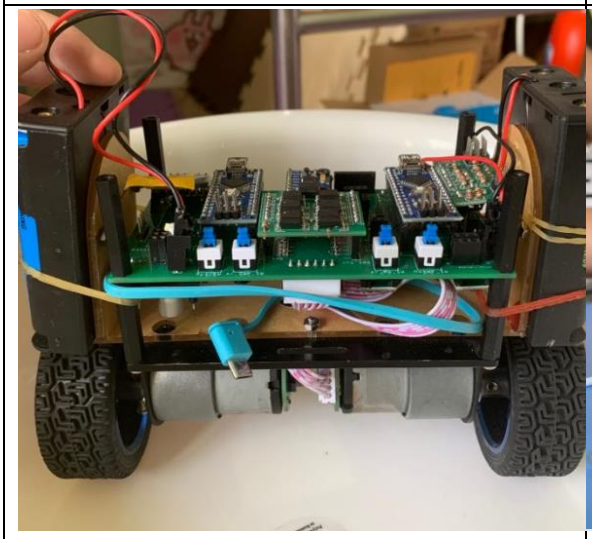
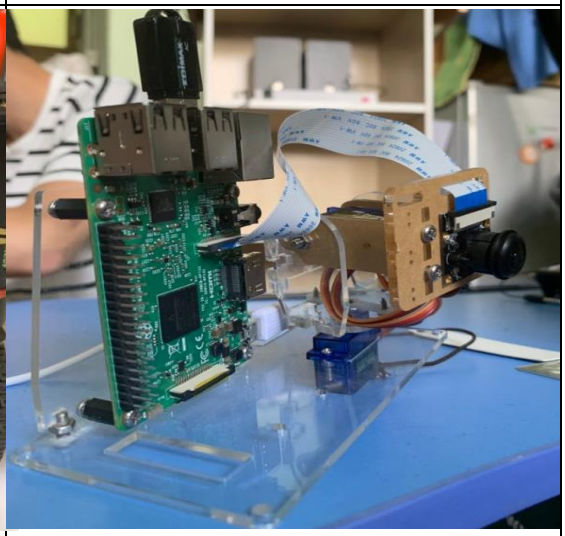
控制系統設計期末報告

第 7 組 0510794 王欽威 0510822 陳紀愷

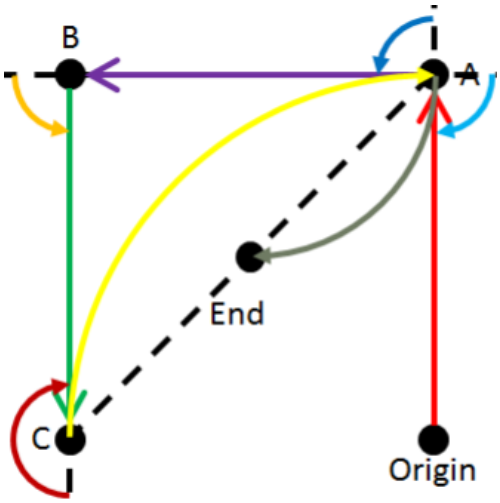
在之前的報告中我們已經成功組裝車體並可以完成位置控制與平衡，且已經組裝完 RPI 層，故在這裡不多加描述，以下報告我們將會分別描述三題我們怎麼因應不同要求來制定最適合的策略。

平衡車主體	RPI 層+鏡頭
	

↑上為平衡車和 RPI 外觀圖(尚未合體前)

一、 指定路徑行走

此題題目要求了車體的前進路徑，主要分成「直走、左(右)旋、弧形前進、停止」四個大部分，剩下就是彼此之間的排列，因此我們使用



計數器加上 switch 邏輯閘來指定車子現在的動作，完成目標後則讓計數器加一，以切換 switch 來指定下一個動作，已完成題目要求。

行走部分我們則分別使用了以下幾種方法：

1) 直行部分

主要是使用位置控制讓他前進 28 rad(約為 1 米長)，但是可能因為地形或是馬達上的差異而造成兩輪不等速，進而造成行走方向歪斜的現象，因此我們加入了

```
if(mode=='g'){
  if((desL-AL)>(desR-RL))
    powerR*=0.95;
  else
    powerL*=0.95;
```

上面判斷式，當左輪的剩餘距離＞右輪的剩餘距離，減少右輪電壓(速度)，反之則減少左輪電壓(速度)，使車體能直線前進而不歪斜。

2) 轉彎部分

轉彎的原理主要是讓兩輪移動方向相反來達到自旋的效果，因此我們加入右邊的判斷式，藉由加減不同輪之間的電壓來讓它旋轉，

```
else if(mode=='l'){
   powerR+=50;
   powerL-=50;
   if((AL-desL)+(desR-RL)<-7.7){
      movenext=true;
      movecmd++;
      mode=' ';
   }
}else if(mode=='r'){
   powerL+=40;
   powerR-=60;
   if((AL-desL)+(desR-RL)>7.7){
      movenext=true;
      movecmd++;
      mode=' ';
   }
}
```

直到兩輪編碼器和原先編碼器之差加起來為 7.7 rad 為止(約為 90 度)，因此我們可以很精準地控制轉角。

3) 弧形部分

弧形前進和直線前進很像，我們主要都是利用位置控制來讓他向前，只差在我們讓一輪電壓變大，而另一輪變小來讓他弧形前進，並用兩輪分別剩餘距離來調整避免車體過度歪斜。

```
else if(mode=='/'){
 powerL*=1.1;
 powerR*=0.9;

 if((desL-AL)*44./52>(desR-RL))
   powerR*=0.95;
 else
   powerL*=0.95;
```

4) 停止

位置控制距離為 0 來讓他可以控制在原處。

除此之外，因為賽道本身可能會有一些稍微上坡或下坡的部分，所以我們在不同階段也有給予車體不同的 reference 角，以抵銷上下坡造成的影響，來讓車體本身較為平穩。

```
if(movenext){
  switch(movecmd){
      case 0://1~2
        ref=3;
        cmd="28p";
        posCmd();
        break;
      case 1:
        ref=2;
        desL = -MotorL.GetAngle();
        desR = MotorR.GetAngle();
        mode='l';
        break;
      case 2://2~3
        ref=1.8;
        cmd="25p";
        posCmd();
        break;
      case 3:
        ref=2;
        desL = -MotorL.GetAngle();
        desR = MotorR.GetAngle();
        mode='l';
        break;
```

右圖為不同時候不同的 reference 角

二、 賽道競走

1) 移動部分

第二題我們則主要是用速度控制，也就是藉由改變他的傾角，來讓車體可以前進或後退，並加上第一題使用的左右轉，來遙控車子前後左右移動。

2) 遙控方式

一開始我們是用手機藍芽 + App Inventor 來和車子上的 Slave 端做連結，直接藉由手機一對一操控車子，但是 App Inventor 常常程式會有卡死的現象發生，一旦發生了就必須把車上的藍芽整個重開重來，很不方便，所以後來我們就改用另一個方法。

後來我們發現如果使用跟電腦連接的 arduino 和車子做連接的話，整體藍芽會穩定很多，也都不會斷線，因此我們決定用電腦跟藍芽做連接，並利用 arduino com port 監視序列阜傳送指令，但是鍵盤實在不太好操控，所以我們最後使用 PS4 搖桿和電腦做連結，再連結到車子，來達到遠端遙控的效果。

而傳輸指令的方式我們主要是透過傳遞不同字元來下不同指令，大概有以下幾種

‘w’ =傾角 + 0.15 rad（前進）

‘s’ =傾角 - 0.15 rad（後退）

‘a’ =左轉

‘d’ =右轉

‘+’ =傾角 = 基準點 + 1.5 rad

‘-’ =傾角 = 基準點 - 1.5 rad

‘*’ =傾角 = 基準點

藉由上面這些字元我們可以成功且順利的控制車子上山下海。

除此之外我們在這題也有特別調整了 PID 的各項參數，把 KP↑，KI↓、KD↑來讓車子遙控起來稍微軟一點但是更加靈敏。

三、影像辨識迷宮自走

判斷形狀方式:在拍照之後先做灰階再去抓出他的輪廓，之後再來判斷他的輪廓的頂點數目，如果他的數目<=3的話代表他為三角形，這時便需要左轉，

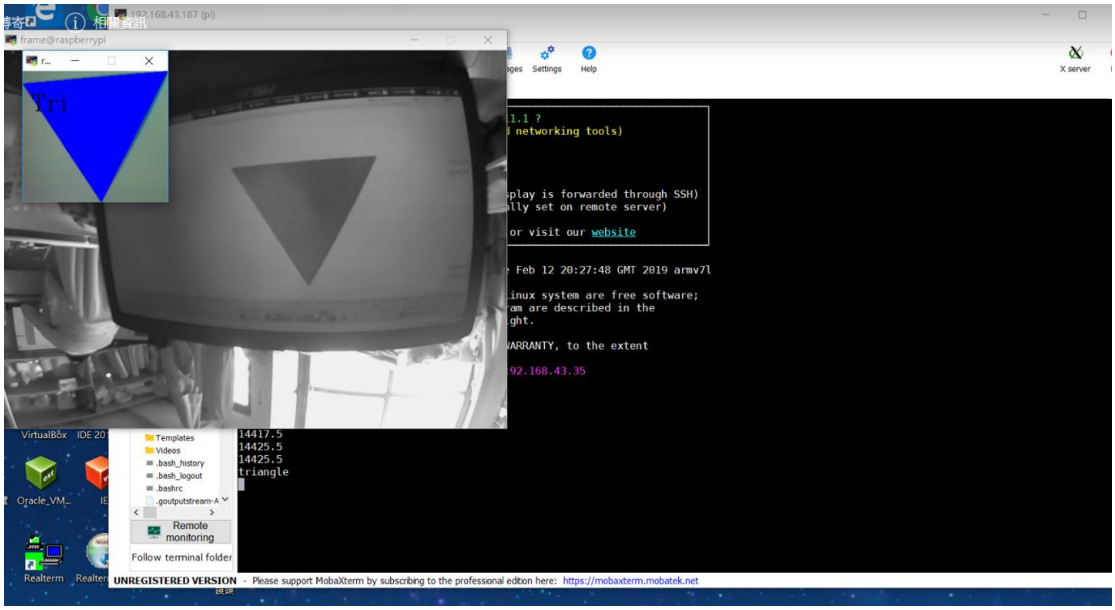
```
if len(approx) == 3:
    cv2.putText(img,"Tri",(x+10,y+10),font,1,(0))
    counttri=counttri+1
    countrec=0
elif 3<len(approx)<8 :
    cv2.putText(img,"Rec",(x+10,y+10),font,1,(0))
    countrec=countrec+1
    counttri=0
    countcir=0
else:
    cv2.putText(img,"Cir",(x+10,y+10),font,1,(0))
    countcir=countcir+1
    counttri=0
    countrec=0
```

所以利用 serial 傳送一個字元’l’給 arduino 端，如果他的頂點數目為 4~7 時(因為魚眼鏡頭會造成正方形的邊變成有弧度的而使判斷起來有可能會變成 4~7 個頂點)，便需要向右轉，這時則傳送’r’給 arduino，而大於 8 個頂點時則傳送’z’使其判斷原地停止斷電。

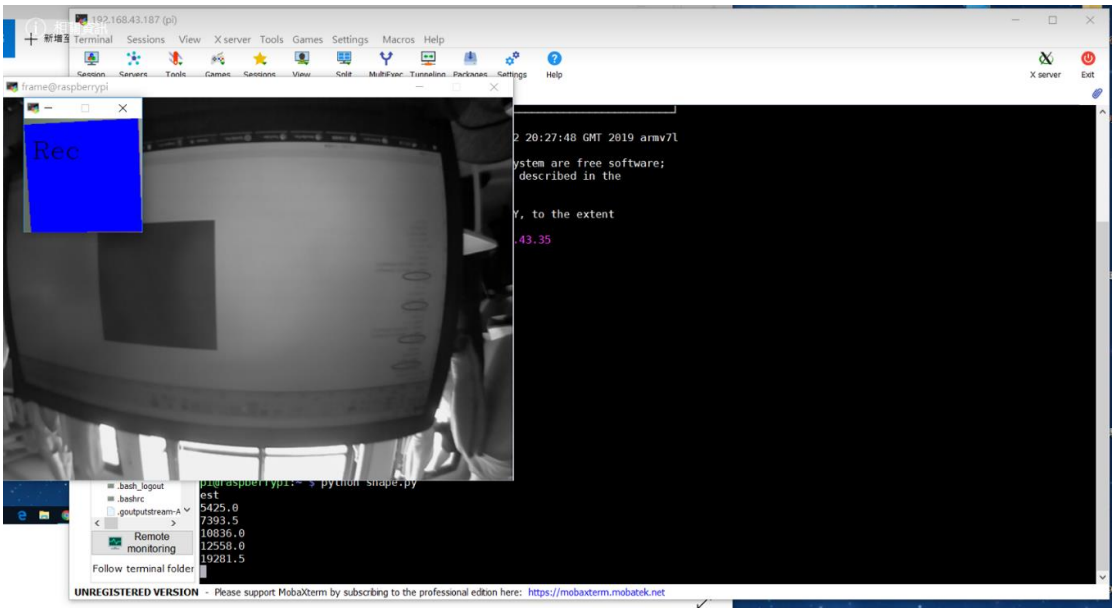
連續抓到五次該圖案(避免誤判)且面積大於各自的該有的值時(三角形較小另外兩個較大)傳送指令

```
if counttri>5 and n>12000 :
    counttri=0
    countrec=0
    countcir=0
    ser.write('l')
    time.sleep(3)
elif countrec>5 and n>30000:
    counttri=0
    countrec=0
    countcir=0
    ser.write('r')
    time.sleep(3)
elif countcir>5 and n>30000:
    counttri=0
    countrec=0
    countcir=0
    ser.write('z')
    time.sleep(1000)
    exit()
```

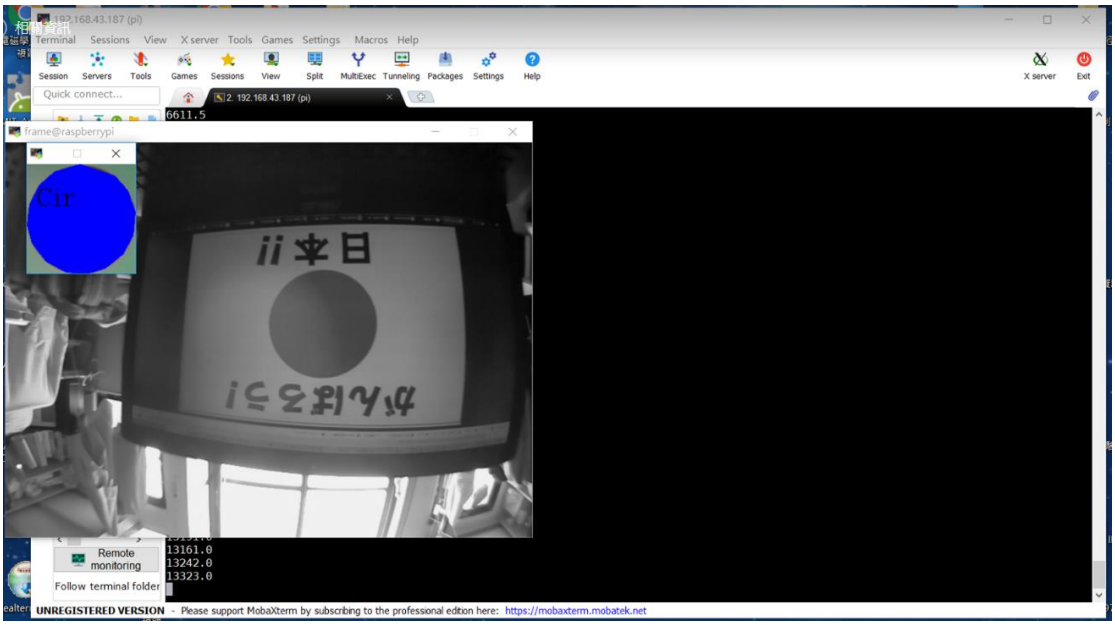
那我們並沒有做濾色的原因是因為我們判斷題目需求，在遇到三角形或正方形而旋轉之後無論如何都會直走直到遇到下一塊板子所以並不需要判斷出顏色便能完成題目。



此為抓到三角形後的畫面↑



此為抓到正方形後的畫面↑


Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
2. 192.168.43.187 (pi)
6611.5
frame@raspberrypi
Cir
13161.0
13242.0
13323.0
Remote monitoring
Follow terminal folder
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

此為抓到圓形後的畫面↑