# 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。

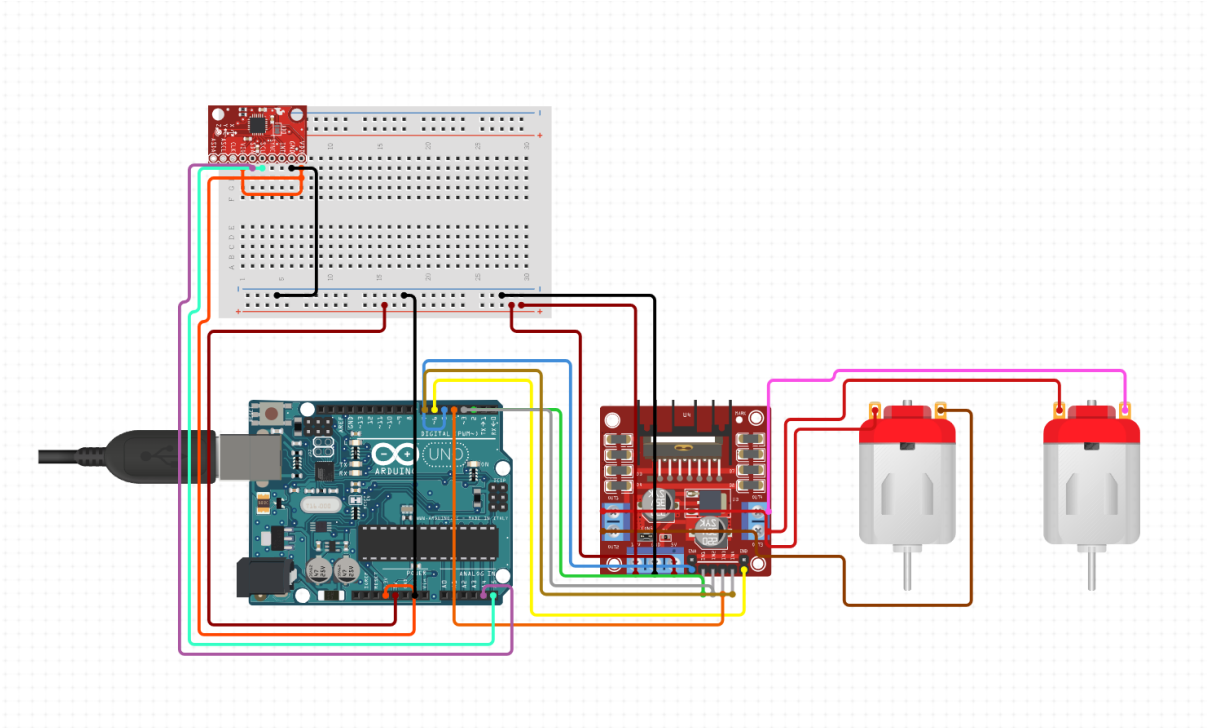
MPU-6050 其主要用途為當測量到的向下加速度高於一定值時，觸發風帆展開，次要用途為量測目前車體的三軸角度。

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)](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。

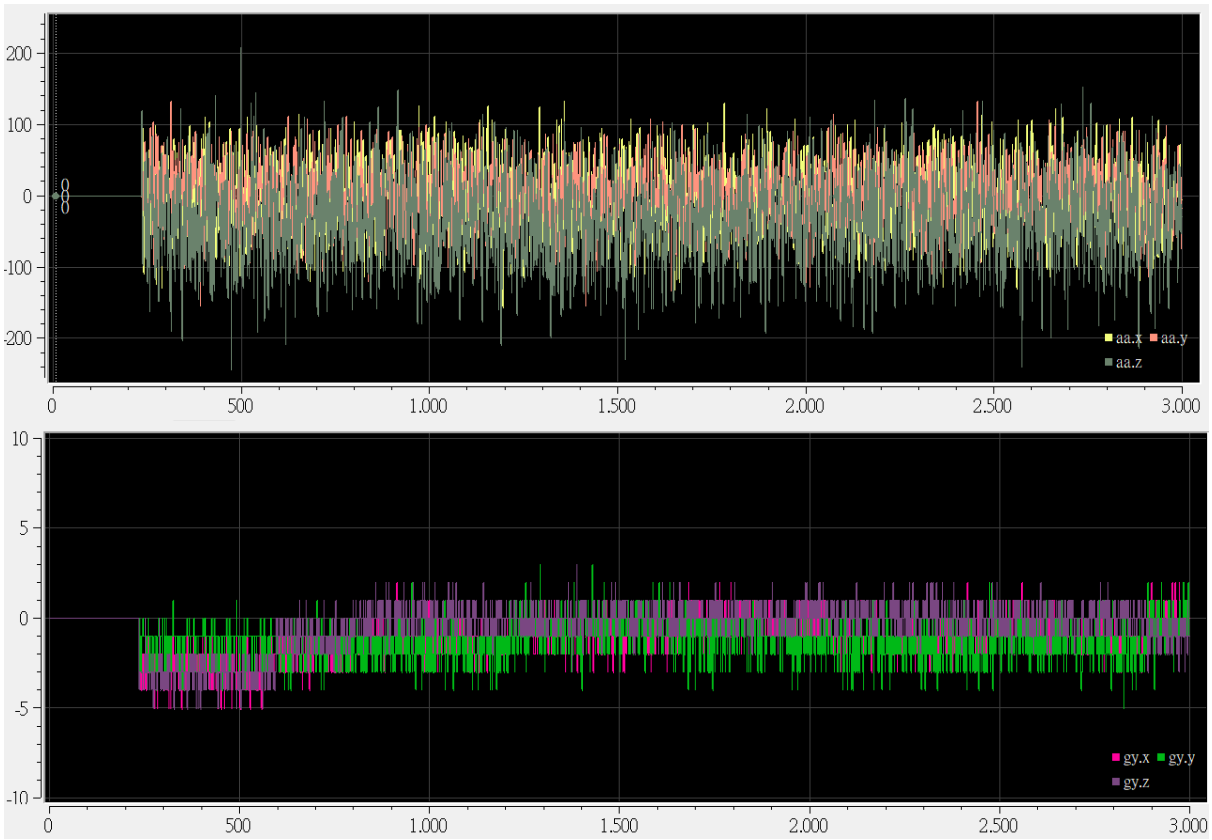



圖 2、加速度及角速度對資料

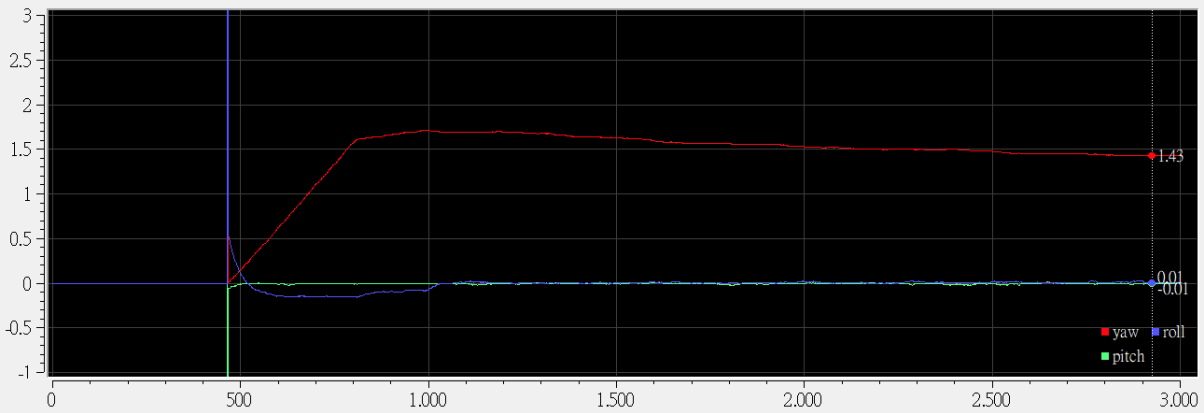



圖 3、傾角值對資料點圖

**-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。

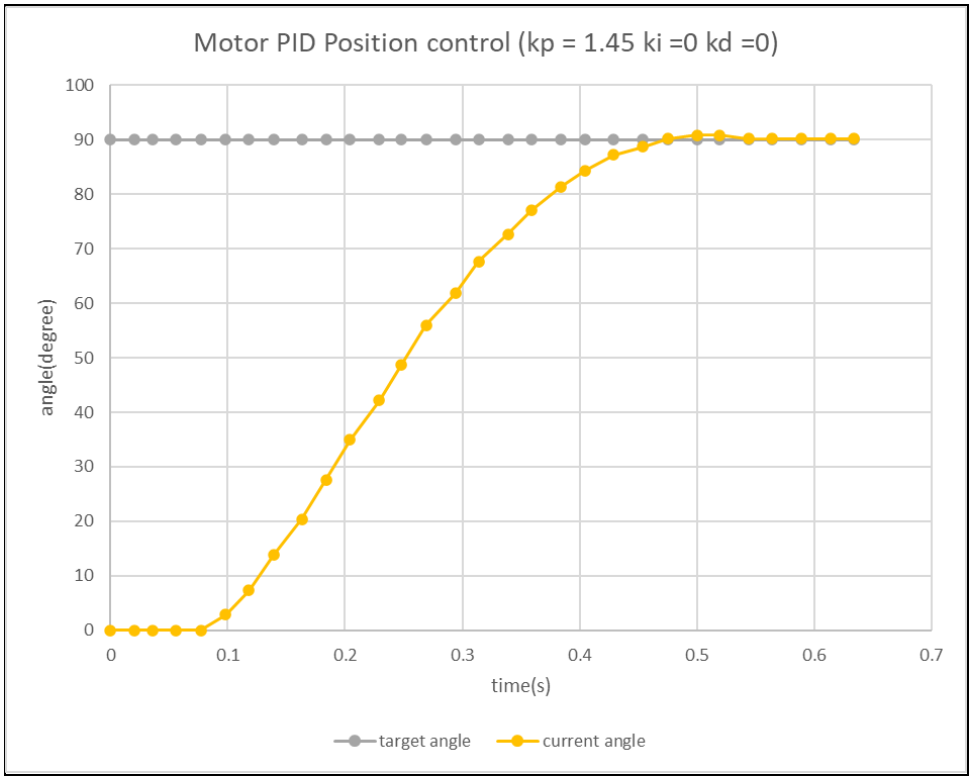



圖 4、PID 控制角度對時間圖

## 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。