# Self Balancing Robot

電機四 b04901035 陳明宏 電機四 b04901153 劉維凱

## 動機：

一般四輪車為車子的縮版，雖然具備馬力較強移動速度較快等優點，但實際需要靈活操作技巧的工作，其操作技巧難易度遠遠遜色於兩輪車，且兩輪車續航力較高，走路起來也較像人類兩條腿的走路方式，因此我們這一組想實踐兩輪車的實際製作，但這堂課的重點顯然不是在硬體製作，因此我們買來現有的平衡車，去對他做平衡性能的優化，以及加上我們的遇到障礙物被擋住後，相機開啟偵測方向的功能。

## 作法：

### 1. 架構

- **Raspberry Pi**

  裝有USB port，使電池座能夠穩定地降壓穩流輸出5V3A的電流，SG90伺服馬達兩組，實現雲台雙軸鏡頭的轉動，以及HC-SR04超音波模組實現遇到障礙物停止移動告訴camera開始辨識的功能，PiCamera搭配opencv實現便是箭頭指示的功能。

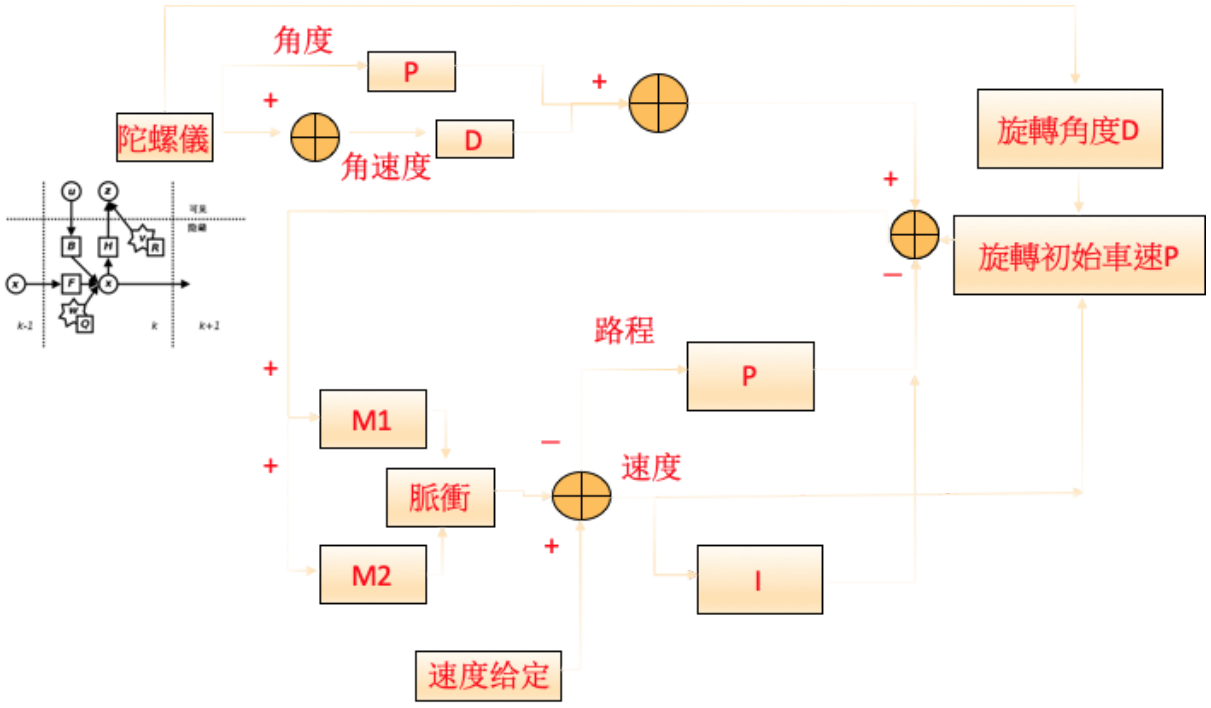
- **Arduino**

  裝有MPU6050，透過Kalman Filter的值使量出的加速度更精準，以及GB37帶測速馬達，和馬達驅動機TB6612FNG，使左右兩輪的值給的相同，如有速度偏差容易造成小車平衡不穩，容易翻倒，最重要的裝上藍芽接收器HC-06，實現和RPi溝通

- **車子架構**

| | 硬體實現 | | | | |
|---|---|---|---|---|---|
| RPi | 18650Bat x 3 | USB port x 1 | SG90 x 2 | HC-SR04 x 1 | Camera x 1 |
| Arduino | 18650Bat x 3 | MPU6050 x 1 | GB37 x 2 | TB66 x 1 | HC-06 x 1 |

### 2. 小車平衡




原理就是應用負反饋控制，由測量到的角度和自身平衡時的自然角度的差作為誤差，通過一個叫做PID的控制算法來控制電機轉速和轉向，偏離目標角度時，往前倒就向前跑一點，往後倒就向後跑一點，只要這個過程做的足夠快，參數合適，小車就能穩穩地站在原地。通過MPU6050檢測小車的角度作為PID 函數的輸入，設定一個平衡角度作為PID函數的目標值，然後把PID函數的輸出作為PWM值驅動電機。然而一般市售的平衡車其實不不穩定，仍然會出現小幅度顯而易見的晃動，因此我們希望在不修改硬體的情況下，實現軟體演算法的平衡，我們在陀螺儀的偵測上加上Kalman Filter，
它是一種自回歸濾波器，能夠從一系列的不完全及包含雜訊的測量中，估計動態系統的狀態。卡爾曼濾波會根據各測量量在不同時間下的值，考慮各時間下的聯合分布，再產生對未知變數的估計，因此會比只以單一測量為基礎的估計方式要準。加上這個濾波器後，再調上適合這個平地或斜坡的參數，我們可以實現看起來像不動般停在原地，不論在平地或斜坡上都可以，算是我們的contribution。

## 3. 超音波測距&伺服馬達

- **超音波模組**

這次我們是使用HC-SR04的超音波模組來測距，其精準範圍為1cm ~ 400cm左右，其主要碰到的問題為他的廣度沒有辦法涵蓋小車，但我們嘗試在車子的左右各安裝一個卻發現會互相干擾，只能以伺服馬達的左右掃瞄來確立車子能夠通過是比較麻煩的。再來由於rpi可能負荷太多外接模組導致電流有時候會不穩定，會出現異常的數值，因此我們還是有對超音波模組回傳的值做一個篩選、去掉極值及標準差比較大的值，此外，在電池電量較低的時候會直接失準，如果有偵測到直接失準程式要自己結束運行這些都是要被特別處理的部份。

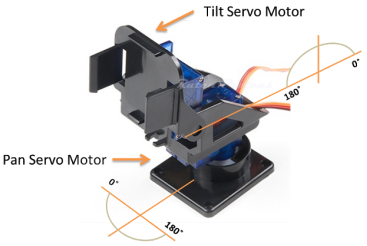
- **伺服馬達**

伺服馬達是用SG90，但RPI在使用伺服馬達會碰上一些問題。首先是控制能力，因為RPI的PWM並沒有特別的優化，因此效果是不甚滿意，在角度上面常常不是那麼精準，但這可以事先校正解決。再來是電壓，因為伺服馬達需要更高的電壓

5V，而RPI的電壓進來並沒有經過整流，所以穩定度很差，因此我們一開始直接用RPI是跑不起來伺服馬達的，後面接了電池座再接一個降壓穩流才有辦法控制。

- **雙軸雲台**

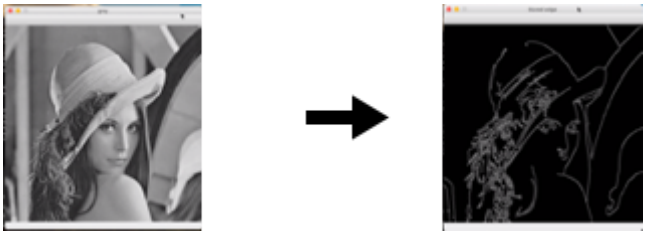
此外，我們用一個雙軸雲台接上兩個伺服馬達來達到可以讓超音波模組和PiCamera來轉向，藉以判別左右的距離和物品，我們之後會參考助教的意見，在辨別圖片時用上下掃描來增加更多的判斷依據以達到更高的正確率。


## 4.相機辨識箭頭 — openCV

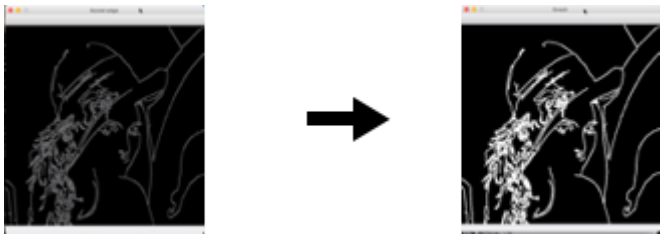
- **preprocessing**
  - 用灰階方式讀取

  - 高斯模糊、Canny邊緣檢測



透過用高斯分佈和圖像的矩陣做convolution來過濾掉圖片高頻的成分。
Canny用滯後閾值去尋找照片中亮度梯度較高的地方當作邊緣。

  - 設threshold轉換二值圖



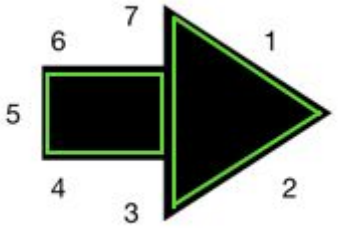
設定0~255之間的值，低於就設為0，高於就是255

- **描繪輪廓、逼近多邊形**

由於必須轉換為2值圖，而圖片又是pixel by pixel，我們看到的斜線實際上是還是由方形的格子拼起來的，因此在轉換的過程中會變為鋸齒狀，因此需要再去用演算法來逼近為多邊形。

- **判斷是否為箭頭**

在這裡我們用比較嚴苛的判斷，趨近多邊形之後要確定是一個七邊型的圖形，接著他們的角度會是一個矩形配上一個三角形才算成立。這部分我們做的判斷成功的機率還不高，可能的原因有在車子移動的時候或是停下來都會有晃動鏡頭，這樣讓照片會增添其他的不穩定因素。再者，由於判斷條件過為嚴苛，因此不論是距離太近、角度不對，只要有切到圖像造成不是七邊型的話就不可能判斷的出來。後續思考覺得可以試試看用ML的CNN model來解決這個問題，不過要再測量RPI上面load model來判斷的時間能不能做到即時，以及training data的取得等問題。




5. app實作

因為廠商有附上操控小車的app，但使用上非常不順，且在藍牙連接與切斷的時候會讓小車摔倒，但無奈廠商說並沒有source code可供參考，為了使小車平衡調參數的過程更加容易，於是我們利用上課學到的知識，自學寫了一個app，可以控制小車，給小車移動的參數，app分成前端與後端。

- **前端**

```
public class MainActivity extends Activity {
    private BluetoothAdapter bluetoothAdapter;
    private Switch bluetoothSwitch;
    private Button bluetoothSearch;
    private Spinner bluetoothList;
    private Button bluetoothConnect;
    private ArrayAdapter<String> adapter;
    private List<String> list = new ArrayList<~>();
    private String strMacAddress;
    private static boolean booleanConnect = false;
    /*方向按鈕定義*/
    private Button mButtonRun;
    private Button mButtonBack;
    private Button mButtonLeft;
    private Button mButtonRight;
    private Button mButtonStop;
    private Button mButtonLevo;
    private Button mButtonDextro;
    /*功能按鈕*/
    //msg 定義
    private static final int msgShowConnect = 1;
    /**************service 命令*********/
    static final int CMD_STOP_SERVICE = 0x01;       // Main -> service
    static final int CMD_SEND_DATA = 0x02;          // Main -> service
    static final int CMD_SYSTEM_EXIT =0x03;         // service -> Main
    static final int CMD_SHOW_TOAST =0x04;          // service -> Main
    static final int CMD_CONNECT_BLUETOOTH = 0x05;  // Main -> service
    static final int CMD_RECEIVE_DATA = 0x06;       // service -> Main

    MyReceiver receiver;
```


前端負責用android studio裡面的指令刻出按鈕的layout，寫在Mainactivity裡面，寫出前進後退左轉右轉左旋右旋停止七個鈕，以及偵測出藍牙時連接上，處理使用者互動，並實作出按下按鈕放大的功能，並將資料傳給後端。

- **後端**

```java
public class MyService extends Service{

    public boolean threadFlag = true;
    MyThread myThread;
    CommandReceiver cmdReceiver;

    /**************service 命令*********/
    static final int CMD_STOP_SERVICE = 0x01;       // Main -> service
    static final int CMD_SEND_DATA = 0x02;          // Main -> service
    static final int CMD_SYSTEM_EXIT =0x03;         // service -> Main
    static final int CMD_SHOW_TOAST =0x04;          // service -> Main
    static final int CMD_CONNECT_BLUETOOTH = 0x05;  // Main -> service
    static final int CMD_RECEIVE_DATA = 0x06;       //service -> Main

    private BluetoothAdapter mBluetoothAdapter = null;
    private BluetoothSocket btSocket = null;
    private OutputStream outStream = null;
    private InputStream  inStream = null;
    public  boolean bluetoothFlag  = true;
    private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
    private static String address = "20:16:05:25:07:95";
```

後端寫在Myservice裡面，和Mainactivity做溝通，使用BluetoothAdapter類能讓使用者建立一個BlueServerSocket，我們使用的是rfcomm傳輸協定，為了相容傳統序列阜的應用，先掃描可使用的Device，當成功建立連線後，兩端都持有BluetoothSocket，以流的方式通訊。由於讀寫操作都是阻塞呼叫，需要開一個thread來管理，不然會使傳輸塞車，其中要注意，掃描是很耗手機資源的過程，一但找到裝置，確保發出要求前要停止掃描，不然手機很容易crash。

問題解決:

1. rpi和arduino溝通問題

一開始我們想解決超音波判斷可以過，但車子因為太寬所以其實不能過的情形，解決的方法是arduino要將陀螺儀的訊號轉成前進速度，不動的時候要將這個訊號告知rpi，但實際上我們的arduino因為插上太多模組，所以限制我們傳輸的方式，I2C/USB/SPI的孔都被插滿，在已經將arduino插上擴充板的限制下，為了不將車體的面積再加大，維持我們車體的靈活度，我們選擇用藍牙與RPi溝通，但這樣的缺點是藍芽傳輸的過程中，車子須保持不斷的平衡，所以我們必須犧牲移動的速度，等arduino傳給rpi後，rpi才能再給指令，但這樣又面臨一個問題，如果每一個時間都去access她的速度，將會造成資源浪費，所以我們必須採取interrupt的方式，使用上課學的方法，做效率的溝通。

2. servo穩壓穩流問題

因為rpi的5V孔供電不穩，再加上是用插座供電的情況，造成我們的伺服馬達常常因為不夠穩壓穩流而不斷晃動，無法固定在一個固定的角度，有微小幅度的晃動，這點arduino就做得很好，所以思考雖然他只是一個小小的馬達，也要給他一個馬達驅動模組，後來我們去買了一個擴充板HAT來做額外供電，確保鏡頭不會晃動，才能辨識的更準。

3. 超音波測距精準度

因為我們一邊移動一邊測距，所以必須考慮移動速度，這樣量出來的值才會更準，後來我們先使超音波穩壓穩流，再調小車子移動的速度，再縮短藍牙測距的時間，並用filter的方式濾掉爆掉的值，實現精確的測量。

4. 擴充板HAT不會用

為了使馬達供電更穩定，消除抖動與晃動，我們加入了rpi專用的擴充板，不用處理傳輸溝通介面，但無奈時間關係，此次現場demo及影片中，無法實際使用，暫時還使用rpi的5V孔供電，未來會加入使用。

成果

https://www.youtube.com/watch?v=gLq52RHsBsU

參考資料

1.51單片機平衡小車
https://item.taobao.com/item.htm?spm=a211ha.10565794.0.0.3cd53ca9aoSh6g&id=42913336181
2.平衡車製作規格
http://wickedlabelectronics.com/self-balancing-robot-projects/
3.PID演算法
https://blog.csdn.net/jsgaobiao/article/details/50643037