建國科技大學

工業工程與管理系

102 學年度專題製作

運用 ARDUINO 專案改善交通工具的行進方式

The Applications of Arduino Projects on Improvements to Vehicle Travel Methods

IE100 日四 B –02

指導老師

陳錫川

製作學生

100405084 高楷翔

100405120 陳倍泓

100405100 陳超倫

100405096 楊崇佑

100405099 溫志龍

中　華　民　國　１０３　年　5　月

建 國 科 技 大 學　工 業 工 程 與 管 理 系

專題製作口試委員會審定書

本系 102 學年度學生：高楷翔、陳倍泓、陳超倫、楊崇佑、溫志龍　等員

所提專題題目：運用 ARDUINO 專案改善交通工具的行進方式

合於大學生畢業資格水準，業經本委員會評審認可。

口試委員：楊哲維　陳錫川

吳章保

指導老師：陳錫川

系 主 任：[illegible]

中 華 民 國　103 年 5 月 29 日

中文摘要

工業革命以來，工廠的人力被機器取代，讓人類從「人力時代」來到「機器時代」，大幅提升了工作效率以及減少人力成本。目前工廠常見的搬運設備有天車、起重機、堆高機、無人搬運車以及手推車，在室內運作將物品做一定的轉移放置或是移動位置。此專題利用 Arduino 來控制兩輪馬達驅動的車輛，先編寫程式碼之後將其寫入 UNO328 控制板中，此專題分成幾種模式來控制兩輪馬達的交通工具，包含遙控的部分，藍牙遙控、紅外線遙控，自走部分包含有超音波感測、循線感測。本專題在四種模式各提出改善的方案，為軟硬共同設計的實作專題，藉由自走車改善的方法，可以改善傳統運輸設備的作業方法，提升工作效率。

關鍵字：Arduino、紅外線遙控、藍牙遙控、超音波避障、紅外線循線。

Abstract

Since the Industrial Revolution, factory machines have replaced human being, letting mankind go from the "human era" to "machine age", and significantly enhance the work efficiency and reduce labor costs. At present, factories have variety transportation handling equipments include crane, forklifts, AGV, trolleys, etc to replace or to move indoor item position. This student project used Arduino to control the two-wheeled-driven motor vehicle. After finished program code, the program code was written into UNO328 control board. The control method in this student project was divided into remote mode and self-propelled mode. The remote mode of two-wheeled-driven motor vehicle proposed in this project included the Bluetooth remote control and infrared remote control. The self-propelled mode in this project contained ultrasonic obstacle avoidance, and autonomous line-tracking. This project proposed improvement plan for each of the four models and was implemented for hardware and software. The proposed improvement plan can be applied to traditional transport equipment, and upgrade the work efficiency.

Keywords: Arduino, infrared remote control, Bluetooth remote control, ultrasound obstacle avoidance, autonomous line-tracking.

目錄

中文摘要 i
Abstract ii
目錄 iii
圖目錄 i
表目錄 i
第一章 緒論 1
1.1 研究動機 1
1.2 研究目的 1
1.3 研究架構 1
第二章 文獻探討 3
2.1 單晶片微電腦 3
2.2 自走車種類 4
2.3 感測器 4
2.4 通訊方式 6
第三章 智能小車模組介紹 8
3.1 驅動裝置 8
3.2 邏輯運算單元 13
3.3 智能小車所需的感測裝置 14
3.4 行進模式 15
第四章 紅外線控制模式 17
4.1 所需零件 17
4.2 運作原理 18
4.3 改善模式 18
4.4 結論 20
第五章 藍牙控制模式 21
5.1 所需零件 21
5.2 運作原理 22
5.3 改善模式 24
5.4 結論 27
第六章 紅外線循線模式 28
6.1 所需零件 28
6.2 運作原理 29
6.3 改善模式 32
6.4 結論 32
第七章 超音波避障模式 33
7.1 所需零件 33

7.2 運作原理.....34
7.3 改善模式.....34
7.4 結論.....38
第八章 結論 39
參考文獻 41
附錄程式碼 42

圖目錄

圖 1.1 研究架構 2
圖 2.3 感測器的組成 5
圖 3.1 L298N 電機驅動版接線 8
圖 3.2 Arduino UNO 328 13
圖 3.3 Arduino 擴充版 14
圖 4.1 紅外線接收發射模組 17
圖 4.2 紅外線接收器接線圖 17
圖 4.3 Arduino 內建紅外線偵測範例 19
圖 4.4 按鍵代碼 19
圖 5.1 HC-06 藍牙接收器 21
圖 5.2 藍牙接線圖 21
圖 5.3 藍牙簡易流程 22
圖 5.4 Appinventor 藍牙遙控程式碼 23
圖 5.5 原本有瑕疵的 APP 24
圖 5.6 舊使用者介面 25
圖 5.7 修改加上停止程式的 APP 25
圖 5.8 修改後新使用者介面 26
圖 6.1 紅外線循線器 28
圖 6.2 紅外線循線器接線圖 28
圖 6.3 偵測到黑線時 29
圖 6.4 未偵測道黑線時 29
圖 6.5 出軌示意圖 32
圖 6.6 修正後示意圖 32
圖 7.1 超音波感測器正面圖 33
圖 7.2 原本的障礙物閃避流程 34
圖 7.3 定義超音波腳位 36
圖 7.4 超音波避障程式碼 36
圖 7.5 更改後的超音波避障流程 37

表目錄

表 3.1 馬達前進 9
表 3.2 馬達後退 9
表 3.3 馬達左轉(單輪) 10
表 3.4 馬達右轉(單輪) 10
表 3.5 馬達往左前方走 11
表 3.6 馬達往右前方走 11
表 3.7 馬達右轉(雙輪) 12
表 3.8 馬達左轉(雙輪) 12
表 3.9 馬達停止 13
表 3.1 紅外線遙控器對應碼 15
表 4.1 紅外線遙控器功能表 18
表 5.1 藍牙接線圖說明 22
表 6.1 邏輯判定 30
表 7.1 超音波感測器 HC-SR04 的腳位描述 33

第一章 緒論

1.1 研究動機

在工廠的運作中，搬運是不可或缺的步驟，搬運的器具包涵機械手臂、輸送帶、無人搬運車、手推車等等。隨著時代的變遷，科技進步的關係，即時控制軟體的發展和導向技術的應用發展出許多不同的科技產品，例如：利用無人搬運車(Automatic Guided Vehicle, AGV)來代替傳統勞力以減少搬運的次數和時間，因為人力在搬運時效率低，基於這個想法，本專題規劃從小兩輪馬達驅動車模型改善起步，透過智能小車的搬運效率及方式改善，希望能應用在大型自走車的搬運效率提昇，藉此提昇搬運的效率與安全性。

1.2 研究目的

本專題是以自走車的避障和尋線進行探討，控制方式包括：紅外線、藍牙，行走方式包括：超音波避障、自動尋線為幾種主要方式。

本專題的主要目的可以分為以下三點：

1.藉由系統設計更深入的擴大自走車的功能。

2.希望利用自走車改善交通工具的行進方式。

3.提升工作效率。

1.3 研究架構

本專題在組裝完智能小車後，依照不同的運作模式，分別尋找可改善的方案，改善後並分析其效益，彙整後再研究如何利用。

整個專題的研究架構表示如下圖：

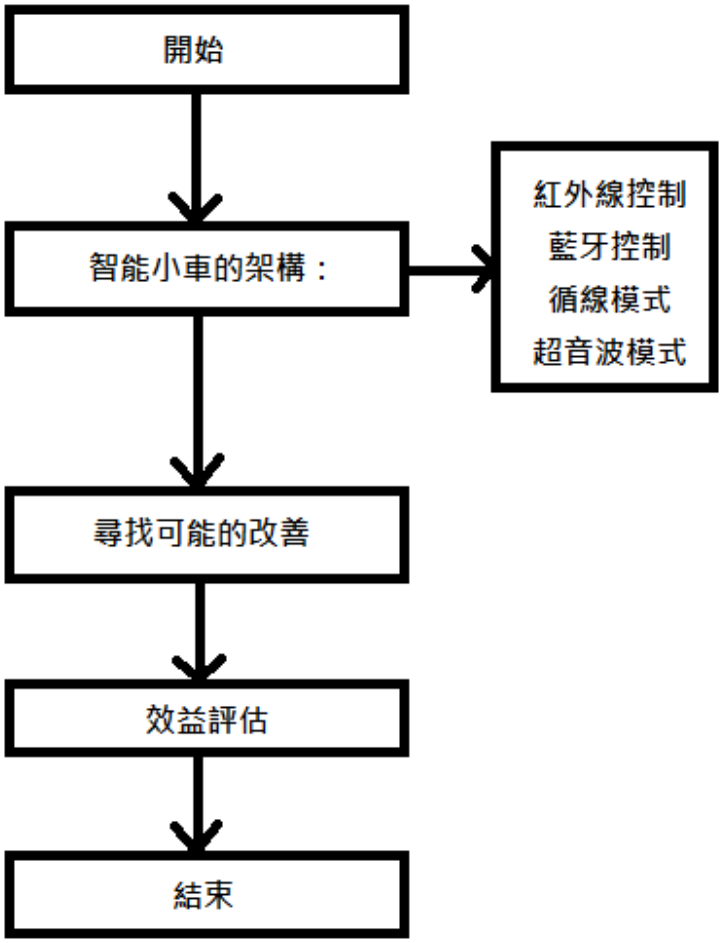


圖 1.1 研究架構

第二章 文獻探討

2.1 單晶片微電腦

全稱單晶片微電腦（英語：Single-Chip Microcomputer），又稱微控制器（Microcontroller），是把中央處理器、記憶體、定時/計數器（Timer/Counter）、各種輸入輸出介面等都整合在一塊積體電路晶片上的微型電腦。與應用在個人電腦中的通用型微處理器相比，它更強調自供應（不用外接硬體）和節約成本。它的最大優點是體積小，可放在儀錶內部，但儲存量小，輸入輸出介面簡單，功能較低。

常見微電腦所使用的單晶片有： Atmel 的 51 系列、MSP430 系列等等

1. Atmel AVR 系列是一種基於改進的哈佛結構、8 位元～32 位元精簡指令集（Reduced Instruction Set Computing，RISC）的微控制器，Arduino 也是 AVR 的其中一種，Arduino UNO R3 所使用的晶片是 Atmega16U2 晶片是八位元處理器。AVR 系列是首次採用快閃記憶體（Flash Memory）作為資料儲存介質的單晶片微控制器之一，同時代的其它微控制器多採用一次寫入唯獨記憶體、EPROM 或是 EEPROM。
2. 8051 8-bit 單晶片是同步式的順序邏輯系統，整個系統的工作完全是依賴系統內部的時脈信號，用以來產生各種動作周期及同步信號。在 8051 單片機中已內建時鐘產生器，在使用時只需接上石英晶體諧振器（或其它振蕩子）及電容，就可以讓系統產生正確的時鐘信號。
3. MSP430 是一系列由德州儀器生產的 16 位微控制器及其品牌，超低功耗 16 位精簡指令集混合信號處理器。頻率有 8/16/25MHz 可供選擇。
4. Raspberry Pi 是基於 LINUX 的單晶片微電腦，只有一張信用卡大小，Raspberry Pi 的核心處理器，是博通(broadcom)所生產的 32 位元的 BCM2835 的系統單晶片(SoC)多媒體處理器。意味著絕大部分的系統原件，包括中央處理器、圖形處理器以及音源等通訊硬體都裝在一個單一原件中，除此之外，Raspberry Pi 還使用 ARM 當作指令集架構，Raspberry Pi 能夠使用內建的

imcro-USB 阜所提供的 5V/1A 的電流運作，就是因為 ARM 架構的 BCM2835 晶片。

2.2 自走車種類

現今的科技日新月異，人類的工作也漸漸得由機器代勞，自走車就是自動控制方面的一項產物，它是利用利用單晶片微電腦的的程式設計來控制馬達和感測器，並互相配合，製作出來的。

1. 尋線自走車：利用單晶片微電腦的程式控制，搭配安裝在車體底部的三個紅外線循線器，藉由循線器的紅外線發送和接收所得到的數據判斷為 0、來判斷黑色與白色，以達到循著著黑線前進的目地。工廠上常見的有無人搬運車：簡稱 AGV（Automated Guided Vehicle），它是一種獨立作業的搬運系統，靠著車上的感測器引導車子依循電線或圖漆前進，來達成無人操控的搬運方式。
2. 超音波避障自走車：利用單晶片微電腦的程式控制，搭配安裝在車體前方的超音波感測器來判斷前方是否有障礙物。安裝的方式多樣化，可一次在車體前方安裝多個超音波感測器來增加判定的範圍，或者在車體前方安裝舵機，在將超音波感測器安裝在舵機上，在要藉由加寫程式控制舵機，也可以多方位偵測前方障礙物。
3. 聲控自走車：一般聲控方式為使用語音辨識方法，難度較高，而且必須發出一定音量的聲音，如果在嘲雜的環境，語音辨識恐怕會失去作用。
4. 磁帶感應自走車：這是可以用在特殊環境下的高感度磁氣 Sensor (magnetic sensor)，可以用在自動倉儲定位、停車場升降塔定位、無人搬運車 AGV 的導引與移動載台的行走或定位檢測。

2.3 感測器

感測器是能接收規定的信號或刺激，並按照一定規律將待測物理量或化學量轉換成可用輸出信號的器件或裝置，通常有敏感元件和轉換元件組成，如圖 2.3 所示。

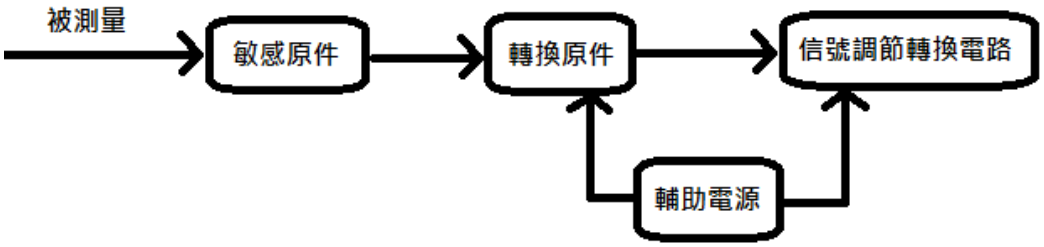


圖 2.3 感測器的組成

一個感測器的輸入對輸出的影響被稱為感測係數或靈敏度（sensitivity）。假設，一個溫度計，每當溫度上升 1℃ 時，水柱上升 1cm，則這個溫度計的感測係數為 1 cm/℃。以 Arduino 為例，int analogRead(pin)類比輸入函數，pin 表示為 0～5（Arduino Diecimila 為 0～5，Arduino nano 為 0～7）。比如可以讀模擬感測器（10 位 AD，0～5V 表示為 0～1023）。

常見的感測器：

1. 功能分類：

   電阻式感測器、電容式感測器、電感式感測器、壓電式感測器、熱電式感測器

2. 技術分類：

   超音波感測器、溫度感測器、濕度感測器、氣體感測器、壓力感測器

3. 電子式感測器：

   IR 紅外線近接/測距、循線循跡 Sensor、超音波距離檢測、雷射區域距離測量儀

2.4 通訊方式

通訊是發送者透過某種媒體以某種格式來傳遞訊息到收信者以達到目的。在古代，人們透過驛站、飛鴿傳書、符號、語言、眼神等方式進行訊息傳遞。到了今天，隨著科技的迅速發展，通訊基本完全利用有線或無線電完成，陸續出現了固定電話、無線電話、手提電話、網際網路甚至視訊電話等方式。通訊技術拉近了人與人之間的距離，提高了通訊的效率，深刻的改變了人類的通訊。以下將探討：WI-FI、紅外線、近場通訊、3G/2.5G/2G、ZigBee。

1. Wi-Fi 是 Wi-Fi 聯盟製造商的商標可做為產品的品牌認證，是建立於 IEEE 802.11 標準的無線區域網路設備。基於兩套系統的密切相關，也常有人把 Wi-Fi 當做 IEEE 802.11 標準的同義術語。
2. 紅外通訊技術不需要實體連線，簡單易用且實現成本較低，因而廣泛應用於小型移動設備互換數據和電器設備的控制。由於紅外線的直射特性，紅外通訊技術不適合傳輸障礙較多的地方，這種場合下一般選用無線電通訊技術或藍牙技術。紅外通訊技術多數情況下傳輸距離短、傳輸速率不高。
3. 近場通訊（英語：Near Field Communication，NFC），又稱近距離無線通訊，是一種短距離的高頻無線通訊技術，允許電子設備之間進行非接觸式點對點資料傳輸，在十公分（3.9 英吋）內交換資料。
4. 3G 是「第三代行動通訊技術」（英語：3rd-Generation）的縮寫，也就是 IMT-2000（International Mobile Telecommunications-2000），是指支援高速資料傳輸的蜂窩移動通訊技術。3G 服務能夠同時傳送聲音（通話）及資訊（電子郵件、即時通訊等）。3G 的代表特徵是提供高速資料業務，速率一般在幾百 kbps 以上。
5. 通用封包無線服務技術（英語：General Packet Radio Service，縮寫：GPRS）是 GSM 行動電話用戶可用的一種移動數據業務。 它經常被描述成「2.5G」，也就是說這項技術位於第二代（2G）和第三代（3G）移動通訊技術之間。它通過利用 GSM 網路中未使用的 TDMA 通道，提供中速的數據傳遞。最初有人想通過擴展 GPRS 來覆蓋其他標準，只是這些網路都正在轉而使用 GSM 標準，這樣 GSM 就成了 GPRS 唯一能夠使用的網路。
6. 全球行動通訊系統（Global System for Mobile Communications），即 GSM，是當

前應用最為廣泛的行動電話標準。全球超過 200 個國家和地區超過 10 億人正在使用 GSM 電話。GSM 標準的廣泛使用使得在行動電話運營商之間簽署「漫遊協定」後用戶的國際漫遊變得很平常。GSM 較之它以前的標準最大的不同是他的信令和語音通道都是數位的，因此 GSM 被看作是第二代（2G）行動電話系統。GSM 標準當前由 3GPP 組織負責制定和維護。

7. ZigBee（也稱紫蜂）是一種低速短距離傳輸的無線網路協定，底層是採用IEEE 802.15.4標準規範的媒體存取層與實體層。主要特色有低速、低耗電、低成本、支援大量網路節點、支援多種網路拓撲、低複雜度、快速、可靠、安全。

第三章 智能小車模組介紹

3.1 驅動裝置

1. 電機驅動板：L298N

L298N 是個價錢便宜且方便的電機驅動版（圖 3.1），價格大概在 120 元左右，它可以驅動兩個直流馬達或是一個步進馬達，控制訊號可以用 Arduino 或是其它微控制器輸入。這塊板子上帶有一個 5V 穩壓器，使用者可選擇是否打開 5V 輸出介面，以提供 5V 電源給 Arduino 或其它微控制器。

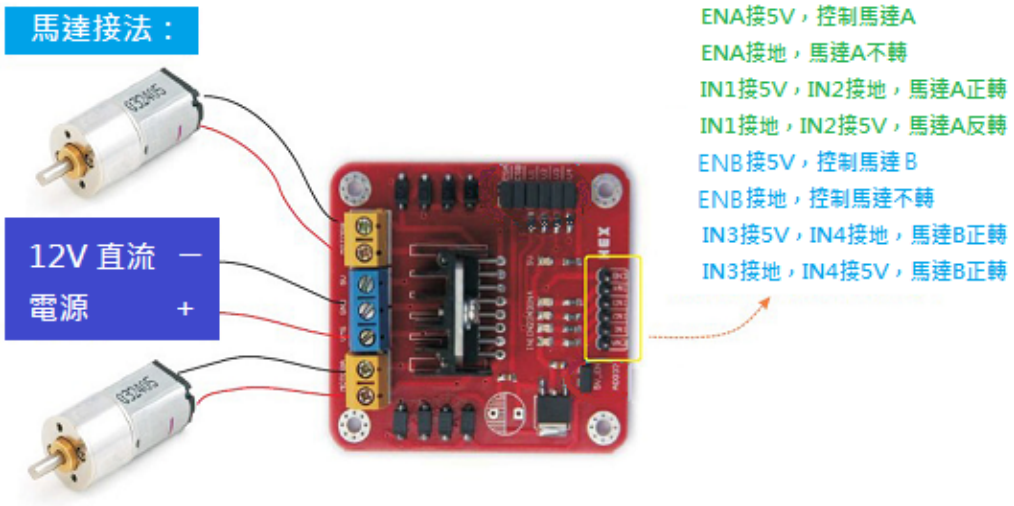


圖 3.1 L298N 電機驅動版接線

2. 馬達驅動裝置

本專題使用的是直流馬達，直流馬達是利用電壓大小控制轉速與轉向，因此在控制上較容易，但是直流馬達不適合於高溫或是易燃的工作環境。一般直流馬達都是有碳刷做為電流變換器，所以直流馬達需要定期清理摩擦產生的髒污以利於馬達正常運作。

當數位訊號右馬達 1、2 分別是 HIGH、LOW；左馬達 1、2 是 HIGH、LOW；則馬達前進，前進的時間為 0.1 秒。在 Arduino 的開發環境中，一秒被分為 1000 等份，所以 1 秒=1000 毫秒；0.1 秒=100 毫秒；0.01 秒=10 毫秒。

表 3.1 馬達前進

void advance(int a)//前進	digitalWrite(MotorRight1, HIGH) digitalWrite(MotorRight2, LOW) digitalWrite(MotorLeft1, HIGH) digitalWrite(MotorLeft2, LOW) delay(100)

當數位訊號右馬達 1、2 分別是 LOW、HIGH；左馬達 1、2 是 LOW、HIGH；延遲 0.1 秒時，則馬達後退。

表 3.2 馬達後退

void back(int g) //後退	digitalWrite(MotorRight1, LOW) digitalWrite(MotorRight2, HIGH) digitalWrite(MotorLeft1,LOW) digitalWrite(MotorLeft2,HIGH) delay(100)

當數位訊號右馬達 1、2 分別是 HIGH、LOW；左馬達 1、2 是 LOW、LOW；延遲 0.1 秒時，則馬達單輪左轉。

表 3.3 馬達左轉(單輪)

void left(int c) //左轉(單輪)	digitalWrite(MotorRight1, HIGH) digitalWrite(MotorRight2, LOW) digitalWrite(MotorLeft1,LOW) digitalWrite(MotorLeft2,LOW) delay(100)

當數位訊號左馬達 1、2 分別是 HIGH、LOW；右馬達 1、2 是 LOW、LOW 時，則馬達單輪右轉。

表 3.4 馬達右轉(單輪)

void right(int b) //右轉(單輪)	digitalWrite(Motor Left1, HIGH) digitalWrite(Motor Left2,LOW) digitalWrite(Motor Right1,LOW) digitalWrite(Motor Right2, LOW) delay(100)

當數位訊號右馬達 1、2 分別是 HIGH、LOW；類比訊號左馬達 1、2 轉速是 0、0；延遲 0.01 秒時，則馬達往前方走。

表 3.5 馬達往左前方走

void Fleft(int c) //往左前方走	digitalWrite(MotorRight1, HIGH) digitalWrite(MotorRight2, LOW) analogWrite(MotorLeft1,0); analogWrite(MotorLeft2,0) delay(10)

當類比訊號右馬達 1、2 轉速是 0、0；數位訊號左馬達 1、2 分別是 LOW、HIGH；延遲 0.01 秒時，則往右前方前進。

表 3.6 馬達往右前方走

void Fright(int b) //往右前方走	analogWrite(MotorRight1,0) analogWrite(MotorRight2,0) digitalWrite(MotorLeft1,LOW) digitalWrite(MotorLeft2,HIGH) delay(10)

當數位訊號右馬達 1、2 轉速是 LOW、HIGH；數位訊號左馬達 1、2 分別是 HIGH、LOW；延遲 0.01 秒時，則馬達右轉(雙輪)。

表 3.7 馬達右轉(雙輪)

void turnR(int d)//右轉(雙輪)	digitalWrite(MotorRight1,LOW) digitalWrite(MotorRight2,HIGH) digitalWrite(MotorLeft1,HIGH) digitalWrite(MotorLeft2,LOW) delay(10)

當數位訊號右馬達 1、2 轉速是 HIGH、LOW；數位訊號左馬達 1、2 分別是 LOW、HIGH 時，則馬達左轉(雙輪)。

表 3.8 馬達左轉(雙輪)

void turnL(int e)//左轉(雙輪)	digitalWrite(MotorRight1,HIGH) digitalWrite(MotorRight2,LOW) digitalWrite(MotorLeft1,LOW) digitalWrite(MotorLeft2,HIGH)

當數位訊號右馬達 1、2 轉速是 LOW、LOW；類比訊號左馬達 1、2 分別是 LOW、LOW 時，則馬達停止。

表 3.9 馬達停止

void stopp(int f) //停止	digitalWrite(MotorRight1,LOW) digitalWrite(MotorRight2,LOW) digitalWrite(MotorLeft1,LOW) digitalWrite(MotorLeft2,LOW) delay(100)

3.2 邏輯運算單元

1. Arduino UNO 328 控制板

Arduino UNO(圖 3.2)是 Arduino USB 接口系列的最新版本，作為 Arduino 平台的參考標準模板。 UNO 的處理器核心是 ATmega328，同時具有 14 路數字輸入/輸出口（其中 6 路可作為 PWM 輸出），6 路模擬輸入，一個 16MHz 晶體振盪器，一個 USB 口，一個電源插座，一個 ICSP header 和一個複位按鈕。

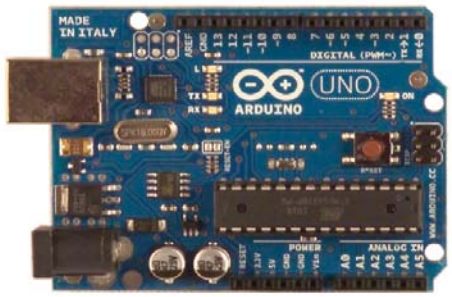


圖 3.2 Arduino UNO 328

2. Arduino 感測器擴充版

Arduino Sensor Shield V5.0 感測器擴展板(圖 3.3),在保留 V4.0 版本優點的基礎上，仍才採用疊層設計，特別設 IIC 介面、 32 路伺服機控制器介面,藍牙模組介面、SD 卡模組通信介面、APC220 無線射頻模組通信介面、超音波感測器介面、LCD 液晶串行與平行介面，獨立擴出更加易用方便。

對於 Arduino 初學者來說，不必為繁瑣複雜電路連線而頭疼了，感測器擴展板將電路簡化，能夠更 容易地將常用感測器連接起來，一款感測器僅需要一種通用 3P 感測器連接線（不分數位連接線與類比連接線），完成電路連接後，編寫相應的 Arduino 程式 傳輸到 Arduino UNO 328 中讀取感測器資料、或者接收無線模組回傳資料，經過運算處理，最後完成作品。

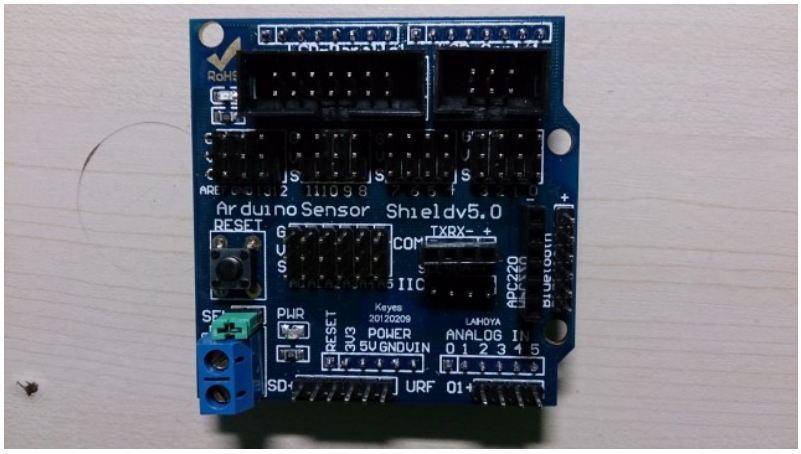


圖 3.3 Arduino 擴充版

3.3 智能小車所需的感測裝置

西元 1666 年牛頓發現光譜並測量出(400nm～700nm)是可見光的波長。1800 年 4 月 24 日英國倫敦皇家學會（ROYAL SOCIETY）的威廉·赫歇爾發表太陽光在可見光譜的紅光之外還有一種不可見的延伸光譜，具有熱效應。

自然界中一切溫度高於絕對零度(- 273℃)的物體都不斷的發射著紅外線，這種現象稱為熱輻射。紅外線是一種人眼看不見的光波，它是由物質內部的分子、

原子運動所產生的電磁輻射，是電磁頻譜的一部分。

光學感測器是利用光敏元件將光訊號轉換為電訊號的感測器，感應波長在可見光波長附近，如紅外線波長和紫外線波長。光學感測器不只是應用於光的測量，更常用於作為探測元件，組成其它類型的感測器，對非電量（如溫度等）進行檢測，只要將這些非電量轉換為光訊號的變化便可實現對非電量的檢測。

1. 紅外線接收器：利用接收紅外線控制器發出的訊號，將光訊號轉為電子訊號，讓自走車判斷電子訊號依指令行動或切換模式。
2. 紅外線循線器：利用反射遮蔽原理，發射器打出紅外線後，經過反射，如果是黑色的話，紅外線會被吸收，感測器輸出的電子訊號為 1，反之白色為 0，自走車藉此訊號來判斷黑線來前進。
3. 超音波感測器：利用反射原理，自走車行徑的途中，持續的輸出超音波，當有障礙物靠近時，立即停止，之後利用回傳的超音波判斷距離，選擇最佳的路線繼續前進。

3.4 行進模式

本專題的行進模式可分為遙控與自走兩種。

表 3.1 紅外線遙控器對應碼

	對應紅外線內碼

紅外線遙控模式	long IRfront= 0x00FF629D; //前進
	long IRback=0x00FFA857; //後退
	long IRturnright=0x00FFC23D; //右轉
	long IRturnleft= 0x00FF22DD; //左轉
	long IRstop=0x00FF02FD; //停止
	long IRcny70=0x00FF6897; //CNY70 自走模式
	long IRAutorun=0x00FF9867; //超音波自走模式

1. 尋線模式

利用 TCR5000 的紅外線偵測模組，運用紅外線的反射遮蔽原理偵測一開始設定好的黑線路徑，之後連接 Arduino UNO 328 控制板，判斷並選擇要執行的指令，再連接 L289N 電機驅動板來驅使馬達動作，便可循著黑線移動，到達操作者設定的目的地。

2. 超音波避障

本智能小車利用超音波模組來偵測外在環境刺激，搭配 Arduino 擴充版 V5 連接 Arduino UNO 328 控制板，判斷並選擇要的指令，在連接 L298N 電機驅動版來驅使馬達動作，便可達到偵測外在物體，以達到自動避障的效果。

3. 手動模式

手動模式是智能小車的基本功能，只要將按鍵代碼對應程式代碼，便能控制小車的動作，「一個口令，一個動作」地命令智能小車，智能小車並不能自行判斷動作，只能全部聽從操作者的指令動作。

第四章 紅外線控制模式

4.1 所需零件

紅外線發射模組(圖 4.1)是一種藉由紅外線 LED 燈和編碼 IC 及石英震盪器、電容等零件…組成的電路，經由編碼 IC 發射出不同調變波的紅外光發送出紅外線訊號。紅外線接收模組則是利用紅外線電晶體及電容等零件和解碼 IC 組成，將接收到的紅外線訊號解碼成對應的編碼。

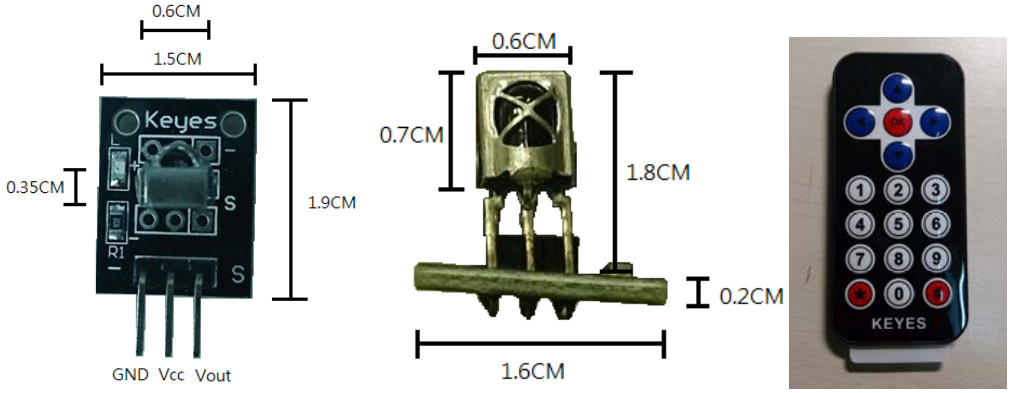


圖 4.1 紅外線接收發射模組

紅外線接收器有三根針角，負極、VCC 以及 VORT，GND 是接地線、VCC 是電流來源為 5 伏特、VORT 是紅外線接收的訊號線，本專題的紅外線接收器，接在擴充板上的 2 號位置，分別是 G 接 GND、V 接 VCC 及 S 接 VORT，接線的方式如圖 4.2 所示。

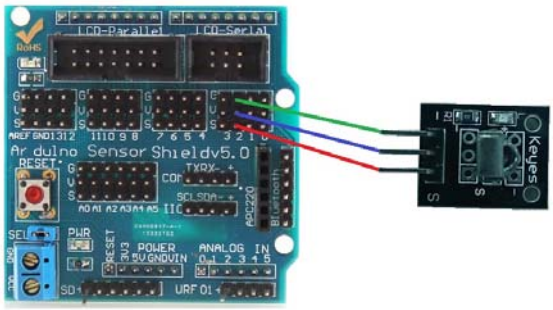


圖 4.2 紅外線接收器接線圖

4.2 運作原理

紅外線控制分為紅外線遙控器（發射器）和紅外線感測器（接收器）兩部分，由紅外線控制負責發射訊號而感測器則負責接收，來達到控制智能小車被動的前進後退以及左右旋轉，還有切換模式如 1 是自動尋線模式、2 是藍牙控制模式，如表 4.1 所示。

表 4.1 紅外線遙控器功能表

功能	按鍵
前進	上
後退	下
左轉	左
右轉	右
停止	OK
自動尋線模式	按鍵 1
超音波避障模式	按鍵 2

4.3 改善模式

智能小車有內建的遙控代碼指令，經測試之後發現無法控制智能小車，可能的原因包括有擴充版接角接錯、紅外線接收器的接腳接錯，或者程式寫錯都是有可能的原因，後來經過數次的檢查後發現，紅外線控制器的按鍵跟 Arduino 的內建程式碼沒有對應，所以導致按鍵都沒有反應。

所以首先把紅外線接收器單獨接在 Arduino 板子上，開啟 Arduino 的內建程式 IRrecvDemo（圖 4.3），該程式可用來偵測紅外線控制器的按鍵代碼。

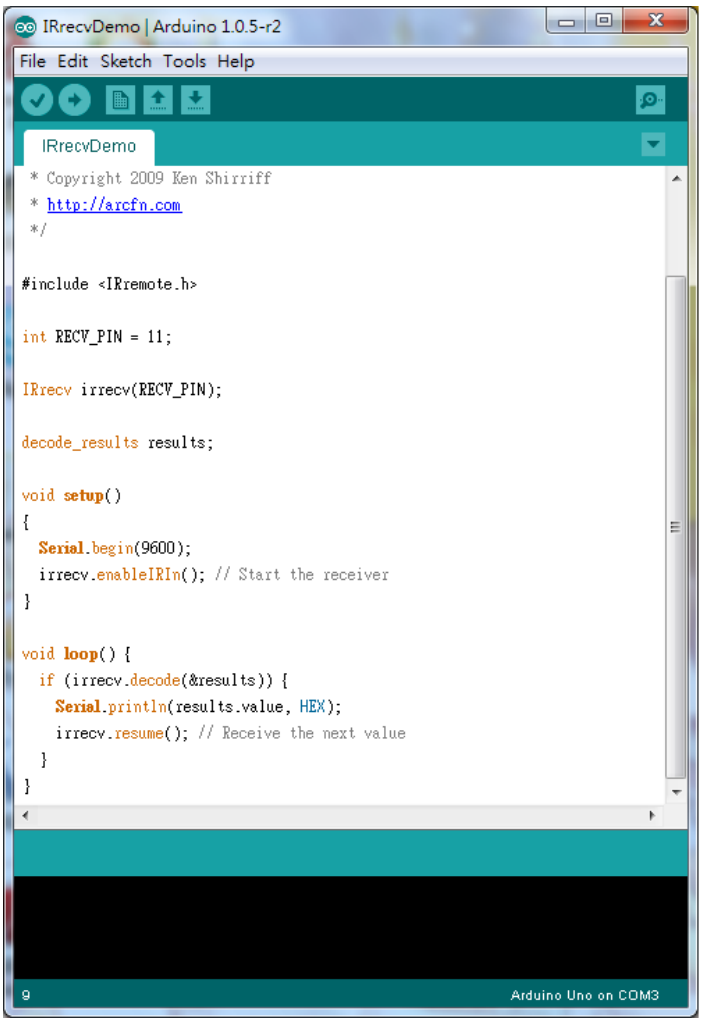


圖 4.3 Arduino 內建紅外線偵測範例

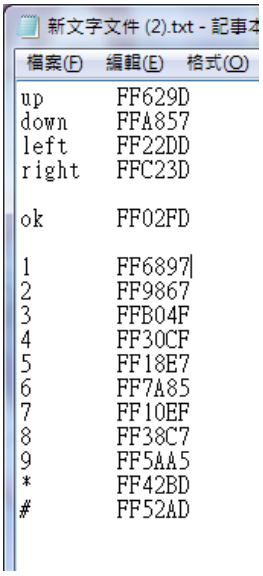
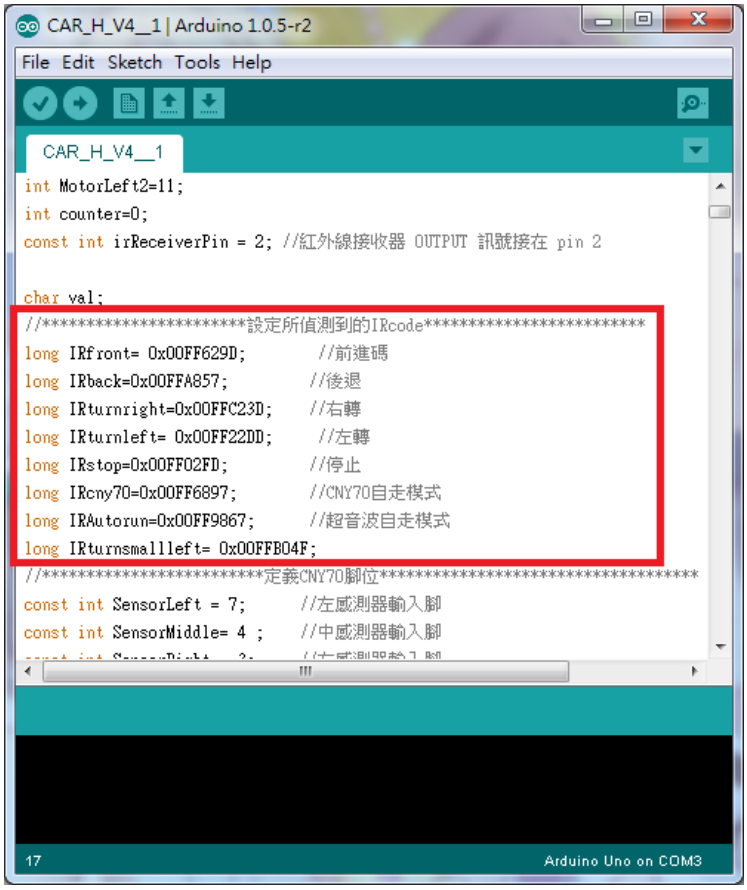


圖 4.4 按鍵代碼

最後整理出來的按鍵代碼如圖 4.4 所示



最後只要將其輸入紅外線控制器的程式碼中，即可達到紅外線控制的目的。

4.4 結論

本專題所用的紅外線控制只是控制智能小車方法裡面其中一種，紅外線控制對本專題來說並不實用，紅外線控制的問題也不少，例如：距離短、只能單一控制與無法穿透障礙物，如果只是一再的增強紅外線的強度或降低干擾，這樣無法解決問題，只是治標不治本。應該用現在方便且快速的藍牙來控制智能小車，本專題對上述改善的缺點提出了一些想法，這些想法除了可以提升智能小車的性能之外，也希望智能小車在控制方面能有多樣化方式。

第五章 藍牙控制模式

5.1 所需零件

藍牙操控分別是由二個端點組成，主端(Server)與從端(Client)，由主端負責控制從端，HC-06 是一個藍牙模組，屬於從端，它的偵測距離在空曠的地方有效距離 10 公尺，最大距離可超過十公尺，電源電壓：3.6V~6V，未配對時電流約 30mA，配對後約 8mA，接口電平 3.3V，可以直接連接各種單片機（51，AVR，PIC，ARM，MSP430 等），5V 單片機最好串聯一個 1K 電阻再與模塊直接連接，採用 CSR 主流藍牙晶片 ，藍牙 V2.0 協議標準。

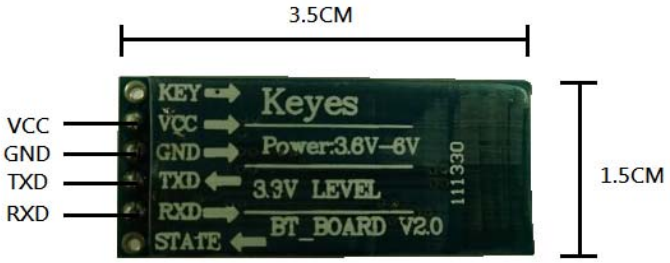


圖 5.1 HC-06 藍牙接收器

圖 5.2 是本專題的藍牙接線圖，接線圖的四條線別為 VCC(正極)、GND(負極)、TXD(發送端)、RXD(接收端)，詳細功能如表 5.1 所示。

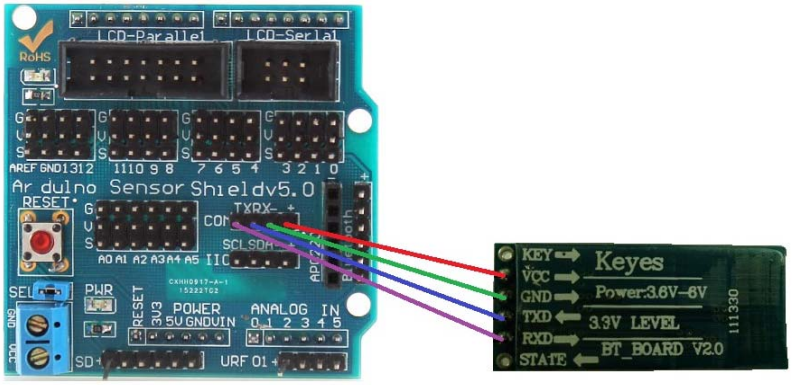


圖 5.2 藍牙接線圖

表 5.1 藍牙接線圖說明

電流流通	功能
VCC(正極)	接電源
GND(負極)	接地
資訊、訊號傳輸	功能
TXD(發送端)	發送端若要傳輸必須與接收端做連結
RXD(接收端)	接收端若要傳輸必須與發送端做連結

5.2 運作原理

連接藍牙 HC-06 的零件 Server 端，在專題規劃是以手機來當藍牙的 Server 端，來發出訊號(圖 5.3)，但是如果要使用手機的藍牙功能必須在手機中設計相關的應用程式，本專題採用 Android 的作業系統。

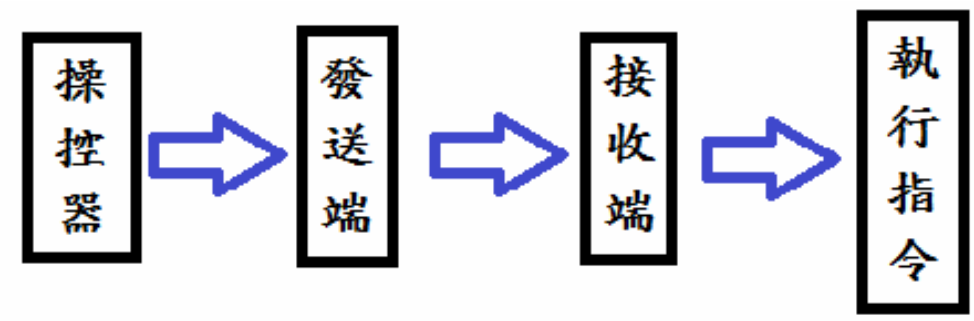


圖 5.3 藍牙簡易流程

Android 應用開發者(Appinventor)是起先由 Google 提供的應用軟體，現在由麻省理工學院所維護及運營。它可以讓不熟悉程式設計的人來創造屬於 Android 作業系統的應用軟體。它使用圖形化界面，這樣使用者拖放圖形對象來創造一個運行在安卓系統上的應用，就可以在許多手機設備上運行。

2010 年 7 月 12 日這應用軟體運行，2010 年 12 月 15 日對公眾開放。2011 年 12 月 31 日 Google 終止了這應用開發者軟體。現在 MIT 中心手機學習部門支

持運營這個應用軟體，名字改為「MIT 應用開發者」，現在已經有新版本的 AppInventor2。

下圖是本專題智能小車的程式碼，每一個模塊都站在不同的崗位上各自代表著不一樣的功能，以 ListPicker1.AfterPicking 為主的模塊功能是讓藍牙與手機可以相互連接，Canvas.TouchDown 與 Canvas.TouchUP 則是負責按鍵的啟動與停止，而最後的 Button2.Click 就是取消手機與智能小車的藍牙連接，整個程式碼串接起來便是一個智能小車的程式，圖 5.4 是藍牙程式控制的判斷邏輯。

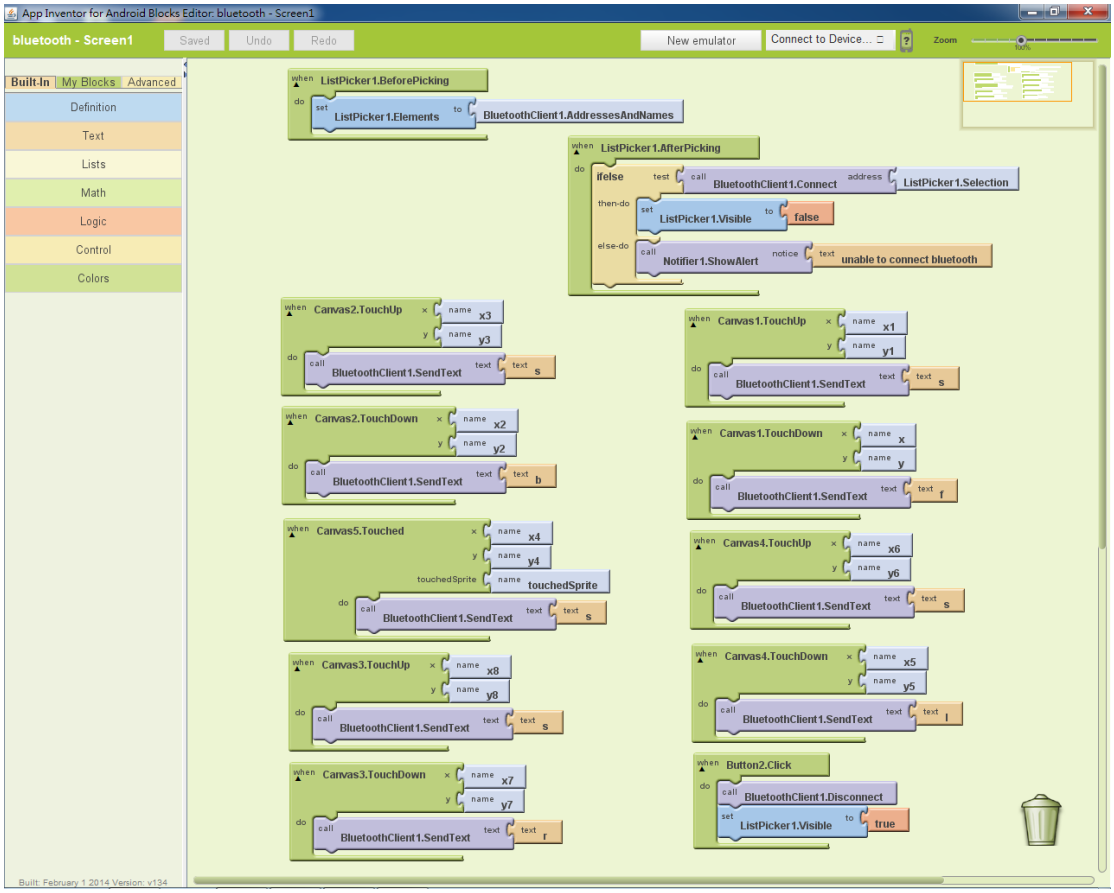


圖 5.4 Appinventor 藍牙遙控程式碼

5.3 改善模式

起初藍牙程式完成要測試時，便發現了藍牙控制器只要按方向鍵，就會向前衝刺而無法停止(圖 5.5)，這樣的情況並不是使用者所樂見的，因此假設有突發狀況發生時執行動作的操作員會反應不及，本專題希望智能小車在適當的情況做適當的事情，需要的是即行即停，就是一個指令一個動作。

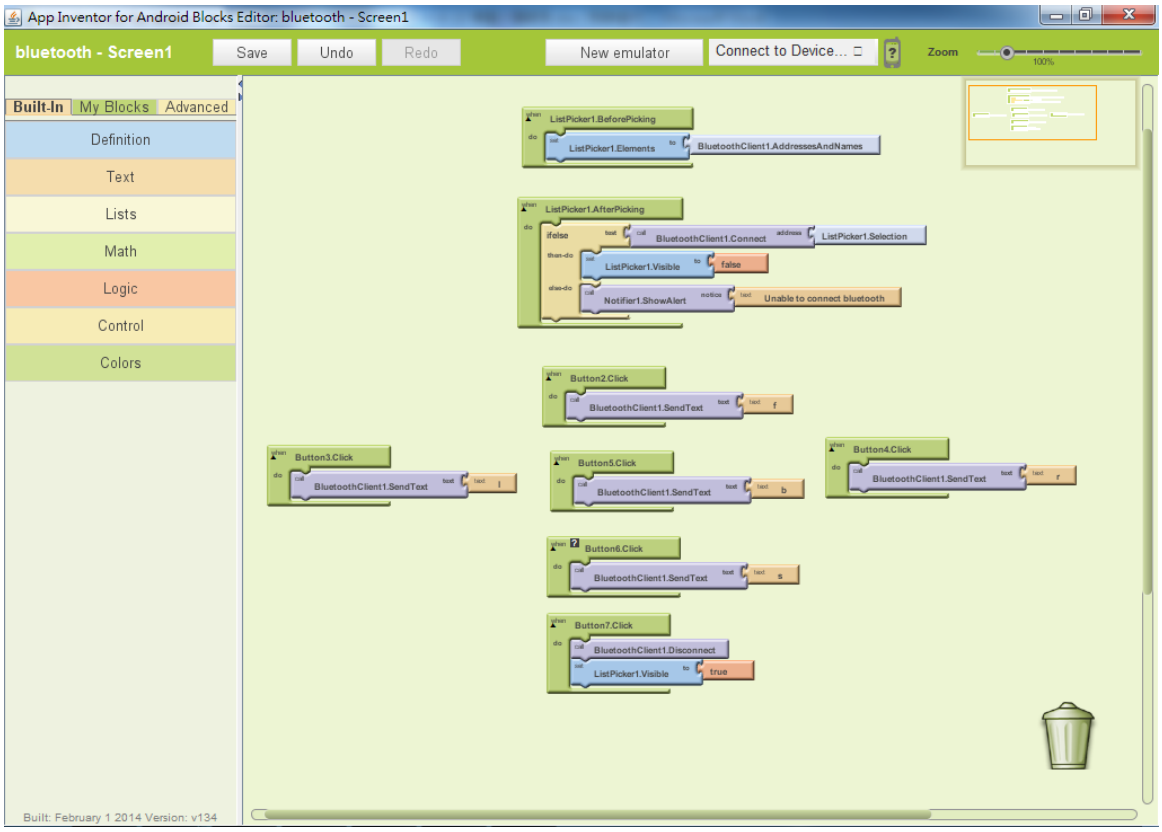


圖 5.5 原本有瑕疵的 APP

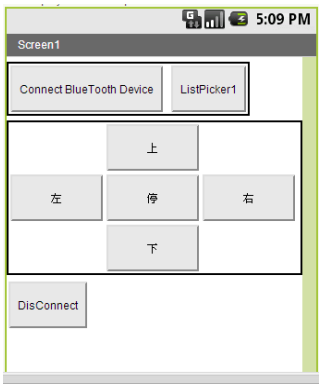


圖 5.6 舊使用者介面

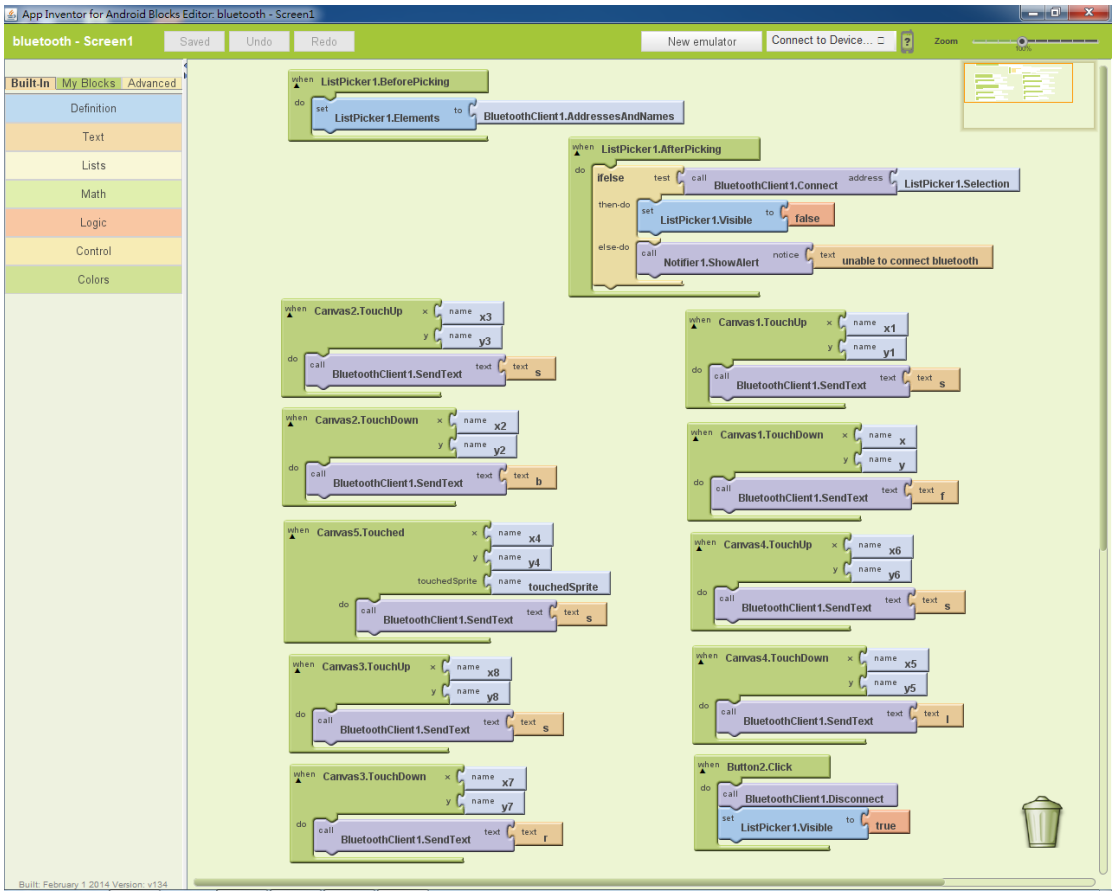


圖 5.7 修改加上停止程式的 APP



圖 5.8 修改後新使用者介面

原始手機設計預設是使用 Button 物件來當作按鈕的使用，可是在程式完成之後，測試時發現，當前進按鈕被按下時，車子就會衝刺而不會自動停止，仍需使用者觸碰停止按鈕來停止智能小車，此種設計靈活度不足容易造成意外。後來本專題將按鈕設計改換成 Canvas 物件(圖 5.7)，當手指離開 Canvas 物件時，智能小車會自動停止，能讓車子的運作更加流暢，實際測試之後便可如實的執行使用者所要求的動作，而不是只會單獨的向前衝刺卻不停止。

5.4 結論

透過手機程式與藍牙遙控智能小車，只要手機安裝遙控程式，就能讓更多使用者遙控同一部智能小車，雖然不能多人同時操控同一部智能小車，但是不至於受限於單一紅外線遙控器只能一人操縱，此外，藍牙的遙控距離遠比紅外線更遠，不至於受到光線及障礙物的影響，此外，GOOGLE 也提供免費的 MIT App Inventor 的服務，能讓更多使用者不需付出更多的成本而附加智能小車更多的功能，例如方向燈與喇叭。

第六章 紅外線循線模式

6.1 所需零件

TCRT5000紅外線循線器(圖6.1)的工作原理是利用紅外線對顏色的反射率不一樣，將反射信號的強弱轉化成電子訊號。黑白尋跡模組在檢測到黑色輸出高電平有效為1，檢測到白色輸出低電平有效為0，檢測高度為 0—3cm。

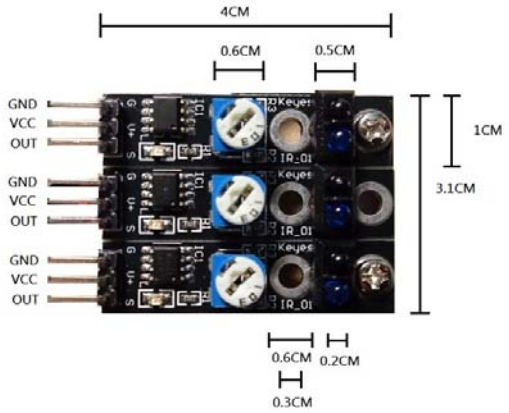


圖 6.1 紅外線循線器

以圖(6.2)中的右循線器為例，接在第三排腳位，GND是接地線接G、V+傳輸5V的電流接V以及S是訊號的傳輸接S，以此類推中感測器接在第四腳位，左感測器接在第七腳位。

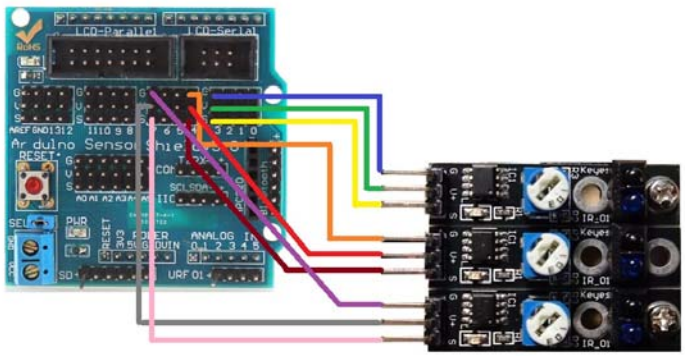


圖 6.2 紅外線循線器接線圖

紅外線感測器介面有3根排針，分別是GND ，VCC與OUT。VCC為供電端，GND為接地，OUT是信號輸出端。主要判斷信號可能值為0或者1，就能判斷物體是否存在。

性能參數：

1. 檢測距離，檢測白紙時約為2釐米。視顏色的不同距離有所不同，白色最遠。
2. 供電電壓：2.5V~12V,不要超過12V

6.2 運作原理

利用黑色對光線的反射率小這個特點，當平面的顏色不是黑色時，感測器發射出去的紅外光被大部分反射回來。於是感測器輸出低電平0。當平面有一黑線，感測器在黑線上方時，因黑色的反射能力很弱，反射回來的紅外光很少，達不到感測器動作的水準，所以感測器輸出1。只要用單片機判斷感測器的輸出端是0或者是1，就能檢測黑線。

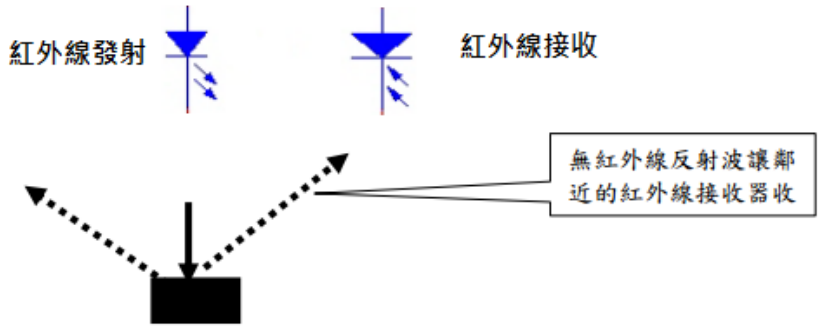


圖 6.3 偵測到黑線時

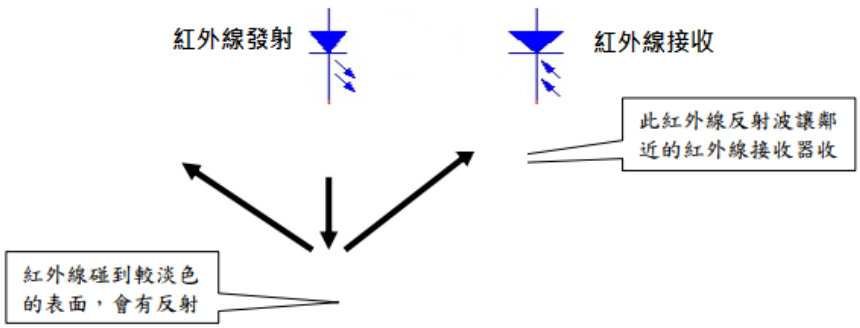


圖 6.4 未偵測道黑線時

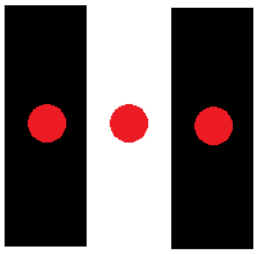
在Arduino開頭需定義紅外線感測器的屬性為輸入訊號：

```
pinMode(SensorLeft, INPUT); //定義左感測器
pinMode(SensorMiddle, INPUT);//定義中感測器
pinMode(SensorRight, INPUT); //定義右感測器
```

當智能小車在行徑的過程中，紅外線感測器可能遇到的狀況有，中感測器偵測到黑色時，左感測器是白色、右感測器是黑色時，車身向左轉，反之則右轉，當全白時則值進。當中感測器偵測到白色時，左感測器是白色、右感測器是黑色時，車身快速左轉，反之則快速右轉，當全白時則停止，當左、中、右感測器為白黑白時，在實務上用不太到，但是依然可循細小的黑線，反之黑白黑時，智能小車會測邊沿著黑線走，在外測會往切線方向衝出黑線，在內測則會轉彎。

表 6.1 邏輯判定

編號	圖示	說明
1		正常行走
2		向左轉
3		向右轉

4		快速左轉
5		快速右轉
6		偏離航道
7		細線上行進
8		沿黑線邊緣行進

6.3 改善模式

智能小車在預設程式執行時過彎時容易出軌，當中間感應器在黑線上，左或右感應器偵測到不在黑線上時，馬達為了讓循線器一直維持在黑線上，會以一輪快一輪慢的的方式，把車體位子矯正，但是外輪會因為旋轉不夠快，而導致轉彎速度跟不上車子前進時的慣性，而往切線方向前進，會造成車體衝出線外。



圖 6.5 出軌示意圖

為了不讓車身出軌，所以將馬達的運作程式稍做調整，將原本的內輪慢外輪快的前進方式改掉，改以內輪瞬間停止，外輪正常前進的方式轉彎，克服了車身前進的慣性，也大幅的修正前述的出軌問題。



圖 6.6 修正後示意圖

6.4 結論

本專題所提的建議方案改善了紅外線循線自走模式容易出軌的問題，雖然降低了行進速度，但提高了軌道內行進的可靠度，如要提高行進速度仍保留軌道內行進的可靠度，則須從馬達前進的方式及調整速度參數來改善。

第七章 超音波避障模式

7.1 所需零件

超音波避障模式主要是以 HC-SR04 超音波感測元件（如圖 7.1）來判斷前方式否有障礙物，原始的智能小車是將超音波感測元件安裝在伺服馬達(Tower Pro sg90，之後稱為舵機)上，藉由伺服馬達的旋轉來判斷左方、前方、右方障礙物距離智能小車的距離，藉此判斷是否該左轉、右轉或直走。

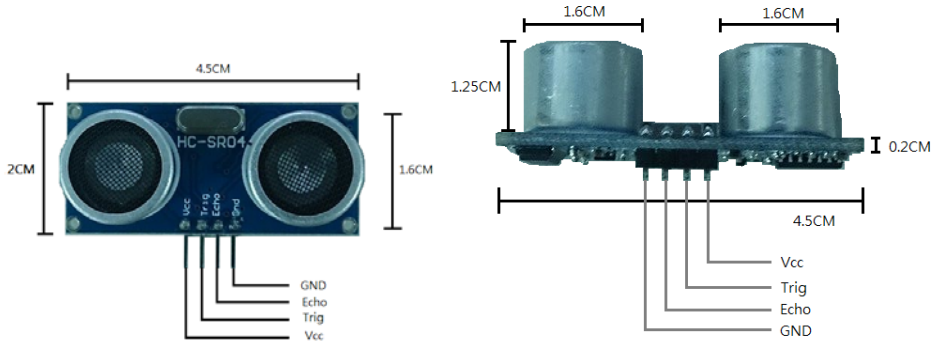


圖 7.1 超音波感測器正面圖

HC-SR04 是一個超音波感測器，它可以探測的距離為 2cm-400cm，精度為 0.3 cm，感應角度為 15 度。此感測器有四根針腳，Vcc 需提供 5v 的電源，GND 需接地，標準輸入輸出有分成發送訊號端的 Trig 接腳與 Echo 的接收訊號接腳。資料如表 7.1 所示。

表 7.1 超音波感測器 HC-SR04 的腳位描述

HC-SR04	Arduino
Vcc	+5V
Trig	GPIO
Echo	GPIO
GND	GND

7.2 運作原理

超音波避障原本的智能小車有提供範例 Arduino 指令(圖 7.2)，當啟動電源開關時，馬達運轉穩定後，若前方距離小於 25 公分但是大於 10 公分，舵機會先左右旋轉，判定左右兩側距離，當左測距離大於右測距離時，則往左轉，反之則右轉，當前方距離小於 10 公分，則自走車先後退 0.2 秒，舵機在旋轉判斷左右距離，左側距離大於右側距離，則向左轉，反之則向右轉〔林能祺、盧嘉泓 2009〕

。

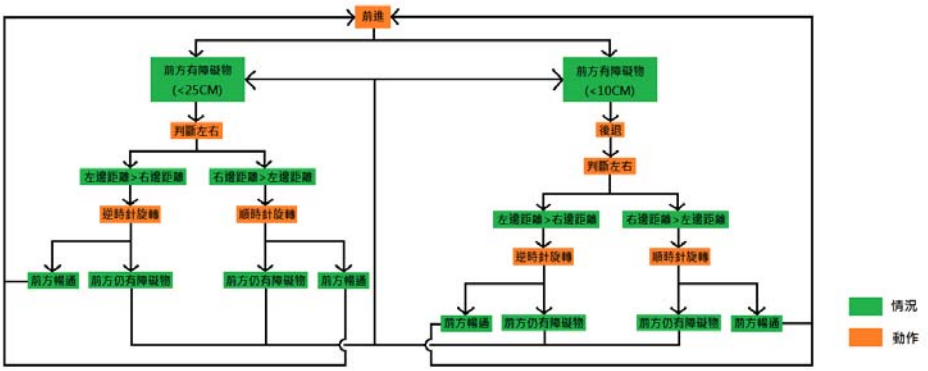


圖 7.2 原本的障礙物閃避流程

7.3 改善模式

在完成智能小車預設的控制程式後發現，舵機旋轉會浪費判斷前方是否有障礙物的時間，導致判定有死角，因此應加以改良。超音波避障的目的是要到達同一方向的終點前閃避所有障礙物，而且還不能超越一定的界線，有時就算閃掉了障礙物，還是淺在一些不可避免除的因素，例如：單輪電壓在 HIGH 時，運作 0.3 秒 車身約偏轉 15 度，雙輪左前右後、或左後右前運作 0.3 秒約 90 度，0.6 秒約 180 度，旋轉角度依照輪胎對地摩擦力，以及旋轉力道都會產生影響，都會造成到達目的地的變數，這些都是專題要去克服的。

經分析研究後，擬以三個超音波測距感測器來判斷左方、前方、右方的障礙物距離，藉以解決需要舵機的問題，但移除舵機後，需重新改寫 arduino 的控制程式。

首先定義超音波腳位，完成後需要新增定義馬達前進的方向，此處以阿拉伯數字代替將行進的方向，以鍵盤右側數字鍵盤為例，8 是前進、2 是後退、6 是右轉、4 是左轉，換句話說，9 就是右前方、7 是左前方、1 是左後方、3 是右後方。腳位與方向變數在 arduino 程式中的定義如圖 7.3 所示。

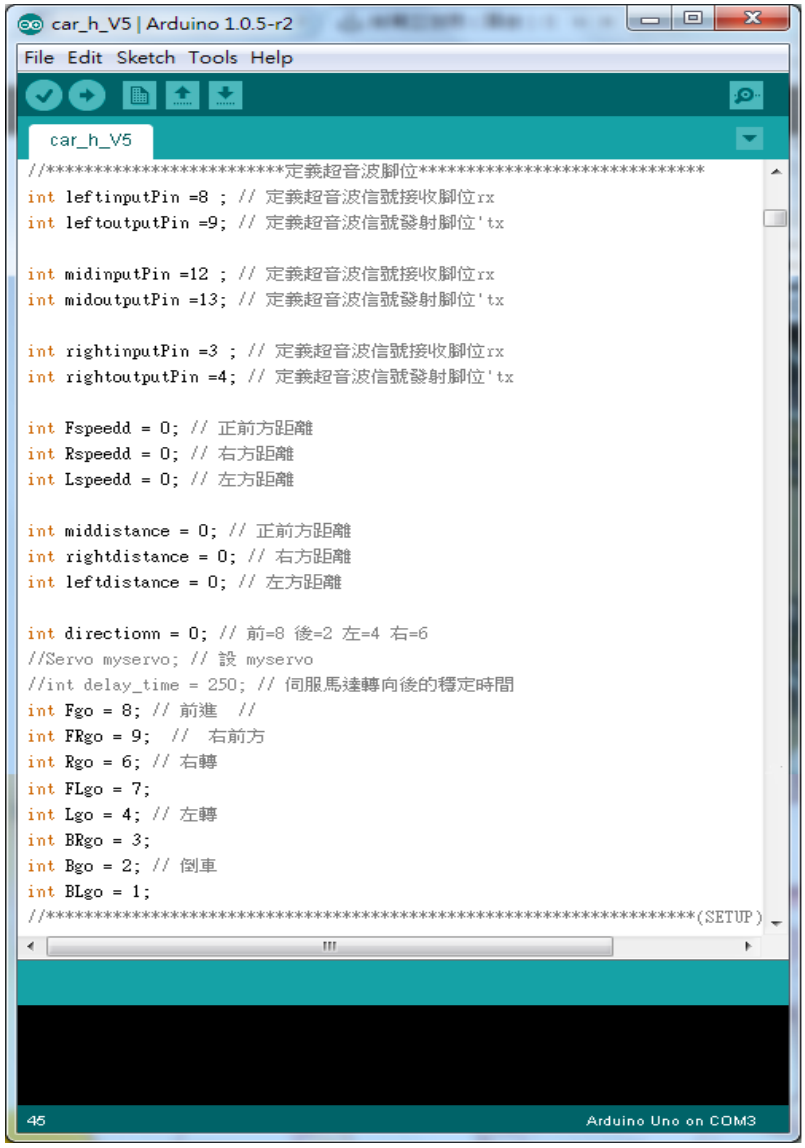


圖 7.3 定義超音波腳位

原有的設計是將超音波感測器裝在舵機上，透過旋轉來判斷左、中、右的障礙物距離，但這樣的是會有時間的誤差，及探測的死角。因此專題決定要將超音波感測器加裝到三個，不僅要將每一次掃描左、中、右的距離儲存在變數中，經由一連串的 IF…ELSE 判斷障礙物，以便繼續完成避障動作。此外避障的動作中新增一變數，當車身旋轉時，向右旋轉變數即加一向左旋轉變數即減一，如變數為零，中感測器感測到物體則優先右轉並變數加一，如圖 7.4 及圖 7.5 所示。

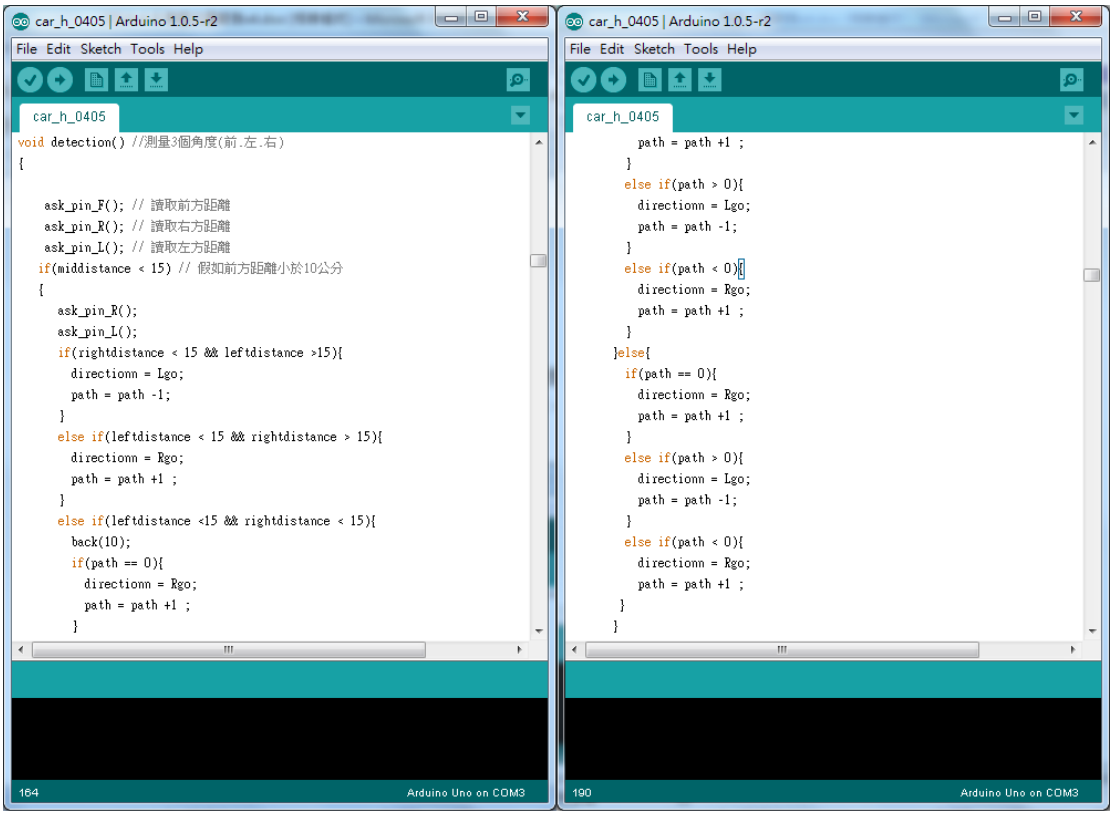


圖 7.4 超音波避障程式碼

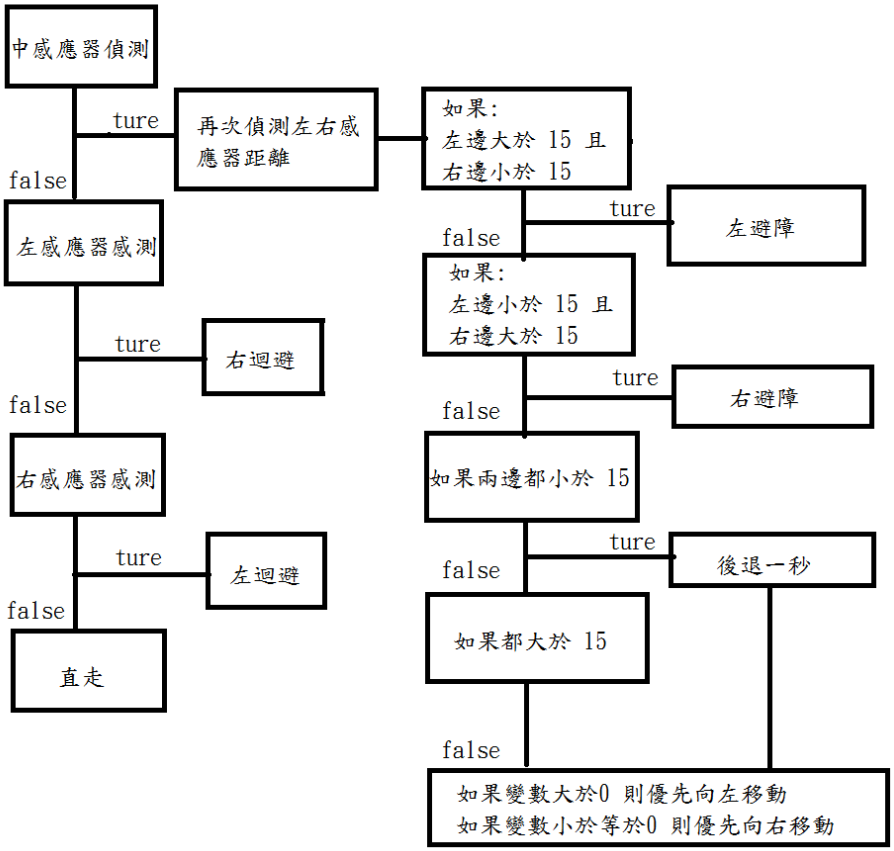


圖 7.5 更改後的超音波避障流程

以左前方有障礙物為例，程式首先檢查是否收到遙控器訊號，如果有停止訓好則立即停止動作，透過 turnR 函數，右轉 0.3 秒，接著透過 advance 函數，前進 0.12 秒，再透過 turnL 函數，左轉 0.25 秒以回正方向，此案例的程式碼如(圖 7.6)所示。

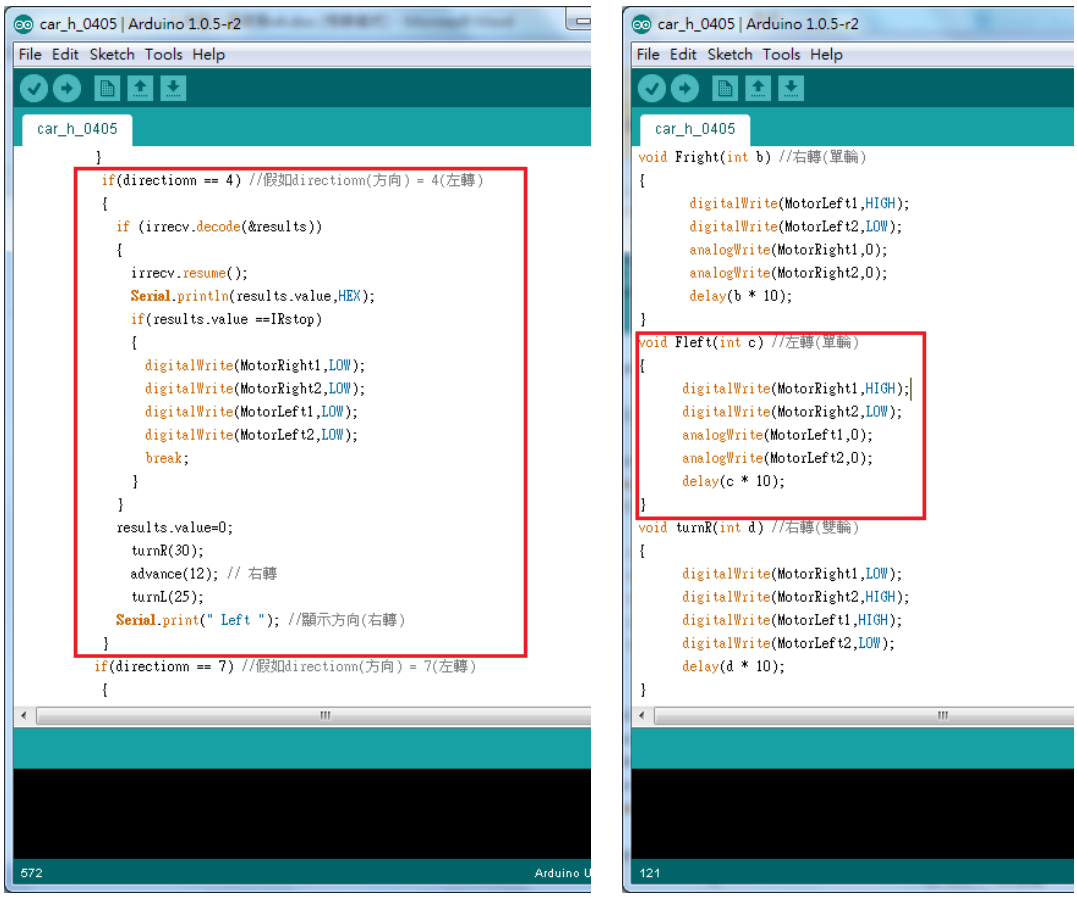


圖 7.3 避障控制(右轉為例)

7.4 結論

本專題透過改善超音波的感測方式，以更多顆超音波避免了判斷的時間延遲與超音波的死角，可以更精準的判斷障礙物位置。並透過變數的設定，能夠使智能小車的行進方向盡量維持在初始方向，而不至於偏離過多，如果要更精確的控制行進方向，可加裝電子羅盤來修正更精準的方向。

第八章 結論

經由這專題研究出可用 Arduino 來控制循線智能小車，並利用藍牙控制的功能，來控制智能小車的行徑及模式切換，進而取代一般的紅外線控制器，讓專題多了點變化性。本專題實現的五大功能：

1. 紅外線偵測：利用反射遮蔽原理來達到循線行動的功能
2. 超音波偵測：利用反射原理達到即時發現障礙物並閃避的功能
3. 紅外線控制：利用紅外線控制器來達到控制自走車的功能
4. 藍牙控制：利用手機的藍牙功能來達到控制自走車的功能
5. 控制面板：L298N 搭配硬體的實做

起初我們在控制智能小車時發現，並不能隨意的操控，後來經過研究發現，原來式遙控器程式碼沒有對應，導致無法隨意控制智能小車，從此可發現到紅外線的缺點：只能一對一，只能由一個操縱者操控。對此本專題才改用藍牙來控制智能小車，現在智慧型手機普及，希望透過其大眾性，利用藍牙的高穿透性及高傳輸距離，來改善紅外線控制器的功能，但並不是讓藍牙取代紅外線，而是讓兩個功能同時並存。

本專題透過手機的藍牙功能遙控智能小車，並修改操控方式，使其能更靈敏的反應操縱者的指令碼。雖然智能小車的藍牙功能還不是很完善，但是改善了現有的紅外線缺點：需對正、只能一對一與距離短的問題。

起初智能小車的紅外線尋線模式，時常因為在過彎時，車身就往切線方向衝出線外，研究了很久，猜測是因為外輪的轉速不夠快的關係，導制外輪來不及轉彎而衝出去線外，所以本專體換了個方式讓內輪以瞬間停止的方式，讓外輪前進達到讓智能小車轉彎，雖然這方法會讓自走的的動作看起來抖抖的，但是他成功的讓智能小車不會衝出線外，經過幾次的實驗，雖然經過調整可以讓車身抖動幅度變小，但是依然無法讓智能小車停止抖動，因為使用的紅外線尋線器只有三顆，能做出的動作比較大，如果增加到七顆，讓智能小車偵測的次數變多，使智能小車做更精密微小的判斷及動作，這部分是可以進一步改善的地方。

原本智能小車在超音波避障時，都必須依靠舵機來旋轉並判斷左右兩側障礙物，但是這樣浪費時間，加上程式設計是旋轉到定點才會打出超音波，所以中間跟兩側之間的死角過大，會因為障礙物過小進入死角而未偵測到，所以決定將舵機移除並加上直接在兩側加裝超音波感測器，減少了偵測的死角，減少舵機旋轉浪費的時間，減少判斷障礙物反應的時間，正確提升自走的的方向性。

參考文獻

1. 吳慈慧、柯振堯(2010)。自動尋線自走車，頁 2、46，台南大學
2. 林能祺、盧嘉泓(2009)。超音波避障之設計，頁 8，元智大學
3. 蔡銘城(2008)。應用遺傳演算法改善自走車循線能力，頁 1，大同大學
4. Maik Schmidt(2012)，Arduino 快速上手指南，馥林文化
5. 程晨(2013)，Arduino 開發實戰指南，上奇資訊
6. 蘇慶輝、蘇慶宏(2001)，藍牙技術解說，全華科技圖書
7. 維基百科：http://zh.wikipedia.org/wiki/Arduino
8. MITAppInventor：http://appinventor.mit.edu/explore/
9. 葉難(2013)，Arduino 練習，紅外線傳送與接收：http://yehnan.blogspot.tw/2013/05/arduino.html

附錄程式碼

```
#include <IRremote.h>
#include <Servo.h>
//************************定義馬達腳位*************************
int MotorRight1=5;
int MotorRight2=6;
int MotorLeft1=10;
int MotorLeft2=11;
int counter=0;
const int irReceiverPin = 2;          //紅外線接收器 OUTPUT 訊號接在 pin 2
char val;
//************************設定所偵測到的 IRcode************************
long IRfront= 0x00FF629D;             //前進碼
long IRback=0x00FFA857;               //後退
long IRturnright=0x00FFC23D;          //右轉
long IRturnleft= 0x00FF22DD;          //左轉
long IRstop=0x00FF02FD;               //停止
long IRcny70=0x00FF6897;              //CNY70 自走模式
long IRAutorun=0x00FF9867;            //超音波自走模式
long IRturnsmallleft= 0x00FFB04F;
```

```
//**********************定義 CNY70 腳位**************************
const int SensorLeft = 7;          //左感測器輸入腳
const int SensorMiddle= 4 ;        //中感測器輸入腳
const int SensorRight = 3;         //右感測器輸入腳
int SL;       //左感測器狀態
int SM;       //中感測器狀態
int SR;       //右感測器狀態
IRrecv irrecv(irReceiverPin);   // 定義 IRrecv 物件來接收紅外線訊號
decode_results results; //解碼結果將放在 decode_results 結構的 result 變數裏
//**********************定義超音波腳位**************************
int leftinputPin =8 ; // 定義超音波信號接收腳位 rx
int leftoutputPin =9; // 定義超音波信號發射腳位'tx

int midinputPin =12 ; // 定義超音波信號接收腳位 rx
int midoutputPin =13; // 定義超音波信號發射腳位'tx

int rightinputPin =3 ; // 定義超音波信號接收腳位 rx
int rightoutputPin =4; // 定義超音波信號發射腳位'tx

int Fspeedd = 0; // 正前方距離
int Rspeedd = 0; // 右方距離
int Lspeedd = 0; // 左方距離

int cyc = 0;
int path = 0;
```

```
int middistance = 0; //  正前方距離
int rightdistance = 0; //  右方距離
int leftdistance = 0; //  左方距離

int directionn = 0; //  前=8  後=2  左=4  右=6
int Fgo = 8;    //  前進
int FRgo = 9;    //右前方
int Rgo = 6;    //  右轉
int FLgo = 7;    //左前方
int Bcyc = 5;    //無功能
int Lgo = 4;    //左轉
int BRgo = 3;    //右後方
int Bgo = 2;    //倒車
int BLgo = 1;    //左後方

//************************************************************(SETUP)
void setup()
{
      Serial.begin(9600);
      pinMode(MotorRight1, OUTPUT);    //  腳位  8 (PWM)
      pinMode(MotorRight2, OUTPUT);    //  腳位  9 (PWM)
      pinMode(MotorLeft1,    OUTPUT);    //  腳位  10 (PWM)
      pinMode(MotorLeft2,    OUTPUT);    //  腳位  11 (PWM)
      irrecv.enableIRIn();          //  啟動紅外線解碼
}
```

```
//************************************************************(Void)
void advance(int a) //  前進
{
      digitalWrite(MotorRight1,HIGH);
      digitalWrite(MotorRight2,LOW);
      digitalWrite(MotorLeft1,HIGH);
      digitalWrite(MotorLeft2,LOW);
      delay(a * 100);
}
void right(int b) //右轉(單輪)
{
      digitalWrite(MotorLeft1,HIGH);
      digitalWrite(MotorLeft2,LOW);
      digitalWrite(MotorRight1,LOW);
      digitalWrite(MotorRight2,LOW);
      delay(b * 100);
}
void left(int c) //左轉(單輪)
{
      digitalWrite(MotorRight1,HIGH);
      digitalWrite(MotorRight2,LOW);
      digitalWrite(MotorLeft1,LOW);
      digitalWrite(MotorLeft2,LOW);
      delay(c * 100);
}
```

```
void Fright(int b) //右轉(單輪)
{
    digitalWrite(MotorLeft1,HIGH);
    digitalWrite(MotorLeft2,LOW);
    analogWrite(MotorRight1,0);
    analogWrite(MotorRight2,0);
    delay(b * 10);
}
void Fleft(int c) //左轉(單輪)
{
    digitalWrite(MotorRight1,HIGH);
    digitalWrite(MotorRight2,LOW);
    analogWrite(MotorLeft1,0);
    analogWrite(MotorLeft2,0);
    delay(c * 10);
}
void turn_L(int d) //左轉(雙輪)
{
    digitalWrite(MotorRight1,LOW);
    digitalWrite(MotorRight2,HIGH);
    digitalWrite(MotorLeft1,HIGH);
    digitalWrite(MotorLeft2,LOW);
    delay(d * 10);
}
```

```
void turn_R(int e) //右轉(雙輪)
{
    digitalWrite(MotorRight1,HIGH);
    digitalWrite(MotorRight2,LOW);
    digitalWrite(MotorLeft1,LOW);
    digitalWrite(MotorLeft2,HIGH);
    delay(e * 10);
}
void stopp(int f) //停止
{
    digitalWrite(MotorRight1,LOW);
    digitalWrite(MotorRight2,LOW);
    digitalWrite(MotorLeft1,LOW);
    digitalWrite(MotorLeft2,LOW);
    delay(f * 100);
}
void back(int g) //後退
{
    digitalWrite(MotorRight1,LOW);
    digitalWrite(MotorRight2,HIGH);
    digitalWrite(MotorLeft1,LOW);
    digitalWrite(MotorLeft2,HIGH);;
    delay(g * 100);
}
```

```
void detection() //測量 3 個角度(前.左.右)
{
    ask_pin_F(); //  讀取前方距離
    ask_pin_R(); //  讀取右方距離
    ask_pin_L(); //  讀取左方距離
    if(middistance < 15) //  假如前方距離小於 10 公分
    {
        ask_pin_R();
        ask_pin_L();
        if(rightdistance < 15 && leftdistance >15)
        {
            directionn = Lgo;
            path = path -1;
        }
        else if(leftdistance < 15 && rightdistance > 15)
        {
            directionn = Rgo;
            path = path +1 ;
        }
        else if(leftdistance <15 && rightdistance < 15)
        {
            back(10);
            if(path == 0)
            {
                directionn = Rgo;
                path = path +1 ;
            }
```

```
        else if(path > 0)
        {
            directionn = Lgo;
            path = path -1;
        }
        else if(path < 0)
        {
            directionn = Rgo;
            path = path +1 ;
        }
    }
    else
    {
        if(path == 0)
        {
            directionn = Rgo;
            path = path +1 ;
        }
        else if(path > 0)
        {
            directionn = Lgo;
            path = path -1;
        }
        else if(path < 0)
        {
            directionn = Rgo;
            path = path +1 ;
        }
```

```
            }
        }
        else if(rightdistance < 10 )
        {
            directionn = FLgo;
            path = path -1 ;
        }
        else if(leftdistance < 10 )
        {
            directionn = FRgo;
            path = path +1 ;
        }
        else
        {
            directionn =Fgo;
        }
}
//************************************************************
void ask_pin_F() //  量出前方距離
{
        digitalWrite(midoutputPin, LOW); //  讓超聲波發射低電壓 2μs
        delayMicroseconds(2);
        //讓超聲波發射高電壓 10μs，這裡至少是 10μs
        digitalWrite(midoutputPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(midoutputPin, LOW); //  維持超聲波發射低電壓
        float Mdistance = pulseIn(midinputPin, HIGH); //  讀差相差時間
        Mdistance= Mdistance/5.8/10; //  將時間轉為距離距离（單位：公分）
        middistance = Mdistance; //  將距離  讀入 Fspeedd(前速)
```

```
}
//*****************************************************************
void ask_pin_L() //  量出左邊距離
{
    digitalWrite(leftoutputPin, LOW); //  讓超聲波發射低電壓 2μs
    delayMicroseconds(2);
    //  讓超聲波發射高電壓 10μs，這裡至少是 10μs
    digitalWrite(leftoutputPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(leftoutputPin, LOW); //  維持超聲波發射低電壓
    float Ldistance = pulseIn(leftinputPin, HIGH); //  讀差相差時間
    Ldistance= Ldistance/5.8/10; //  將時間轉為距離距离（單位：公分）
    leftdistance = Ldistance; //  將距離  讀入 Fspeedd(前速)
}
//*****************************************************************
void ask_pin_R() //  量出右邊距離
{
    digitalWrite(rightoutputPin, LOW); //  讓超聲波發射低電壓 2μs
    delayMicroseconds(2);
    //  讓超聲波發射高電壓 10μs，這裡至少是 10μs
    digitalWrite(rightoutputPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(rightoutputPin, LOW); //  維持超聲波發射低電壓
    float Rdistance = pulseIn(rightinputPin, HIGH); //  讀差相差時間
    Rdistance= Rdistance/5.8/10; //  將時間轉為距離距离（單位：公分）
    rightdistance = Rdistance; //  將距離  讀入 Fspeedd(前速)



}
```

```
//**************************************************************(LOOP)
void loop()
{
    performCommand();
//**************************************************************正常遙控模式
    if (irrecv.decode(&results))
    {
        // 解碼成功，收到一組紅外線訊號

//*****************************************************************
        if (results.value == IRfront)//前進
        {
            advance(10);//前進
        }

//*****************************************************************
        if (results.value ==    IRback)//後退
        {
            back(10);//後退
        }

//*****************************************************************
        if (results.value == IRturnright)//右轉
        {
            right(6); //  右轉
        }
```

```
//*****************************************************************
                if (results.value == IRturnleft)//左轉
                {
                    left(6); //  左轉)
                }

//*****************************************************************
                if (results.value == IRstop)//停止
                {
                    digitalWrite(MotorRight1,LOW);
                    digitalWrite(MotorRight2,LOW);
                    digitalWrite(MotorLeft1,LOW);
                    digitalWrite(MotorLeft2,LOW);
                }
//*****************************cny70 模式自走模式  黑:LOW  白:HIGH
                if (results.value == IRcny70)
                {
                    pinMode(SensorLeft, INPUT); //定義左感測器
                    pinMode(SensorMiddle, INPUT);//定義中感測器
                    pinMode(SensorRight, INPUT); //定義右感測器
                    digitalWrite(2,HIGH);
                    while(IRcny70)
                    {
                        SL = digitalRead(SensorLeft);
                        SM = digitalRead(SensorMiddle);
                        SR = digitalRead(SensorRight);
```

```
if (SM == LOW)//中感測器在黑色區域
{
    // 左黑右白，向左轉彎
    if (SL == HIGH & SR == LOW)
    {
        analogWrite(MotorRight1,LOW);
        analogWrite(MotorRight2,255);
        analogWrite(MotorLeft1,0);
        analogWrite(MotorLeft2,LOW);
    }
    //左白右黑，向右轉彎
    else if (SR == HIGH & SL == LOW)
    {
        analogWrite(MotorRight1,0);
        analogWrite(MotorRight2,LOW);
        analogWrite(MotorLeft1,LOW);
        analogWrite(MotorLeft2,255);
    }
    else   // 兩側均為白色，直進
    {
        digitalWrite(MotorRight1,HIGH);
        digitalWrite(MotorRight2,LOW);
        digitalWrite(MotorLeft1,HIGH);
        digitalWrite(MotorLeft2,LOW);
    }
}
```

```
else // 中感測器在白色區域
{
    // 左黑右白，快速左轉
    if (SL == HIGH & SR == LOW)
    {
        turnR(1);
    }
    // 左白右黑，快速右轉
    else if (SR == HIGH & SL == LOW)
    {
        turnL(1);
    }
    else // 都是白色，停止
    {
        digitalWrite(MotorRight1,HIGH);
        digitalWrite(MotorRight2,LOW);
        digitalWrite(MotorLeft1,HIGH);
        digitalWrite(MotorLeft2,LOW);
    }
}
```

```
            if (irrecv.decode(&results))
            {
                irrecv.resume();
                Serial.println(results.value,HEX);
                if(results.value ==IRstop)
                {
                    analogWrite(MotorRight1,200);
                    analogWrite(MotorRight2,200);
                    analogWrite(MotorLeft1,200);
                    analogWrite(MotorLeft2,200);
                    break;
                }
            }
        }
        results.value=0;
    }
//*************************************************超音波自走模式
    if (results.value ==IRAutorun )
    {
        pinMode(leftinputPin, INPUT); //  定義超音波輸入腳位
        pinMode(leftoutputPin, OUTPUT); //  定義超音波輸出腳位
        pinMode(midinputPin, INPUT); //  定義超音波輸入腳位
        pinMode(midoutputPin, OUTPUT); //  定義超音波輸出腳位
        pinMode(rightinputPin, INPUT); //  定義超音波輸入腳位
        pinMode(rightoutputPin, OUTPUT); //  定義超音波輸出腳位
        while(IRAutorun)
```

```
{
    detection(); //測量角度  並且判斷要往哪一方向移動
    if(directionn == 8) //假如 directionn(方向) = 8(前進)
    {
        if (irrecv.decode(&results))
        {
            irrecv.resume();
            Serial.println(results.value,HEX);
            if(results.value ==IRstop)
            {
                digitalWrite(MotorRight1,LOW);
                digitalWrite(MotorRight2,LOW);
                digitalWrite(MotorLeft1,LOW);
                digitalWrite(MotorLeft2,LOW);
                break;
            }
        }
        results.value=0;
        advance(1); //  正常前進
        Serial.print(" Advance "); //顯示方向(前進)
        Serial.print(" ");
    }
    if(directionn == 5) //假如 directionn(方向) = 5(前進)
    {
        if (irrecv.decode(&results))
        {
            irrecv.resume();
            Serial.println(results.value,HEX);
```

```
            if(results.value ==IRstop)
            {
                digitalWrite(MotorRight1,LOW);
                digitalWrite(MotorRight2,LOW);
                digitalWrite(MotorLeft1,LOW);
                digitalWrite(MotorLeft2,LOW);
                break;
            }
        }
        results.value=0;
        turnR(2); //  正常前進
        Serial.print(" Advance "); //顯示方向(前進)
        Serial.print(" ");
    }
    if(directionn == 2) //假如 directionn(方向) = 2(倒車)
    {
        if (irrecv.decode(&results))
        {
            irrecv.resume();
            Serial.println(results.value,HEX);
            if(results.value ==IRstop)
            {
                digitalWrite(MotorRight1,LOW);
                digitalWrite(MotorRight2,LOW);
                digitalWrite(MotorLeft1,LOW);
                digitalWrite(MotorLeft2,LOW);
                break;
            }
        }
```

```
        results.value=0;
        back(10); //  倒退(車)

        Serial.print(" Reverse "); //顯示方向(倒退)
        stopp(1)    ;
}
if(directionn == 6) //假如 directionn(方向) = 6(右轉)
{
        if (irrecv.decode(&results))
        {
                irrecv.resume();
                Serial.println(results.value,HEX);
                if(results.value ==IRstop)
                {
                        digitalWrite(MotorRight1,LOW);
                        digitalWrite(MotorRight2,LOW);
                        digitalWrite(MotorLeft1,LOW);
                        digitalWrite(MotorLeft2,LOW);
                        break;
                }
        }
        results.value=0;
        turn_R(25);
        advance(12); //  直進
        turn_L(30);
        Serial.print(" Right "); //顯示方向(右轉)
}
if(directionn == 4) //假如 directionn(方向) = 4(左轉)
```

```
{
    if (irrecv.decode(&results))
    {
        irrecv.resume();
        Serial.println(results.value,HEX);
        if(results.value ==IRstop)
        {
            digitalWrite(MotorRight1,LOW);
            digitalWrite(MotorRight2,LOW);
            digitalWrite(MotorLeft1,LOW);
            digitalWrite(MotorLeft2,LOW);
            break;
        }
    }
    results.value=0;
    turn_L(30);
    advance(12); //  直進
    turn_R(25);
    Serial.print(" Left "); //顯示方向(左轉)
}
if(directionn == 7) //假如 directionn(方向) = 7(左轉)
{
    if (irrecv.decode(&results))
    {
        irrecv.resume();
        Serial.println(results.value,HEX);
        if(results.value ==IRstop)
        {
```

```
                digitalWrite(MotorRight1,LOW);
                digitalWrite(MotorRight2,LOW);
                digitalWrite(MotorLeft1,LOW);
                digitalWrite(MotorLeft2,LOW);
                break;
            }
        }
        results.value=0;
        Fright(50);
        advance(6);
        Fleft(50);
        Serial.print(" Left "); //顯示方向(右轉)
}
if(directionn == 9) //假如 directionn(方向) = 9(右轉)
{
        if (irrecv.decode(&results))
        {
            irrecv.resume();
            Serial.println(results.value,HEX);
            if(results.value ==IRstop)
            {
                digitalWrite(MotorRight1,LOW);
                digitalWrite(MotorRight2,LOW);
                digitalWrite(MotorLeft1,LOW);
                digitalWrite(MotorLeft2,LOW);
                break;
            }
        }
```

```
        results.value=0;
        Fleft(50);
        advance(6);
        Fright(50);
        Serial.print(" Left "); //顯示方向(右轉)
}
if(directionn == 1) //假如 directionn(方向) = 1(右轉)
{
        if (irrecv.decode(&results))
        {
                irrecv.resume();
                Serial.println(results.value,HEX);
                if(results.value ==IRstop)
                {
                        digitalWrite(MotorRight1,LOW);
                        digitalWrite(MotorRight2,LOW);
                        digitalWrite(MotorLeft1,LOW);
                        digitalWrite(MotorLeft2,LOW);
                        break;
                }
        }
        results.value=0;
        back(1);
        turnL(3);
        advance(3);
        turnR(3);
        Serial.print(" Left "); //顯示方向(右轉)
}
```

```
if(directionn == 3) //假如 directionn(方向) = 3(右轉)
{
    if (irrecv.decode(&results))
    {
        irrecv.resume();
        Serial.println(results.value,HEX);
        if(results.value ==IRstop)
        {
            digitalWrite(MotorRight1,LOW);
            digitalWrite(MotorRight2,LOW);
            digitalWrite(MotorLeft1,LOW);
            digitalWrite(MotorLeft2,LOW);
            break;
        }
    }
    results.value=0;
    back(1);
    turnR(3);
    advance(3);
    turnL(3);
    Serial.print(" Left "); //顯示方向(右轉)
}
if (irrecv.decode(&results))
{
    irrecv.resume();
    Serial.println(results.value,HEX);
    if(results.value ==IRstop)
```

```
                    {
                        digitalWrite(MotorRight1,LOW);
                        digitalWrite(MotorRight2,LOW);
                        digitalWrite(MotorLeft1,LOW);
                        digitalWrite(MotorLeft2,LOW);
                        break;
                    }
                }
            }
            results.value=0;
        }
//*****************************************************************
            else
            {
                digitalWrite(MotorRight1,LOW);
                digitalWrite(MotorRight2,LOW);
                digitalWrite(MotorLeft1,LOW);
                digitalWrite(MotorLeft2,LOW);
            }
            irrecv.resume();       // 繼續收下一組紅外線訊號
        }
}
void performCommand()
{
    if (Serial.available())
    {
        val = Serial.read();
    }
```

```
if (val == 'f')      // Forward
{
      advance(10);
}
else if (val == 'z')         // Stop Forward
{
      stopp(10) ;
}
else if (val == 'b')         // Backward
{
      back(10);
}
else if (val == 'y')         // Stop Backward
{
      back(10);
}
else if (val == 'r')          // Right
{
      turnR(10);
}
else if (val == 'l')         // Left
{
      turnL(10);
}
else if (val == 'v')         // Stop Turn
{
      stopp(10) ;
}
else if (val == 's')         // Stop
```

```
        {
            stopp(10) ;
        }
    }
```