修平科技大學

資訊網路技術系實務專題

兩車聯行循跡自走車製作

指導老師：陳松雄 老師

學生：BN102028 林秉威

BN102058 黃思碩

中華民國 106 年 6 月 9 日

修平科技大學

資訊網路技術系實務專題

兩車聯行循跡自走車製作

指導老師：陳松雄 老師

學生：BN102028 林秉威

BN102058 黃思碩

指導老師:_____

評審老師:_____

_____

_____

中華民國 106 年 6 月 9 日

目錄

摘要

現在我們的生活中汽機車是很重要的交通工具，有的人是把它當成賺錢吃飯的工具，近年來汽車電子更是被廣泛運用在電子輔助汽車駕駛上，輔助的部分包含與前車保持距離，或是車道偏離輔助系統。如今台灣的機車族那麼多，我們是不是也能把這套系統轉移到機車上，所以這次我們藉由 iPOE A1 輪型機器人去更深入了解這些輔助系統，之後再更深的研討如何運用在機車上，保護機車族的行車安全。

由上述可知，汽車電子應用在輔助駕駛、自動駕駛已經位於相當重要的地位，然而市面上針對輪型機器人或自走車的介紹，都還只著重於競速、迷宮等功能上的介紹使用，這次就在我們的專題中做點不一樣的測試，其實汽機車的使用已經與我們每天息息相關，只要再加上各種感應裝置，就可以使車輛更安全，減少不必要的事故發生。

第一章前言

1.1 製作動機與目的

現在的社會中許多人都是把汽機車作為交通工具，或是賺錢的工具，在上下班的途中難免會精神不濟而未注意前方路況或是偏離原本車道而，產生事故。

為了避免事故的發生我們想要藉由 iPoe A1 的這台下去模擬平常路上的路況再太接近前方時會自動煞車減速或是偏離車道時會提醒駕駛人注意前方路況，我知道就以目前的技術有些高級的汽車已經能做到，但我更希望也能在運用在機車族上，因為大部分的學生都是以機車為代步工具所以如果機車上能加上這套系統就更能保護更多機車族，減少交通事故的發生。

1.2 軟硬體需求介紹

1.2.1 iPOE A1 核心基本架構

電路採用 Arduino Mega 2560 Rev3 架構

微控制器核心採用 ATmega 2560

有 54 組數位 I/O 端（其中 14 組可做 PWM）16 組類比輸入端

4 組 UART（硬體串列埠）

時脈頻率：16 MHz。

1.2.2 軟體需求

Arduino IDE，Android 介紹

Android 分為三個部分，我們先簡單的介紹：

第一部分 Android 的起源：Android 系統最初由安迪・魯賓(Andy Rubin)等人開發製作，最初開發這個系統的目的是創建一個數位相機的先進操作系統。

第二部分 Android 的特色：目前使用 Android 系統的手機數量已超越 iOS 系統，成為全球使用量最大的手機作業系統 Android，應用需求也越來越大。

第三部分 AndroidStudio 的介紹：我們目前用 Arduino YUN 製作的智慧居家監控裝置就是用 Android Studio 製作出來的。

Android 的起源:

Android 系統最初由安迪・魯賓(Andy Rubin)等人開發製作，最初開發這個系統的目的是創建一個數位相機的先進操作系統;但是後來發現市場需求不夠大,加上智慧型手機市場快速成長,於是 Android 被改造成一款面向智慧型手機的作業系統。於 2005 年 8 月被美國科技企業 Google 收購，2007 年 11 月，Google 與 84 家硬體製造商、軟體開發商及電信營運商成立開放手機聯盟來共同研發改良 Android 系統，隨後 Google 以 Apache 免費開放原始碼許可證的授權方式，發佈了 Android 的原始碼,讓生產商推出搭載 Android 的智慧型手機，Android 作業系統後來更漸漸的拓展到平板電腦及其他裝置上。

Android 的特色:

高完整度的軟體整合應用(True App Integration)：在 Android 系統中，很多的功能本身設計時就與系統無縫的整合在一起，文中 Google Voice 例子來說明,撥號器是手機語音通話時最常用的功能，你可以輕易地變更預設的程式（可參考 Android 預設程式的變更)，如同在 Windows 系統上改變預設開啟的程式對應一樣,如將撥號的功能以 Google Voice 為預設程式，則只要有關的電話連結（如地圖上的電話號碼連結)，當點擊時會以 Google Voice 取代原先撥號器，讓任何地三方軟體輕易的語系統整合，這類對應用程式如電話、簡訊、語音郵件和瀏覽器應用可以無縫地整合到系統中。雖然 iPhone 也將會有 Google Voice，但可能沒辦法與系統整合，想要語音撥號得另外啟動，感覺就沒那麼便利。

完美的Flash的支援:當你同時使用Android與iPhone手機上網，你可能會發現差別很大，不管是瀏覽網頁、觀看視頻，還是玩一些遊戲，安裝 Flash 之後，使用者都可以獲得更多的內容與互動性，Android 最大的優勢是支援 Flash Player，因為目前網頁大部分還是使用 Flash 動畫的影子，iOS 最為人詬病的就是不支援 Flash。

ArduBlock 介紹:

以往微電腦控制均具有電子背景技術人員學習,但這次我們的輪型機器人是選擇 arduino 來編寫程式，運用 arduino 的外掛程 ardublock(圖形化程式)來進行編寫。

ArduBlock 是屬於 Arduino 官方程式環境下的協力軟體，必須依附 Arduino 軟體下進行,兩者的差別在於一個是本文式的程式環境另一個是圖形化積木結合。

ArduBlock 只是依附在 Arduino IDE 下的圖形介面，當寫好程式後燒錄進 Arduino 記憶體中，之後全權由 Arduino 控制。

1.2.3 硬體需求

有了軟體需要後，尚需有硬體上的支援，需要支援的如圖(1)：PC 主機一台。

項目	說明
微控制器	ATmega 2560
紅外線循跡感測器	前 3 後 5，共 8 顆 cnb10010 紅外線光反射型感測器。
超音波測距 HC-SR04	使用電壓：DC5V；靜態電流：小於 2mA；感應角度：大於 15 度；探測距離：2cm~450cm，精準度可達 0.2cm。
紅外線測距	紅外線光反射器，主要供平衡車使用。
馬達編碼器	採用 5 葉光柵，減速比 30:1 情況下，300pulse/圈。
N20 微型金屬減速馬達	2 顆，額定 6V，空載轉速 13000RPM，減速比 30：1，空載電流 30mA
輪子	2 個，直徑：42mm 寬：19mm 材質：ABS 塑料+橡膠。
繪圖型 LCD Nokia 5110	84*48 繪圖型點陣液晶 3.3V。

項目	說明
蜂鳴器	3-7V 電磁式無源蜂鳴器。
電池	鋰電池 3.7V×2 顆。
按鈕與 LED	設定與指示燈。

(圖 1)硬體簡介

第二章元件介紹

本專題最主要使用到紅外線循跡感測器、超音波感測器、直流馬達，我們就藉由這幾個主要的元件解說。

2.1 紅外線循跡感測器

相關知識:

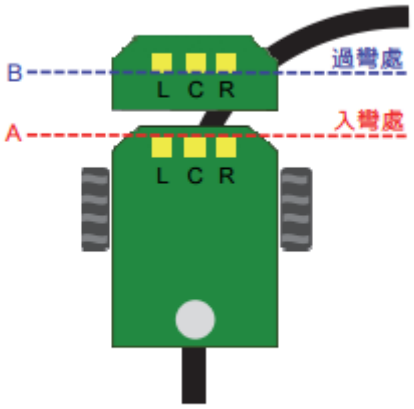
CNB10010 是使用在輪型機器人線循跡的一般型反射式紅外線感測器，該感測器的工作原理是由 IR LED 發射光線，再由光電晶體來量測反射量，而因為白色的表面比黑色的表面反射更多的光，所以當面對白色的表面時電容放電比面對黑色時快。應用此原理，此紅外線感測器就可以被用來偵測白色和黑色區域之間的差別，可以用來幫助機器人確認地面上的白/黑線位置，如此機器人就可以沿著現到達目的地。

Ipoe A1 輪型機器人則是使用前 3 顆後 5 顆的 CNB10010 反射式紅外線感測器來做為線的偵測器，它也可以用來做為近接或反射式感測器。

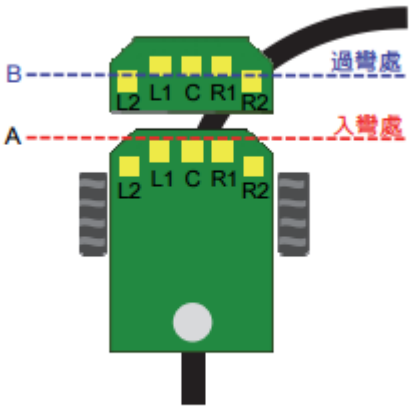
前 3 顆 CNB10010 反射式紅外線感測器，因為間格距離較大，不適合用來偵測路徑方向，但可以用來偵測路徑旁的提示點和前方路線的狀態，也就是說，當感測器偵測到左方提示點時，即準備進入轉彎路徑，第二次感測器在偵測到左方提示點時，即表示轉彎完成，恢復直線路徑，而後面 5 顆感測器因為間隔距離較小，適合用來偵測路徑方

向，即所謂的循跡感測，也是自走車循跡最重要的感測器。

如圖(2)、(3)



(a) 三個模組



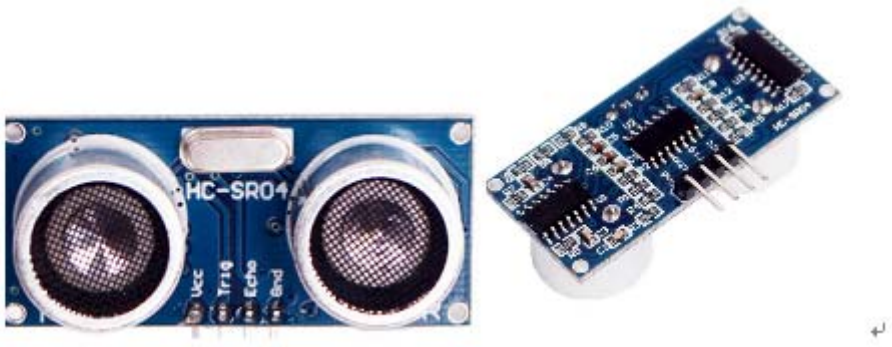
(b) 五個模組

圖(2)紅外線模組　　圖(3)紅外線模組

如圖(2)所示使用三個紅外線模組，進入軌道 A 點入彎處，紅外線感應到彎道，回傳至為控制驅動左右邊馬達控制，但若速度太模組會來不及感應則會衝出軌道，三組紅外線模組優點成本低，缺點車速慢。

如圖(3)所示五個紅外線模，組進入軌道 A 點入彎處，紅外線感應到彎道，回傳至為控制驅動左右邊馬達控制，但若速度太快，R1 紅外線模組來不及感應繼續直線前進時進入 B 點過彎處 R2 紅外線模組仍可感應到彎道使車能順利轉彎，優點車速快缺點成本高。

2.2 超音波感測器介紹



圖(4)超音波元件

超音波感測器（國外好像把它叫作 PING))) sensor）是由超音波發射器、接收器和控制電路所組成。當它被觸發的時候，會發射一連串 40 kHz 的聲波並且從離它最近的物體接收回音。超音波是人類耳朵無法聽見的聲音，因為它的頻率很高。可以做即時觀看，當感測器偵測到我們所設定的偵測值，可以第一時間傳送資訊至雲端並可由手機 APP 來控制是否自動手動關閉電器。

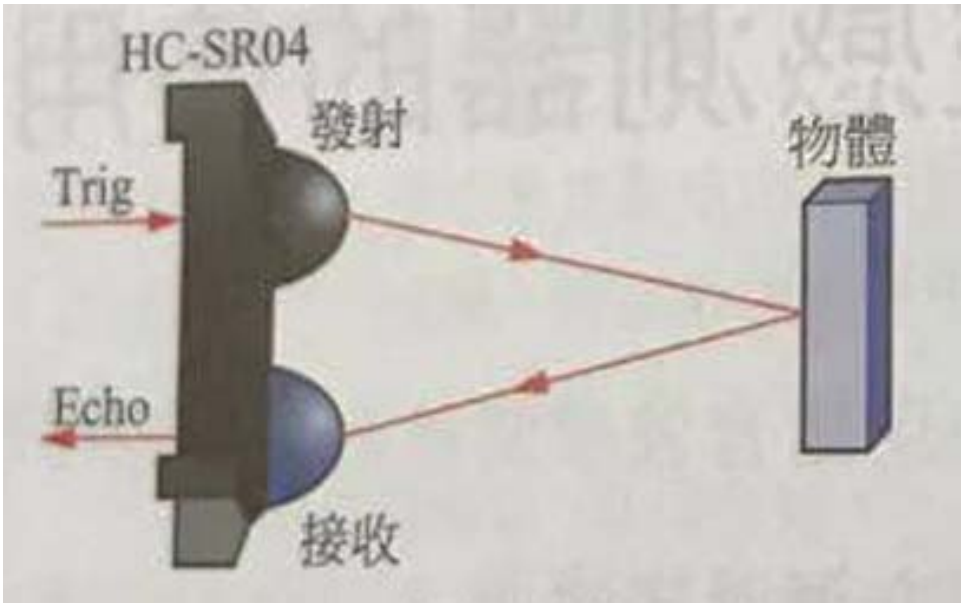
人類耳朵能聽到的聲波頻率約在 20Hz~20KHz(赫茲)之間。當聲波的振動頻率小於 20Hz 或大於 20KHZ 時，便聽不見了。因此，頻率高於 20KHz 的聲波稱為「超音波」(Ultrasonic)，在此將介紹一款市面上最為成熟且最常見的超音波測距模組 HC-SR04 來做為本專題測量距離的主角，其所使用之超音波頻率為 40KHz。

聲音在空氣中的傳播速度大約是每秒 331+0.6t 公尺，傳播速度會受溫度的影響，溫度愈高，則傳播速度愈快。現假設以 340 公尺 / 秒計算，340*100/1000000=0.034 公分/微秒，可知聲音傳播 1 微秒(US)可行進 0.034 公分，因此只要測出發射出去到接收的總時間，乘以每

微秒型進的距離(0.034 公分)，就可測出總行進距離了，但要特別注意的是，因為超音波從發射到返回是兩段距離，因此在計算時必須將結果除以 2 才是正確的物體距離。所以可以利用底下的公式算出物體距離(距離單位為公分，其中 timing 是測量得到的音波傳播時間)：S=timing*0.034 / 2。

超音波如何測距離:

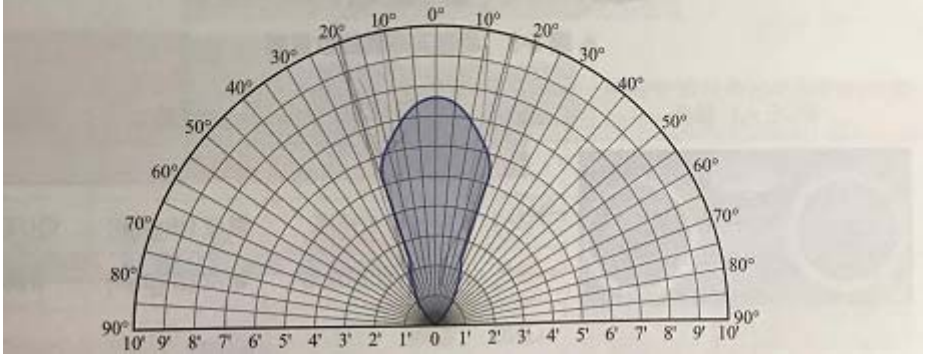
超音波測量距離的方法如下圖(5)所示，是測量聲音在感測器與物體之間往返經過的時間。



圖(5)往返經過時間

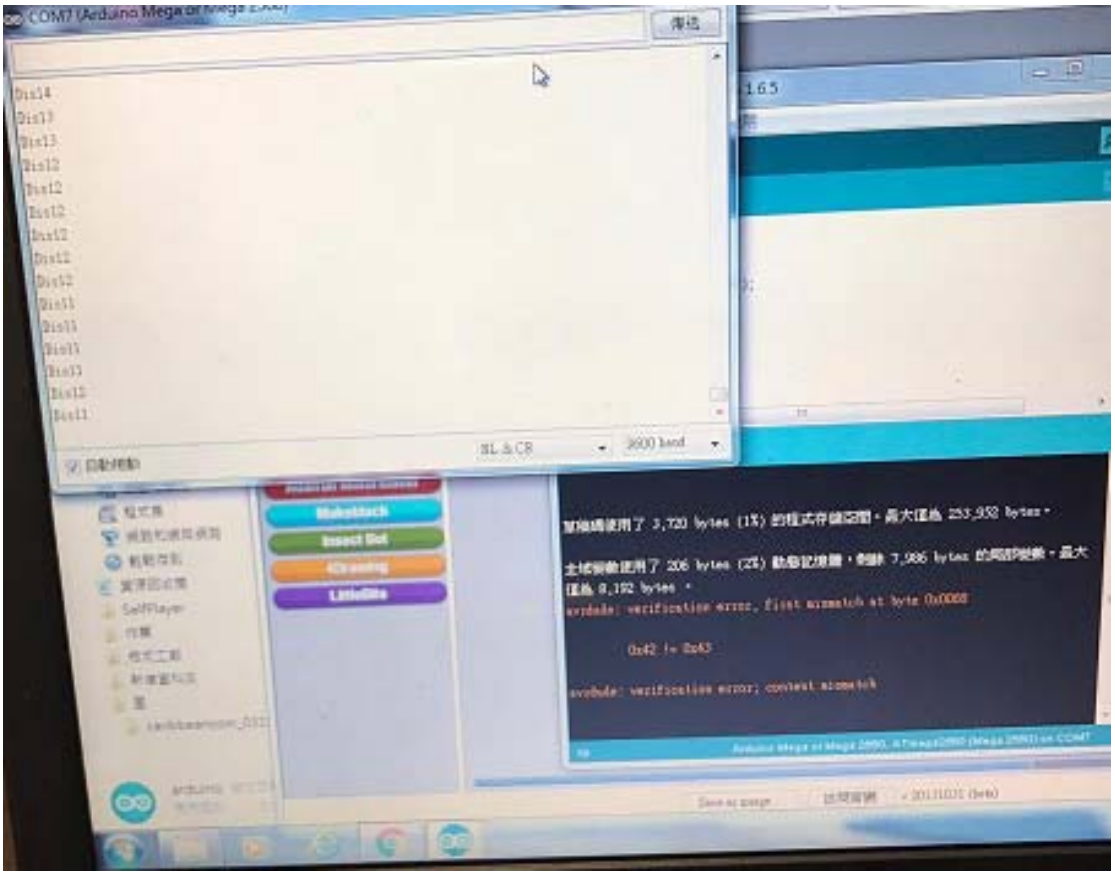
超音波規格及感應角度如圖(6)

工作電壓	5 V
靜態電流	< 2 毫安
輸出信號	電位信號，高電位 5V，低電位 0V
感應角度	< 15 度
檢測距離（自稱）	2 厘米～ 450 厘米
精確	3 毫米
輸入觸發信號	10us 的 TTL 脈衝
回波信號	TTL 輸出信號 PWL
接腳	① 引腳 VCC ② Trig（T） ③ Echo（R） ④ GND



圖(6)超音波規格

測試超音波感測距離如圖(7)



圖(7)程式測試

如上圖測試超音波感是否準確，接著測量超音波距離障礙物的距離。

2.3 直流馬達介紹及使用方式

相關知識:

在全球節能減碳的共同意識下，直流馬達的使用漸漸頻繁，從電動車、直流變頻冷氣機、直流風扇、直流吊扇、直流抽排風機等的電氣設備一一出現。

隨著電動車開發技術漸漸的純熟及節能概念下，電動機已成為未來車輛的一大趨勢，而電動車最重要的直流馬達也就相形越來越受重視，本單元介紹目前最受歡迎的直流馬達及如何控制直流馬達的啟動、停止和正反轉，讓讀者對直流馬達有一基本的概念。

直流馬達介紹:

直流馬達(direct current motor, DC motor)是最早發明能將電力轉換為機械功率的裝置，其具有良好的線性特性、簡單容易控制及輸出功率大的優點，所以仍是目前最常應用於變速控制的馬達。

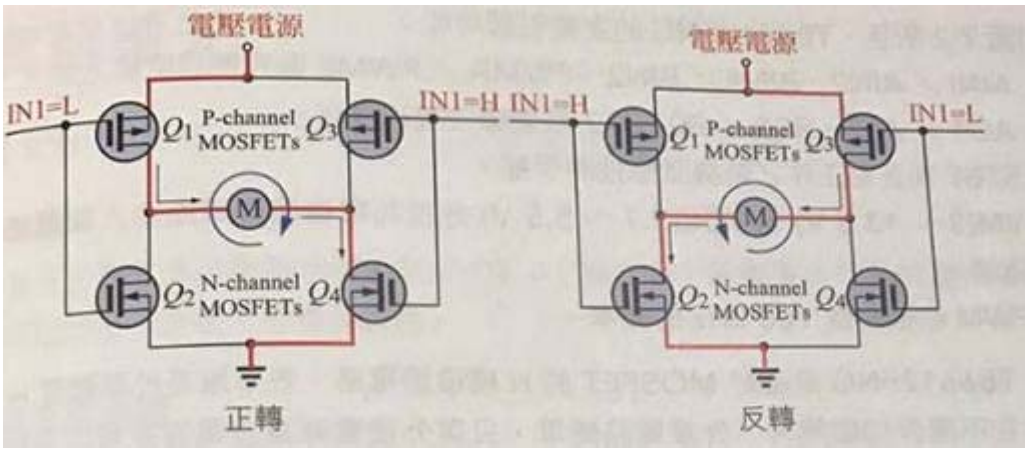
轉速控制:

在直流馬達控制系統中，為了減少流經馬達繞線電流及降低功率消耗等目的，常常使用脈波寬度調變信號(PWM)來改變輸出脈波寬度或頻率來改變馬達的轉速。

轉向控制:

一般 MCU 像 Arduino 輸出大約只有 40mA 的電流，因此如果要驅動直流馬達的話，就必須使用由兩個電晶體組成的「精靈頓電路」來做電流放大，但這樣的電晶體並不控制電流方向，只能使電動機往單一方向運轉，如果要改變方向就必須要使用能改變電流流向的電路，

這時就要用所謂「H Bridge」也就是俗稱的「H 橋」電路，如圖 8-Bridge 動作原理說明如下。



圖(8)做動原理

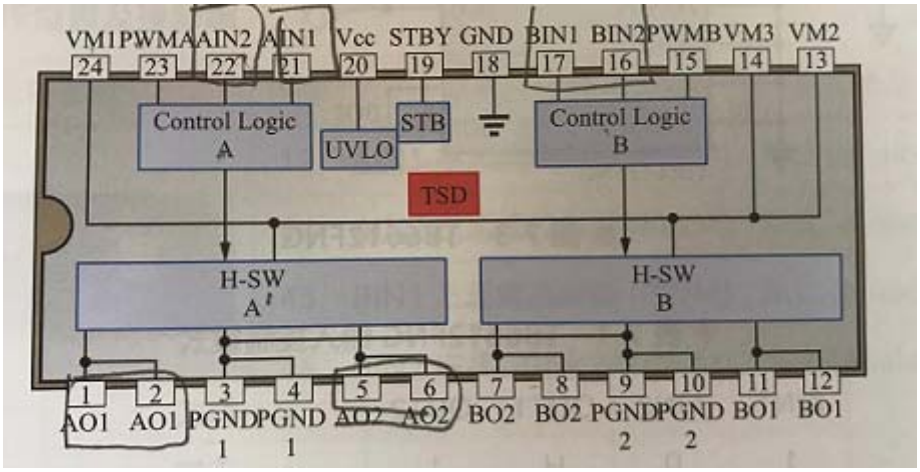
圖(8)一個一個典型的 H-Bridge 馬達驅動電路，其動作原理如下。

當 1N1=HIGH，1N2=LOW 時，Q1,Q4 電晶體 ON 而 Q3,Q2 為 OFF，即可使馬達正轉。

當 1N1=LOW，1N2= HIGH 時，Q3,Q2 電晶體 ON 而 Q1,Q4 為 OFF，即可使馬達反轉。

本專題板是使用 TB6612 雙驅動 1C， TB6612FNG 是東芝半導體公司所生產的一款直流電機驅動器，有 4 種控制模式：正轉/反轉/煞動/停止，且具有大電流 MOSFET-H 橋結構，雙通道電路輸出，可同時驅動 2 個馬達。TB6612FNG 每通道輸出最高 1A 的連續驅動電流，啟動峰值電流達 2A/3A(連續脈衝/單脈衝)。

直流馬達腳位介紹:



圖(9)腳位介紹

1、A1N1 / A1N2、B1N1/ B1N2、PWMA/PWMB 為控制信號輸入端。

2、AO1/AO2、B01/B02 為 2 路電機控制輸出端。

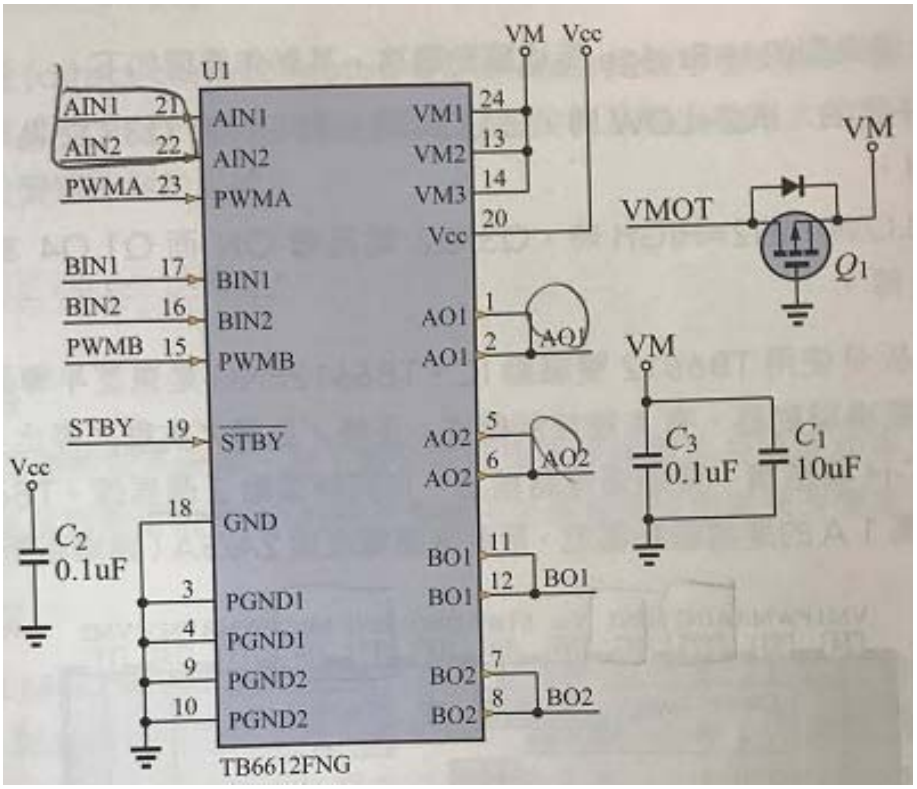
3、STBY 為正常工作/待機狀態控制引腳。

4、VM(3~13.5 B)和 VCC(2.7~5.5V)分別為電機驅動電壓輸入和邏輯電平輸入端。

5、PWM 信號高達 100KHZ 的頻率。

TB6612FNG 是基於 MOSFET 的 H 橋積體電路，效率遠高於晶體管 H 橋驅動器。它不需外加散熱片，外接電路簡單，只需外接電源濾波電容就可以直接驅動馬達，方便縮小系統尺寸。

如下圖 10、11 為 TB6612FNG 腳位介紹及輸入控制模式



圖(10)腳位

AIN1	AIN2	OUT1	OUT1	模式
1	0	H	L	正轉
0	1	L	H	反轉
1	1	L	L	煞車
0	0	X	X	停止

圖(11)輸入控制模式

第三章製作過程、程式發展及程式說明

3.1 專題流程

以下就是我們製作這次專題的流程:

(1)我們先到圖書館一起討論專題的題目，討論後我們的兩車聯行循跡自走車製作

(2)找到相關資料及書籍來研究

(3)跟學校借需要用到的設備

(4)了解所有元件的功能跟使用方式

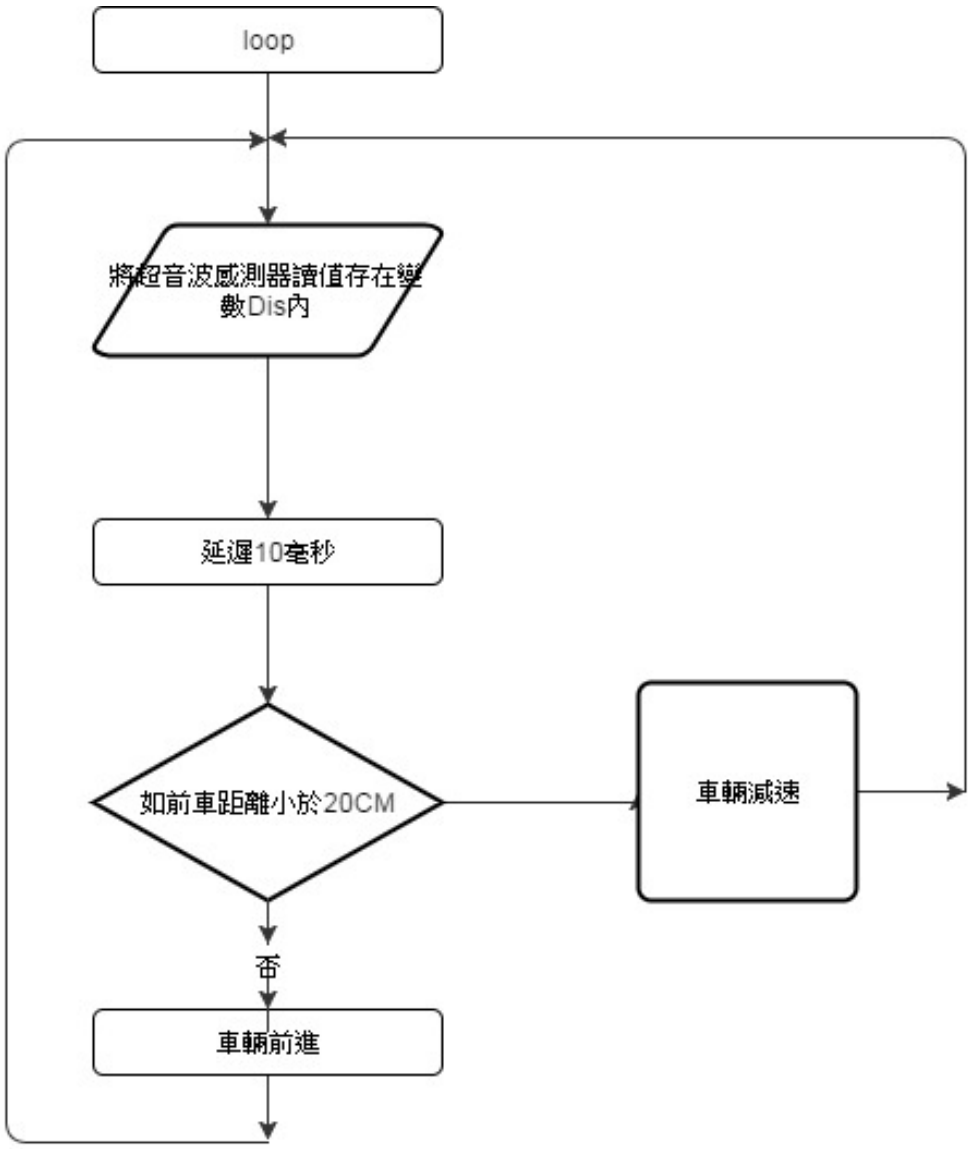
(5)開始撰寫程式及測試和修改

(6)專題報告的內容。

3.2 程式發展

程式設計大都有一定規則可循，一般程式會根據需求將程式分成許多子功能系統，先完成各子功能程式，再將各個字程式做’結合成一個完整程式。當然如果程式很小也可以從頭寫到最後，依每個人寫作習慣而定。在姐寫程式的過程中發現必須分兩段去寫，一個是自走車本身的程式，另一個是個個元件要編譯他的含式，執行是必須放置草稿夾中，才能正確地執行程式。

3.3 軟體流程圖

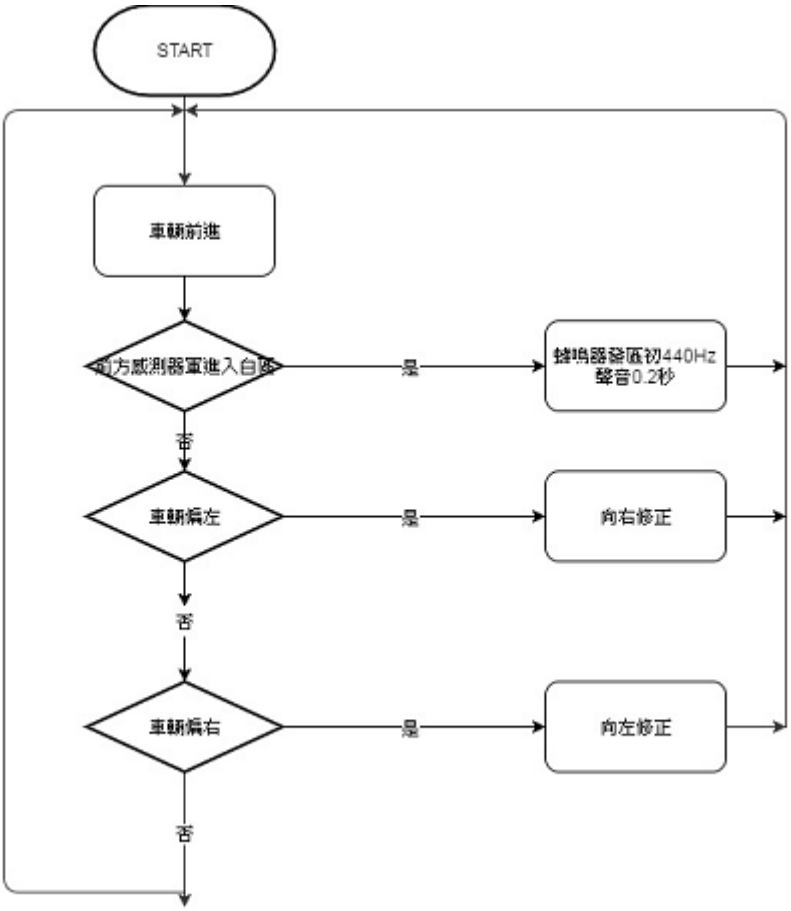


3.4 個別作動的感測器

3.4.1、紅外線感應

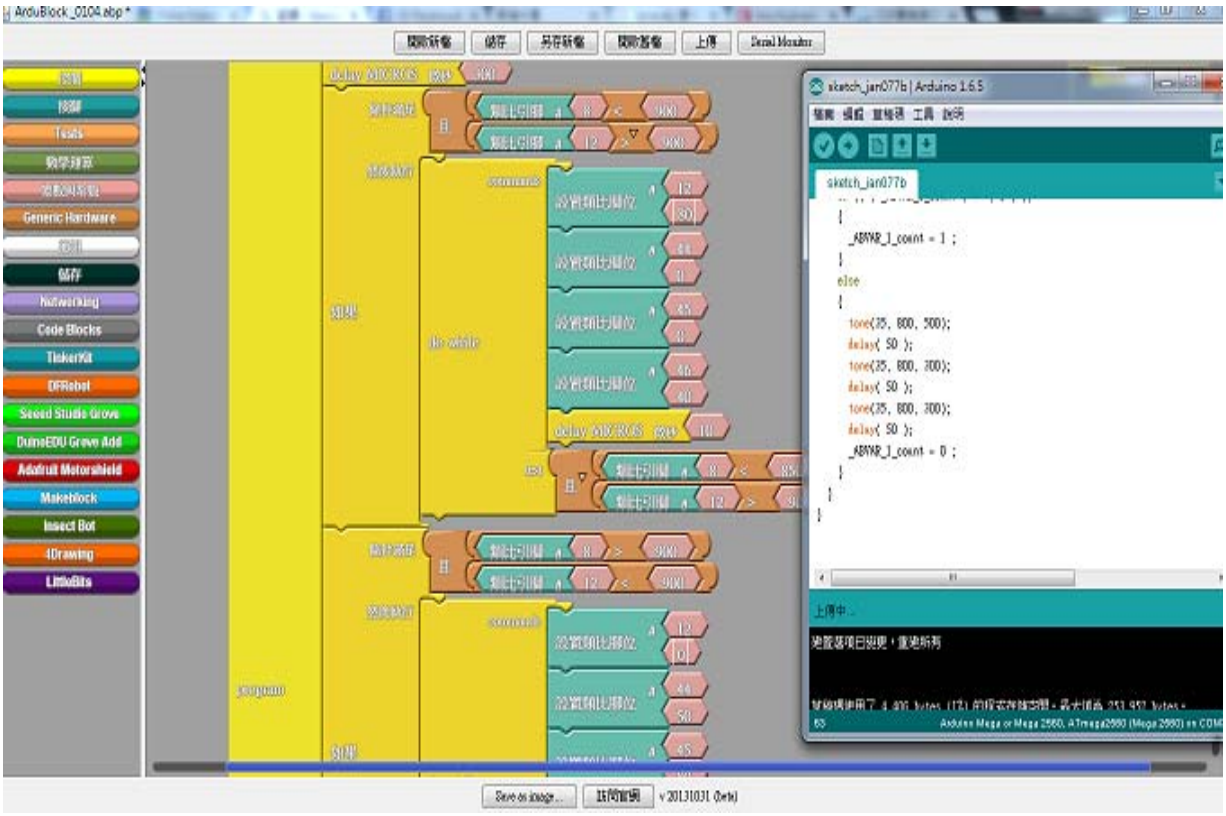
利用紅外線循跡感測器繞著八字形軌道前進在遇到交叉點時會發出蜂鳴器的聲音。

下圖為程式執行的流程，看完流程後再進行程式撰寫。



圖(12)程式流程圖

程式實作



圖(13)程式內容

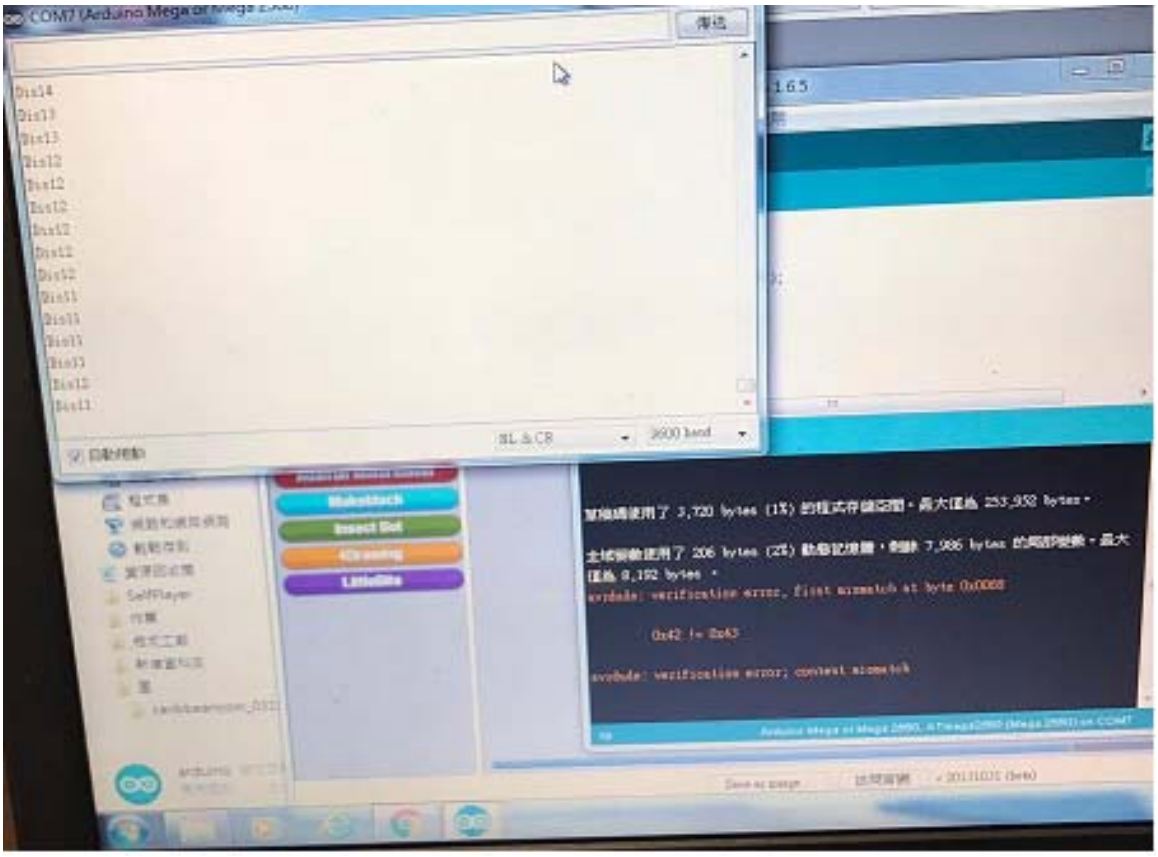
3.4.2、超音波

利用超音波感測器進行對障礙物距離量測，測量後才能知道車輛的安全包持距離大概要多長的距離。

下圖為程式流程圖

圖(14)程式流程圖

執行結果

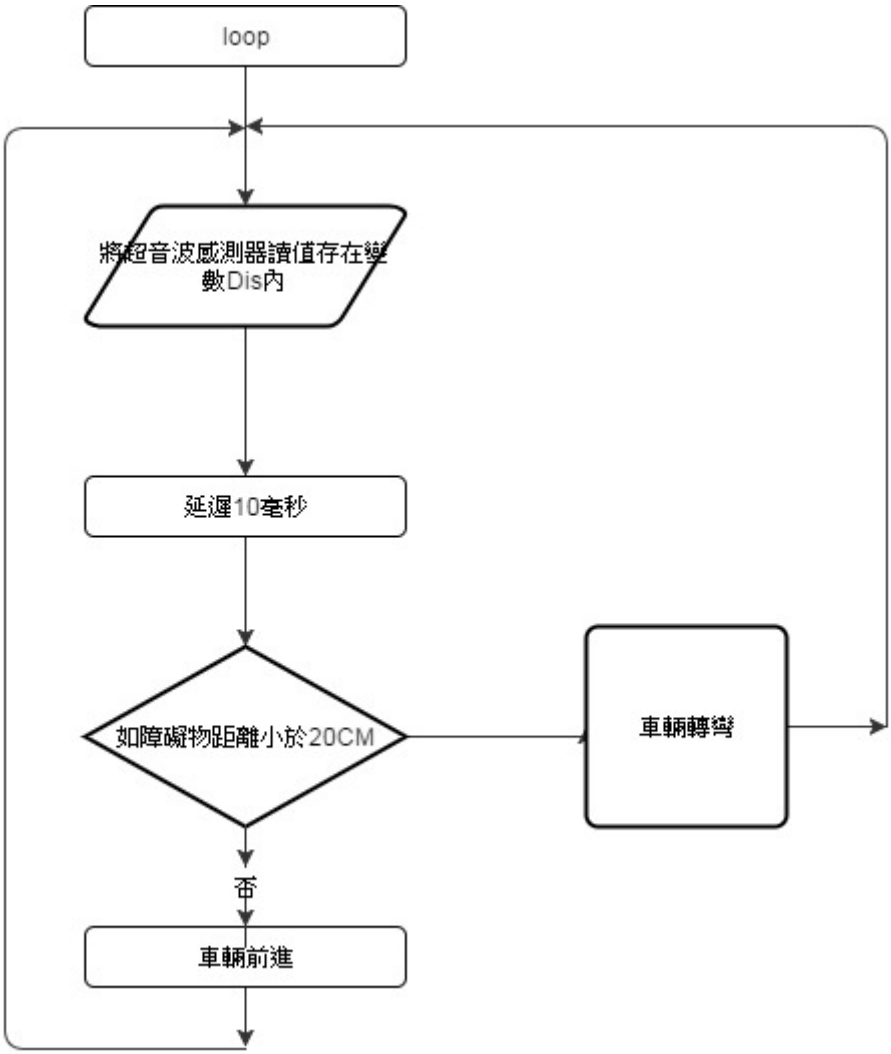


圖(15)程式執行

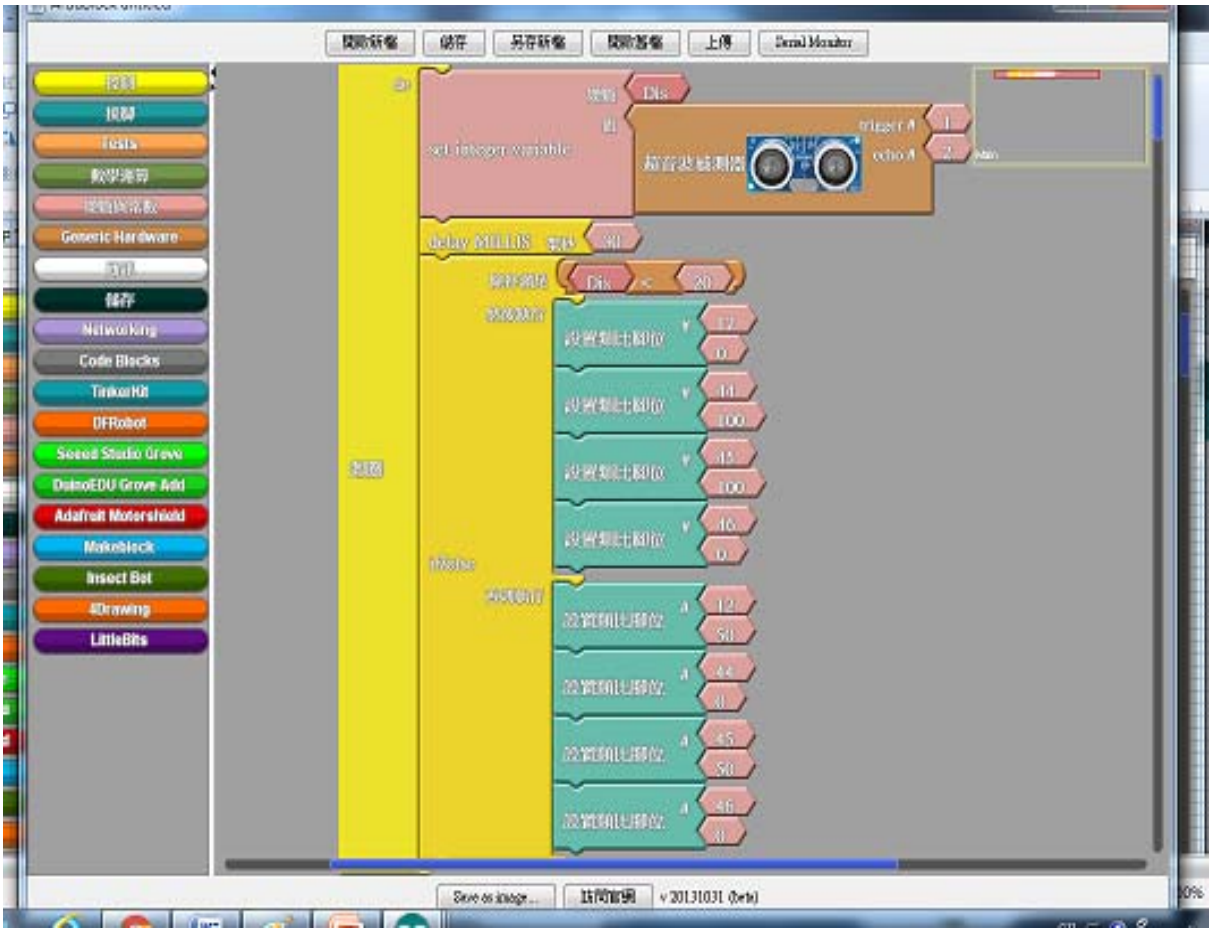
3.4.3、直流馬達

利用前兩個感測器與直流馬達做結合，當自走車輛遇到前方有障礙物時會自動的煞車或轉彎後再繼續地完成後續的動作。

程式流程圖



圖(16)程式流程圖



圖(17)程式內容

藉由以上的幾個分解動作即對感測器的使用方式了解，再來就是把，所有做一個整合，讓自走車能夠順利地轉彎也能夠閃避障礙物。

3.4.4、兩車聯行循跡自走車，防止追撞前車系統程式

1、自走車程式

```
boolean WaitData() {
  boolean dt=false;
  if (Serial2.available()) {      // 判斷串列埠緩衝區有無資料
    // get the headline word
    if (Serial2.find("SP")) {        // 判斷串列資料的 head 是否為 SP
        maximum=Serial2.parseInt();  // 從串列埠緩衝區中讀取下一個有效的整數資料
        maximum=constrain(maximum,0,255);
        Kp=Serial2.parseFloat();         // 從串列埠連續讀取 3 個有效的浮點數資料
        Ki=Serial2.parseFloat();    // 分別存入 PID 的三個參數 Kp, Ki, Kd
        Kd=Serial2.parseFloat();
        Serial2.print("Speed=");         // 在 Serial Monitor 2 中顯示訊息
        Serial2.print(maximum);
        Serial2.print("\tKp=");
        Serial2.print(Kp);
        Serial2.print("\tKi=");
        Serial2.print(Ki);
```

```
            Serial2.print("\tKd=");
            Serial2.println(Kd);
            tone(Buzzer,400,100);       // 嗶 1 聲通知 iPOE-A1 有收到新設
定值
            dt=true;          // 設定有傳回值
        }
        Serial2.flush();              // 清除串列埠緩衝區
    }
    return dt;
}

void Pause(byte ButtonPin)
{
    while (digitalRead(ButtonPin)==HIGH) {    // 若按鍵未按則繼續
        if (WaitData()) break;     // 若有藍牙資料傳入則跳出迴圈
        delay(5);
    }
    delay(50);   // 等待開關彈跳時間
    while (digitalRead(ButtonPin)==LOW);// 等待按鍵放開
}
// 顯示電池電壓
void DisplayBattery()
```

```
{
   float v;
   int x, y;
   // 若按鍵未按則繼續顯示
   while (digitalRead(AButton)==HIGH) {
         myGLCD.clrScr();
         myGLCD.print("Voltage Test", CENTER, 0);
         // 計算實際電壓值並顯示
         v=analogRead(A6)*5.00/512;
         myGLCD.printNumF(v, 3, CENTER, 15);
         // 顯示條狀圖
         myGLCD.drawRect(5,30,78,35);  // 畫一個矩形
         x=map(v*100,680,850,6,77); // 將電壓值 6.8~8.5V 轉換至 x 軸
         x=constrain(x,6,77);
         for (y=31; y<35; y++)       // 根據電壓轉換後的 x 填滿矩形
            myGLCD.drawLine(6, y, x, y);
         myGLCD.update();// 更新顯示
         delay(50);
   }
   delay(50);  // 等待開關彈跳時間
   while (digitalRead(AButton)==LOW);  // 等待按鍵放開
}
```

```
// 直線前進並修正的副程式
void Running(int dis)
{
    position = qtra.readLine(sensorValues,1,1);  // 讀取後 5 感測器的數值
    proportional = (int)position - 2000;         // 計算與中間值的比例誤差量
    // 計算位置的變化量(derivative)與累積量 integral
    derivative = proportional - last_proportional;
    integral += proportional;
    // 記住上次的比例誤差量，並計算輸出控制量 power_difference
    last_proportional = proportional;
    power_difference = proportional*Kp + integral*Ki + derivative*Kd;
    // 限制 power_difference 的最大值範圍
    if (power_difference > maxs) power_difference = maxs;
    if (power_difference < -maxs)    power_difference = -maxs;
    // 計算左、右馬達實際控制量
    if (power_difference < 0) {
        LM=maxs + power_difference;
        RM=maxs;
    }
    else {
```

```
        LM=maxs;
        RM=maxs - power_difference;
    }

        if (dis > 10 || dis == -1)
      myMotor.setSpeeds(LM, RM);
        else
          myMotor.Brake();

}
// 檢測並顯示感測器狀態的副程式
void display_check_sensor(byte ButtonPin)
{
  while (digitalRead(ButtonPin)==HIGH) {        // 若按鍵未按則繼續
顯示
        position = qtra.readLine(sensorValues,1,1);  // 讀取後 5 感測器
的數值
        pos0 = qtra0.readLine(sensor0Values,1,1);       // 讀取前 3 感測
器的數值
        // 顯示位置值
        myGLCD.clrScr();
        myGLCD.printNumI(position, LEFT, 0);
```

```
        myGLCD.printNumI(pos0, 50, 0);
        // 橫向條狀圖 感測器顯示
        for (unsigned char i = 0; i < NUM_SENSORS; i++)
          myGLCD.drawRect(i*7, 47, i*7+6,
47-map(sensorValues[i],0,1024,0,35));
        for (unsigned char i = 0; i < 3; i++)
          myGLCD.drawRect((i+8)*7, 47, (i+8)*7+6,
47-map(sensor0Values[i],0,1024,0,35));
        // 更新顯示
        myGLCD.update();
        delay(50);
    }

    delay(50);   // 等待開關彈跳時間
    while (digitalRead(ButtonPin)==LOW);     // 等待按鍵放開
}

void IR_calibrate()
{
    digitalWrite(RED_LED, HIGH); // 打開紅色 LED, 顯示現在感測器校
正模式
    // 感測值校正
```

```
  for (counter=0; counter<60; counter++)
  { // 原地右轉、左轉移動
    if (counter < 16 || counter >= 45)
        myMotor.setSpeeds(75, -75);
    else
        myMotor.setSpeeds(-75, 75);

    // 校正感測器的最大最小值
    qtra.calibrate();
    qtra0.calibrate();

    delay(10);
  }
  // 依據校正後的感測值, 轉正回白線上
  for (counter=0; counter<60; counter++) {
    position = qtra.readLine(sensorValues,1,1);
    if (sensorValues[2]<200) break;
    delay(10);
  }
  myMotor.Brake();
  digitalWrite(RED_LED, LOW);        // turn off Arduino's LED to indicate
we are through with calibration }
```

2、元件含式庫

```
// iPOE-A1
//
//
//
#include <QTRSensors.h>
#include <LCD5110_Graph.h>
#include <IpoeA1.h>

IpoeMotor myMotor;      // 宣告馬達控制物件
LCD5110 myGLCD(24,25,26,27,28);  // 宣告 LCD5110 物件
extern uint8_t SmallFont[];   // 外部引入 Small 字型

#define NUM_SENSORS   5        // number of sensors used
#define NUM_SAMPLES   2        // average 4 analog samples per sensor
reading
int maxs, maximum = 100;             // 設定馬達行走最高速
int LM,RM;          // 左、右馬達控制量的變數

// sensors 0 through 5 are connected to analog inputs 0 through 5,
respectively
QTRSensorsAnalog qtra((unsigned char[]) {12, 11, 10, 9, 8},
```

```
NUM_SENSORS, NUM_SAMPLES, QTR_NO_EMITTER_PIN);
QTRSensorsAnalog qtra0((unsigned char[]) {15, 14, 13}, 3,
NUM_SAMPLES, QTR_NO_EMITTER_PIN);
unsigned int sensorValues[NUM_SENSORS]; //後 5 感測器的感測值陣列
unsigned int sensor0Values[3];        //前 3 感測器的感測值陣列
unsigned int position;          //後 5 感測器的位置值
unsigned int pos0;          //前 3 感測器的位置值

int counter;
// PID 控制器使用的變數
unsigned int last_proportional;   // 上次誤差值
long integral;                    // 誤差累積值
int proportional,derivative,power_difference;   // 誤差值, 誤差變化量
float Kp=0.2, Ki=0.0001, Kd=5.0;       // 行走的 PID 參數控制值
int count = 2;
boolean enableSonicSensor = true;

int _ABVAR_1_ultra_dis = -1 ;
int ardublockUltrasonicSensorCodeAutoGeneratedReturnCM(int trigPin,
int echoPin)
{
   long duration;
```

```
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(20);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    duration = duration / 59;
    if ((duration < 2) || (duration > 300)) return false;
    return duration;
}

void setup()
{
    pinMode(AButton, INPUT_PULLUP);   // IO 設定
    pinMode(RED_LED, OUTPUT);
    pinMode(Buzzer, OUTPUT);
    tone(Buzzer, 1000, 200);          //  開機的嗶聲

    myGLCD.InitLCD();              // LCD 初始化
    myGLCD.setFont(SmallFont);
```

```
  myMotor.motor_init();          // 馬達函式庫初始化

  Serial.begin(9600);
  Serial2.begin(57600);          // 藍牙裝置的鮑率
  analogWrite(LCDLight, 255);    // 關閉 LCD 背光
  DisplayBattery();              // 顯示電池狀態
  IR_calibrate();                // 紅外線感測值校正
  display_check_sensor(AButton); // 顯示感測狀態
}

void loop()
{
  int crossCnt=0;      // 經過交叉路口計數值
  int markFlag=0;      // 行進中,感測器停留在交叉路口的標記旗標
  myGLCD.clrScr();     // 顯示進行中的標題
  myGLCD.print("IPOE-A1", CENTER, 0);
  myGLCD.print("IRA Mission 8", CENTER, 10);
  myGLCD.print("#01", CENTER, 25);
  myGLCD.update();
  maxs = enableSonicSensor ? maximum : maximum - 20;          // 設
定車子的最高速
```

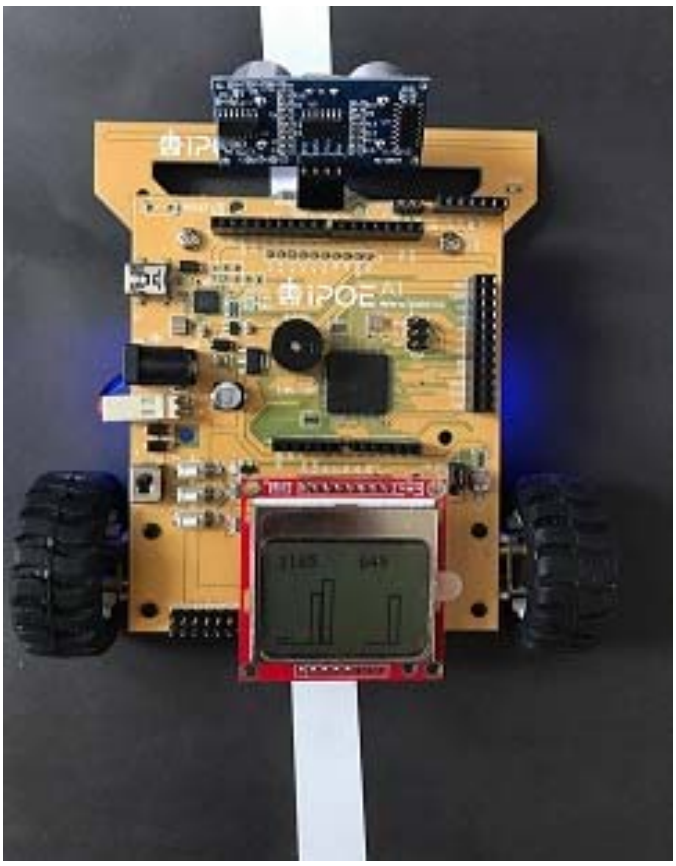
```
  while (1) {
        if (enableSonicSensor) {
          _ABVAR_1_ultra_dis =
ardublockUltrasonicSensorCodeAutoGeneratedReturnCM( 36 , 37 ) ;
        }

        Running(_ABVAR_1_ultra_dis);
        pos0 = qtra0.readLine(sensor0Values,1,1);

        if (sensorValues[0]<200 && sensorValues[1]<200 &&
sensorValues[2]<200) {      // 若是交叉路口(cross)
          markFlag++;    // 標記旗標+1
          if (markFlag==2) {  // 持續感測到交叉點 2 次以上(避免誤感
測)
            crossCnt++;       // 交叉點計數值加 1
//            if (crossCnt==1) {  // B 點的處理
//              myMotor.Brake();
//              for (counter=0; counter<3; counter++) {  // 發出嗶 3
聲
//                tone(Buzzer, 800, 166);
//                delay(333);
//              }
```

```
//                }
                if (crossCnt == count) {   // C 點的處理
                   myMotor.Brake();
                   myGLCD.clrScr();  // 顯示抵達終點
                   myGLCD.print("Finally!!", CENTER, 10);
                   myGLCD.update();
                   for (counter=440; counter<1000; counter+=50) {     // 發
出一小段音樂
                      tone(Buzzer, counter, 40);
                      delay(80);
                   }
                   Pause(AButton);
                   break;
                }
             }
          }
          else
             markFlag=0;     // 非交叉路口,標記旗標歸 0

          if (sensorValues[0]>800 && sensorValues[1]>800 &&
sensorValues[2]>800 && sensorValues[3]>800 && sensorValues[4]>800)
{
```

```
            myMotor.Stop();
            myGLCD.clrScr();
            myGLCD.print("Stop!!", CENTER, 10);
            myGLCD.print("Press Key A", LEFT, 20);
            myGLCD.print("or Parameter", CENTER, 30);
            myGLCD.update();
            Pause(AButton);
            break;
        }

        //myGLCD.clrScr();
       // myGLCD.print(String(_ABVAR_1_ultra_dis, DEC), CENTER, 10);
        //myGLCD.print("CM", LEFT, 20);
        //myGLCD.update();
}
```

第四章實務成果

4.1 啟動自走車放在路徑上時會先偵測路徑

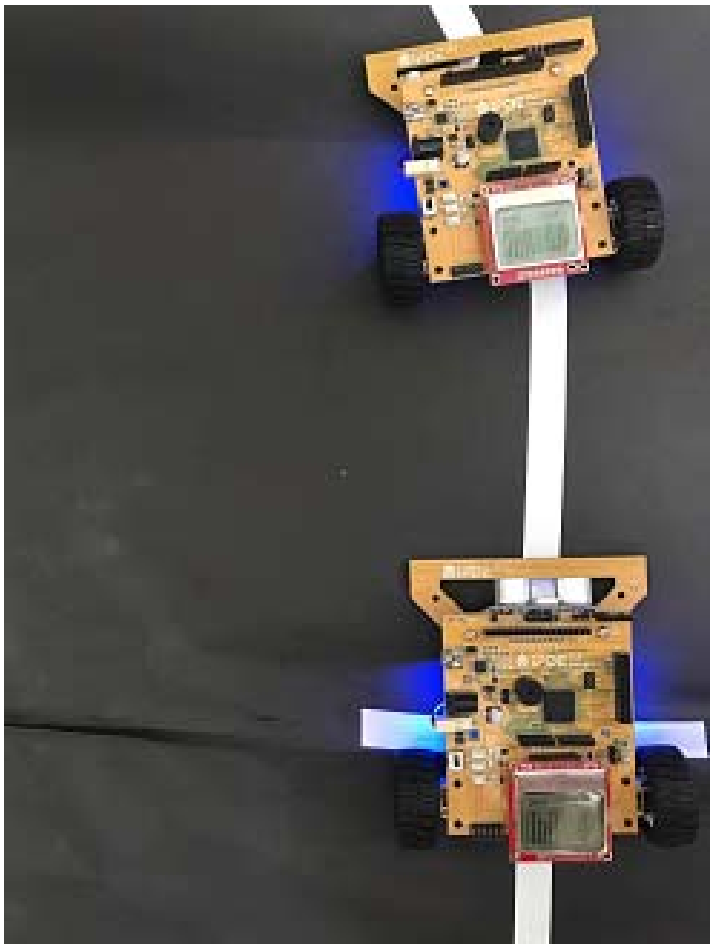


圖（17）自走車開機畫面

自走車開機時會紅外線感應會先偵測軌跡，才會執行下一個動作。

4.2 將有超音波感測的放在後車

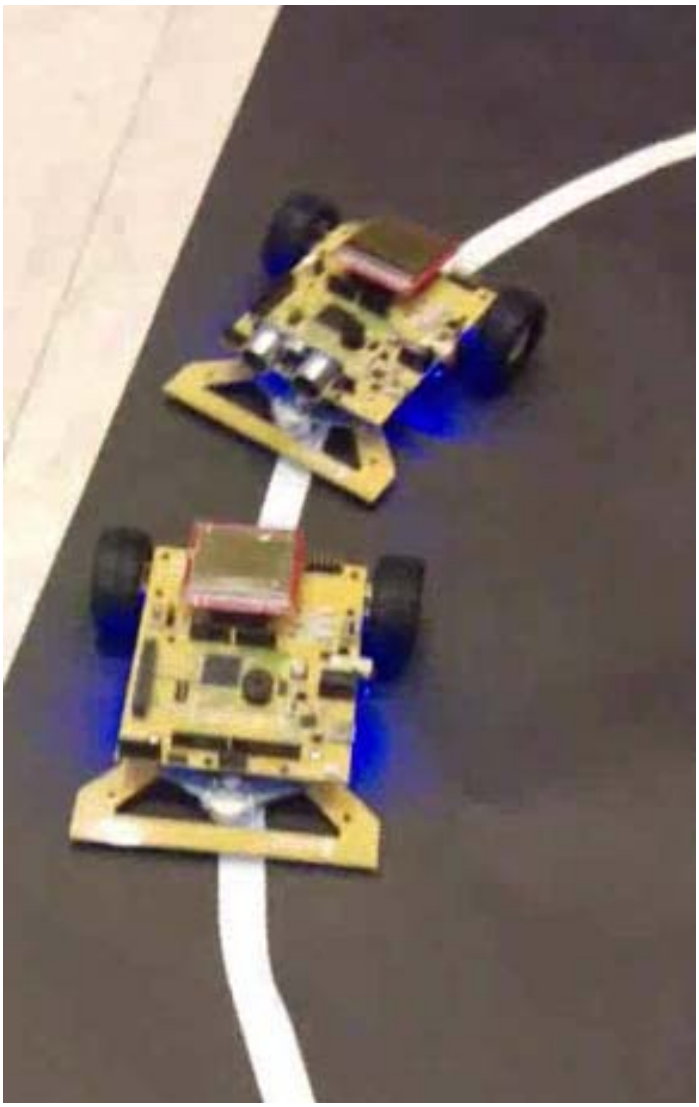
再依序啟動



圖(18)依序啟動畫面

在實作的過程中，我們不斷的測試怎樣才能走很多的路徑，還有測試前後車的車速，候車要跑得比較快才能追上前車，才啟動感測器保持行車距離。

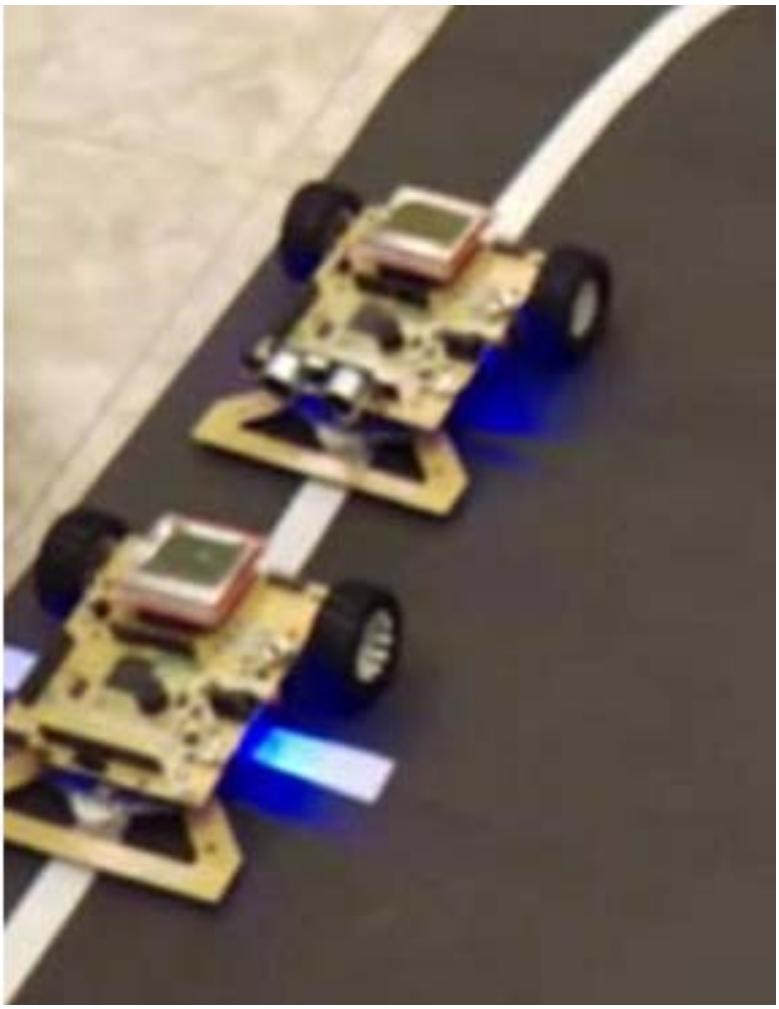
4.3 兩車相遇時與前車會保持距離



圖(19)兩車相遇

當後車追上前車時因後車有超音波感測器，所以會與前方保持距離不會撞再一起，如上圖所示當後車追上來時會保持兩車間的距離。

4.4 完成動作後動作停止



圖(20)動作停止

到達目的地時會有一個指令讓前車停止，而前車停止時因後車的超音波感測器啟動並不會撞到前車，也會跟著停止。

第五章結論

未來研究方向

在做專題的過程中，我們泡到很多問題，大部分都卡在寫程式上面，解決的方式就是把一整個該完成的結果拆成小部分小部份去理解完成後再做整合，這樣做起來比較有理解性。

在未來生活中我們可以真的把它安裝在更大台的車上去做測試，或許與可以直接安裝在機車上去做測試，在我們的生活上越來越依賴科技，我們就更應該好好去研究如何實際運用在生活上。

或許未來我們可以模擬在各種狀況或是環境去發展出特別的功能，延伸到救災需要能前進的功能，車子能減少碰撞也能減少醫療資源或社會資源的消耗，這些都是過去不曾讓人想像的功能，在過去認為不可能的科技，現在不一定不能實現。

參考文獻

1.iPOE-A1 簡介

http://jiader.blogspot.tw/2014/10/1-ipoe-a1-1-1-ipoeintelligent-public.html

2.紅外線循跡感測器 CNB10010

http://epaper.gotop.com.tw/pdf/AEH003200.pdf

3.超音波感測器

http://coopermaa2nd.blogspot.tw/2012/09/hc-sr04.html

4.元件簡介

http://jiader.blogspot.tw/

5.函式庫簡介

http://jiader.blogspot.tw/2014/10/ipoe-a1.html

6.i POE　A1 輪型機器人互動設計(書籍)