控制系統設計\_期末報告

電機09 0510876 林信吉、0511094 門玉臻

**由於決賽前車子壞掉了，準備過程中也沒有錄影片紀錄，因此此份報告中沒有影片檔，不好意思了，謝謝助教體諒。**

**第一題競賽題目：指定路徑行走**

這一題的競賽目的主要是要看誰的雙輪車的位置控制誤差最小，使得雙輪車在依序走完指定路線後可以離終點距離最近。為了完成此題競賽我們主要改了三版，分別為：1. 使用move() function直接給馬達固定電壓、2. 使用兩組PID Controller控制雙輪車的平衡與車輪轉角、3. 只用一組PID控制器負責控雙輪車的平衡，再使用一個while loop透過車輪轉角計算車子移動距離。由於我們一開始設計時忽略了車子給馬達的電壓必須要包含車輪平衡的電壓與車子移動的電壓，因此若使用之前有家輔助輪時的移動function move() ，車子便會完全不平衡的向前衝。由於雙輪車的平衡方法是靠著偵測到車身傾角而給予馬達一定比例(由PID Controller計算出)的電壓來驅動，因此我們後來讓雙輪車可以邊平衡邊移動的方式就是讓車子誤以為車身有傾斜，進而造成馬達會向傾斜的角度移動，而使得車子移動。

實驗過程中，我們一下子就可以達到讓車子穩定且準確的直線移動到固定距離，而沒有多久也可以讓車子旋轉，然而過程中卻一直遇到各種奇怪的狀況。一開始我們一旦讓車子旋轉，車子即使轉到給定的角度後仍然會不停的旋轉(但是那時的code應該已經是回歸到執行原地平衡)。為了解決車子一直不停旋轉的問題，我們問了其他組同學讓車子旋轉的方式主要為兩種：多加了一組控制車輪轉角的PID Controller或是直接給予兩邊車輪一個固定的電壓差。過程中我們試了好幾種不同的旋轉控制方法(其他組都成功控制了)但是都沒有解決問題，最後我們讓車子一旦旋轉完給定角度後就把所有計算中用到的資料都初始化，才暫時讓車子旋轉後直接回到平衡模式。

好不容易解決完雙輪車的旋轉問題後，我們的車子有一段時間可以順利地從點1走到點2再旋轉90度，結果有一次燒完code後車子便突然不平衡了而只會向前衝。我們重複檢查了好幾次程式修改的部分並且一步一步的trace每一個使用到的function後仍然查不到問題。後來我們將每一個會影響馬達電壓值的變數都用Serial.println()印出來後發現，有一個使用的變數會在初始化(我們設定初始化後的數值皆會變成0)後的輸出不等於0，造成雙輪車左右馬達的理論輸出電壓與實際輸出電壓值不相同(程式相當於int a=0; 結果馬上印出來的a值卻為一個亂數)。我們花了很久的時間一直找不到解決的辦法，結果後來我們把所有的應用程式都關閉重新啟動後，再燒了同一份code到車子裡，車子卻恢復邊平衡邊移動的理想狀態(我們還是不知道是哪裡出問題)。

為了減少車子移動時的誤差(也為了避免車子再度出問題)，每次平衡車執行完一個動作指令(直走、旋轉)我們都會把控制器的積分項初始化，將每個動作當成獨立的state、彼此不會互相影響。由於初賽時的場地不適完全水平的，因此我們也會依據上下坡給予車子不同的傾角，讓車子可以順利到達目標位置(由於我們的馬達扭力很強，因此當車子移動到目標距離的1/7時我們就開始讓車子的傾角像後，讓車子減速)。比賽前我們都可以順利的從第一個點到達第三個點，結果正式比賽時我們車子卻直接衝過第二點撞向牆壁(完全沒有要轉彎的跡象)。後來好像也有不少組發生相類似的問題，但是大家也都不知道到底是什麼造成這些問題的(因為還是有組別順利的完成比賽規定的6個點)。

**第二題競賽題目：賽道遙控競走**

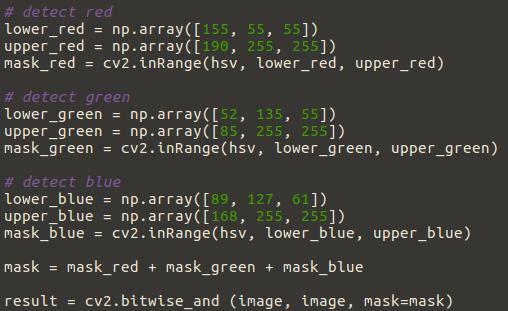
由於此題的比賽要用自己的遙控介面來控制雙輪車移動，因此我們後來決定使用這個學期最早寫的手機控制介面當作遙控器。相對於助教提供的全像輪控制器，我們的介面只有前、後、左、右和原地平衡等五個移動指令。由於在處理第一題問題時我們已經完成前進與左轉90度的指令，因此這題我們只需再將車子的移動方向相反就可以完成後退與向右轉的指令。為了讓我們可以更及時的一句車子行走的狀況進行控制，因此我們的控制概念是每當按完一個有箭頭的按鍵(前、後、左、右)後，若沒有按下平衡按鍵則車子便會持續執行該按鍵的指令動作。然而我們後來發現讓車子移動的參數卻與第一題移動時輸入的參數相差甚遠(第一題若要讓車子移動，我們只需要將車傾角減少0.01度就可以讓車子向前移動，但是第二題我們卻要把車子的目標車傾角設為至少-2.85度才可以讓車子向前移動一點點)。實驗中我們主要是沿用第一題的移動概念讓接收到手機藍芽指令後開始執行動作，但是不知道為什麼，我們的電池電壓對車子馬達的影響非常大。有時候甚至是相同電壓下進行相同的實驗所需要的車身傾斜參數也不相同。

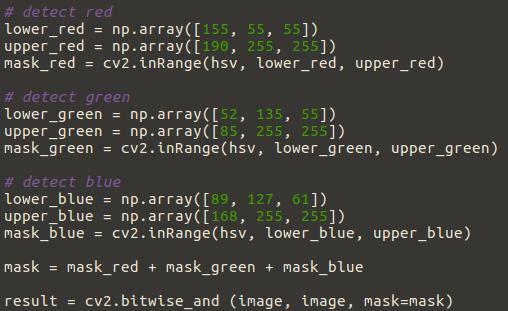
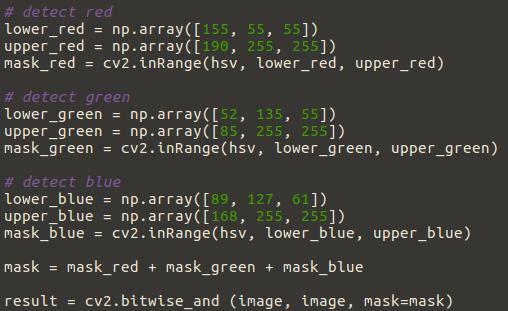
當車子可以大致順利的依照手機搖控介面執行指令後，我們有拿到初賽場地測試車子，結果發現用app inventor寫的遙控器與車子進行藍芽傳輸時不太能夠快速的下達指令。過程中我們發現只要每秒按下超過5次，車子的接收端就會出問題造成車子不再受控制而暴衝。為了解決這個狀況我們有試著改變車子接收端的timer interrupt時間，也試著每次按下不同按鍵時，程式便會自動將使用到的積分參數初始化為0，但是後來我們還是沒有用軟體的方式解決問題。最後我們是改變個人操作技巧，盡量減少每秒按下按鍵的次數，並且透過操作經驗盡量在車子暴衝前把他抓起來的方式減少撞到木塊的個數。

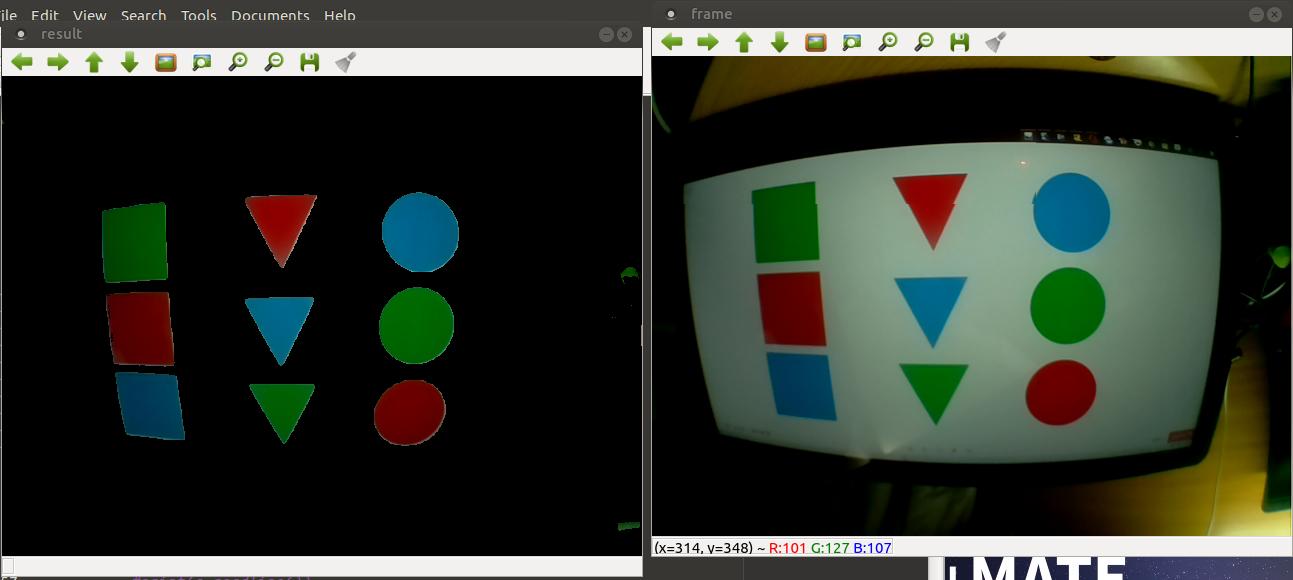
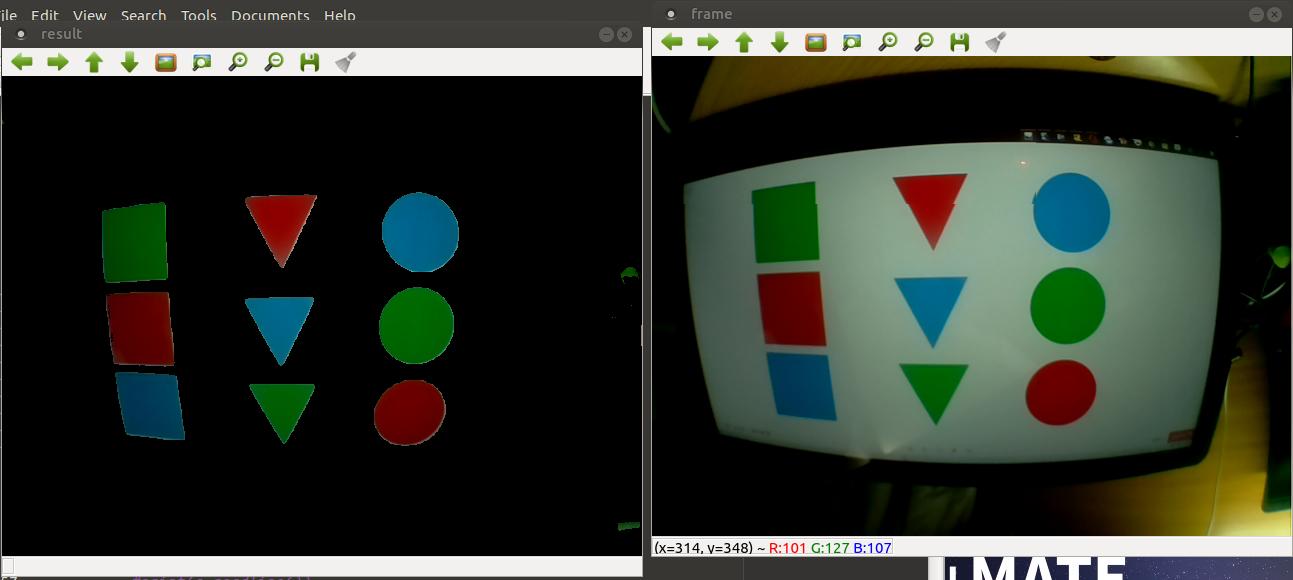
不太確定是什麼造成我們馬達輸出馬力較大的原因，但是實驗中我們發現上坡對我們來說一直都不是問題，我們這組的雙輪車即使是像決賽時使用的上坡高度也可以很輕易的跨越，但是反而是下坡減速對我們的車子是非常難控制的。我們有試著將車傾角加大非常多，但是卻成效不彰(車子一樣會像前衝，而沖完後在向後衝)。後來因為時間的關係，我們還是用人工的方式克服車子向前衝的問題。

**第三題競賽題目：影像辨識迷宮自走**

本項不再敘述平衡控制與位置控制之內容，將著重在影像辨識這部分。首先本次項目所要辨識的顏色有紅，綠，藍三種，所以我們分別先將三種顏色（mask\_red, mask\_green, mask\_blue）各自過濾出來，程式碼為下圖：

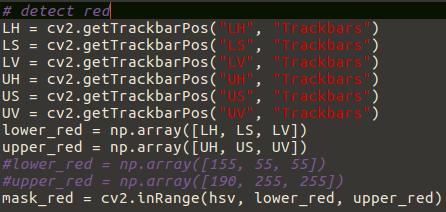
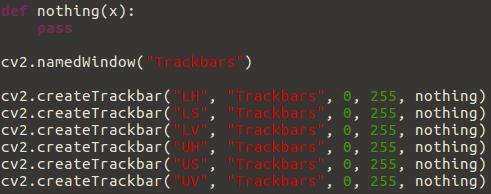


後來將三種過濾後的顏色合成在一起（mask），最後只顯示出過濾後剩下的三種顏色（result） 。呈現的結果如下圖，image為原圖，而result為過濾後得到的結果。

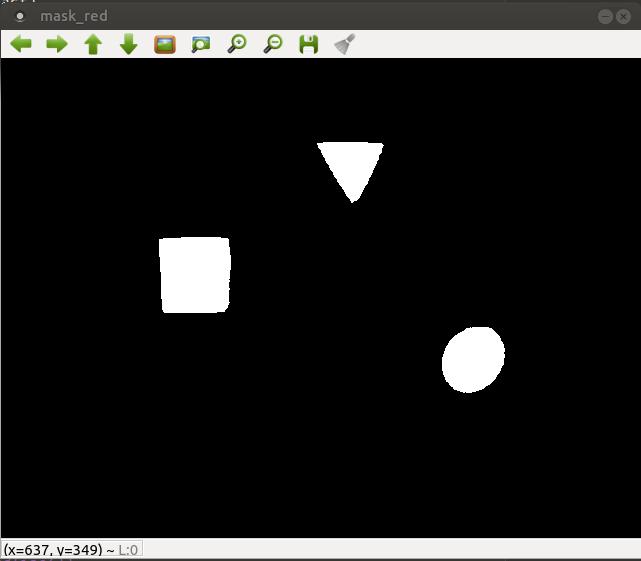
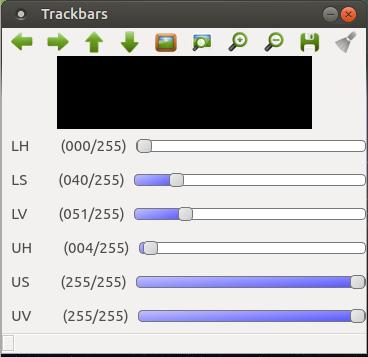
 

（image） （result）

為了讓可以更容易的過濾出我們要的三種顏色，我們使用了另一種工具：（圖1）的Trackbars，可以讓我們分別調整的影像的Lower Hue（色相LH），Lower Saturation（飽和度LS），Lower Value（明度LV），Upper Hue（UH），Upper Saturation（US）和Upper Value（UV）六個變數。這六個變數範圍都是0到255，使用的方式我們用mask\_red(如圖2）作為說明例子：getTrackbarPos是讓我們可以直接透過縮拉數值條後直接得到0至255的參數（圖3），後來個別輸入lower\_red和upper\_red，最後得到過濾後的圖為mask\_red（圖4）。可看到mask\_red所顯示的只剩下紅色的部分（判斷處理後為白色），其他不是紅色的則不顯示出來（處理過後為黑色）。我們有額外增加過濾工具是因為我們認為過濾顏色很重要，因為過濾好顏色的圖形拿去判斷形狀結果才會更加的準確(抓到的圖形邊界更清晰)。



（圖1） （圖2）

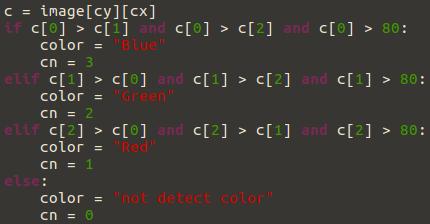
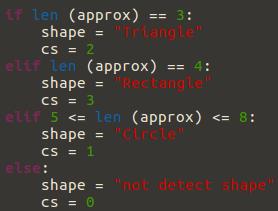


（圖3） （圖4）

以上說明的都完成後，接下來就可以開始辨識形狀，本次項目要辨別的形狀有三角形，正方形和圓形。我們用的方法如下圖：

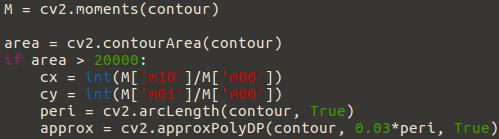
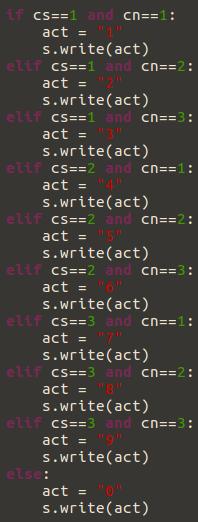


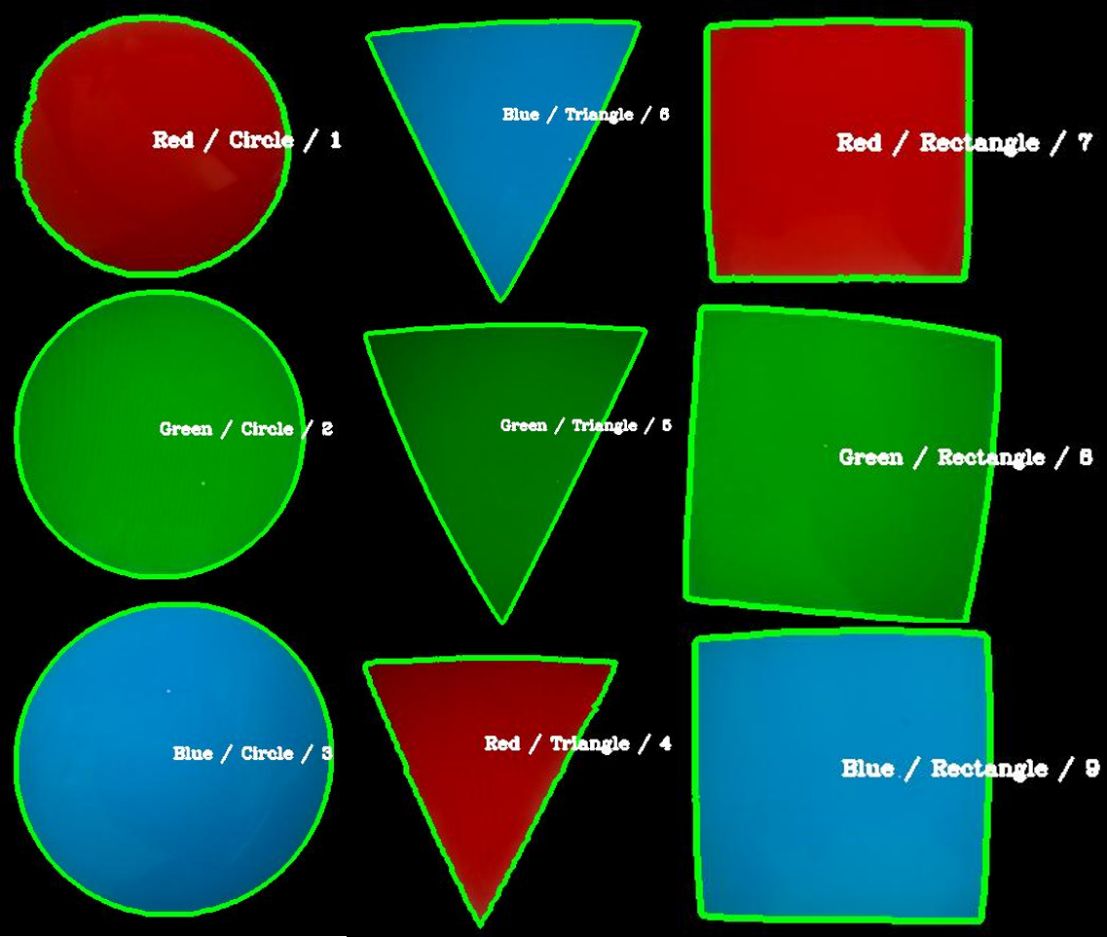
首先找出形狀的轮廓（contour），如果過濾好顏色所找出的轮廓就更準確，所以之前有提到過濾顏色是非常重要的。有了形狀的轮廓後，就可以知道它是幾邊型的形狀，例如三邊的就是三角形，四邊的就是正方形，其他就是圓形。因為我們過濾顏色沒有很好，所以在複雜的環境可能會辨識錯東西，而本次不用判斷五邊形和星形後，我們為了不要因為複雜的環境讓我們判斷錯形狀，做出錯誤的移動，所以可看到（圖5）我們將5到8邊形判斷為圓形，其他如果判斷到東西就設為沒有detect到形狀，這樣讓我們判斷更加的準確。判斷了形狀就要知道它是什麼顏色，Picamera所得到的影像有RGB的大小，從（圖6）看得知c[0]最大是藍色，c[1]最大是綠色和c[2]最大是紅色，其他就當做沒有detect到顏色。（圖6）的cx和cy是（圖7）中的目標物的中心點，而c是中心點的RGB大小。我們以這樣的方式判斷顏色非常的準確，而形狀經過我們不斷的調整準確度可以高達80%到90%。



（圖5） （圖6）

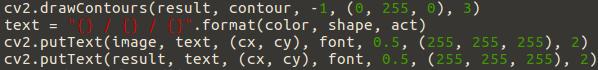
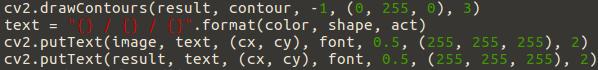
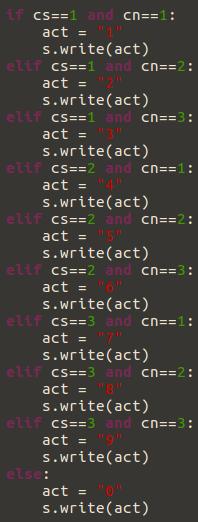
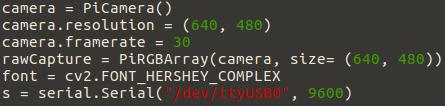
在程式中還算重要的是我們為了讓它可以到一定的距離才開始判斷與傳輸任務，所以我們設了一個area（圖7），當目標物在影像（640\*480）中的面積大於20000時，才開始判斷與傳輸，否則就前進直到偵測到目標物，進入平衡與偵測判斷傳輸下一個移動行為。



（圖7）

（圖8） （圖9）

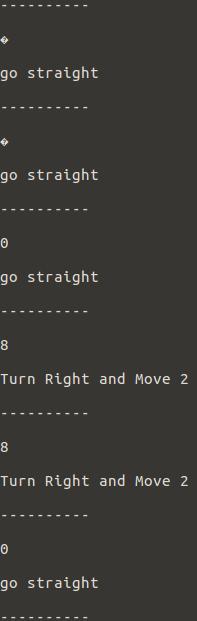
（圖8）判斷之結果和（圖9）是當完成判斷形狀和顏色後，Rpi將會傳輸給Nano做相對的移動行為，首先要設好Nano的Port和Baud rate ，後來當要將訊息傳給Nano可以使用 ，而讀取Nano的資料可以使用s.readline ()，會使用這個是為了要證明是否有傳對的訊息到Nano，否則可以不用使用。（圖8）中形狀的轮廓是以下的程式所畫出來的。這個也可以不用執行，而顯示出資訊的是以下的程式，我們將顏色，形狀與移動行為結合成text，後來使用putText，這樣可以顯示在影像中。

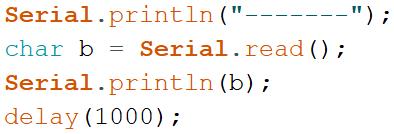


最後要顯示出影像可以使用。這些只是要證明判斷是否正確，確認後可以將以上這些可以不用的東西，這樣當在判斷時可以更加的順暢。以上就是我們處理影像辨識的方法與傳輸訊息的方式。



本項競賽我們順利進入複賽，在複賽前一天的中午我們已經完成了本項比賽，呈現結果我們覺得還算不錯的，直到硬體出現問題，原本Rpi與Nano的傳輸是正常的，不知道為什麼過後會傳送亂碼，導致無法執行該執行的移動行為。從右圖（圖13）可以看到我們從Rpi傳值給Nano應該是0至9的char，可是不知道為什麼硬體出現問題後傳輸資料也跟著出現問題，一開始硬體出現問題後，傳輸完全只有亂碼，弄了好久才偶爾出現正確的訊息，但因為正確的資訊只有幾秒而已，所以平衡車還沒完全轉彎，就已經讀到往前的指令了。硬體沒出現問題前，一切都是正常的，可是就是不知道為什麼馬達出現問題，傳輸就也有問題了。我是認為程式沒有問題，因為將資料傳給Nano就只可以使用s.write()和確認Nano讀到的值是否正確可以使用s.readline()這樣的程式是沒錯的。而且初賽我們也沒出現什麼問題，複賽前一天練習也沒問題，就是馬達出現問題，傳輸就不行了。過後我們用簡單的傳輸方法去測試看是不是一樣（圖10（Python）和圖11（arduino）），結果是一開始會傳亂碼（圖12），後面就一直傳2了。



C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\28.jpg  

（圖10） （圖11） （圖12） （圖13）

我們到複賽時和到寫結報都還沒解決出到底什麼問題，現在可以確定的是Nano和程式是沒有問題的，至於Rpi我是覺得沒有那麼容易壞吧，不過以後可以再找時間驗證一下。

<檔案名稱>

Balantbot資料夾: 車子arduino code

library資料夾: arduino的函式庫

detect.txt：比賽第三題(視覺影像處理)使用的python檔案

PID\_05\_18.aia: phone PID BTcontroller (app inventor)

**心得：**

林信吉：

平衡車競賽已經到一段落了，最後的成績對我們來說已經算不錯了，但還可以更好的，而我們的好運應該也在初賽就用完了吧。很感謝也很抱歉的是在平衡車上我幫不上隊友什麼忙，都是隊友在解決平衡車的問題，RPi的問題就交給我處理。一開始我們的辨識做好後，我要弄傳輸訊號時，程式中都需要import serial，但不知道為什麼我們的RPi import不到serial，所以我重新安裝過，那時候的我們已經覺得我們能做到那裡就做到那裡，我剛剛有說到初賽我們是幸運過關，是因為我們在初賽前，將第三項的競賽做了出來，所以就順利進入複賽，而第二項競賽，我們怎樣都沒想到我們可以進入複賽，所以這就是初賽的好運，到了複賽我們決定攻第三項，有時間才去弄第二項，在還沒吃晚餐前我們是多麼的高興啊，但不知為什麼馬達會出現問題，什麼都完蛋了。我用了整個晚上的時間，最後只弄出可以傳出幾秒是對的，但這還是不行的，因為沒時間解決了，所以沒辦法出賽，也只能放棄。最後我們希望可以在結報上拿多點的分數，這門課真的非常棒，但如果有更多的時間會更好，我們組保留車也希望過後可以解決出問題，從而得到寶貴的經驗。這學期真的要感謝隊友，也要向她說聲抱歉，我一直都幫不上她什麼忙，也只有越幫越忙，複賽第三項沒能拿下前三名真的很遺憾，這學期真的辛苦她了，遇到我這個雷隊友，感謝這學期的合作，謝謝。

門玉臻：

為了完成比賽各項項目，不管是初賽還是複賽大家基本上都是連續熬夜測試，沒有幾位同學有回去宿舍睡覺。雖然那幾天真的非常累，但是現在比賽都告一段落後回頭看也認為那幾天的回憶是很珍貴且有意義的。這是一門loading非常重的課，但是如果讓我再回到三下選課階段我還是會再選一次，因為透過這一個學期的訓練我認為自己除了更了解之前學到的自動控制理論在應用上的實質意義，更多的是學到了做實驗的習慣與態度。

雖然這是一門非常有意義的課程，但是其中也有一些我認為可以稍微改善避免造成比賽遺憾的部分。相對於其他組的同學，我們運氣很好的進入了決賽(有一組同學在初賽時遇到跟我們複賽時相類似的問題，結果他們最後好像完全沒有辦法比賽)，但是比賽前一天晚上12:00左右時我們的車子卻突然開始出問題。一開始是車子有一邊的馬達出現空轉的狀況，結果後來又發生我們初賽實驗時遇到的問題：完全相同的一份code卻跑出不相同的動作。我們換過了馬達、把code燒到別人的車子上驗證、把別人的code燒到我們的車子上等等各種能夠想到的解決或debug辦法我們都嘗試了，但是最後車子卻還是沒有恢復原狀。比賽當天早上6:00左右時，我們的車子又用樣的code成功平衡了，但是剩下的功能卻全部無法完成。

我認為我們第三題真的做的很不錯，但是後來遇到車子壞掉的問題使得我們只能棄賽看別組同學的車子執行指令，真的是滿難過的。希望之後若是再有相類似的課程助教可以盡量購買品質較好的原件，使得硬體壞掉的機率盡量降低(否則遇到了真的也很無奈)。雖然最後沒有順利出賽，但是我還是非常感謝助教們在一整個學期中的幫忙與協助。每次我們遇到問題時，助教們都非常熱心的幫我們四處找零件，或是直接來幫我們看code的問題。如果不是助教們的幫忙，我想我們的車子應該也沒有辦法順利的比完初賽進入決賽。真的很謝謝助教們的幫助，也辛苦助教們了。