競速自走車

利基應用科技
freddie@innovati.com.tw
www.innovati.com.tw

競速自走車影片

循跡自走車

Line Tracer

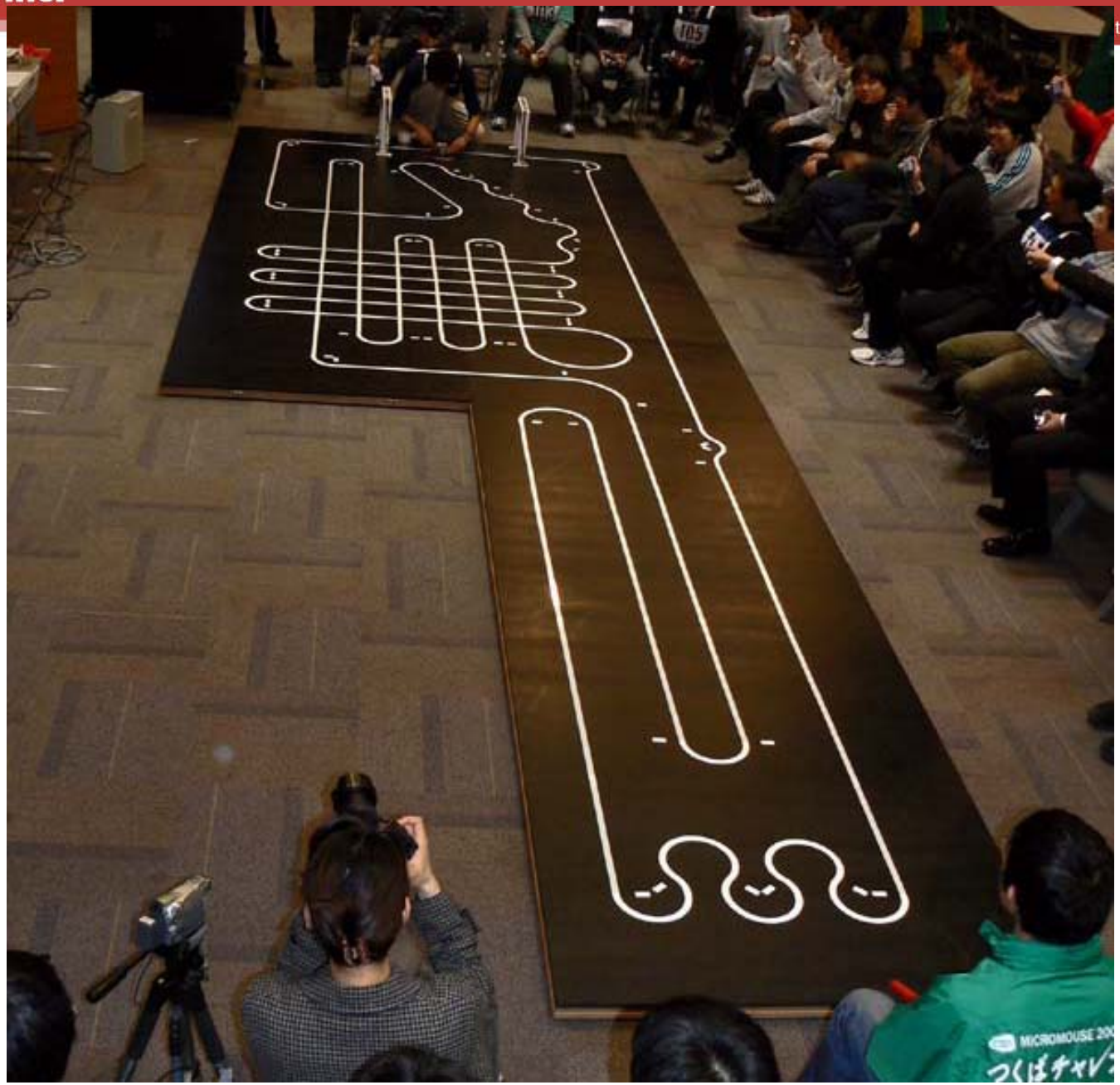
Singapore Robo Grand Prix 2009

2009年第五屆人工智慧單晶片競速自走車決賽

2009 日本RoboTracer

2009 日本自走車比賽

競速場地說明


105
MICROMOUSE

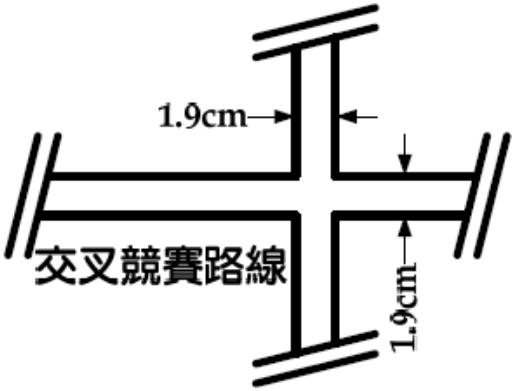
競賽場地說明

1. 競賽場地之表面為黑色，競速路線則是使用 1.9 公分寬的白色條紋來標示。

2. 競速路線是由圓弧與直線所組成，圓弧的最小半徑為 15 公分。

3. 相同曲率的圓弧至少有 15 公分以上才會改變曲率。

4. 競速路線的總長度不會超過 60 公尺。競速路線可能會交叉(交叉的角度為 90±5 度，請參考右圖)，但競速自走車在競速路線交叉的地方必須直行。



5. 競速路線的起點與終點會在同一個直線區域上，而且終點在起點的後方 100 公分處。沿著競速路線的方向的右側，在起點與終點處都會有「記號」。在起點線與終點線的記號處也都各會有一個標示「START」與「GOAL」內徑寬 40 公分、高 25 公分的門。在起點與終點之間的區域稱為「起始與終點區」(請參考圖 1a-c 說明)。

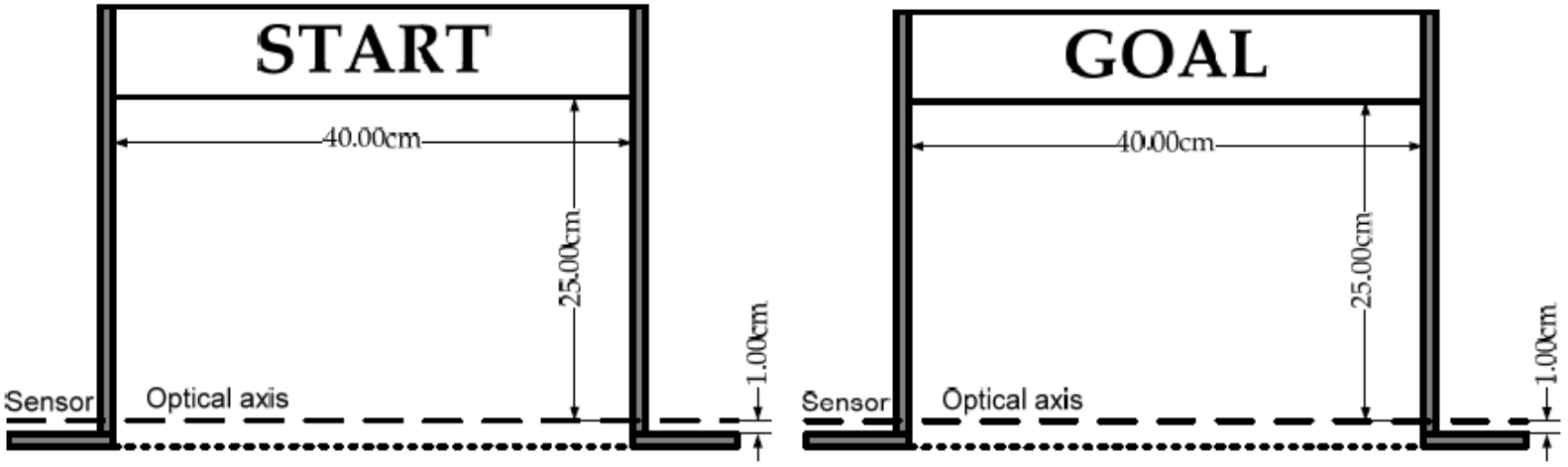


圖 1 **(a)** 「START」與「GOAL」內徑寬 40 公分、高 25 公分的門

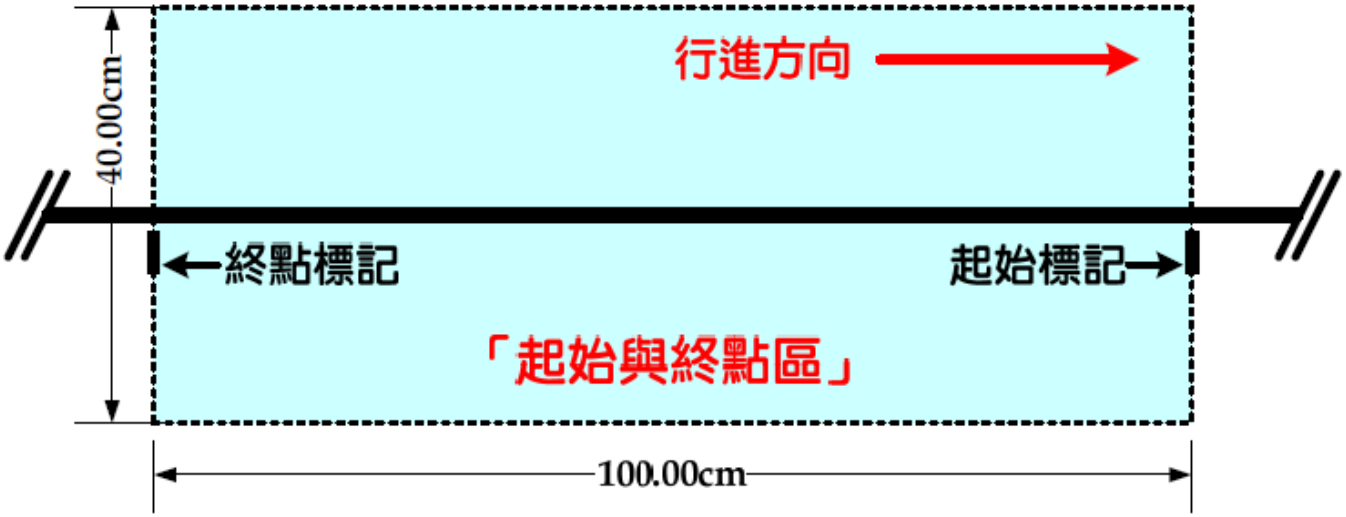


圖 1 **(b)** 在起點與終點之間的「起始與終點區」

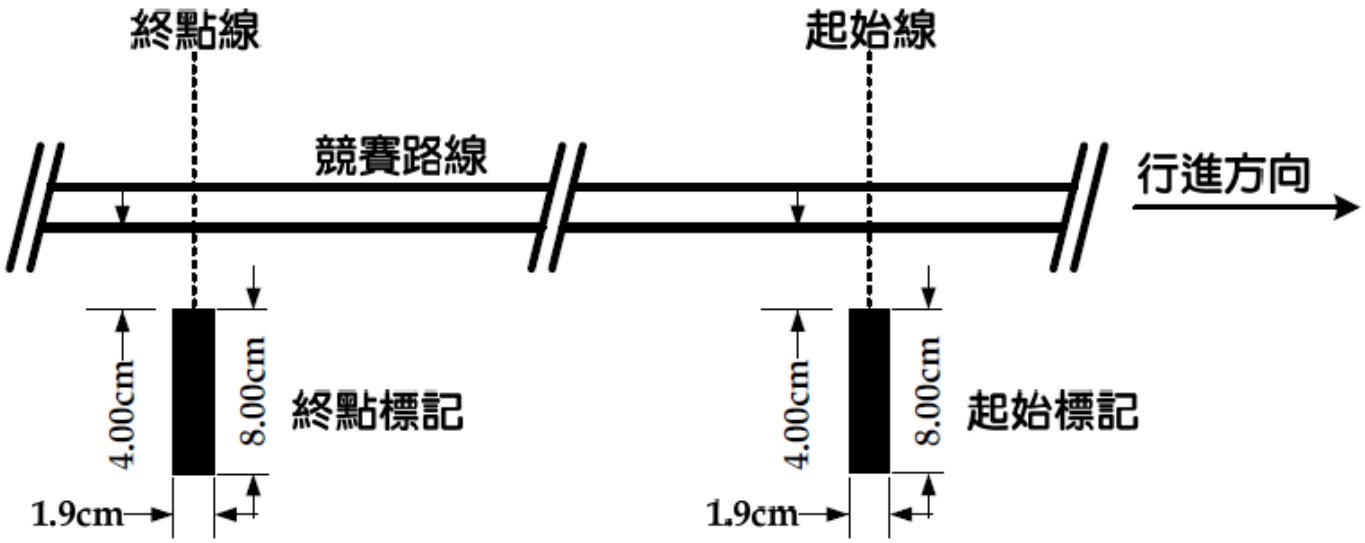


圖 1 (c) 在「起始與終點區」中的起始與終點標記

6. 競速路線上距離起點與終點 25 公分以內的路線，或是距離路線交叉點 25 公分以內的路線都是直線。

7. 競速路線上發生曲率變化路線的起始位置與終止位置，都會在沿著競速路線的方向的左側以記號標示(請參考圖 2 說明)。

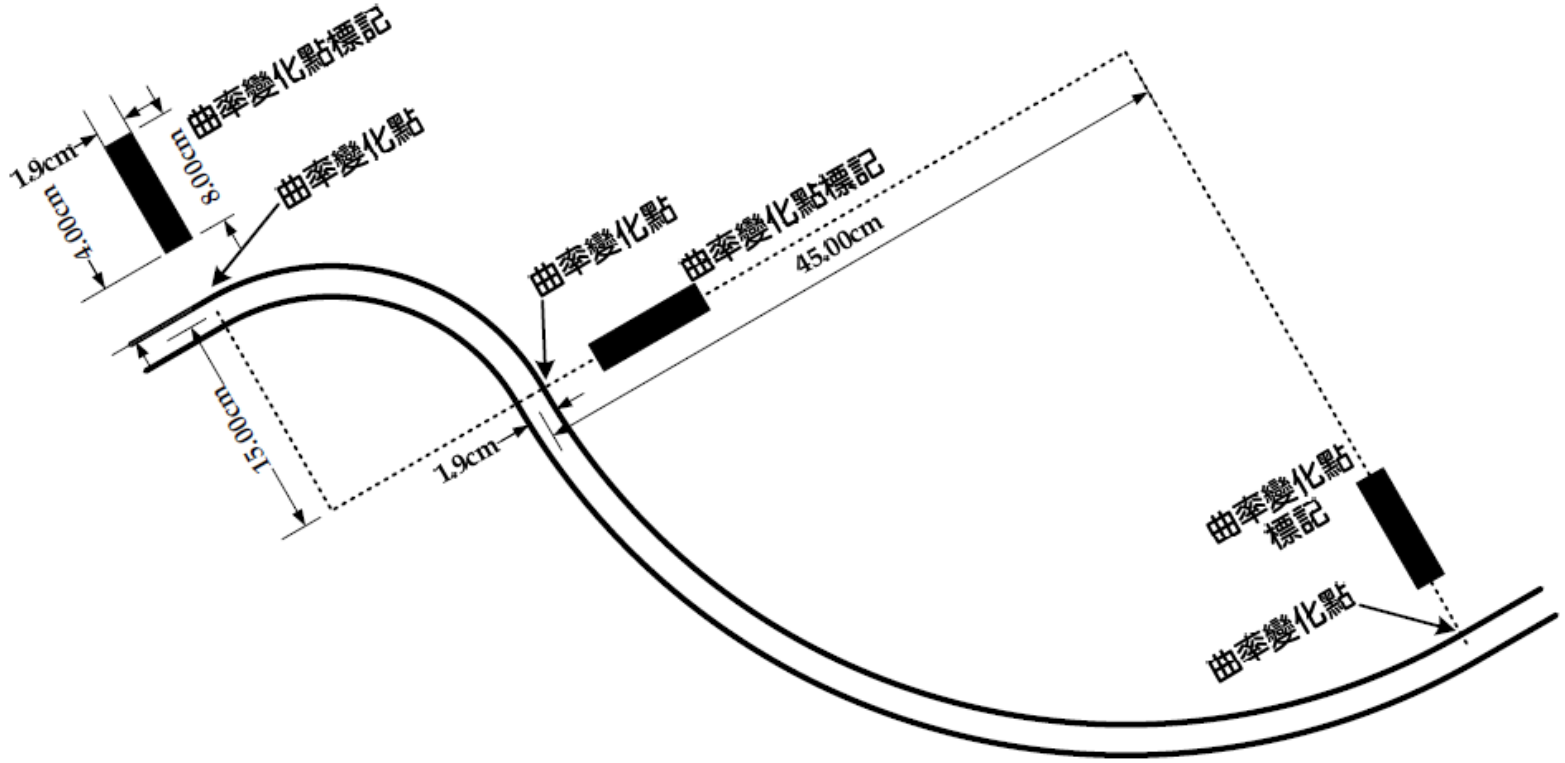
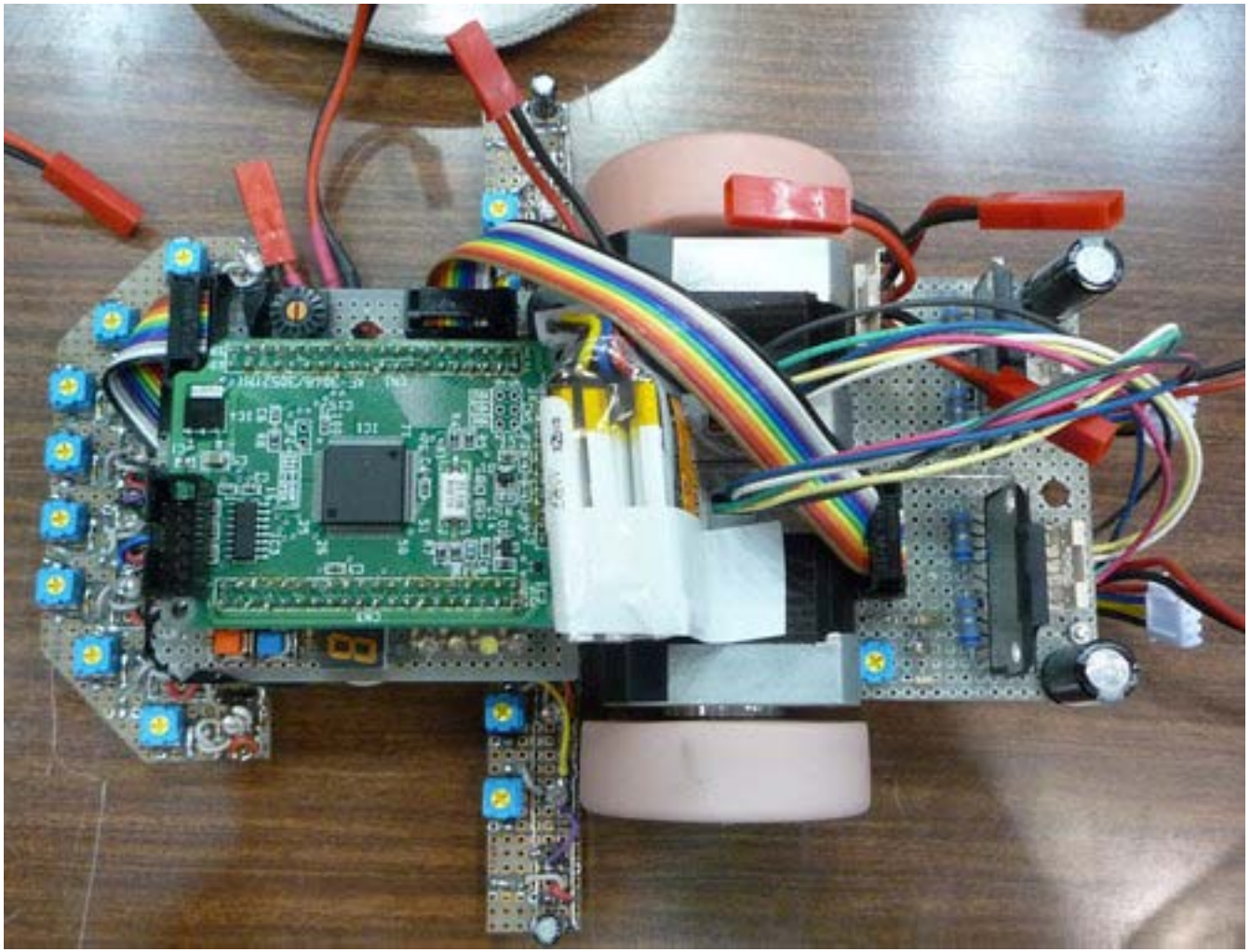
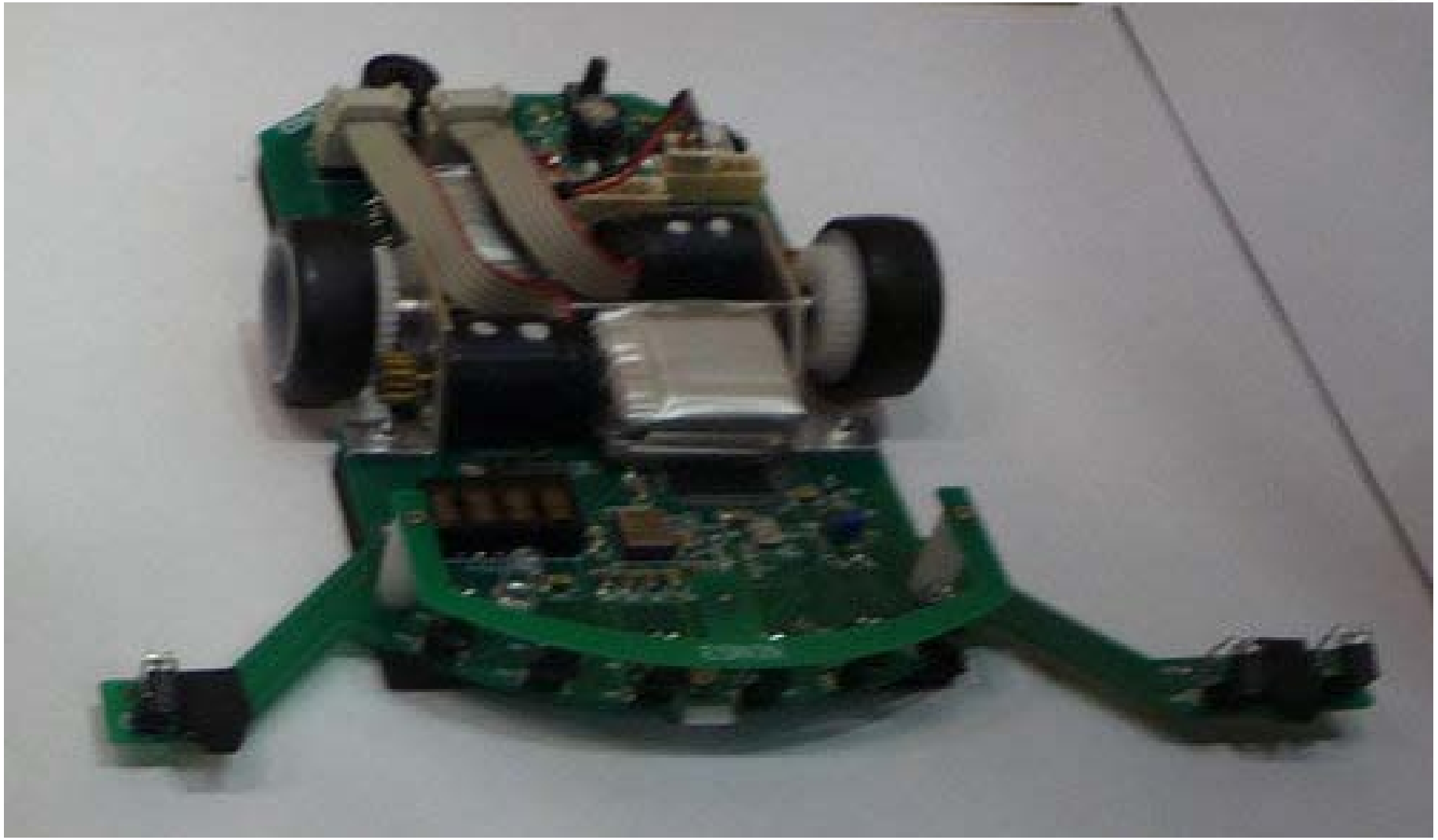


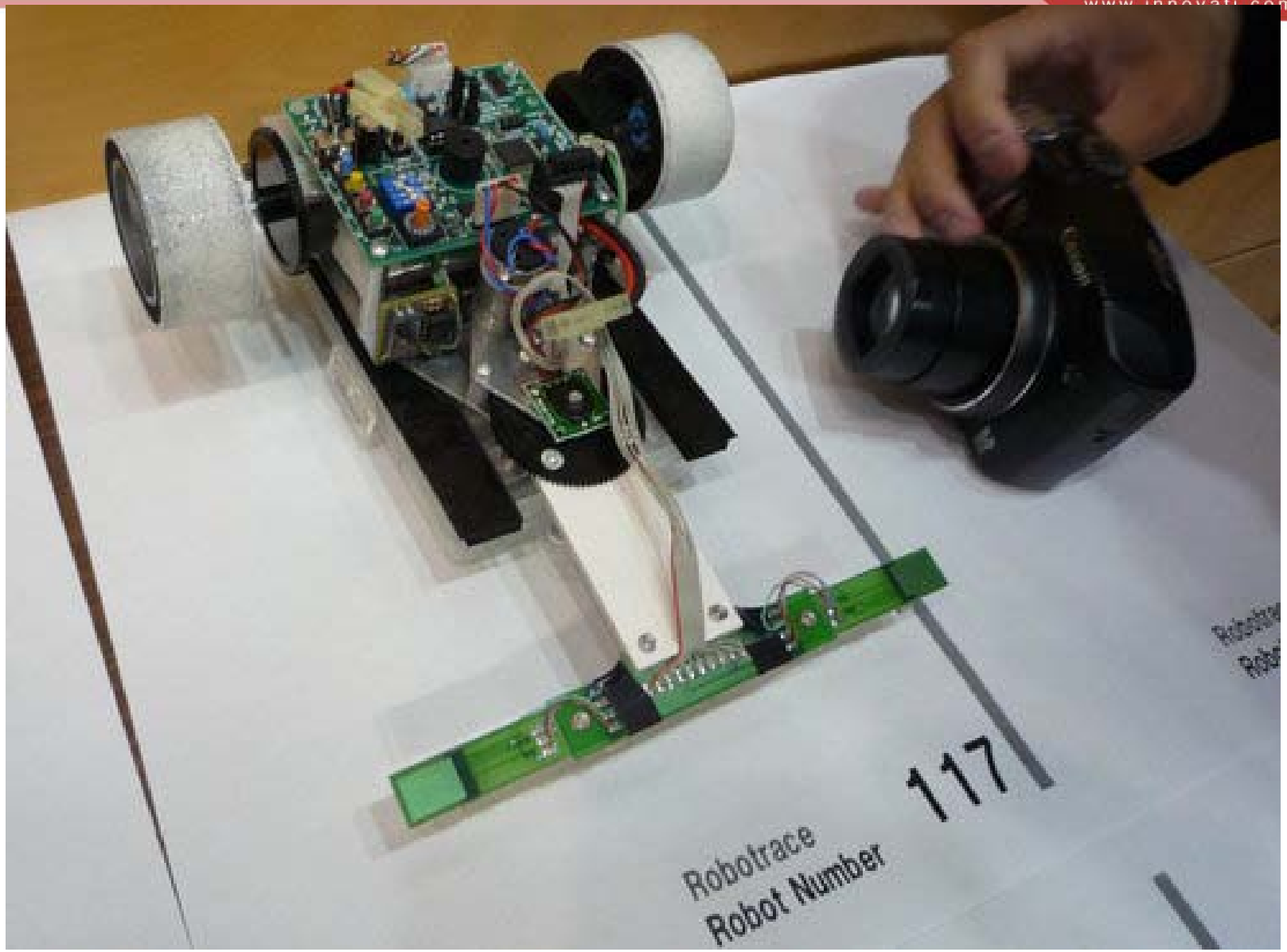
圖 2 競速路線示意圖

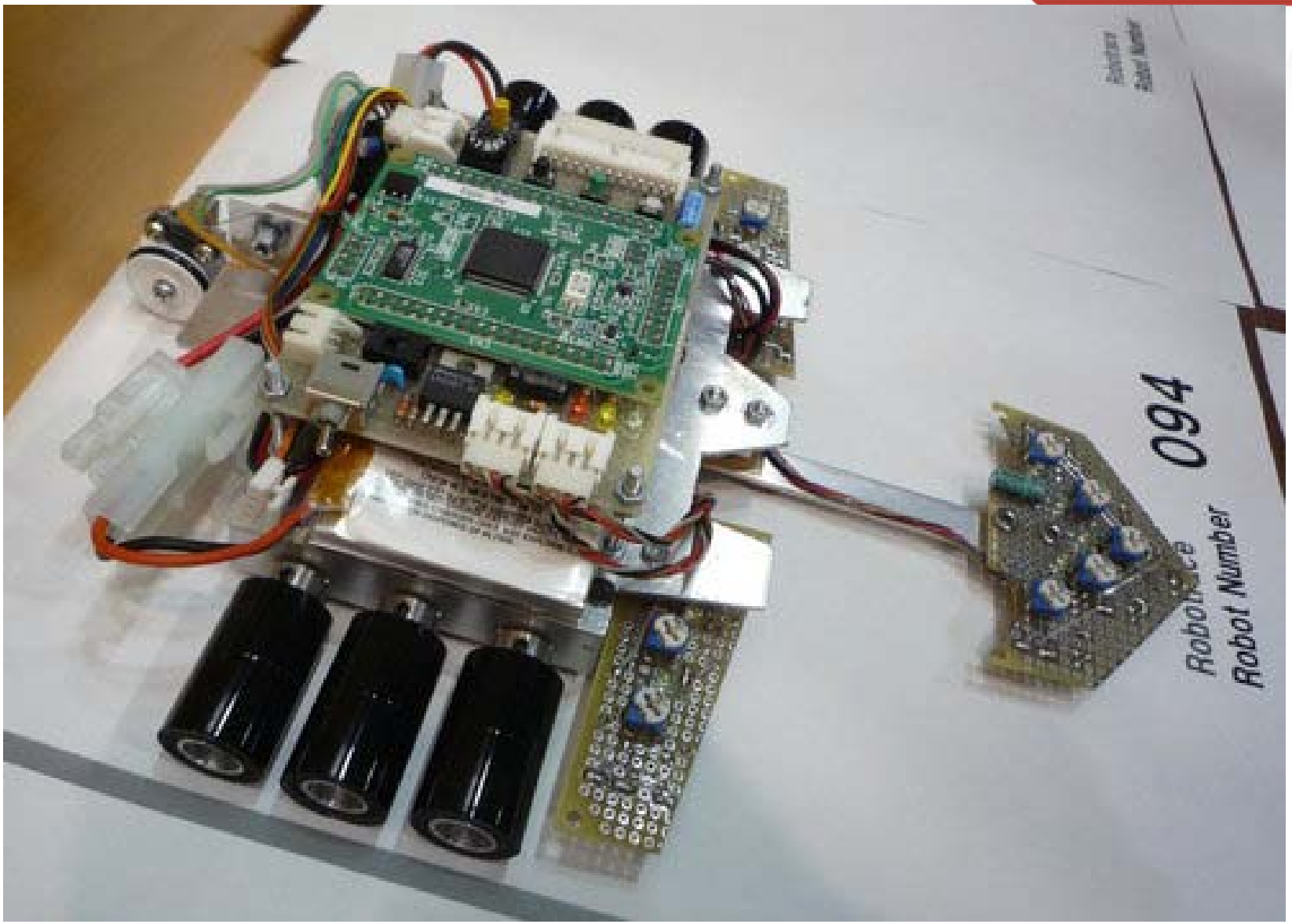
8. 比賽場地大都是水平的，但某些部分可能有至多 5 度的傾斜。

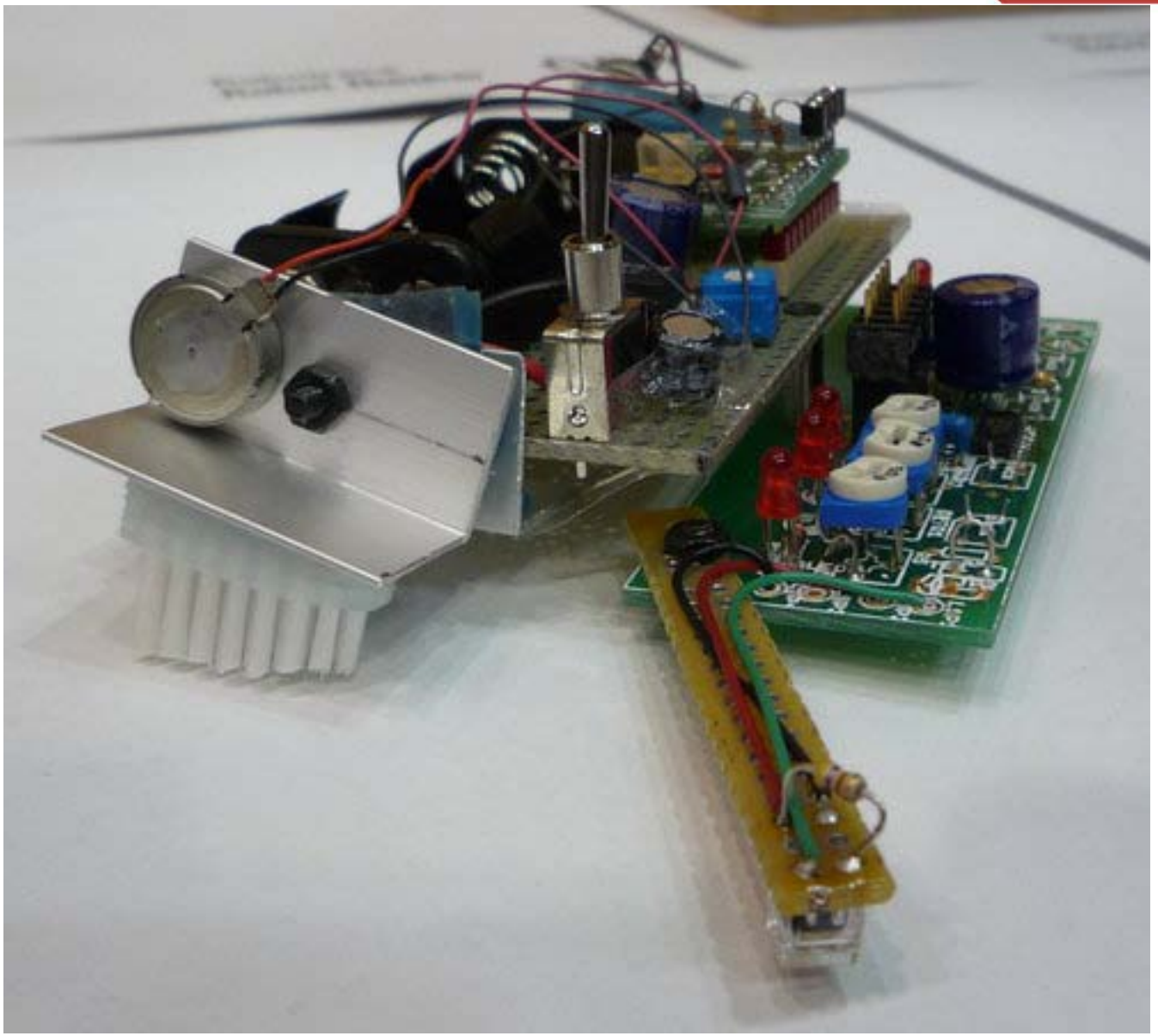
自走車設計

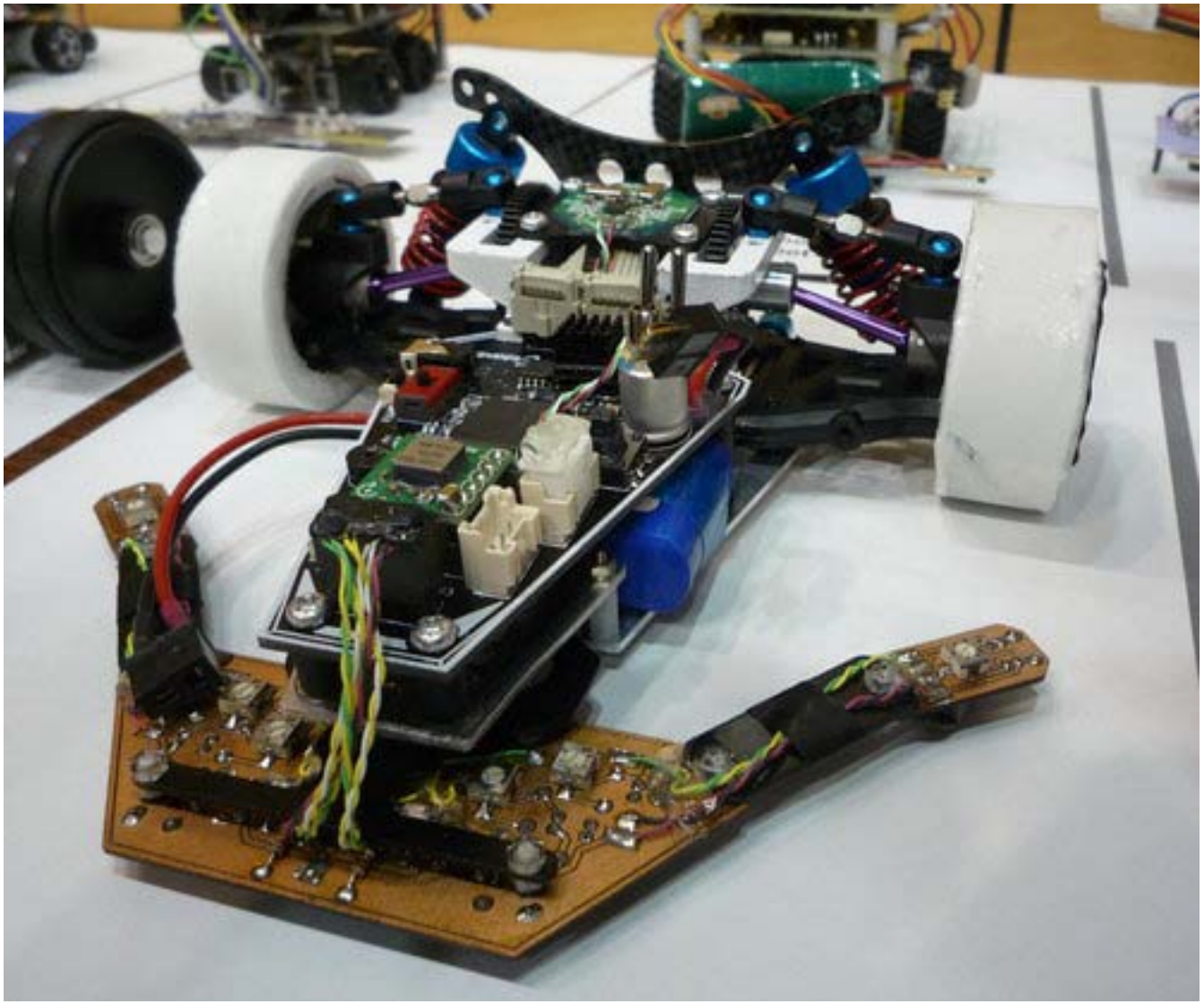




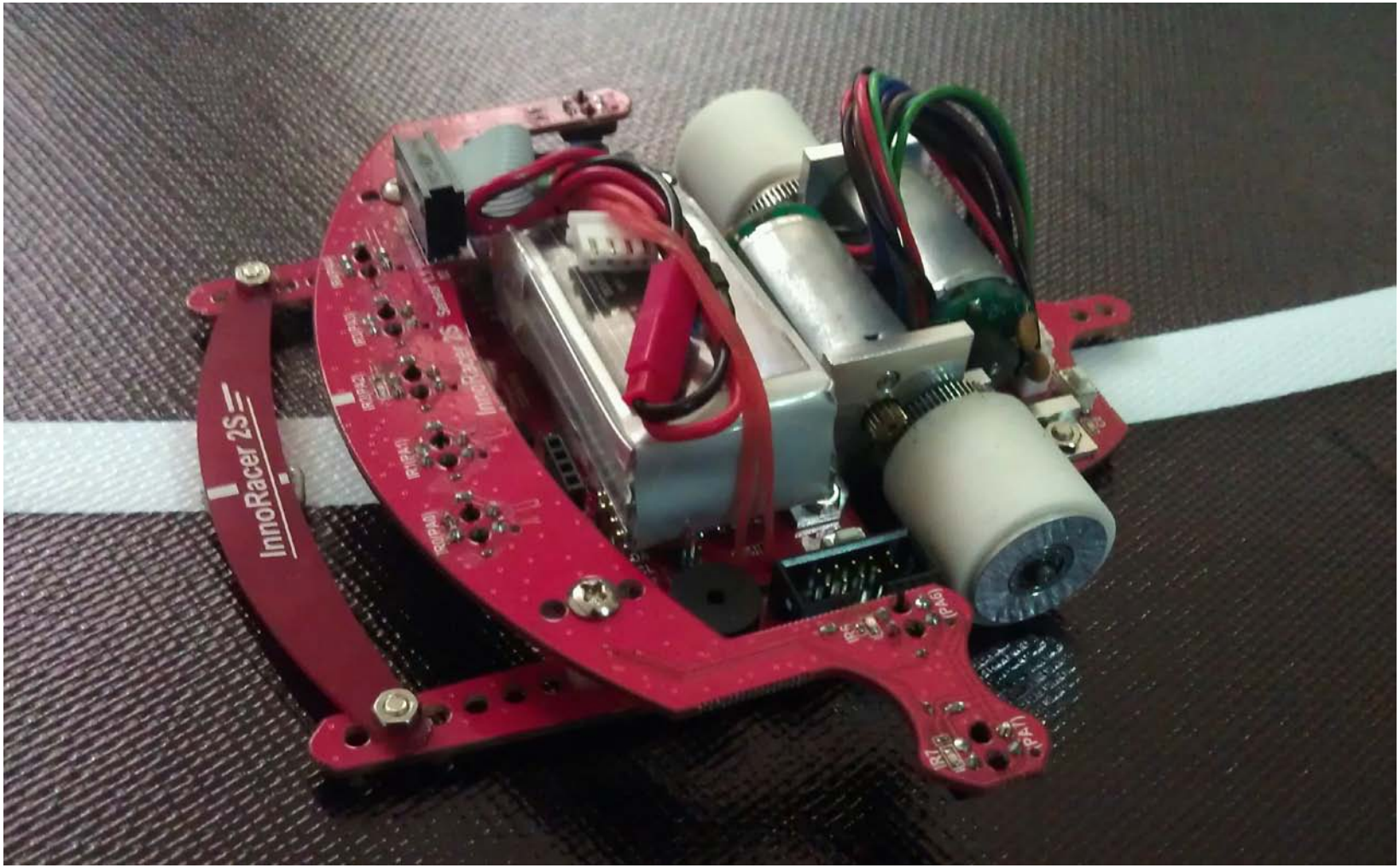

Robotrace
Robot Number
117

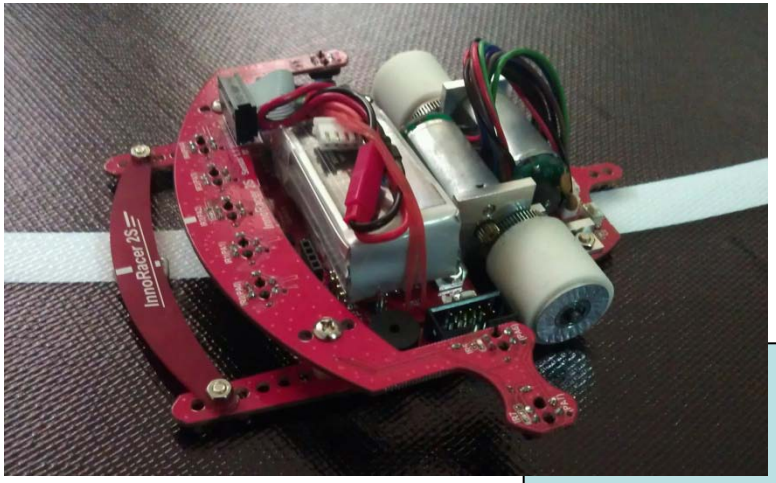

094
Robot Number





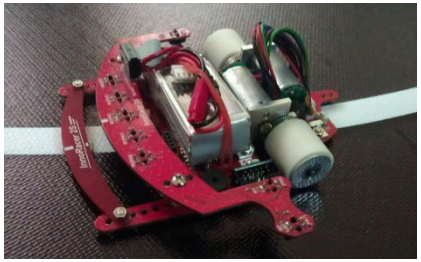
認識InnoRacer


InnoRacer 2S

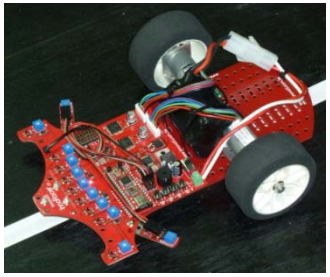


InnoRacer2s 簡介:

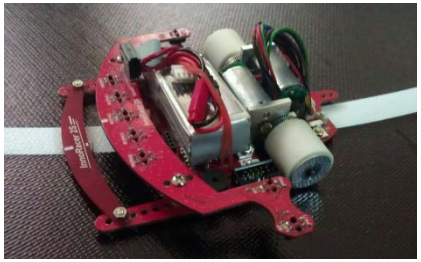
1. Arminno 核心
2. USB-LINK程式下載
3. C 語言及函數指令控制
4. 提供數位及類比PID控制
5. 具馬達轉速回饋及過載保護功能
6. 跑道記憶功能
7. 內建G-Sensor,Gyro
8. 額外cmdbus模組擴充功能
9. 適合教育學習及比賽用車



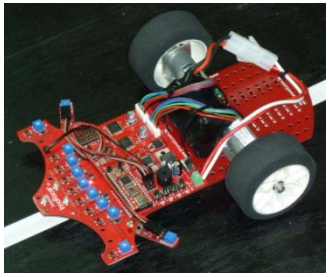
Innoracer 2S 與Innoracer 基本架構比較



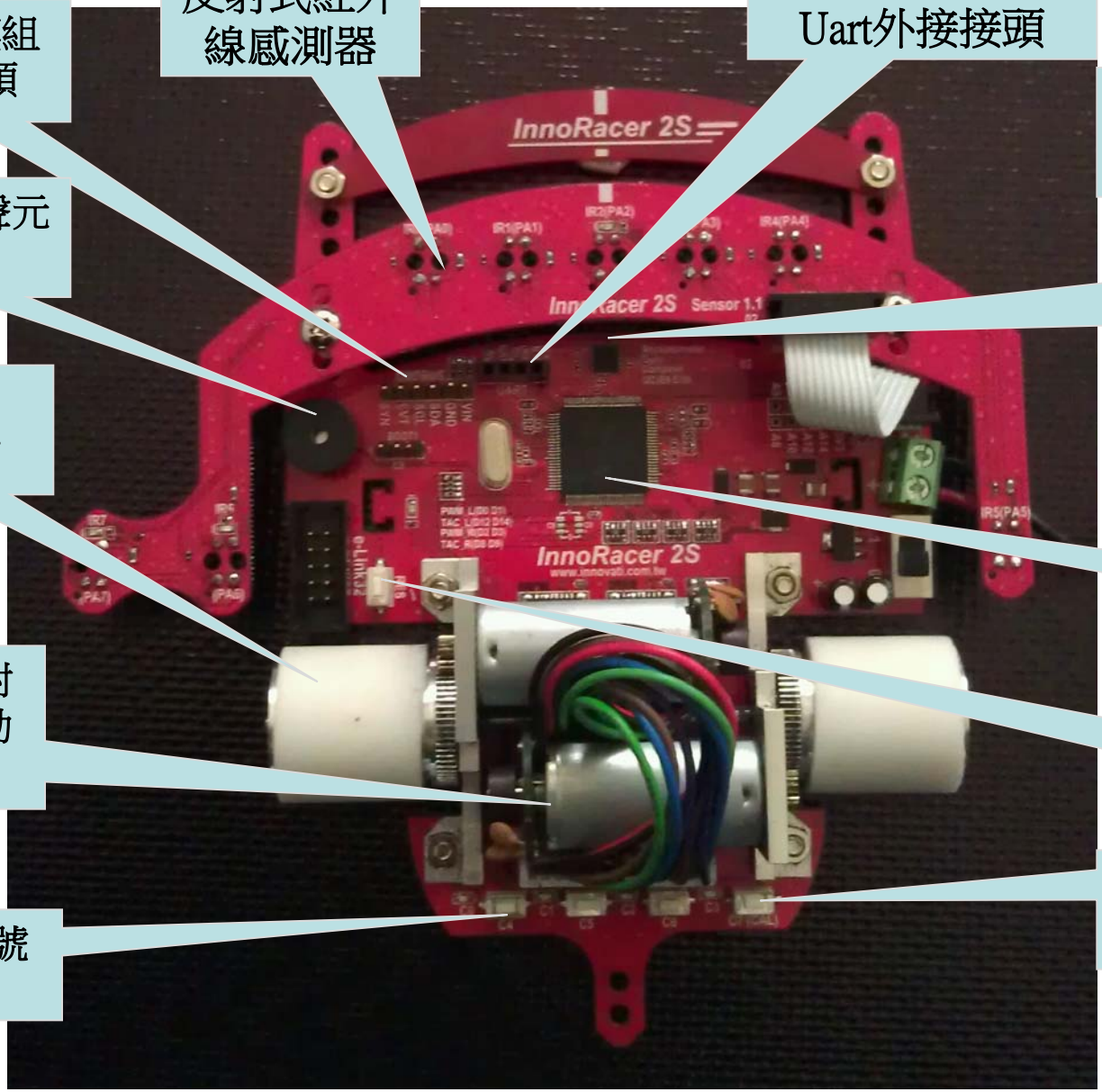
	Innoracer 2S	Innoracer
車體結構(cm)	長x寬 15x15 輪距x感測軸距 7.5x8	長x寬 20x18.5 輪距x感測軸距 12x16
車重(含電池/克)	280	435
處理器速度(MCU)	ARM Cortex M3 32位元, 72MHz	PIC like MCU 8位元, 8MHz
輪胎材質	矽膠胎	海棉胎
Motion感測器	Accelerometer, Gyro	Accelerometer
使用語言	C語言 Open source code	Basic語言 固定M1及P1
電池種類	11.1V 850mah LiPo	10.8V 700mah NiMh



Innoracer 2S 與Innoracer 軟體層架構比較



	Innoracer 2S	Innoracer
IR感測器感測速度	2000Hz	1000Hz
PID處理速度	500Hz	333Hz
Motion感測資料	Accelerometer, Gyro	Accelerometer
速度控制	PWM及絕對速度單位控制	PWM控制
提示點感測器數量	2	1
交叉路口偵測	Cross distance	Cross time
Motor Dead zone 控制	YES	NO
電池電壓偵測	YES	NO



Cmdbus模組外接接頭

反射式紅外線感測器

Uart外接接頭

12V 850mah LiPo電池

Buzzer發聲元件

G-Sensor Gyro 感測器

競賽用矽膠胎

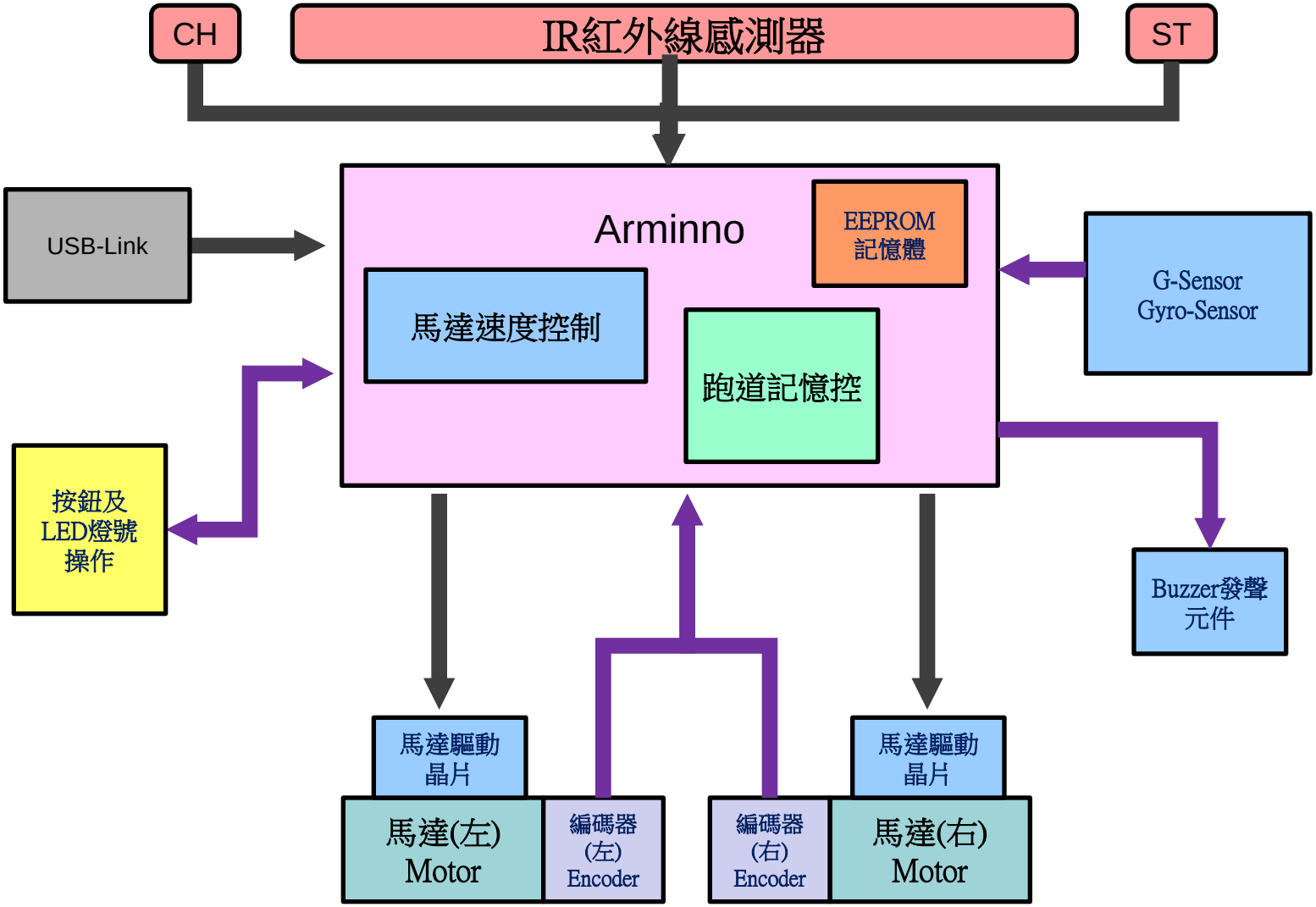
Arminno核心

18000rpm 附編碼器驅動馬達

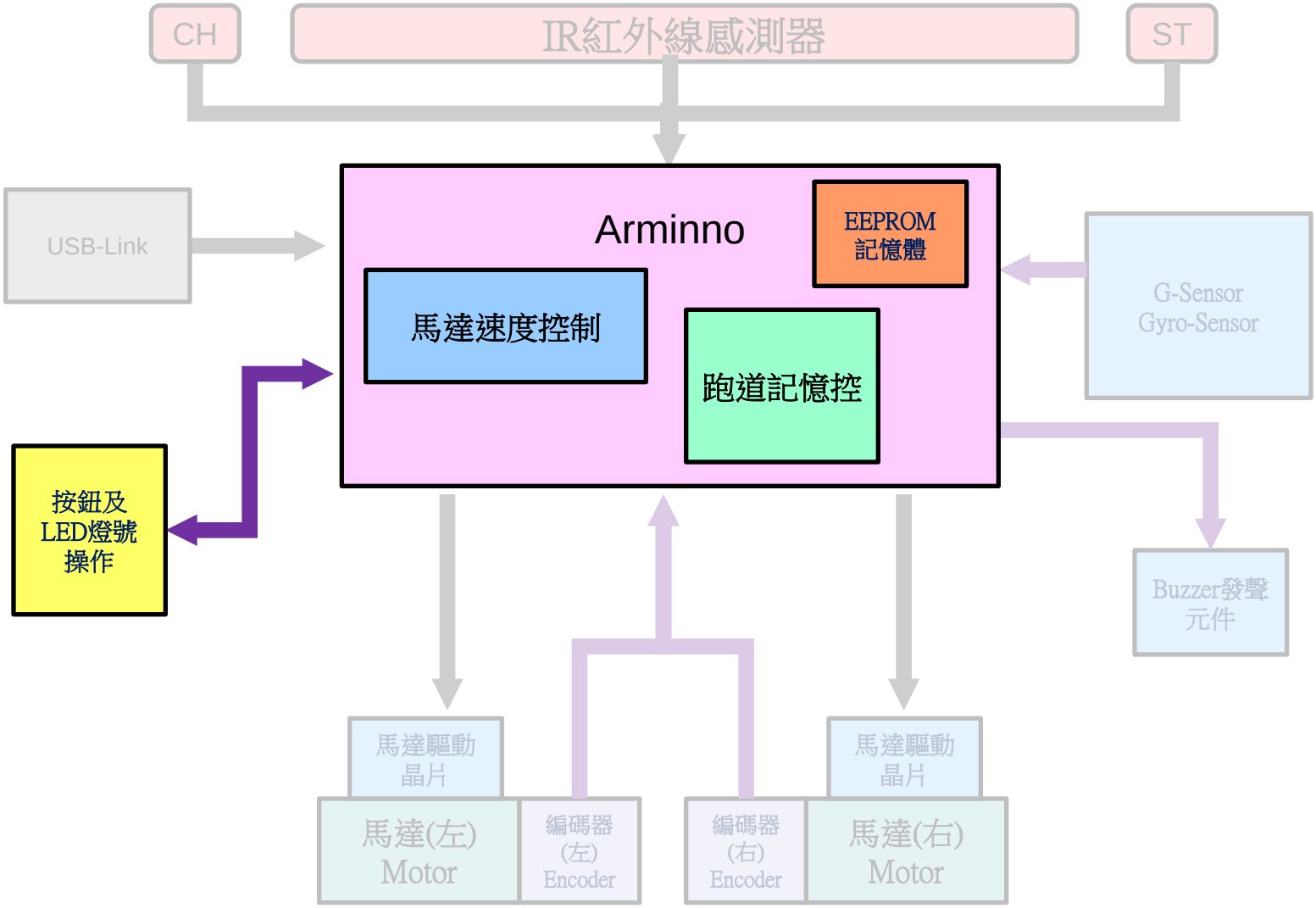
系統重置按鈕

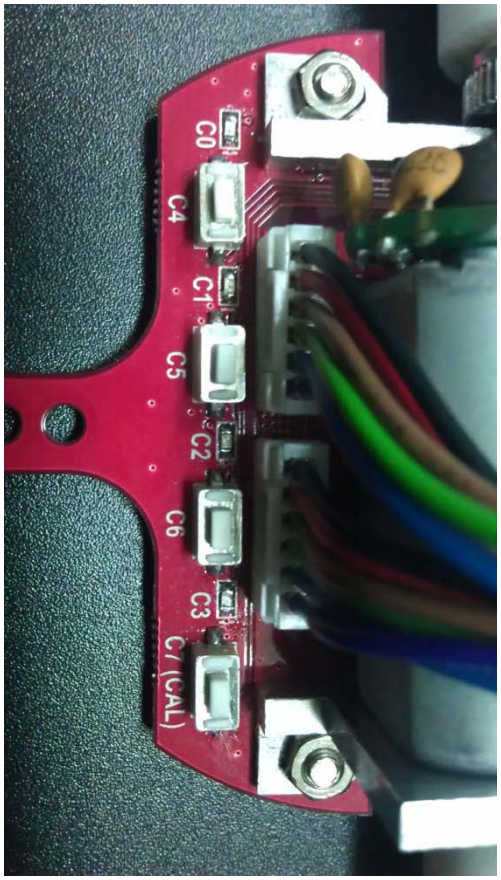
感測器校正按鈕

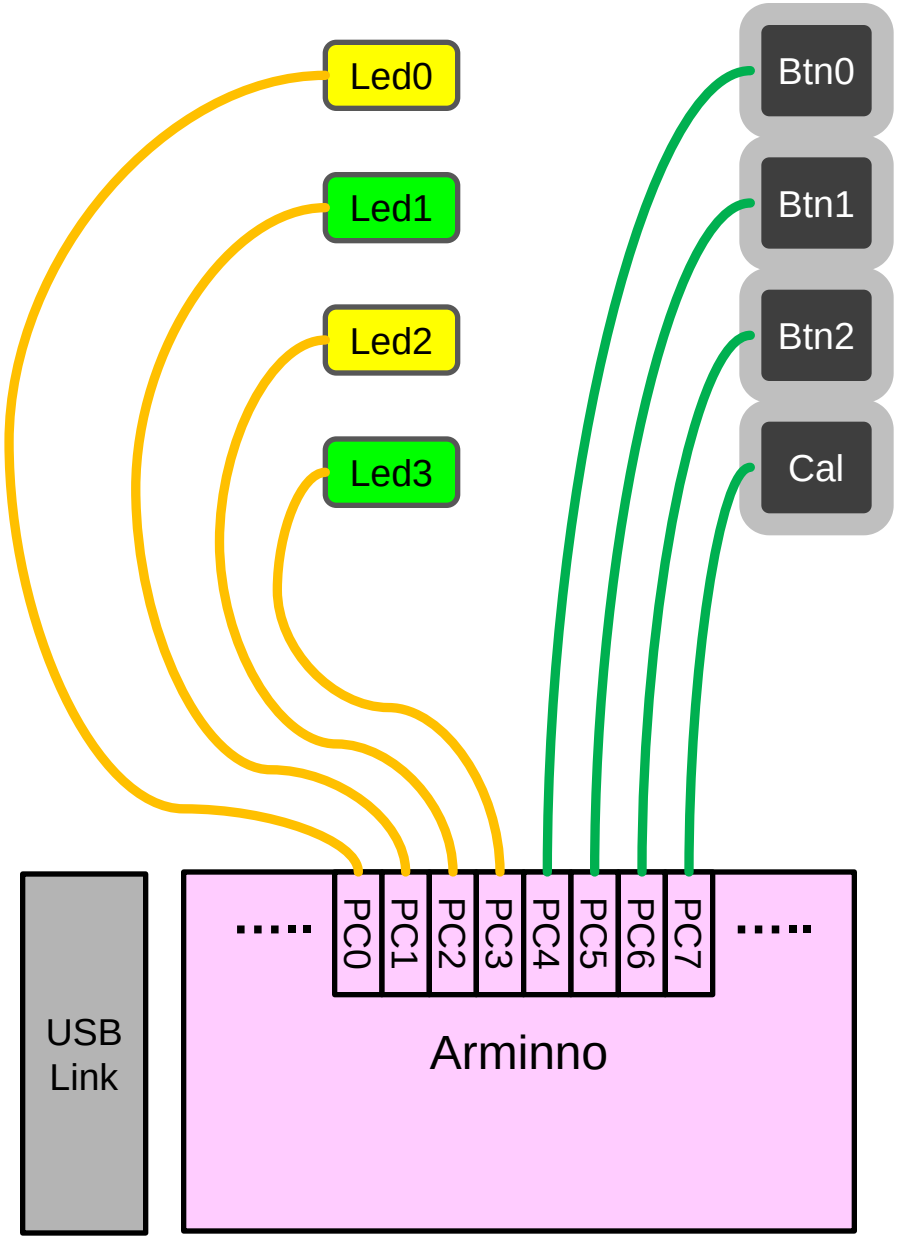
按鈕及燈號操作


CH
IR紅外線感測器
ST
USB-Link
Arminno
EEPROM
記憶體
馬達速度控制
跑道記憶控
G-Sensor
Gyro-Sensor
按鈕及
LED燈號
操作
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)
Encoder
編碼器
(右)
Encoder
馬達(右)
Motor

按鈕及LED燈號操作


CH
IR紅外線感測器
ST
USB-Link
Arminno
EEPROM
記憶體
馬達速度控制
跑道記憶控
G-Sensor
Gyro-Sensor
按鈕及
LED燈號
操作
Buzzer發聲
元件
馬達驅動
晶片
馬達(左)
Motor
編碼器
(左)
Encoder
編碼器
(右)
Encoder
馬達驅動
晶片
馬達(右)
Motor


C0
C4
C1
C5
C2
C6
C3
C7 (CAL)


Led0
Led1
Led2
Led3
Btn0
Btn1
Btn2
Cal
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
USB Link
Arminno

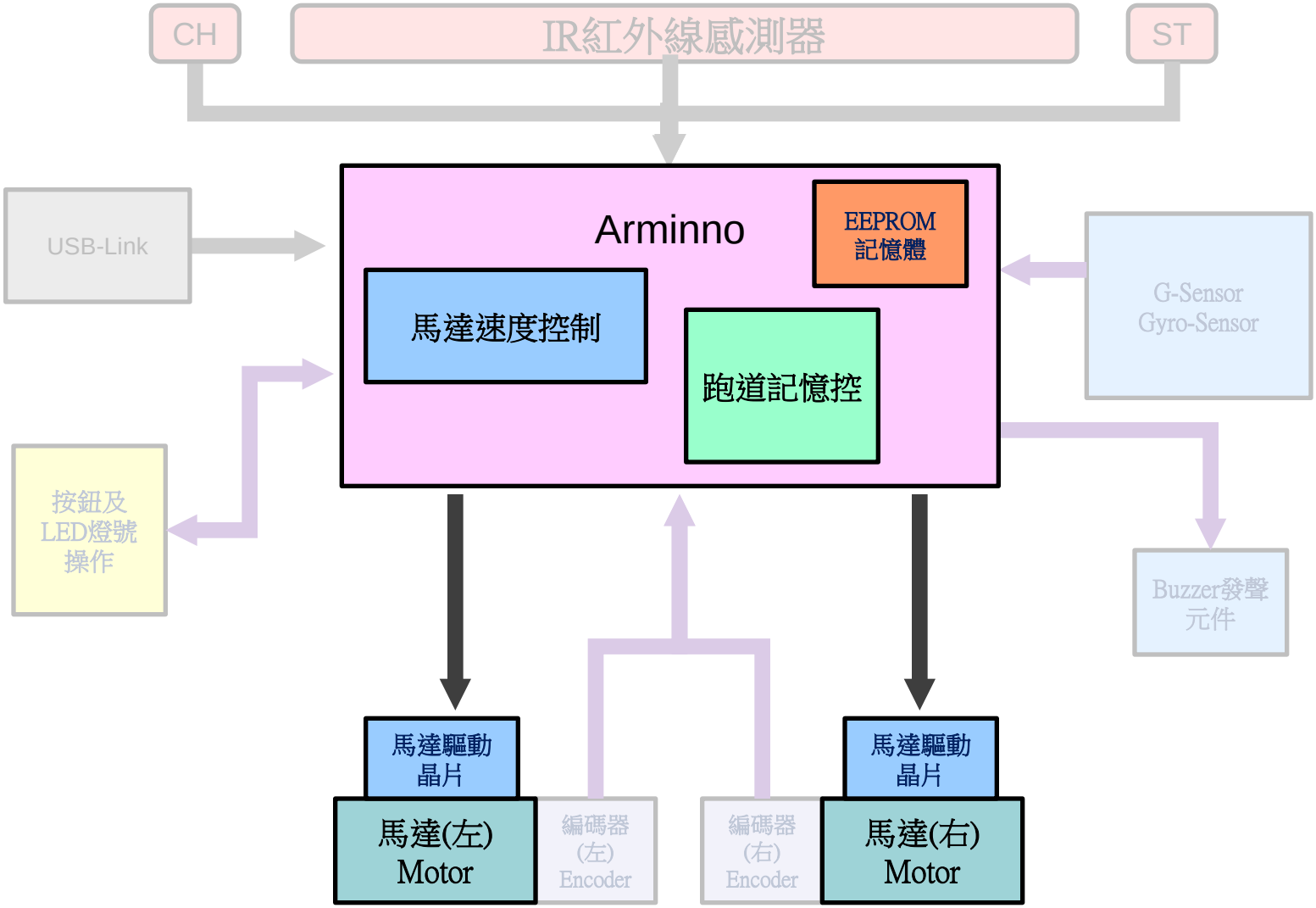
實驗一: LED控制

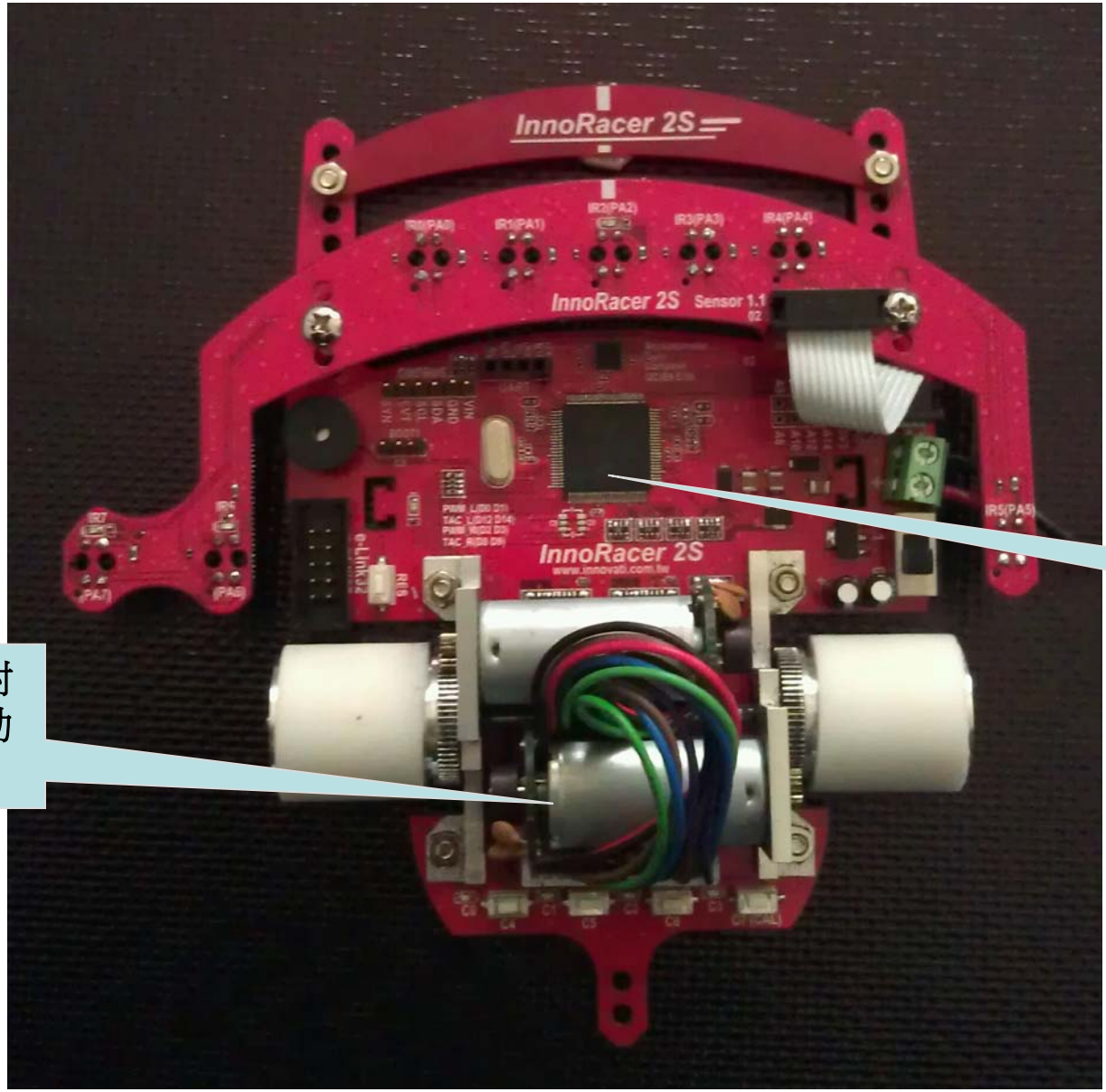
```
//說明：點亮與熄滅LED
#include "arminno.h"
#include "innoRacer2.h"
innoRacer2 myRacer;

int main(void)
{
        //無窮迴圈，反覆亮滅LED
        while(1)
        {
                myRacer.Led0On( ); //點亮LED 0
                Pause(5000);       //暫停0.5秒
                myRacer.Led0Off(); //熄滅LED 0
                Pause(5000);       //暫停0.5秒
        }
}
```

實驗二: 按鈕與LED

```
//說明：偵測按鈕點亮與熄滅LED
#include "arminno.h"
#include "innoRacer2.h"
innoRacer2 myRacer;

int main(void)
{
        //無窮迴圈
        while(1)
        {
            if(myRacer.GetButton0State()==0)    //判斷按鈕狀態
              myRacer.Led0On();                 //點亮LED 0
            else
              myRacer.Led0Off();                //熄滅LED 0
        }
}
```

如何控制馬達


CH
IR紅外線感測器
ST
USB-Link
Arminno
EEPROM
記憶體
馬達速度控制
跑道記憶控
G-Sensor
Gyro-Sensor
按鈕及
LED燈號
操作
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)
Encoder
編碼器
(右)
Encoder
馬達(右)
Motor



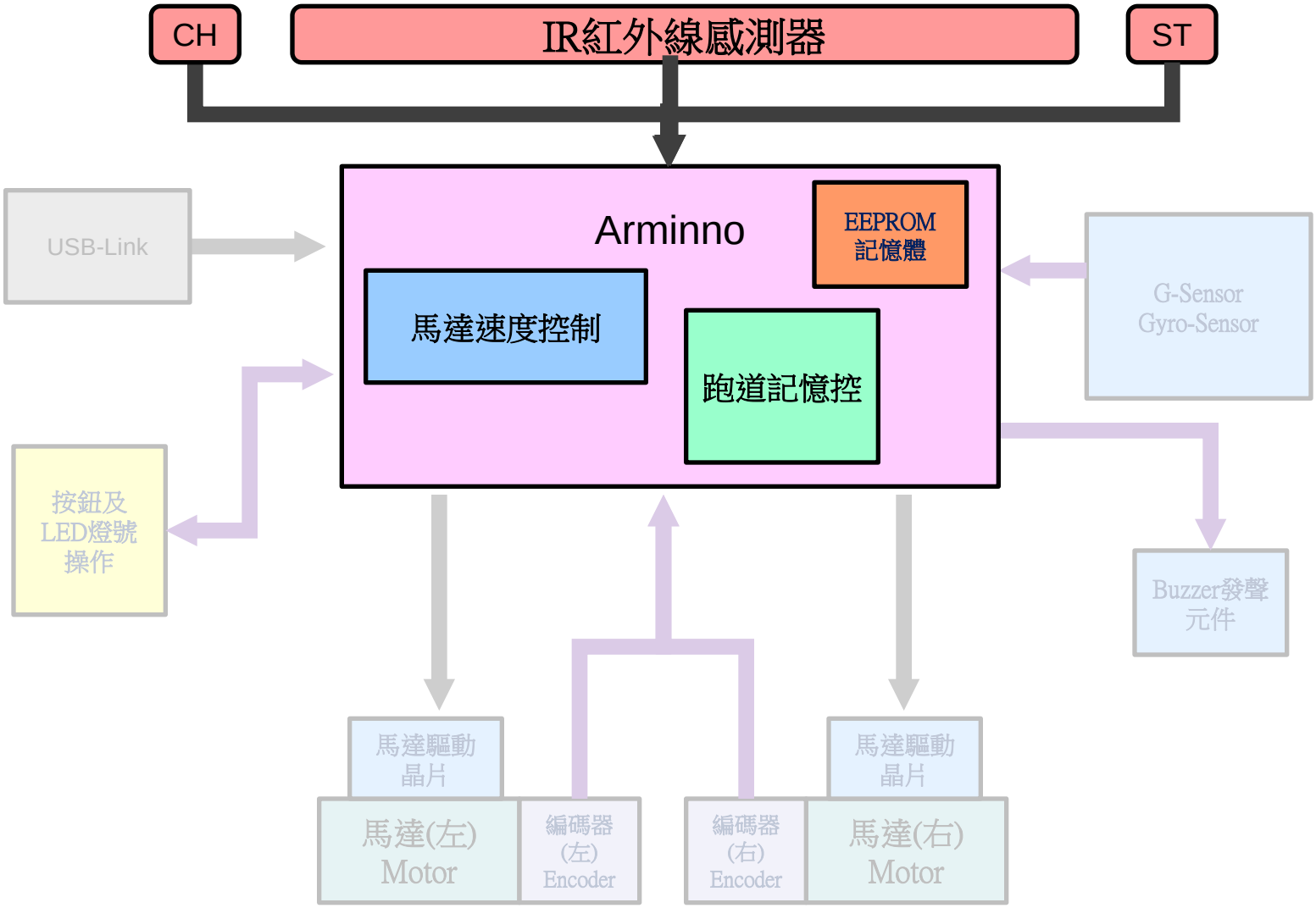
Arminno核心

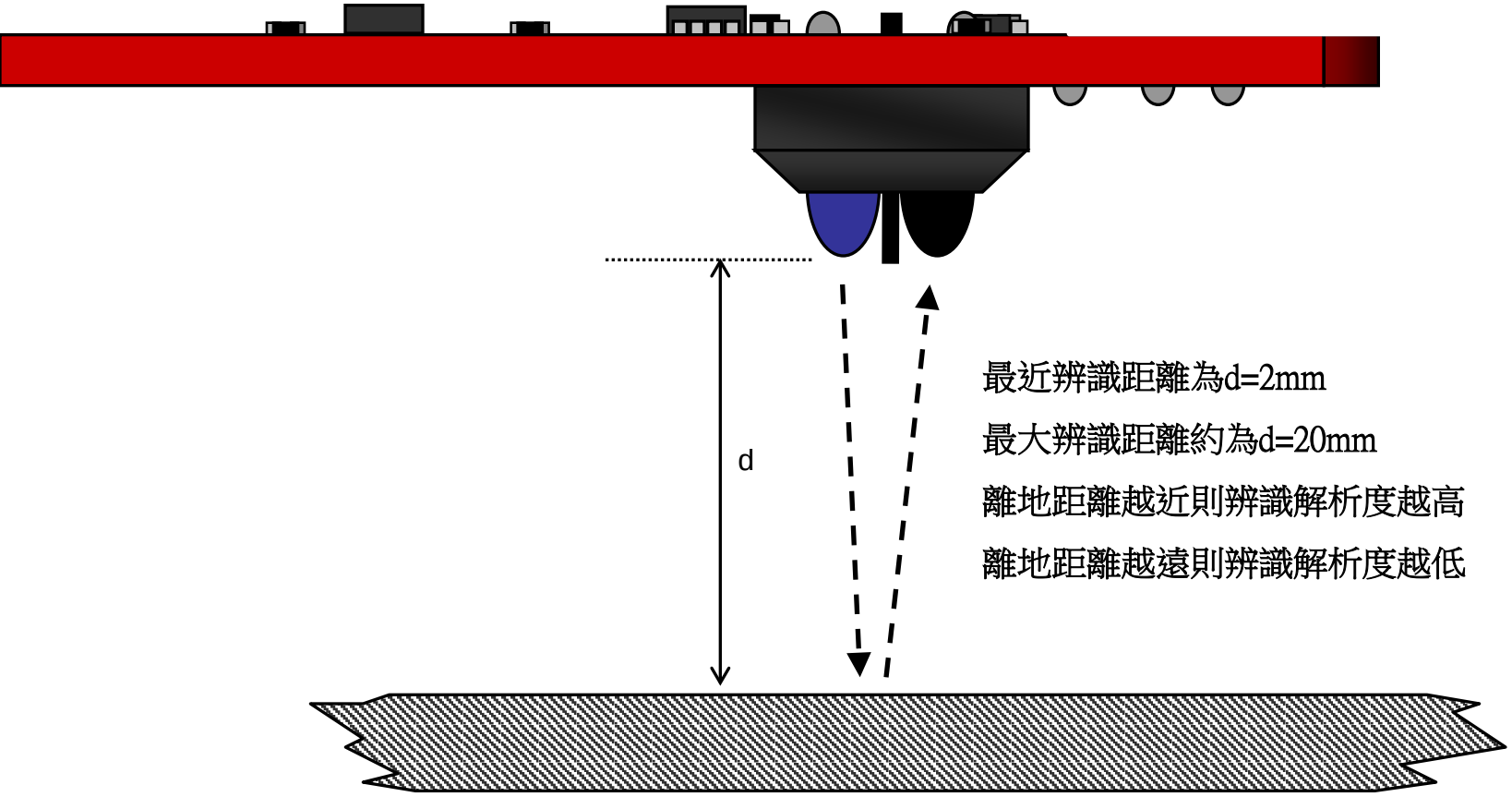
18000rpm 附編碼器驅動馬達

實驗三: 馬達控制

```
‘//說明：依照輸入值，控制馬達方向與轉速
#include "arminno.h"
#include "innoRacer2.h"
innoRacer2 myRacer;

int main(void)
{
        int iVelL,iVelR; //左右輪參數
        while(1)
        {
            printf("\033[J\n");                          //清除視窗文字
            printf("Please enter the value of iVelL and iVelR... \n");
           scanf(“%d%d”,&iVelL,&iVelR);                  //輸入左右輪參數
           printf("L:%d R:%d\n",iVelL,iVelR);            //顯示輸入值
           myRacer.SetVelLR(iVelL,iVelR);                //設定左右輪轉速與方向
           printf("\nPress Any Key to Stop...\n");
           scanf("%d",&iVelR);                           //等待輸入任意值
           myRacer.BrakeDual();                          //停止兩輪轉動

        }
}
```

IR紅外線感測器


CH
IR紅外線感測器
ST
Arminno
EEPROM
記憶體
馬達速度控制
跑道記憶控
USB-Link
G-Sensor
Gyro-Sensor
按鈕及
LED燈號
操作
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)
Encoder
編碼器
(右)
Encoder
馬達(右)
Motor



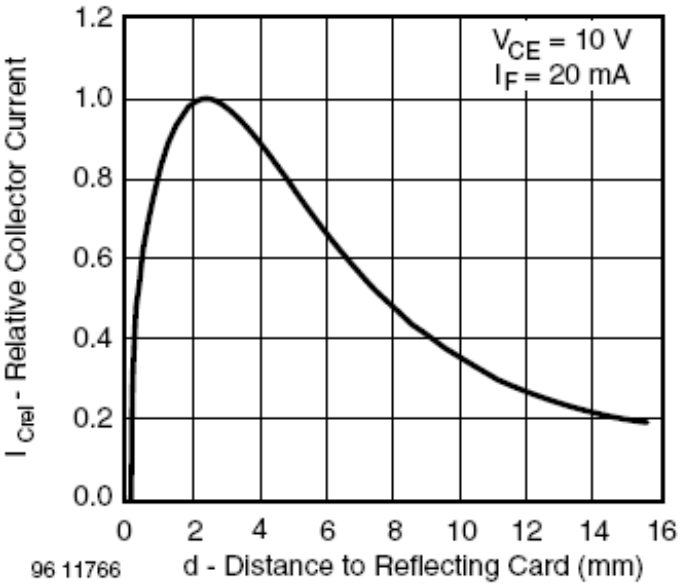
最近辨識距離為d=2mm

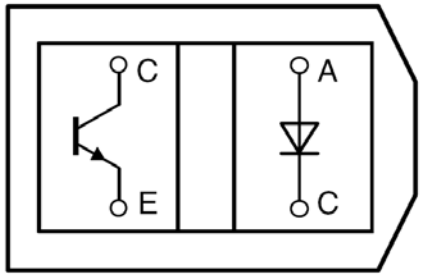
最大辨識距離約為d=20mm

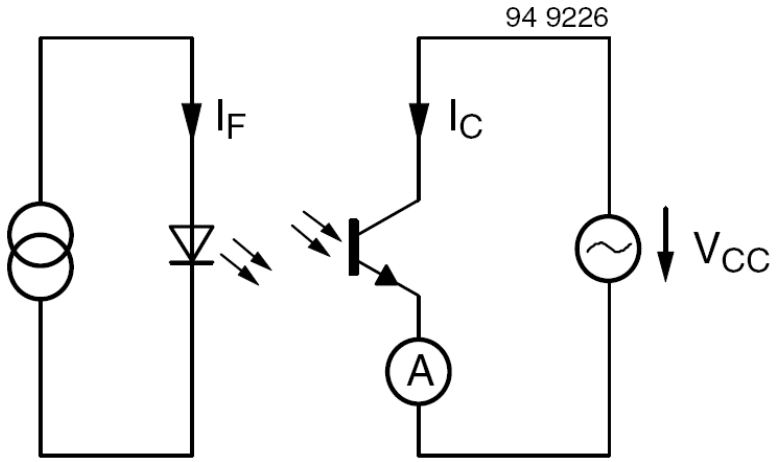
離地距離越近則辨識解析度越高

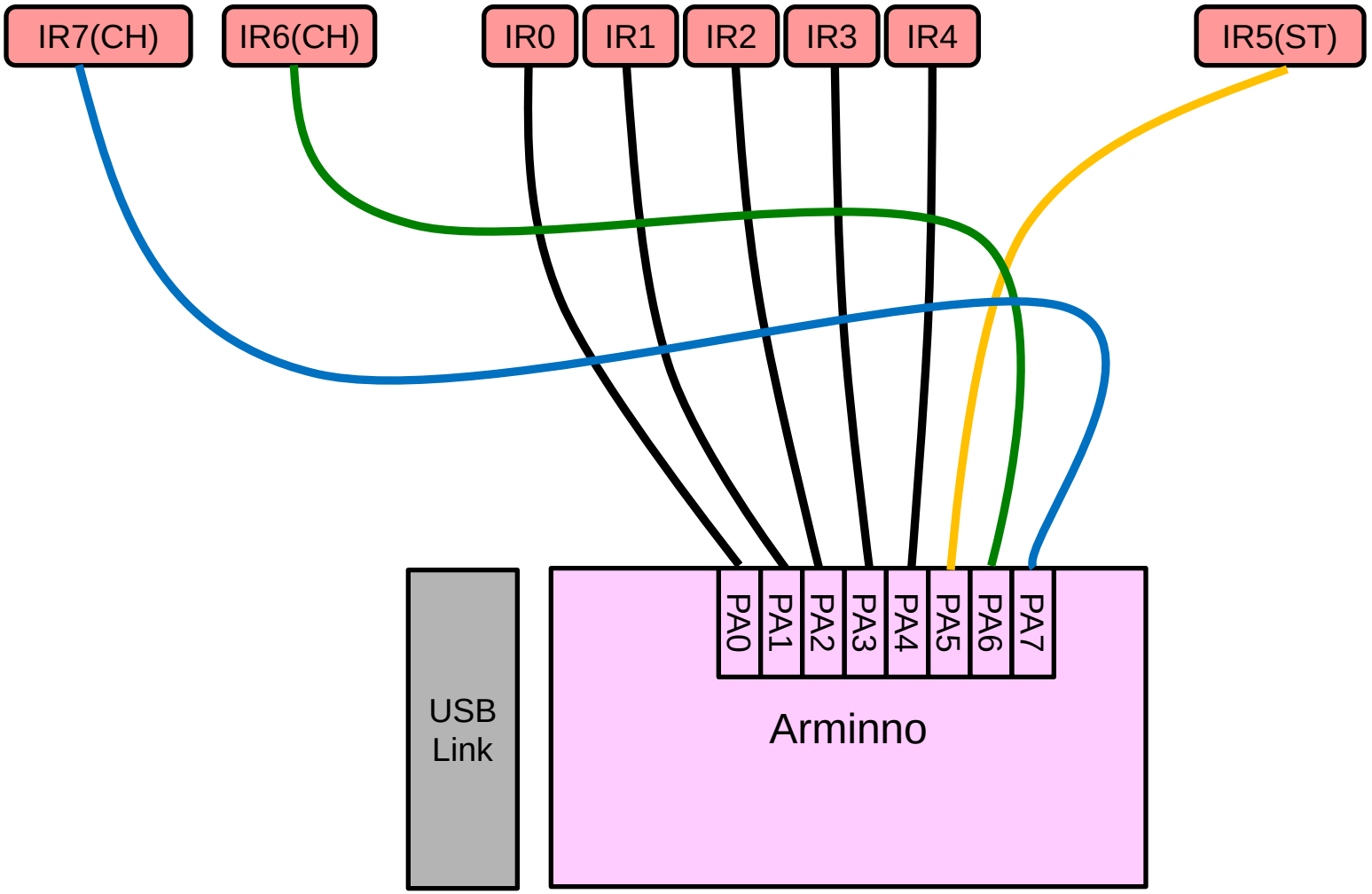
離地距離越遠則辨識解析度越低

特性曲線及電路

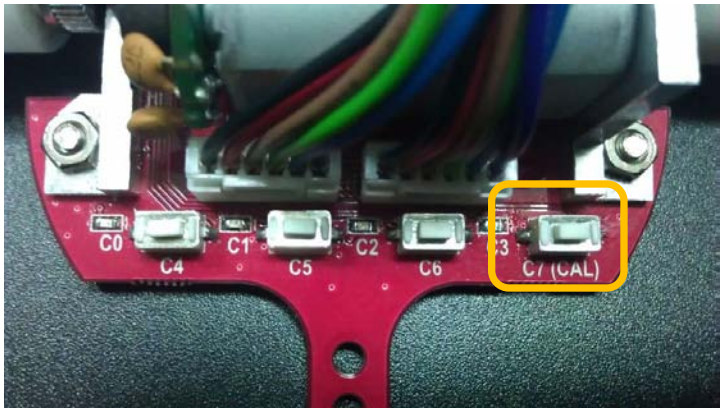

V_{CE} = 10 V
I_F = 20 mA
I_{Crel} - Relative Collector Current
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
12
14
16
d - Distance to Reflecting Card (mm)
96 11766

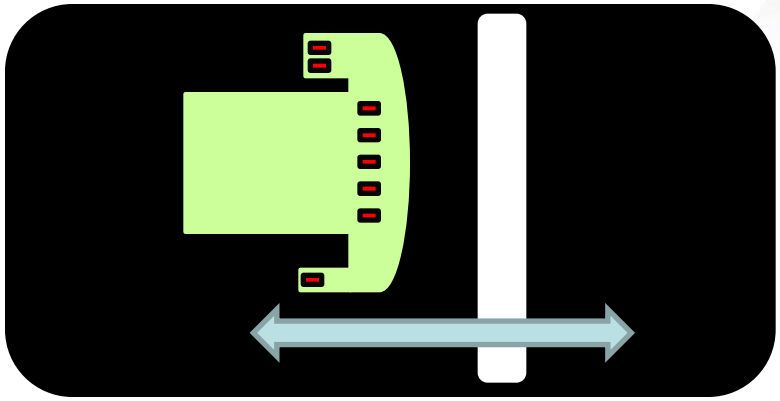

C
A
E
C


94 9226
I_F
I_C
V_{CC}
A


IR7(CH)
IR6(CH)
IR0
IR1
IR2
IR3
IR4
IR5(ST)
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
USB
Link
Arminno

內建感測器校正功能





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

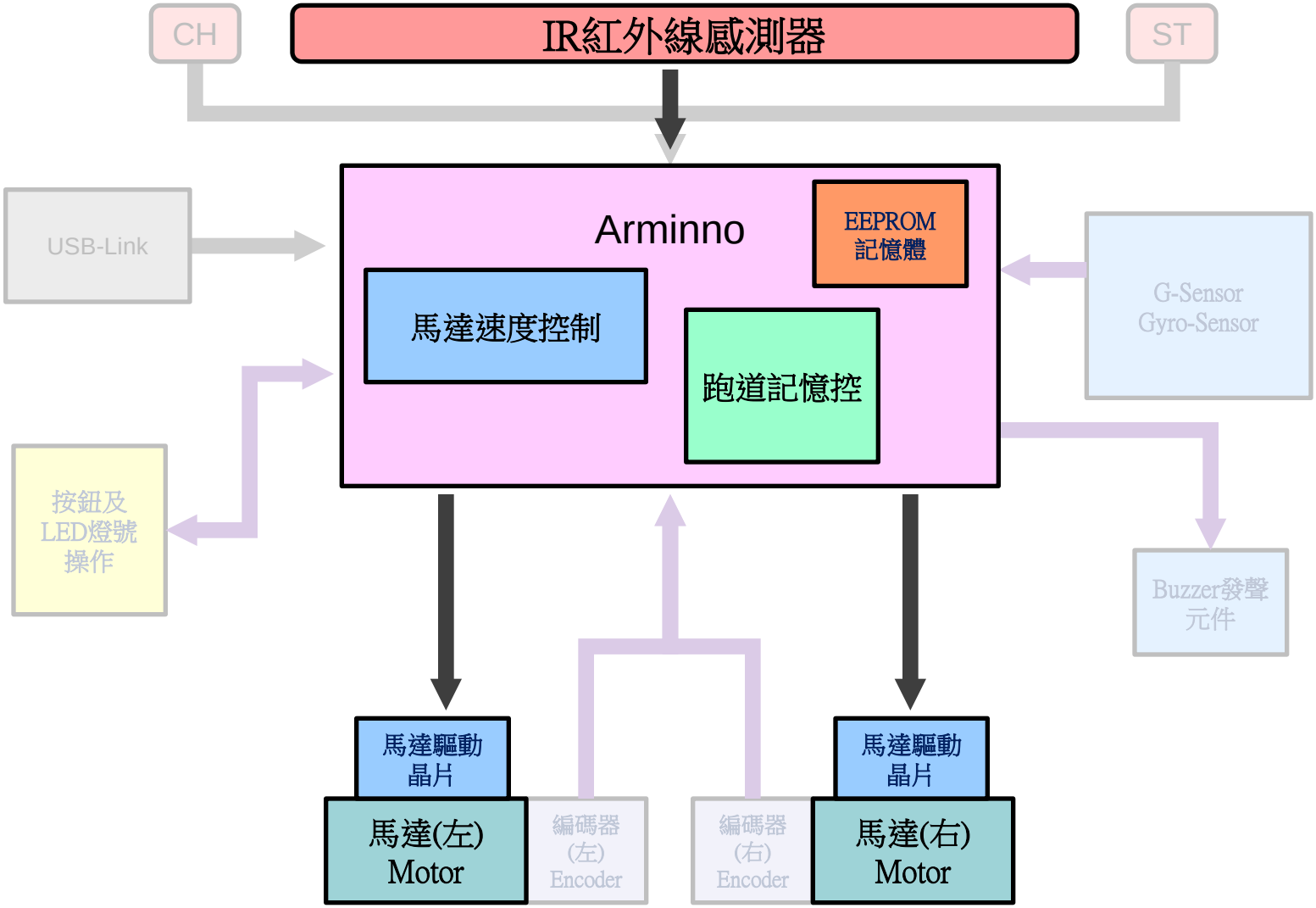
1. 按下CAL_BTN按鈕一直到紅色CAL_LED亮起 (約按住2秒)
2. 此時將InnoRacer平放在競速場地中 並對齊跑道白線，如圖所示！
3. 將自走車前後推動，以一定速度緩慢的將所有IR紅外線感測器來回劃過白線部份，並確認所有的紅外線感測器，都有經過場地與白線。
4. 完成以上動作之後再按下CAL_BTN按鈕以結束校正。

注意事項：過程中請保持自走車與地面的高度，以取得正確的校正值。

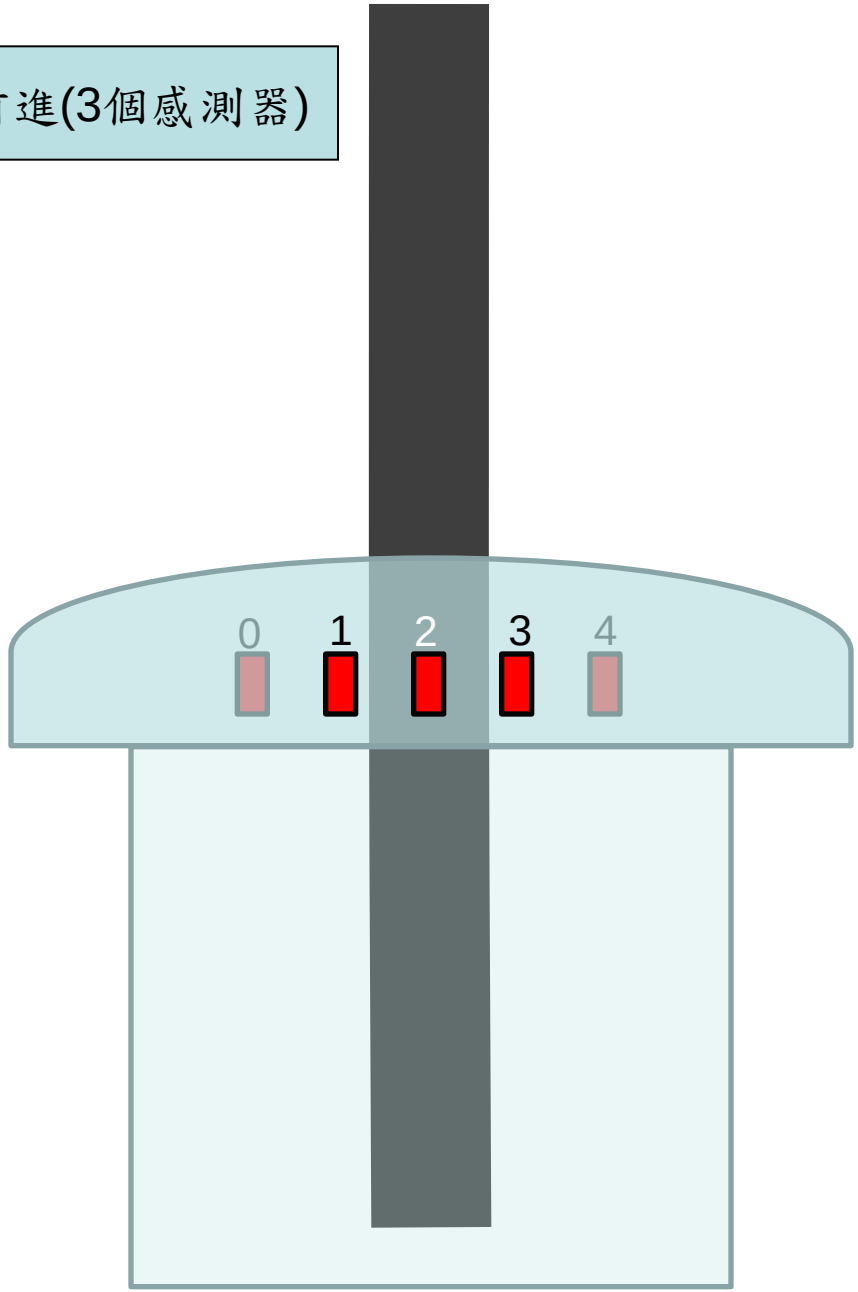
實驗四: 紅外線感測

```
//說明：偵測紅外線感測值
#include "arminno.h"
#include "innoRacer2.h"
innoRacer2 myRacer;

int main(void)
{
        unsigned char bIR,i;
        while(1)
        {
                myRacer.GetIR(bIR);          //取得IR感測值
                printf("\033[0;0f IR:");     //將游標移至0,0
                //使用二進制方式顯示
                for (i=0;i<8;i++)
                {
                        printf("%d",bIR&1);
                        bIR>>=1;
                }
        }
}
```

紅外線尋跡控制


CH
IR紅外線感測器
ST
USB-Link
Arminno
EEPROM
記憶體
馬達速度控制
跑道記憶控
G-Sensor
Gyro-Sensor
按鈕及
LED燈號
操作
Buzzer發聲
元件
馬達驅動
晶片
馬達(左)
Motor
編碼器
(左)
Encoder
編碼器
(右)
Encoder
馬達驅動
晶片
馬達(右)
Motor

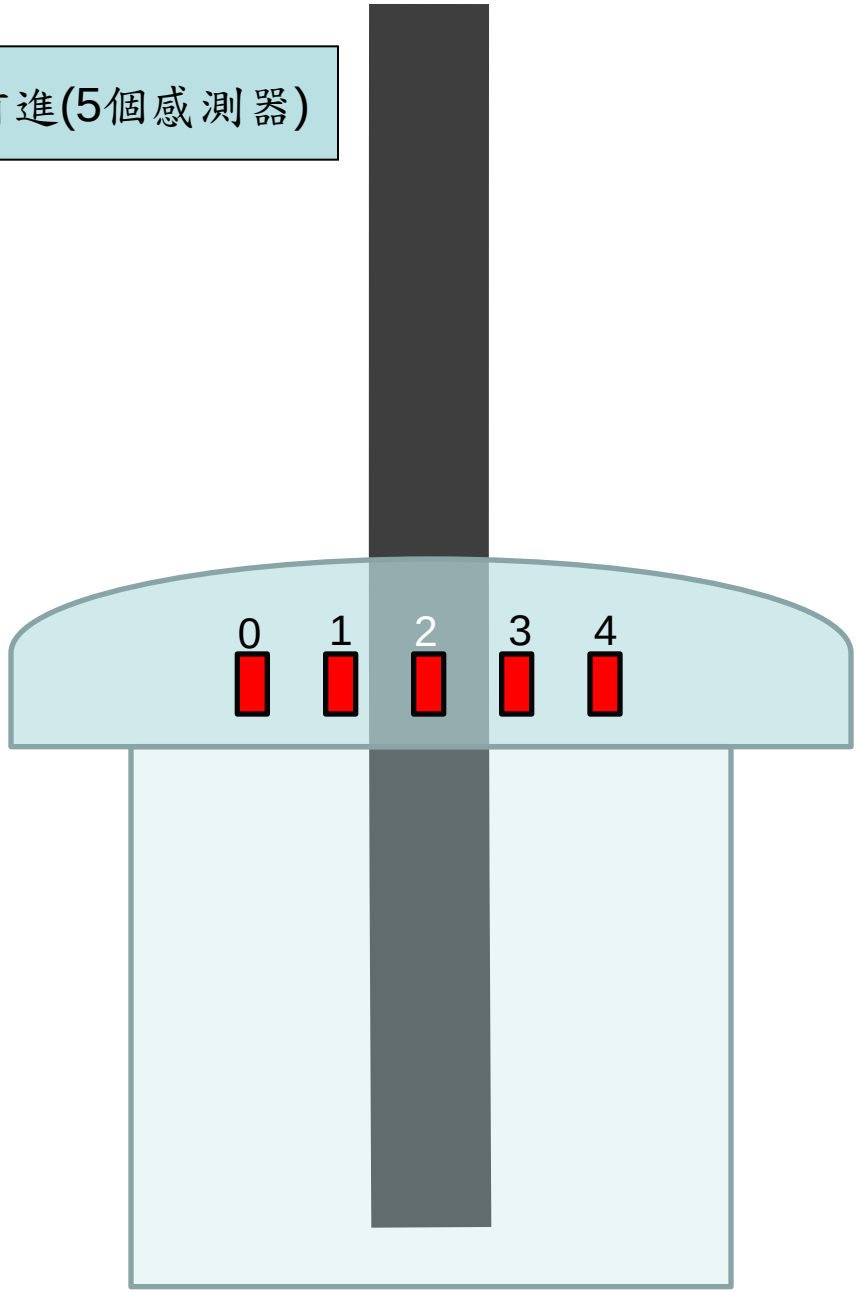
實驗五: 紅外線尋跡前進(3個感測器)



實驗五: 紅外線尋跡前進(3個感測器)

```
//說明：偵測紅外線感測值
#include "arminno.h"
#include "innoRacer2.h"
innoRacer2 myRacer;

const short sErrSet[ ]={-200,-130,-80,0,80,130,200};
const short Normal_Speed_R = 220;
const short Normal_Speed_L = 220;
int main(void)
{
   unsigned char bIR;
   short R,L,Err;
   while(1)
      {
         myRacer.GetIR(bIR); //取得IR感測值
         bIR>>=1;
         //依照線段位置選擇校正值
         switch (bIR&0x07)
             {
              case (2): //010
                  Err = sErrSet[3];
                  break;
             case (6): //011
              Err = sErrSet[4];
                break;
              case (4): //001
                 Err = sErrSet[5];
                 break;
              case (3): //110
                 Err = sErrSet[2];
                 break;
              case (1): //100
                 Err = sErrSet[1];
                 break;
              case (0):     //000 離開線段，依照上一個
                 if(Err<0)  //Err做基準，加大修正值
                    Err = sErrSet[0];
                  else if (Err>0)
                    Err = sErrSet[6];
                  break;
             }
         R = Normal_Speed_R - Err;
         L = Normal_Speed_L + Err;

         myRacer.SetVelLR(L,R);
          //依照校正過後的數值，控制馬達

             }

}
```

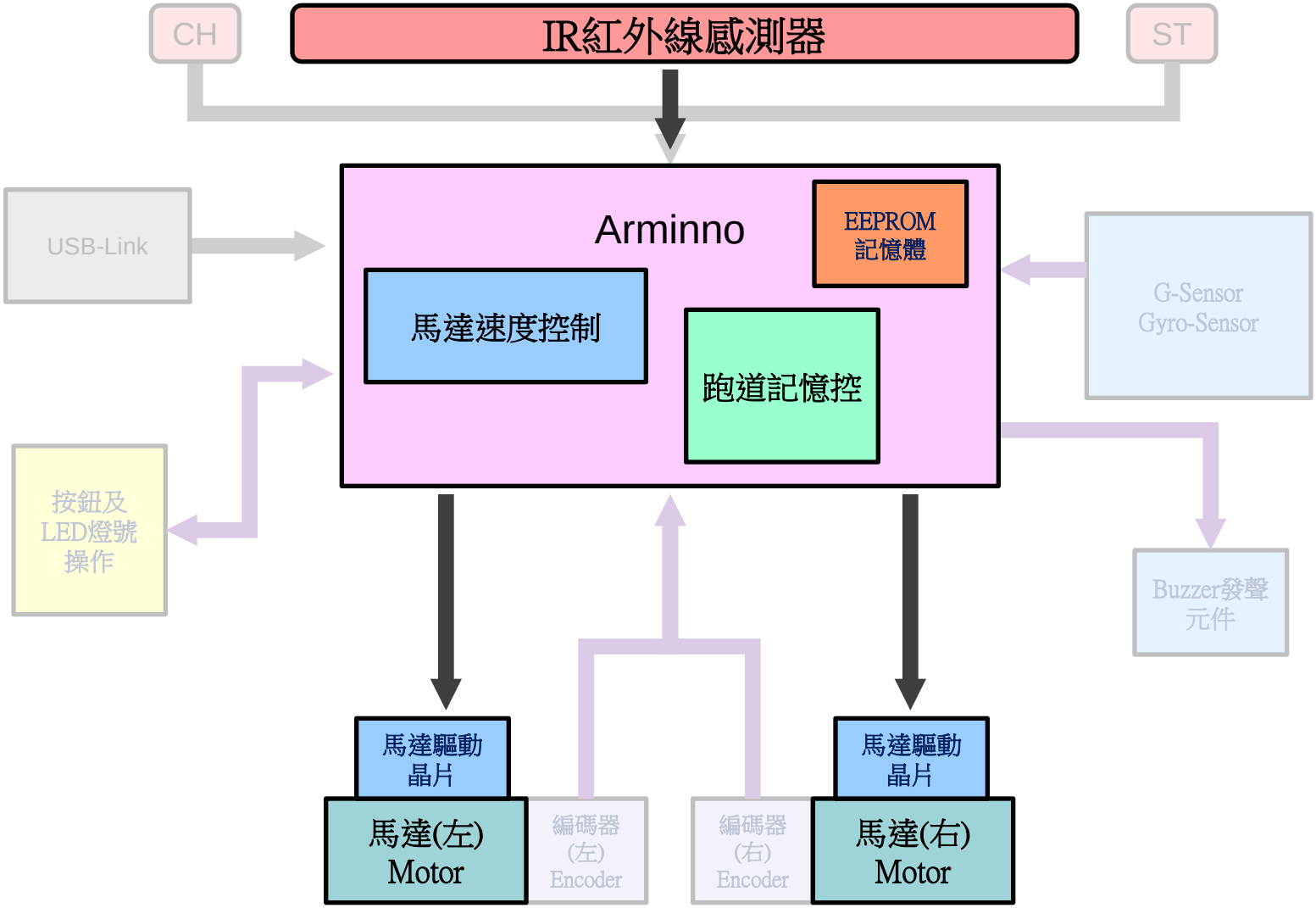
實驗五: 紅外線尋跡前進(5個感測器)



實驗六: 紅外線尋跡前進(5個感測器)

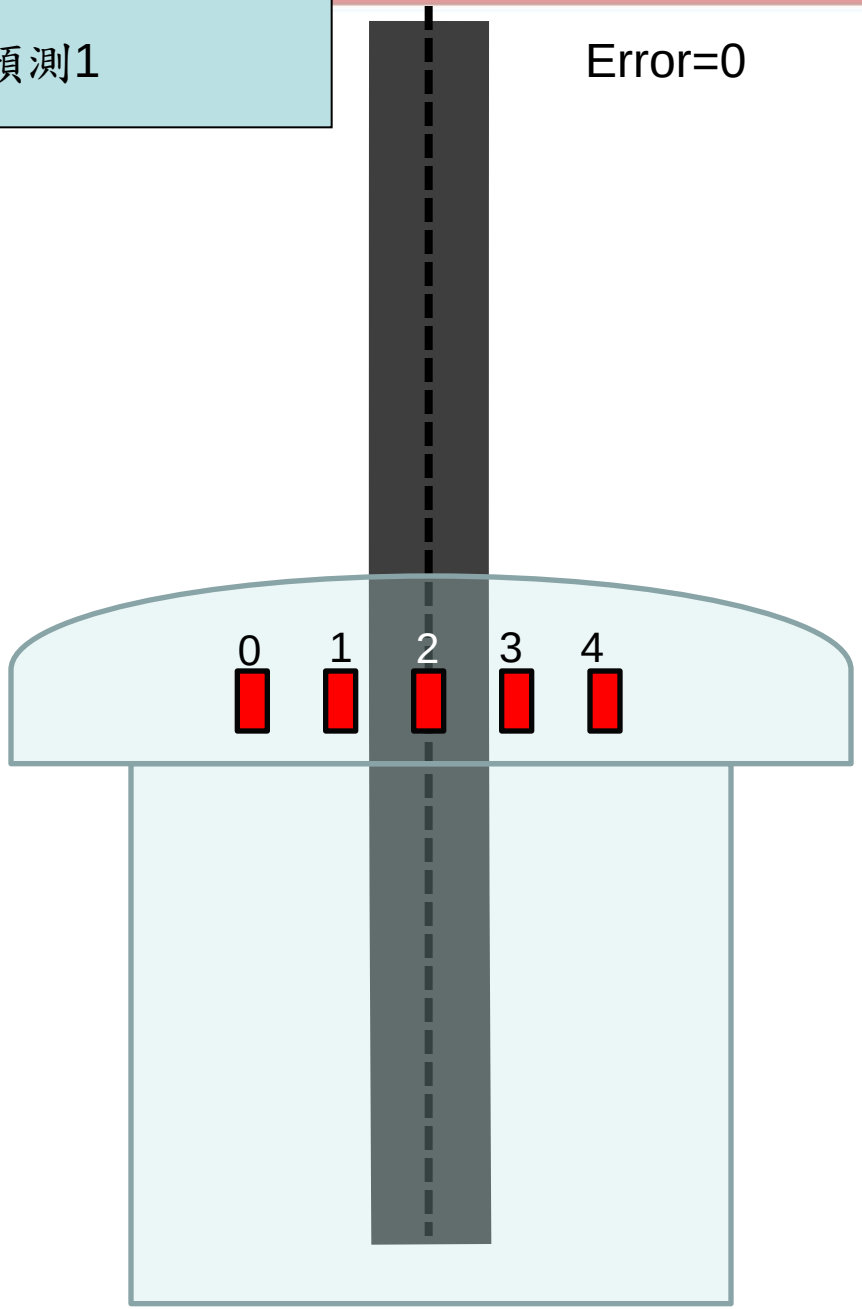
```
//說明：偵測紅外線感測值
#include "arminno.h"
#include "innoRacer2.h"
innoRacer2 myRacer;

const short sErrSet[]={-220,-190,-160,-130,-80,0,80,130,160,190,220};
const short Normal_Speed_R = 220;
const short Normal_Speed_L = 220;
int main(void)
{
    unsigned char bIR;
    short R,L,Err;
    while(1)
      {
        myRacer.GetIR(bIR);    //取得IR感測值

        //依照線段位置選擇校正值
        switch (bIR&0x1F)
          {
              case (4): //00100
                 Err = sErrSet[5];
                 break;
              case (12): //00110
                 Err = sErrSet[6];
                 break;
              case (8): //00010
                 Err = sErrSet[7];
                 break;
              case (24): //00011
                 Err = sErrSet[8];
                 break;
              case (16): //00001
                 Err = sErrSet[9];
                 break;
              case (6): //01100
                 Err = sErrSet[4];
                 break;
              case (2):  //01000
                 Err = sErrSet[3];
                 break;
              case (3): //11000
                 Err = sErrSet[2];
                 break;
              case (1): //10000
                 Err = sErrSet[1];
                 break;
              case (0): //000 離開線段，依照上一個Err做基準，加大修正值
                 if(Err<0)
                   Err = sErrSet[0];
                 else if (Err>0)
                   Err = sErrSet[10];
                 break;
          }
        R = Normal_Speed_R - Err;
        L = Normal_Speed_L + Err;
        myRacer.SetVelLR(L,R);    //依照校正過後的數值，控制馬達

    }

}
```

自走車控制理論介紹(PD控制)

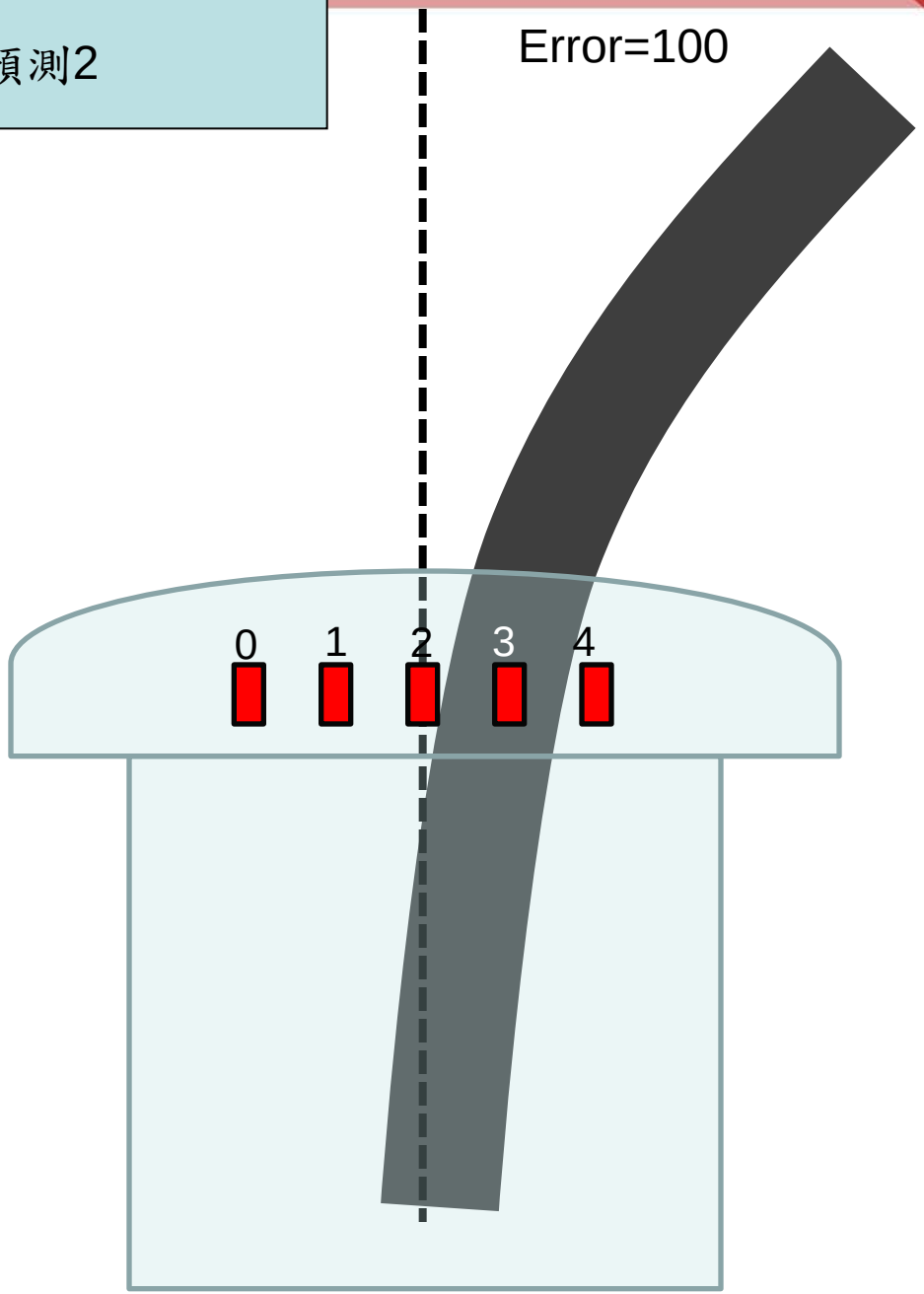

CH
IR紅外線感測器
ST
USB-Link
Arminno
EEPROM
記憶體
馬達速度控制
跑道記憶控
G-Sensor
Gyro-Sensor
按鈕及
LED燈號
操作
Buzzer發聲
元件
馬達驅動
晶片
馬達(左)
Motor
編碼器
(左)
Encoder
編碼器
(右)
Encoder
馬達驅動
晶片
馬達(右)
Motor

軌跡預測1

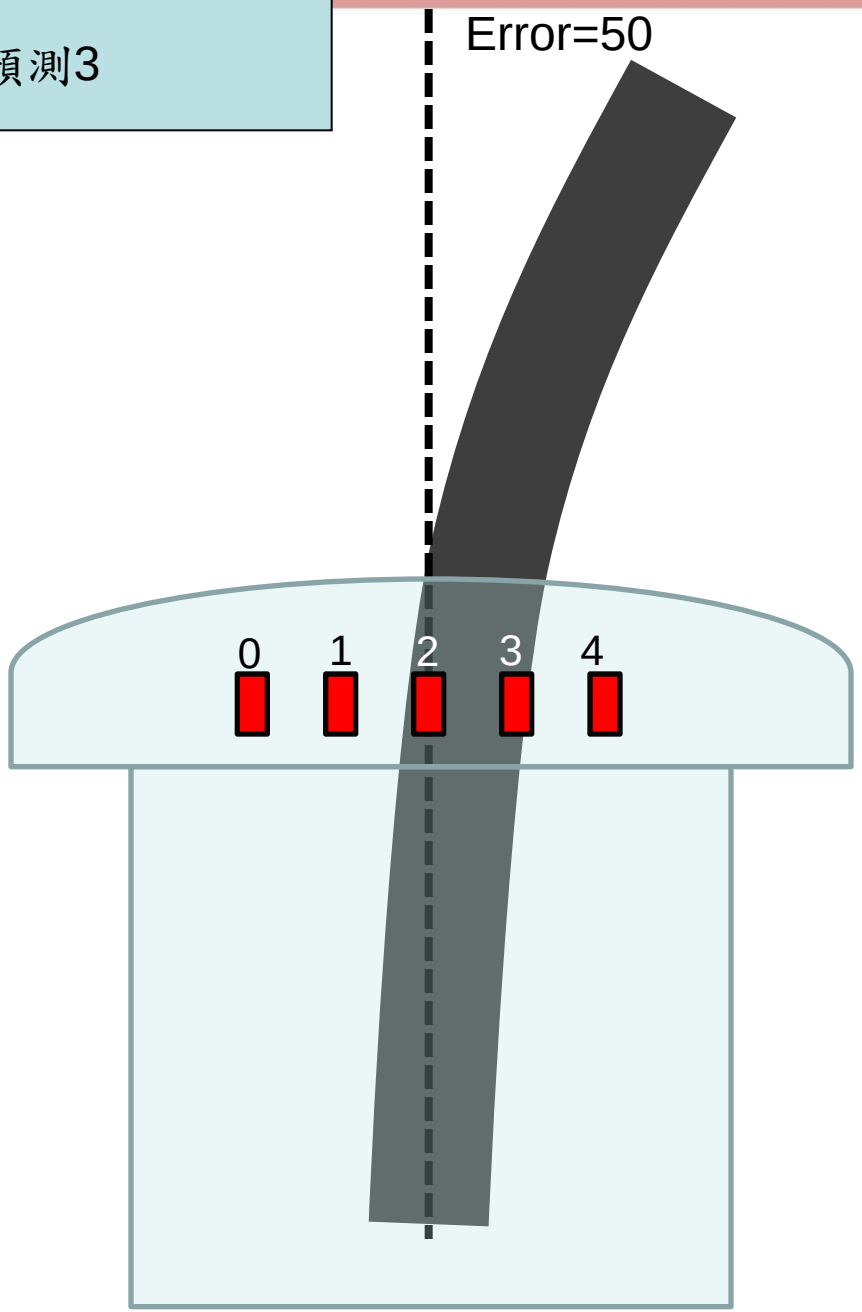
Error=0



軌跡預測2



軌跡預測3



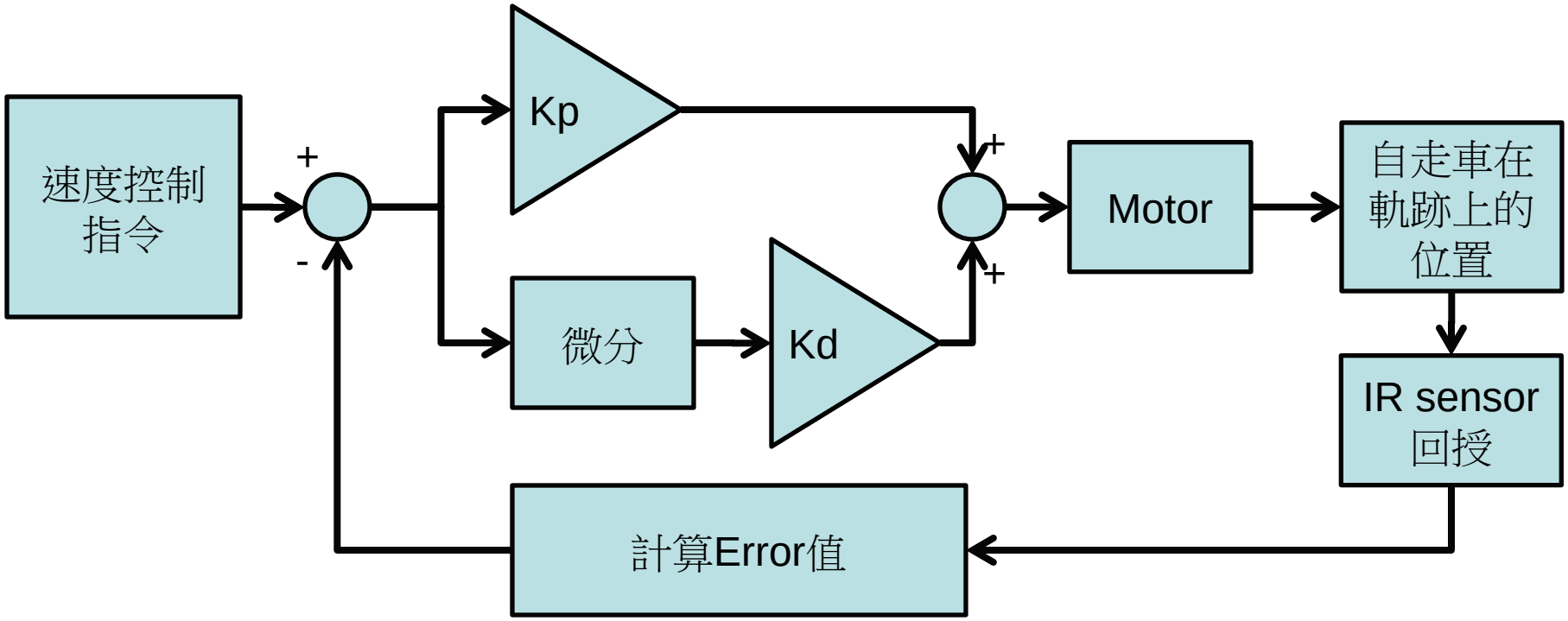
數位回授控制

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，誤差為負值，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

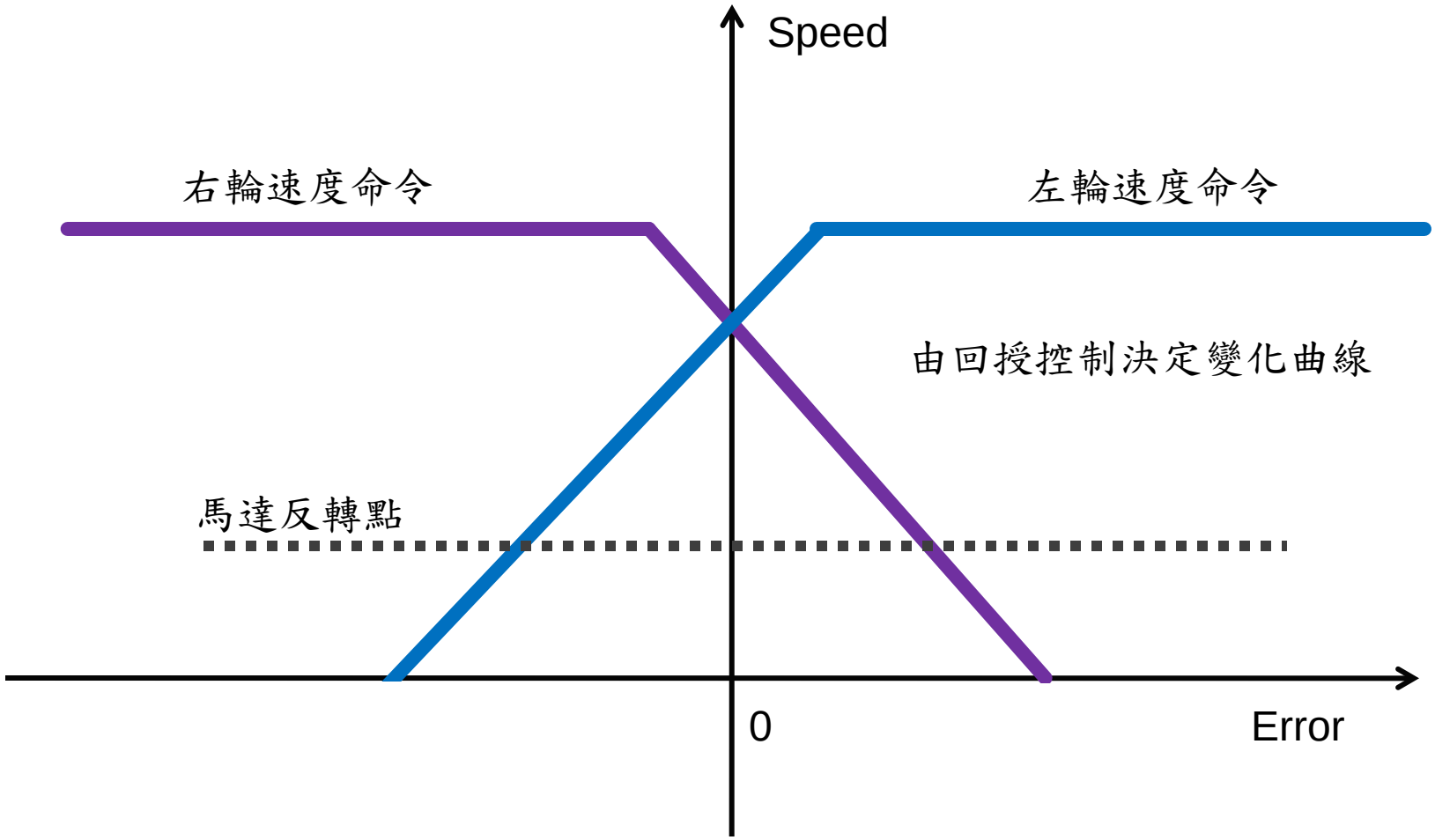
數位回授控制

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



數位回授控制

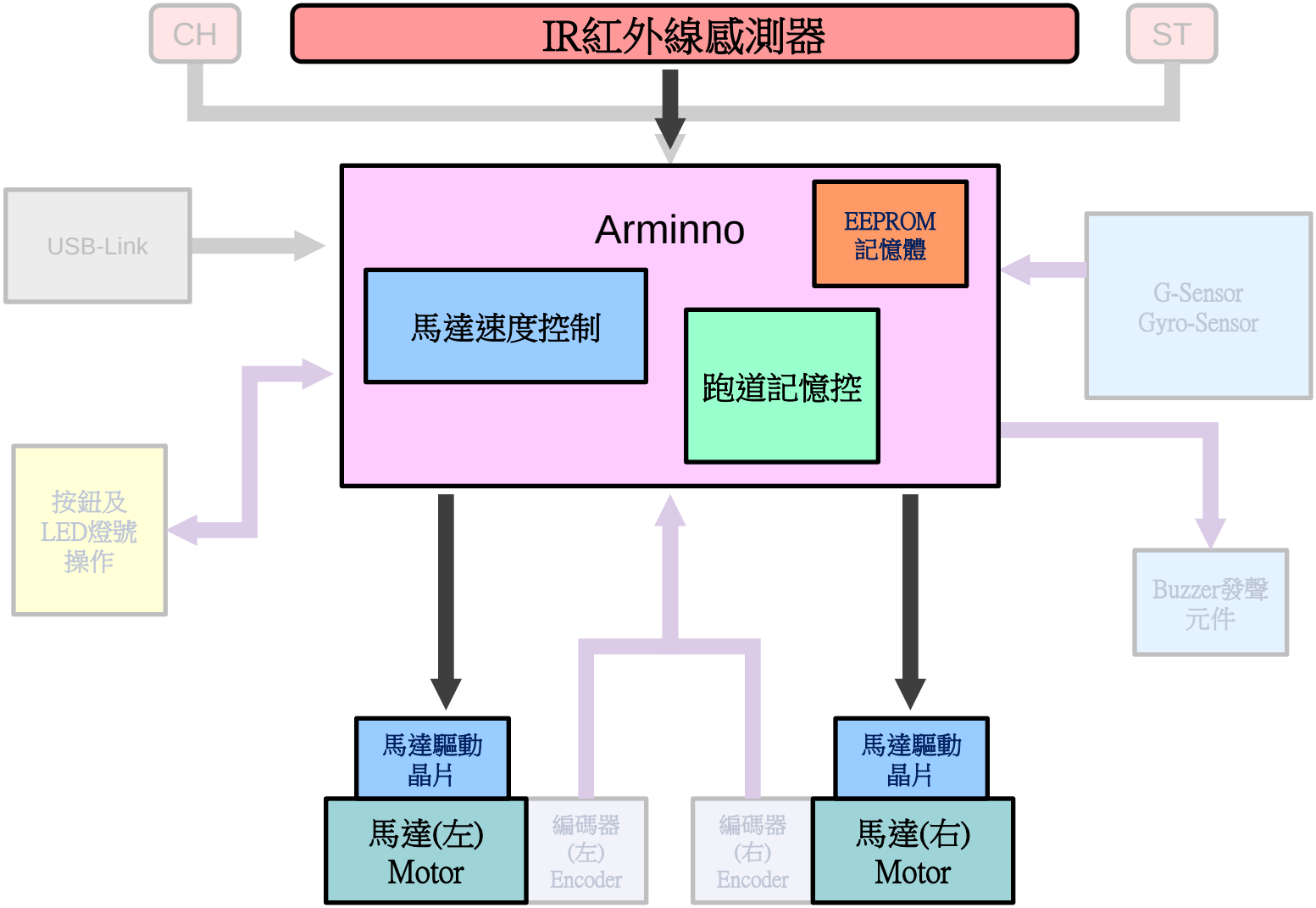
誤差值(Error)與左右輪速度變化值



實驗七: PD控制

```
//說明：偵測紅外線感測值
#include "arminno.h"
#include "innoRacer2.h"
innoRacer2 myRacer;

// 設定不同感測結果的誤差設定值
const short sErrSet[]={-220,-190,-160,-130,-80,0,80,130,160,190,220};
const short Normal_Speed_R = 220; //設定右輪直線速度值
const short Normal_Speed_L = 220; //設定左輪直線速度值
//設定PID的參數值
const char SCALE = 0;
const char kP = 1;
const char kI = 0;
const char kD = 10;

int main(void)
{
    unsigned char bIR;
    short R,L,Err,PreErr,Integral,Derivative,Out,Control;
    //無窮迴圈，反覆偵測黑線感測值，修正馬達轉向與速度
    while(1)
      {
        myRacer.GetIR(bIR);       //取得IR感測值
        //依照線段位置選擇校正值
        switch (bIR&0x1F)
          {
            case (4): //00100
              Err = sErrSet[5];
              break;
            case (12): //00110
              Err = sErrSet[6];
              break;
            case (8): //00010
              Err = sErrSet[7];
              break;
            case (24): //00011
              Err = sErrSet[8];
              break;
            case (16): //00001
              Err = sErrSet[9];
              break;
            case (6): //01100
              Err = sErrSet[4];
              break;
            case (2):  //01000
              Err = sErrSet[3];
              break;
            case (3): //11000
              Err = sErrSet[2];
              break;
            case (1): //10000
              Err = sErrSet[1];
              break;
            case (0): //000 離開線段，依照上一個Err做基準，加大修正值
              if(Err<0)
                Err = sErrSet[0];
              else if (Err>0)
                Err = sErrSet[10];
              break;
          }

        //PID計算公式
        Integral = Integral + Err;
        Derivative = Err - PreErr;
        Out = (kP*Err)+(kI*Integral)+(kD*Derivative);
        PreErr = Err;
        Control = Out>>SCALE;

        //根據誤差值計算左右輪的轉速值
        R = Normal_Speed_R - Control;
        L = Normal_Speed_L + Control;

        // 判斷設定值是否超過極限值
        if (R>1024) {
            R = 1024;
          }
        else if (R<-1024){
            R = -1024;
          }
        if (L>1024){
            L = 1024;
          }
        else if (L<-1024){
            L = -1024;
          }
        myRacer.SetVelLR(L,R);       //依照校正過後的數值，控制馬達
    }

}
```

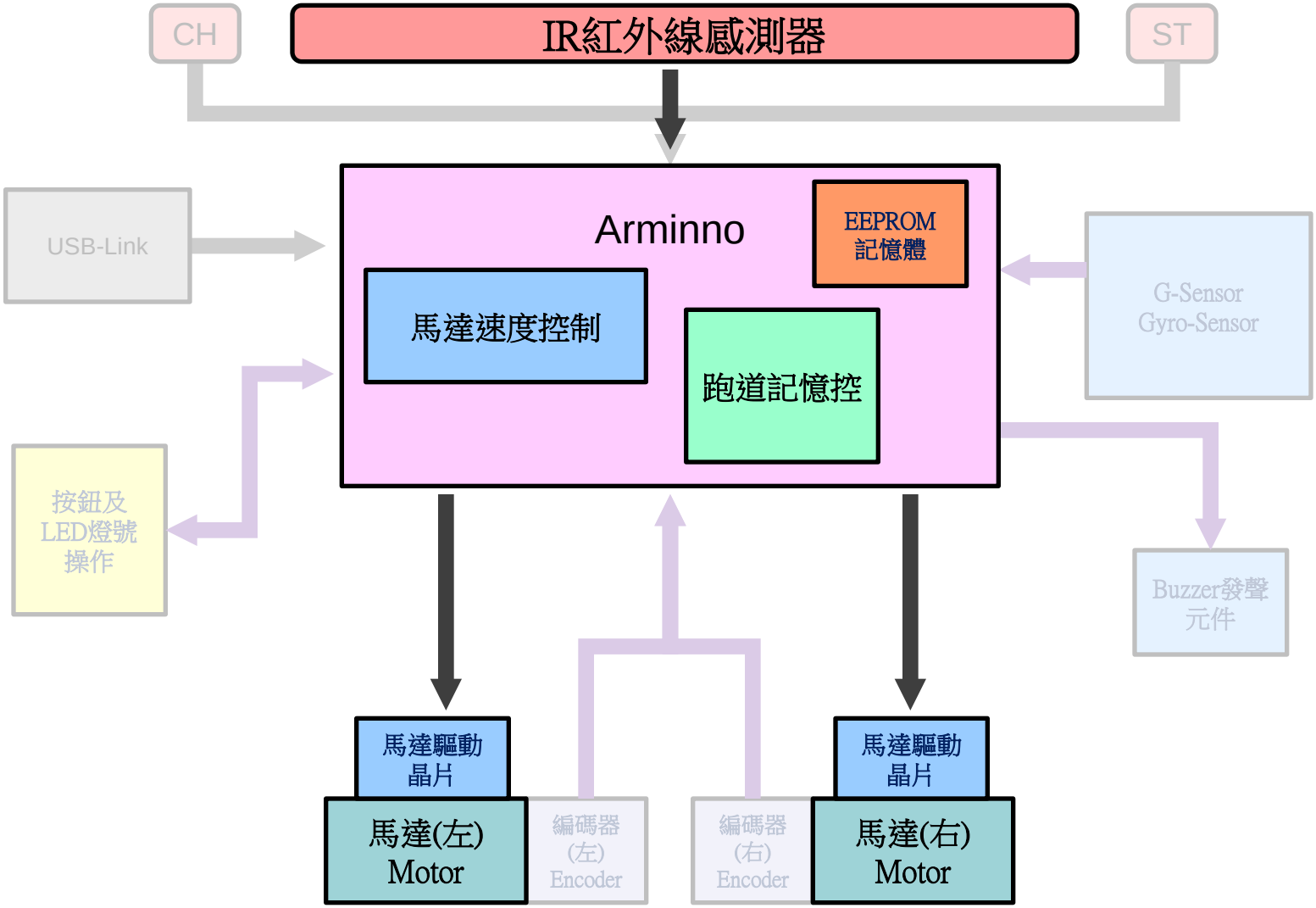
內建PD控制


CH
IR紅外線感測器
ST
USB-Link
Arminno
EEPROM
記憶體
馬達速度控制
跑道記憶控
G-Sensor
Gyro-Sensor
按鈕及
LED燈號
操作
Buzzer發聲
元件
馬達驅動
晶片
馬達(左)
Motor
編碼器
(左)
Encoder
編碼器
(右)
Encoder
馬達驅動
晶片
馬達(右)
Motor

實驗八: 內建PD控制

```
#include "arminno.h"
#include "innoRacer2.h"
innoRacer2 myRacer;

#define FREQ_CTRL                4

//模組內部最高最低速限，以及直線速度 (-1024 ~ 1024)

const short Normal_Speed_R = 130; //設定右輪直線速度值
const short Normal_Speed_L = 130; //設定左輪直線速度值

const short kP=22;
const short kI=0;
const short kD=240;

//模組各別感測階段的誤差值
#define PID_SCALAR           4
const short sStrErr[ ]={18,30,48,78,126,144,330,534};

//設定模組各項預設值
void Init(void)
{
   //設定PID參數
   myRacer.SetCtrlFreq(FREQ_CTRL);
   myRacer.SetP(kP);
   myRacer.SetI(kI);
   myRacer.SetD(kD);
   myRacer.SetScalar(PID_SCALAR);
   //設定各別感測階段的誤差值
   myRacer.SetErrScale(sStrErr[0], sStrErr[1], sStrErr[2], sStrErr[3],
   StrErr[4], sStrErr[5], sStrErr[6], sStrErr[7]);
   //設定中心速度
   myRacer.SetStraight(Normal_Speed_L, Normal_Speed_R );
   myRacer.SetIRMode(0); //切換數位模式
   //設定路線顏色
   myRacer.SetLineColor(0);
   myRacer.SetMotorDeadZone(136);
}

int main(void)
{
   unsigned char bIR ;
   Init();
   do {
            myRacer.GetIR(bIR);
   } while((bIR & 0x04) == 0x00); //判斷暫道是否在中間位置
   myRacer.BuzzerOn();

   Pause(20000);
   myRacer.SpdCtrlOn(0); //啟動PID速度控制
   while(1);    //無窮迴圈
}
```

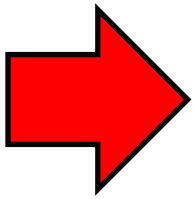
進階IR紅外線感測器類比控制


CH
IR紅外線感測器
ST
USB-Link
Arminno
EEPROM
記憶體
馬達速度控制
跑道記憶控
G-Sensor
Gyro-Sensor
按鈕及
LED燈號
操作
Buzzer發聲
元件
馬達驅動
晶片
馬達(左)
Motor
編碼器
(左)
Encoder
編碼器
(右)
Encoder
馬達驅動
晶片
馬達(右)
Motor

類比感測器數值正規化

調整**前**值域

- IR0 =0~350
- IR1 =20~480
- IR2 =50~330
- IR3 =7~457
- IR4 =0~545
- IR5 =100~350
- IR6 =14~407



調整**後**值域

- IR0 =0~100
- IR1 =0~100
- IR2 =0~100
- IR3 =0~100
- IR4 =0~100
- IR5 =0~100
- IR6 =0~100

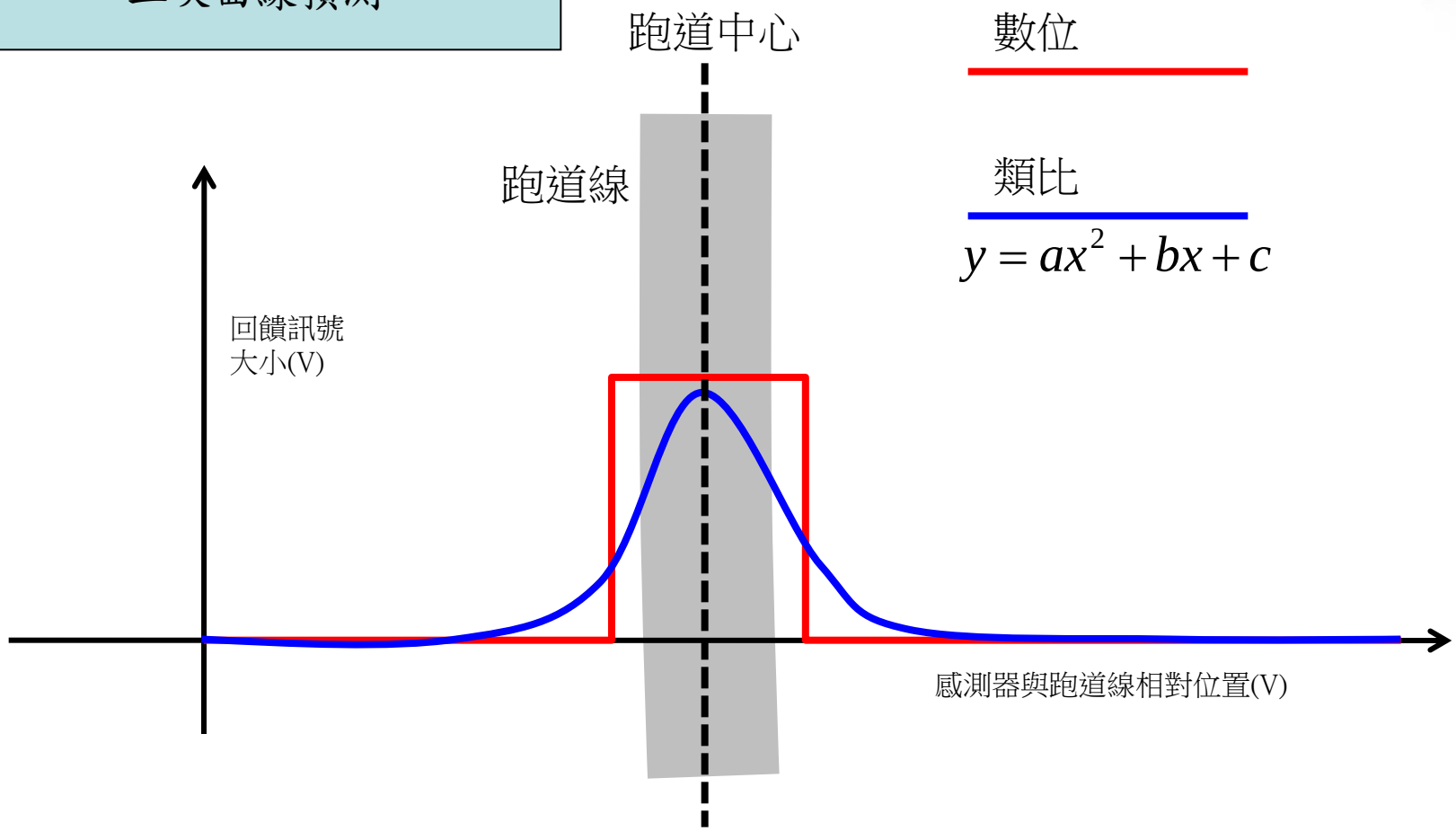
類比感測器數值正規化

$$\frac{Y_{輸出值} - Y_{最小值}}{X_{輸入值} - X_{最小值}} = \frac{Y_{最大值} - Y_{最小值}}{X_{最大值} - X_{最小值}}$$

$$Y_{輸出值} = Y_{最小值} + \left(\frac{Y_{最大值} - Y_{最小值}}{X_{最大值} - X_{最小值}} \right) * \left(X_{輸入值} - X_{最小值} \right)$$

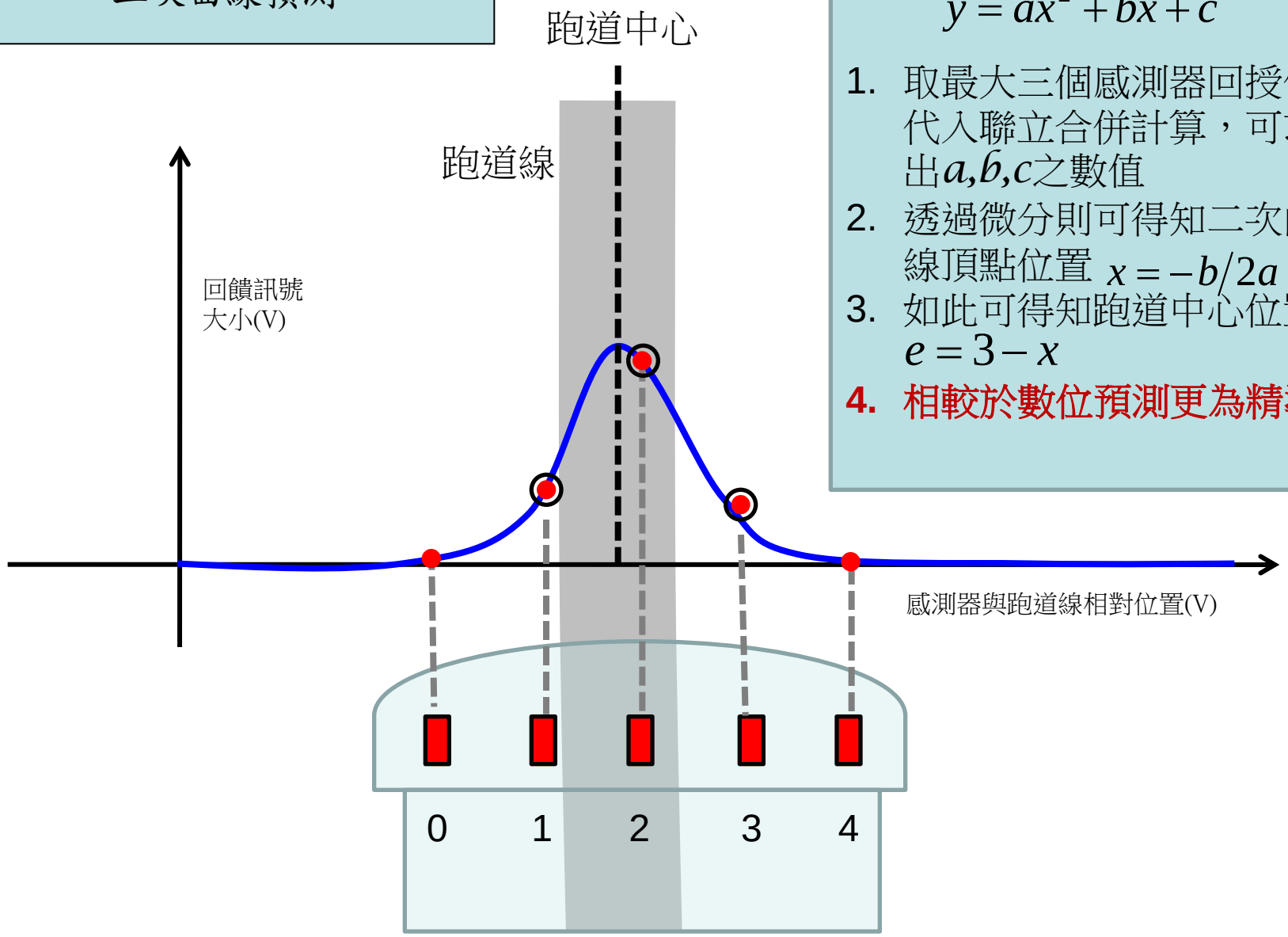
每顆感測器的類比輸出特性會因實際狀況而有所差異
因此需透過演算法將感測器輸出調整至相同的值域範圍

二次曲線預測



1. 透過IR感測器類比式回授可更精準得知感測器相對於跑道現的偏移量
2. 類比回授訊號曲線可近似於一二次曲線 $y = ax^2 + bx + c$

二次曲線預測



$$y = ax^2 + bx + c$$

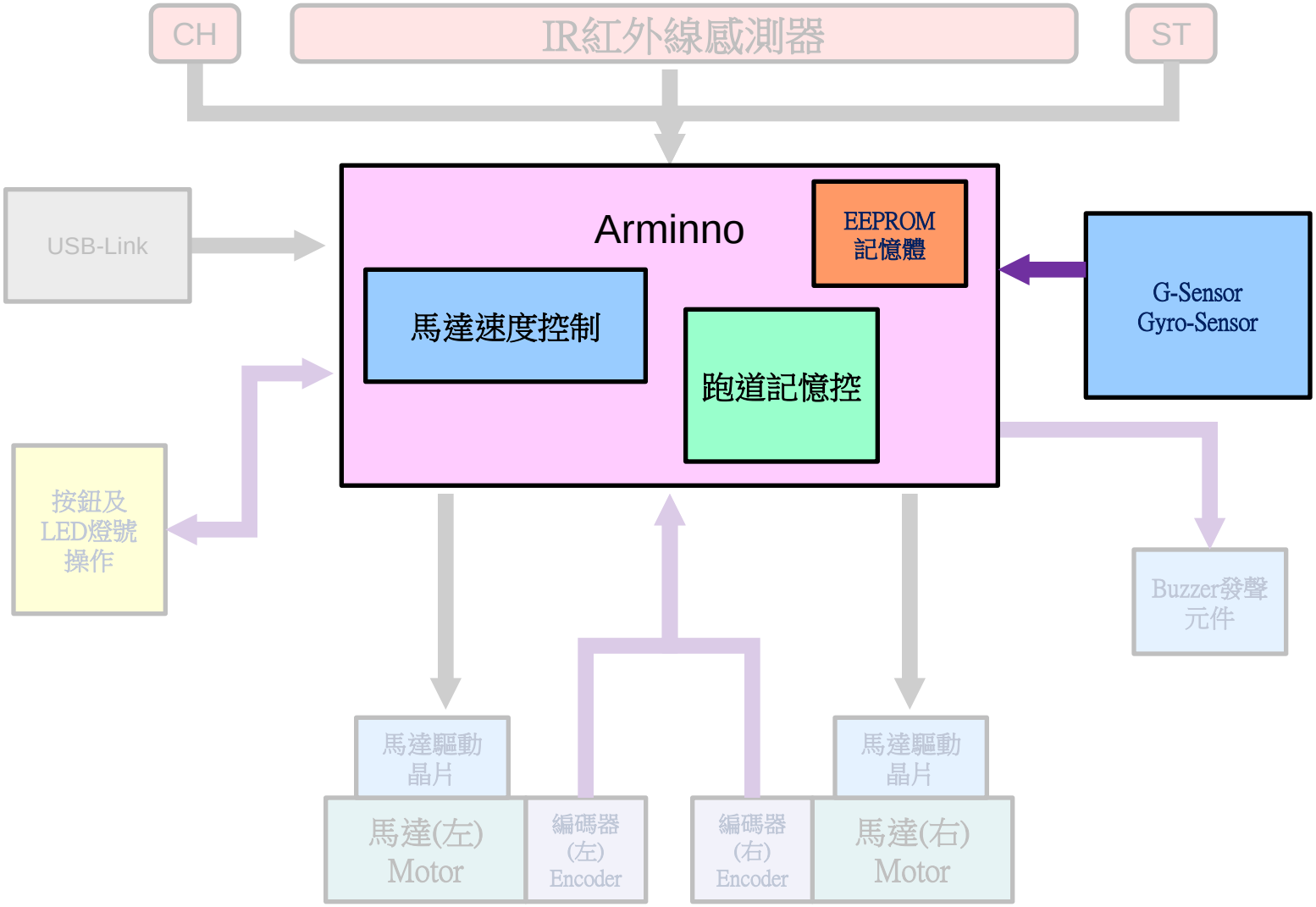
1. 取最大三個感測器回授值代入聯立合併計算，可求出a,b,c之數值
2. 透過微分則可得知二次曲線頂點位置 $x = -b/2a$
3. 如此可得知跑道中心位置 $e = 3 - x$
4. **相較於數位預測更為精準**

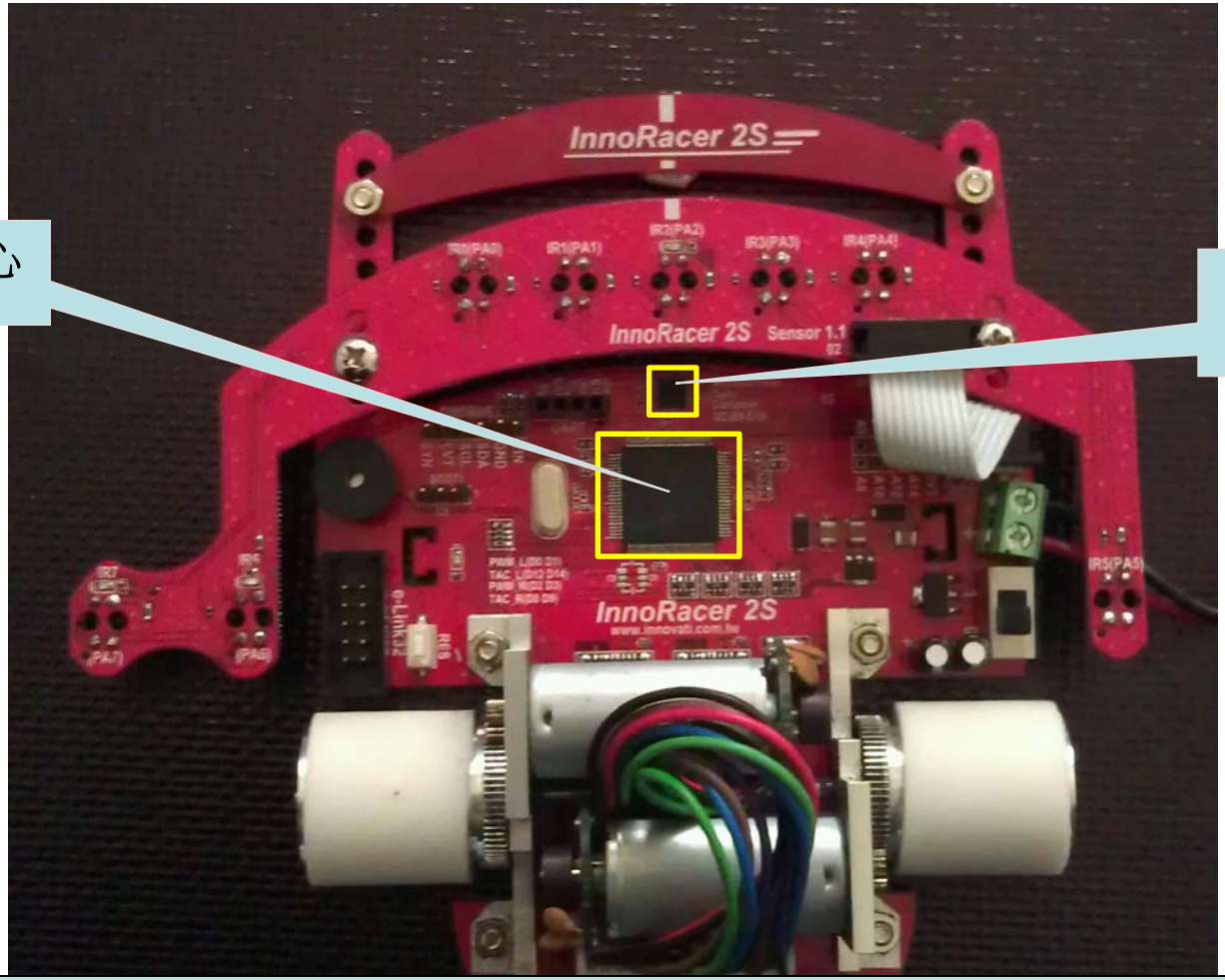
實驗九: 內建PD控制(類比)

```
#include "arminno.h"
#include "innoRacer2.h"
innoRacer2 myRacer;

#define FREQ_CTRL                4

//模組內部最高最低速限，以及直線速度 (-1024 ~ 1024)

const short Normal_Speed_R = 130; //設定右輪直線速度值
const short Normal_Speed_L = 130; //設定左輪直線速度值

const short kP=22;
const short kI=0;
const short kD=240;

//模組各別感測階段的誤差值
#define PID_SCALAR            4
const short sStrErr[ ]={18,30,48,78,126,144,330,534};

//設定模組各項預設值
void Init(void)
{
   //設定PID參數
   myRacer.SetCtrlFreq(FREQ_CTRL);
   myRacer.SetP(kP);
   myRacer.SetI(kI);
   myRacer.SetD(kD);
   myRacer.SetScalar(PID_SCALAR);
   //設定各別感測階段的誤差值
   myRacer.SetErrScale(sStrErr[0], sStrErr[1], sStrErr[2], sStrErr[3],
   StrErr[4], sStrErr[5], sStrErr[6], sStrErr[7]);
```

```
 //設定中心速度
     myRacer.SetStraight(Normal_Speed_L, Normal_Speed_R );
     myRacer.SetIRMode(1); //切換類比模式
     //設定路線顏色
     myRacer.SetLineColor(0);
     myRacer.SetMotorDeadZone(136);
}

int main(void)
{
     unsigned char bIR ;
     Init();
     do {
                myRacer.GetIR(bIR);
     } while((bIR & 0x04) == 0x00); //判斷暫道是否在中間位置
     myRacer.BuzzerOn();

     Pause(20000);
     myRacer.SpdCtrlOn(0); //啟動PID速度控制
     while(1);    //無窮迴圈
}
```

加速度感測器


CH
IR紅外線感測器
ST
USB-Link
Arminno
EEPROM
記憶體
馬達速度控制
跑道記憶控
G-Sensor
Gyro-Sensor
按鈕及
LED燈號
操作
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)
Encoder
編碼器
(右)
Encoder
馬達(右)
Motor

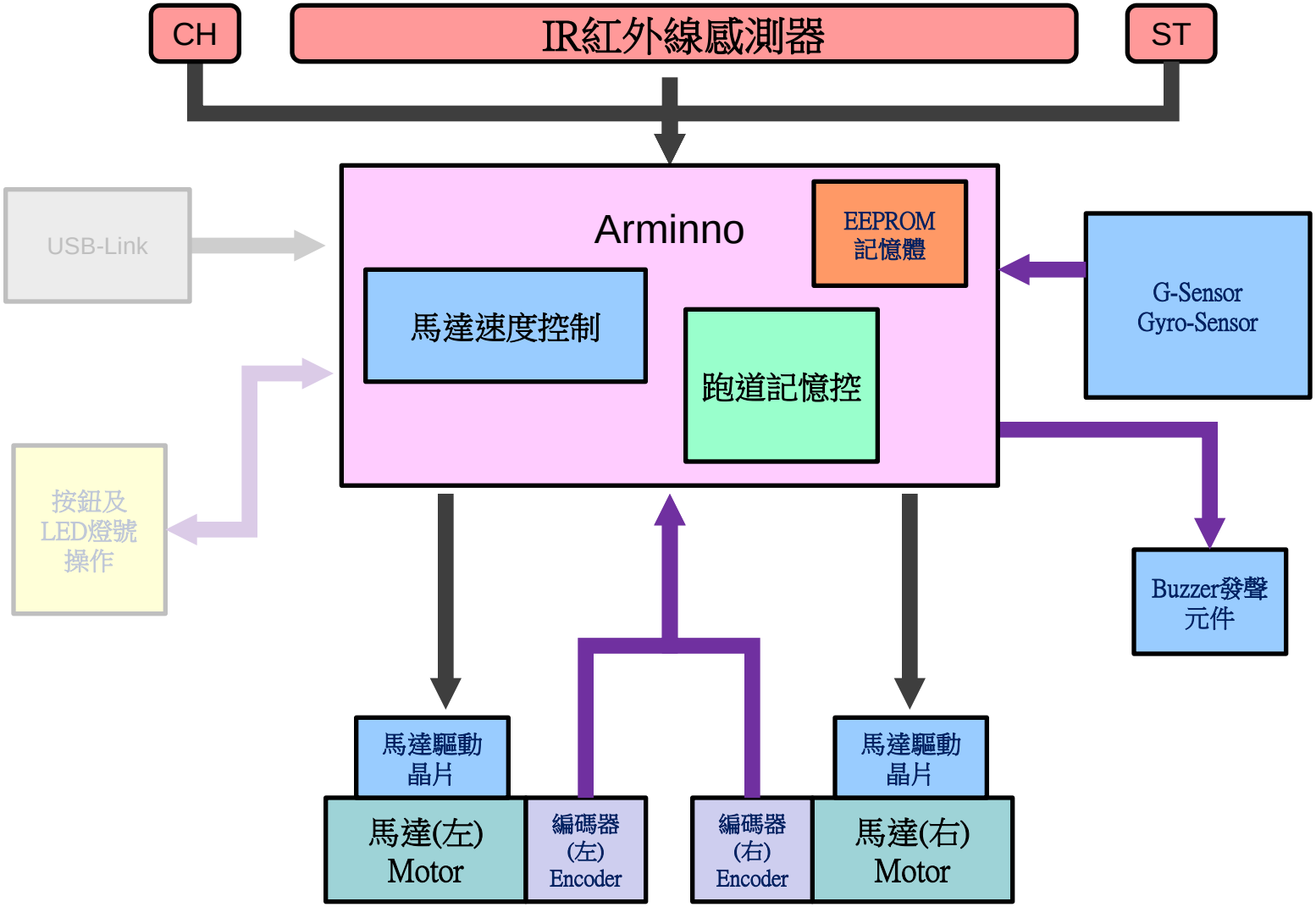


1. 輪胎抓地力決定過彎向心加速度極限值
2. 每台自走車都會有在過彎時的向心加速度極限值
3. 利用過彎時產生的向心加速度值來決定要用多少速度過彎

實驗十: 加速度與陀螺儀感測器

```
#include "arminno.h"
#include "innoRacer2.h"
innoRacer2 myRacer;
short  AccY, GyroZ;
int main(void)
{
        while(1)
        {
                myRacer.GetAyGz(AccY, GyroZ); //取得Gyro G 感測值
                printf("\033[0;0f Accelerometer Y:");          //將游標移至0,0
                printf("%d", AccY);
                printf("   Gyro Z:");
                printf("%d", GyroZ);
                printf("\033[K");                  //清除游標後資訊
        }

}
```

跑道資料紀錄控制


CH
IR紅外線感測器
ST
USB-Link
Arminno
EEPROM
記憶體
馬達速度控制
跑道記憶控
G-Sensor
Gyro-Sensor
按鈕及
LED燈號
操作
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)
Encoder
編碼器
(右)
Encoder
馬達(右)
Motor

實驗十一: 繞圈紀錄資料

```
#include "arminno.h"
#include "innoRacer2.h"
innoRacer2 myRacer;

#define FREQ_CTRL              4
#define CROSS_COUNT        14 //設定交叉軌道判斷長度

const short Normal_Speed_R = 130; //設定右輪直線速度值
const short Normal_Speed_L = 130; //設定左輪直線速度值

//設定PID參數
const short kP=22;
const short kI=0;
const short kD=240;

// 模組各別感測階段的誤差值
define PID_SCALAR                  4
const short sStrErr[ ]={18,30,48,78,126,144,330,534};
unsigned char bStatus;
unsigned char bIR;
// 設定模組各項預設值
void Init(void)
{
    //設定PID參數
    myRacer.SetCtrlFreq(FREQ_CTRL);
    myRacer.SetP(kP);
    myRacer.SetI(kI);
    myRacer.SetD(kD);
    myRacer.SetScalar(PID_SCALAR);
    //設定速度相關資料
    myRacer.SetStraight(Normal_Speed_L , Normal_Speed_R );
    myRacer.SetIRMode(1);
    //設定誤差值
    myRacer.SetErrScale(sStrErr[0], sStrErr[1], sStrErr[2],sStrErr[3], sStrErr[
    myRacer.SetCrossCount(CROSS_COUNT);   //設定交叉軌道判斷距雄
    myRacer.SetMotorDeadZone(136);
}
```

```
int main(void)
{

    Init();
    myRacer.AutoBeep(1);        //設定經過曲率變化點發出提醒

    //判斷軌道是否在中間
    do {
                myRacer.GetIR(bIR);
     } while((bIR & 0x04) == 0x00);

     myRacer.StartRec(1);          // 啟動記錄
     //判斷是否進入記錄狀態
     do {
                myRacer.GetRecStatus(bStatus);
        } while(bStatus != 1);
     //啟動PID速度控制
     myRacer.SpdCtrlOn(0);
     //判斷是否偵測到開始點
     do {
                  myRacer.GetRecStatus(bStatus);
        } while (bStatus != 2);
      //判斷是否偵測到結束點
       do {
                myRacer.GetRecStatus(bStatus);
          } while(bStatus != 0);

                //結束記錄
                myRacer.StopRec();
                myRacer.StopDual();
}
```

實驗十二: 讀取紀錄值

```
#include "arminno.h"
#include "innoRacer2.h"
innoRacer2 myRacer;

int main(void)
{

    unsigned char SecCnt;
    short L, R, MaxAcc, AvgAcc, MaxGyro, AvgGyro;

    myRacer.GetTotalSecCnt(SecCnt);          //取得上次記錄的總路段數

    printf("Count = %d\r\n", SecCnt);
    printf("   \t L  \t R   \tMaxAcc\tAvgAcc\tMaxGyro\tAvgGyro\r\n");

    //依序取得各路段的資料
    for(unsigned char i = 0 ; i < SecCnt ; i++) {
            myRacer.GetSecLen(i, L, R);
            myRacer.GetSecMaxAyGz(i, MaxAcc, MaxGyro);
            myRacer.GetSecAvgAyGz(i, AvgAcc, AvgGyro);
            printf("%3d\t%4d\t%4d\t%6d\t%6d\t%6d\t%6d\r\n", i, L, R, MaxAcc, AvgAcc, MaxGyro, AvgGyro);
    }
}
```

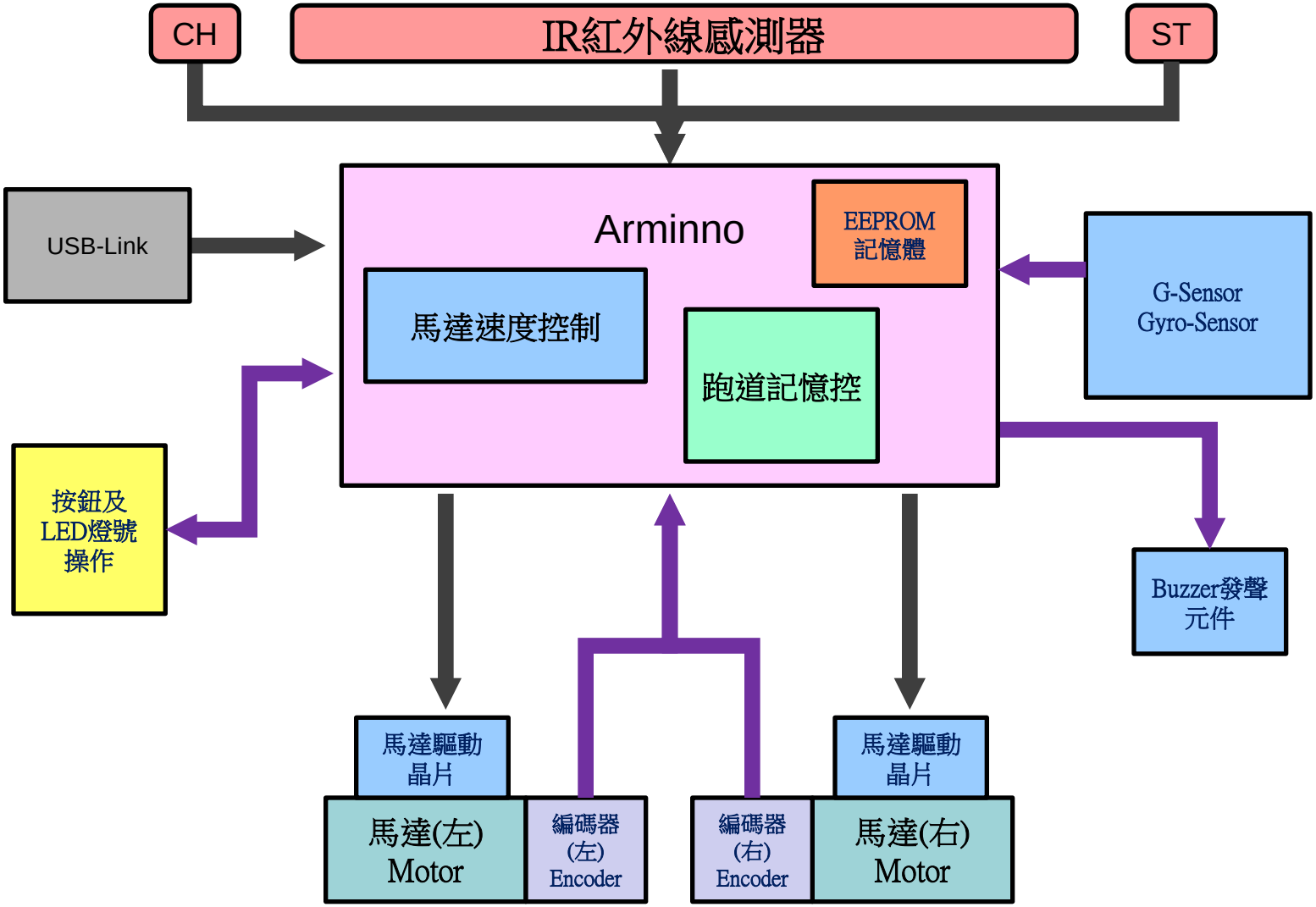
直線加減速測試

實驗十三: 繞圈加減速控制

```
#include "arminno.h"
#include "innoRacer2.h"
innoRacer2 myRacer;
#define FREQ_CTRL                4
#define CROSS_COUNT          14 //設定交叉軌道判斷長度
#define STOP_TACH                100
const short Normal_Speed_R = 130; //設定右輪直線速度值
const short Normal_Speed_L = 130; //設定左輪直線速度值
const short Crazy_Speed_R = 150;
const short Crazy_Speed_L = 150;
//設定PID參數
const short kP=22;
const short kI=0;
const short kD=240;
// 模組各別感測階段的誤差值
#define PID_SCALAR               4
const short sStrErr[ ]={18,30,48,78,126,144,330,534};
unsigned char bStatus;
unsigned char bIR;
// 設定模組各項預設值
void Init(void)
{
   //設定PID參數
   myRacer.SetCtrlFreq(FREQ_CTRL);
   myRacer.SetP(kP);
   myRacer.SetI(kI);
   myRacer.SetD(kD);
   myRacer.SetScalar(PID_SCALAR);
   //設定速度相關資料
    myRacer.SetStraight(Normal_Speed_L , Normal_Speed_R );
    myRacer.SetIRMode(1);
   //設定誤差值
   myRacer.SetErrScale(sStrErr[0], sStrErr[1], sStrErr[2],sStrErr[3],
                       sStrErr[4], sStrErr[5], sStrErr[6], sStrErr[7]);
   myRacer.SetCrossCount(CROSS_COUNT); //設定交叉軌道判斷距離
   myRacer.SetMotorDeadZone(136);
}
```

```
int main(void)
{
   Init();
   myRacer.AutoBeep(1);          //設定經過曲率變化點發出提醒
   //判斷軌道是否在中間
   do {
         myRacer.GetIR(bIR);
       } while((bIR & 0x04) == 0x00);
    myRacer.BuzzerOn();
    Pause(20000);
    myRacer.StartRec(1);          // 啟動記錄
    //判斷是否進入記錄狀態
    do {
          myRacer.GetRecStatus(bStatus);
       } while(bStatus != 1);
     //啟動PID速度控制
      myRacer.SpdCtrlOn(0)

     //判斷是否偵測到開始點
     do {
           myRacer.GetRecStatus(bStatus);
        } while (bStatus != 2);
      //修改速度值
     myRacer.SetStraight(Crazy_Speed_L, Crazy_Speed_R );

      do{
          myRacer.GetTotalLen(LenL,LenR);
        }while(LenL < STOP_TACH);
      //結束記錄
     myRacer.StopRec();
      myRacer.StopDual();
}
```

InnoRacer整合控制


CH
IR紅外線感測器
ST
USB-Link
Arminno
EEPROM
記憶體
馬達速度控制
跑道記憶控
G-Sensor
Gyro-Sensor
按鈕及
LED燈號
操作
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)
Encoder
編碼器
(右)
Encoder
馬達(右)
Motor

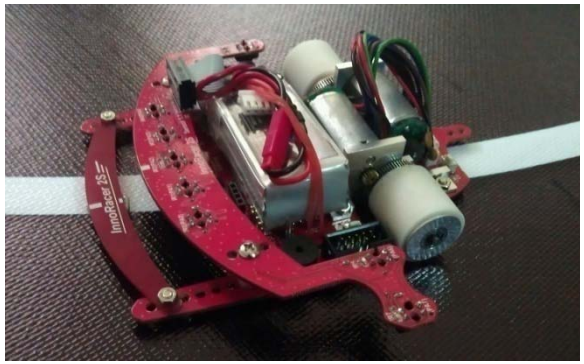
實驗十四: 繞圈加速整合控制

1. 請直接參考範例程式
2. 試著調整各項參數
3. Button 2 跑道紀錄
4. Button 1 開始競速

如何才能讓車子跑的快?

如何才能讓車子跑的快?

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



革命尚未成功 同志仍需努力

Passion for innovation

The
END