車在走天在看 PC 前的三輪車日記

書面報告

老師:林達德教授
隊員:生機三 張恆維 蔣壽山 陳龍 陳柏維
題目:電腦控制步進馬達定位車
日期:2005/6/27

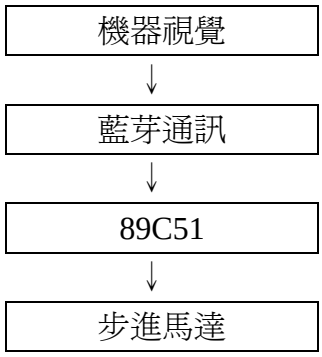
大綱

1.前言

這次期末專題的題目是電腦控制的步進馬達定位車，涵蓋的範圍相當的廣。有機器視覺，藍芽通訊，89C51 串列傳輸，步進馬達車。其實市面上類似的系統已經不少，像是拆除未爆彈的機器人，或是火場或地震現場救災用的機器人。我們希望可以藉由這次專題接觸類似的東西。以下就會有詳細的介紹。

2.摘要

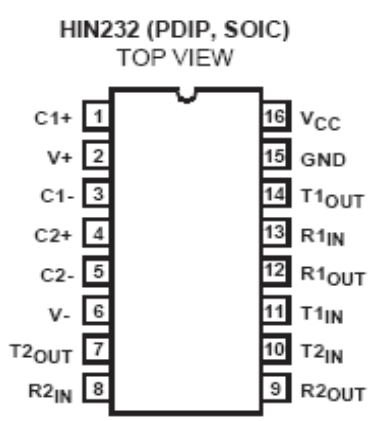
本系統利用網路攝影機拍攝一塊區域，然後把攝影機的資料跟電腦連結，所以在這塊區域的遙控車就可以利用無線通訊由電腦控制，甚至做到定位的功能。系統分成四個部份，A.機器視覺，B.藍芽通訊，C.89C51 串列傳輸，D.步進馬達控制。其中機器視覺還有藍芽通訊是在電腦端，而 89C51 和步進馬達是在定位車上。機器視覺(小六)。藍芽通訊我們使用學長留下來的模組(小六補充)。89C51 串列傳輸的部份，大家比較常聽到的 89C51 I/O port 應該就是 P0，P1，P2，P3 等”平行傳輸 port”，但是在 P3.0 還有 P3.1 有兩個 pin 處理串列通訊，也就是資料是以一定的頻率，一個 bit 一個 bit 傳送進來。我們的定位車上還有一個藍芽通訊的接收端，而我們再由接收端用串列傳輸把資料讀到 89C51。步進馬達主要是由 89C51 的 port0 來控制，在電路流程圖會有更詳細的介紹。

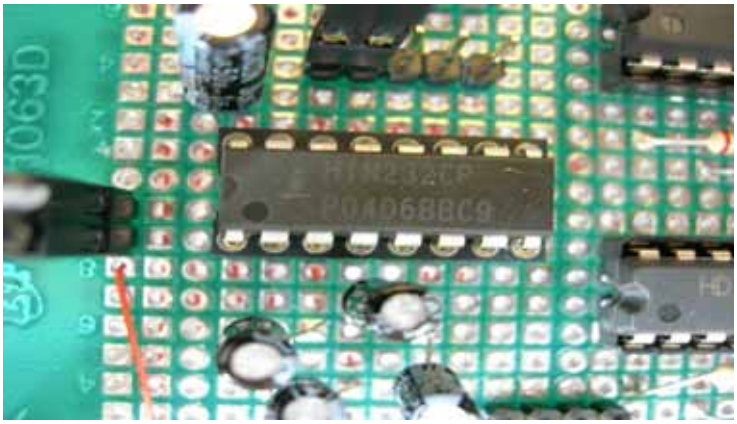


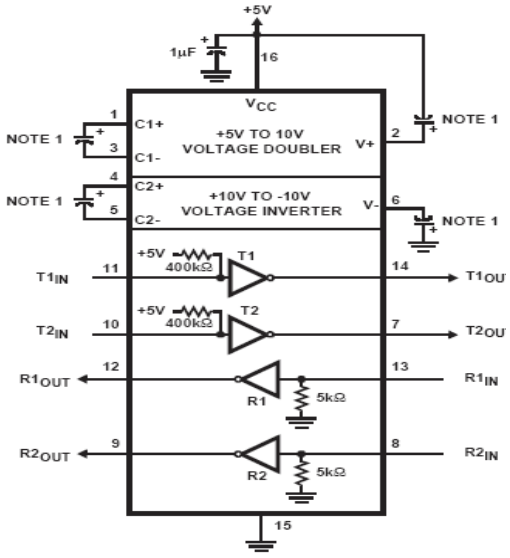
3.使用元件

Hin232 IC

這個 IC 是 RS232 串列傳輸的介面，如下圖。

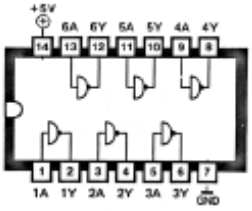


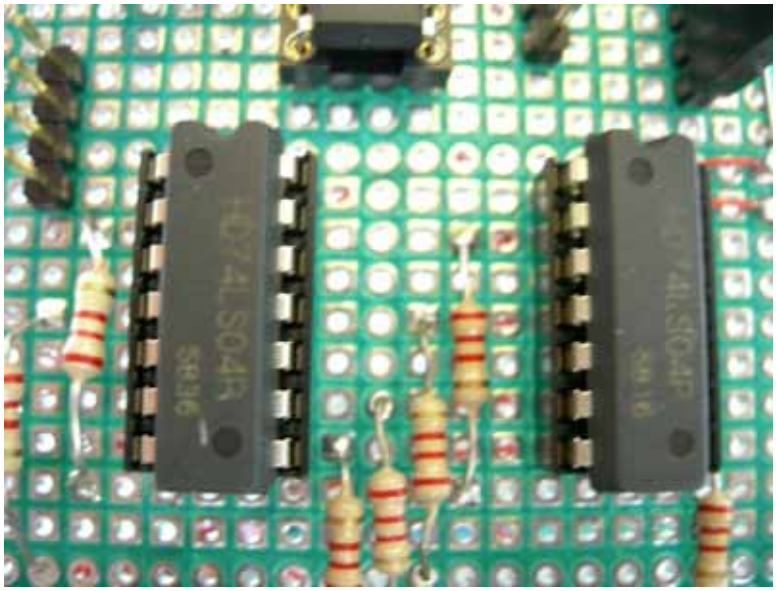




7404 反相器

反向器可以使輸入信號跟輸出反相，如下圖。

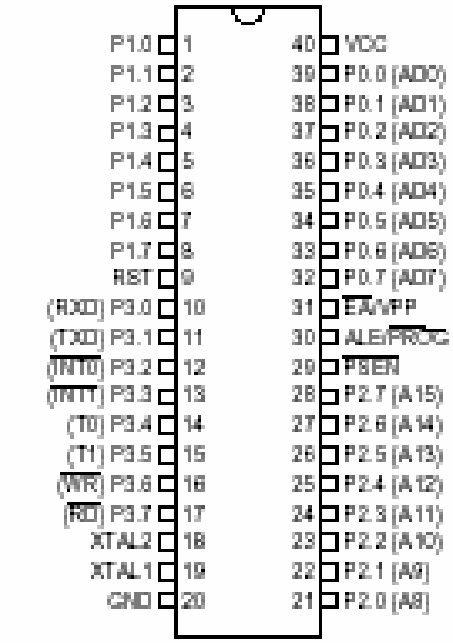




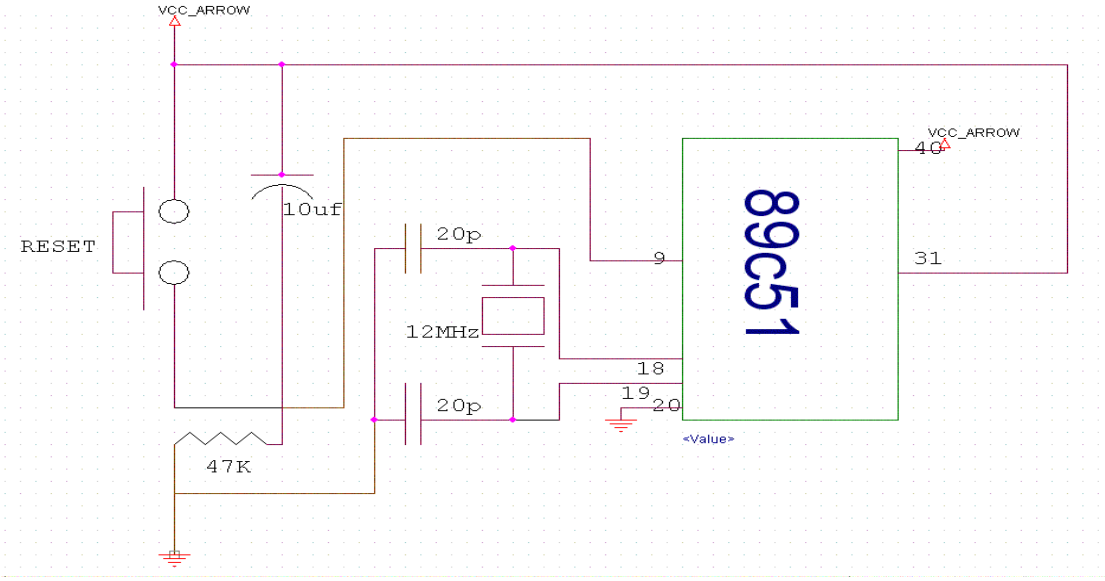
89C51 單晶片控制器

89C51 是本系統定位車的處理器，可以進行平行傳輸，串列傳輸等功能。我們主要利用其串列輸入和平行輸出的功能。外型請參考下圖。

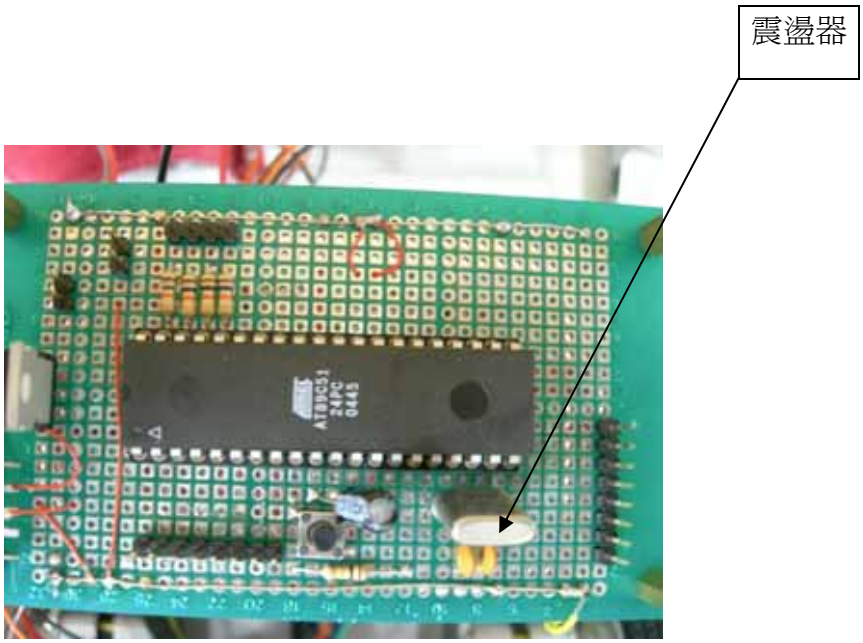
89C51 有四個平行傳輸的 port。分別爲 port0，port1，port2，port3。其中 port3 的前兩個 pin 可以當成串列傳輸的輸入跟回傳。詳細的腳位圖請參考下圖。



要使用 89C51 之前需要有一些電路配合，我們這次使用了”震盪電路”以及”重置電路”請參考下圖。其中震盪電路是爲了提供單晶片一個工作的時脈，我使用的是 12MHz 的石英震盪器，而重置電路就是提供使用者 reset 的功能。

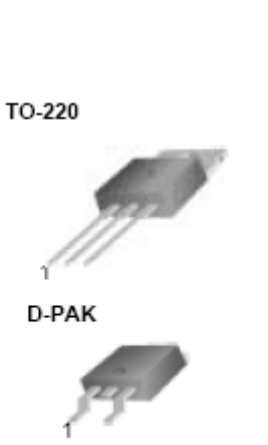


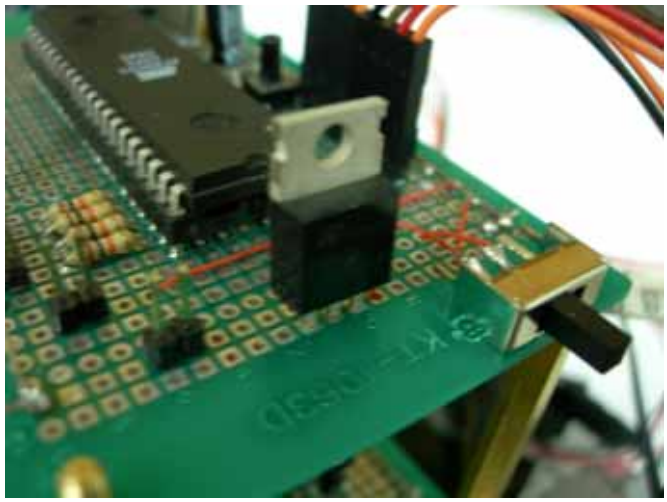
下圖是 89C51 周邊電路的實體圖



7805 穩壓 IC

穩壓 IC 是使輸入的電壓可以維持在 5V。見下圖。





步進馬達

步進馬達(stepping motor)的功能主要是使馬達一次可以只動”一步”，也就是可以做到精準定位的目的。我們使用的是四相式的步進馬達，所以總共有五條線(最後一條是電源)。請見下圖。

4.作動原理介紹

PC 端

PC 端介紹：

1. 程式：

程式主要部分有 WebCam 影像輸入、影像二元化、追蹤目標以及 RS232 序列訊號輸出，分別介紹如下：

a. WebCam 影像輸入：

```
#include <vcl.h>
#include <vfw.h>
#include <vcl\Clipbrd.hpp>
```

以上爲所需載入的 Header File，影像載入程式如下：

```
TClipboard *pCb = Clipboard() ;
HWND hwnd ;
int DriverIndex ;

hwnd =    capCreateCaptureWindow("My Own
CaptureWindow",WS_CHILD | WS_VISIBLE , 0,
0,352 ,288 ,Form1->Handle , 0);
capDriverConnect( hwnd , DriverIndex ) ;
capPreviewRate( hwnd , 66 ) ;
capPreview( hwnd , true) ;
```

以上較重要的地方爲 DriverIndex 爲 WebCam 連結的 Com 埠位置，與硬體連結時需要注意。

```
 Graphics::TBitmap *Bmap ;
 Byte *pBmap ;

 capEditCopy( hwnd ) ;
```

```
Bmap->LoadFromClipboardFormat( CF_BITMAP,pCb->GetAsHandl
e(CF_BITMAP),0 ) ;
Bmap->PixelFormat = pf24bit ;
```

此部分將影像存至 Bitmap 裡，需要注意 PixelFormat 的設定，這裏使用 24bit，可將影像做 256 階的灰階變化。

b. 影像二元化：

```
POSITION Total ;                    //POSITION 爲平面位置類型
Total = POSITION(0,0) ;             //建構子設定初始位置
int Threshold ;                     //二元化臨界值
```

這次將要用到的類別寫在另一個 Header File 裡，這樣在計算程式上可提供更高的便利性。

```
for(int i=0 ; i < Bmap->Height ; i++)
{
     Form1->pBmap = (Byte *)Form1->Bmap->ScanLine[i] ;
 for(int j=0 ; j < Bmap->Width ; j++)
 {
   if( (pBmap[j*3]+pBmap[j*3+1]+pBmap[j*3+2])/3 <Threshold)
   {
        pBmap[j*3] = pBmap[j*3+1] = pBmap[j*3+2] = 0 ;
        Total.X = Total.X + j ;
        Total.Y = Total.Y + i ;
        Count++ ;
   }
     else
     {
       pBmap[j*3] = pBmap[j*3+1] = pBmap[j*3+2] = 255 ;
     }
 }
}
```

以上將影像二元化後，取出 X、Y 中點作爲車體位置。

c. 追蹤目標：

追蹤目標主要的想法是將車子的新舊位址連線，以及車子目標位址連線，利用向量方法求 tangent 角，即可得出需要轉彎的方向及角度。

若 P、T 分別爲新舊車子位址、車子目標位址向量，則計算方式如下：

$$\theta = \tan^{-1}(\frac{P \times T}{P \bullet T}) = \tan^{-1}(\frac{\begin{vmatrix} P_x & P_y \\ T_x & T_y \end{vmatrix}}{P_x T_x + P_y T_y})$$

若θ小於０則目標在左方需往右轉，反之亦然，而利用步進馬達走一圈需要 200 步的特性，配合硬體上量得兩軸與輪胎直徑 D、d，則可利用數學方法求出馬達需要轉動的步數 S：

$$S = (\frac{D}{d} \times \theta) \times \frac{200}{\pi}$$

d. RS232 序列訊號輸出：

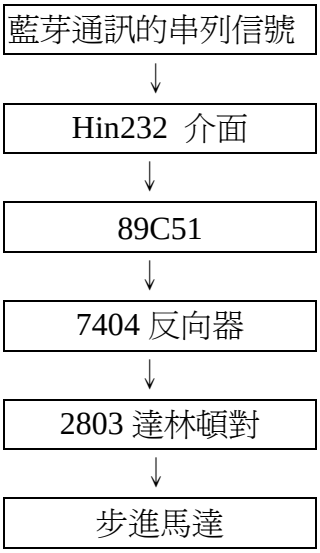
訊號輸出使用 Spcom 物件爲基礎，將方向訊號簡化爲單一字元（char，Forward－F，Back－B，Left－L，Right－R），將訊號輸出包入回圈內，次數則爲步數。

2. 介面：

介面主要分爲 WebCam 影像、二元化影像（臨界值控制、游覽）和控制介面（角度臨界值、控制面板），另外有一手控介面，大致上相同，但無影像部分。

定位車端

定位車上主要是處理 PC 端丟過來的信號。先由流程圖開始。

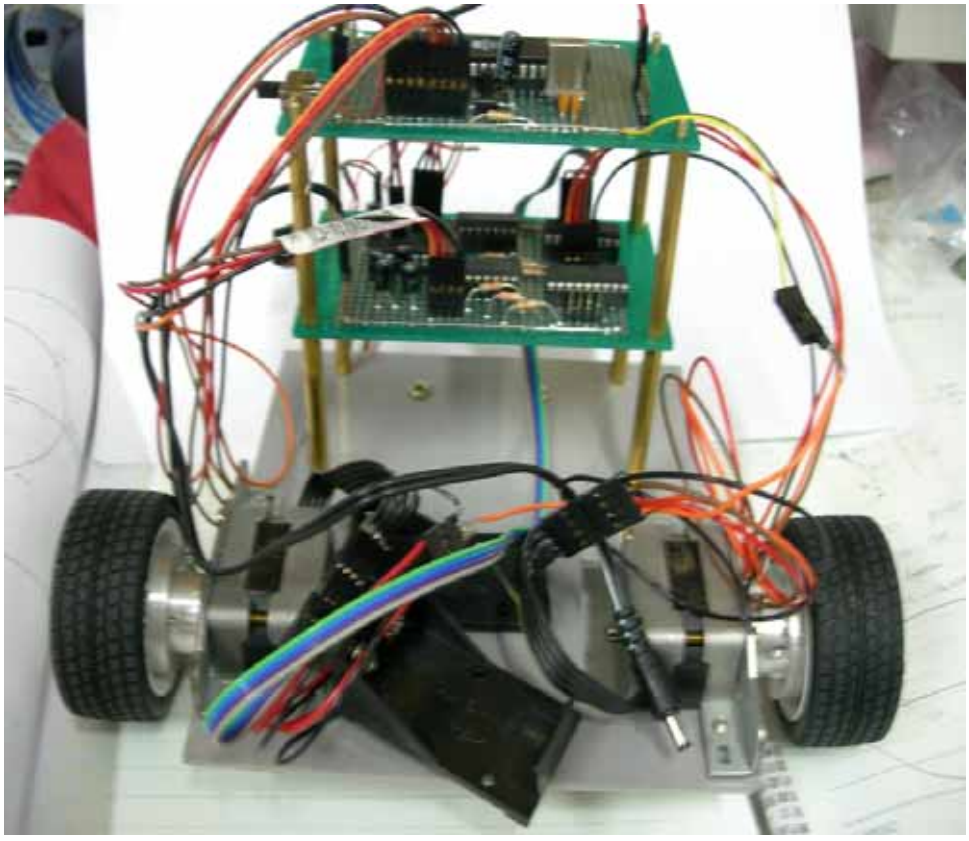


藍芽通訊的串列信號經過 Hin２３２介面處理之後，把串列訊號輸出給 89C51 的 P3.0，P3.1，經過 89C51 的處理之後從 P1 port 輸出八個 bit 的平行訊號。分成兩組(前四個 bit 還有後面四個 bit)，送到兩個反向器以及 2803(放大電流)之後驅動步進馬達。

現在我將就系統的細節來做介紹，首先介紹我們的場地。我們爲了配合網路攝影機的定位，我們利用工廠的材料設計了這個場地(請見下圖)。

場地的下面其實是兩個日光燈管來強化背景跟車體的對比，當初考慮由外界給光但是燈具架設困難而且背景跟目標都會很亮，此外我們使用白報紙包住以平均光源。

下圖爲本組的實體圖。共分成車體，輪子與步進馬達，下層板，上層版。車體是工廠切割鑽孔製作，輪軸也是自己車的。上層板主要是 89C51 以及週邊電路。下層板主要是 Hin232 介面 IC，反相器還有 2802 達林頓對。



5.程式碼(89C51 端)

```
org 0x0
start:   mov R0,#00H     ;馬達正轉出始值
         mov R3,#1    設定輸入一次信號走的步數
         jmp main

Main:
     mov scon, #01010000b  ;    mode 1,  接收資料致能
     mov tmod, #00100000b ;    timer #1, mode 2(timer 初始化)
     mov th1, #-3          ; 9600 baud rate
     mov tl1, #-3
     setb tr1
        jmp    DIRECTION
TRANS:    mov scon, #01010000b       ; mode 1,  使可以接收資料
          ANL PCON, #01111111b       ;timer 初始化
     mov tmod, #00100000b ; timer #1, mode 2
     mov th1, #-3          ; 9600 baud rate
     mov tl1, #-3
     setb tr1
          MOV    IE,#10010000b
          MOV    A,#'C'
          MOV    SBUF,A
DIRECTION:     jnb ri, $              ;等待  PC  送字元過來.
          clr ri                      ;清除 ri 旗標  使
          mov A, sbuf                 ;將收到的字元存入 A
          cjne A, #'R',LEFT           ;以下判斷收到的字原決定須執行的副函式
          call    TURNR
LEFT:          cjne A, #'L',BACK                    判斷轉向
               call    TURNL
BACK:          cjne A, #'B',FORWARD
          call    TURNB
FORWARD:       cjne A, #'F', DIRECTION
          call    STR


TURNR:    MOV    A,R0                    ;A 爲輸出到左輪的暫存器(R0 爲
讀表的指標)
          MOV    DPTR,#TABLE2R           ;依照 TABLE2R 轉動步進馬達
```

```
        MOVC    A,@A+DPTR              ;   把步進馬達的 code 讀到 A
        CPL     A
        MOV     P1,A                   ;       從 P1 輸出
        CALL DELAY                     ;       設定延遲時間的函式
    INC     R0
    CJNE    R0,#4,TESTR                ;判斷是否讀完表中正轉的值
    MOV     R0,#0                      ;若是,則重新設指標爲零
TESTR:  DJNZ    R3,TURNR
        MOV     R3,#1
        CALL    TRANS
TURNL:  MOV     A,R0                   ;把指標的起始值給 A
    MOV     DPTR,#TABLE2L              ;由 DPTR 取表中的值
    MOVC    A,@A+DPTR                  ;由 DPTR 和 R0 把步進馬達的
                                       值給 A
    CPL     A
    MOV     P1,A                       由 P1 輸出
        CALL DELAY
    INC     R0
    CJNE    R0,#4,TESTL
    MOV     R0,#
TESTL:  DJNZ    R3,TURNL
        MOV     R3,#1
        CALL    TRANS
STR   :  MOV    A,R0                   ; 直走程式 請參考 TURNL 或
                                       TURNR 函式
        MOV     DPTR,#TABLE2F
        MOVC    A,@A+DPTR
    CPL     A
    MOV     P1,A

        CALL DELAY
    INC     R0
    CJNE    R0,#4,TESTS
    MOV     R0,#0

TESTS:  DJNZ    R3,STR
        MOV     R3,#1
        CALL    TRANS
```

```
TURNB:   MOV    A,R0                    ;後退的函式
         MOV    DPTR,#TABLE2B
         MOVC   A,@A+DPTR
         CPL    A
         MOV    P1,A
         CALL DELAY
     INC    R0
     CJNE   R0,#4,TESTB
     MOV     R0,#0
TESTB:   DJNZ   R3,TURNB
         MOV     R3,#1
         CALL   TRANS

DELAY:   MOV    R2,#40
D1:      MOV    R4,#248
         DJNZ   R4,$
         DJNZ   R2,D1
     RET

TABLE2F:  DB     33H,99H,0CCH,66H        直走的表
TABLE2B:  DB     33H,66H,0CCH,99H        後退的表
TABLE2L:  DB     33H,69H,0CCH,96H        右轉的表
TABLE2R:  DB     33H,96H,0CCH,69H        左轉的表
END
```

6,討論

這次的預設目標是希望可以達到定位而不單單只是搖控。只是當我們用有線遙控測試 OK 換成藍芽系統的時候出了問題，無法搖控。而定位的程式最後也來不及趕出來。所以最後只剩下有線的電腦遙控車。我們的問題應該是沒有及早測試藍芽系統吧。還有太晚才發現定位程式出了問題。而車體跟場地的方面，就是一些加工方面的技巧還要加強，不過以這次的水準，還算堪用了。電路的部份，雖然焊線技巧比以前進步很多(一次 OK)但是還是有點亂。所幸沒有電路的問題。至於 89C51 端的程式，我們的程式概念本來是由 P0，P1 各自控制一個馬達，程式主體類似指示 DB 表的值不同，也就是

DB 03 06 0C 09 ;正轉

DB 03 09 0C 06 ;反轉

這樣子不只造成程式龐大，測試也有問題。所以我們改用現在的方法，也就是利

用一個 port 的前四個 bit 還有後四個 bit 控制兩個馬達。這樣子不只可以做到輸出完全的同步，也可以減低線路跟程式的複雜度。最後討論的是一個 IC，hin232，因爲我們的系統不是很穩定，平常都 OK，但是就是有些時候會突然掛掉(就像 demo 那樣)，其實除了電池沒電以外，我們從示波器 debug 的時候，有懷疑過是 hin232 的問題，但是因爲測試的時候當機的問題都不嚴重，所以也沒有太在意。沒想到在 demo 的時候出問題。下次應該要朝錯誤率低的目標邁進。

影像擷取方面，這次利用影像二元化的方法，所以在硬體上希望利用底座的燈管造成逆光，使車體更突出，但因爲全自動形的 webcam 會自動調整最適合影像，導致效果不如預期，另外不知道是不是和底座使用日光燈有關，影像上的光源會有閃爍的現象，可能是 webcam 掃描頻率和日光燈頻率不同的關係。另外二元化的方法雖然簡單，但會造成誤將雜訊加入計算的誤差，因此若能加上遮罩、尋找邊緣等其他影像處理方法消除雜訊的影響，不但對位置計算上會更準確，實用性也更高，因爲場地上可以擺上其他物體，而不影響計算，未來要是想要達成一開始定的目標，影響處理的部份一定要朝這部份改進。

訊號傳輸的方式也是一大問題，因爲這次是採取一次一字元代表一步，但是 PC 端和 8051 端的 Delay 設定要一致，才不會像這次 8051 端會遺失訊號，導致與 PC 端所指定的步數不同，目前想到更好的方式，便是將方向、步數一次告訴 8051，使 8051 將資料收齊後再動作，待動作結束後傳回 PC 並等待下次指令，這樣將可以減低訊號遺失的機率，系統資料傳輸的負荷量也可大大降低，甚至未來可以使用好的演算法，將路徑規畫好後傳給 8051，PC 端只需作監控進行微調，這樣車子移動可以更平順，若是有障礙物或是進行停車等目標，此種方法也能使 PC 端的運算更有效率。

7.結論

這次的專題我們野心不小，也付出了相當多的心血，雖然最後只完成了一半，但是嘗試的過程跟製作成品付出的心血當讓大家得到了寶貴的實作經驗。而且那種熬夜趕工，最後做出成果的喜悅，真的是難以形容。