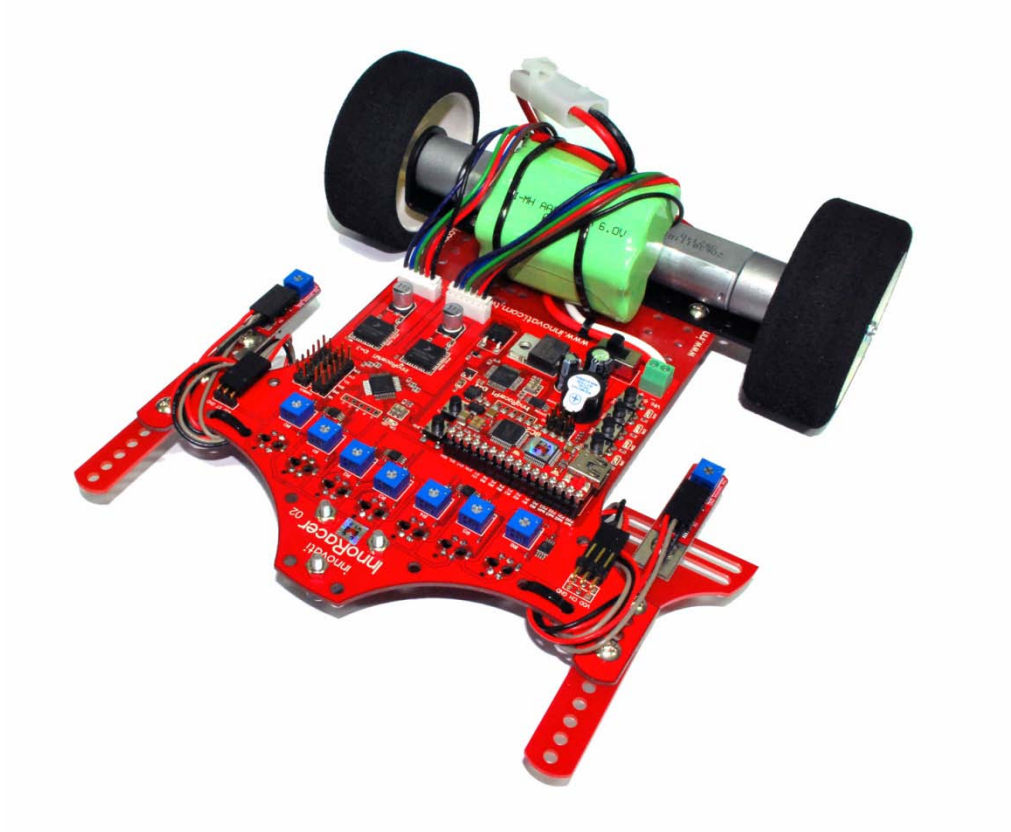
Innoracer™

循跡競速自走車

使用手冊

版本: 1.2



Passion for innovation

商標

Innovati®，  圖案與BASIC Commander® 為利基應用科技股份有限公司之註冊商標。innoBASIC™及cmdBUS™為利基應用科技股份有限公司之商標。

基於對產品的持續改善，本公司得不經通知隨時變更本資料或本資料中所提及之產品。未經本公司書面同意或授權，不可重製、散布本產品局部或全部內容。

免責聲明

使用者在使用本產品所做的任何應用，使用者須自行承擔一切風險。公司對於因使用本產品所生之直接、間接或附帶損害，包括且不限設備損失、人身安全健康損失、利潤信譽損失，不負任何責任。本公司產品不可使用於救生或相關儀器設備。未滿14歲兒童須有成人陪同方可使用本產品進行相關實驗。

勘誤

希望使用者會覺得這是一本生動而且實用的使用手冊。我們花費很多心力於讓這本手冊更加完整而正確的傳達我們希望使用者了解的訊息，然而難免仍有疏漏之處。為了提供使用者手冊提供最新最翔實的資訊，我們會持續改善增補手冊內容。如在本手冊中發現錯誤之處， 歡迎利用電子郵件 service@innovati.com.tw 與我們連絡。如有任何相關資訊更新皆會揭露於網站上，請經常瀏覽我們的網站 http://www.innovati.com.tw，以便獲知最新資訊。

注意事項

- 本套件含有 BASIC Commander® 2 單板電腦，有附帶的說明書與使用特性介紹，請參閱說明書讓套件可以發揮最好的效能。
- 更換其他電池或輸入電源時，**請確定輸入電壓於 6-12V 之間**，避免造成模組毀損。
- 馬達的電壓輸入須依據連接之馬達提供對應的電壓，本套件提供的**馬達請輸入 6V 之電壓**，過高與過低的電壓，將造成無法預期的動作，甚至可能燒毀馬達，連接電源前請確定提供的電壓值。
- 本套件含兩個直流馬達，馬達同時運作需要較大電源，請確定連接至本套件的電源供應器或電池，能提供 **2A 以上的電流值**，讓套件能正常動作。提供電流不足時，可能造成無法預期之動作，損壞套件。
- 如果使用電池做為套件的電源，在操作一段時間後，電池電壓降低會造成套件無法正常動作，此時請關閉電源，於充電完成後再行使用。如果需要長時間的測試與操作，建議使用電源供應器，維持一致的效能。
- 此套件內建模組所使用的指令，須安裝 innoBASIC™ Workshop v2.0.2.9 以後版本，才能正常動作。

開始安裝套件前，請先依照光碟片內容，安裝 innoBASIC™ Workshop，並確定 PC 可透過 USB 線與 BASIC Commander®連接，才能正確完成整個組裝動作。

目錄

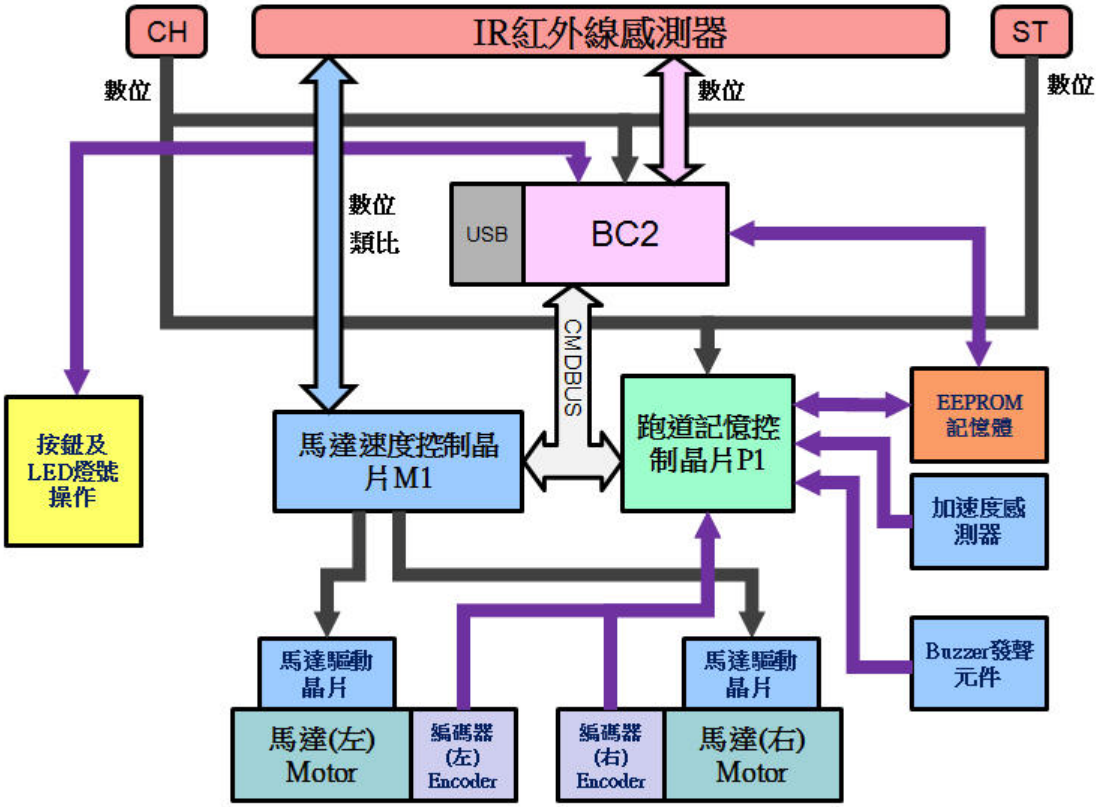
產品介紹:

利基 Innoracer™是以 BC2(BASIC Commander® 2)爲核心，控制兩個內建模組，Racer M1 與 Racer P1，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，不同賽到的極限速度，完成競速的目的。

產品特色:

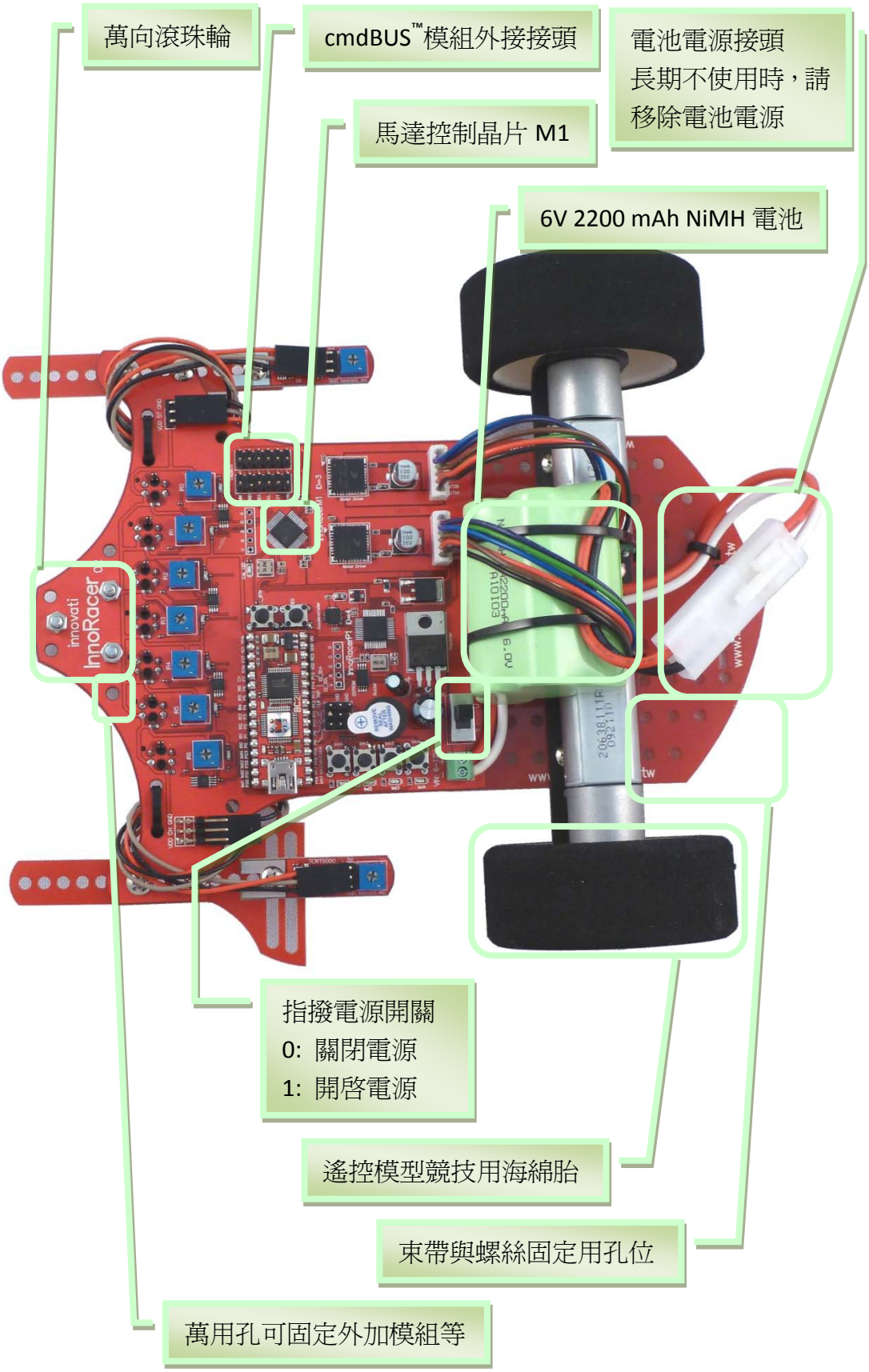
- 使用 BC2 爲核心，可以隨時於電腦上更改程式，再透過 USB 下載至 BC2。
- 四組 cmdBUS™接腳，可以直接與具有 cmdBUS™的應用模組連接，如超音波感測器，增加各種應用功能。
- 7 個軌道偵測固定紅外線感測器。
- 左右有兩個可移動紅外線感測器，可以偵測軌道外圍的軌道變化資訊，也可以隨時更改位置，配合不同軌道變化應用。
- 紅外線校正按鈕，可以隨時隨地校正軌道偵測紅外線值域。
- 重置按鈕，可以隨時快速重新啓動程式。
- 數位紅外線感度，可以透過可變電阻調整。
- 四組可程式控制按鈕，經由程式設計，可以根據需求，設定不同按鈕效果。
- 四組可程式控制 LED，由程式設計，可以判斷狀態產生提醒或警示。
- 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。
- 提供 Scalar 參數，加大 Scalar 可以讓 PID 參數有更多的範圍做微調。
- 內建加速度感測晶片，可以偵測 X 與 Y 兩軸向加速度值。
- 內建軌道紀錄模組(Racer P1)，可以記錄軌道資訊。
- 能設定數位與類比紅外線軌道感測方式，也能以指令讀取感測值。
- 可以儲存路段長度，X 與 Y 軸向的平均加速度值，最大加速度值，曲率半徑與方向等資訊。
- 可以儲存 256 組不同路段資訊。
- 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。
- 在連接方式相同的條件下，能自由更換其他直流馬達，獲得不同馬達特性。

系統架構:

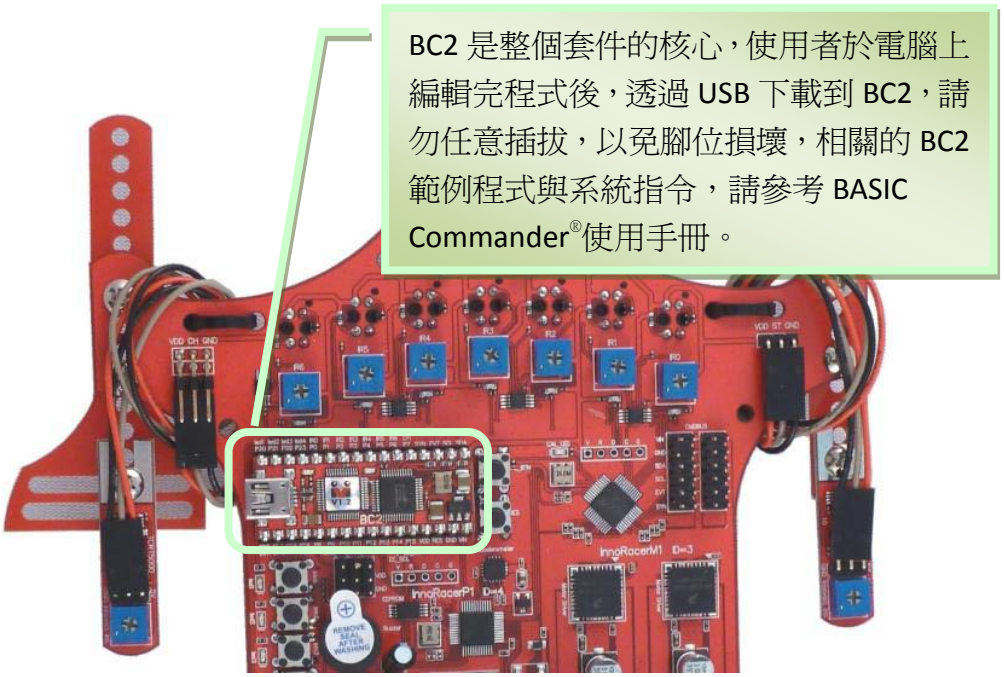


額外的擴充 cmdBUS™，輪胎，以及電池等非直接由 BC2，M1 或 P1 控制的元件，不列在此架構中。

元件說明:


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: 開啓電源
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

BC2

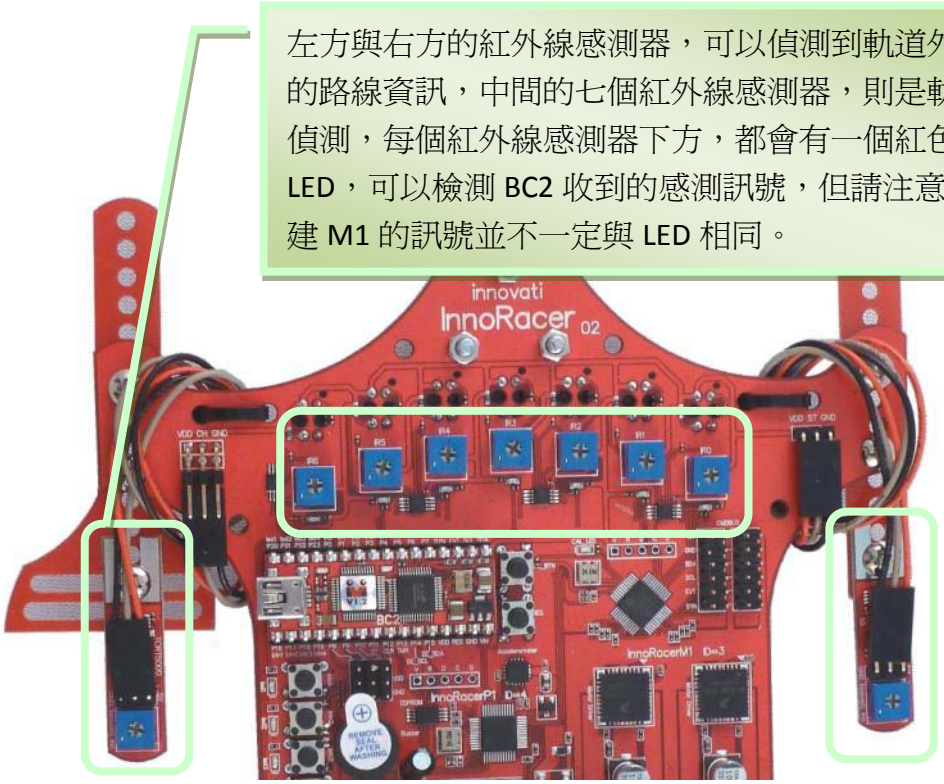


BC2 是整個套件的核心，使用者於電腦上編輯完程式後，透過 USB 下載到 BC2，請勿任意插拔，以免腳位損壞，相關的 BC2 範例程式與系統指令，請參考 BASIC Commander®使用手冊。

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
```

紅外線感測器



左方與右方的紅外線感測器，可以偵測到軌道外圍的路線資訊，中間的七個紅外線感測器，則是軌道偵測，每個紅外線感測器下方，都會有一個紅色 LED，可以檢測 BC2 收到的感測訊號，但請注意內建 M1 的訊號並不一定與 LED 相同。

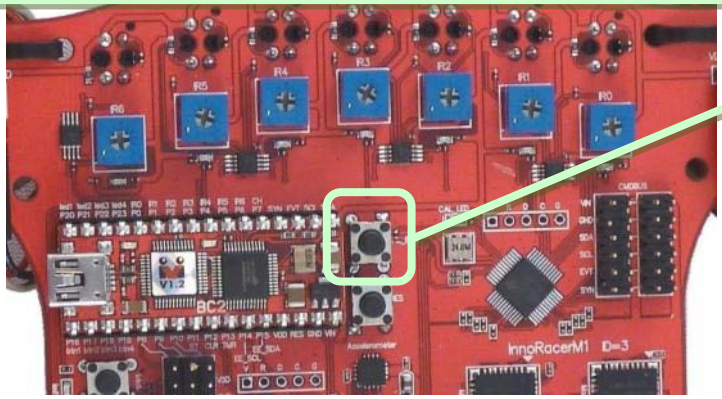
測試程式

```
'===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
```

紅外線校正按鈕

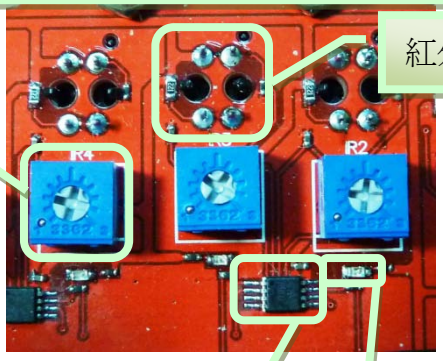
校正程序方法爲:

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爲圓心將車子旋轉
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



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

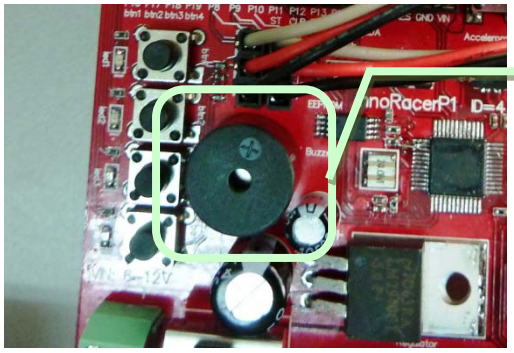
紅外線感測指示燈
有反射訊號時亮紅燈，沒收到反射訊號則熄滅

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

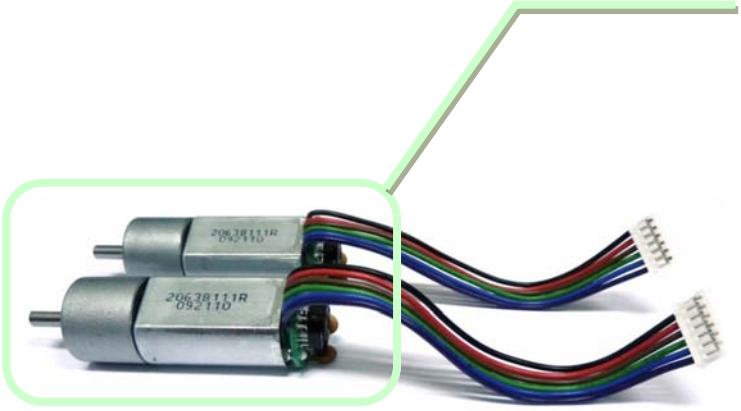
蜂鳴器



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
```

直流馬達



16000rpm
附編碼器驅動馬達
操作電壓 6V
齒輪比 1:13
連接線依序爲
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

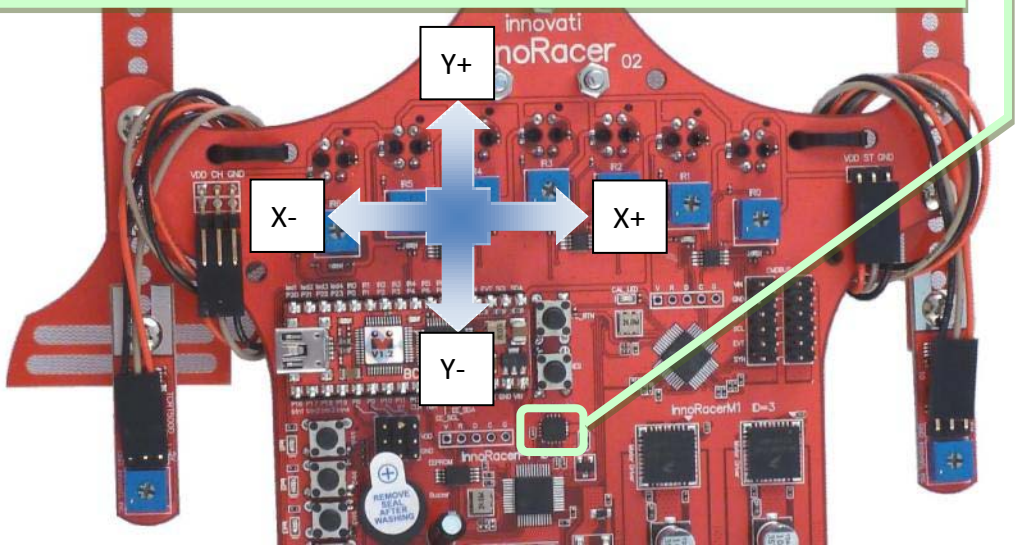
```
'說明: 根據輸入值操作馬達動作，並設定停止方式
'注意: 馬達轉動會讓自走車移動，請先墊高自走車讓輪胎離地，再開始
'       測試
Peripheral myM As RacerM1 @ 3  ' 設定馬達控制模組參數名稱

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
```

充電器

充電器僅提供充電用，請勿做爲套件電源使用。
此充電器專用於此套件所提供的鎳氫電池充電，請不要使用在其他特性不符合的充電電池，以免危險。

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，回傳值爲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爲校正值===
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，儲存爲校正值
    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
```

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 大寫與小寫會視爲相同字。

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	指令功能
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轉向。***Vel*** 可以輸入-1024 ~ 1024 間的整數值。
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。 ***IR*** 回傳值爲 0 ~ 127 間的整數值。
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。 ***IR*** 回傳值爲 0 ~ 4095 間的整數值。
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。 3: 校正模式開始 30 秒後結束。 4: 校正模式開始 60 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。 ***Min*** 與 ***Max*** 會回傳 0 ~ 4095 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 ~ 100 間的整數值。 預設值爲 50。
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。 ***Rate*** 會回傳 0 ~ 100 間的整數值。
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。 0: 以數位值控制。

	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***。 ***Rate*** 會回傳 0 或 1。
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 間的整數值。 預設值皆爲 0。
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 則會關閉 PID 控制。 預設值爲 255。
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 ***Err6*** = 6
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***。 ***Err1 ~ Err6*** 回傳值爲 0 ~ 127 間的整數值。
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*** 必須大於 ***SpdMin***。 預設值皆爲 0。
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 ~ 1024 間的整數值。
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-1024 ~ 1024 間的整數值。
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 間的整數值。
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***。 ***SpeedA*** 與 ***SpeedB*** 回傳值範圍爲-1024 ~ 1024。
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***。 ***SpeedA*** 與 ***SpeedB*** 回傳值範圍爲-1024 ~ 1024。
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。 預設爲 1。
GetCtrlFreq(*Period*)	取得設定速度重置的間隔時間，儲存於 ***Period***。 回傳值爲 0 ~ 100 間的整數值。
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: 經過交叉軌道執行 Brake。 預設爲 0。
GetCrossMode(*Mode*)	取得設定經過交叉軌道的模式，儲存於 ***Mode***。

	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: 跑出軌道執行 Brake。 預設爲 0。
GetOutsideMode(*Mode*)	取得設定跑出軌道的模式，儲存於 ***Mode***。 ***Mode*** 回傳值爲 0 ，1 或 2。
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。 1: 軌道爲黑色。
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 大寫與小寫會視爲相同字。

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	指令功能
馬達回傳脈波相關指令	
***bStatus* = TACHInR(*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 1。 回傳的脈波數爲 125 ms 單位時間內的量測值。
***bStatus* = TACHInL(*TACH*)**	
***bStatus* = TACHInDual(*TACHL, TACHR*)**	取得左右馬達回傳的脈波數，儲存於 ***TACHL*** 與 ***TACHR*** 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。 3: 皆爲更新值。 回傳的脈波數爲 125 ms 單位時間內的量測值。
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。 1: 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***。 ***Rate*** 回傳範圍爲 0 ~ 4294967295 間的整數值。

GetSecCnt(*Cnt*)	取得經過的曲率變化點個數，儲存於 ***Cnt***。 ***Cnt*** 回傳範圍爲 0 ~ 255 間的整數值。
GetSecLen(*Num, LengthL, LengthR*)	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 間的整數值。
GetCurSecTACH(*LengthL, LengthR*)	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。 ＊需要啓動紀錄模式才有效。
GetTotalLen(*LengthL, LengthR*)	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。 ＊需要啓動紀錄模式才有效。
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*** 可以輸入 0 ~ 1000 間的整數值，輸入 0 則代表關閉 Timer。單位爲 10Hz。
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 爲曲率變化記號判斷值。***IR*** 回傳值爲 0 ~ 3 間的整數值。
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。 ***Gx*** 與 ***Gy*** 回傳值爲-2048 ~ 2047 間的整數值。
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 間的整數值。 ＊需要啓動紀錄模式才有效。
GetAvgG(*Gx, Gy*)	取得上一個路段，X 與 Y 軸向平均的加速度感測

	值，存於 ***Gx*** 與 ***Gy*** 中。 ***Gx*** 與 ***Gy*** 回傳值爲-2048 ~ 2047 間的整數值。 ＊需要啓動紀錄模式才有效。
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。 ***Gx*** 與 ***Gy*** 回傳值爲-2048 ~ 2047 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。 ***Gx*** 與 ***Gy*** 回傳值爲-2048 ~ 2047 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 軸與 Y 軸 0G 的感測值。
Load0G(*Gx, Gy*)	讀取加速度感測器，0G 的設定值，存放於 ***Gx, Gy*** 中。
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。 ***Gx*** 與 ***Gy*** 可以輸入-2048 ~ 2047 間的任意整數值。
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)。 ***Radius*** 回傳值爲 0 ~ 4294967295 間的整數值。 ＊需要啓動紀錄模式才有效。
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(順時鐘方向)。 ***Radius*** 回傳值爲 0 ~ 4294967295 間的整數值。
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 Buzzer 播放 0.2 秒。 預設爲 0。
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 ~ 250 間的整數值。單位爲 ms。預設爲 1。
GetCrossTime(*Time*)	取得設定的交叉軌道判定時間，存於 ***Time*** 中。 ***Time*** 會回傳 0 ~ 250 間的整數值。單位爲 ms。
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。 1: 軌道爲黑色。
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。

附錄:

1. 已知問題: (版本貼紙於自走車中間上方雷射貼紙)
   - v1.1 版以前，**GetSecLen** 會回傳從開始記號，到該路段結束的總距離，可以用實驗十一的程式先記錄跑道，再用實驗十二的程式，確認是否讀回的距離值爲累加，做爲判斷的依據，如果讀到的距離值是不斷增加，就是 v1.1 版，在實驗十四就需要選擇舊版的範例。
   - 若希望更新爲 v1.2 版本請再將自走車寄回更新。