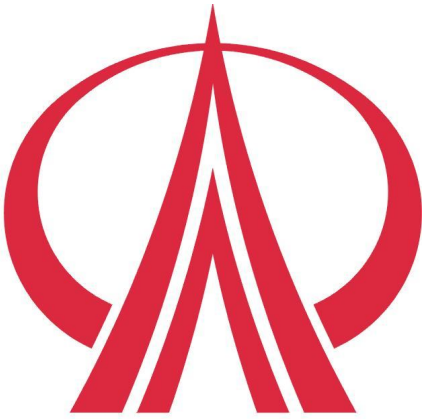
修平技術學院
電機工程系

DEPARTMENT OF ELECTRICAL ENGINEERING

HSIU-PING INSTITUTE OF TECHNOLOGY

實務專題報告書

無人競速自走車之PID控制



指導老師： 蔣忠誠 老師

專題製作學生:四技電四乙 黃議霈 BD96087

四技電四乙 張竣銘 BD96085

中華民國 100 年 08 月 01 號

摘要

本專題是　利用利基 Innoracer™ 以 BC2(BASIC Commander®2)為核心，控制兩個內建模組，Racer M1 與 Racer P1,和利用7 個軌道偵測固定紅外線感測器,和左右有兩個可移動紅外線感測器，可以偵測軌道外圍的軌道變化資訊，也可以隨時更改位置，配合不同軌道變化應用。配置一組蜂鳴器，能由軟體設定產生提示音的時機，也可以透過指令，自動於經過軌道記號時，產生記錄提醒。內建馬達控制模組(Racer M1)，透過指令可以控制兩組馬達轉速，並且能做出1024 段速差變化。提供 PID 設定值，可以直接更改參數，控制循跡效能。

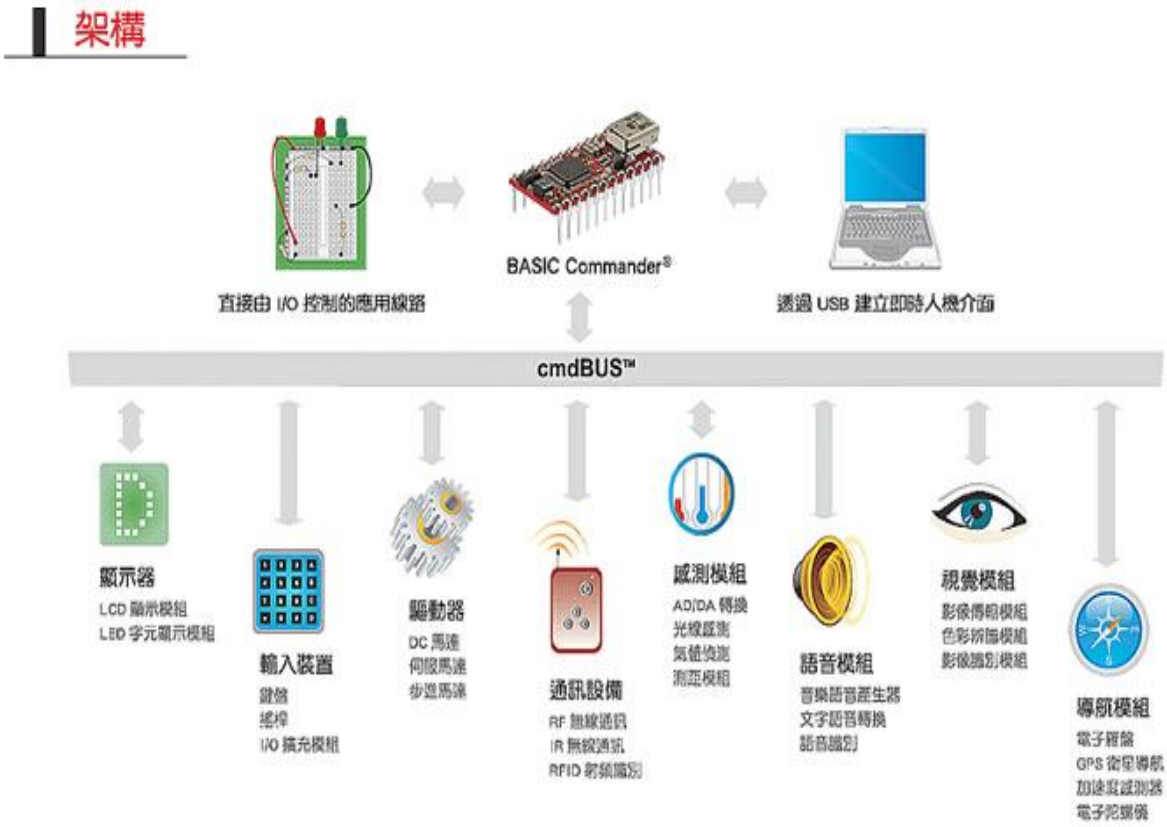
另外提供多組固定孔，可以配合需求更換馬達與輪胎位置，達到縮小軸距或加大輪距等應用，在必須通過較小曲率半徑彎道時，能發揮極大效果。讓使用者可以快速達到循跡控制，更可以記錄軌道資訊，　於不同路段行駛不同速度，讓使用者可以不斷挑戰，不同賽道的極限速度，完成競速的目的。

目錄

第一章、前言

1-1 簡介

InnoRacer 簡介:1. BC2 核心
2. USB 程式下載
3. BASIC 語言及函數指令控制
4. 提供數位及類比 PID 控制
5. 具馬達轉速回饋及過載保護功能
6. 跑道記憶功能
7. 內建 G-Sensor, 提供彎道曲率計算
8. 額外 cmdbus 模組擴充功能
9. 適合教育學習及比賽用車

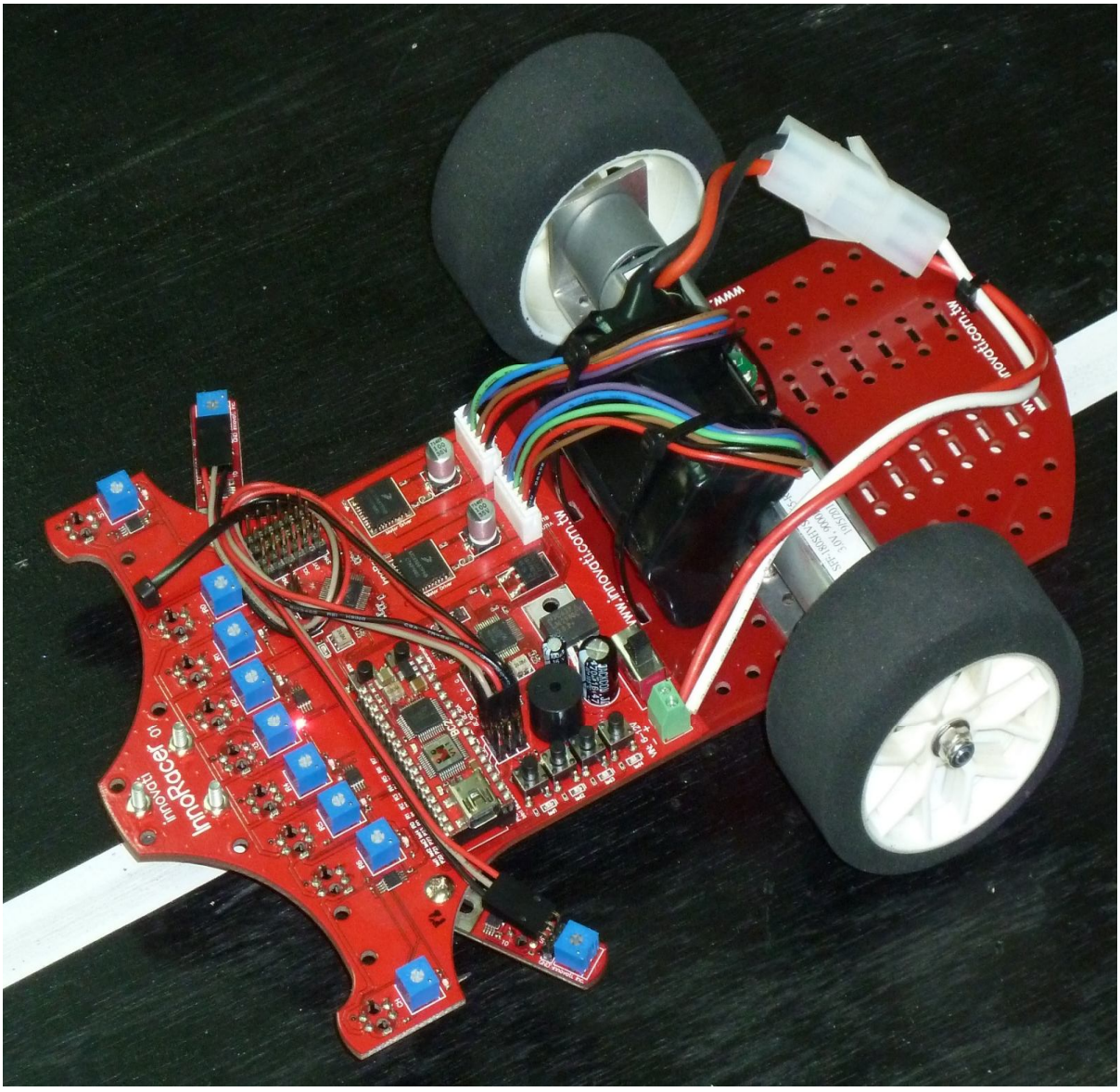


1-2 研究動機

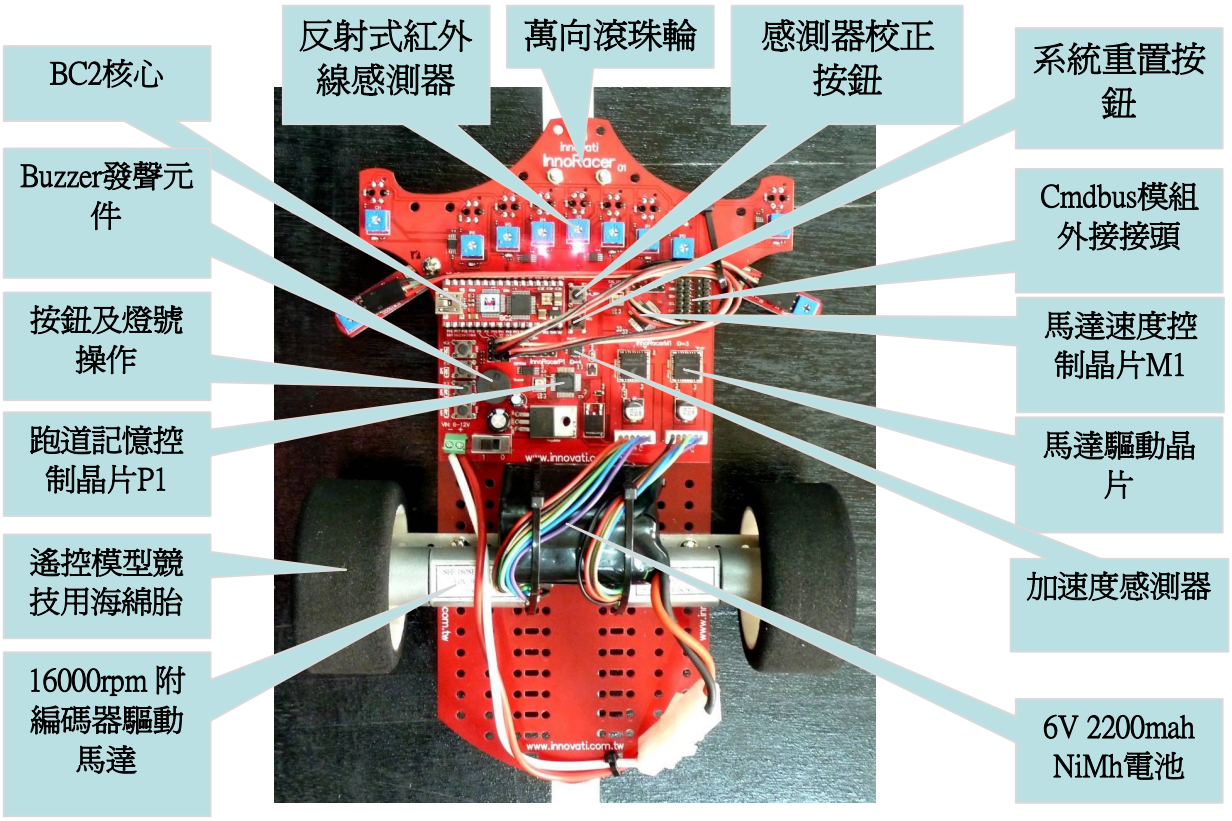
剛開始接觸到是有一次學校有辦一個活動,請到利基的總經理來演講,聽了之後覺這個東西很新奇也很好玩的樣子,-而且都模組化了很方便,甚至連程式都有了,應該不會很難吧!直到我們要去參加循跡競速自走車的比賽感覺都不一樣了,一開始拿到車,它可以跑卻跑不完全程,面對沒看過的程式也摸不著頭緒,有一天我們約了利基的工程師,到利基的公司請教他,他花了一天的時間交我們該如何下手,遇到什麼問題該如何處理,我們也漸漸比較瞭解 innoBASIC 了!
後來回到學校,我們開始改裡面的程式、調整速度、調整參數、改裝車體的架構和一直不斷的去測試,我們的車慢慢的可以跑完全程,速度也越來越快了!

第二章、介紹元件說明

2-1 自走車車型



自走車各位置之說明


BC2核心
反射式紅外線感測器
萬向滾珠輪
感測器校正按鈕
系統重置按鈕
Buzzer發聲元件
Cmdbus模組外接接頭
按鈕及燈號操作
馬達速度控制晶片M1
跑道記憶控制晶片P1
馬達驅動晶片
遙控模型競技用海綿胎
加速度感測器
16000rpm 附編碼器驅動馬達
6V 2200mah NiMh電池

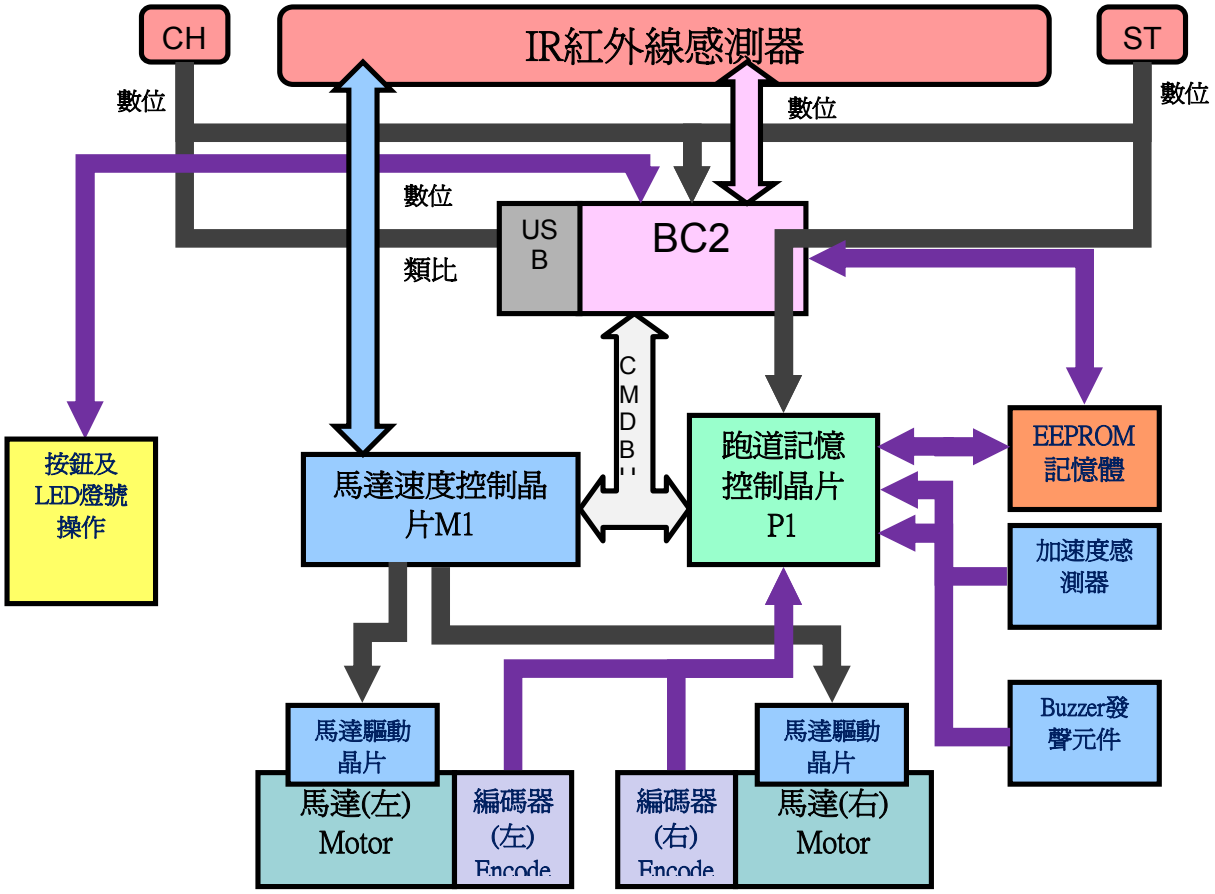
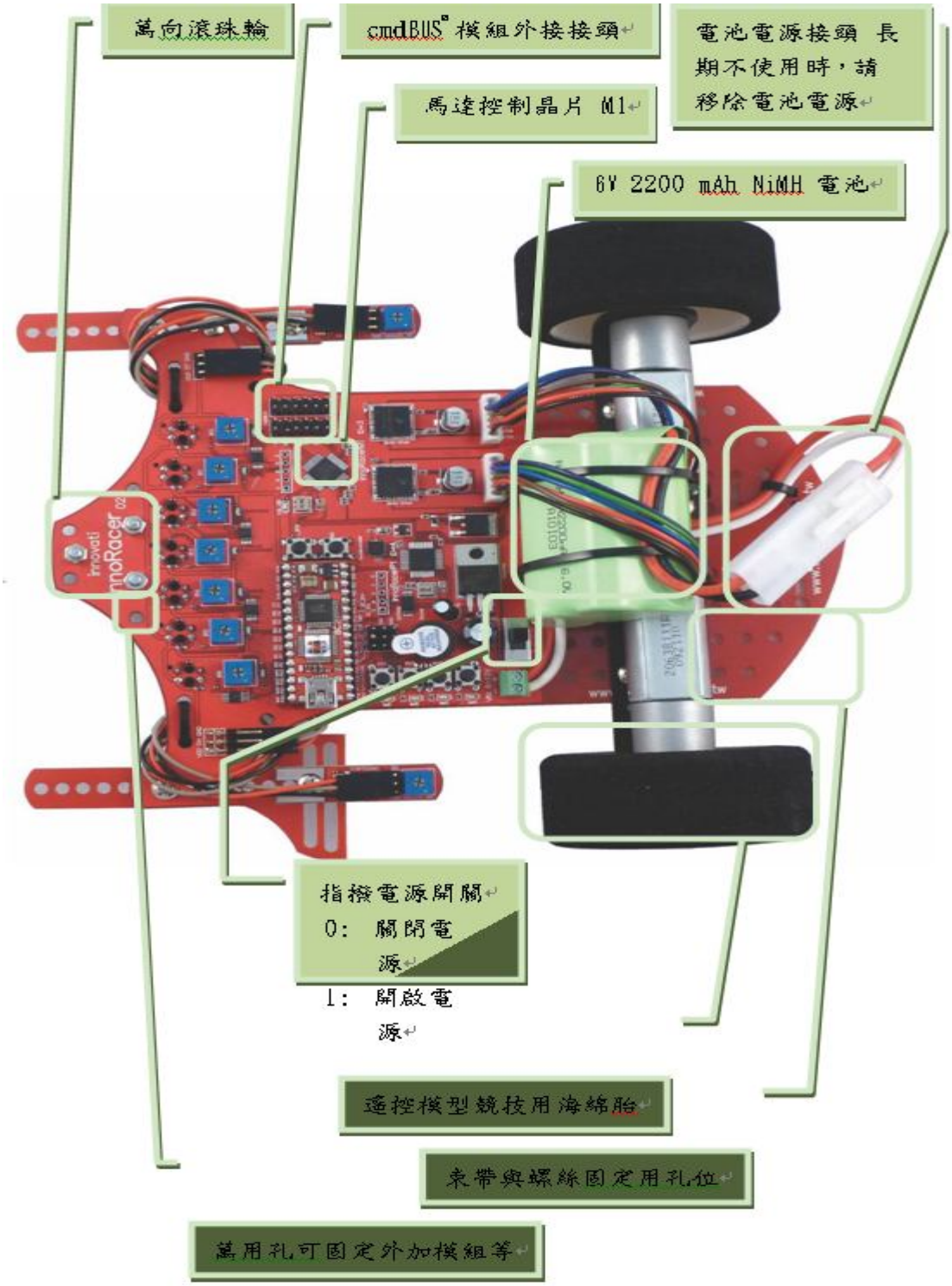
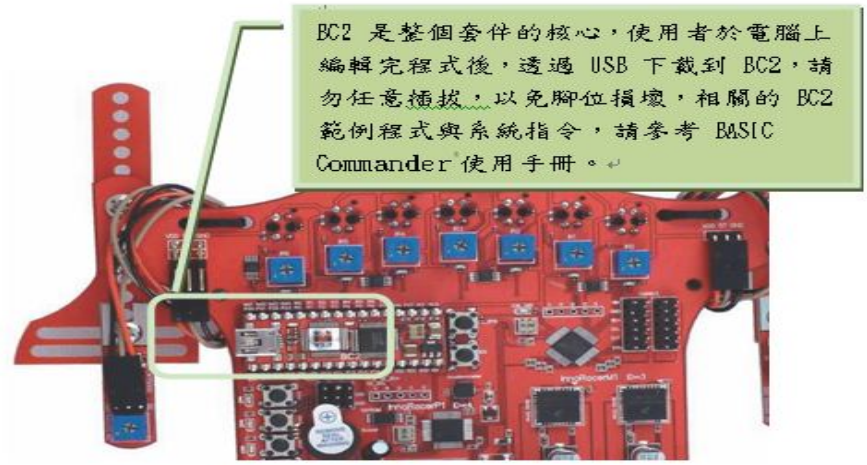

CH
IR紅外線感測器
ST
數位
數位
數位
數位
數位
USB
BC2
類比
CMDB
按鈕及LED燈號操作
馬達速度控制晶片M1
跑道記憶控制晶片P1
EEPROM記憶體
加速度感測器
Buzzer發聲元件
馬達驅動晶片
馬達驅動晶片
馬達(左)Motor
編碼器(左)Encode
編碼器(右)Encode
馬達(右)Motor

圖 1: 系統架構

2-2 基本固定元件


萬向滾珠輪
cmdBUS® 模組外接接頭
電池電源接頭 長期不使用時，請移除電池電源
馬達控制晶片 M1
6V 2200 mAh NiMH 電池
指撥電源開關
0: 關閉電源
1: 開啟電源
遙控模型競技用海綿胎
束帶與螺絲固定用孔位
萬用孔可固定外加模組等

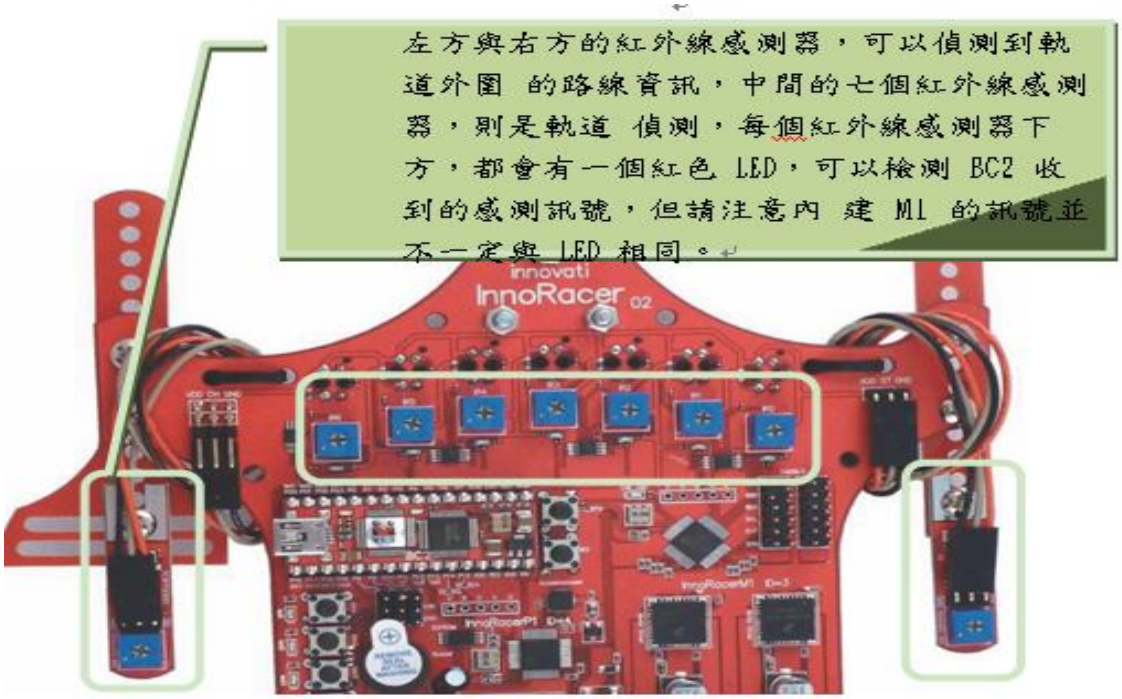
2-3 BC2



測試程式

```
'===說明：BC2 功能測試，會清除終端視窗原有訊息，再顯示"Hello"文字
Sub Main()
    Debug CLS,
  "Hello"  End Sub
```

2-4紅外線感測元件



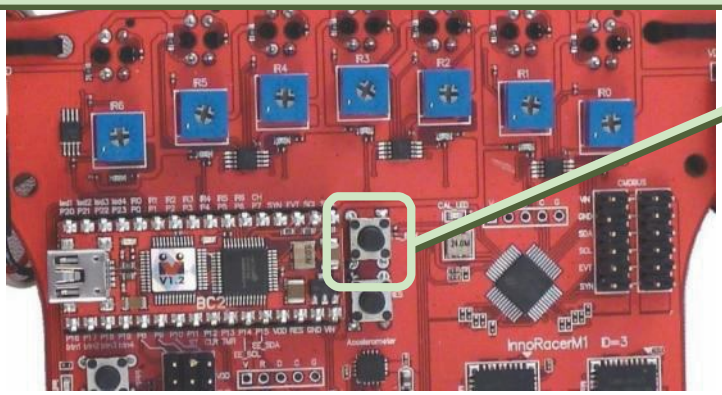
測試程式

```
===說明: 反覆偵測紅外線感測值，並顯示在終端視窗=======
Sub Main()
    Dim bIR, ST, CH As Byte          ’  儲存紅外線感測值

    Debug CLS     ’  清除終端視窗文字
    Debug "紅外線感測值: ", CR  ’ 輸出文字訊息到終端視窗
    Debug "    ST 感測值: ", CR  ’ 輸出文字訊息到終端視窗
    Debug "    CH 感測值: "      ’ 輸出文字訊息到終端視窗

    ’無窮迴圈，反覆偵測紅外線感測值
    Do
        bIR = Readport(0)          ’  讀取紅外線感測器回傳值
        bIR = bIR And &HF7
        ST  =  in(11)
        CH = in(7)
        Debug CSRXY(15, 1), %BIN bIR ’   以二進制顯示感測值
        Debug CSRXY(15, 2), %BIN ST
        Debug CSRXY(15, 3), %BIN CH
    Loop
```

2-5紅外線校正按鈕

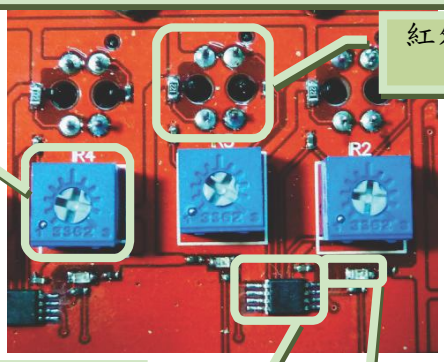
校正程序方法為:

1. 按下 CAL_BTN 按鈕一直到紅色 CAL_LED 亮起 (約按住 5 秒)
2. 此時將 InnoRacer™ 平放在競速場地中 並對齊跑道白線
3. 並以馬達軸線與白線的交插點為圓心將車子旋轉
4. 以一定速度緩慢的將所有 IR 紅外線感測器來回劃過白線部份
5. 完成以上動作之後再按下 CAL_BTN 按鈕以結束校正



2-6紅外線感度旋鈕

用十字起旋轉調整感度
逆時針旋轉感度增加 順時針旋轉感度減少 請注意對軌道偵測紅外線感測器，只會影響 BC2 從 IO 埠讀取的 感測值，不會影響 M1 的感測值。



紅外線感測器背面位置

IR 感測電壓感度調整晶片

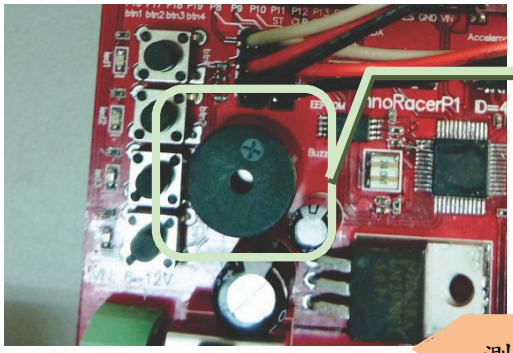
紅外線感測指示燈
有反射訊號時亮紅燈，沒收到反射訊號則熄滅

紅外線感測器接收訊號端

紅外線感測器正面

紅外線感測器發射訊號端

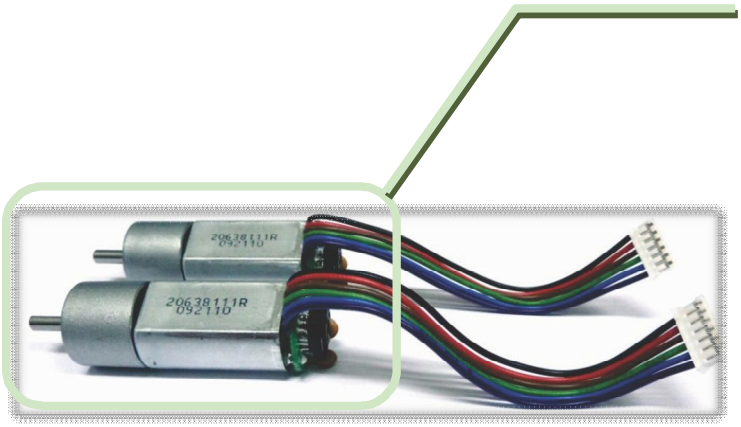
2-7蜂鳴器



測試程式

```
'===說明: 每隔一秒讓蜂鳴器發出提示音===
Peripheral myP1 As RacerP1 @ 4

Sub Main()
    Do
        myP1.Beep()
        Pause 1000
    Loop
End Sub
```

2-8直流馬達



16000rpm 附編碼器驅動馬達 操作電壓 6V 齒輪比 1:13 連接線依序為
黑: 馬達連接線 紅: 馬達連接線
棕: 編碼電路 5V 電源 綠: 編碼電路 GND
藍: 編碼器輸出 1
紫: 編碼器輸出 2

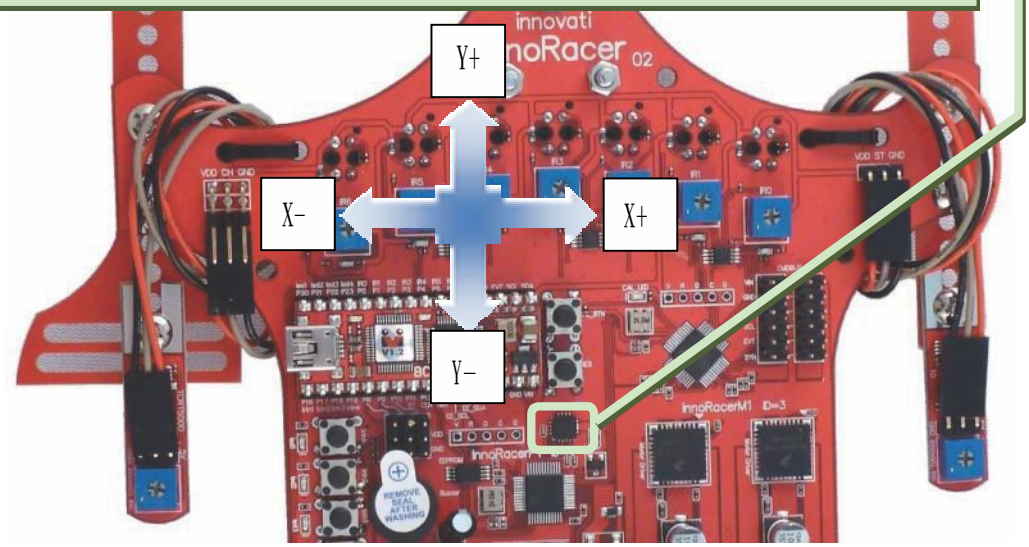
此直流有刷馬達有固定運轉壽命，但在長時間高速運轉，或是多次快速切換轉向，都會加速縮短馬達壽命。

測試程式

```
'說明:  根據輸入值操作馬達動作，並設定停止方式
'注意:  馬達轉動會讓自走車移動，請先墊高自走車讓輪胎離地，再開始
'       測試
Peripheral myM As RacerM1 @ 3  '  設定馬達控制模組參數名稱

Sub Main()
    Dim bKey As Byte          '  儲存輸入值
    Dim iVelL, iVelR As Integer  '  左右輪控制參數

    '無窮迴圈，可以反覆輸入馬達控制參數
    Do
        Debug CLS             '  清除終端視窗顯示文字
        Debugin "請輸入左輪參數(-1024~1024): ",
        iVelL Debug iVelL, CR
        Debugin "請輸入右輪參數(-1024~1024): ", iVelR
        Debug iVelR, CR

        myM.SetVelAB(iVelL, iVelR)'  設定左右輪轉速與方向

        Debugin "請決定停止方式(0: Stop, 1: Brake): ", bKey
        Debug bKey, CR

        If bKey=0 Then
            myM.StopDual()  '  以 Stop 方式停止兩輪轉動
        Else
            myM.BrakeDual()  '  以 Brake 方式停止兩輪轉動
        End If

        Keyin "按下任意鍵繼續", %CHR bKey
    LOOP
End Sub
```

2-9加速度感測器

圖中綠框所標示的元件，就是加速度感測晶片，可以量測 X 軸向與 Y 軸向的加速度，X 與 Y 軸方向如圖。 加速度感測晶片的校正，以及加速度值的讀取，都必須透過 Racer P1 的指令做控制，回傳值為電壓轉換過後的數位值。

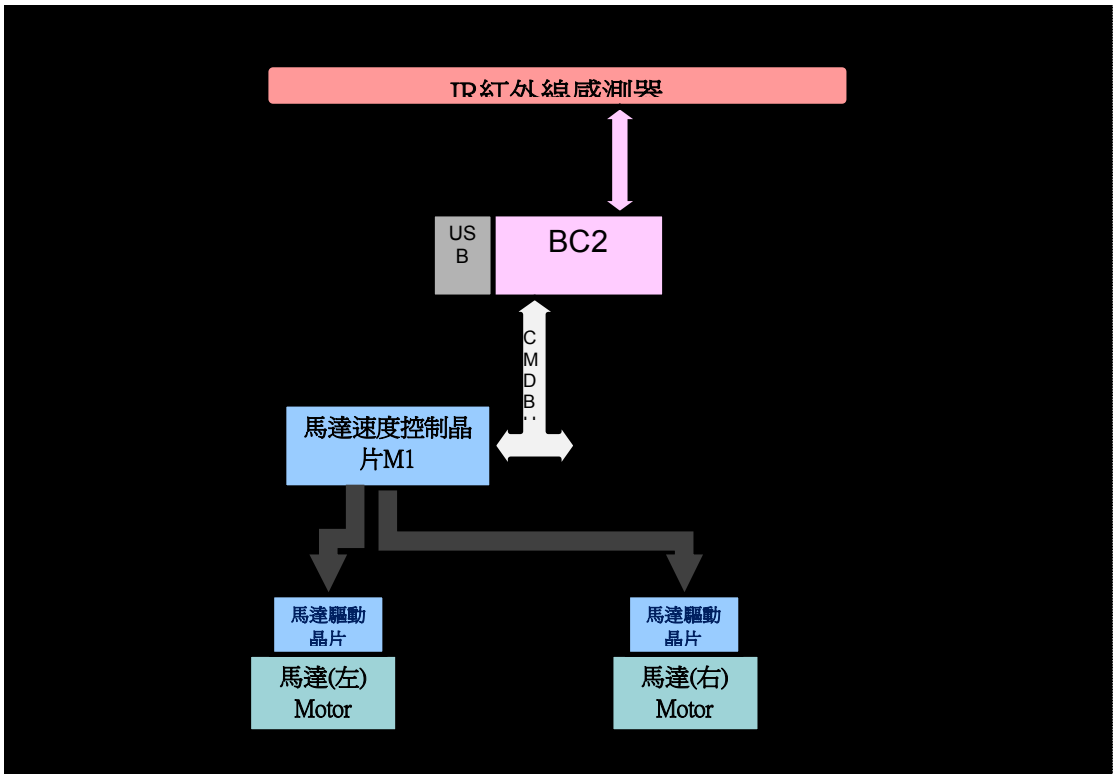


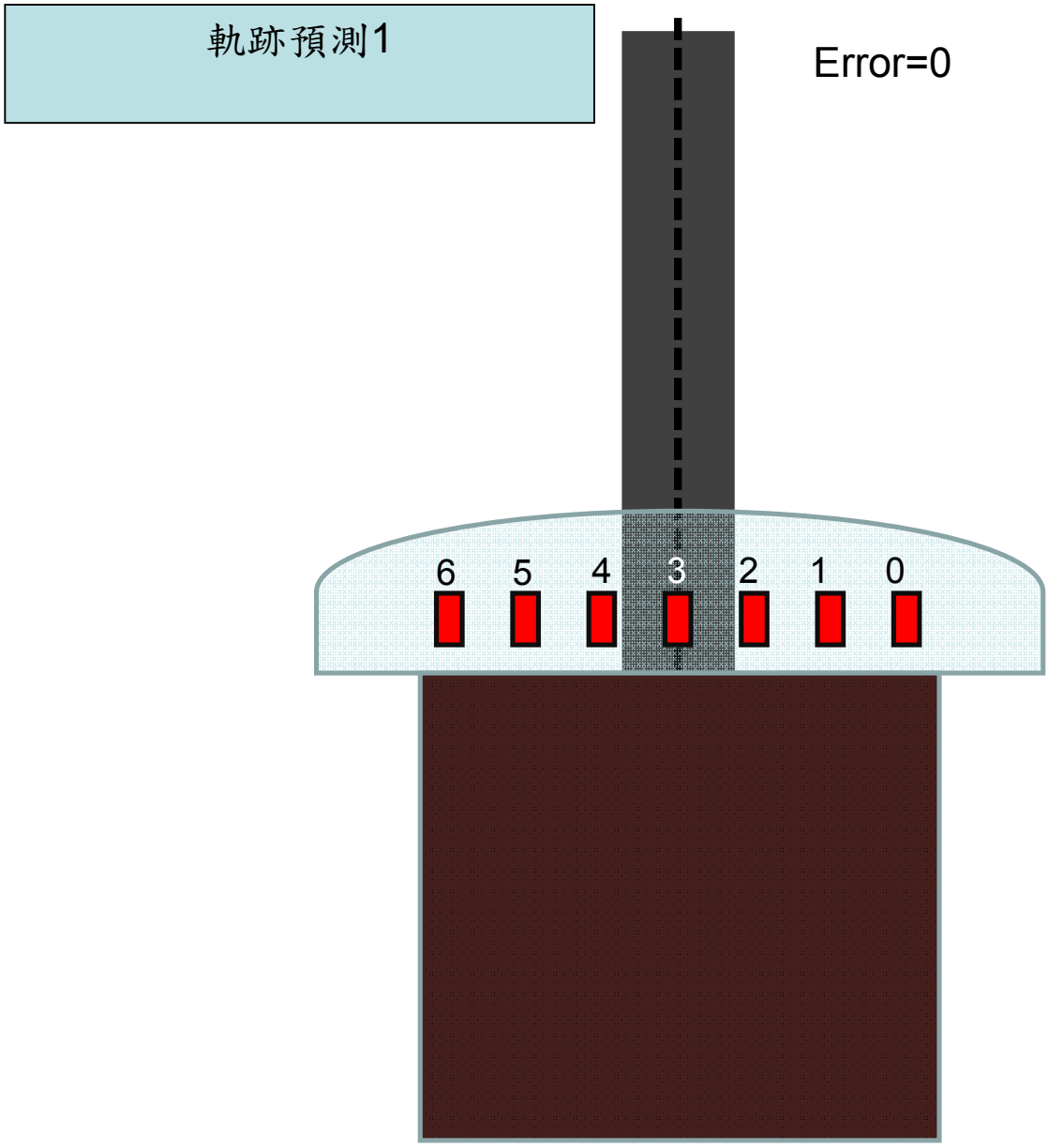
校正程式

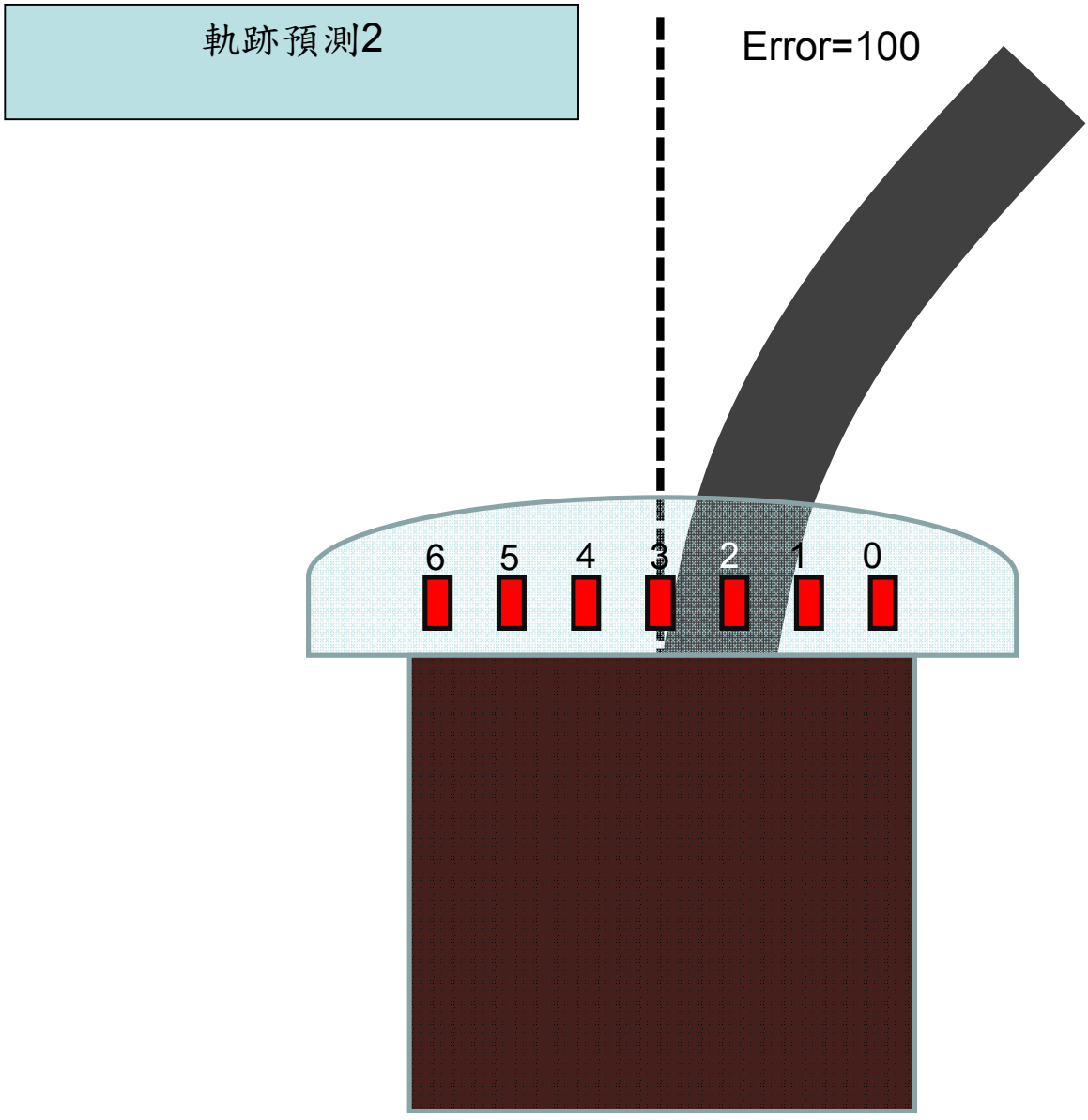
```
===說明: 重新紀錄現有加速度值為校正值===
Peripheral myP1 As RacerP1 @ 4

Sub Main()
    Dim  iX, iY As Integer

    Debug CLS
    Debug CSRXY(1,1), "更新加速度校正值"

    myP1.SaveCur0G()      '  將現在讀取加速度值，儲存為校正值
    myP1.Load0G(iX, iY)   '  取得加速度校正值
    Debug CSRXY(1,2), "X: ", CSRXY(4, 2), %DEC5R iX'  顯示 X 軸向校正值
    Debug CSRXY(1,3), "Y: ", CSRXY(4, 3), %DEC5R  iY'顯示 Y 軸向校正值
End Sub
```

第三章自走車控制理論介紹 PD 控制

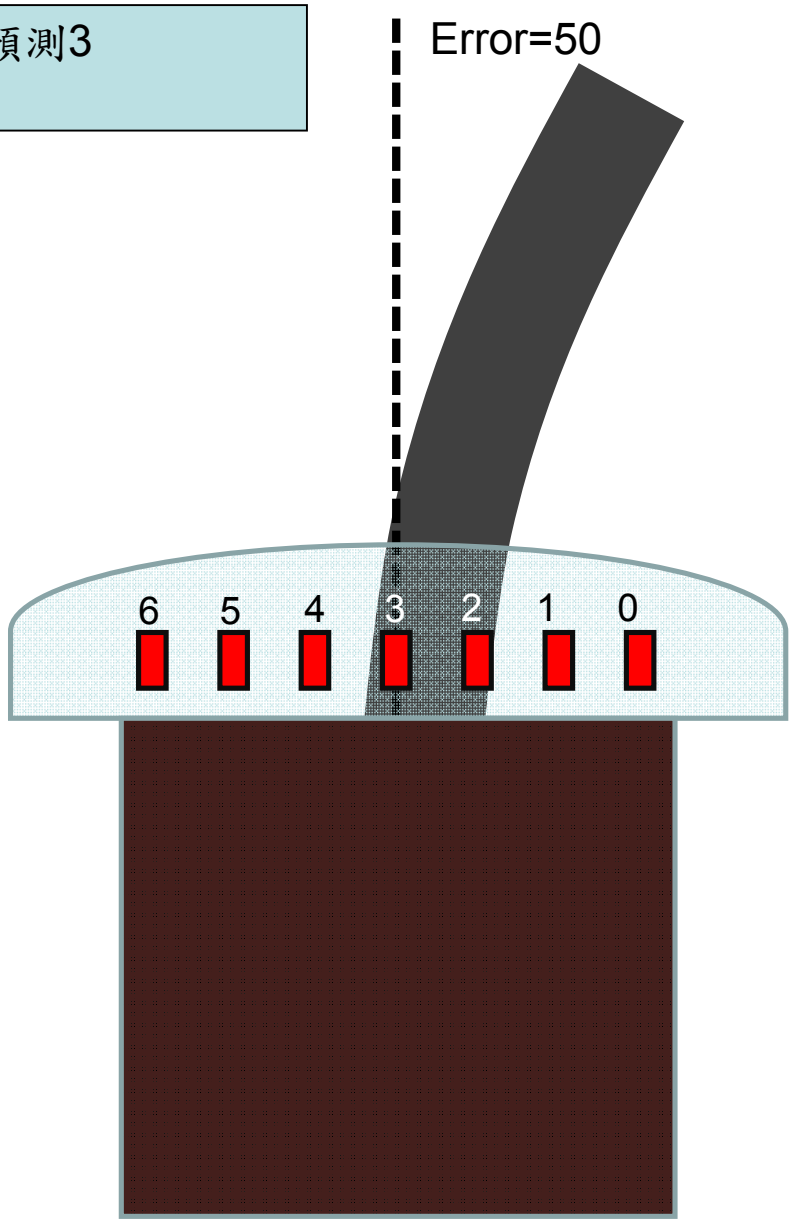
3-1 自走車控制理論介紹




軌跡預測1
Error=0
6
5
4
3
2
1
0


軌跡預測2
Error=100
6
5
4
3
2
1
0

軌跡預測3



3-2 數位回授控制

比例微分控制法則(PD):

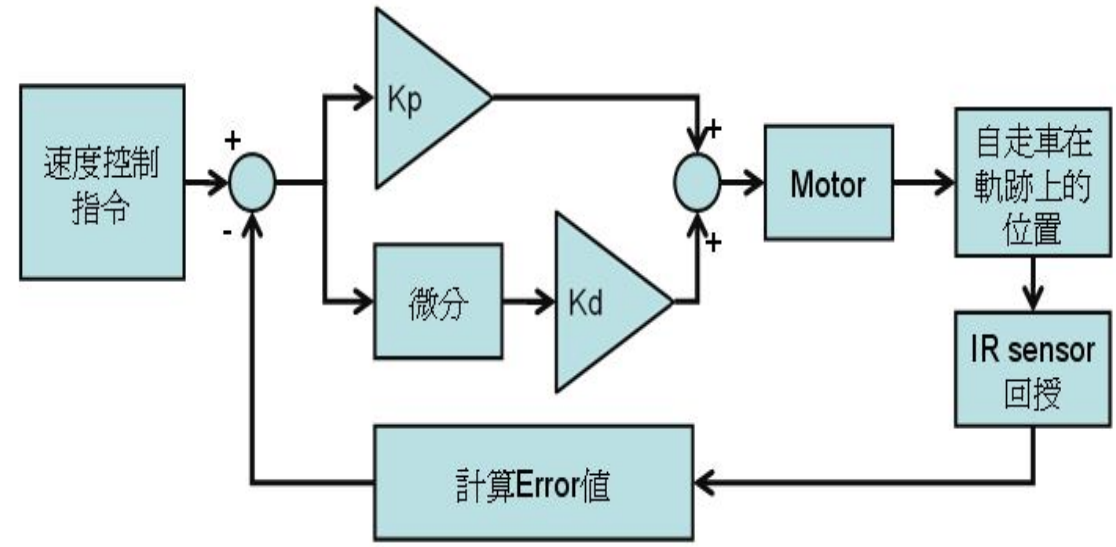
◆根據誤差值(Error)來調整左右輪的速度差

◆若是軌跡在中心軸左邊，誤差爲負值，比例控制根據誤差量調整右輪速度大於左輪速度；
反之亦然

◆若誤差持續擴大，微分控制可以加大速差

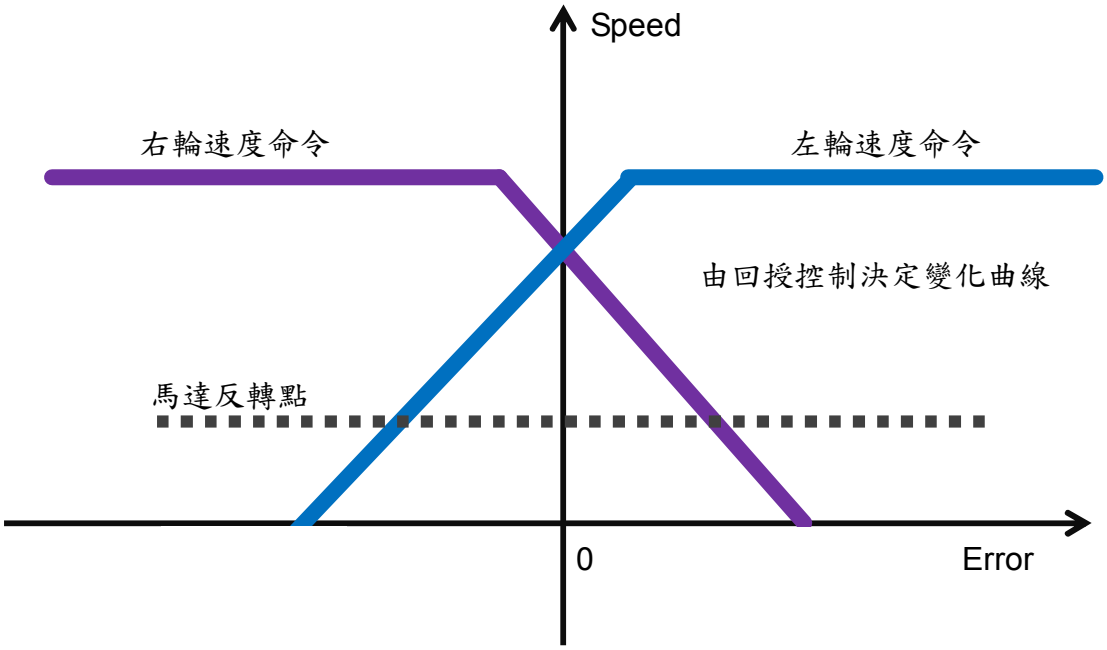
數位回授控制

$$Sd(n) = Kp*e(n) + Kd[e(n)-e(n-1)]$$
$$S(n)=S\_normal-Sd(n)$$



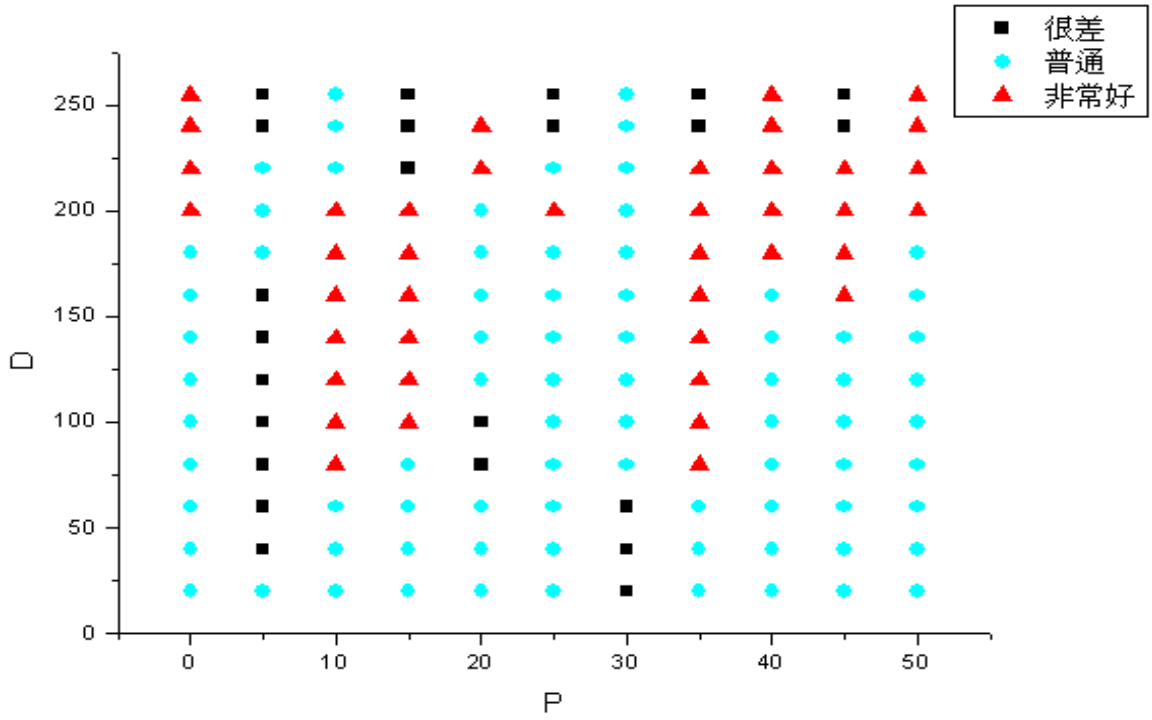
數位回授控制

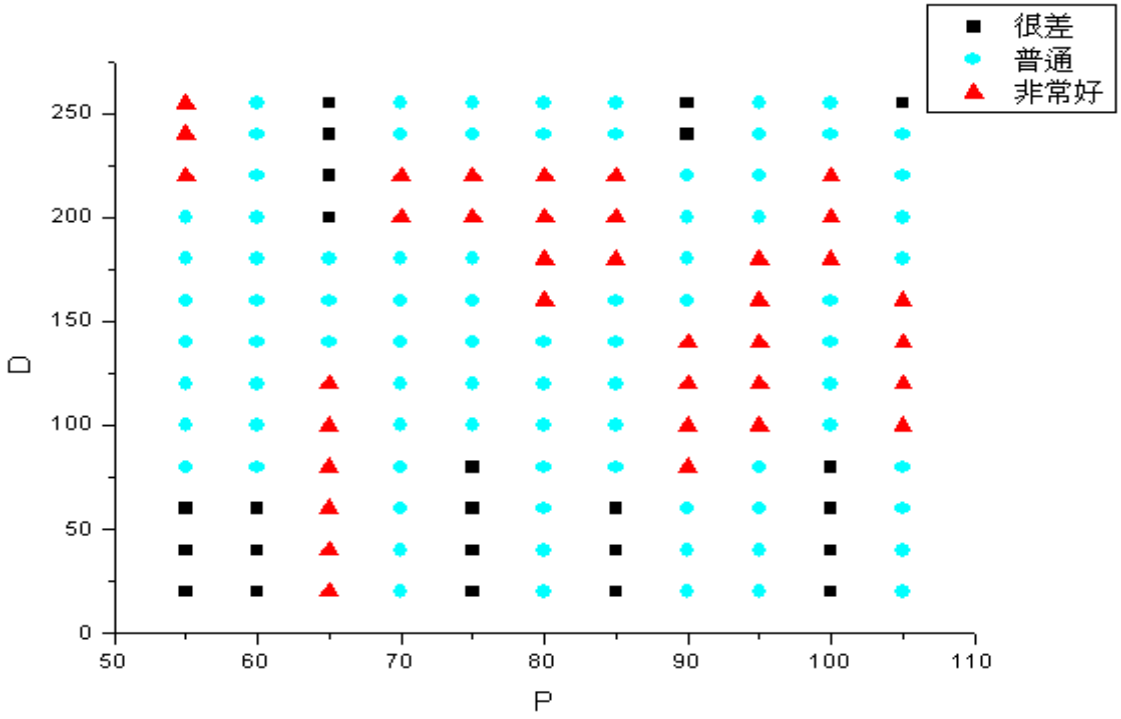
誤差值(Error)與左右輪速度變化值



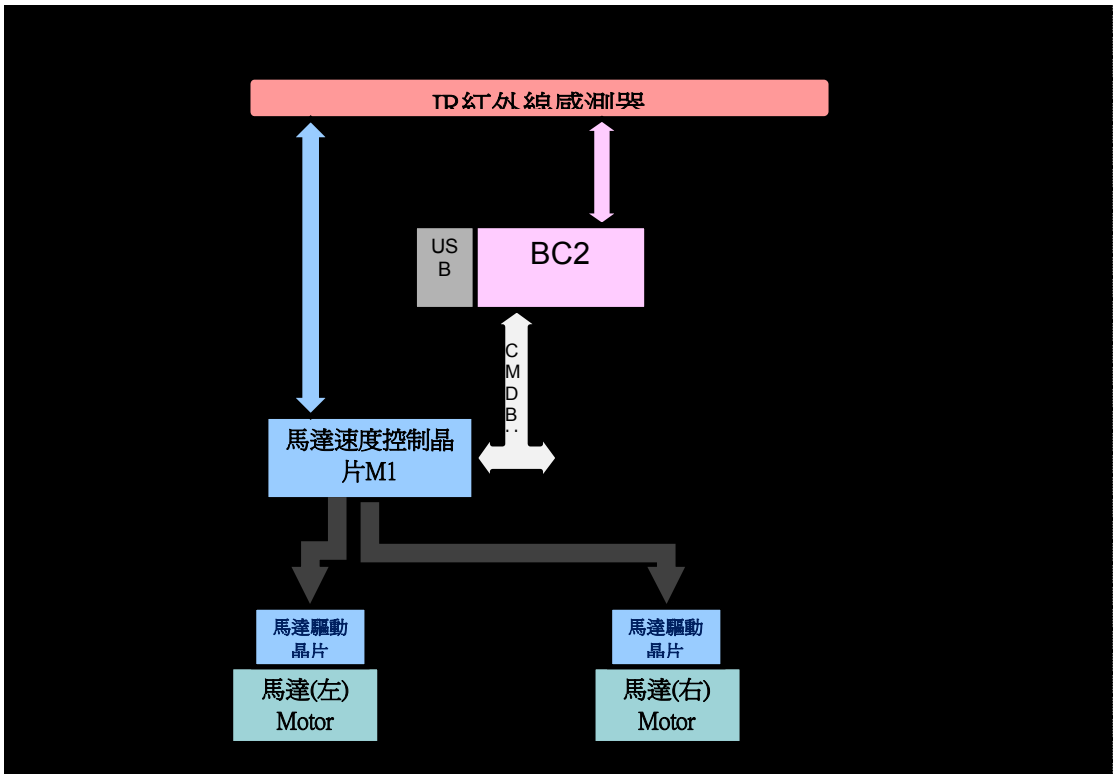
3-3 PD控制

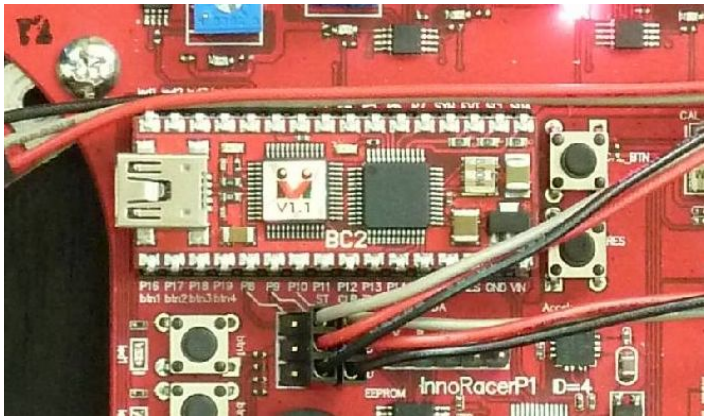
```
‘說明: 使用PID控制自走車沿軌道行走。
Peripheral myM As RacerM1 @ 3    ’設定馬達控制模組參數名稱
#DEFINE KP       1    ’設定PID的參數值
#DEFINE KI       0
#DEFINE KD       10
#DEFINE SCALE    0
#DEFINE NORMAL_SPEED_R250    ’設定右輪直線速度值
#DEFINE NORMAL_SPEED_L250    ’設定左輪直線速度值
#DEFINE ERROR_1  360    ’設定不同感測結果的誤差設訂值
#DEFINE ERROR_2  280
#DEFINE ERROR_3  180
#DEFINE ERROR_4  130
#DEFINE ERROR_5  80
#DEFINE ERROR_6  40
#DEFINE ERROR_7       0
#DEFINE ERROR_8  -40
#DEFINE ERROR_9  -80

Case &B1001111
      Err = ERROR_4
            Case &B1101111
      Err = ERROR_5
            Case &B1100111
      Err = ERROR_6
            Case &B1110111
      Err = ERROR_7
            Case &B1110011
      Err = ERROR_8
            Case &B1111011
      Err = ERROR_9
            Case &B1111001
      Err = ERROR_10
            Case &B1111101
      Err = ERROR_11
            Case &B1111100
```


很差
普通
非常好
D
P
0
50
100
150
200
250
0
10
20
30
40
50


很差
普通
非常好
D
P
0
50
100
150
200
250
50
60
70
80
90
100
110

3-4 M1 內建 PD 控制





M1 內建感測器校正功能

利用 M1 內建 PD 控制時
需要先完成 M1 內建之 IR 紅外線感測器感度校正
校正程序方法為:

1. 按下 CAL_BTN 按鈕一直到紅色 CAL_LED 亮起（約按住 5 秒）

2. 此時將 InnoRacer 平放在競速場地中 並對齊跑道白線
3. 並以馬達軸線與白線的交插點為圓心將車子旋轉
4. 以一定速度緩慢的將所有 IR 紅外線感測器來回劃過白線部份
5. 完成以上動作之後再按下 CAL_BTN 按鈕以結束校正

3-5 M1 內建 PD 控制

```
說明: 使用 RacerM1 模組，使用內建 PID 控制，偵測跑道行駛。
Peripheral myM1 As RacerM1 @ 3    ’設定馬達控制模組參數名稱
#DEFINE PID_P  4    ’M1 模組內部 PID 參數值 (0 ~ 255)
#DEFINE PID_I  0
#DEFINE PID_D  40
#DEFINE PID_SCALE    0
#DEFINE IR_POWER    70       ’設定紅外線感測強度
 ‘模組內部最高最低速限，以及直線速度 (-1024 ~ 1024)
#DEFINE    MAX_SPEED_L      1024
#DEFINE          MAX_SPEED_R      1024
#DEFINE    CENTER_SPEED_L   220
#DEFINE    CENTER_SPEED_R   220
#DEFINE    MIN_SPEED_L     -1024
#DEFINE          MIN_SPEED_R    -1024
’模組各別感測階段的誤差值 (0 ~ 127)
#DEFINE ERROR_1 10
#DEFINE ERROR_2 20
#DEFINE ERROR_3 32
#DEFINE ERROR_4 45
#DEFINE ERROR_5 70
#DEFINE ERROR_6 90
Sub InitM1()      ’設定 M1 模組各項預設值
 ‘設定 M1 模組內部 PID 參數值
    myM1.SetP(PID_P)
    myM1.SetI(PID_I)
```

```
    myM1.SetD(PID_D)
    myM1.SetScalar(PID_SCALE)
    myM1.SetIRThreshold(IR_POWER)
 ‘設定 M1 模組內部最高最低速限，以及直線速度
    myM1.SetSpdCtrlA(MIN_SPEED_L, MAX_SPEED_L)
    myM1.SetSpdCtrlB(MIN_SPEED_R, MAX_SPEED_R)
    myM1.SetStraight(CENTER_SPEED_L, CENTER_SPEED_R)
 ‘設定 M1 模組各別感測階段的誤差值
    myM1.SetErrScale(ERROR_1, ERROR_2, ERROR_3, ERROR_4,
ERROR_5, ERROR_6)
End Sub
```

3-6 M1 內建 PD 控制(類比)

```
’說明: 使用 RacerM1 模組，使用內建 PID 控制，偵測跑道行駛。
Peripheral myM1 As RacerM1 @ 3    ’ 設定馬達控制模組參數名稱
#DEFINE PID_P    4    ’M1 模組內部 PID 參數值 (0 ~ 255)
#DEFINE PID_I     0
#DEFINE PID_D    40
#DEFINE PID_SCALE    0
#DEFINE IR_MODE     1         ‘設定紅外線感測模式
#DEFINE IR_POWER    70        ’ 設定紅外線感測強度
 ‘模組內部最高最低速限，以及直線速度 (-1024 ~ 1024)
#DEFINE   MAX_SPEED_L     1024
#DEFINE         MAX_SPEED_R     1024
#DEFINE   CENTER_SPEED_L  220
#DEFINE   CENTER_SPEED_R  220
#DEFINE   MIN_SPEED_L    -1024
#DEFINE         MIN_SPEED_R    -1024
#DEFINE ERROR_1    10     ‘模組各別感測階段的誤差值 (0 ~ 127)
#DEFINE ERROR_2    20
#DEFINE ERROR_3    32
#DEFINE ERROR_4    45
```

```
#DEFINE ERROR_5    70
#DEFINE ERROR_6    90
Sub InitM1()     ’設定 M1 模組各項預設值
    myM1.SetP(PID_P)      ‘設定 M1 模組內部 PID 參數值
    myM1.SetI(PID_I)
    myM1.SetD(PID_D)
    myM1.SetScalar(PID_SCALE)
    myM1.SetIRMode(IR_MODE)    ‘類比數位選擇
    myM1.SetIRThreshold(IR_POWER)
 ‘設定 M1 模組內部最高最低速限，以及直線速度
    myM1.SetSpdCtrlA(MIN_SPEED_L, MAX_SPEED_L)
    myM1.SetSpdCtrlB(MIN_SPEED_R, MAX_SPEED_R)
    myM1.SetStraight(CENTER_SPEED_L, CENTER_SPEED_R)
 ‘設定 M1 模組各別感測階段的誤差值
    myM1.SetErrScale(ERROR_1, ERROR_2, ERROR_3, ERROR_4,
ERROR_5, ERROR_6)
End Sub
```

第四章:指令表

Module RacerM1: 下面的指令表是專供控制 Racer M1 模組的各種指令，必要輸 入的指令名稱與參數，以粗底或粗斜體表示，粗體的文字在輸入時請不要更改， 粗斜體的文字請自行定義適當格式的參數填入。輸入時請注意 innoBASIC™ Workshop 大寫與小寫會視為相同字。

在執行 RacerP1 指令前，請先於程式開頭定義對應參數與編號，例:

Peripheral *ModuleName* As RacerM1 @ 3

指令格式	指
馬達控制相關指令	
ForwardA(*Speed*)	以 ***Speed*** 輸入的速度值,設定馬達向前轉的速度。 可以輸入 0 ~ 1024 間的整數值。其中 A 爲左輪馬 達，B 爲右輪馬達。
ForwardB(*Speed*)	
ForwardAB(*SpeedA, SpeedB*)	
ForwardDual(*Speed*)	
BackwardA(*Speed*)	以 ***Speed*** 輸入的速度值,設定馬達向後轉的速度。 ***Speed*** 可以輸入 0 ~ 1024 間的整數值。
BackwardB(*Speed*)	
BackwardAB(*SpeedA, SpeedB*)	
BackwardDual(*Speed*)	
StopA()	停止指定馬達轉動。
StopB()	
StopDual()	
BrakeA()	快速停止指定馬達的轉動。
BrakeB()	
BrakeDual()	
SetDirA(*Dir*)	以 ***Dir*** 輸入的方向，設定指定馬達的轉向。 ***Dir*** 可以輸入 0 代表向前轉，1 代表向後轉。
SetDirB(*Dir*)	
SetDirAB(*DirA, DirB*)	
SetDirDual(*Dir*)	
SetDCA(*Speed*)	以 ***Speed*** 輸入的速度值，設定馬達轉動的速度。 ***Speed*** 可以輸入 0 ~ 1024 間的整數值。
SetDCB(*Speed*)	
SetDCAB(*SpeedA, SpeedB*)	
SetDCDual(*Speed*)	
SetVelA(*Vel*)	以 ***Vel*** 輸入的速度值，設定馬達轉動的速度。並且以正負值設定馬達的
SetVelB(*Vel*)	

SetVelAB(*VelA, VelB*)	
SetVelDual(*Vel*)	
取得馬達轉速與轉向相關指令	
GetDCA(*Speed*)	取得馬達轉速儲存於 ***Speed*** 參數中。
GetDCB(*Speed*)	***Speed*** 回傳值範圍爲 0～1024 間的整數值。
GetDCAB(*SpeedA, SpeedB*)	
GetDirA(*Dir*)	取得指定馬達的轉向，儲存於 ***Dir*** 參數中。 ***Dir*** 回傳值爲 0 或 1。
GetDirB(*Dir*)	
GetDirAB(*DirA, DirB*)	
GetVelA(*Vel*)	取得馬達轉速與方向儲存於 ***Vel*** 參數中。 ***Vel*** 回傳值範圍爲-1024～1024 間的整數值。
GetVelB(*Vel*)	
GetVelAB(*VelA, VelB*)	
紅外線感測相關指令	
GetIR(*IR*)	取得紅外線數位的感測值，儲存於 ***IR*** 參數中。***IR*** 回傳的 bit0~bit6，對應到各個紅外線感測的數位 感測值。
GetAnalogIR(*ID, IR*)	根據 ***ID*** 設定的感測器編號，取得對應紅外線感測 器的類比感測值，儲存於 ***IR*** 參數中。
NormStart(*Mode*)	以 ***Mode*** 輸入的設定值，啓動紅外線感測器的校 正模式。***Mode*** 可以輸入 0～4 間的整數值。 0: 持續校正直到按下校正按鈕。 1: 校正模式開始 10 秒後結束。 2: 校正模式開始 20 秒後結束。
GetNorm (*ID, Min, Max*)	取得 ***ID*** 指定的紅外線感測器，校正所得的最大與 最小值，存放於 ***Min*** 與 ***Max*** 中。 ***ID*** 可以設定 0～6 間的整數值。
SetIRThreshold(*Rate*)	以 ***Rate*** 輸入的設定值，設定該比例爲紅外線感測 器，將偵測值設定爲高準位的 最小值。 ***Rate*** 可以輸入 0～

GetIRThreshold(*Rate*)	取得紅外線感測器，判斷爲高電位的最小比例 值，儲存於 ***Rate*** 中。
SetIRMode(*Mode*)	以 ***Mode*** 輸入的設定值，設定速度控制時，所要 使用的紅外線感測模式。 ***Mode*** 可以輸入 0 或 1。
	1: 以類比值控制。 預設值爲 0。
GetIRMode(*Mode*)	取得速度控制時，使用的紅外線感測模式，儲存 於 ***Mode***。
PID 設定與讀取相關指令	
SetP(*Val*)	以 ***Val*** 輸入的設定值，設定 PID 相關參數的數值。 ***Val*** 可以輸入 0 ~ 255 間的
SetI(*Val*)	
SetD(*Val*)	
GetP(*Val*)	取得設定的 PID 值，儲存於 ***Val*** 參數中。 ***Val*** 回傳值爲 0 ~ 255 間的整數值。
GetI(*Val*)	
GetD(*Val*)	
SetScalar(*Val*)	以 ***Val*** 輸入的設定值，設定 PID Scalar 的數值。 ***Val*** 可以輸入 0 ~ 255 間的 整數值。 若輸入大於 32 則
GetScalar(*Val*)	取得設定的 PID Scalar 值，儲存於。 ***Val*** 回傳值爲 0 ~ 255 間的整數值。
SetErrScale(*Err1, Err2, Err3, Err4, Err5, Err6*)	以 ***Err1 ~ Err6*** 輸入的設定值，分別設定各種紅外 線感測狀況的誤差值。 ***Err1 ~ Err6*** 可以輸入 0 ~ 127 間的 整數值。 預設值如下: ***Err1*** = 1 ***Err2*** = 2 ***Err3*** = 3 ***Err4*** = 4 ***Err5*** = 5

GetErrScale (*Err1, Err2, Err3, Err4, Err5, Err6*)	取得設定的紅外線感測誤差值，儲存於 ***Err1*** ~ ***Err6***。
速度控制相關指令	
SetSpdCtrlA(*SpdMin, SpdMax*)	以 ***SpdMin*** 與 ***SpdMax*** 輸入的設定值，分別設定速 度控制啓動後的最大速度值與最小速度值。 ***SpdMin*** 與 ***SpdMax*** 都可以輸入-1024～1024 間的 整數值，但 ***SpdMax***
SetSpdCtrlB(*SpdMin, SpdMax*)	
GetSpdCtrlA(*SpdMin, SpdMax*)	取得設定的速度控制最大與最小值，分別儲存於 ***SpdMin*** 與 ***SpdMax*** 中。 ***SpdMin*** 與 ***SpdMax*** 回傳值爲-1024 ~
GetSpdCtrlB(*SpdMin, SpdMax*)	
SetStraight(*SpeedA, SpeedB*)	以 ***SpeedA*** 與 ***SpeedB*** 輸入的設定值，分別設定速 度控制啓動後，直線行走時的左右馬達速度值。 ***SpeedA*** 與 ***SpeedB*** 都可以輸入-
GetStraight(*SpeedA, SpeedB*)	取得設定的速度控制直線行走值，分別儲存於 ***SpeedA*** 與 ***SpeedB*** 中。 ***SpeedA*** 與 ***SpeedB*** 回傳值爲-1024～1024
SpdCtrlOn(*Mode*)	以 ***Mode*** 設定的模式，啓動速度控制。 0: 若切換速度自動結束速度控制。 1: 切換速度仍繼續速度控制。
SpdCtrlOff()	關閉速度控制。
GetMax(*SpeedA, SpeedB*)	取得速度控制下，各馬達設定到的最大值，分別 儲存於 ***SpeedA*** 與 ***SpeedB***。
GetMin(*SpeedA, SpeedB*)	取得速度控制下，各馬達設定到的最小值，分別 儲存於 ***SpeedA*** 與 ***SpeedB***。
ClearRec()	清除最大與最小速度值紀錄

SetCtrlFreq(*Period*)	以 ***Period*** 設定的參數值，設定速度控制的間隔時 間。***Period*** 可以設定爲 0 ~ 100 間的整數值，單位 爲 ms。
GetCtrlFreq(*Period*)	取得設定速度重置的間隔時間，儲存於 ***Period***。
各項設定相關指令	
SetCrossMode(*Mode*)	以 ***Mode*** 設定的參數值，設定經過軌道交叉點的 執行動作。 ***Mode*** 可以設定爲 0，1 或 2。 0: 經過交叉軌道不做任何動作。 1: 經過交叉軌道執行 Stop。 2: 經過交叉軌道執
GetCrossMode(*Mode*)	取得設定經過交叉軌道的模式，儲存於
	Mode 回傳值爲 0 ，1 或 2。
SetOutsideMode(*Mode*)	以 ***Mode*** 設定的參數值，設定跑出軌道的執行動 作。 ***Mode*** 可以設定爲 0，1 或 2。 0: 跑出軌道不做任何動作。 1: 跑出軌道執行 Stop。 2: 跑出軌道執
GetOutsideMode(*Mode*)	取得設定跑出軌道的模式，儲存於 ***Mode***。
SetLineColor(*Color*)	以 ***Color*** 設定的參數值，設定軌道的顏色。預設爲 0。 ***Color*** 0: 軌道爲白色。
GetLineColor(*Color*)	取得設定軌道的顏色，儲存於 ***Color***。 ***Color*** 回傳值爲 0 或 1。

Module RacerP1: 下面的指令表是專供控制 Racer P1 模組的各種指令，必要輸入 的指令名稱與參數，以粗底或粗斜體表示，粗體的文字在輸入時請不要更改，粗 斜體的文字請自行定義適當格式的參數填入。輸入時請注意 innoBASIC Workshop 大寫與小寫會視為相同字。在執行 RacerP1 指令前，請先於程式開頭定義對應參數與編號，例:Peripheral *ModuleName* As RacerP1 @ 4

指令格式	指
馬達回傳脈波相關指令	
bStatus **= TACHInR(*TACH*)** ***bStatus*** **= TACHInL(*TACH*)**	取得指定馬達回傳的脈波數，儲存於 ***TACH***，並將 狀態值存於 ***bStatus*** 中。***TACH*** 會回傳 0~65535 間 的整數值。 ***bStatus*** 爲 0 代表尚未更新，反之爲
bStatus **= TACHInDual(*TACHR, TACHL*)**	取得左右馬達回傳的脈波數，儲存於 ***TACHR*** 與 ***TACHL*** 中，並將狀態值存於 ***bStatus*** 中。 ***TACHL*** 與 ***TACHR*** 會回傳 0~65535 間的整數值。 ***bStatus*** 0: 尚未更新。 1: 左輪尚未更新。 2: 右輪尚未更新。
軌道紀錄相關指令	
StartRec(*Mode*)	開始紀錄軌道資訊，並根據 ***Mode*** 設定，儲存資 料到 EEPROM 中。 ***Mode*** 0: 不儲存資料於 EEPROM。
StopRec()	停止軌道紀錄。

GetRecStatus(*Status*)	取得軌道紀錄狀態，存於 ***Status*** 中。 ***Status*** 0: 沒有開始紀錄模式或紀錄結束。 1: 進入紀錄模式，但沒有經過開始點。 2: 進入紀錄模式，且已通過開始點。
ClrTotalLen()	清除 **StartRec**()的累計計數值。
GetRateRL(*Rate*)	取得左右輪的脈波數比，乘上 65536 後，儲存於 ***Rate***。
GetSecCnt(*Cnt*)	取得經過的曲率變化點個數，儲存於 ***Cnt***。
GetSecLen(*Num, LengthR, LengthL*)	取得 ***Num*** 指定路段左右輪行駛的距離，分別儲存 於 ***LengthL*** 與 ***LengthR***。 ***Num*** 可以設定 0～255 間的整數值。 ***LengthL*** 與 ***LengthR*** 回傳 0～4294967295
GetCurSecTACH(*LengthR, LengthL*)	取得最近一次曲率變化點(含開始點)，到現在位置 的左右輪行駛距離，儲存於 ***LengthL*** 與 ***LengthR*** 中。***LengthL*** 與 ***LengthR*** 回傳 0～4294967295 間的整數 值。
GetTotalLen(*LengthR, LengthL*)	取得從開始記錄到下指令讀取間的左右輪行駛距 離，儲存於 ***LengthL*** 與 ***LengthR*** 中。 ***LengthL*** 與 ***LengthR*** 回傳 0～4294967295 間的整數 值。
計數設定相關指令	
SetTimer(*Cnt*)	根據 ***Cnt*** 設定計數要提醒的時間值。Timer 的啓 動，必須在 CLR 腳位產生一個低到高電位，並在 時間到達時，會在 TMR 腳位產生一個高電位訊 號，直到再次偵測到 CLR 有低到高的電位變化。 ***Cnt*** 可以輸
GetTimer(*Cnt*)	取得計數設定的時間值，存於 ***Cnt*** 中。 ***Cnt*** 回傳值爲 0～1000 間的整數值。

紅外線感測相關指令	
GetIR(*IR*)	取得紅外線感測值，存於 ***IR*** 中。回傳值的 bit0 爲 開始與結束記號判斷值，bit1 爲曲率變
加速度感測相關指令	
GetG(*Gx, Gy*)	取得 X 與 Y 軸向加速度感測值，存於 ***Gx*** 與 ***Gy*** 中。
GetMaxG(*Gx, Gy*)	取得上一個路段，X 與 Y 軸向最大的加速度感測 值，存於 ***Gx*** 與 ***Gy*** 中。 ***Gx*** 與 ***Gy*** 回傳值爲-2048～2047 間的整數
GetAvgG(*Gx, Gy*)	取得上一個路段，X 與 Y 軸向平均的加
	值，存於 ***Gx*** 與 ***Gy*** 中。 ***Gx*** 與 ***Gy*** 回傳值爲-2048～2047 間的整數值。
GetSecMaxG(*Num, Gx, Gy*)	取得 ***Num*** 指定路段，X 與 Y 軸向最大的加速度感 測值，存於 ***Gx*** 與 ***Gy*** 中。 ***Num*** 可以設定 0～255 間的整數值。
GetSecAvgG(*Num, Gx, Gy*)	取得 ***Num*** 指定路段，X 與 Y 軸向平均的加速度感 測值，存於 ***Gx*** 與 ***Gy*** 中。 ***Num*** 可以設定 0～255 間的整數值。
SaveCur0G()	將現在量測到的電壓值，設定爲加速度感測器，X
Load0G(*Gx, Gy*)	讀取加速度感測器，0G 的設定值，存放於 ***Gx, Gy***
Set0G(*Gx, Gy*)	以 ***Gx*** 與 ***Gy*** 設定的參數值，設定加速度感測器在 0G 的設定值。
曲率半徑相關指令	

GetRadius(*Dir, Radius*)	取得上一個路段，曲率半徑的方向與半徑，存於 ***Dir*** 與中 ***Radius***。 ***Dir*** 會回傳 0(逆時鐘方向)或 1(順時鐘方向)。
GetSecRadius(*Num, Dir, Radius*)	取得 ***Num*** 指定路段，曲率半徑的方向與半徑，存 於 ***Dir*** 與中 ***Radius***。 ***Num*** 可以設定 0～255 間的整數值。 ***Dir*** 會回傳 0(逆時鐘方向)或 1(順時鐘方
其他設定指令	
Beep()	啓動 Buzzer 播放 0.2 秒。
AutoBeep(*Mode*)	根據 ***Mode*** 設定，自動啓動或關閉 Buzzer 播放。 ***Mode*** 0: 關閉自動撥放功能。 1: 啓動自動撥放功能，經過曲率變化點就會啓動
SetCrossTime(*Time*)	以 ***Time*** 設定交叉軌道的判定時間。
	如果曲率變化點偵測與開始結束變化點偵測，在 設定時間內偵測到另外一項，就視爲經過交叉軌 道，不做曲率變化紀錄，也不會開始或停止。 ***Time*** 可以輸入 0～
GetCrossTime(*Time*)	取得設定的交叉軌道判定時間，存於 ***Time*** 中。
SetLineColor(*Color*)	以 ***Color*** 設定的參數值，設定軌道的顏色。預設爲 0。 ***Color*** 0: 軌道爲白色。
GetLineColor(*Color*)	取得設定軌道的顏色，儲存於 ***Color***。 ***Color*** 回傳值爲 0 或 1。
EnWP()	啓動記憶體防寫保護。
DisWP()	關閉記憶體防寫保護。

第五章、結論

從一開始完全沒碰過的車子，到後來去比賽才慢慢的了解，才知道哪邊要注意的小細節，才能讓車子可以發揮更好的性能，以下是如何才能讓車子跑的更快更穩

1. 足夠的速度
2. 適當的PD設定
3. 適當的Error比例
4. 將跑道資訊充份利用
5. 更多段的變速
6. 足夠的磨擦力
7. 跑道與輪子的清潔
8. 適當的輪距與軸距
9. 更輕的電池
10. 適當的配重

從這些的條件下去一一的作測試，要調到最好最好的狀態，需要花很多很多的時間每項去研究，也需要去詢問或是去參考書籍，或許這一台小小的車子，但如果可以替代人力的話，可以節省許多的時間。

參考資料

利基應用科技股份有限公司:

http://www.innovati.com.tw/website/index.php

作者簡介

姓名:黃議霈
就讀學校:修平技術學院
就讀科系:電機工程系
e-mail:coolyoyomam771020@hotmail.com

姓名:張竣銘
就讀學校:修平技術學院
就讀科系:電機工程系
e-mail:steve771016@hotmail.com