中華民國第 55 屆中小學科學展覽會

作品說明書

國中組　生活與應用科學科

第三名

030805

有『跡』可『循』

學校名稱：嘉義市私立嘉華高級中學(附設國中)

作者：	指導老師：
國一 林揚益 國二 李冠郁 國二 李岳霖	李威璋

關鍵詞：循跡、Arduino、比例控制法

摘　　要

近年來由於自動控制與通訊技術的突破，使得無人自走車的導引系統從傳統的軌道式、磁導式進步到視覺影像系統，讓無人自走車的應用範圍更廣、功能更多。在自動化工廠中無人自走車是最常見的配備，且在自動倉庫系統中亦必備的伙伴。無人自走車如何在既定的路徑行走且不偏離路徑其最重要的是路徑判別的控制方法。為了改善循線時的不平順感，並減少偏移量的增加，故希望能夠對於現今的自走車做更進一步的改良。為了提升自走車之多功能性以及增強其機體運作準確度，本研究設計一輛採用光敏電阻循跡自走車並探討光敏電阻數目與演算法不同之差異。實驗結果顯示 Z 字型跑法有較佳之穩地定度，而比例控制循跡法雖然穩定度沒那麼高，但相對的有著較佳之速度。

壹、研究動機

近年來由於自動控制與通訊技術的突破，使得無人自走車的導引系統從傳統的軌道式、磁導式進步到視覺影像系統，讓無人自走車的應用範圍更廣、功能更多。在自動化工廠中無人自走車是最常見的配備，且在自動倉庫系統中亦必備的伙伴。

而現在無人自走車除了是工廠的物流搬運系統外，依其原理的應用也越來越多，如辦公大樓自動送公文機器人、探險機器人、導盲機器人與輪椅上下車動搬運裝置等。這表示無人自走車的應用已經朝著生活化、智慧化與人性化的趨勢發展，而本研究道路自動標線自走車之研製也是順應此趨勢提出。無人自走車如何在既定的路徑行走且不偏離路徑其最重要的是路徑判別的控制方法。

通常循線式自走車的馬達轉速是根據軌道狀態而決定，當車子行進於彎道型軌道時容易造成循線不平順感發生，同時也讓車體中心線與軌道中心線的產生偏移量。為了改善循線時的不平順感，並減少偏移量的增加，故希望能夠對於現今的自走車做更進一步的改良。為了進一步提升自走車之多功能性以及增強其機體運作準確度，本研究設計一輛採用光敏電阻循跡自走車並探討光敏電阻數目與演算法不同之差異。

貳、研究目的

一、 設計並製作出循線自走車

二、 透過程式撰寫讓自走車能循線移動

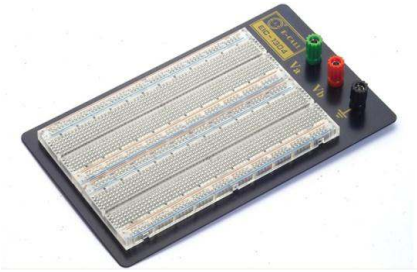
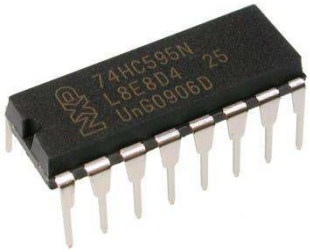
三、 透過 Arduino 將馬達之相關數據回傳電腦分析

四、 研究並探討現有機器人循跡的方式

五、 探討感測器數目、高度與演算法不同之影響

參、研究設備及器材

鍵盤敲擊辨識系統製作工具及材料

		
電阻	電路實驗板	9V 電池盒
		
74HC595	RGB LED 5050	Arduino Uno
		
光敏電阻	馬達減速組	萬向輪

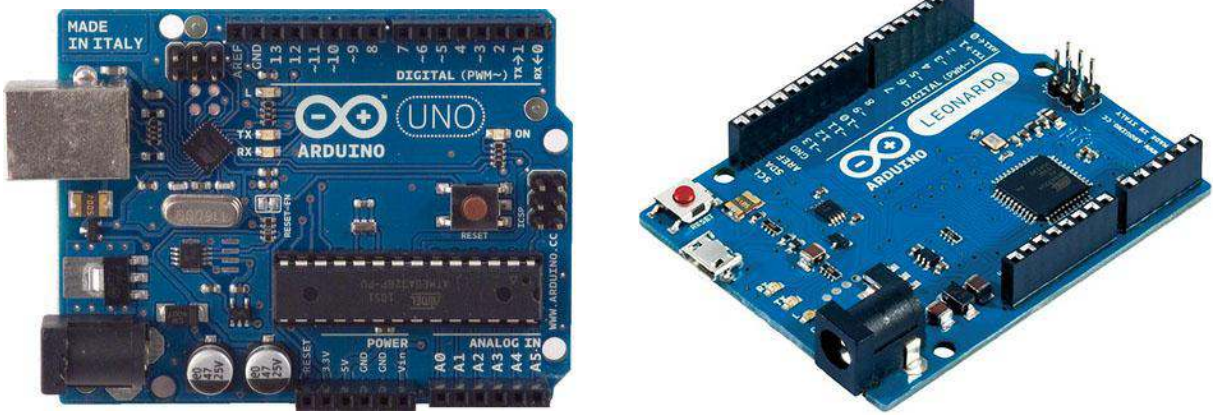
肆、研究過程或方法

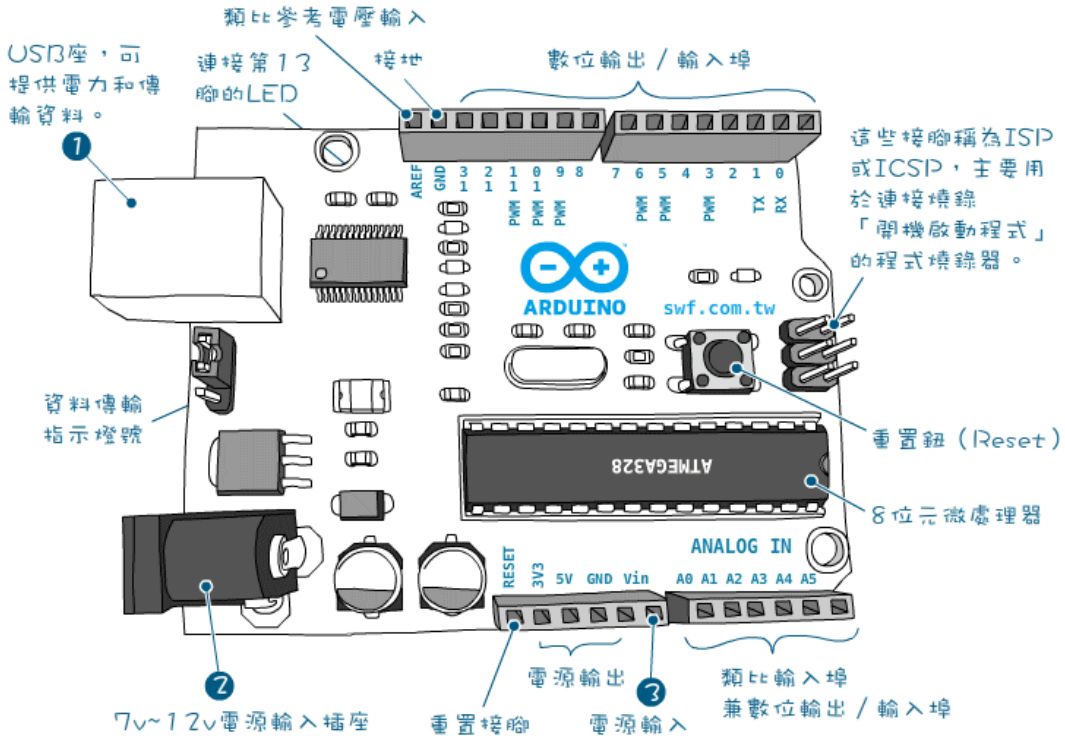
一、 Arduino說明

Arduino(圖1)是一塊基於開放原始碼發展出來的 I/O介面控制板，並且具有使用類似java,C語言的開發環境，讓使用者可以快速使用Arduino語言與Flash或 Processing...等軟體，作出互動作品。Arduino是在 2005 年 1 月由米蘭互動設計學院的教授 David Cuartielles和 Massimo Banzi所設計出來了，原始構想是希望讓設計師及藝術家們，透過 Arduino 很快的學習電子和感測器的基本知識，快速的設計、製作作品的原型，很容易與目前設計系所學的FLASH ,MAX/MSP,Virtool等軟體整合，使得虛擬與現實的互動更加容易。互動的內容設計才是設計師的主要訴求，至於怎麼拼湊一個單晶片開發板，或是當中涉及如何構築電路之類的知識，就並非設計師需要了解的，因此非常適合不具電子背景的人使用，以設計出各種不同的互動裝置。

二、 控制板

Arduino包含了硬體與軟體兩大部分，硬體部分是一個約手掌大小的控制板(寬 70mmX 高54mm)，核心使用八位元 ATMEGA8系列的的微控制器，提供14個數位式輸出/入端，6個類比式輸出/入端，支援 USB 資料傳輸，可以使用電源(5V~9V)或是直接使用 USB 電源，使用者可以在數位式輸出/入端上接上不同的電子裝置，例如 LED 燈、喇叭、馬達，然後再由控制器來驅動燈的亮滅、喇叭發聲、馬達運轉。Arduino控制板採用開放式源碼設計的概念，電路設計圖、韌體都可以在網路上下載，稍具電子知識就可以自行製作；也可以在網路上用很便宜的價錢買到。





產品規格

控制器核心：ATmega328

控制電壓：5V

最大輸入電壓 (limits)：6-20 V

數位 I/O Pins：14 (of which 6 provide PWM output)

類比輸入 Pins：6 組

Flash Memory：32 KB (of which 0.5 KB used by bootloader)

SRAM：2 KB

EEPROM：1 KB

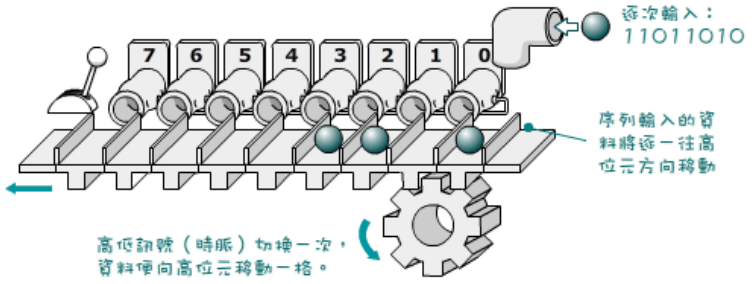
Clock Speed：16 MHz

圖1、Arduino 控制板架構圖

1. IC 74HC595

IC 74HC595：IC 74HC595是一種移位暫存器IC，如圖2所示，顧名思義，它可以暫時儲存部分資料並將其逐步移位，IC腳位及功能如下:

腳位編號	名稱	說明
1-7, 15	Q0 ~ Q7	輸出腳位
8	GND	接地
9	Q7'	序列輸出 (Serial Out)
10	MR	Master Reset, 清除所有資料, 低電位有效 (Active low)
11	SH_CP	SHift register clock pin (Clock Pin)
12	ST_CP	STorage register clock pin (Latch Pin)
13	OE	Output Enable, 允許輸出, 低電位有效 (Active low)
14	DS	序列資料輸入 (Serial data input)
16	Vcc	供應電壓



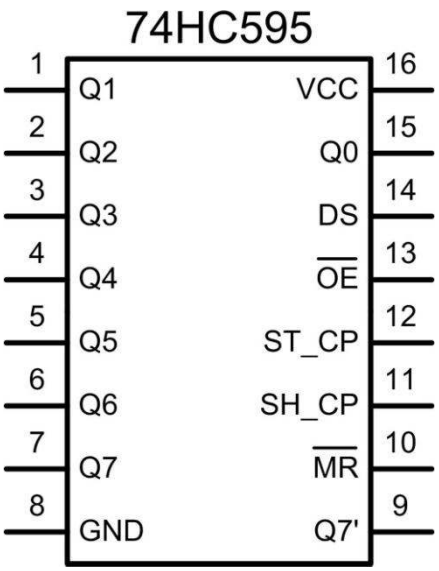


圖2、IC 74HC595 IC腳位

原理解說:

Vcc為高電壓電源，Gnd為低電壓電源，齒輪即為Clock(ST_CP)，每當Clock Pin有正緣(由低電壓變成高電壓)時，會讓當時Data Pin(DS)、水管)的狀態(低/高壓)存入Q0、Q0存入Q1、Q1存入Q2、…、Q7則會送至Q7’(資料輸出)，因此每當Clock變化一次，Q0~7腳位即會往前一次，藉此達到擴增輸出腳位的目的(以訊號增加輸出腳)。

如圖3所示，一開始Data in的資料隨著Clock的跳動(時間變化)而往下移動，藉此將資料一個一個往下推，所以只要讓輸入的Data in照著我們想要輸出的組合逐次送出，便可以得到相同排列的Q0~Q7。

而 Latch(SH_CP)則是一個開關，輸入高電壓時可以讓實際 Q0~Q7 的資料(高/低壓)隨著內部 Clock 跳動而變化，所以可以透過控制 Latch 讓資料未更新完時先不要輸出，等到 Clock 工作結束再把 Latch 打開讓資料出去，如此即不會看到未更新完成的資料移動的狀態。

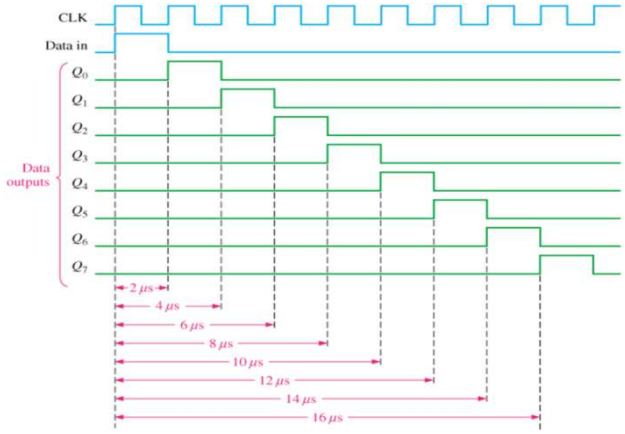


圖 3、資料移動狀態示意圖

三、循跡自走車製作

(一) 車體製作

步驟一： 馬達組裝於車體

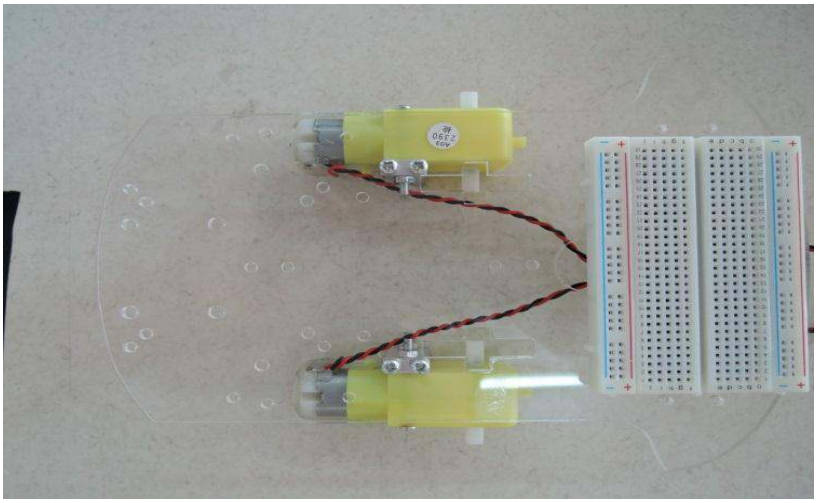


圖4、馬達組裝

步驟二：輪子組裝於車體

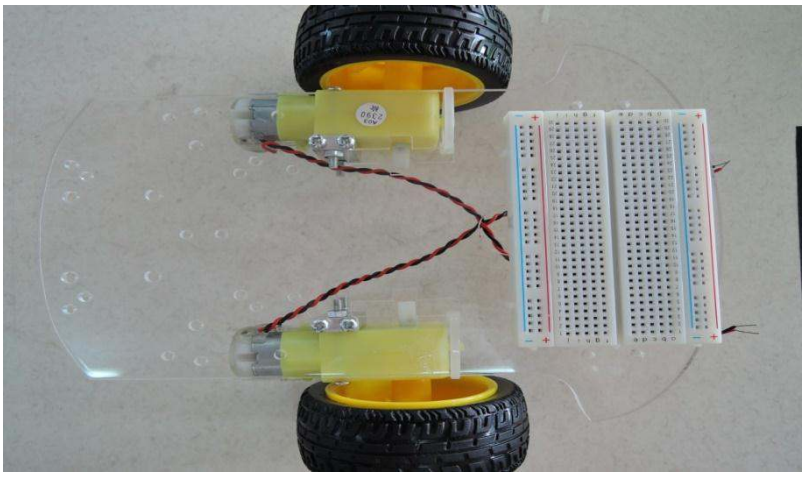


圖5、車輪組裝於車體

步驟三：將萬向輪用螺絲鎖上車體

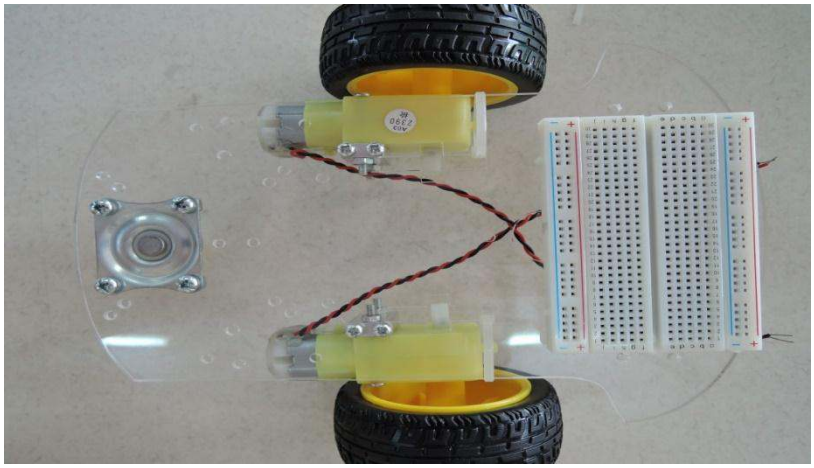


圖6、萬向輔助輪組裝

(二) 光感遮罩製作

步驟一： 利用3D繪圖軟體繪製光感遮罩

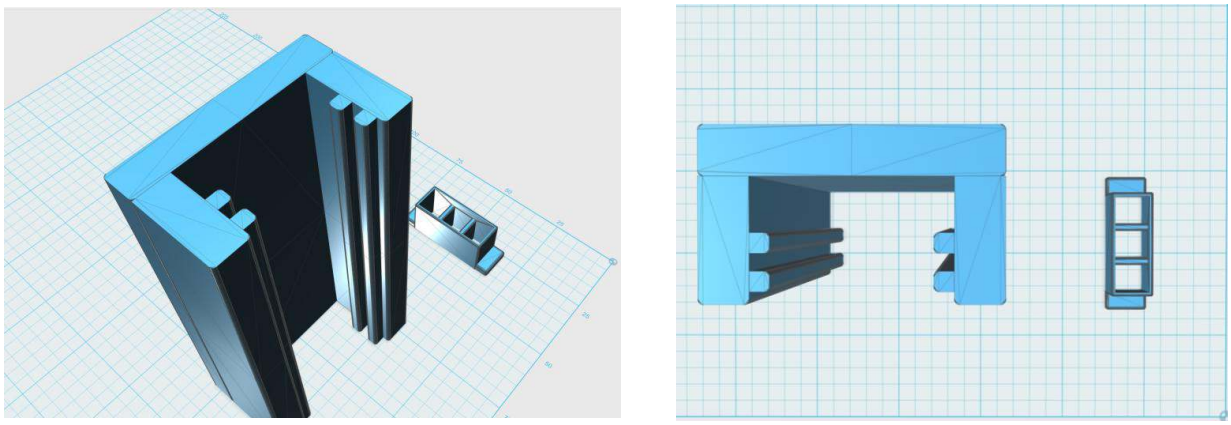


圖7、光感測器遮光罩繪製

步驟二： 利用3D列印機列印出光遮罩

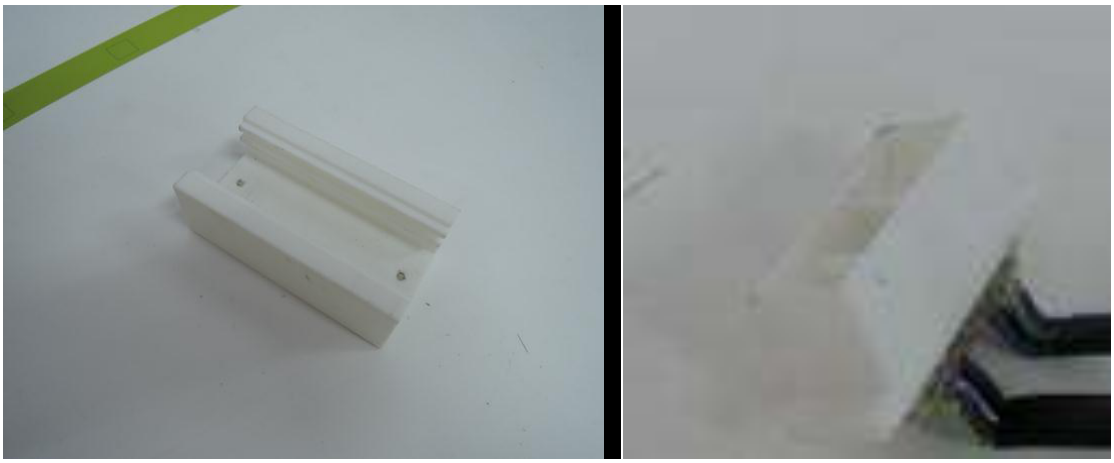


圖8、3D列印模型結果

步驟三： 將光敏電阻與LED燈固定於光遮罩

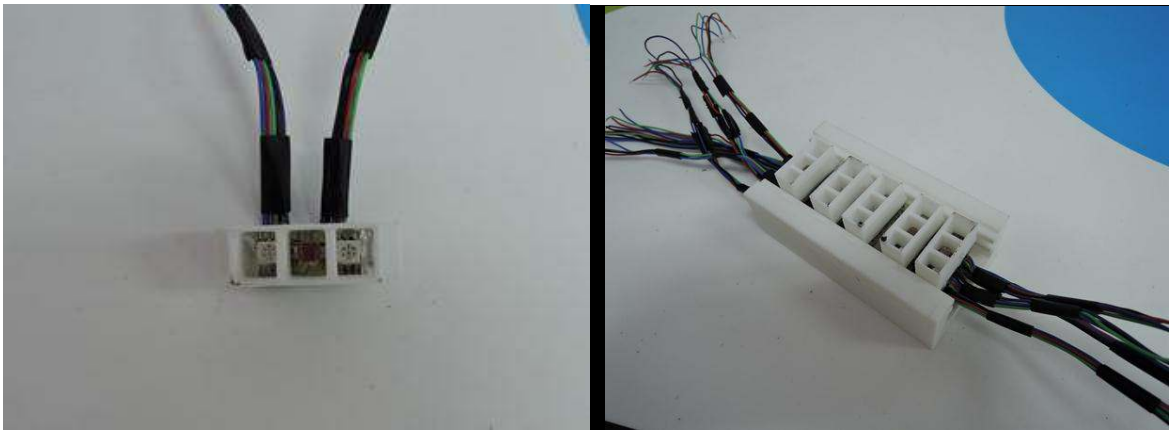


圖9、遮光罩模組整合

(三) 電路製作並固定各個元件

步驟一：將兩組電池固定於車體上，分別供應電源給LED燈與Arduino

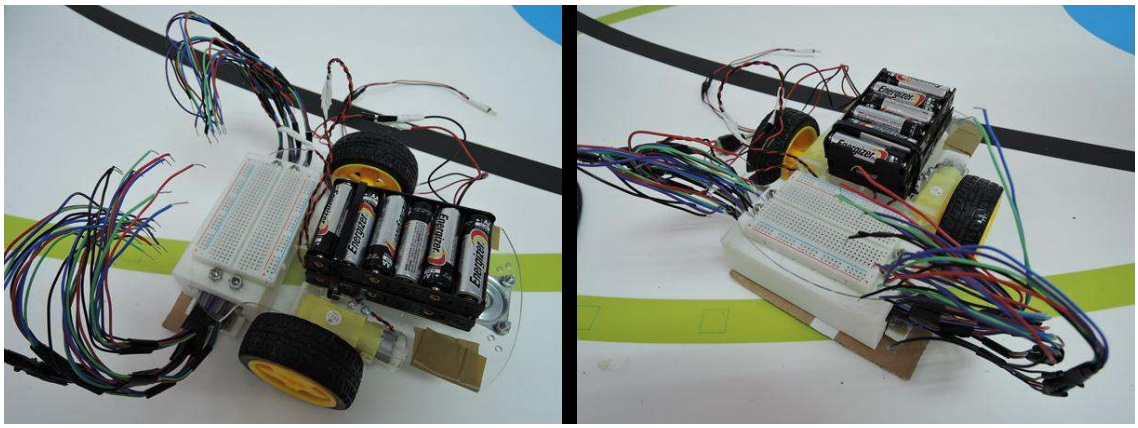


圖10、電源模組固定

步驟二：Arduino Uno 板 與麵包板、線路固定於車體外殼上

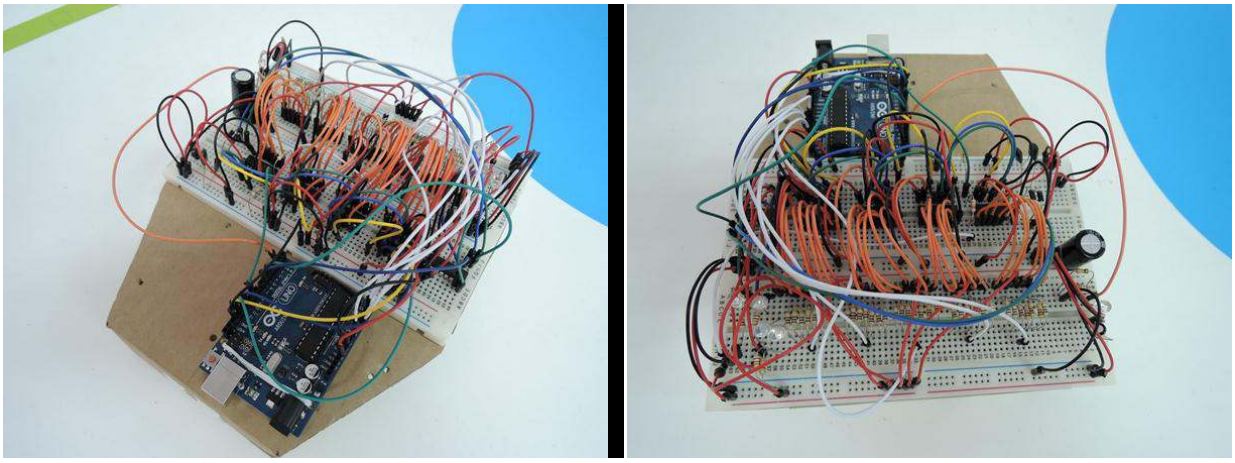


圖11、實體線路固定

步驟三：接上電源 ，即完成自走車設計

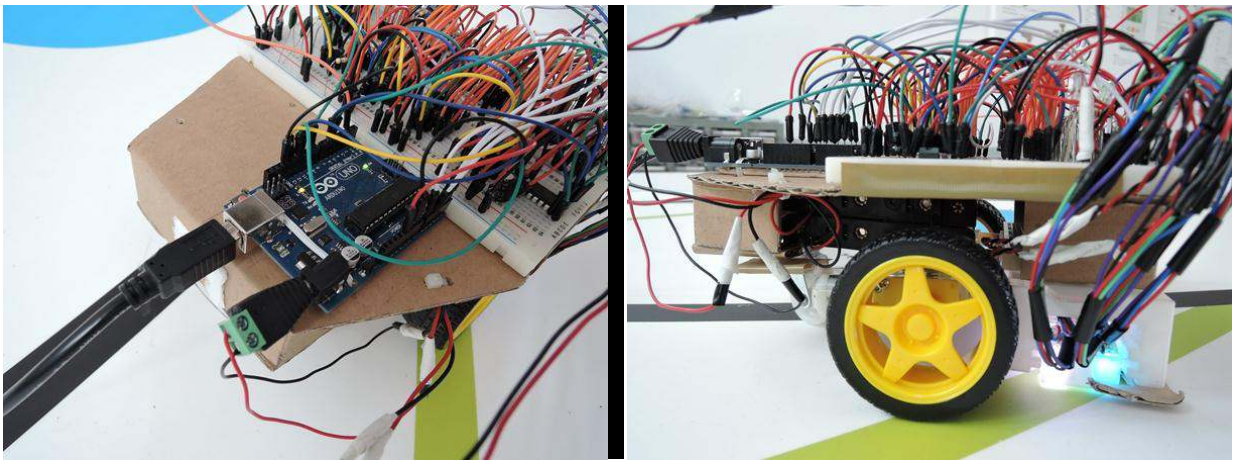


圖12、自走車電源連接測試

(四) 程式撰寫

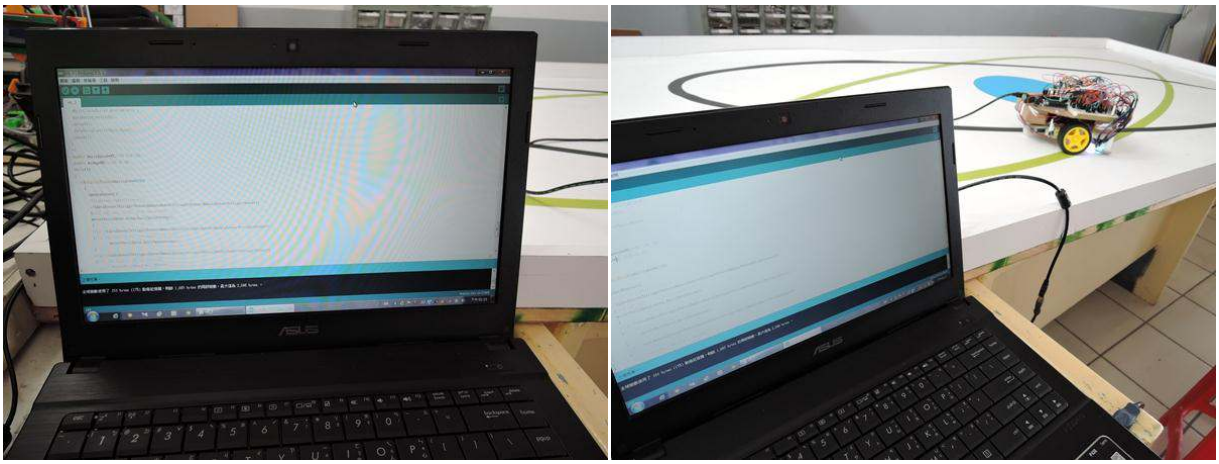


圖 13、自走車程式撰寫

循跡自走車演算法

1. Z 字型循跡法

車子走動時呈 Z 字型的運動軌跡，遇到黑線就朝另一個方向避開，遠離黑線則朝黑線靠近，行動時只有左轉或右轉兩個動作，因在轉彎時會降低另一邊的馬達轉速，所以會慢慢前行。也就是”非黑即白”的控制方式。

光線感測模組，可以藉由讀取到不同比例的光線反射量來判別是否為黑線，如下圖14，用數字來舉例，當光線感測器放在完全黑色的線上面，感測器讀取到40，而當放在完全白色的地面上的時候，感測器讀取到50，而當我有一半看到黑色，有一半看到白色的時候則會讀取到45，因為讀取進來的光線反射量恰好是白色反射和黑色反射的平均值。

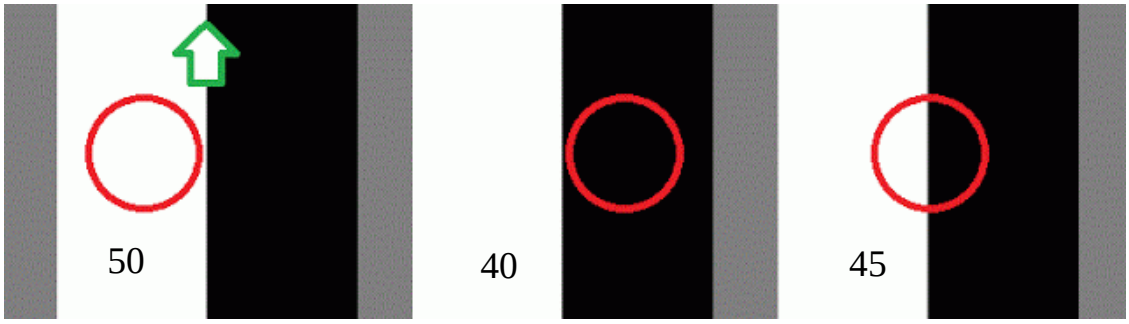


圖14、光線感測器示意圖

經過簡單的測量知道，當數字越接近40則表示車子越靠右黑線區域，當數字越大越接近50時，則表示車子越靠近左(白線區域)，而當車子偏左時，得下指令給車子向右邊走，反之亦然，這邊使用簡單的二分法來決定偏左或是偏右，就是剛剛的黑白平均值(50+40)/2 = 45，這個數值我們稱為offset，當紅外線小於45則我要讓車子左轉，大於的時候要右轉，如下圖15所示

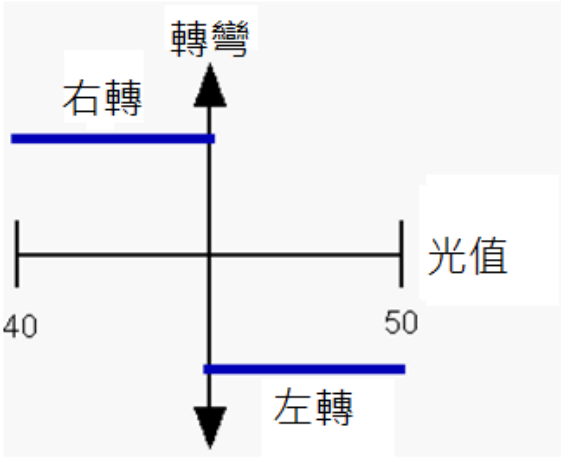


圖15、二分法循跡狀態圖

2. 比例控制循跡法(P.I.D)

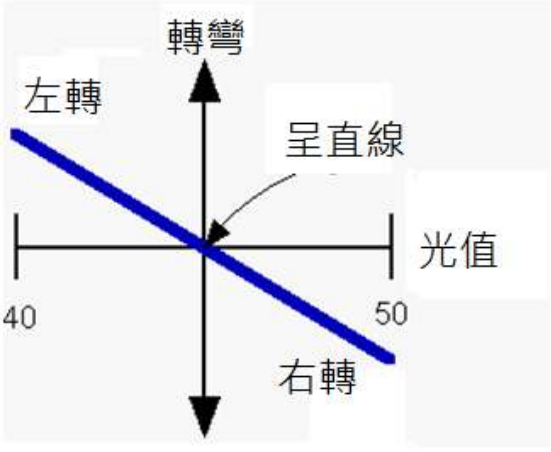


圖16、比例控制循跡狀態圖

比例控制循跡法，顧名思義就是按照光亮的強度，來決定轉彎的程度。讓車子保持行走的順暢，這方法可消弭一些外界變異或干擾影響，讓系統控制可預測，參數設定得好，就可以又快又穩又準，以上圖16為例我們希望車輛之轉彎是能夠很平滑的，上方圖16經過轉換後，呈現如下方圖17所示：

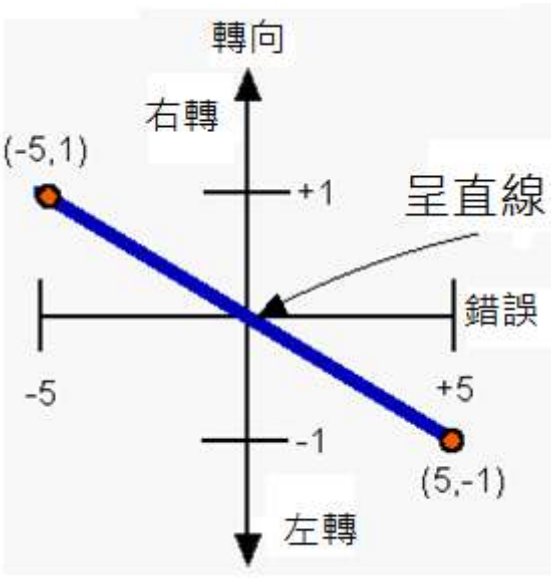


圖17、比例控制循跡狀態圖(轉換後)

我們將本來的40~50減去他們的平均值45，則變成-5～5，經轉換後我們將X軸的數值稱為ERROR，也就是誤差量，Y軸的部分將設定成轉速差百分比(100% = 1)，因此圖中直線之斜率為

$$m=(-1-1) / (5- (-5)) = -0.2$$

m是一個很重要的比例常數(proportionality constant)，之後將感測器所讀取到的的資料轉換成error之後就可以藉由這個比例常數將error直接轉換成Turn，這是一件很重要的事情，而這個m在PID控制理論稱之為K值，K這個數值是一個轉換值，可以讓輸入進來的數據轉換成馬達轉速的差異。在本研究中只使用PID法中的P控制，P指的就是線性範圍，在這邊的範例來說，紅外線感測器的proportional range 是40~50，而馬達我們是假設從-100~100。

伍、研究結果

一、 實驗初始化

在實驗開始前子走車需區分何者為黑、何者為白的判斷標準值，故操作者只需在程式開始前幾秒讓循跡車在黑色部分與白色部分來回滑動，透過掃描黑色與白色的光感值，再取其最大值(白)與最小值(黑)的中間值，即可將所算出的數值作為區分黑與白的標準。系統將重複 10000 次做五顆感測器資料紀錄，並計算出所記錄數值中的最大最小值並算出其中間值。

二、 實驗軌道

實驗軌道採 2014 國際奧林匹克樂高機器人競賽國中組場地，如下圖 18 所示，尺寸如下所示，實驗平台軌道寬度為 30 ㎜，主要以循黑色橢圓形為主，每次循跡一圈並記錄其時間與馬達之狀態。

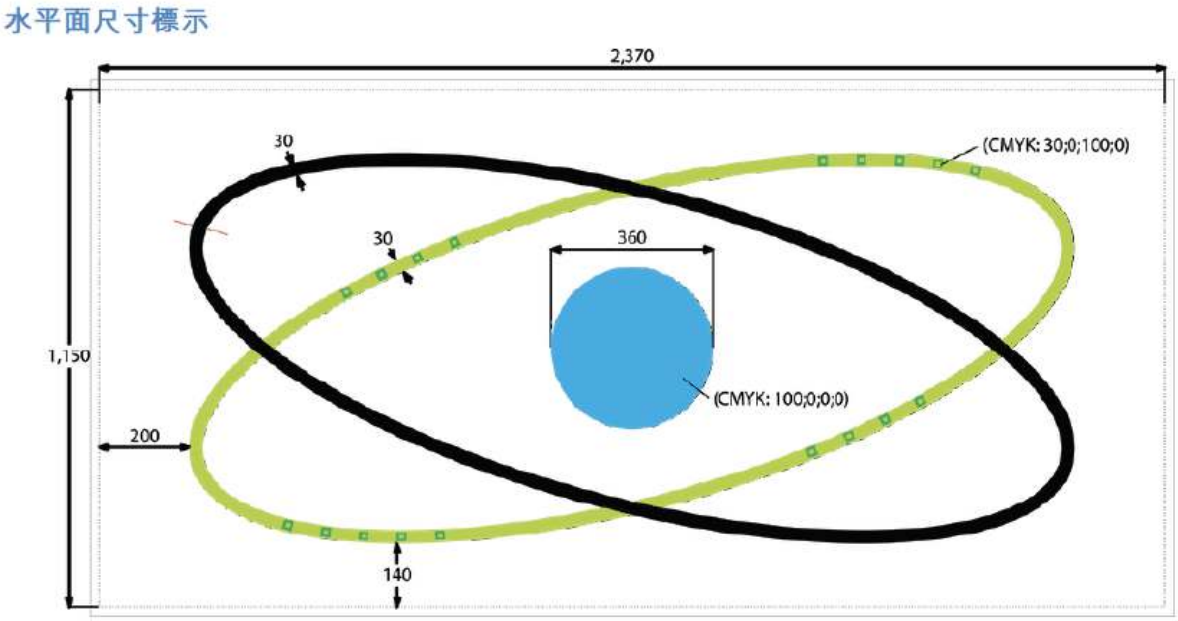


圖 18、實驗場地尺寸規格

三、 實驗結果

實驗一、單一光感實驗

單一光感實驗指的是透過單一感測器進行循跡，採用兩種演算法進行循跡，結果如下表 1、2 所示：

表1、 單一光感 Z 字型循跡法

基礎速度	誤差值	左輪平均轉速	右輪平均轉速	中間光感標準差	單圈秒數
80	40	74.60	85.40	8.80	19.252
80	60	79.10	80.90	8.30	20.957
80	80	67.52	92.50	8.20	未完成
90	40	78.01	102.00	9.00	未完成
90	60	70.79	109.21	5.48	17.520
90	80	65.89	114.00	8.50	未完成
110	40	112.28	108.00	7.90	未完成
110	60	100.00	120.00	9.40	未完成
110	80	93.91	126.00	8.90	未完成
130	40	134.13	126.00	8.80	未完成
130	60	121.95	138.00	8.60	未完成
130	80	114.33	146.00	8.30	未完成

表2、 單一光感比例控制循跡法(P.I.D)

基礎速度	K 值	左輪平均轉速	右輪平均轉速	中間光感標準差	單圈秒數
80	3	79.70	80.30	6.50	24.901
80	5	74.20	85.80	6.90	19.539
80	7	72.40	87.60	6.50	19.982
90	3	77.87	102.00	6.60	未完成
90	5	74.73	105.27	4.18	28.500
90	7	76.87	103.00	6.60	未完成
110	3	106.19	114.00	7.30	未完成
110	5	90.34	127.10	8.40	17.391
110	7	100.05	120.00	6.50	未完成
130	3	120.48	140.00	8.80	未完成
130	5	125.72	134.00	8.50	未完成
130	7	113.71	146.00	8.90	未完成

實驗小結：

1. 在單一光感情況下，Z 字型循跡法，基礎速度 80 誤差值 40、60，基礎速度 90 誤差值 60，有著較佳之結果，隨著速度拉高至 130 自走車容易衝出軌道。
2. 在單一光感情況下，Z 字型循跡法，隨著誤差值提高，自走車不穩定性也相對的提升。
3. 在單一光感情況下，比例控制循跡法，基礎速度 110，KP 值 5 的時候有著較佳之結果，當基礎速度拉高至 130 時，自走車容易衝出軌道。
4. 在單一光感情況下，比例控制循跡法中，K 值為 5 時相較於 3 與 7 有著較佳之結果。
5. 根據上方實驗可看出，比例控制循跡法，基礎速度 90，KP 值 5 完成一圈之秒數為 17.391，相較於 Z 字型跑法 17.50，有著較佳之結果，由此可見 Z 字型循跡法與 PID 法在單一光感情況底下，兩者之間並無太大之差異下。
6. 以穩定性來看，比例控制循跡法中間光感平均標準差為 8.35，相較於比例控制循跡法 7.14 要來的低，代表比例控制循跡法偏離跑道狀況較少，也相對較穩定。

實驗二、 三光感實驗

三光感實驗指的是透過三顆光感測器進行循跡，本實驗採用兩種演算法進行循跡，結果如下表 3、4 所示：

表3、 三光感 Z 字型循跡法

基礎速度	誤差值	左輪平均轉速	右輪平均轉速	中間光感標準差	單圈秒數
80	40	57.60	74.90	7.50	未完成
80	60	55.20	51.00	0.60	41.621
80	80	45.00	42.50	1.40	43.045
90	40	69.97	88.50	8.10	未完成
90	60	65.80	61.00	4.30	26.058
90	80	54.20	50.10	1.30	33.953
110	40	85.84	117.00	7.40	未完成
110	60	116.63	101.00	7.40	未完成
110	80	57.66	116.00	8.40	未完成
130	40	103.87	143.00	8.70	未完成
130	60	104.61	123.00	8.00	未完成
130	80	130.00	73.40	9.40	未完成

表4、 三光感比例控制循跡法(P.I.D)

基礎速度	K 值	左輪平均轉速	右輪平均轉速	中間光感標準差	單圈秒數
80	3	77.80	82.20	7.53	25.540
80	5	62.90	97.10	5.74	24.520
80	7	79.00	81.00	4.60	23.836
90	3	90.72	89.30	8.10	未完成
90	5	70.50	89.50	6.40	24.007
90	7	90.10	89.90	5.70	未完成
110	3	97.92	125.00	8.70	未完成
110	5	89.86	130.14	8.44	未完成
110	7	88.63	131.00	8.40	未完成
130	3	131.97	128.00	9.20	未完成
130	5	117.73	142.00	8.50	未完成
130	7	128.18	132.00	8.60	未完成

實驗小結：

1. 在三顆光感情況下，Z 字型循跡法自走車，基礎速度 90，誤差值 60 底下，有著較佳之實驗結果，但隨著基礎速度提升至 110 時，自走車容易衝出軌道。
2. 在三顆光感情況下，Z 字型循跡法自走車中誤差值為 60 時，均有著較佳之實驗結果，代表著誤差值 60 有著較佳之修正。
3. 比例控制循跡法中基礎速度為 80 時進行實驗，均能完整跑完一圈。基礎速度 90，KP 值 5，有著較佳之結果，當基礎速度拉高至 110 時，自走車容易衝出軌道。
4. 在三顆光感情況下比例控制循跡法中，相較於單光感實驗有著較佳之結果，隨著光感增加協助循跡，循跡也有著較佳之結果。
5. 比例控制循跡法之穩定性隨著基礎速度由 80 提升至 90 時，隨著速度提升穩定性相對的降低，結果顯示基礎速度 80，KP 值 7 時有較佳之結果。
6. 相較於單光感實驗中，隨著光感數量增加相較於單光感循跡，有著較佳之結果。

實驗三、五光感實驗

表5、 五光感 Z 字型循跡法

基礎速度	誤差值	左輪平均轉速	右輪平均轉速	中間光感標準差	單圈秒數
80	40	76.10	63.20	7.20	39.774
80	60	76.80	63.10	4.00	37.234
80	80	71.80	63.00	6.60	28.436
90	40	69.55	92.68	6.22	25.270
90	60	67.41	88.88	1.64	27.180
90	80	78.30	87.80	7.90	未完成
110	40	86.18	121.41	9.22	17.340
110	60	85.72	113.75	7.35	未完成
110	80	79.86	127.00	8.30	未完成
130	40	120.21	134.00	6.60	未完成
130	60	121.19	125.00	8.00	未完成
130	80	118.68	120.00	7.00	未完成

表6、 五光感比例控制循跡法(P.I.D)

基礎速度	K 值	左輪平均轉速	右輪平均轉速	中間光感標準差	單圈秒數
80	3	84.90	73.50	6.20	32.234
80	5	74.40	85.70	3.50	23.675
80	7	76.70	83.70	3.70	24.538
90	3	79.60	100.00	5.90	20.187
90	5	86.90	93.10	6.10	18.553
90	7	80.90	99.20	4.00	22.397
110	3	93.69	126.31	7.82	17.240
110	5	95.46	124.54	7.57	16.620
110	7	97.47	122.53	8.11	未完成
130	3	114.83	145.00	8.10	未完成
130	5	120.00	140.00	6.90	未完成
130	7	116.67	143.33	9.41	未完成

實驗小結：

1. 在 5 顆光感情況下，Z 字型循跡法，基礎速度 80 誤差值 40、60 與 80，基礎速度 90 誤差值 60 與 40、基礎速度 110 誤差值 40 能完成單圈，代表著隨著光感顆數的增加，容許的誤差範圍降低至 40，也能有著不錯的結果。
2. 在 5 顆光感情況下，Z 字型循跡法，基礎速度 80 時，誤差值範圍為 80 時穩定度相對比較下，有著較佳之結果。
3. 在 Z 字型循跡法下，隨著車速提升至 130 時自走車容易衝出軌道。
4. 在 Z 字型循跡法下速度 90 誤差值 60 時有著較佳之穩定度，中間光感標準差為 1.64。
5. 比例控制循跡法，基礎速度 80、90 均能完成單圈，Kp 值 5 的情況下有著較佳之速度，但與 Kp 值 3、7 差距不大。
6. 比例控制循跡法中，基礎速度 110，Kp 值為 3、5 情況下，有著較佳之結果，兩者間穩定度與速度差異不大，其中當 KP 值為 5 時有著最佳時間 16.620。

陸、討論

一、 對於 Z 字型跑法速與比例控制循跡法，隨著速度增加，所需之容許誤差與 KP 值也要相對拉高，否則自走車容易衝出跑道。

二、 Z 字型跑法中，當車速低，容許誤發範圍大時其穩定度也相對的提高。

三、 以下圖 19 為例，在比例控制循跡法中速度與 K 值拉高時，發現在未經大轉彎前穩定度比起經大轉彎後高出很多，所以當 K 值拉高時對不穩定之狀態的修復能力較差。

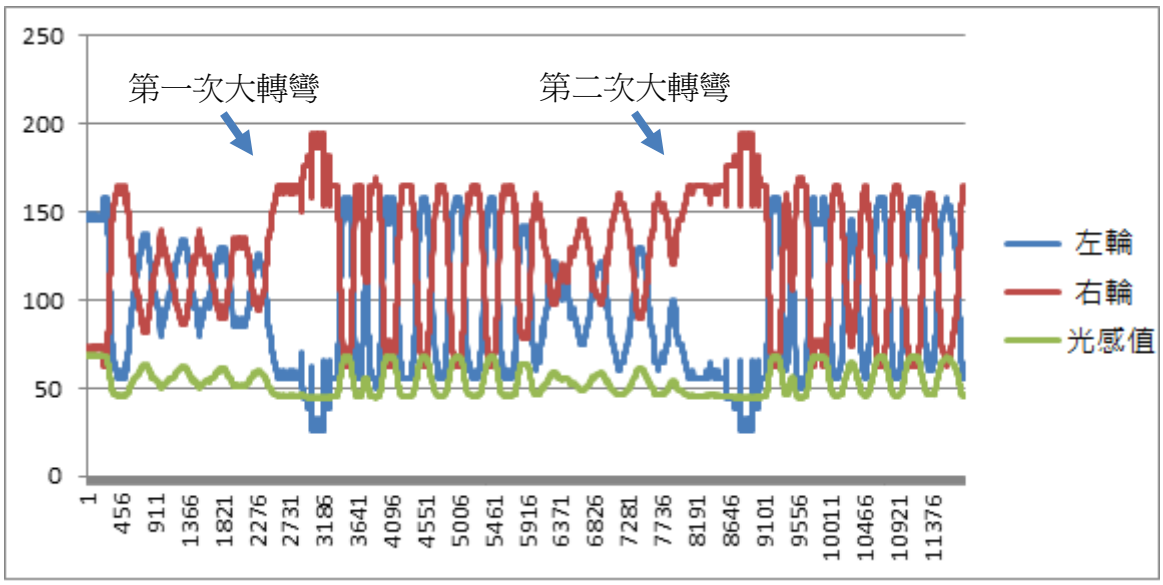


圖 19、5 光感 PID 循跡法 (速度 110，K 值 3)

四、 隨著光感的數目增加，對於 Z 字型跑法與比例控制循跡法能有著較佳之穩定性與速度。

五、 由實驗可發現使用 Z 字形跑法的穩定度相較於使用 PID 跑法還要高，是由於相較於 PID 法，Z 字形跑法在改變馬達轉速的頻率較低，提供一小段容許的範圍才讓馬達改變速率狀態，而不是 PID 只要小小的光感差異就使馬達速度即產生大幅改變(尤其是在高速與高誤差值 K 值的時候)。

六、 輸入的動態範圍要大點比較好，因為輸入的動態範圍就直接影響到 Error 的範圍，這將決定我們能夠有多好的辨識結果。

七、 輸出的動態範圍也得大一點，不然即使有了很大的輸入動態範圍，結果馬達卻只

能夠作正反轉的輸出，那麼就一切都免談了，又回到之前說的 2-Level 的模式，所以說輸入的動態範圍要盡可能的大，但是這兩者還是要能夠互相匹配才能夠互相發揮良好的作用。

八、 程式的所在地控制板運算速度，運作速度不能夠太慢，否則當車子超出邊界才進行修正，那也是白白浪費了 P 控制的強大效益了。

柒、參考資料及其他

[1] 江文川、陳貴一、張琮貞、蘇樂群、陳劍鴻，「自動導引車定位技術發展現況與展望」，技術學刊，pp.173-180(2007)。

[2] 許哲源，2003 年，《自走車之驅動控制與避障》，國立成大學工程科學系碩士論文。

[3] Android SDK 實戰演練，第 2 版，何孟翰作者悅知文化出版社。

[4] 謝宗翰、曾吉弘、侯俊宇(2009)。機器人新視界 NXC 與 NXT。台灣：藍海文化。

附件：

Z 字型跑法程式碼

```
#include <SoftwareSerial.h>
SoftwareSerial dataSerial(7, 8); // RX, TX
#define IClatchPin 12
#define ICclockPin 11
#define ICdataPin 13
#define sensor1 A0
#define sensor2 A1
#define sensor3 A2
#define sensor4 A3
#define sensor5 A4
#define leftMotorPin 6
#define rightMotorPin 5
#define modePin 2
int dataSensor1;
int dataSensor2;
int dataSensor3;
int dataSensor4;
int dataSensor5;

int triggerSensor1 = 0;
int triggerSensor2 = 0;
int triggerSensor3 = 0;
int trigge
rSensor4 = 0;
int triggerSensor5 = 0;

boolean
RGBmap[32]={1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1};

void arraySet(boolean value)
{
   for(int a=0;a<32;a++)
   {
      RGBmap[a]=value;
   }
}
void lightTest()
{
   arraySet(0);
   displayRGB();
   delay(200);
   arraySet(1);
   displayRGB();
   delay(200);
}
void updataTrigger()
{
   motor(0,0);
   lightTest();
   int sensor1Max=0;
   int sensor1Min=1000;
   int sensor2Max=0;
   int sensor2Min=1000;
   int sensor3Max=0;
   int sensor3Min=1000;
   int sensor4Max=0;
   int sensor4Min=1000;
   int sensor5Max=0;
   int sensor5Min=1000;
   for(int i=0;i<10000;i++)
   {
      updataSensor();
      if(dataSensor1>sensor1Max)
      sensor1Max = dataSensor1;
      if(dataSensor1<sensor1Min)
      sensor1Min = dataSensor1;

      if(dataSensor2>sensor2Max)
      sensor2Max = dataSensor2;
      if(dataSensor2<sensor2Min)
      sensor2Min = dataSensor2;

      if(dataSensor3>sensor3Max)
      sensor3Max = dataSensor3;
      if(dataSensor3<sensor3Min)
```

```
        sensor3Min = dataSensor3;

        if(dataSensor4>sensor4Max)
        sensor4Max = dataSensor4;
        if(dataSensor4<sensor4Min)
        sensor4Min = dataSensor4;

        if(dataSensor5>sensor5Max)
        sensor5Max = dataSensor5;
        if(dataSensor5<sensor5Min)
        sensor5Min = dataSensor5;
    }
    triggerSensor1=(sensor1Max+sensor1Min)/2;
    triggerSensor2=(sensor2Max+sensor2Min)/2;
    triggerSensor3=(sensor3Max+sensor3Min)/2;
    triggerSensor4=(sensor4Max+sensor4Min)/2;
    triggerSensor5=(sensor5Max+sensor5Min)/2;
    lightTest();
}

void updataSensor()
{
    dataSensor1 = analogRead(sensor1);
    dataSensor2 = analogRead(sensor2);
    dataSensor3 = analogRead(sensor3);
    dataSensor4 = analogRead(sensor4);
    dataSensor5 = analogRead(sensor5);
}
void displayRGB()
{
        digitalWrite(IClatchPin, LOW);
        for (int b=0;b<32;b++)
        {
                digitalWrite(ICdataPin, RGBmap[b]);
                digitalWrite(ICclockPin, HIGH);
                digitalWrite(ICclockPin, LOW);
        }
        digitalWrite(IClatchPin, HIGH);
}

void motor(int leftMotor,int rightMotor)
{
    analogWrite(leftMotorPin,leftMotor);
    analogWrite(rightMotorPin,rightMotor);
}

void setup()
{
    pinMode(IClatchPin,OUTPUT);
    pinMode(ICclockPin,OUTPUT);
    pinMode(ICdataPin,OUTPUT);
    pinMode(leftMotorPin,OUTPUT);
    pinMode(rightMotorPin,OUTPUT);
    pinMode(sensor1,INPUT);
    pinMode(sensor2,INPUT);
    pinMode(sensor3,INPUT);
    pinMode(sensor4,INPUT);
    pinMode(sensor5,INPUT);
    pinMode(modePin,INPUT);
    lightTest();
    updataTrigger();
    dataSerial.begin(115200);

void loop()
{
    double basicSpeed=70;
    double wrong=30;
    while(1)
    {
        updataSensor();
        if(dataSensor1<triggerSensor1)
        {
         motor(basicSpeed-wrong,basicSpeed+wrong);
        }
        else
        {
         motor(basicSpeed+wrong,basicSpeed-wrong);
        }
    }
}
```

PID跑法程式碼

```
#include <SoftwareSerial.h>
SoftwareSerial dataSerial(7, 8); // RX, TX
#define IClatchPin 12
#define ICclockPin 11
#define ICdataPin 13
#define sensor1 A0
#define sensor2 A1
#define sensor3 A2
#define sensor4 A3
#define sensor5 A4
#define leftMotorPin 6
#define rightMotorPin 5
#define modePin 2
int dataSensor1;
int dataSensor2;
int dataSensor3;
int dataSensor4;
int dataSensor5;

int triggerSensor1 = 0;
int triggerSensor2 = 0;
int triggerSensor3 = 0;
int triggerSensor4 = 0;
int triggerSensor5 = 0;

boolean
RGBmap[32]={1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1};

void arraySet(boolean value)
{
  for(int a=0;a<32;a++)
  {
    RGBmap[a]=value;
  }
}

void lightTest()
{
  arraySet(0);
  displayRGB();
  delay(200);
  arraySet(1);
  displayRGB();
  delay(200);
}

void updataTrigger()
{
  motor(0,0);
  lightTest();
  int sensor1Max=0;
  int sensor1Min=1000;
  int sensor2Max=0;
  int sensor2Min=1000;
  int sensor3Max=0;
  int sensor3Min=1000;
  int sensor4Max=0;
  int sensor4Min=1000;
  int sensor5Max=0;
  int sensor5Min=1000;

  for(int i=0;i<10000;i++)
  {
    updataSensor();
    if(dataSensor1>sensor1Max)
    sensor1Max = dataSensor1;
    if(dataSensor1<sensor1Min)
    sensor1Min = dataSensor1;

    if(dataSensor2>sensor2Max)
    sensor2Max = dataSensor2;
    if(dataSensor2<sensor2Min)
    sensor2Min = dataSensor2;

    if(dataSensor3>sensor3Max)
    sensor3Max = dataSensor3;
    if(dataSensor3<sensor3Min)
```

```
        sensor3Min = dataSensor3;

        if(dataSensor4>sensor4Max)
        sensor4Max = dataSensor4;
        if(dataSensor4<sensor4Min)
        sensor4Min = dataSensor4;

        if(dataSensor5>sensor5Max)
        sensor5Max = dataSensor5;
        if(dataSensor5<sensor5Min)
        sensor5Min = dataSensor5;
    }
    triggerSensor1=(sensor1Max+sensor1Min)/2;
    triggerSensor2=(sensor2Max+sensor2Min)/2;
    triggerSensor3=(sensor3Max+sensor3Min)/2;
    triggerSensor4=(sensor4Max+sensor4Min)/2;
    triggerSensor5=(sensor5Max+sensor5Min)/2;
    lightTest();
}

void updataSensor()
{
    dataSensor1 = analogRead(sensor1);
    dataSensor2 = analogRead(sensor2);
    dataSensor3 = analogRead(sensor3);
    dataSensor4 = analogRead(sensor4);
    dataSensor5 = analogRead(sensor5);
}

void displayRGB()
{
        digitalWrite(IClatchPin, LOW);
        for (int b=0;b<32;b++)
        {
                digitalWrite(ICdataPin, RGBmap[b]);
                digitalWrite(ICclockPin, HIGH);
                digitalWrite(ICclockPin, LOW);
        }
        digitalWrite(IClatchPin, HIGH);
}

void motor(int leftMotor,int rightMotor)
{
    analogWrite(leftMotorPin,leftMotor);
    analogWrite(rightMotorPin,rightMotor);
}

void setup()
{
    pinMode(IClatchPin,OUTPUT);
    pinMode(ICclockPin,OUTPUT);
    pinMode(ICdataPin,OUTPUT);
    pinMode(leftMotorPin,OUTPUT);
    pinMode(rightMotorPin,OUTPUT);
    pinMode(sensor1,INPUT);
    pinMode(sensor2,INPUT);
    pinMode(sensor3,INPUT);
    pinMode(sensor4,INPUT);
    pinMode(sensor5,INPUT);
    pinMode(modePin,INPUT);
    lightTest();
    updataTrigger();
    dataSerial.begin(115200);
}

void loop()
{
    double basicSpeed=70;
    double Kp=4;
    while(1)
    {
        //updataSensor();
        double
error=analogRead(sensor3)-triggerSensor3;
        double turn=error*Kp;
        double A=basicSpeed-turn;
        double B=basicSpeed+turn;
        motor((int)A,(int)B);
    }
  }
```

【評語】030805

利用光感測器在自走車的訊號，自動校正行進方向，能在曲線軌跡上行走程式控制利用 Arduino 系統語言撰寫決策邏輯。系統整合完整能比較不同速度下的表現，可增加不同軌跡型態如直角轉變或叉路等的控制以增加實用性應用。