VCC = +3V DC
D1
IR LED
R2
33
Q1
PNP
C3
104
CN1
3V Battery
Title
<Title>
Size
A
Document Number
<Doc>
Rev
1.0
Date:
Sheet
1
of
1

圖 21 遙控器線路

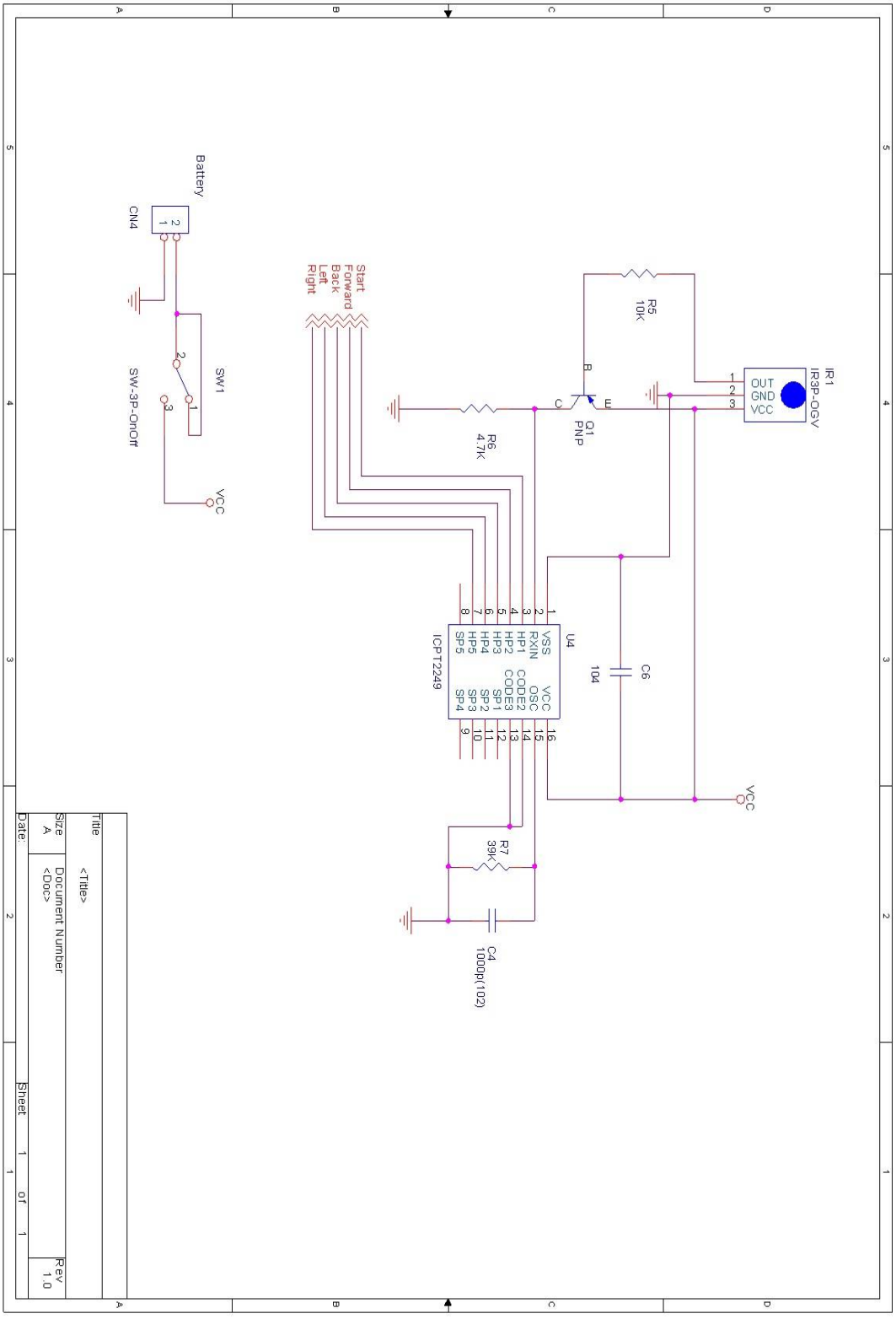
Battery
CN4
SW1
SW-3P-OnOff
VCC
Start
Forward
Back
Left
Right
R5
10K
R6
4.7K
Q1
PNP
IR1
IR3P-OGV
OUT
GND
VCC
U4
ICPT2249
VSS
RXIN
OSC
VCC
HP1
HP2
HP3
HP4
HP5
SP5
CODE3
CODE2
SP1
SP2
SP3
SP4
C6
104
R7
39K
C4
1000p(102)
Title
<Title>
Size
A
Document Number
<Doc>
Rev
1.0
Date:
Sheet
1
of
1

圖 22 紅外線接收電路

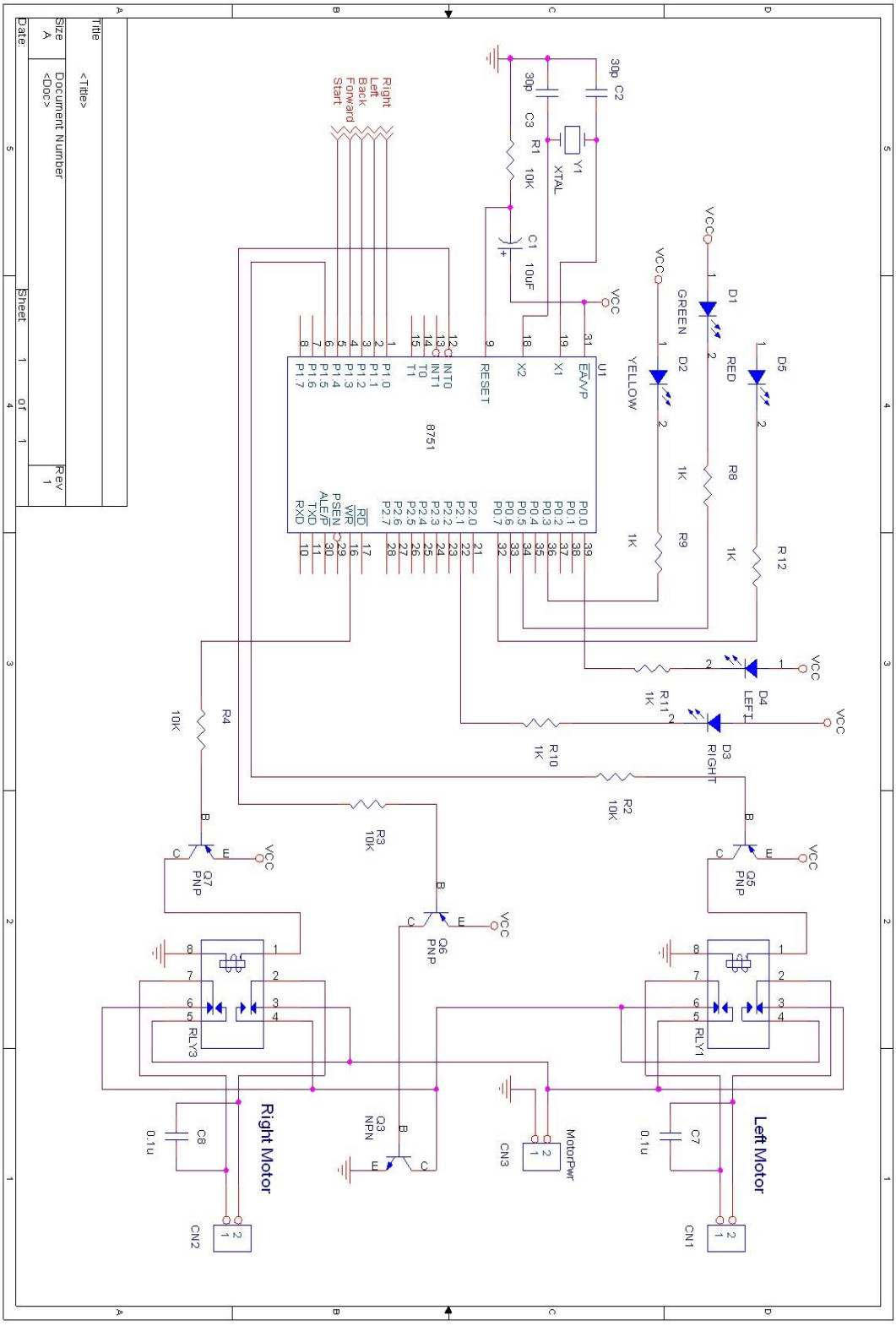
Right
Left
Back
Forward
Start
30p
C2
C3
R1
10K
Y1
XTAL
C1
10uF
VCC
U1
8751
RESET
X2
X1
EA/VP
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
INT0
INT1
T0
T1
P2.0
P2.1
P2.2
P2.3
P2.4
P2.5
P2.6
P2.7
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
RD
WR
PSEN
ALE/P
TXD
RXD
D1
GREEN
D2
YELLOW
D5
RED
R8
R9
R12
1K
D4
LEFT
D3
RIGHT
R11
R10
R4
R2
R3
10K
Q7
PNP
Q6
PNP
Q5
PNP
RLY3
RLY1
Right Motor
Left Motor
Q3
NPN
CN3
MotorPwr
C8
C7
0.1u
CN2
CN1
Title
<Title>
Size
A
Document Number
<Doc>
Rev
1
Date:
Sheet
1
of
1

圖 23 車體控制線路圖

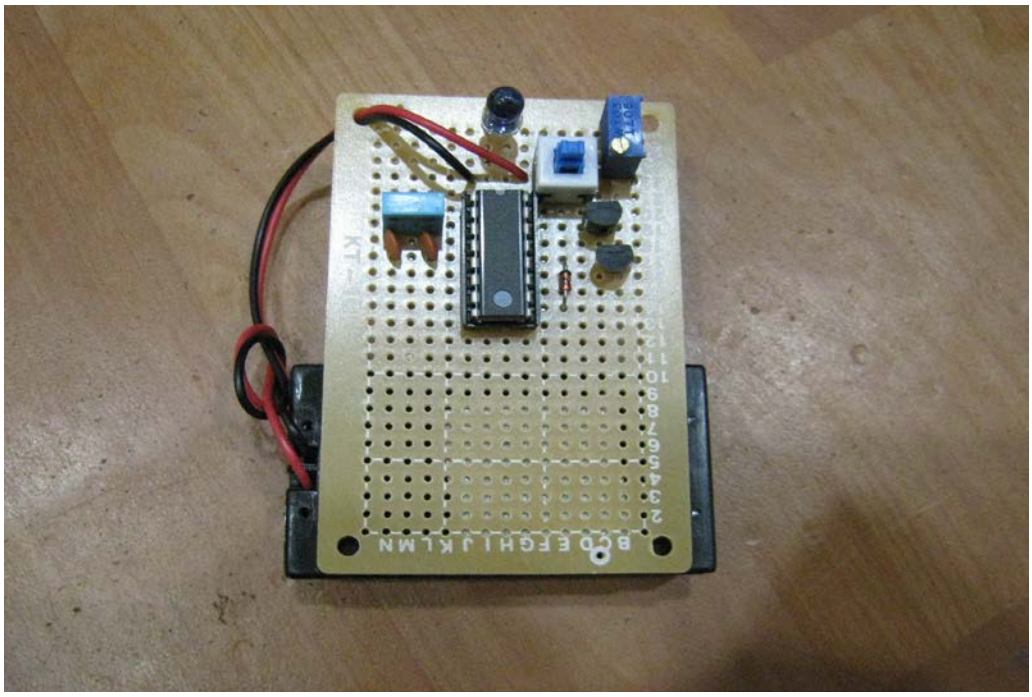

圖 24 紅外線接受器實體

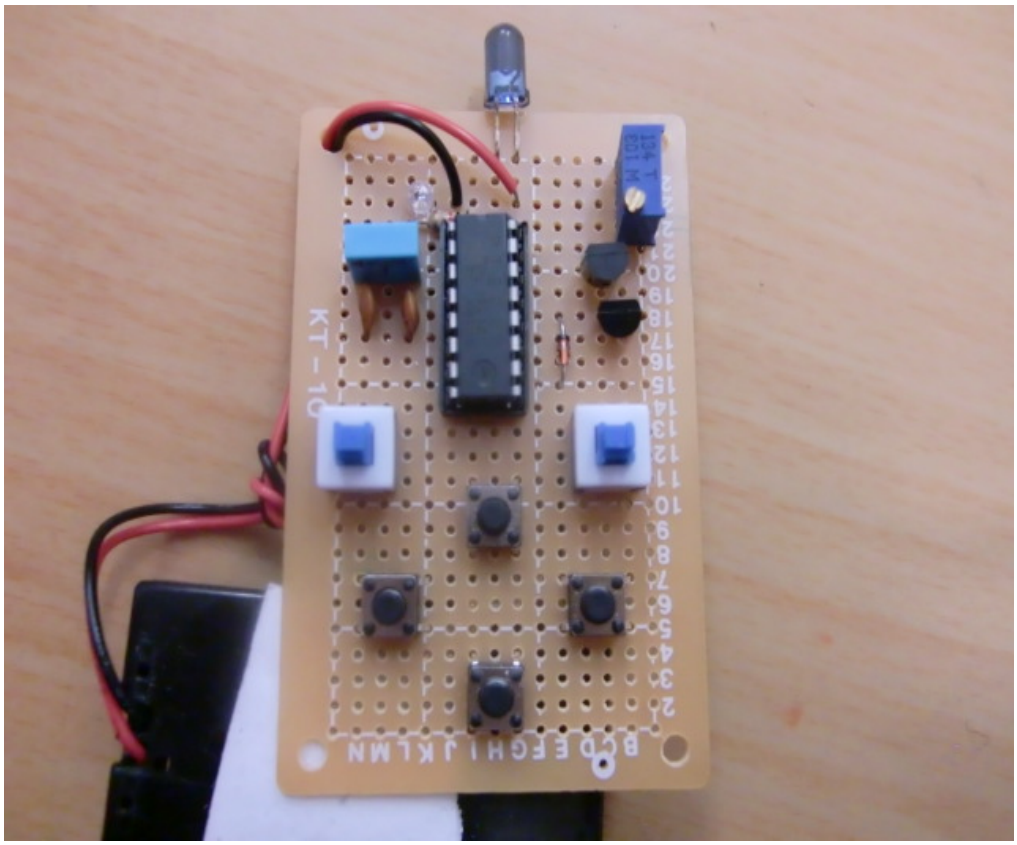

圖 25 紅外線發射器實體

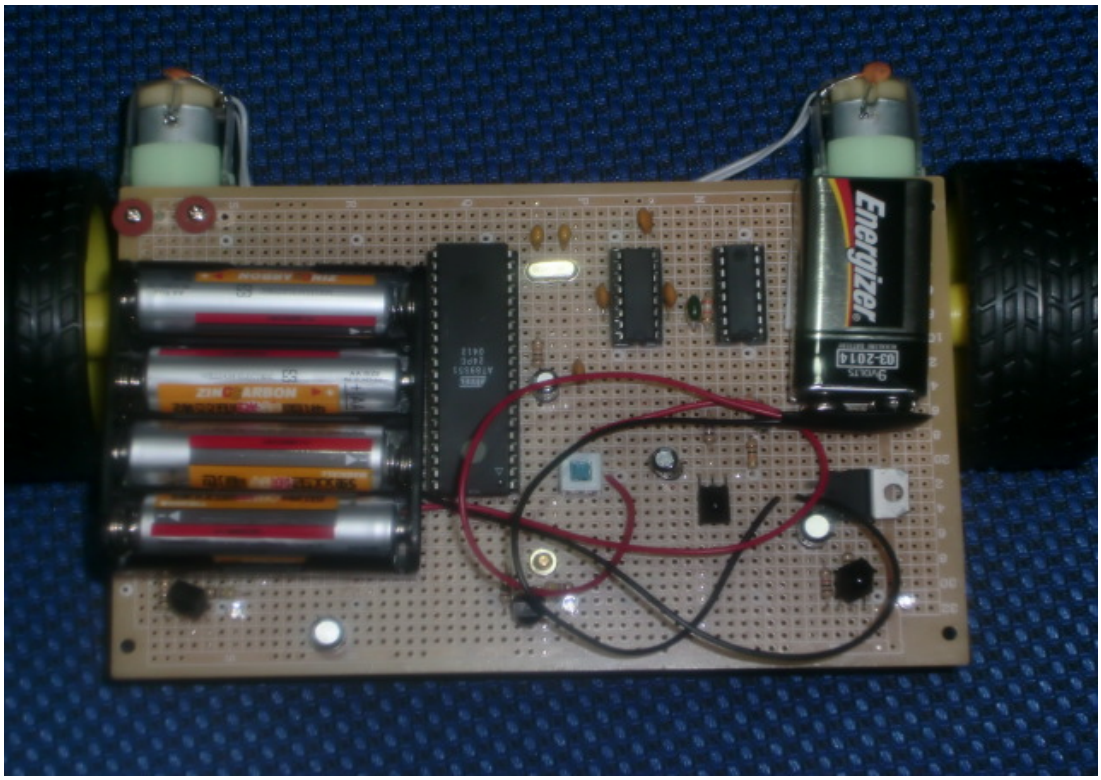

圖 26 紅外線追蹤車零件面

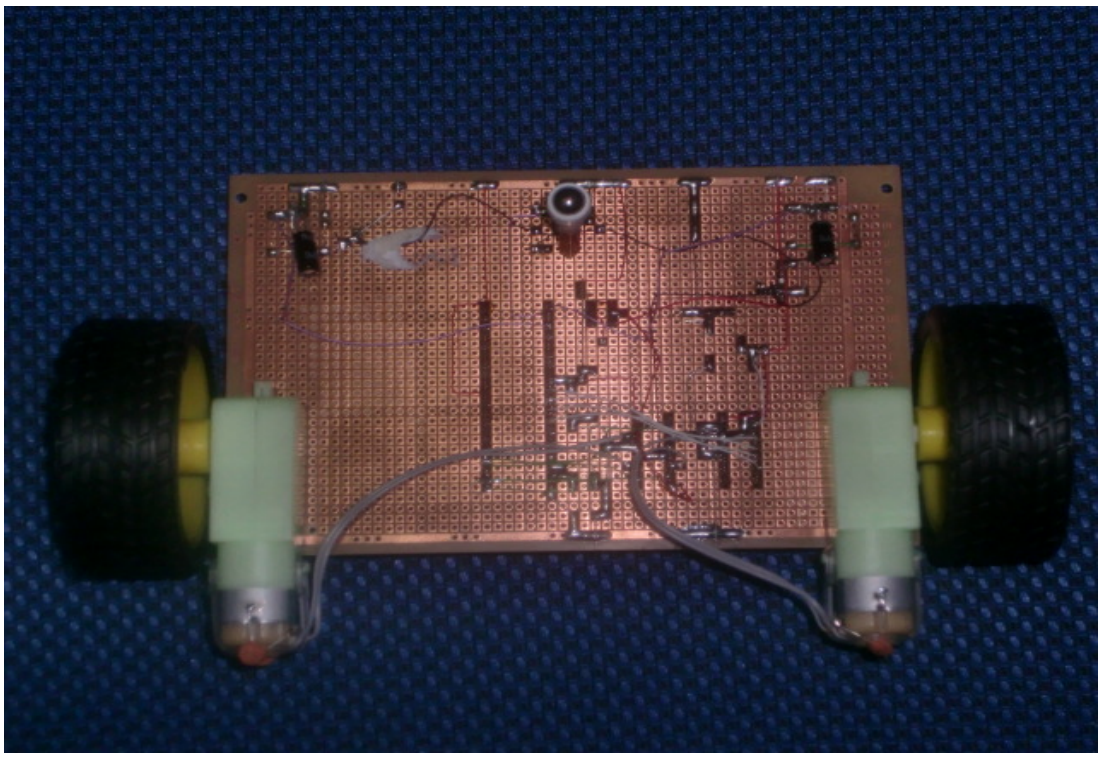

圖 27 紅外線追蹤車焊接面

二、軟體

程式結構主要分成 3 個程式模組：

Main.c：主程式，執行主要的程式迴路、感應輸入辨別及相關程序處理、以及馬達、燈號控制。

SensorIn.c 及 SensorIn.h：感應輸入偵測、輸入信號彈跳處理。

Tick.c 及 Tick.h：計時器中斷處理，提供標準鐘控時間，用於標準延遲時間計算。

用 Keil uVision2 組譯後，燒錄檔 HCR200.HEX 會在 EXE 的目錄下。

Main.c

```
#include <reg51.h>
#include "tick.h"
#include "sensorin.h"
#include "Led8x8.h"

#define WHEEL_FORWARD    1    //車輪向前
#define WHEEL_BACK       0    //車輪向後
#define MOTOR_ON         0    //馬達起動
#define MOTOR_OFF        1    //馬達停止
#define MOVING_STRAIGHT  0    //直走狀態
#define MOVING_LEFT      1    //左轉狀態
#define MOVING_RIGHT  2    //右轉狀態

#define BLINKONTIME      2    //閃燈亮的時間
#define BLINKOFFTIME   50   //閃燈滅的時間

#define LED_ON  0    //LED 亮
#define LED_OFF1    //LED 滅
#define MUSIC_ON    0    //啟動音樂
#define MUSIC_OFF   1    //關閉音樂
#define DIR_OFF     0    //方向指示關閉
#define DIR_UP      1    //方向指示向前
#define DIR_DOWN    2    //方向指示向後
#define DIR_LEFT    3    //方向指示向左
#define DIR_RIGHT   4    //方向指示向右

sbit LeftMotorDir = P1^7;  //控制左馬達方向
sbit RightMotorDir    = P3^6;  //控制右馬達方向
sbit MotorSwCtrl      = P1^6;  //控制左右馬達電源

sbit MotorSwLed       = P3^2;  //馬達起動顯示 LED
```

```
sbit RunBackLed       = P3^3;  //倒轉顯示 LED
sbit BlinkLed      = P1^5;  //閃燈 LED
sbit RunLeftLed       = P3^0;  //左轉顯示 LED
sbit RunRightLed      = P3^1;  //右轉顯示 LED
sbit MusicCtrl        = P3^7;  //音樂控制

void BlinkLight(void);

bit fBlinkOn;
bit fMovingForward;
unsigned int BlinkDelay;
unsigned int BlinkTimeRec;
unsigned int    SensorStatus,LastStatus;
unsigned char MovingDir;
unsigned char **CharTableBase;      //文字字型表指標
unsigned int FontRunIndex;      //字型跑馬位置索引值
bit fDisplayMode;//矩陣顯示模式

main()
{
    InitTick();     //起始計時器
    InitSensorIn();     //起始感應輸入狀態
    InitLedMatrix();   //初始 LED 矩陣

    LeftMotorDir = WHEEL_FORWARD;     //左馬達初始向前
    RightMotorDir = WHEEL_FORWARD;   //右馬達初始向前
    MotorSwCtrl = MOTOR_OFF;   //馬達關閉
    MotorSwLed = LED_OFF;  //馬達起動燈號關閉
    LastStatus = SEN_None;    //上一狀態設為無狀態
    fMovingForward = 1;       //預設車向前走旗標
    MovingDir = MOVING_STRAIGHT;      //車初始狀態設為向前狀態
```

```
    //閃燈初始設定
    fBlinkOn = 0;
    BlinkDelay = BLINKOFFTIME;
    BlinkTimeRec = GetSystemTick();

CharTableBase = DirectionTable;    //取得方向表位置
    fDisplayMode = SCAN_PASTE;//貼字顯示模式
    FontRunIndex = DIR_OFF; //初始顯示索引值

    while (1)
    {
        BlinkLight(); //閃爍燈號處理
        SensorStatus = GetSensorStatus();   //讀取按鈕感應輸入狀態
        if (SensorStatus != LastStatus)       //檢查按鈕感應輸入狀態是否
有改變
        {
            // 按鈕感應輸入狀態有改變, 執行相對應的功能
            switch (SensorStatus)
            {
                case SEN_Start:         //馬達起動與關閉按鈕
                if (MotorSwCtrl == MOTOR_ON)   //檢查目前馬達是否是
ON
                    {
                    MotorSwCtrl = MOTOR_OFF;  //關閉馬達
                    MotorSwLed = LED_OFF;       //LED 滅
                    if(fMovingForward)    //是否方向向前
                        {
                        FontRunIndex = DIR_OFF; //方向指示關閉
                        MusicCtrl = MUSIC_OFF; //倒車音樂關閉
                        }
                    }
                    else //目前馬達是 OFF
```

```
            {
                MotorSwCtrl = MOTOR_ON;    //起動馬達
                    MotorSwLed = LED_ON;   //LED 亮
                    if (fMovingForward)
                    { //車為向前狀態
                    FontRunIndex = DIR_UP;  //方向指示向前
                    }
                    else
                    { //車為倒車狀態
                    FontRunIndex = DIR_DOWN;//方向指示向後

MusicCtrl = MUSIC_ON;   //倒車音樂啟動
                    }
                }
                break;

            case SEN_Forward:     //向前按鈕
                // 設定車向前
                LeftMotorDir = WHEEL_FORWARD;     //設定左馬達
向前
                RightMotorDir = WHEEL_FORWARD;   //設定右馬達
向前
            RunBackLed = LED_OFF;  //關閉倒車 LED
            fMovingForward = 1;  //設定向前旗號
        if (MotorSwCtrl == MOTOR_ON)    //檢查目前馬達是否是 ON
                    FontRunIndex = DIR_UP;  //方向指示向前
                else
                    FontRunIndex = DIR_OFF; //方向指示關閉
                MusicCtrl = MUSIC_OFF;  //倒車音樂關閉
                break;
```

```
case SEN_Back:  //向後按鈕
                    // 設定車向後
                LeftMotorDir = WHEEL_BACK;     //設定左馬達向後
                RightMotorDir = WHEEL_BACK;        //設定右馬達向後
                    RunBackLed = LED_ON;  //打開倒車 LED
                    fMovingForward = 0;  //清除向前旗號
                    FontRunIndex = DIR_DOWN;  //方向指示向後
                    MusicCtrl = MUSIC_ON;  //倒車音樂啟動
                    break;
case SEN_Left:      //左轉按鈕
                    if (fMovingForward)
                    { //前進狀態下向左轉
                    LeftMotorDir = WHEEL_BACK;     //設定左馬達向後
                    RightMotorDir = WHEEL_FORWARD;  //設定右馬達
向前
                    }
                    else
                    { //倒車狀態下向左轉
                    LeftMotorDir = WHEEL_FORWARD;     //設定左馬達
向前
                    RightMotorDir = WHEEL_BACK;   //設定右馬達向後


}
                RunLeftLed = LED_ON;    //打開左轉 LED
                MovingDir = MOVING_LEFT;  //設定左轉旗號
                FontRunIndex = DIR_LEFT;     //方向指示向左
                break;

                case SEN_Right:        //右轉按鈕
                if (fMovingForward)
                { //前進狀態下向右轉
```

```
                    LeftMotorDir = WHEEL_FORWARD;     //設定右馬達向前
                        RightMotorDir = WHEEL_BACK;    //設定左馬達向後
                        }
                        else
                        { //倒車狀態下向右轉
                        LeftMotorDir = WHEEL_BACK;      //設定左馬達向後
                        RightMotorDir = WHEEL_FORWARD;   //設定右馬達
向前
                        }
                        RunRightLed = LED_ON;  //打開右轉 LED
                        MovingDir = MOVING_RIGHT;      //設定右轉旗號
                        FontRunIndex = DIR_RIGHT;   //方向指示向右
                        break;

                    case SEN_None:   //其他狀態
                        // 無按鈕感應
                        if (MovingDir != MOVING_STRAIGHT)
                        {     // 在左轉或右轉中放開按鈕
                            if (fMovingForward)
                            { //車為向前狀態,所以讓車繼續向前
                            LeftMotorDir = WHEEL_FORWARD;
                            RightMotorDir = WHEEL_FORWARD;
                            FontRunIndex = DIR_UP;  //方向指示向前
                            }
                            else
                            { //車為倒車狀態,所以讓車繼續向後
LeftMotorDir = WHEEL_BACK;
RightMotorDir = WHEEL_BACK;


FontRunIndex = DIR_DOWN;   //方向指示向後
                            }
```

```
                    RunLeftLed = LED_OFF;   //關閉左轉 LED
                     RunRightLed = LED_OFF; //關閉右轉 LED
                     MovingDir = MOVING_STRAIGHT;      //回到直行狀態
(前進或倒車)
                          }
                          break;
                }
                LastStatus = SensorStatus; //儲存此次按鈕感應狀態
          }
          ScanMatrixTabShift(CharTableBase,FontRunIndex,fDisplayMode);
     //LED 矩陣掃瞄顯示
     }
}

/*-------------------------------------------------
     閃爍燈號處理
-------------------------------------------------*/
void BlinkLight(void)
{
     if ((GetSystemTick() - BlinkTimeRec) >= BlinkDelay) //檢查閃燈亮滅時
間到
     {
          BlinkTimeRec = GetSystemTick();   //記錄目前鐘控時間
          if (fBlinkOn) //是否要亮
          {
               BlinkLed = LED_ON; //閃燈亮
               BlinkDelay = BLINKONTIME; //保持亮的時間
               fBlinkOn = 0;      //亮的時間到後，處理閃燈滅
          }
          else //閃燈滅直處理
          {
               BlinkLed = LED_OFF;//閃燈滅
```

```
                BlinkDelay = BLINKOFFTIME;//保持滅的時間
                fBlinkOn = 1;      //滅的時間到後，處理閃燈亮
            }
        }
}
```

相關函數程式

```
e KeyPort     P1   //按鈕讀取埠

typedef enum KeyStatusEnum {
    SEN_None    = 0x000,
    SEN_Right   = 0x001,
    SEN_Left    = 0x002,
    SEN_Back    = 0x004,
    SEN_Forward = 0x008,
    SEN_Start   = 0x010,
} KeyStatusType;

#define   KP_MASK          (SEN_Forward | SEN_Start | SEN_Back | SEN_Left
| SEN_Right)

#define AnyKey() ((KeyPort & KP_MASK) != 0)  //檢查有無任何按鈕被按
下
#define GetKey()  (KeyPort & KP_MASK)          //讀取按鈕狀態

unsigned char GetSensorStatus(void);

#ifndef   SENSORIN_C
    extern    unsigned char    KeyStatus;
```

```
#endif

void InitSensorIn(void);
void ScanSensorIn(void);
```

【SensorIn.c】

```
#define   SENSORIN_C

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

#define BOUNCE_CNT      3

typedef enum KeyPhaseEnum {
      KB_Idle,
      KB_Debounce,
      KB_Pressed
} KeyPhaseType;

      unsigned charKeyPhase;
      unsigned char       KeyStatus;
      unsigned char       KeyRecord;
      unsigned charBounceTime;

/*-------------------------
      掃瞄參數設定
-------------------------*/
void InitSensorIn(void)
{
      KeyPort |= KP_MASK;             //將按鈕輸入腳設為輸入狀態
      KeyPhase = KB_Idle;
```

```
    KeyStatus = SEN_None;
    KeyRecord = SEN_None;
    BounceTime = 0;
}

/*-------------------------
    讀取按鈕掃瞄狀態
-------------------------*/
unsigned char GetSensorStatus(void)

{
    return KeyStatus;
}
/*-------------------------
    按鈕掃瞄階段處理
-------------------------*/
void ScanSensorIn(void)
{
    switch ( KeyPhase )
    {
        case KB_Idle:                   //閒置階段
                if (AnyKey())           //檢查有無按鈕被按下
                {                                       //發現按鈕被按
下
                BounceTime = BOUNCE_CNT; //設定彈跳過濾時間
                KeyPhase = KB_Debounce;//設定下個階段為過濾彈跳階
段
                }
                break;

        case KB_Debounce:               //過濾彈跳階段
                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
                    if ( AnyKey() )
                    {    //有按鈕被按下
                    KeyStatus = GetKey();
                    KeyRecord = KeyStatus;
                    KeyPhase = KB_Pressed;   //設定下個階段為按鈕按
下階段
                    }
                    else
                    {    //沒有按鈕被按下
                    KeyPhase = KB_Idle;  //設定下個階段為閒置階段
                    KeyStatus = SEN_None;
                    }
                }

break;

        case KB_Pressed:                        //按鈕按下階段
                if ( KeyRecord != GetKey() )
                {
                /*  按鈕狀態有改變, 需回到過濾彈跳階段來辨別新的按
鈕狀態  */
                    BounceTime = BOUNCE_CNT;
                    KeyPhase = KB_Debounce;
                }
                break;

        default:
                break;
    }
}
```

【Led8x8.c】

```
#define LED8X8_C
#include <reg51.h>
#include "Led8x8.h"

#define ScanPort  P2   //掃瞄控制埠
#define SCAN_MASK 0xff //掃瞄埠罩遮位元
#define MatrixDataPort      P0   //字型資料埠

unsigned char DataTrans(unsigned char);
unsigned char ByteToScanBit(unsigned char);

unsigned char ScanIndex;   //掃瞄索引值

/* 字型 */

unsigned char code EmptyFont[]= {
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
};

unsigned char code SYM_Up[]= {
0x10,0x20,0x40,0xff,0x40,0x20,0x10,0x00
};

unsigned char code SYM_Down[]= {
0x08,0x04,0x02,0xff,0x02,0x04,0x08,0x00
};

unsigned char code SYM_Left[]= {
0x10,0x38,0x54,0x92,0x10,0x10,0x10,0x10
};
```

```
unsigned char code SYM_Right[]= {
0x10,0x10,0x10,0x10,0x92,0x54,0x38,0x10
};

yy1 = DataTrans(*(icon+ScanIndex));     //取得矩陣的資料
      //yy1 = *(icon+ScanIndex);//取得矩陣的資料
      port = ByteToScanBit(ScanIndex);

      ScanPort = SCAN_MASK; //關閉 LED 矩陣顯示

      MatrixDataPort = yy1; //送出字型資料

      //輸出掃瞄值
      if(yy1 != 0)
            ScanPort = port;   //送出掃瞄值到掃瞄埠

      //計算下一條掃瞄值
      ScanIndex++;//掃瞄索引值遞增
      if(ScanIndex == FONT_WIDE) //檢查是否到達最大值
            ScanIndex = 0;     //重置掃瞄索引值
}
/*----------------------------
   LED 矩陣資料值轉換
----------------------------*/
unsigned char DataTrans(unsigned char dat)
{
      unsigned char newdat;
      newdat = 0;
#if 1
      newdat |= (dat&0x01)? 0x80:0;
      newdat |= (dat&0x02)? 0x40:0;
      newdat |= (dat&0x04)? 0x20:0;
```

```
    newdat |= (dat&0x08)? 0x10:0;
    newdat |= (dat&0x10)? 0x08:0;
    newdat |= (dat&0x20)? 0x04:0;
    newdat |= (dat&0x40)? 0x02:0;
    newdat |= (dat&0x80)? 0x01:0;
#else
    newdat |= (dat&0x01)? 0x20:0;
    newdat |= (dat&0x02)? 0x80:0;
newdat |= (dat&0x04)? 0x10:0;
    newdat |= (dat&0x08)? 0x40:0;
    newdat |= (dat&0x10)? 0x02:0;
    newdat |= (dat&0x20)? 0x08:0;
    newdat |= (dat&0x40)? 0x04:0;
    newdat |= (dat&0x80)? 0x00:0;
#endif
    return(newdat);
}
/*----------------------------
  LED 矩陣掃瞄值轉換
----------------------------*/
unsigned char ByteToScanBit(unsigned char index)
{
                        unsigned char scanbit;

    switch(index)
    {
    #if 1
        case 0:
            scanbit = ~0x01;
            break;
        case 1:
            scanbit = ~0x02;
            break;
```

```
case 2:
            scanbit = ~0x04;
            break;
        case 3:
            scanbit = ~0x08;
            break;
        case 4:
            scanbit = ~0x10;
            break;
        case 5:
            scanbit = ~0x20;
            break;
case 6:
            scanbit = ~0x40;
            break;
        case 7:
            scanbit = ~0x80;
            break;
    #else
        case 0:
            scanbit = ~0x01;
            break;
        case 1:
            scanbit = ~0x80;
            break;
        case 2:
            scanbit = ~0x40;
            break;
        case 3:
            scanbit = ~0x04;
            break;
        case 4:
```

```
                scanbit = ~0x20;
                break;
            case 5:
                scanbit = ~0x02;
                break;
            case 6:
                scanbit = ~0x08;
                break;
            case 7:
                scanbit = ~0x10;
                break;
    #endif
    }
    return (scanbit);
}    /*----------------------------
      初始 LED 矩陣掃瞄顯示
----------------------------*/
void InitLedMatrix(void)
{
    ScanIndex = 0;    //初始掃瞄索引值
}
```

【Led8x8.h】

```
#define FONT_WIDE  8    //字點寬
#define FONT_SIZE   8    //字型位元組數量
#define LED_MATRIX_ON      0    //打開 LED 矩陣顯示
#define LED_MATRIX_OFF     1    //關閉 LED 矩陣顯示
#define SCAN_PASTE0    //貼字顯示
#define SCAN_SHIFT 1    //跑馬顯示
```

```
void InitLedMatrix(void);
void ScanMatrixTabShift(unsigned char **,unsigned int,bit);

#ifndef LED8X8_C
extern code unsigned char *DirectionTable[];
#endif
```

【Tick.h】

```
#define TICK_INTERRUPT_PERIOD_MS     10   //鐘控時間中斷週期(ms)

#define MS_20    (20 / TICK_INTERRUPT_PERIOD_MS)
#define MS_30    (30 / TICK_INTERRUPT_PERIOD_MS)
#define MS_40    (40 / TICK_INTERRUPT_PERIOD_MS)
#define MS_50    (50 / TICK_INTERRUPT_PERIOD_MS)
#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)
#define SEC_2    (2 * SEC_1)
#define SEC_3    (3 * SEC_1)

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

【Tick.c】

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

#define TICK_INTERRUPT_PERIOD_CNT
     (((XTAL*TICK_INTERRUPT_PERIOD_MS)/1000)/12)
     /*********************************************************
     MICRO_ADJUST = 計時器中斷時間準確度微調，如中斷誤差，單位=
指令時間,
                          值減少則調慢(中斷週期時間調長)
     *********************************************************/
#define MICRO_ADJUST   22          //鐘控計時準確度微調
#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 */
}
始系統鐘控計時器中斷設定參數
----------------------------------------*/
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

    ScanSensorIn();  //掃瞄按鈕狀態
}
```

第六章、結論

現在的科技進步很快，產品數位化自動化已經是一個趨勢，電子產品就是讓人們生活更加方便、便利，以節省時間，現在自動化的東西在這社會上已經非常普遍的存在。

本文設計紅外線追蹤系統時，是參考了網路上別人研究的一些資料做為初步的構想和設計，因為紅外線可以控制許多東西，經過分析與探討，最後決定用紅外線控制車子來做為教學相長的計畫。

本論文設計並實作出紅外線追蹤系統。本文採用以二元化、去除雜訊、連結區塊、驗證、追蹤的順序來進行偵測，本系統經實驗得到的良好的操控性，而加入了 tracking 的機制之後更可以達到自動化功能。但是在某些情況，如障礙物造成紅外線的遮蔽、過多的雜訊導致效果未臻完美，希望將來系統可以導入藍芽無線機制，使得追蹤能力更加提升。

參考文獻

[1]. 黃宏彥、余文俊、楊國輝編著，感測器原理與應用電路實習，高立圖書有限公司，
1999。

[2]. 蔡朝陽，電工實習（四），全華科技圖書股份有限公司，中華民國八十三年五月。

[3]. 鄧錦城編著，8051單晶片實作寶典，益眾資訊有限公司，2000。

[4]. M. Nilsson, D. Binnie, and A. Armitage, “Pedestrian Detection using Low-resolution Thermal Imager Versus Visual Imager,” *PREP 2004*, pp.156-157, UK, 2004

[5] A. Shashua, Y. Gdalyahu, and G. Hayon, “Pedestrian Detection for Driving Assistance Systems: Single-frame Classification and System Level Performance,” *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp.1-6, Parma, Italy, 2004.

[6] F. Xu, X. Liu, and K. Fujimura, “Pedestrian Detection and Tracking with Night Vision,” *IEEE Trans. on Intelligent Transportation Systems*, VOL.6, pp. 63-71, 2005.