智慧掃地機器人

組員：劉明宏、林偉晟、何駿延、江宗翰

第一章、簡介

對於目前市面上所販售的掃地機器人，常常會因為Random的巡走導致卡在某個角落或是多次走過某個路徑導致大量耗能，因此我們針對此問題做發想，想透過建立地圖的方式來做更有效率的行走。
為了完成我們的目標有下列事項是我們需要達成的：

1.自走車硬體架構

一般正常行走、轉彎是基本的，搭配超音波sensor再交由Banana pi中央處理

2.行走演算法

根據sensor得到的數據，決定一般正常時的巡走演算法，邊走邊紀錄map資訊，以及未偵測到的路徑採用另外一個演算法來解決

第二章、硬體架構

一、整體架構

1.四輪自走車一台
2.嵌入式開發板Banana Pi 一片 (Cortex-A7 Dual core 1.0GHz)
3.Ultrasonic sensor 4個
4.馬達驅動板 2片
5.麵包板，杜邦線數個
6.小米行動電源1個

二、超音波模組原理

以GPIO寫入觸發訊號和收取回傳訊號，觸發端觸發後會產生8個40kHz超聲波，當超聲波碰到障礙物時會反射回接收端，時間差即距離的兩倍。此模組已有自動的溫度校正，故直接以一般聲波速度340m/s來計算即可，前面左右的模組可偵測左前方或者右前方障礙物，來左轉右轉，後方配置是因為左右轉後，靠牆那側再一次轉彎時需要判斷距離，因此放置在後方左右各一個，共四個。

三、開發板選用

Banana Pi有著比Raspberry Pi更豐富的接口，雖然號稱相容但其實還是很多要再稍加修改，效能上因為不像Raspberry Pi的bandwidth問題，可結合多個應用於一身，且價格便宜，因此選用Banana Pi當作此計畫主角。

四、馬達驅動板

原來搭載在自走車上的是使用L298N，體積較大，且網路傳言指稱這模組常是拆件後焊接的假新品，所以穩定性會相對較差，因此選用L298P，體積小了很多，使用上跟接腳定義與L298N雷同，以PWM輸入給EN跟Digital signal輸入給 IN1、IN2、IN3、IN4來控制直流馬達的動作:

ENA	IN1	IN2	功能
HIGH	HIGH	LOW	馬達正轉
HIGH	LOW	HIGH	馬達反轉
HIGH	IN1=IN2	IN1=IN2	馬達快速停止
LOW	ignored	ignored	馬達慢速停止

五、自走車

此四輪自走車為4個直流馬達為驅動核心，壓克力板為主體，可在上面搭載各式板子或者sensor，因為為直流馬達，所以需要馬達驅動板來達到控制速度、前進後退、左右轉。
預設的Arduino控制函式:
MoveForward()-當前方無障礙物時，車子前進
MoveBackward()-當車子進入凹槽卡住時，車子倒退
TurnLeft()-當右前方有障礙物時，車子左轉
TurnRight()-當左前方有障礙物時，車子右轉
Stop()-車子停止

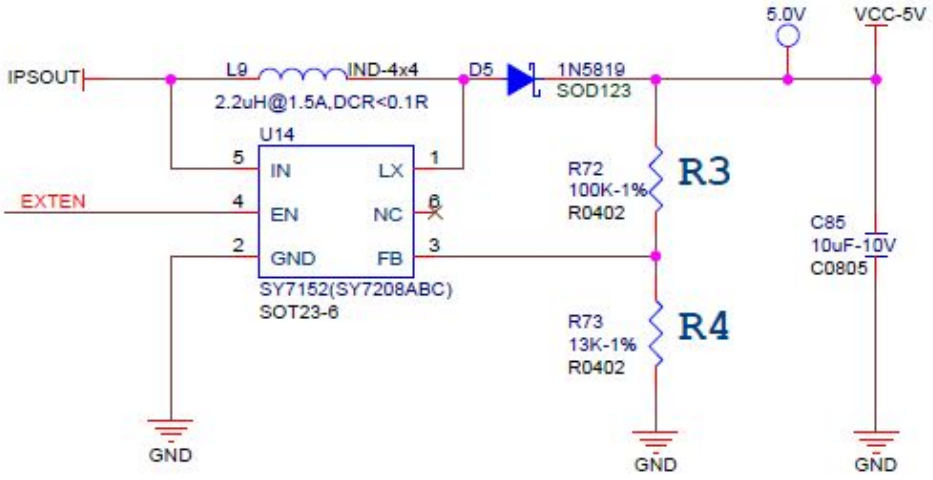
六、電源

本實作是使用5V輸出的小米行動電源，體積小且重量輕，但因為馬達驅動可達到12V的輸入電壓，可預期扭力可能稍嫌不足，會造成速度降低且影響左右轉，之後可能尋求更適合的雙電源5V+12V當作主電源供應。

七、電流問題

因Banana Pi 有著過電流保護電路，因此電流過載時會自動切電，但因設計的電流值已經高出保護電路的350mA電流，造成保護電路上的Shockley diode D5發熱冒煙並燒毀，因此替換了耐流較高的diode (=1A)後可正常工作。

VCC-5V



$$Vout = 0.6*(1+R3/R4)$$

Fig. 1 Banana Pi 電源端部分電路

八、接線

			P1				
<50mA	3V3		1	2		5V	
BCM GPIO00/02	SDA0/1	8	3	4		5V	
BCM GPIO01/03	SCL0/1	9	5	6		GND	
BCM GPIO04		7	7	8	15	TX	BCM GPIO14
	GND		9	10	16	RX	BCM GPIO15
BCM GPIO17		0	11	12	1	PWM0	BCM GPIO18
BCM GPIO27		2	13	14		GND	
BCM GPIO22		3	15	16	4		BCM GPIO23
<50mA	3v3		17	18	5		BCM GPIO24
BCM GPIO10	SPIMOSI	12	19	20		GND	
BCM GPIO9	SPIMOSO	13	21	22	6		BCM GPIO25
BCM GPIO11	SPI SCLK	14	23	24	10	SPI CE0 N	BCM GPIO08
	GND		25	26	11	SPI CE1 N	BCM GPIO07
			P5				
<50mA	3V3		2	1		5V	
BCM GPIO29	SCL0	18	4	3	17	SDA0	BCM GPIO28
BCM GPIO31		20	6	5	19		BCM GPIO30
	GND		8	7		GND	

Fig. 2 Bananapi GPIO in Wiring C 紅色框起處即GPIO x

內側1 外側2 腳位數起

接線:

超音波L

VCC-5V

TRIG-GPIO0

ECHO-GPIO1

GND-GND

GND-GND

超音波R

VCC-5V

TRIG-GPIO2

ECHO-GPIO3

GND-GND

GND-GND

超音波LL

VCC-5V
TRIG-GPIO4
ECHO-GPIO5
GND-GND
GND-GND

超音波RR
VCC-5V
TRIG-GPIO6
ECHO-GPIO7
GND-GND
GND-GND

馬達驅動下片:
IN1-GPIO8
IN2-GPIO9
IN3-GPIO10
IN4-GPIO11

馬達驅動上片
IN1-GPIO12
IN2-GPIO13
IN3-GPIO14
IN4-GPIO15

編譯位置:
/home/bananapi/WiringBP/test.c
/home/bananapi/WiringBP/test_thread.c

編譯:
gcc –Wall –o test test_thread.c –lwiringPi –lpthread

執行檔位置:
/home/bananapi/WiringBP/test
/home/bananapi/WiringBP/test.sh

九、GPIO控制

本實作使用Banana Pi第三方支援的GPIO函式庫WiringPi改成的，原來為供給Raspberry Pi使用，經官方和第三方改良成Banana Pi使用。

十、程式的執行

Banana Pi搭載了3.4的Linux Kernel和Lubuntu 14.04 LTS，利用開機後autologin和autostart執行程式test來進行實際的demo，source code已放在github上。

十一、成品

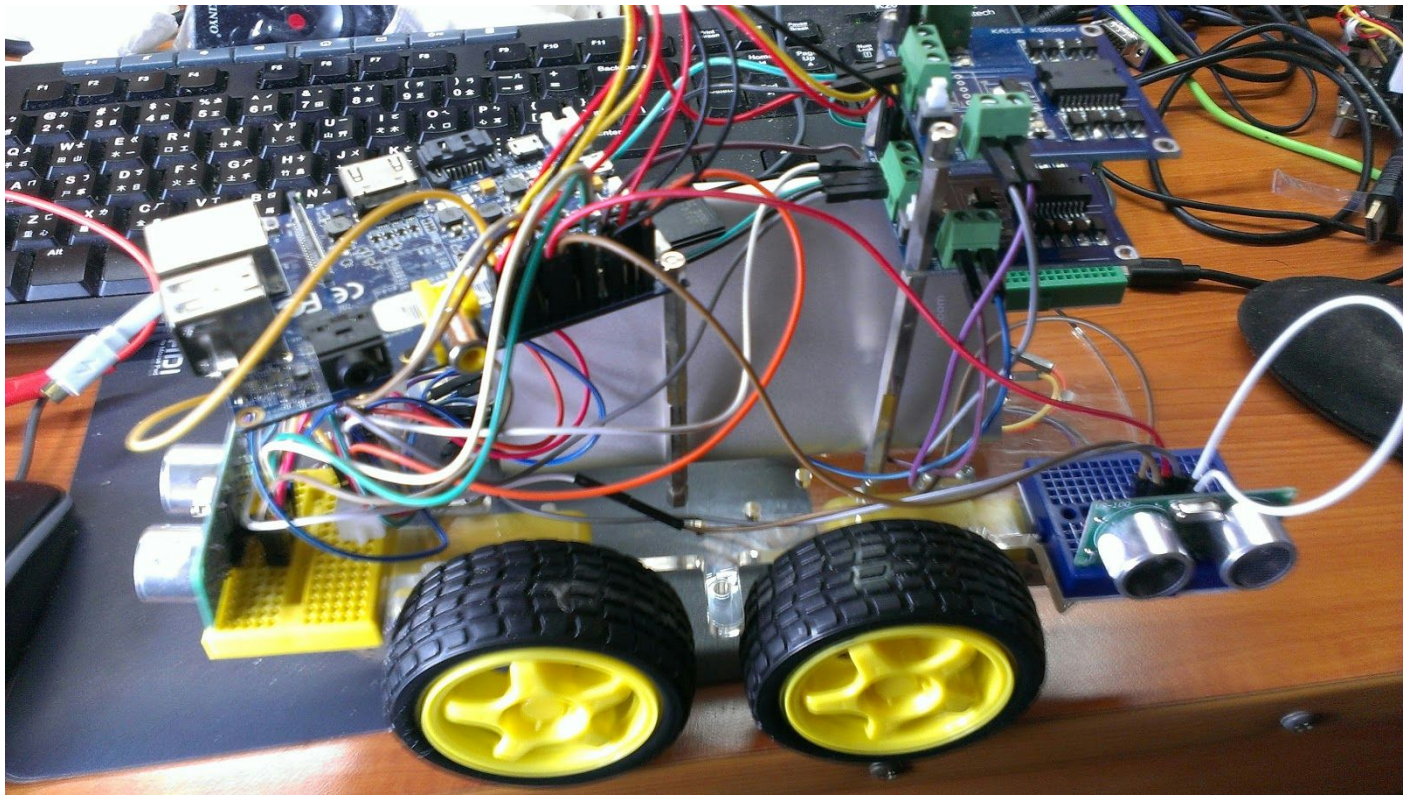
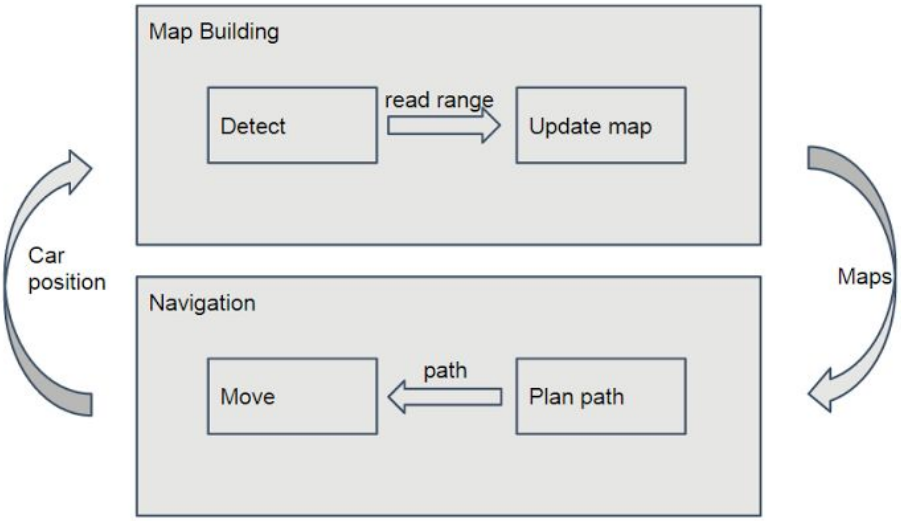


Fig. 3 成品

第三章、演算法

一、程式碼github

二、程式流程



detect.h
1.trigger()、pulseIn()、distance()三個function是用來計算出超音波所取得到的距離
2.detect()得到我們前方兩個以及左右四個超音波的距離

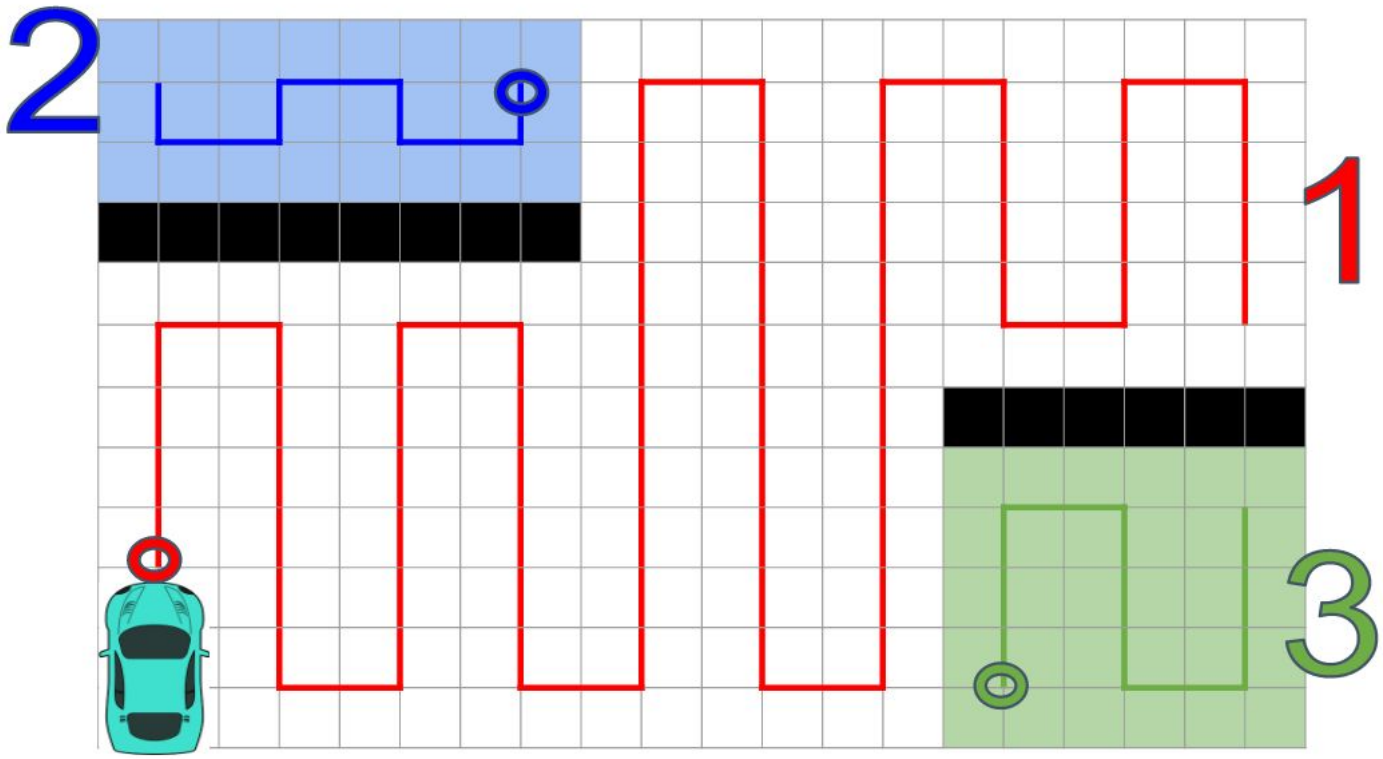
updatemap.h
1.先透過基本單位unit算出各個超音波的單位
2.再根據車子的方位來更新目前車子所在位置前後左右的狀態

planpath.h
由Scanning 演算法搭配Routing 演算法實作出。

move.h
根據規劃出的路徑，進行轉彎或是直走後退，並更新所在位置。

三、Scanning 演算法

如下圖所示，一開始我們根據地圖的不同以行為一個單位下去掃描一整個地圖，但因為一些障礙物路線會被阻擋，故將被阻擋的地方先記錄下來直到不能繼續掃描為止(如下圖1的位置)，再優先以最近未掃瞄過的區塊(如下圖2、3)作routing的演算法找尋最佳路徑。



四、Routing 演算法

Routing採用的是Hadlock's Algorithm，Hadlock演算法是在Lee演算法的變型，通過在目標的方向的搜索減小在膨脹階段的執行時間。不像Lee算法，它使用從源作為其代價函數的距離，Hadlock的算法使用稱為迂迴數值。從源到一個格點的路徑P的迂迴號D（P），反映了該路徑有"迂迴"遠離目標網格的數量。由於擴張算法選擇具有最低成本的擴展節點，此偏壓在目標的方向的搜索。當達到一個障礙繞道是必要的，標記有較高值的節點並將其擴展，而回溯階段與Lee演算法相同動作。如下圖所示，如果使用Hadlock's Algorithm則須往反方向6格的次數才可找到目的地。

		6	6	6	6							
	6	5										
6	5	4	3	2	1	0	0	0		6	T	
6	5	4	3	2	1	0	0	0		6	6	
6	5	4	3	2	1	S	0	0		6	6	
	6	5	4	3	2	1	1	1		6	6	
		6	5							6	6	
			6	6	6	6	6	6	6	6	6	

第四章、結論

一、成果

https://www.youtube.com/watch?v=aaJvCm3fb_8

如影片中所示，車子以演算法所描述的方式前進：朝一方向前進且來回掃蕩，但其轉彎能力乏善可陳，且在影片的最後，因轉彎過慢，而撞上門。

二、需改善之處

其實我們作品的瓶頸大多落在輪子驅動的問題，誠如先前提到電源供應只有5V的小米電源，造成輪子的動力低落。若驅動的問題改善，估計能連動的改善下列現有的情況：

- 尚未能走鼻挺的直線
- 轉彎不順暢
- 無法原地旋轉

在軟體的部分，我們期待能在新增以下功能：

- 地圖動態新增及新產生地圖的合併
- 將Hadlock's algorithm用於另一階段的地圖掃描

三、未來展望

如果有經費資助我們繼續進行計畫長達三個月的話，我們估計能做到以下功能：

- 車子驅動有改善的話，可繼續沿用四輪架構；改成兩輪驅動佐一輔助輪，可減少驅動電源的重量。
- 目前的地圖皆是靜態宣告，且宣告很多未使用的空間，須將用於地圖的記憶體優化。
- 目前的實作是遇到障礙物就會轉彎，那清理壁延和角落就是需再特別實作的部分。
- 我們需要一個準確的定位模型。可用超音波定位或在安插另一個定位系統，以修正誤差使走直線或轉彎的動作更準確。
- 期待能做出這產品的原型。