Bi-weekly Personal Journal

Name: 林士程　　Group: 第 26 組　　Role: 機電系統控制

Objectives

討論出優先度較高或確定要使用的機電系統元件，並大致規劃其用途。畫出上述元件的線路連接圖。

Summary of Outcomes

-Task 1:

目前初步規劃將使用 Arduino Uno R3 來當作微處理器，讀取 MPU-6050 量測到的三軸加速度值和角速度值，解碼出編碼器的 AB 相數位訊號，以推算出直流馬達的角位移量，並透過馬達驅動模組 L298N 控制其轉動。

-Task 2:

Arduino Uno R3 的主要用途，為透過 I2C 接口讀取 MPU-6050 的訊號，和輸出 PWM 訊號以控制馬達。

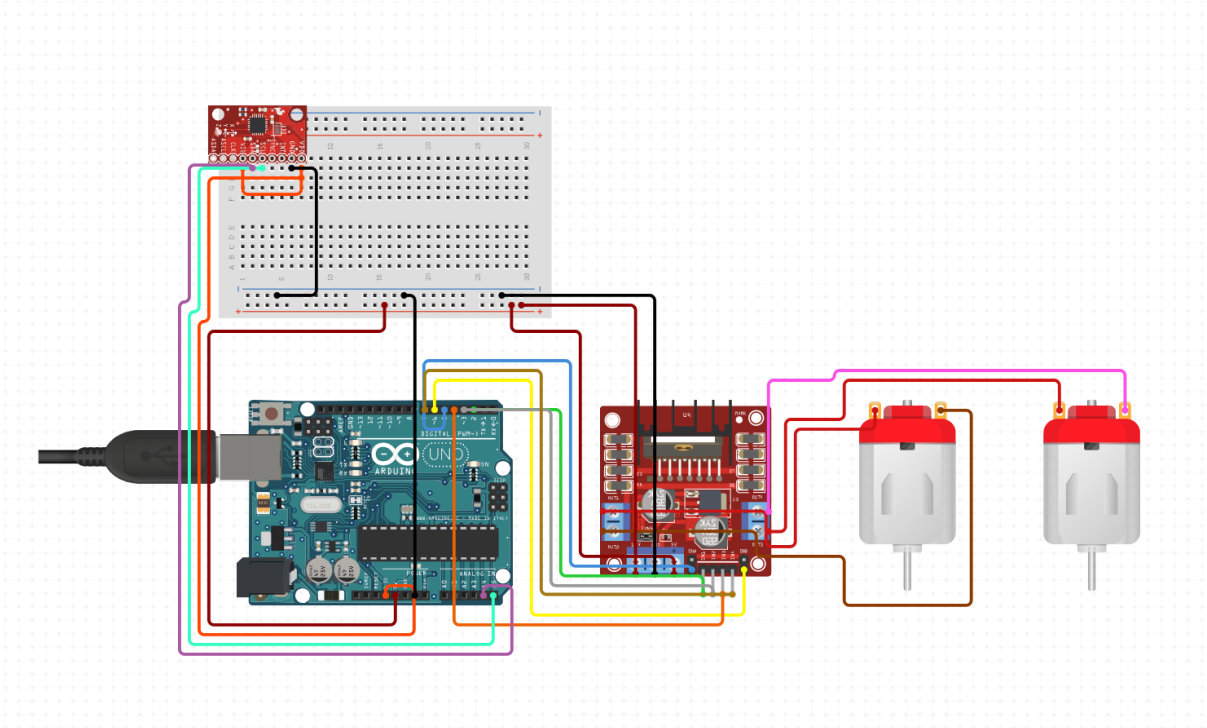
MPU-6050 其主要用途為當測量到的向下加速度高於一定值時，觸發風帆展開，次要用途為量測目前車體的三軸角度。

25GA-370 整合了直流馬達、齒輪箱和編碼器為一個模組以方便使用，其兩項主要用途為控制風帆的展開和調整風帆的轉向。

L298N 的主要用途為驅動 25GA-370 轉動，次要用途為提供微處理器供電，但這部分考慮到可能會因微處理器端阻抗不夠高而影響直流馬達的轉動，可能會另外加 buffer 提高阻抗或不使用其供電，直接獨立直流馬達和微處理器的電源。

-Task 3:

元件的線路圖大致如下圖，作為實際接線路的參考，此處微處理器未使用電池，因為之後測試時應該會直接用 usb 線供電和燒錄程式編譯碼。



Issues and Challenge

- Challenge 1:

目前只能大致預估風帆的重量和長寬，因而在挑選直流馬達時只能大概選用較大的扭矩量來避免扭矩不夠。

- Challenge 2:

目前還不確定要不要使用編碼輪，若使用的話可能可以完全取代 MPU-6050 的功能，這部分希望可以等聽過第五周的課再做考量，但目前認為用編碼輪有可以完整的確定車體在車道上的位置的優勢，這是加速度計相對無法做到的。

Perspectives

之前在社團就有稍微碰過編碼器和馬達相關了，但相對來說陀螺儀和加速度計就沒什麼經驗，這次是第一次使用，也藉由這次機會，了解了其腳位接法、供電電壓限制、原理和其局限性。

Plans for Next Week

- Perspective 1:

Challenge 1 應該還是要等到機構那邊稍微設計完，或是測量完帆和密集板的重量才能稍微估算出來，規劃上來說可能要到下幾周才能達成。

- Perspective 2:

Challenge 2 如上所述，希望聽完第五周的課後再來做決定，會在下一次的日記呈現。

- Perspective 3:

鑒於上述的問題可能都要等到第五周才能解決，我決定下周先在網路上找和寫一下之前沒碰過的 MPU-6050 的程式庫和程式。

Remarks

因為目前只決定用 MPU-6050，所以編碼輪的部分在上述沒有提很多，但這應該還是專題相關的，所以想在這邊提一下目前的想法，如果要確立車體在車道上的位置，至少要有兩顆分別量測 x 和 y 方向的位移量，但這樣一來就必須將其固定在一定方向，這必定會增加摩擦力使車體行進距離變短。但若能確立車體位置，並透過流場模擬軟體來決定風帆在哪裡轉向能得到最大的推力，感覺對車子行進應該還是利大於弊。

Objectives

搜索 MPU-6050 相關的程式庫，並利用其寫讀取三軸加速度值和角速度值的程式。

Summary of Outcomes

-Task 1:

目前找到了由 Adafruit 和 Jeff Rowberg 提供的 MPU-6050 相關程式庫。

-Task 2:

Adafruit 的可以由其提供的範例，可以用其程式庫的函式直接定義陀螺儀和加速度計的測量範圍，還可以選擇濾波器頻寬來濾除雜訊，並可直接讀取三軸加速度值和角速度值。

Jeff Rowberg 的就相對複雜，這部分我直接參考了網上的教學影片和 MPU-6050 的 DATASHEET，來學習如何透過 I2C 將它從 Sleepmode 喚醒，定義陀螺儀和加速度計的測量範圍，和取得三軸加速度值和角速度值。

Issues and Challenge

- Challenge 1:

這周花最久理解的部分都是理解別人寫的程式的部分，畢竟沒有一些先備知識，只有計算機程式學到的一些 c 語言基礎，所以一開始看到 Wire.write()內的 16 進制的代碼不知道是什麼意思。

- Challenge 2:

也是程式理解的部分，Wire.read()<<8 | Wire.read() 當中的 bitwise shift operator 和 bitwise or operator，要查才知道其運作原理。

Perspectives

學到 I2C 的通訊方法，能透過 DATASHEET 上提供的 register 16 進制代碼用 Wire.write() 寫入數據，改變 MPU-6050 的各項參數或讀值等功能。也學到了如何透過 bitwise operator 快速地將兩個 8bit 的值合併成一個 16bit 的值。

Plans for Next Week

- Perspective 1:

因為 MPU-6050 很容易受雜訊干擾，網路上是推薦用卡爾曼濾波算法來濾波，可以稍微研究看看，但若太過複雜可以考慮直接用 Adafruit 的寫好的濾波函式就好。

- Perspective 2:

下禮拜應該就會拿到 MPU-6050，將實際利用這周寫好的兩個程式測量三軸加速度值和角速度值，並且利用角加速度值計算三軸傾角。

- Perspective 3:

利用上學期量測原理的客參考網路上的教學影片寫的程式，寫出透過編碼器測量馬達的角位移量的程式。

Remarks

這周基本上都在處理跟程式相關的，想到剛好之前有碰過 Git 和 GitHub 一下，所以趁這個機會順便建了一個專門處理這次實務專題的 remote repository，來方便之後用筆電拿程式資料和測試，或許下次開會能跟組員分享一下，但感覺 google cloud 共編和 excel 還是蠻方便地，另外，也順便看了一下之前沒學的 branch 方面，結果看了之後的心得是好像用不到哈哈，畢竟程式量也不是那麼大，add、commit、push 和 pull 應該就很夠用了，也順便複習 Markdown 的語法來寫 README，雖然還是不太知道要寫啥，之後再說吧。GitHub 網址：rumsey000/Practice-of-Mechanical-Engineering (github.com)。

Objectives 實際利用這周寫好的程式測量三軸加速度、角速度和傾角值;寫好使用編碼器測量馬達角位移量的程式，並透過其值控制馬達。

Summary of Outcomes

-Task 1:

使用上週找到的由 Jeff Rowberg 開發的開源程式庫，來讀取 MPU-6050 的三軸加速度、角速度和傾角值，圖 1 為實際測試的資料畫面，直接以 serial monitor 呈現。

```
-> yaw : 0.42[°]  pitch : -0.09[°]  roll : -0.06[°]  Accel X : -0.00[g]  Accel Y : -0.00[g]  Accel Z : 1.00[g]  Gyro X : 0.02[dps]  Gyro Y : 0.02[dps]  Gyro Z : -0.01[dps]
-> yaw : 0.42[°]  pitch : -0.09[°]  roll : -0.06[°]  Accel X : -0.01[g]  Accel Y : -0.00[g]  Accel Z : 1.00[g]  Gyro X : -0.01[dps]  Gyro Y : 0.05[dps]  Gyro Z : 0.02[dps]
-> yaw : 0.42[°]  pitch : -0.09[°]  roll : -0.06[°]  Accel X : -0.00[g]  Accel Y : 0.00[g]  Accel Z : 0.99[g]  Gyro X : -0.01[dps]  Gyro Y : -0.01[dps]  Gyro Z : -0.01[dps]
-> yaw : 0.42[°]  pitch : -0.09[°]  roll : -0.05[°]  Accel X : -0.01[g]  Accel Y : 0.00[g]  Accel Z : 1.00[g]  Gyro X : -0.01[dps]  Gyro Y : 0.02[dps]  Gyro Z : 0.02[dps]
-> yaw : 0.42[°]  pitch : -0.09[°]  roll : -0.05[°]  Accel X : -0.00[g]  Accel Y : 0.00[g]  Accel Z : 1.00[g]  Gyro X : 0.02[dps]  Gyro Y : 0.02[dps]  Gyro Z : -0.04[dps]
-> yaw : 0.42[°]  pitch : -0.08[°]  roll : -0.06[°]  Accel X : -0.01[g]  Accel Y : 0.00[g]  Accel Z : 1.00[g]  Gyro X : -0.01[dps]  Gyro Y : -0.07[dps]  Gyro Z : 0.02[dps]
-> yaw : 0.42[°]  pitch : -0.08[°]  roll : -0.06[°]  Accel X : -0.00[g]  Accel Y : 0.00[g]  Accel Z : 1.00[g]  Gyro X : -0.07[dps]  Gyro Y : -0.04[dps]  Gyro Z : 0.02[dps]
```

圖 1、MPU-6050 的資料讀取

-Task 2:

對讀取的資料進行分析確認應需使用何種校正。靜置時尚未經單位轉換的加速度及角速度值如圖 2 所示，皆始終有小幅度的抖動，因而須採取另外撰寫校正程式的方式，透過加總一定次數內的單次誤差量取得總誤差，再將其除以總次數以估算誤差平均值，來校正加速度及角速度值。傾角值則如圖 3 所示，在一定時間內，會到達穩態，因而直接以其值來做校正。

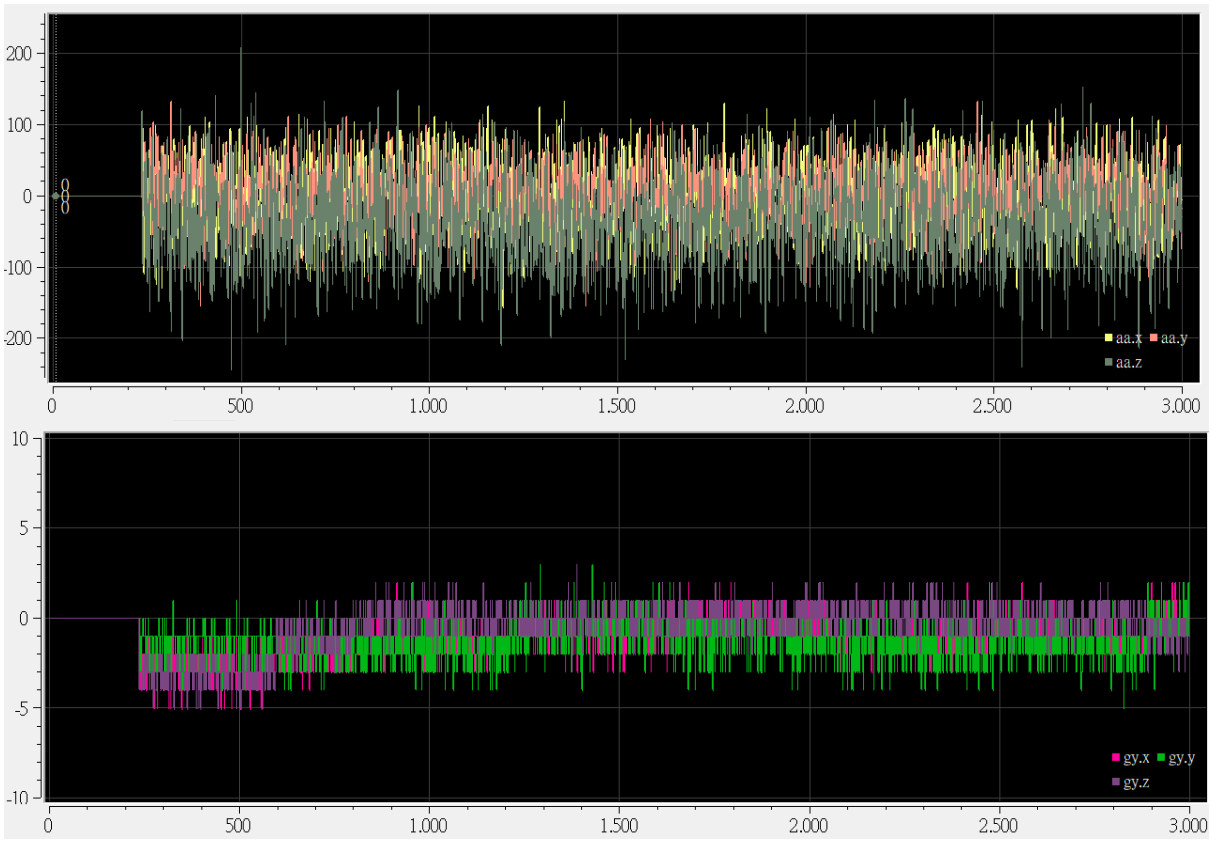


圖 2、加速度及角速度對資料

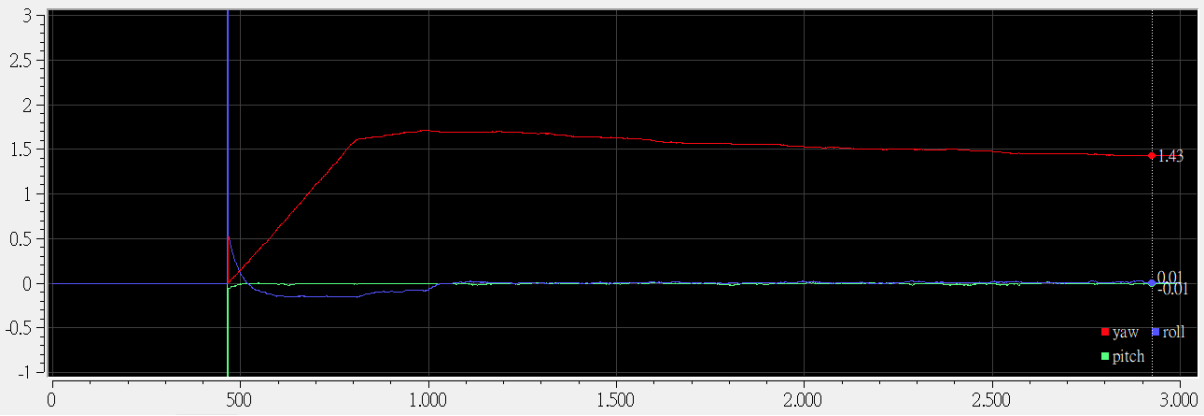


圖 3、傾角值對資料點圖

-Task 3:

將編碼器、麵包板及 Arduino 板的線路接好，寫好使用編碼器測量馬達角位移量的程式，並先透過手轉編碼器來確認角位移量讀值正確。

-Task 4:

將馬達、馬達控制器、電池、麵包板及 Arduino 板的線路接好，先以 PWM 驅動馬達，確認轉速功能和程式控制的正反轉功能正常，再透過 **Task 3** 量測到的讀值來做 PID 控制，讓馬達旋轉 90 度，圖四為測試結果。

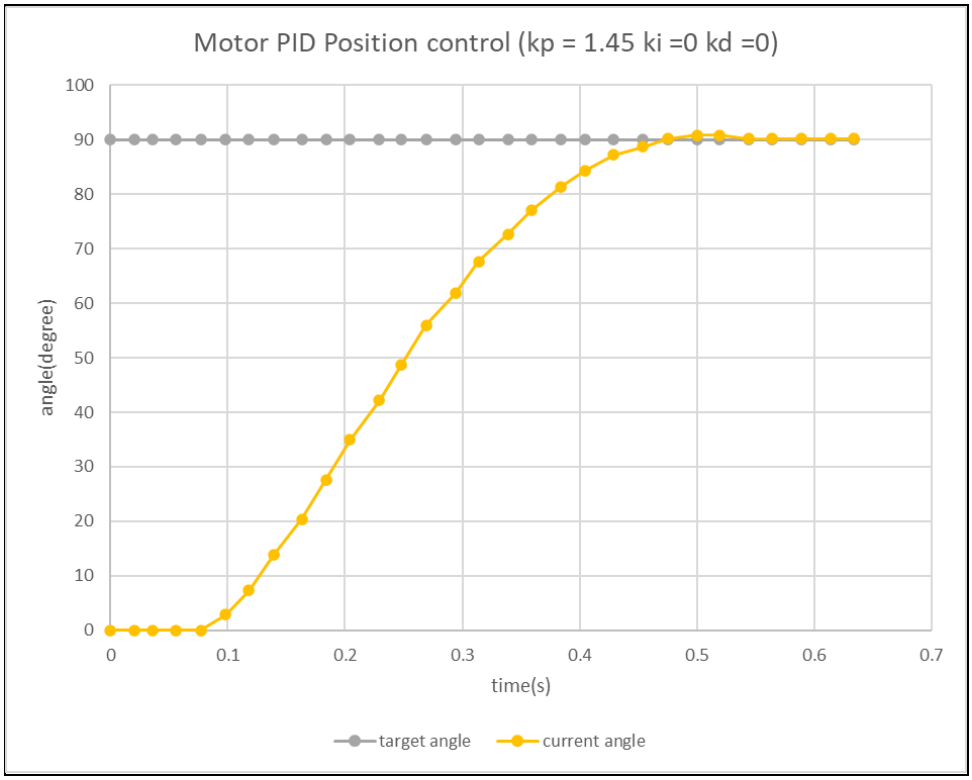


圖 4、PID 控制角度對時間圖

Issues and Challenge

- Challenge 1:

測試時發現 MPU6050 在開始運作時期其 X 軸和 Y 軸構成的平面必須平行於地面才能讀到正常的值

- Challenge 2:

Jeff Rowberg 的程式庫使用程式儲存空間使用率已達 50%，若之後期末的程式量較大，可能無法負荷。

- Challenge 3:

若使用 serial monitor 讀編碼器值時會在開始時出現中途斷電的現象，導致測試不易。

- Challenge 4:

若 PWM 值設太低會造成電壓不在馬達的工作電壓範圍內，無法順利轉動馬達。

- Challenge 5:

以 PID 追角度時，將馬達放在桌上和以手抓住時，能跑到的角度不同，需重新設 kp、ki 及 kd 值才能重新追到 90 度。

Perspectives

學習到馬達驅動器的內部電路構造，例如 H 橋的邏輯閘和電晶體開關是如何同時控制馬達的轉速和轉向，及線路接法。學會編碼器的資料讀取和利用程式、解析度及齒輪減速比將其轉換為馬達角位置的方法。將自動控制的 PID 控制實際應用到馬達位置控制。

Plans for Next Week

- Perspective 1:

寫只用 I2C 的通訊讀加速度及角速度和傾角的程式，而不使用 Jeff Rowberg 直接使用 MPU6050 內建 DMP 來處理資料的方法。

- Perspective 2:

寫校正程式來校正 MPU6050 加速度及角速度量。

- Perspective 3:

改換 SerialPlot 軟體來讀取編碼器資料。

- Perspective 4:

加上開關來控制何時要運行馬達部分的程式。

Remarks

這周上課時，機電課程介紹了測程法，才知道前面提到的那種方法不可行，因為編碼輪不能測走，而應使用兩顆編碼輪，將其轉動角度差透過矩陣換算為車子轉動角度，目前覺得這可能會是期末測試時需用上的，組員也建議我試試用加速度計來測距離，但礙於時間因素，仍未動工。

上週原定的目標是用編碼器讀值而已，但組員說想測一下連接件在馬達運轉時的穩定度才把目標稍微變更。

Objectives

寫出校正程式來校正 MPU6050 的加速度及角速度量，撰寫用開關來控制何時要運行馬達部分的程式。

Summary of Outcomes

-Task1:

校正完 MPU6050 的加速度及角速度量，總次數 2000 的程式及結果如圖 1 所示。

```
217       //Calculation for average error
218       if(calib_count < 2000){
219         cax += aa.x;
220         cay += aa.y;
221         caz += aa.z;
222         cgx += gy.x;
223         cgy += gy.y;
224         cgz += gy.z;
225         Serial.print("calib_count:");
226         Serial.println(calib_count);
227         Serial.print("accel_offset_x:");
228         Serial.print( 0 - cax/calib_count );
229         Serial.print("  accel_offset_y:");
230         Serial.print( 0 - cay/calib_count );
231         Serial.print("  accel_offset_z:");
232         Serial.println( 16384 - caz/calib_count );
233
234         Serial.print("gyro_offset_x:");
235         Serial.print( 0 - cgx/calib_count );
236         Serial.print("  gyro_offset_y:");
237         Serial.print( 0 - cgy/calib_count );
238         Serial.print("  gyro_offset_z:");
239         Serial.println(0 - cgz/calib_count );
240
241         calib_count = calib_count + 1;
242       }
243       else{
244           run_calib = false;
245       }
246       if(run_calib = true){}
247       else{
248         //stop add calib_count
249       }
250     }
Output  Serial Monitor ×
Message (Enter to send message to 'Arduino Uno' on 'COM3')
calib_count:1999
accel_offset_x:11.83  accel_offset_y:-1.87  accel_offset_z:20.05
gyro_offset_x:0.39  gyro_offset_y:0.41  gyro_offset_z:0.52
```

圖 1、校正程式及校正結果

-Task 2:

寫完開關程式，來控制馬達何時轉動，並透過 Pin13 的 LED 燈來確認狀態。

Issues and Challenge

- Challenge 1:

由於目前手邊只有復位式的開關，所以我先讓來自 Arduino 並通過開關的 5V 或因為開關不通而接地的 0V 代表高低電位，再使用 attachInterrupt()函式來讀高低位變化，當它由高變為低電位時，才將另一自訂布林參數反向，並以此值決定是否運行馬達。但實際操作時發現有時太快按下再放開時，卻無法停下馬達或執行馬達部分的程式。

Perspectives

開關的程式及接線使用方法，並知道上拉電阻和下拉電阻目的在於避免電位處於 float 的狀態隨環境上下跳動而無法讓其依開關保持在高或低電位，及其電路圖原理，並且由於 arduino 每個腳位都有內建電阻，因而能直接用 pinMode(BUTTON_PIN, INPUT_PULLUP)來直接使用，不須再另外接 10k 電阻。

Plans for Next Week

- Perspective 1:

研究上述 **Challenge 1** 所提及的問題並試著解決。

- Perspective 2:

繼續撰寫只用 I2C 的通訊讀加速度及角速度和傾角且程式儲存空間使用率較低的程式。

- Perspective 3:

試著撰寫通過加速度計來測距的程式並做實際距離和測量距離比較圖。

Remarks

之前都是用 9V 電池來驅動馬達，但之後那顆應該會拿來供 Arduino 板電，馬達應該會換電壓再高一點的電池來驅動，到時 PID 參數還要再調，如果順利的話應該會跟機構那邊一起測負重時的轉速狀況。

Objectives

找出並解決上週 Challenge 1 的開關彈跳問題，寫完程式儲存空間使用率較低的 IMU 程式。

Summary of Outcomes

-Task 1:

上週所遇問題經研究是開關的「彈跳」現象，經由加入 Debounce 的程式以軟體解決這項問題了。

-Task 2:

寫完上周所提及的程式儲存空間使用率較低的 IMU 程式並校正。

Issues and Challenge

- Challenge 1:

上網蒐集開關的問題後，發現是大部分開關的常見問題，由於在按下和放開開關時，接觸點可能會因衝力與彈力一起作用下，發生連續好幾次的接觸、斷開現象，通常稱此現象為「彈跳」。

- Challenge 2:

先寫好從 I2C 讀取角速度和加速度後，這兩項值是沒問題的，但只使用角速度或加速度來推傾角變化的話，會有明顯的誤差產生。

Perspectives

要解決談彈跳問題可以用硬體方式，例如我自己在機電系統原理的課上剛好有學到的 SR flip-flop 以邏輯閘來解決，或軟體方式，這次使用的即是這種，直接使開關確認到剛開始切換時，不要讀取一定時間內的電壓值，這段時間後確認開關有切換後，再反轉指定的變數狀態。

上網查了以後，發現大部分 IMU 通常都會融合陀螺儀和加速規的資料來透過卡爾曼濾波器來融合這兩項資料求得傾角，因而我直接使用卡爾曼濾波器的相關開源程式庫來求得較準確的傾角。

Plans for Next Week

- Perspective 1:

將馬達的程式與開關的程式整合在一起，並測試其功能是否正常。

- Perspective 2:

將整合的程式所需的部分程式包成函數，並撰寫期中測驗要使用的主程式。

Remarks

這周討論過後，組內認為 IMU 期中不太會用到，因為打算以 time-based 當作控風帆的方式，因此先主要將馬達和開關程式處理好就可以，期末部分，因為有轉向部分，所以可能仍然會用得到。

Objectives

將馬達的程式與開關的程式整合並測試，將部分程式包成函數並撰寫期中測驗要使用的主程式。

Summary of Outcomes

-Task 1:

將馬達的程式與開關的程式整合在一起了，並確認了功能正常，測試內容為按一下馬達轉 90 度，再按一下馬達暫停，持續這個循環，並確認無彈跳現象發生。

-Task 2:

將馬達部分的程式合併為一項為只有 kp、ki、kd 和角度參數的函數 PIDControlMotor()，開關部分則將讀取與 Debounce 整合為另一函數 ReadAndDebounceBotton()。

-Task 3:

撰寫期中測驗要使用的主程式，主要是以開關或 delay()來切換變數 state，再以 state 的狀態決定馬達狀態，並且切換狀態時還會切換 LED 來提醒使用者，目前設計為按下開關後，停兩秒，在讓馬達轉特定角度，再按一下，即停止，重複這個循環以利期中測試不停錶的要求。

Issues and Challenge

- Challenge 1:

由於目前機構那邊還有點小調整，所以仍不確定馬達要轉幾度，目前只能先大致測一下功能正常。

Perspectives

這周撰寫期中測驗的方法，尤其是 state 變數來控狀態的方法，其實是從上禮拜網路找的 IMU 校正程式那邊借鑒過來的，也有點類似另一堂課 labview 業師講的狀態機，由於我認為這樣寫對讀程式或要臨時改哪個階段的參數也方便，就剛好將學習到的東西實作出來看看。

Plans for Next Week

- Perspective 1:

將部分杜邦線替換成 30 公分長，並將所有機電設備固定在車子上。

- Perspective 2:

測試馬達接上桅杆的穩定性，調整 pid 參數及目標角度。

- Perspective 3:

畫好機電設備的工程圖。

Remarks

這周雖然已經寫完了期中需要的程式，但下周就要測的話感覺時間上還是有點趕，或許期末時應該在早個一周開始寫就能這禮拜開始測。

Objectives

將電線延長並把機電設備固定在車子上，及測試調整馬達的 PID 參數及目標角度。

Summary of Outcomes

-Task 1:

將部分杜邦線替換成 30 公分長，焊接部分連接到馬達或電池的電線以延長，並將所有機電設備透過泡棉膠、螺絲或銅柱等方式固定在車子上。

-Task 2:

畫好機電設備的工程圖，大部分是從 grabcad 網站找的 3D 圖檔。

-Task 3:

測好接上風帆的馬達的 PID 參數及目標角度 160 度。

Issues and Challenge

- Challenge 1:

馬達與桅杆的軸以熱熔膠黏不住會空轉，最後透過三秒膠才順利解決。

- Challenge 2:

馬達在轉動時能轉到目標角度，但因風的推力風帆會逐漸往後倒，因而我們在桅杆後方加上擋塊以避免此現象。

Perspectives

學習焊接電線的方法及銅柱的用法，如墊高電路板及避免 PCB 背板的焊點短路。

Plans for Next Week

- Perspective 1:

先以期中的車跑一次期末測試的場地。

- Perspective 2:

討論應改善或前進的方向。

Remarks

這周檢錄之前，突然發現馬達轉不動，及時測試才找到問題是微控制器壞了，幸好有別組同學願意借我們，才勉強過關，下次測驗前，真的要提找到再檢查有沒有零件的問題和備好可能會壞的電子零件。

Objectives

期末場地預先測試及討論車子整體應改善或調整的機構或控制。

Summary of Outcomes

-Task1:

先以期中的車跑一次期末測試的場地，並討論問題點。

-Task 2:

討論還應增加那些機電設備。

Issues and Challenge

- Challenge 1:

測驗時，發現會微微地撞到草地，並有部分機率輾過去後，再順利達到終點，但若風帆角度在撞到草地後，正對著風的面積太小可能會因而不動，若無正對賽道則會往偏離賽道終點的方向前進而脫離賽道。

Perspectives

目前認為應該先增加風帆的面積來提高風的推力，盡可能避免車子不同的情況發生。

Plans for Next Week

- Perspective 1:

購買新的微控制器及伺服馬達以張開風帆。

- Perspective 2:

撰寫兩顆伺服馬達程式。

Remarks

再增加風帆面積後，就是要處理會撞到草地的問題，及最後轉彎角度的問題，轉彎問題應該原本預計是配合感測器，來控風帆方向但還沒實作，至於撞到草地就還再思考，或許應考慮讓風帆再進入轉彎前不要完全張開風帆，使風的推力變小這個方向來做。

Objectives

買完電子材料及撰寫伺服馬達控制的程式。

Summary of Outcomes

-Task1:

買完 Arduino 板及兩顆 SG90 伺服馬達。

-Task 2:

使用 Servo.h 函式庫來控制伺服馬達的轉動方向和角度。

Issues and Challenge

- Challenge 1:

測試時，伺服馬達轉 180 度時，通常會少轉大約 10 度。

Perspectives

學會如何使用函式 attach()及 write()來控制伺服馬達。

Plans for Next Week

- Perspective 1:

測試利用伺服馬達將風帆張開的機構。

- Perspective 2:

前項測試若穩定度高，再思考轉彎問題如何透過編碼器或陀螺儀感測器解決或單純以 time-based 解決。

- Perspective 3:

風帆轉向控制討論。

Remarks

Challenge 1 的問題我想等到實際接上副桅桿測在看看是否需要考慮，因為考量我們的機構只需左右兩張帆都轉 90 度就能達成張開的目的，或許這項問題只要單純增加角度就能解決了。

Objectives

將伺服馬達連接副桅桿，測試風帆張開與關閉功能，並重新繪製了接線圖。

Summary of Outcomes

-Task 1:

將伺服馬達的舵與 3D 列印出來的套筒用束帶綑緊，成功使兩顆馬達連接到兩根副桅桿上，如圖 1 所示，按下開關後，會再 5 秒後啟動兩顆馬達後，使其各轉 90 度，並讓風帆展開，如圖 2 所示，最後再次按下開關後，會讓馬達將風帆關閉，呈現與圖 1 一樣的狀態。

圖 1、馬達連接副桅桿圖

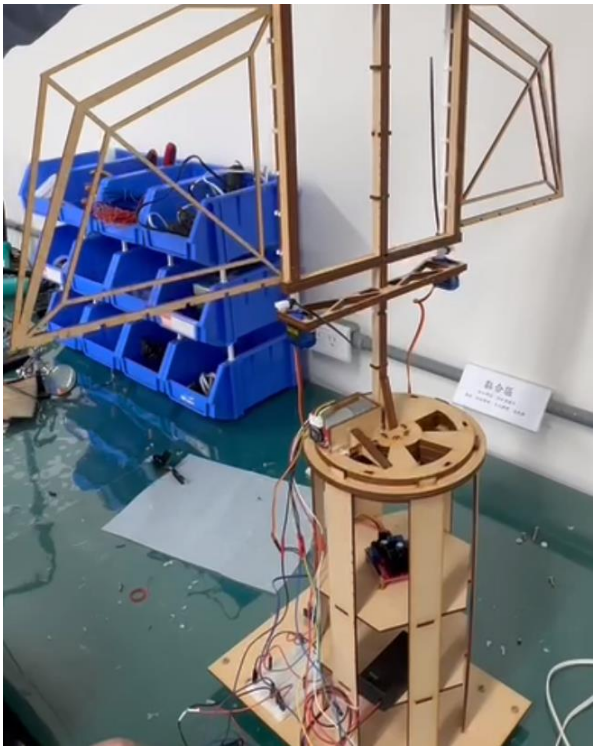


圖 2、風帆展開圖

-Task 2:

由於目前的電子元件及電線數目逐漸變多，因此我又畫了另一張接線圖，如圖 3 所示，來幫助我確認電線的接法正確，由於目前的線上軟體都無囊括所有我需要的電子元件，因此我使用 Block Circuit EDIT 這個線上電路圖繪製軟體來匯入所有電子元件示意圖，再透過其提供的功能完成拉線的圖示。

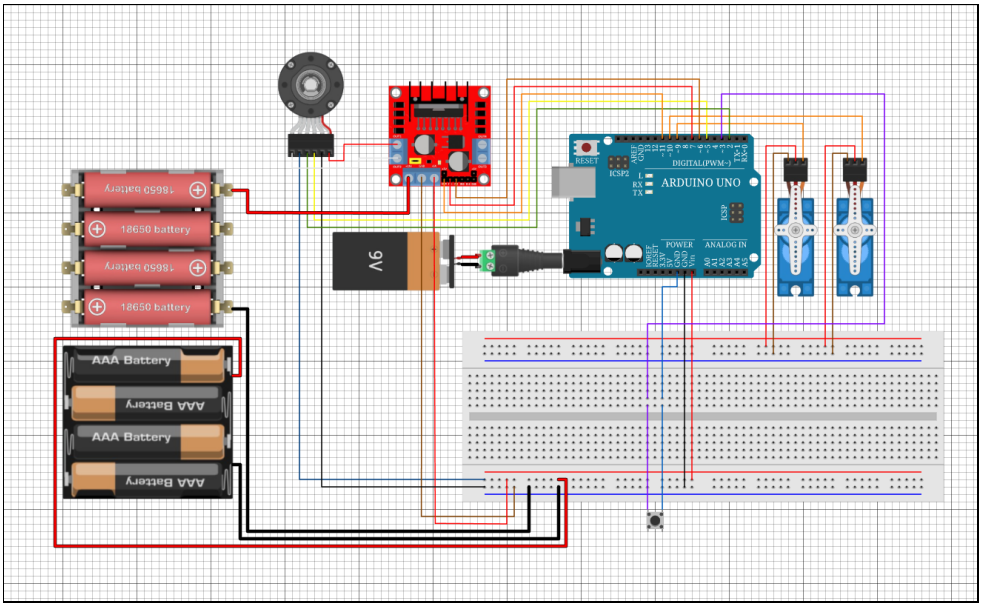


圖 3、元件接線圖

Issues and Challenge

- Challenge 1:

副桅桿的機構由於原本設計圖的錯誤而需重新印一次，且原先風帆的結構也有部分地方快斷裂的感覺，因而須將部分結構重新雷切，因而機構那邊又花了不少時間調整設計和組合風帆。

- Challenge 2:

在上述的 Task 1 有遇到因馬達轉速過快而使風帆震動太大力的情形，因而除了角度控制外，可能還需加上速度控制。

Perspectives

在上述的 Task 1 中，我學會了如何使伺服馬達一顆反轉，一顆反轉，由於其只能指定角度，因而我使其中一顆由 0 度轉到 90 度，另一顆則從 90 度轉回 0 度，即能達成需求。

Plans for Next Week

- Perspective 1:

研究伺服馬達如何調整速度的程式，並實際測試其功能。

- Perspective 2:

研究用來轉動轉盤，帶動風帆轉向的步進馬達的程式。

Remarks

目前感覺每周都有點進度緩慢，組員中還有人在每週五下午原定的上課時間提早離開，實在認為進度有點堪憂。

Objectives

伺服馬達速度控制程式撰寫與測試，並尋找控制步進馬達程式庫，使其能達成角度控制。

Summary of Outcomes

-Task 1:

伺服馬達的速度控制，我參考了伺服馬達，回授控制入門 - 互動的風格 - Medium 這個網站提供的 for 迴圈程式，如圖 1 所示，不同的是，由於我需要同時我需要兩顆馬達同時作動，因而採取了 while 迴圈的寫法，如圖 2 所示，並將條件改為 AND 閘型式，讓控制馬達的參數都需跑到特定位置才停下。

```
for(int i=0;i<180;i++)
{
  myservo.write(i);
  delay(20);  //旋轉1度，所需的時間，可調整參數
}

for(int i=179;i>=0;i--)
{
  myservo.write(i);
  delay(100);
}
```

圖 1、以 for 迴圈驅動一顆馬達程式

```
// close side sails
while(pos_l < 90 && pos_r > 0){
  leftservo.write(pos_l); delay(15);
  //Serial.print("pos_l:");
  //Serial.println(pos_l);
  rightservo.write(pos_r); delay(20);
  //Serial.print("pos_r:");
  //Serial.println(pos_r);
  pos_l += 1;
  pos_r -= 1;
}
pos_l = 90;
pos_r = 0;
```

圖 2、以 while 迴圈驅動兩顆馬達程式

-Task 2:

步進馬達的程式庫，我參考了[Arduino 範例] ULN2003 驅動板+28BYJ-48 步進馬達 (jmaker.com.tw)這個網站的教學，來做使用，其直接使用第三方函式庫 Unistep2 來控制步進馬達，最終，我能使用 move(step)這個函數來控制馬達的角度，但也有許多可調控的參數，例如若想調整馬達的轉速，我能在建立 Unistep2 物件時直接調整參數，其中第五個參數為設定馬達轉一圈總共需要的步數，第六個參數則為每步的時間，因此調整這兩個參數都能讓馬達的轉速改變。

Issues and Challenge

- Challenge 1:

在調整伺服馬達速度時，我發現由於檢測開關的程式也會在同一個 loop 底下跑，因而在使用 delay()函數讓伺服馬達在直流馬達升起主風帆後，再張開副風帆時，會不斷延遲 3 秒，而若再這 3 秒內觸碰開關後，就不會有任何反應，因而我最後須將這個 delay()函數也寫再 while 迴圈內，使其只跑一次就停止，才能避免發生上述現象。

- Challenge 2:

在調整伺服馬達速度時，我發現若圖 2 的 delay()中同樣輸入 15ms 時，不知為何 rightmotor 會轉得相對的快，最終，我只好將其提高到 20ms 讓兩顆馬達轉速大概一致，但目前仍不清楚會造成轉速不一的原因。

Perspectives

這周主要學會了如何利用 arduino 來控制伺服和步進馬達的角度和速度，起初原本認為角度控好就行了，經由實際經驗發現速度也是需多加考慮的一項因素，也藉由這次機會，認識了伺服馬達的內部構造與運作原理。

Plans for Next Week

- Perspective 1:

將步進馬達連接到圓盤的孔內，測試能不能轉動風帆。

- Perspective 2:

研究風帆旋轉角度策略，並試圖以程式控制。

- Perspective 3:

將新加入的電子元件畫進接線圖內。

Remarks

目前有在考慮是否有需要煞車功能，但可能要等到這周五的時間才能測試，應該會以不升起也不張開風帆得情況下測試新車子滑下去會不會還撞到草地。

Objectives

步進馬達與直流馬達測試，發現力矩不夠，提出風帆旋轉策略想法。

Summary of Outcomes

-Task 1:

將步進馬達與圓盤孔連接，試圖轉動風帆時，發現無法順利轉動圓盤及其上的直流馬達與風帆。

-Task 2:

由於側風帆的測試已完成，我測試了之前期中用的直流馬達能否帶動主風帆和側風帆，在測試階段， 一樣是用同樣的 pid 值的話，只能將其轉動 70 度，剛好在風帆造成的重力力矩最大值的地方，而將 kp 的值調大後，可帶動風帆到最高點，但會讓桅杆轉動超過設定的 160 度，並且撞壞後方支撐的三角塊。

-Task 3:

這周我也向組員們提出風帆的策略，主要是想利用編碼輪來感測車子目前的位置，並利用其來決定風帆的轉向，而因為車身的轉向也會影響風帆的轉向，因而須利用 IMU 來感測車子的偏轉角，最後反算回帶動風帆的馬達應該轉幾度。

Issues and Challenge

- Challenge 1:

在發現步進馬達轉不動的問題後，我查了組員提供的馬達，其力矩為 300 gf.cm，但光在圓盤上的直流馬達就已占 92g，因此我推估是力矩不夠的問題。

- Challenge 2:

直流馬達的問題，我認為是因為我們的風帆對馬達所造成的力矩會隨著高度，有不同的變化，而 pid 只能在位置的調控相對精準，但可能不太能處理這種力矩變動的問題。

Perspectives

由於這周的問題基本都遇到了馬達轉矩不夠的問題，我認為之後要選馬達時也須還是要稍微評估一下其需負載的轉矩，特別是這次步進馬達沒有先查就相信組員說可以真的是有點耽誤了可以做其他事的時間。

Plans for Next Week

- Perspective 1:

將抬升和轉動風帆的馬達都使用 mg996r 伺服馬達，其轉矩是之前馬達的 3 倍。

- Perspective 2:

寫程式將伺服馬達結合 IMU 來達成 Task 3 所提及的策略。

Remarks

目前已測試過若張開風帆情況下，肯定會撞到草地，因而希望能讓在撞到草地前就讓風帆轉向正對風機，讓車子轉向，但仍不確定其成效如何，但現階段須先把轉向功能做好，在考慮煞車功能。

Objectives

測試 mg996r 升起風帆的轉矩足夠，轉向風帆的馬達能配合 IMU 測得的偏轉角決定轉動量以供風帆旋轉策略，將風帆轉向策略中需用到的編碼輪改為 TIME-BASED 決定，發現 IMU 問題並解決。

Summary of Outcomes

-Task 1:

將直流馬達換成伺服馬達後，其可成功轉動 160 度帶動風帆升起，並將其作速度控制使風帆升起速度降低。

-Task 2:

轉動風帆的馬達，我首先測試其跟隨 IMU 的偏轉角而跟著轉動相同的角度，測試結果十分順利，但在測到一半卻發現，IMU 會在隨機時間點不再像電腦序列埠回傳值，並且馬達也不再會跟隨其轉動。

-Task 3:

跟組員討論過後，編碼輪的部分由於底盤尺寸有限，因而須把目前後輪的部分移置中間，才能騰出空間給編碼輪，因而組員建議先以時間來決定風帆轉動的控制點，先看具體成效再討論是否真的需要編碼輪。

Issues and Challenge

- Challenge 1:

經過網路查找與我使用相同函式庫並有相同情況的資料後發現，是由於 Wire 函式庫中的 bug 所導致的，其會讓 I2C 接收和或傳輸功能在某些條件下阻塞，因而需要手動加入 Wire.setWireTimeout(3000,true)這項程式碼，來讓 I2C 通訊阻塞時解鎖。

Perspectives

這周遇到的問題比較少，但還是至少學到了 Wire 函式庫原來有這項還沒修復的 bug 存在，並學會了如何解決，但最重要的還是遇到問題應該趕快查找網路資料，才不會浪費太多時間。

Plans for Next Week

- Perspective 1:

測試轉向的馬達能否順利帶動風帆轉向。

- Perspective 2:

跑一次期末測驗的場地看看還有哪邊需要加強。

Remarks

感覺時間還是稍趕，希望這周五至少能先測一遍現在的功能都正常達成。