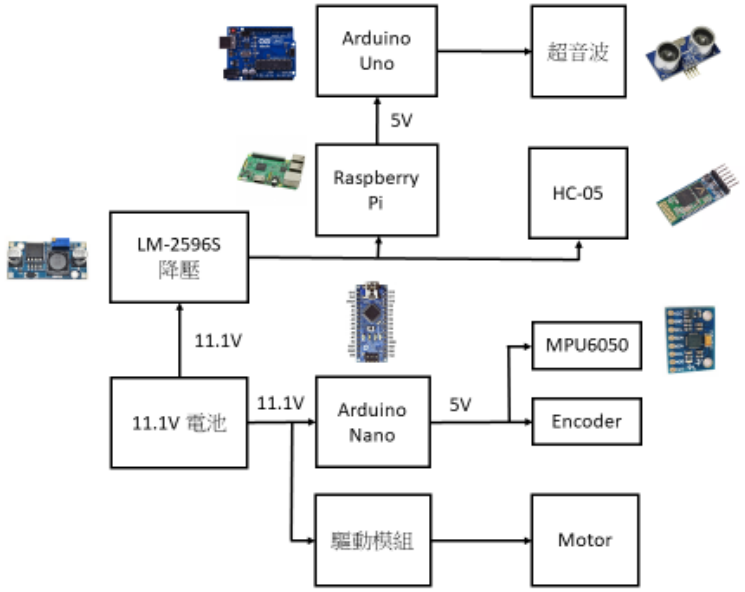
雙輪自走平衡車之設計與實現

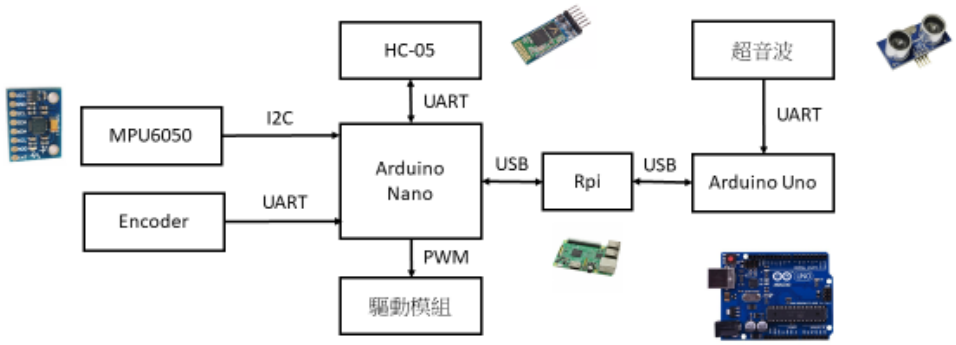
組別：14
組員：陳正恩 0310891
吳禹欣 0410854
指導助教：魏胤哲

一、系統架構示意圖

(一)電力架構

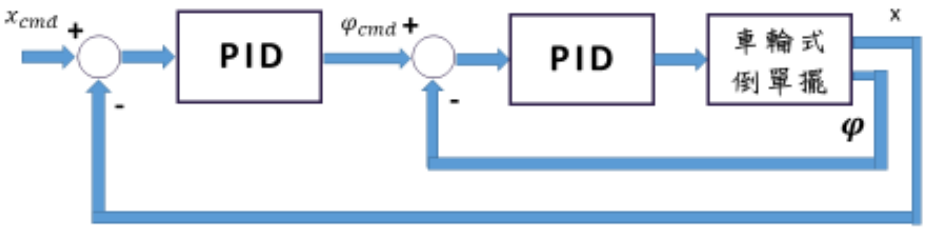


(二)訊號架構



二、控制架構

使用 PID 的控制器，並使用 cascade structure(位置)：



$$\begin{cases} X=(R-\varphi)*Gc*Gp_x \\ \phi=R*\dfrac{Gc*Gp_\varphi}{1+Gc*Gp_\varphi} \\ \dfrac{X}{R}=\dfrac{Gc*Gp_\varphi}{1+Gc*Gp_\varphi} \end{cases}$$

1. 平衡 (使用 PID 控制)

```
//參數
KA_P=12;
KA_I=120;
KA_D=0.5;
//PID
Pt = KA_P * Angle_Car; // Angle_Car:車傾角
It = KA_I * (Et_total);    //Et_total:車傾角積分
Dt = KA_D * Gyro_Car; // Et_total:車傾角微分
Et_total += Angle_Car*(micros()-angle_dt)*0.000001; //計算車傾角積分
angle_dt = micros();
pwm = int(Pt+It+Dt);
pwm_r = pwm; //右輪 pwm
pwm_l = pwm; //左輪 pwm
```

2. 位置控制 (使用 cascade PID)

```
//------------------車子位置 PID------------------
Speed_L = (encoderPosL - encoderL_past)/CPR*PI*2*TIRE_RADIUS/(millis()-
SpeedTimer)*1000; //cm/sec 計算左輪速度
Speed_R = (encoderPosR - encoderR_past)/CPR*PI*2*TIRE_RADIUS/(millis()-
SpeedTimer)*1000; //cm/sec 計算右輪速度
ErrorXR= xdR-encoderPosR*0.01/CPR*PI*2*TIRE_RADIUS;   //xdR: 右輪目標位置
ErrorXL= xdL-encoderPosL*0.01/CPR*PI*2*TIRE_RADIUS;   //xdL: 左輪目標位置
EpR_total += ErrorXR*(millis() - SpeedTimer)*0.001;
EpL_total += ErrorXL * (millis()-SpeedTimer)*0.001;
SpeedTimer = millis();
//參數
KP_P = 5.5;
KP_I = 1;
KP_D = 5;
Pp = KP_P * ErrorXR;
Dp = -KP_D *Speed_R*0.01;
Ip = KP_I * EpR_total;
tempR = Pp+Ip+Dp;
Pp = KP_P * ErrorXL;
Dp = -KP_D *Speed_L*0.01;
Ip = KP_I * EpL_total;
tempL = Pp+Ip+Dp;
//------------------車子平衡 PID------------------
//參數
KA_P=12;
KA_I=120;
KA_D=0.5;


Pt = KA_P * (Angle_Car-tempR);
It = KA_I * (EtR_total);
Dt = KA_D * (Gyro_Car-(tempR-templastR)/(micros()-angle_dt)*1000000);
EtR_total += (Angle_Car-tempR)*(micros()-angle_dt)*0.000001;
pwmR = Pt+It+Dt;

Pt = KA_P * (Angle_Car-tempL);
It = KA_I * (EtL_total);
Dt = KA_D * (Gyro_Car-(tempL-templastL)/(micros()-angle_dt)*1000000);
```

```
EtL_total += (Angle_Car-tempL)*(micros()-angle_dt)*0.000001;
pwmL = Pt+It+Dt;
angle_dt = micros();
encoderR_past = encoderPosR;
encoderL_past = encoderPosL;
templastR = tempR;
templastL = tempL;
```

3. 速度控制

```
//------------------車子速度 PID------------------
Speed_L = (encoderPosL - encoderL_past)/CPR*PI*2*TIRE_RADIUS/(millis()-
SpeedTimer)*1000; //cm/sec  計算左輪速度
Speed_R = (encoderPosR - encoderR_past)/CPR*PI*2*TIRE_RADIUS/(millis()-
SpeedTimer)*1000; //cm/sec 計算右輪速度
SpeedTimer = millis();
EpL_total+=Speedwant*( millis()-SpeedTimer)*0.001;
EpR_total+=Speedwant*( millis()-SpeedTimer)*0.001;
//參數
KP_P = 1;
KP_I = 0.2;
KP_D = 0;
Pp = KP_P * ( Speedwant-Speed_L*0.01); // Speedwant:  目標速度
Ip = KP_I * (EpL_total-encoderPosL*0.01/CPR*PI*2*TIRE_RADIUS);
Dp = KP_D * (Speed_L-SpeedlastL)*0.01/ (millis()-SpeedTimer)*1000;
SpeedlastL=Speed_L;
tempL=(Pp+Dp+Ip);
Pp = KP_P * (Speedwant-Speed_R*0.01);
Ip = KP_I * (EpR_total-encoderPosR*0.01/CPR*PI*2*TIRE_RADIUS);
Dp = KP_D * (Speed_R-SpeedlastR)*0.01/( millis()-SpeedTimer)*1000000;
SpeedlastR=Speed_R;
tempR=(Pp+Dp+Ip);
//------------------車子平衡 PID------------------
//參數
KA_P=12;
KA_I=120;
KA_D=0.5;
Pt = KA_P * (Angle_Car-tempR);
It = KA_I * (EtR_total);
```

Dt = KA_D * (Gyro_Car+(tempR-templastR)/(micros()-angle_dt)*1000000);

EtR_total += (Angle_Car-tempR)*(micros()-angle_dt)*0.000001;

pwmR = Pt+It+Dt;

Pt = KA_P * (Angle_Car-tempL);

It = KA_I * (EtL_total);

Dt = KA_D * (Gyro_Car+(tempL-templastL)/(micros()-angle_dt)*1000000);

EtL_total += (Angle_Car-tempL)*(micros()-angle_dt)*0.000001;

pwmL = Pt+It+Dt;

angle_dt = micros();

encoderR_past = encoderPosR;

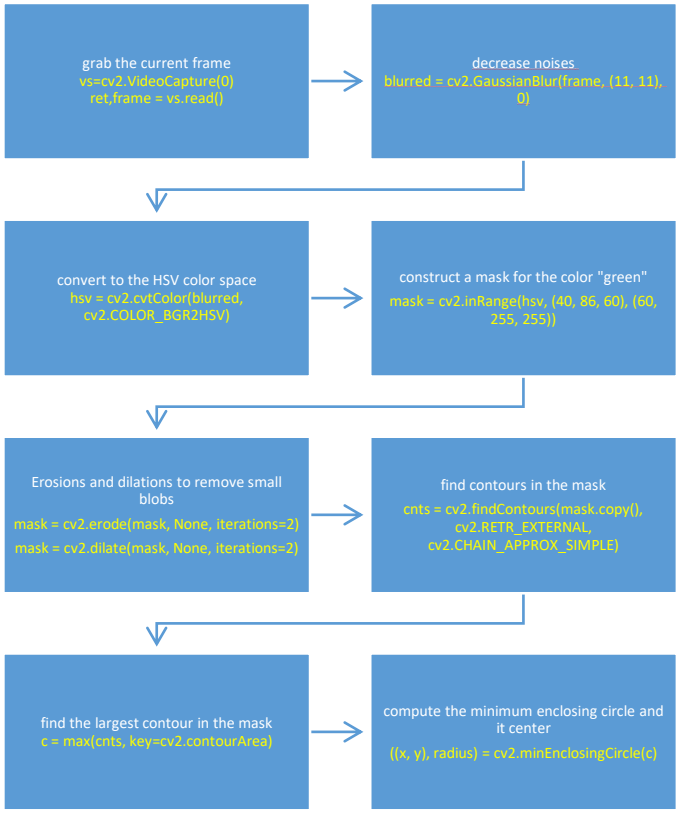
encoderL_past = encoderPosL;

templastR = tempR;

templastL = tempL;

(參考文獻置於報告之末)

三、Rpi 影像處理



If radius increase, the car go backward;

else if decrease, the car go forward.

If x increase, the car turn right;

else if x decrease, the car turn left.

四、SCAMPER 創意發想

【SCAMPER 資料庫】

方法/說明	設計架構中對應的素材
替代(S) 何物可被取代？	將A代替成B
	可以讓相機架的部分改造成為有一個簡單可以用來盛裝東西的勺子，讓平衡車有一個類似機械手臂的組織
合併(C) 和某物合併成為一體？	將A和B合起來，且能看到兩個原本的性質
	由手機軟體作操控，讓車子處於被動行進的狀態；又或者使用影像辨識作偵測，讓車子在不受操控的情形下仍能自行運作
調適(A)(微調) 原物可否有需要調整的地方？	進行變動
	為了讓自走車在運作上更顯為方便，可以利用程式將車子所走過的路徑記錄下來，以便日後可以重複執行先前所移動的路徑
修改(M)(大幅更動) 改變原物的某些特質或意義？	大幅更動
	有時候認為車子倒下去便是一個不好的表現，但其實可以利用這個部分使得車子可以有更加便利的移動，藉著多加一個滾輪，讓車子可以以水平方式行走
其他用途(P) 除了原功能外，是否能開發其他用途？	不同領域應用
	可以用於運輸的用途，如同物流車在進行工廠內物品的運送上，其實路徑大多是制式化、變化不太的，因此可以利用車子倒下時，將物品撈進所加裝的鏟子中，當車子起身時便可以做物品的運送，既方便又簡易
消除(E) 將原物變小？濃縮？或省略某些部分？	刪除某一部分
	原本有想說機器人的手臂一定要是多軸的手臂，但其實沒有此必要性，也徒增麻煩，其實只要向類似鏟土機的模式，便可做物品的拾取
重排(R) 重新安排原物的排序？或把相對的位置對調？	調動前後順序
	在進行路徑的紀錄時，先不要做實際的操作，而是先用電腦做模擬，測試看看是否能夠順利於出發點與目的地之間往返，否則紀錄了一個錯誤的路線，是沒有效率的

【SCAMPER 設計說明】

流程	說明
1.原設計要素	(1)具有平衡穩定功能 (2)車體是二輪自走車
2.定義問題	(1)穩定之精確度 (2)超音波模組於感應上之精確度
3.問題分析	(1)由於目前在自走車上的 Arduino 裝置之讀取裝置有限，因此在使用超音波測距時，無法同時執行位置控制之指令，因此目前先以人工輔助方式使車子移動 (2)由於希望能夠測得前後車之距離，如同「空氣量尺」般作距離上的量測，因此在車子之前後皆加裝了超音波模組，使得測距之功能發揮得更加多元
4.SCAMPER 使用	**替代（S）** Rpi相機→超音波模組 **合併（C）** 手動操控+自動偵測 **調適（A）** 需要時對使用者提出警示 **修改（M）** 避障感測 **其他用途** **消除（E）** **重排（R）**
5.產生新設計要素	(1)距離小於某依設定值，即會對使用者發出提醒 (2)利用雙超音波，可測量出前後總距離

【可行性分析】

（一）執行可行性(這個功能真的是需要的嗎？)

在現今之車輛，已經大多數都已經附帶有測距功能，像是在倒車時若接近一公尺時會有逼逼聲，若小於 50 公分時則變得越來越急促，若是真的要撞到後方車輛了，則會發出長聲的警示音。

（二）技術可行性(如果要實作這個功能，必須使用哪些資源？)

首先需要有測距功能較精準且可攜帶於車輛之超音波模組，且頭尾皆須加裝，然而此測距功能實際上僅為輔助性功能，因此應當具有良好的隱密性，不能對車體外觀帶來過大的影響，因此體積也不能太大。

（三）時程可行性(實作這個功能需要多少時間？)

由於不須事先作運算建立資料庫，因此只要適時的修正超音波測量之精準度即可。

（四）經濟可行性(這個功能花費項目有哪些？)

由於大多數為程式內部之作業，且除了功能加強版的超音波之外，其餘的器具實驗室皆拿的到，因此不需要太多的花費。

五、過程中遇到之問題

（一）藍芽連接問題

一開始在作車子初步的藍芽連接即出現問題，因為平衡車 Slave 端一直無法接收到來自電腦的控制，因此平衡程式一直無法順利執行

解決方法 拿去詢問助教，之後才發現是因為平衡車端的藍芽若是處於開啟的狀態，有機會會連接到其他電腦的藍芽，才導致無法順利連接，之後重新在壯志上進行 Master & Slave 之設定後便解決此問題

(二)動態方程式之推導

先前由於參考助教所提供之論文進行推導，但是發現到一些步驟時即遇到瓶頸，無法順利推導出正確之結果

解決方法 自行進行推導，當過程是根據自己的邏輯撰寫，便能夠很暢通的得到自己的方程式，同時很容易找到運算錯誤之處，也利於操作日後的簡化與分析，最後終於得到一組屬於我們自己推演出的結果

(三)PID 各組控制器之設定

當完成程式碼後，以為差不多就完成本次自走車專題，沒想到花費最多的時間其實是在此部分，常常只要環境改了，參數就跟著變更，而且 Kp、Ki、Kd 參數之間還會互相影響，因此設定的過程相當的麻煩

解決方法 先對各參數進行分析，明白各自對於車子之影響為何，因此可藉由觀察平衡車目前的運作情形，進行各參數之些微調整，有系統性與方向性的進行操作，便可得享倒吃甘蔗之福，最後也如期地得到最佳化數值

(四)Rpi 的影像處理

撰寫影像處理程式時，讀取相機影像資料的函式 cv2.VideoCapture()一直讀不到傳回 null

解決方法 請教 Google 大神，但是網路上眾說紛紜，有很多不同原因產生相同的錯誤結果，必須找出哪一個才是與自己的情況相符，關鍵字很重要，最後多方嘗試，找出了原因要先執行 sudo modprobe bcm2835-v412 相機的驅動程式

六、期末成果 demo 影片(詳見影片資料夾)

1. 位置控制(方向)
2. 位置控制(手機操作)
3. 速度控制
4. State-feedback
5. 指定路線關卡
6. 創意發想(防撞)→若是太過於靠近，銀幕會顯示出”danger!!!”字樣
7. 創意發想(測距)→可以測量平衡車兩端之距離

七、相關程式內容以及使用方式介紹

第一關:燒錄第一關 Arduino 程式到 nano

第二關:燒錄第二關 Arduino 程式到 nano，電腦用 ssh 遠端連線至 Rpi，執行 Rpi 上的控制程式，Rpi 與 nano 用 usb 傳輸 char 格式訊號

第三關: 燒錄第三關 Arduino 程式到 nano，電腦用 ssh 遠端連線至 Rpi，執行 Rpi 上的控制程式，Rpi 與 nano 用 usb 傳輸 char 格式訊號

八、參考文獻

(一) Christian Sundin and Filip Thorsten, Autonomous balancing robot Design and construction of a balancing robot, Göteborg, Sweden(2012)

(二) Balancing a Two Wheeled Robot(2009)

(三) Michael Baloh and Michael Parent, Modeling and Model Verification of an Intelligent Self-Balancing Two-WheeledVehicle for an Autonomous Urban Transportation System, France(2003)

(四)詹志元，以動態模型為基礎之雙輪機器人運動控制系統開發，國立台灣科技大學自動化及控制研究所碩士學位論文，民國九十八年

(五)李珈松，車輪式倒單擺平衡 PID 控制，國立台灣科技大學電機工程學系碩士學位論文，民國九十四年

(六)陳英傑，雙輪機器人之機構設計與平衡，國立台灣科技大學自動化及控制研究所碩士學位論文，民國九十七年