

# 物聯網實物專題--遙控自走車

高瑞鴻

指導老師：倪同亮

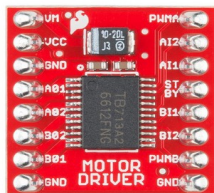
# 專題發想和動機

- 1. 希望能設計一個在上課中學習到的整合系統：
  - 課堂過程中學到了許多的知識，希望可以設計出一個系統可以把所學的東西做個統整應用以及整合
- 2. 練習如何構想, 研擬, 開發, 測試之過程：
  - 把一個很抽象或者一個小小的想法一步步的去實現並且呈現出來
- 3. 練習計畫排程, 修改：
  - 一件企劃或專案不是一蹴可幾的，需要做排程和不斷的修改慢慢的完成
- 4. 自我學習：
  - 課堂雖然習得的許多，但不是每一個細節或方法都會教到，可以藉此專題去練習如何解決問題
- 5. 計畫包裝和展現：
  - 把過程和成果做一個完美的展現

# 系統介紹

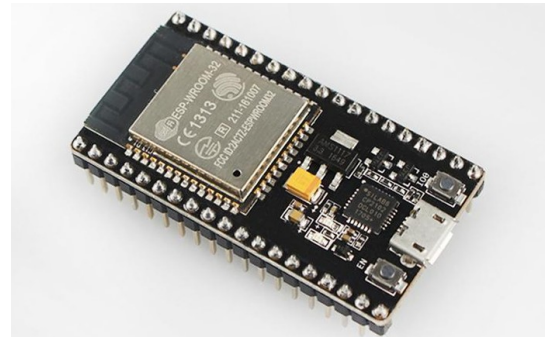
- 1.做出一輛可以遙控的自走車
- 2.利用手機的APP程式做遙控
- 3.在車子行走過程中可以依照目前狀態回傳相關資訊，並且在手機尚呈現該相關資訊
- 4.建立具有CURD的資料庫
- 5.讓自走車在手機的切換功能下做出不同功能
- 6.具有自走避碰功能

# 硬體



# ESP32

- ESP32是一系列低成本，低功耗的單晶片微控制器，整合了Wi-Fi和雙模藍牙。ESP32系列採用Tensilica Xtensa LX6微處理器，包括雙核心和單核變體，內建天線開關，RF變換器，功率放大器，低雜訊接收放大器，濾波器和電源管理模組。
- 處理器：
  - CPU: Xtensa 雙核心 (或者單核心) 32位元 LX6 微處理器, 工作時脈 160/240 MHz, 運算能力高達 600 DMIPS
- 記憶體：
  - 448 KB ROM (64KB+384KB)
  - 520 KB SRAM
  - 16 KB RTC SRAM,SRAM 分為兩種
    - 第一部分 8 KB RTC SRAM 為慢速儲存器,可以在 Deep-sleep 模式下被次處理器存取
    - 第二部分 8 KB RTC SRAM 為快速儲存器,可以在 Deep-sleep 模式下RTC 啟動時用於數據儲存以及 被主 CPU 存取。
  - 1 Kbit 的 eFuse，其中 256 bit 為系統專用（MAC 位址和晶片設定）；其餘 768 bit 保留給使用者應用，這些 應用包括 Flash 加密和晶片 ID。
  - QSPI 支援多個快閃記憶體/SRAM
  - 可使用 SPI儲存器 對映到外部記憶體空間，部分儲存器可做為外部儲存器的 Cache
  - 最大支援 16 MB 外部 SPI Flash
  - 最大支援 8 MB 外部 SPI SRAM

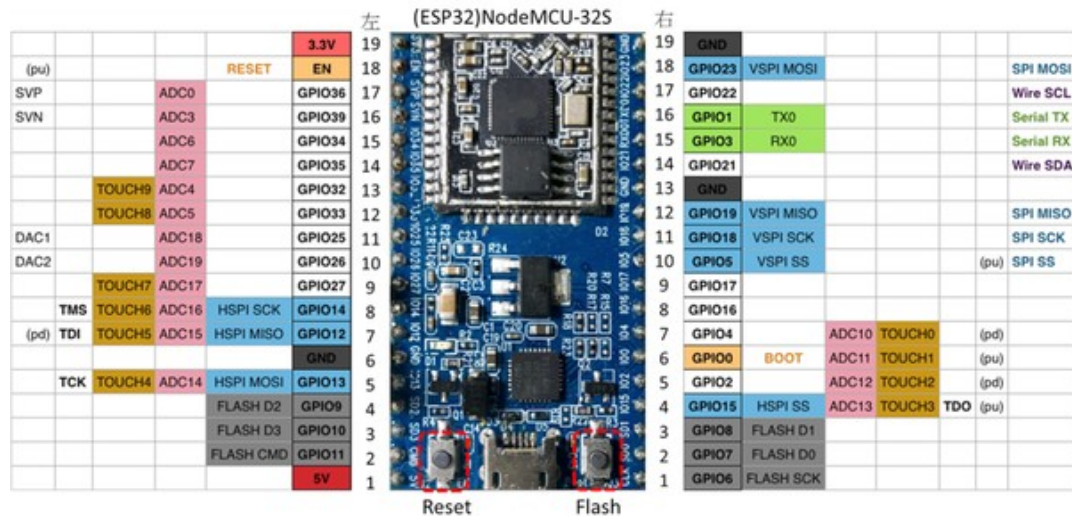


# ESP32

- 外部介面：
- 34個 GPIO
- 12-bit SAR ADC，多達18個通道
- 2個 8位元 D/A 轉換器
- 10 個觸控感應器
- 4個 SPI
- 2個 I2S
- 2個 I2C
- 3個 UART
- 1個 Host SD/eMMC/SDIO
- 1個 Slave SDIO/SPI
- 帶有專用 DMA 的乙太網路介面,支援 IEEE 1588
- CAN 2.0
- 紅外線傳輸
- 電機 PWM
- LED PWM, 多達16個通道
- 霍爾感應器

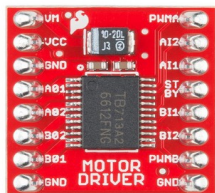
其它：

- DHT11不可以用GPIO 5、26
- GPIO 12, 2燒錄時不可接任何裝置，請空接，燒錄完成後，再接回，否則會上傳失敗。
- WiFi啟動後，2,4,12,13,14,15,25,26,27僅能數位讀取，不可類比



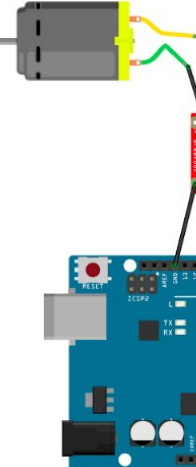
# TB6612FNG

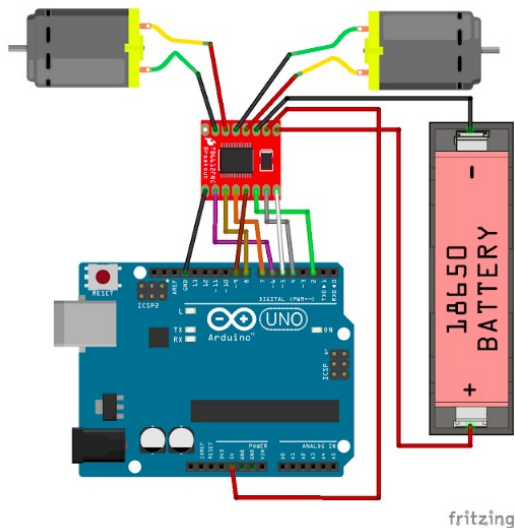
- TB6612FNG是東芝生產的馬達驅動與控制IC，內部包含兩組H橋式電路，可驅動和控制兩個小型直流馬達，或者一個雙極性步進馬達。
- TB6612FNG 電機控制模組可以在1.2A（3.2A峰值）的電流下控制兩個 DC 馬達或 1個雙極步進馬達，兩個輸入信號（IN1和IN2）可用控制馬達四種模式：正轉、反轉、短制動(Short Brake)和停止(Stop)。兩個馬達輸出可以單獨控制，且其速度可透過 PWM 輸入信號控制，頻率高達100kHz。其中 STBY 引腳高電位時以使馬達退出待機模式。
- TB6612FNG 電機控制模組商品規格如下：
- 晶片型號：TB6612FNG
- 晶片工作電壓：2.7V-5.5V
- 馬達工作電壓：2.5V-13.5V
- 輸出電流：1.2A-3.2A
- H橋式電路：MOSFET
- 運行模式：順時針、時針、剎車、停鈍



# TB6612FNG

[SparkFun\_TB6612FNG 程式庫]

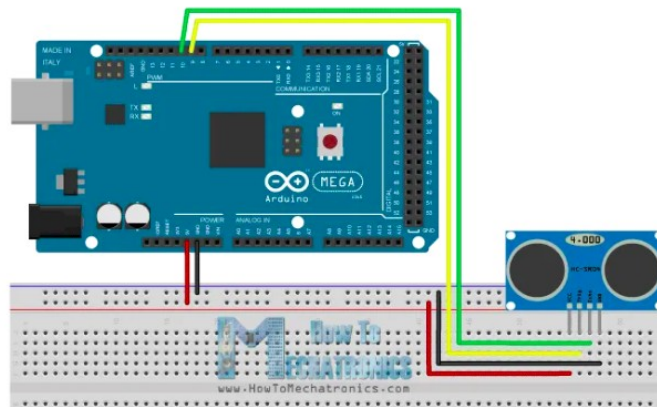
- SparkFun\_TB6612FNG 程式庫對馬達物件的方法如下：
  - `Motor(int In1pin, int In2pin, int PWMpin, int offset, int STBYpin);`
  - 建立馬達物件，In1pin 及 In2pin 為控制的
  - `drive(int speed, int duration)`
  - 驅動馬達。speed 是介於 -255 至 255 的整數值，數字越大，轉速越高，負值代表反轉。duration 設定轉動的時間，為選擇性參數，不輸入代表一直轉動。
  - `standby()`
  - 設定成待機模式。開始執行 drive、forward、back、left 及 right 指令，待機模式會停止。
  - `forward(Motor motor1, Motor motor2, int speed)`
  - 驅動兩個馬達前進。speed 是介於 -255 至 255 的整數值，若為負值代表反轉，若不輸入 speed 值，以預設的速度值 255 轉動。
  - `back(Motor motor1, Motor motor2, int speed)`
  - 驅動兩個馬達後退。
  - `left(Motor motor1, Motor motor2, int speed)`
  - 讓左、右馬達分別反轉和正轉，方向向左轉動。
  - `right(Motor motor1, Motor motor2, int speed)`
  - 讓左、右馬達分別正轉和反轉，方向向右轉動。
  - `brake(Motor motor1, Motor motor2)`
  - 煞車。將兩個馬達驅動設成高電位讓馬達停止。
- 





# Ultrasonic Sensor(HC-SR04)

- 超音波感測器( Ultrasonic Sensor )主要作為距離偵測使用，在生活中非常常見，像汽車倒車雷達就是採用防水型的超音波感測器，原理為利用超聲波於空氣中的傳遞反射特性，來計算與障礙物的距離。
- 超音波感測器是利用聲波在空氣中的傳播原理來計算距離，  
聲波速度為 340 公尺/ 1秒，所以每走1公尺所需時間為  
0.002941 秒，走1 cm就是 0.00002941 秒( 29.41us )，  
簡單來說我們僅需要計算超音波的行走時間就需要可以知道  
物體與我們的距離。
- 這邊有個地方需要注意一下，因為我們從感測器上得到的時間是為  
傳遞時間+反射時間 的總值  $2T$ ，實際的距離時間就要將 行走總時間  
除以2 才為正確的數值，所以得到物體距離的數學式就為  
『 距離cm =  $T/29.41\mu s$  』，之後就可以將這條數學式放入程式中進行運算  
就可以得到距離了。



# 軟體




# C++

- C++是一種被廣泛使用的電腦程式設計語言。它是一種通用程式設計語言，支援多重程式設計範式，例如過程化程式設計、資料抽象化、物件導向程式設計、泛型程式設計和設計模式等。
- 比雅尼·斯特勞斯特魯普博士在貝爾實驗室工作期間在20世紀80年代發明並實現了C++。起初，這種語言被稱作「C with Classes」（「包含『類』的C語言」），作為C語言的增強版出現。隨後，C++不斷增加新特性。虛擬函式（virtual function）、運算子多載（operator overloading）、多繼承（multiple inheritance）、標準模板庫（standard template library, STL）、例外處理（exception）、執行時型別資訊（runtime type information）、命名空間（namespace）等概念逐漸納入標準。1998年，國際標準組織（ISO）頒布了C++程式設計語言的第一個國際標準ISO/IEC 14882:1998，目前最新標準為ISO/IEC 14882:2020。根據《C++編程思想》（Thinking in C++）一書，C++與C的代碼執行效率往往相差在±5%之間。

# ARDUINO IDE

- IDE:整合開發環境（Integrated Development Environment，簡稱IDE，也稱為Integration Design Environment、Integration Debugging Environment）是一種輔助程式開發人員開發軟體的應用軟體，在開發工具內部就可以輔助編寫原始碼文字、並編譯打包成為可用的程式，有些甚至可以設計圖形介面。
- IDE通常包括程式語言編輯器、自動構建工具、通常還包括除錯器。有些IDE包含編譯器／直譯器，如微軟的Microsoft Visual Studio，有些則不包含，如Eclipse、SharpDevelop等，這些IDE是通過呼叫第三方編譯器來實現代碼的編譯工作的。有時IDE還會包含版本控制系統和一些可以設計圖形使用者介面的工具。許多支援物件導向的現代化IDE還包括了類別瀏覽器、物件檢視器、物件結構圖。雖然目前有一些IDE支援多種程式語言（例如Eclipse、NetBeans、Microsoft Visual Studio），但是一般而言，IDE主要還是針對特定的程式語言而量身打造
- Arduino整合開發環境(IDE)是使用C和C++的函數編寫的跨平台應用程序。它用於編寫程序並將其上載到Arduino兼容的開發板，而且還可以藉助第三方內核和其他供應商開發板。IDE源代碼是在GNU通用公共許可第2版下發布的。Arduino IDE使用特殊的代碼結構規則支持C和C++語言。

A screenshot of the Arduino IDE interface. The title bar reads "Blink | Arduino 1.8.5". The menu bar includes "File", "Edit", "Tools", "Window", and "Help". The main text area contains the following code:

```
This example code is in the public domain.  
  
http://www.arduino.cc/en/Tutorial/Blink  
*/  
  
// the setup function runs once when you press reset or power the board  
void setup() {  
  // initialize digital pin LED_BUILTIN as an output.  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000); // wait for a second  
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW  
  delay(1000); // wait for a second  
}
```

The status bar at the bottom shows "32" on the left and "Arduino/Genuino Uno on COM1" on the right.

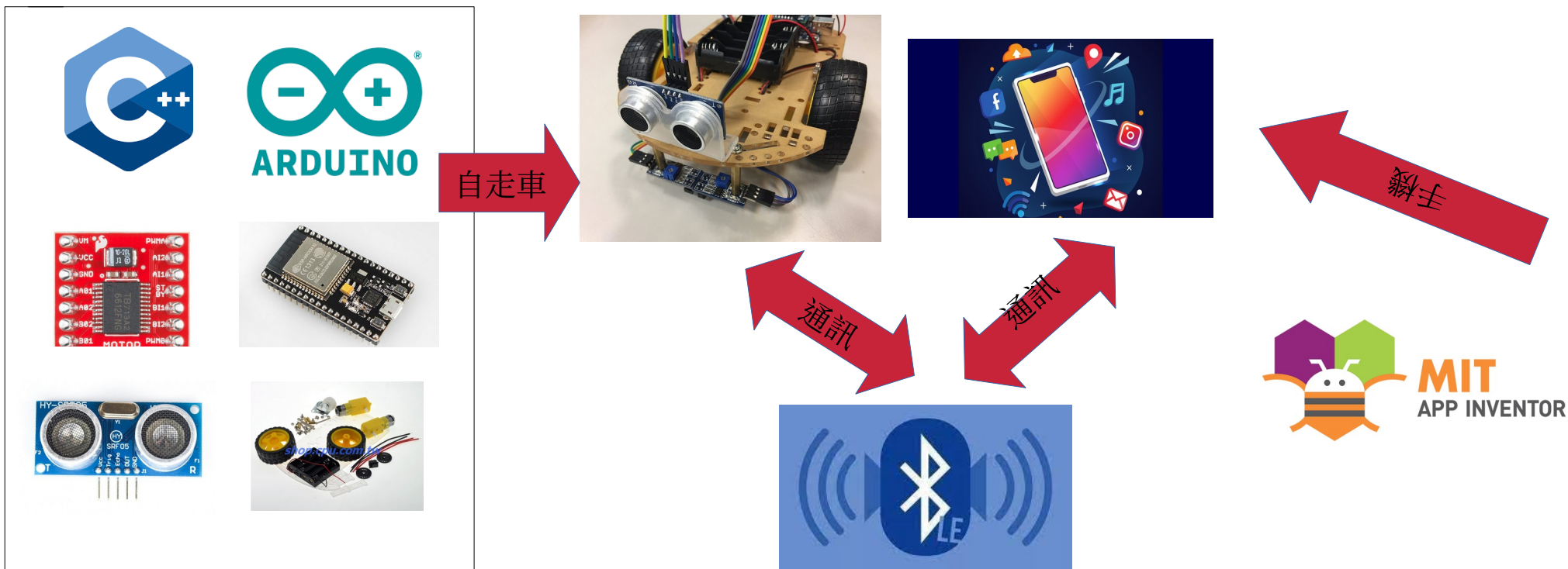
# APPINVENTOR2

- 視覺化程式設計語言（英語：Visual programming language，簡稱：VPL），又稱『圖形化程式語言』、『視覺化程式編成語言』，是一類程式設計語言。使用者利用圖形化元素進行程式設計，較文字式程式設計簡單。VPL以視覺表達為基礎，利用『文法』或是某種『輔助標記』進行圖形與文字的排列。許多VPL建基於『方塊與箭頭』的概念之上，以方塊或螢幕上的物件為本體，以箭頭相連接，以直線段與弧線段代表相互之間的關係。
- Android應用開發者（英語：App Inventor）是一款卡通圖形界面的Android智慧型手機應用程式開發軟體。它起先由Google提供的應用軟體，現在由麻省理工學院維護及營運。
- 它可以讓任何熟悉或不熟悉程序設計的人來創造基於Android作業系統的應用軟體。它使用圖形化界面，非常類似於Scratch語言和StarLogo TNG用戶界面。用戶可以拖放圖形對象來創造一個運行在安卓系統上的應用，它就可以在許多手機設備上運行。



# 系統分析

- 利用arduino IDE使用C++撰寫code來編輯設計自走車的如何行徑，之後燒錄進esp32,esp32控制TB6612FNG來控制自走車的移動,然後TB6612FNG驅動自走車馬達，使它可以移動，再利用esp32中的傳統藍芽和行動裝置（手機）中的藍芽做通訊。而利用APPINVENTOR2設計APP控制自走車。



# 製作過程和問題

- 材料中有兩種馬達驅動模組馬達控制板LD293和TB6612FNG，在測試中都有啟用成功，但由於體積TB6612FNG比較輕巧而且我也只需要用到兩個輪子驅動車子，故選擇TB6612FNG比較適合。驅動板測試成功後，再進一步組裝完成自走車。

# 程式碼

- arduino
  - #define AIN1 8 控制輸入A1
  - #define AIN2 7 控制輸入A2
  - #define BIN1 10 控制輸入B1
  - #define BIN2 9 控制輸入B2
  - #define PWMA 6
  - #define PWMB 5
  - #define STBY 11 「待機」控制接Arduino的11腳
1. const int offsetA = 1; 正反轉設定A，可能值為1或-1。
  2. const int offsetB = 1; 正反轉設定B，可能值為1或-1。
  3. Motor motor1 = Motor(AIN1, AIN2, PWMA, offsetA, STBY);
  4. Motor motor2 = Motor(BIN1, BIN2, PWMB, offsetB, STBY);

- void setup() {
- 這裡沒有程式碼
- }
- 
- void loop() {
- motor1.drive(255,1000); 驅動馬達1全速正轉1秒
- motor1.drive(-255,1000); 驅動馬達1全速反轉1秒
- motor1.brake(); 停止馬達1
- delay(1000);
- 
- motor2.drive(255,1000); 驅動馬達2全速正轉1秒
- motor2.drive(-255,1000); 驅動馬達2全速反轉1秒
- motor2.brake(); 停止馬達2
- delay(1000);

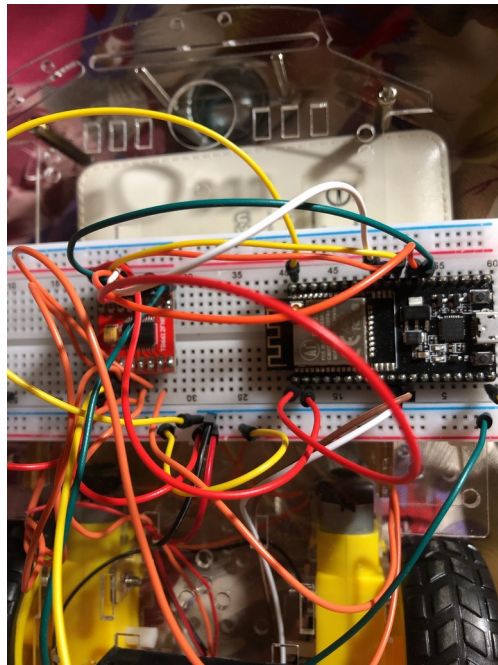


# 程式碼

```
→ forward(motor1, motor2, 150); 驅動雙馬達半速前進
→ delay(1000);          持續1秒
→
→ back(motor1, motor2, -150); 驅動雙馬達半速前進
→ delay(1000);
→
→ brake(motor1, motor2);  停止兩個馬達
→ delay(1000);
→
→ left(motor1, motor2, 100); 左轉
→ delay(1000);
→ right(motor1, motor2, 100); 右轉
→ delay(1000);
→
→ brake(motor1, motor2);  停止兩個馬達
→ delay(1000);
→ }
```

# 製作過程和問題

- 原先是利用Arduino開發板去驅動馬達，後來由於需要用藍芽連線，所以改用了esp32，需要把程式碼和腳位都改掉
- `//esp32`
- `#define AIN1 16 // 控制輸入A1`
- `#define AIN2 4 // 控制輸入A2`
- `#define BIN1 5 // 控制輸入B1`
- `#define BIN2 17 // 控制輸入B2`
- `#define PWMA 26`
- `#define PWMB 25`
- `#define STBY 18 // 「待機」控制接esp32的11腳`



# 製作過程和問題

- esp32雖然可以用arduino IDE去編寫程式代碼，但是esp32中不支援AnalogWrite()這個函式，後來上網找到答案，可以用ledcWrite()代替，所以進入SparkFun\_TB6612.cpp檔裡面把AnalogWrite()函式通通改成ledcWrite()即可。

```
51 void Motor::fwd(int speed)
52 {
53     digitalWrite(In1, HIGH);
54     digitalWrite(In2, LOW);
55     analogWrite(PWM, speed);
56 }
57
58
59 void Motor::rev(int speed)
60 {
61     digitalWrite(In1, LOW);
62     digitalWrite(In2, HIGH);
63     analogWrite(PWM, speed);
64 }
65
66 void Motor::brake()
67 {
68     digitalWrite(In1, HIGH);
69     digitalWrite(In2, HIGH);
70     analogWrite(PWM, 0);
71 }
72
73 void Motor::standby()
74 {
75     digitalWrite(Standby, LOW);
```

第55,63,70  
列改成  
ledcWrite()

```
49 }
50
51 void Motor::fwd(int speed)
52 {
53     digitalWrite(In1, HIGH);
54     digitalWrite(In2, LOW);
55     ledcWrite(PWM, speed);
56 }
57
58
59 void Motor::rev(int speed)
60 {
61     digitalWrite(In1, LOW);
62     digitalWrite(In2, HIGH);
63     ledcWrite(PWM, speed);
64 }
65
66 void Motor::brake()
67 {
68     digitalWrite(In1, HIGH);
69     digitalWrite(In2, HIGH);
70     ledcWrite(PWM, 0);
71 }
72
73 void Motor::standby()
74 {
75     digitalWrite(Standby, LOW);
76 }
77
78 void forward(Motor motor1, Motor motor2, int speed)
79 {
```

# 製作過程和問題

- 連接Ultrasonic sensor超音波感測器測量距離，設計出避碰自走車

```
int trigPin = 26;      超音波感測器接收訊號腳
int echoPin = 27;      超音波感測器發出超聲波腳
long distance;         避碰距離
const int dangerThresh = 1740; 避碰距離30cm * 58
byte dir = 0;
副程式避碰感測
long ping(){
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    return pulseIn(echoPin, HIGH);
}
```

# 製作過程和問題

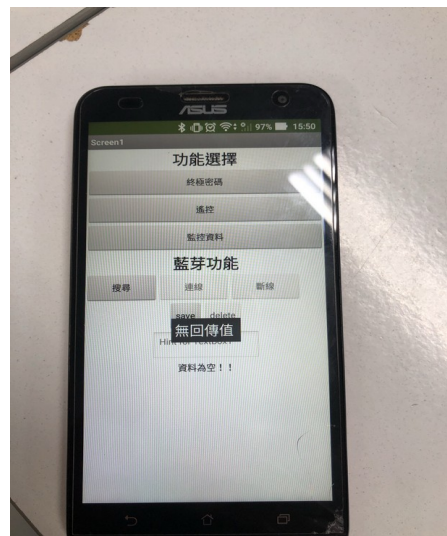
→ 避碰測距開始程式開始：

```
void setup() {  
    pinMode(trigPin, OUTPUT);    // 定義輸入及輸出  
    pinMode(echoPin, INPUT);  
    Serial.begin (9600);  
}
```

```
void loop(){  
    distance = ping();  
    Serial.print("Distance : ");  
    Serial.println(distance);  
    if (distance > dangerThresh){  
        if ( dir != 0){  
            dir = 0;  
            brake(motor1, motor2);  
            delay(500);  
        }  
        forward(motor1, motor2, 100);  
    }else{  
        if (dir != 1 ){  
            dir = 1;  
            brake(motor1, motor2);  
            delay(500);  
        }  
        right(motor1, motor2, 500);  
    }  
    delay(1000);  
}
```

# 製作過程和問題

- ➔ 自走車功能完成，再來是測試esp32藍芽和手機的藍芽連線功能。這裡採用傳統藍芽而不是低功耗藍芽BLE，因為傳統藍芽連接方式較為簡易。
- ➔ 在Appinventor上先設計功能頁面再來就是藍芽傳輸頁面做簡易測試





# 製作過程和問題

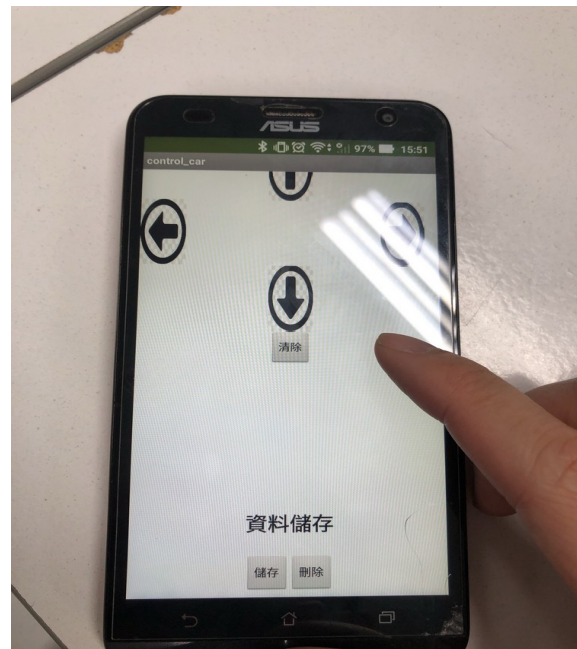
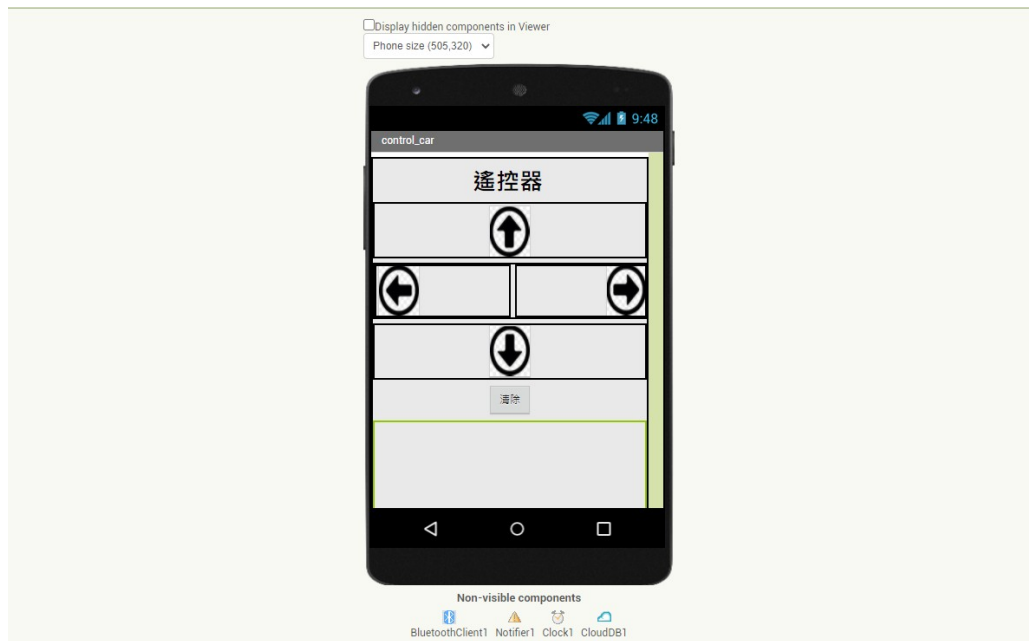
1. 改進方式：上網找尋資料後，參考(<http://drho.club/2018/06/arduino-bluetooth-read-sensor-value/>)，方法為用字串陣列一個字元一個字元傳輸。

```
if (cmd[0]=='a') {  
    digitalWrite(LED, HIGH);  
} else if (cmd[0]=='b') {  
    digitalWrite(LED, LOW);  
} else if (cmd[0]=='c') {  
    float h = dht.readHumidity();  
    float t = dht.readTemperature();  
    char hum_buf[10], tem_buf[10];  
    if (!isnan(h) && !isnan(t)) {  
        dtostrf(h, 3, 2, hum_buf);  
        dtostrf(t, 3, 2, tem_buf);  
        for(int i=0; i<5; i++) {  
            BT.write(hum_buf[i]);  
        }  
        for(int i=0; i<5; i++) {  
            BT.write(tem_buf[i]);  
        }  
    }  
    cmd[0]='d';  
}
```



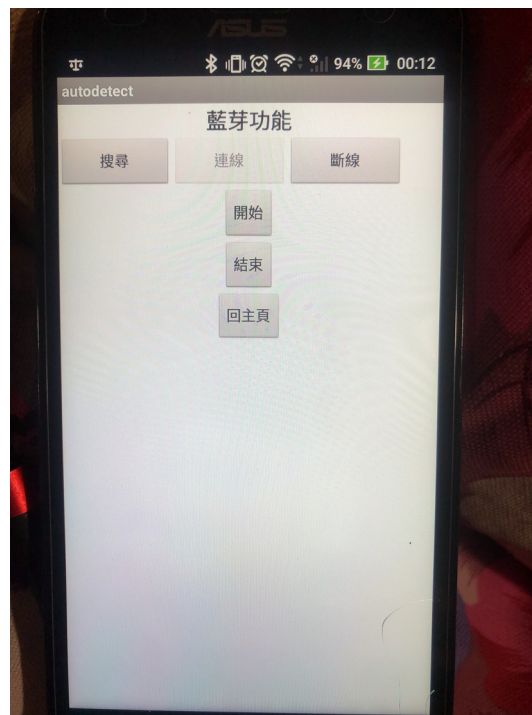
# 製作過程和問題

- 傳輸部份沒問題後，再來就是設計出可以切換功能的頁面和頁面的設計，遙控器，資料庫顯示和自動避碰模式
- 遙控頁面：



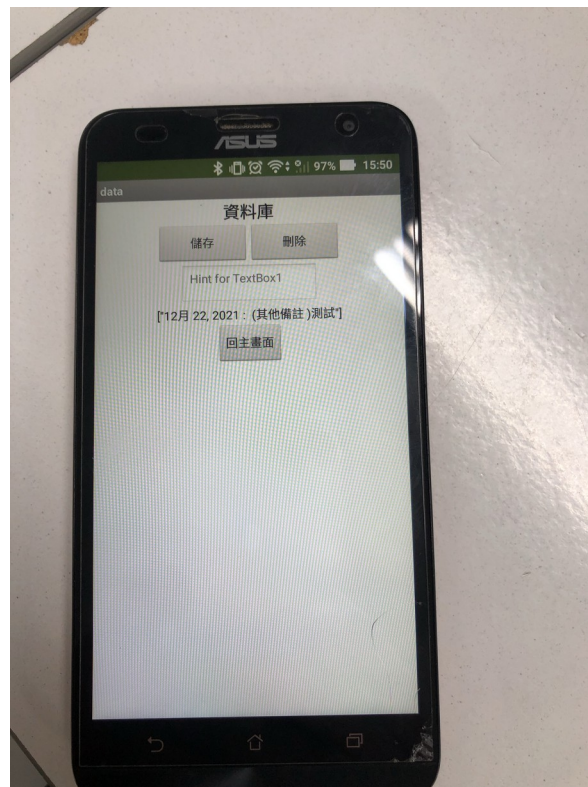
# 製作過程和問題

自走避碰頁面：



# 製作過程和問題

→ 資料庫頁面：



# 製作過程和問題

- ➔ 資料庫部份用的是Appinventor2上面所提供的clouddb,此資料庫是根據專案上面得id建立不同的資料庫，所以不會有重複覆蓋的問題，且簡易的使用鍵-值對(Key-value pair)建立出不同的資料列表

# SWOT分析

S優勢：可以利用上課所學的知識和不同得感測器做整合呈現一個多功能的平台

W劣勢：感測器整合和通訊技術有待加強

O機會：隨著5G技術發展，物聯網也著受惠，相關產品需求增加

T威脅：越來越多人才相繼投入相關研究，競爭力相對增加

# SWOT 策略方案

## SO攻擊策略：

持續加入多種感測器裝置做更強大的功能

## WO補強策略：

改善目前感測器的整合技術，連網通訊技術，讓使用者有更加的體驗

## ST防禦策略：

調查目前消費者所需要的感測裝置並且學習想辦法整合在裝置平台上，更貼近消費者

## WT迴避策略：

太困難或目前不太受歡迎的感測技術不會優先考慮在整合裝置上

# 心得

- 在此專案過程中，由於自身一個人獨自製作，構想，所以時間必須要有良好的掌控，在此專案中，進度大致按照該規劃時間內完成。
- 由於當初計畫設想不夠周全縝密，在途中發現不太適合專案理念，但是立刻做調整和改進所以沒有拖延到進度，但是之後在開始規劃上須做更仔細的思考。
- 途中一直有在思考是否添加更多的感測裝置能和系統做整合，能否用更複雜的BLE低功耗藍芽做更複雜的傳出，甚至是否能用BLE做出MESH網路連接方式，但是過程中心中已經先有把基本原先規劃的東西做出來後再添加新的或是更複雜功能的想法，雖然最後沒有添加到新的感測裝置和更複雜的通訊方式，但是至少有按照進度完成基本該完成得，相信未來可以在花時間去陸續添加和完成。
- 在這過程中透過自己查詢資料，詢問師長等方式學會了很多課堂上沒有學習到的東西，過程雖然艱辛，些功能雖然也沒完成，但是已經克服而完成的部份真的很有成就感。

# 資料來源

- <https://swf.com.tw/?p=1066>
- <https://atceiling.blogspot.com/2020/12/arduino100tb6612fngn20.html>
- <https://jimirobot.tw/arduino-tutorial-ultrasonic-sensor-501/>
- <https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/>
- <https://zh.wikipedia.org/wiki/ESP32>
- <https://youyouyou.pixnet.net/blog/post/119410732>
- <https://zh.wikipedia.org/wiki/%E9%9B%86%E6%88%90%E5%BC%80%E5%8F%91%E7%8E%AF%E5%A2%83>
- <https://zh.wikipedia.org/wiki/C%2B%2B>
- <https://zh.wikipedia.org/wiki/MIT%E5%BA%94%E7%94%A8%E5%BC%80%E5%8F%91%E8%80%85>