Logic Design Lab

Final Project

Team34

組員：

106030009 葉蓁

106030028 劉多聞

PID balance ball

1. Motivation

這學期我和葉蓁修了很多課，所以在想邏設實驗final project的時候就在想到底要做甚麼題目才能把所學的課程能用的都用到，並且以邏設為核心。大概算了一下，我們在動機系修了控制系統一、機械設計、電子學二，在資工系修了邏輯設計實驗、嵌入式系統，於是看了許多網路上的影片就決定是PID Balance Ball了。

大概講一下我們學的課跟final project的關係：

(1) 邏輯設計實驗：這個是絕對的核心，負責連接project的程式運作和所有訊號的連接和控制。

(2) 嵌入式系統概論：紅外線測距儀的部分是靠arduino算出來的，因為從超音波測距改成紅外線測距的時間有點緊急，所以測距的部分就交給arduino然後再將居離的訊號回傳給fpga，我想這樣也可以用到lab裡面的chip2chip。

(3) 機械設計：這門課幫我們設計的final project的支撐架系統和多連桿系統。

(4) 控制系統一：這堂課最後半段就是在教PID Control(原理下面會講)，所以想說final project可以用這個原理把它實作出來也比較知道自己控一在學甚麼。

(5) 電子二：這們課幫到的部分比較小，大概就是麵包板的電路連接，跟使用電容濾波的一些觀念。

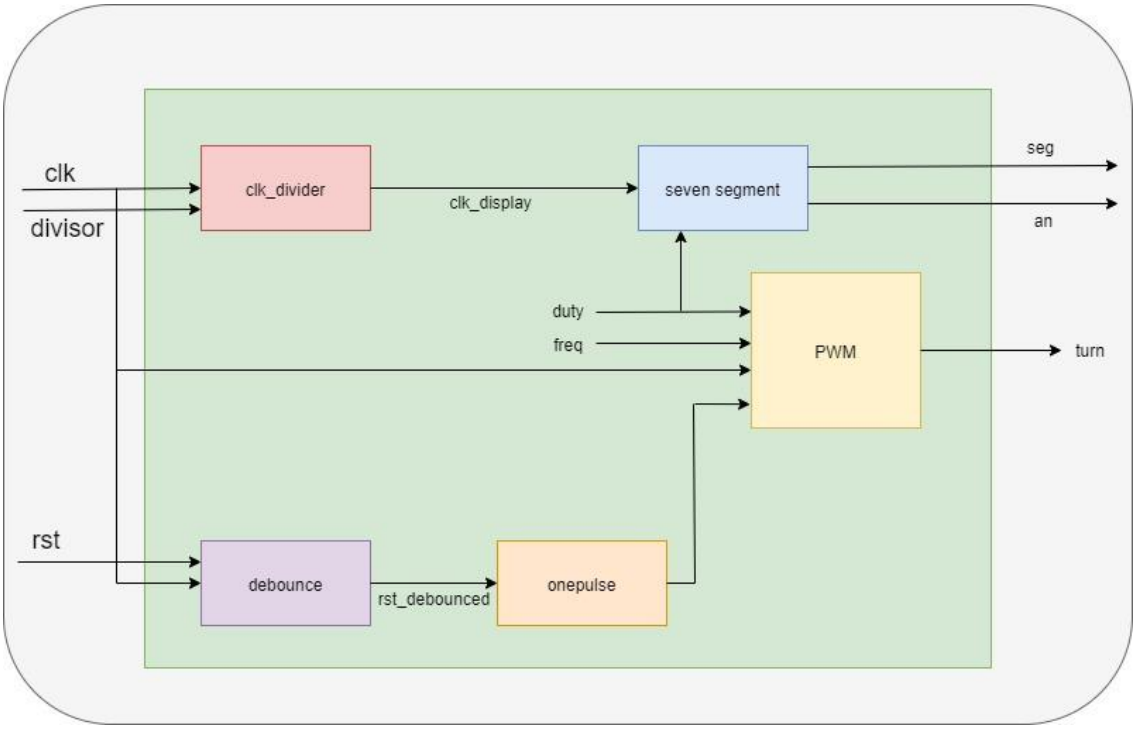
2. Design flow

(1)Mode1 : Use Switches to balance the ball

A.設計過程：一開始PID還無法用verilog寫出來，於就用Switch去調整 PWM的duty cycle來讓馬達轉不同的角度。

B.電路圖：

Servo Motor：



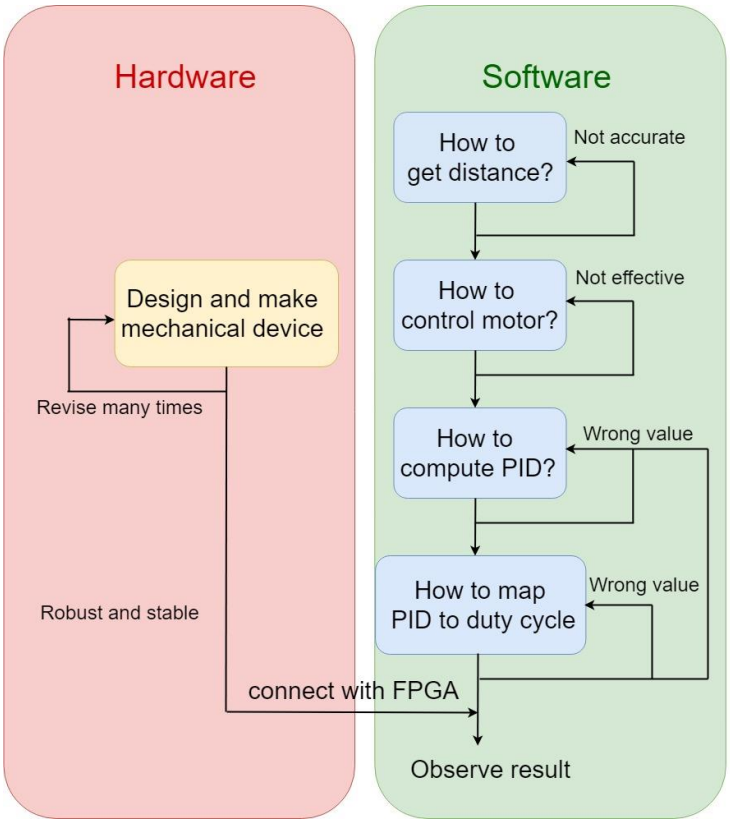
C.電路分析：

主要是利用SW[0]~SW[10]去調整duty cycle，在SW[0]~SW[10]我們有設定不同的加減量來調整duty cycle，然後duty cycle的值會顯示在seven
segment上。PWM的block會去換算馬達需要轉多少角度。

D.控制平衡分析：

由於人眼和手之間的神經訊息傳導需要時間，然後switch調整也需要一段時間，所以從人腦判斷到switch調整完的延遲會很久，所以用這個方法來控制球來達到平衡需要經過大量的練習，才有辦法在50公分左右的軌道上把球控制在正中間。這個模式相對來說比較有趣，因為很像在打電動，不可能一次就控制好，然後就算控制好也要馬上讓答停下來，所以總體來說這種控制比PID難且不方便。

(2)Mode2 : Automatic balance the ball



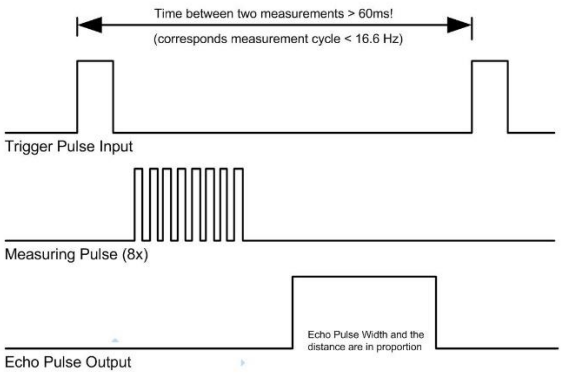
如上圖，基本上硬體軟體兩邊是同步進行的，以下先由軟體方面說明：

(1)Software

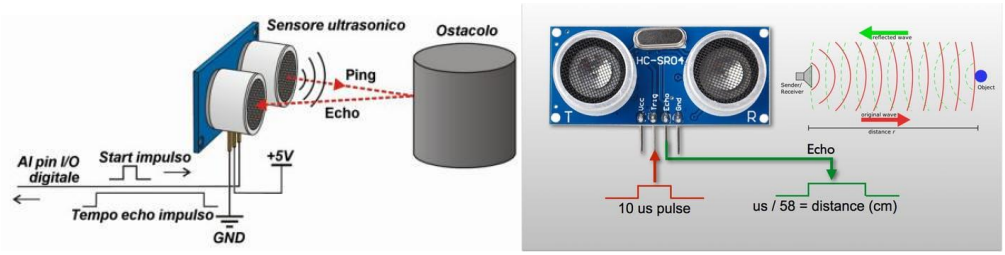
軟體方面先構思此project需要做到哪些事情。我分為三件事，剛好搭配block Diagram來看分別為讀取距離、計算PID值、驅動馬達：

A. Get Distance

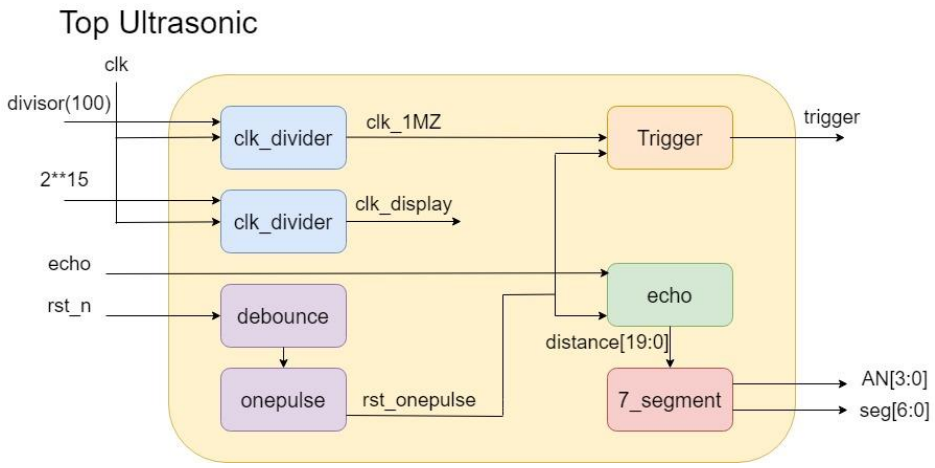
這個project要成功的先決條件，就是要能準確的讀取球現在距離。如此才能得到誤差，並做後續的運算。而根據我所學過這類的sensor，較簡單的就是arduino最常用的超音波測距模組，如下圖(HC-SR04)



這個超音波運作原理相當簡易，可以看到該模組有四個腳位，扣掉power supply的Vcc跟gnd後，一個是output的trigger pin, 一個是input 的 echo pin。使用者根據右上的波形圖，當要量測距離時，將trigger pin設為High並持續至少10ms，然後再設回low，此時模組就會發出八個連續的measure pulse。接著使用者只要等到echo pin收到High訊號的時候，開始計時，計到變回Low後，這段High持續的時間，就是「兩倍」超聲波走這段距離所需的時間。兩倍是因為包含了去跟回程。而超聲波的速度就是聲波的速度，在空氣中常溫下大約340公尺/秒，用所求單位的公式換算，可得出要的距離單位。以下為網路上找到助於理解的運作波形圖。

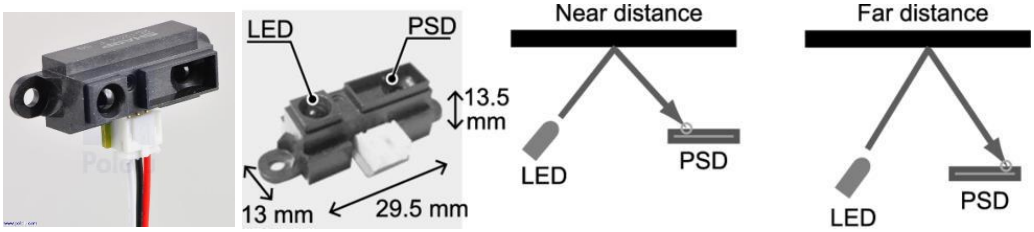


而我有實際做出用FPGA成功讀取距離，再用七段顯示器呈現出來的Ultrasonic module。Block diagram如下：



後來因為超音波sensor好像必須偵測比較有穩定平面的物件，比如我們的球，他可能會繞過球導致波的傳遞有偏差，量出的距離非常不精準。有效範圍非常小，大概只有15公分左右，在百般不願意之下我們選擇換另一個測距模組。

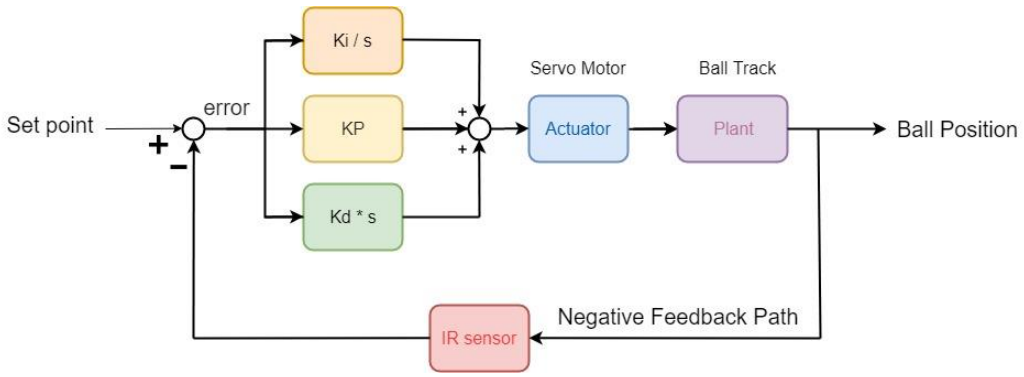
我們採用網路上許多人用的SHARP GP2Y0A21YK0F 有效範圍10-80cm(data sheet宣稱)的紅外線測距sensor。



但當時時間比較趕，它的運作原理也稍複雜，短時間內可能來不及用FPGA刻出來，因此我們想說能否透過Arduino讀取資料，再將十進位的distance用decimal to binary conversion轉換，包成6個bit的資料用jumper傳給FPGA。如此一來實驗結果：真的能有效讀取資料並傳送到FPGA板(用Seven segment顯示驗證)。我們終於順利解決第一關，進而進入下個環節。

B. Compute PID

首先要先了解PID control的原理。參考網路上的資料，我先畫出以下簡易回授圖：



簡單解釋一下：受控場(plant)是平衡架，由伺服馬達(actuator)驅動它，得到的output是球的位置(Ball Position)。藉由一個IR sensor去讀取output，然後回授到input固定的位置(set point)，兩者相減得到一個誤差值。這個誤差值如果最簡單的控制方法，就是用一個單純的P gain去放大然後送給驅動器，讓他去追命令，減少誤差。這種只有一個放大倍率的控制叫做P control。但這種簡易的控制器有它的限制，可能會有太多的過衝量(overshoot)，或是因為system type是0的原因，即使input只是一個step還是會有穩態誤差。所以我們想用PID control，就是PD加上PI，D是Differential微分，將誤差微分後乘以比例係數(ki)，再加上I是Integral積分，最後送給驅動器。

簡單講PID三個項的目的：以我們project為例：P 能夠讓我們的Ball去趨近中間setpoint的位置，D能夠考慮球的速度(位置微分)改變馬達輸出的大

小，減少球過衝停不下來的問題，而I則是避免球卡再一個很靠近(誤差3-5公分卻停住不動)但錯的位置的窘境。

因此，根據上面的分析，我們的code大致如下：

首先宣告PID的參數：

```
29  //Control Parameters.
30  parameter kp = 8;
31  parameter ki = 0.2;  //0.2
32  parameter kd = 2000;
33  parameter SetPoint = 35;
34
35  //For sample count. //Sample period 50ms
36  reg [36:0]count;
37  wire [36:0]next_count;
38  reg [19:0] PID_i, PID_d, PID_p,PID_total, last_PID_i;
39  reg [19:0] next_PID_d, next_PID_p ,next_PID_total;
40
41  //20bit.
42  reg [19:0]err_difference;
43  reg [19:0]error;
44  reg [19:0]last_error, next_last_error;
45
```

然後是Sequential的部分：更新PID運算的各種變數：

```
46  //Sequential : Update parameters every sample period (50ms)
47  always@(posedge clk_20Hz)begin
48      if(rst_one_pulse)begin
49          err_difference <= 20'd0;
50          error <= 20'd0;
51          last_error <= 20'd0;
52          PID_i <= 20'd0;
53          PID_p <= 20'd0;
54          PID_d <= 20'd0;
55          PID_total <= 20'd0;
56      end
57      else begin
58          err_difference <= error - last_error; //error difference.
59          error <= SetPoint - distance;
60          last_error <= next_last_error;
61          PID_i <= next_PID_i;
62          PID_p <= next_PID_p;
63          PID_d <= next_PID_d;
64          PID_total <= next_PID_total;
65      end
66  end
```

最後是Combinational，當input一變化，馬上改變值的部分。

```
67  //Combinational
68  always@(*)begin
69                  next_PID_p = kp * error;
70                  next_PID_d = kd * (err_difference/period);
71                  if( -8<error  &&  error < 8)begin
72                      next_PID_i = PID_i + (ki * error);
73                  end
74                  else next_PID_i = 0;
75          next_last_error = last_error;
76          if( -2 <= error && error >= 2)
77                    next_PID_total = 30; //Let servo motor to zero degree.
78           else
79          next_PID_total = next_PID_p + next_PID_d + next_PID_i ;
80  end
```

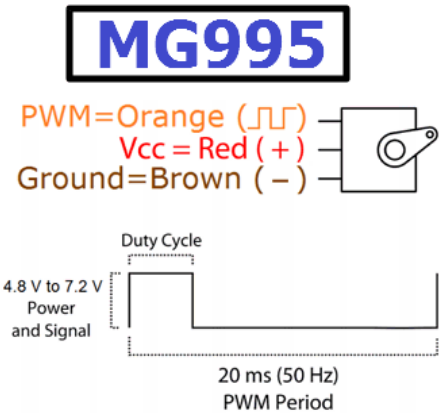
再71~74行的部分，我們讓PID的I項部分，在球的誤差小於8公分內才作用，因為實驗發現球在這個範圍內很容易卡住。加了這項有助於隨著時間變長，誤差會透過這項慢慢放大，機器才會繼續去平衡它。

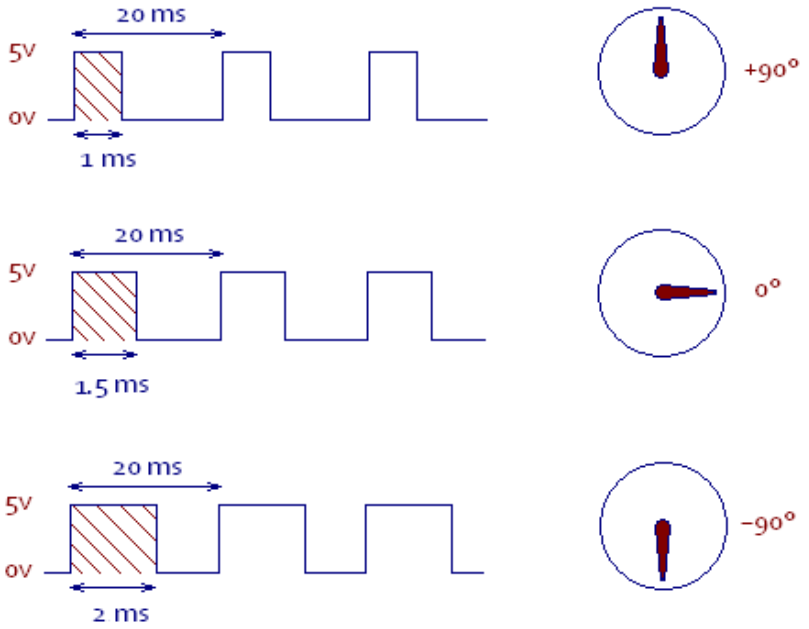
然後在76~79的部分：如果誤差小於2公分內，我們就讓balance beam停住(穩定在0度角)因為發現球很容易在+-2公分左右，明明看起來平衡了，可能因為雜訊或干擾，導致sensor讀到的值一直覺得不穩定，給FPGA後會不停的震盪，雖然是輕微震盪，但我們想要看到它平衡停住的樣子，所以算是硬寫了一個邊界條件。

PID的compute到這邊算是完成了。接著要看如何用這個結果有效送給馬達，讓他聽從我們的命令去運作。

C. Drive Servo motor

我們選用的這顆MG995 Servo motor，主要原因是它能夠提供高達13公斤的扭力，因為要推動平衡架和球應該需要不少力量，我們選用這顆馬達。伺服馬達的運作原理是PWM，簡易說明圖如下：





PWM的原理就是根據不同的duty cycle，會有不同的波形結果。Pulse-Width-Modulation，脈衝寬度調變，顧名思義就是要調變脈衝的寬度，就是要有多寬的High(or low)。Duty cycle是指一個周期內，有多少比例(%)的時間signal是High的訊號。以上面波形圖觀察，我們這顆馬達datasheet寫說最大運轉角度是180度，如果我們取中間值為0度，剛好讓平衡架維持在0度的地方，那我們想讓它正負各轉90度。根據lab5的音樂模組交的PWM，我幾乎完全沿用，因為原理是一樣的!都是要產生PWM的訊號。唯一要注意的地方有兩點：

(i) 這個伺服馬達PWM的頻率是 50Hz，所以freq = 50。

(ii) 根據duty cycle的比較，我們最小有2.5%，最大12.5%，跟1024比較大約是26 ~ 128的值。所以關鍵在這裡：

如何將PID算出來的值mapping到合理的duty cycle 範圍？

因為如果超出這個範圍，馬達很有可能有非預期的行為。

我們先單純測PID的值，讓他印出在7 segment去觀察值的範圍：

發現大概介於-400到+200之間。

要轉換到26~126之間，我們用最簡單的比例方式，

就是先shift到同個基準，在除以6。

如此一來理應要有合理的轉換，code如下：

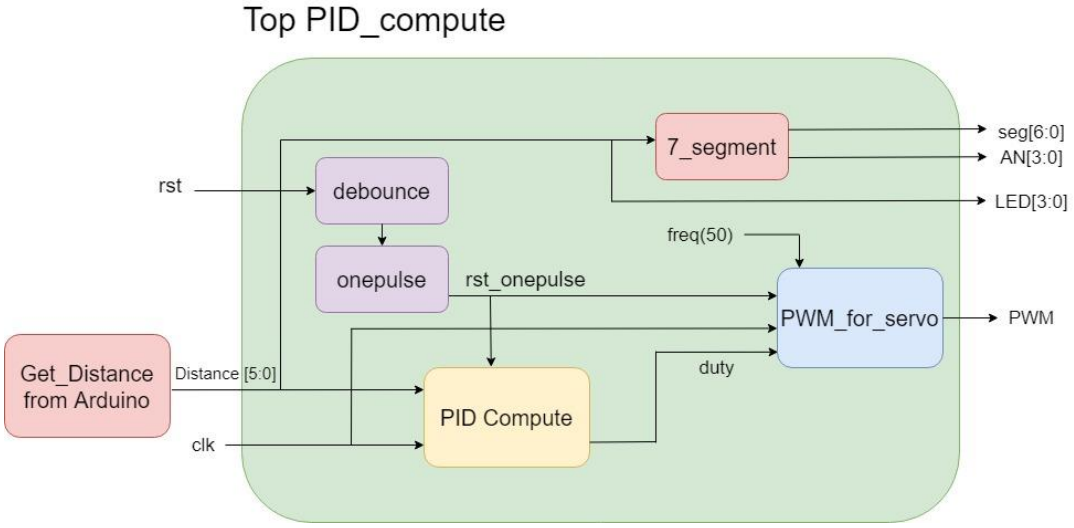
```
90   //From -400 +200  to  duty cycle 26- 126
91   assign PID_to_duty = (PID_total + 426)/6 ;
92
```

接著就是把map的值送給PWM模組去驅動servo motor!

```
92   ////////////Control Servo Motor./////////////////
93    //PID should be mapped to 'duty cycle' (26-128) to write servo motor.
94   parameter freq = 32'd50; //50HZ.
95   PWM PWM_for_servo(
96       .clk(clk),
97       .reset(rst_one_pulse),
98       .freq(freq),
99       .duty(PID_to_duty),
100      .PWM(PWM)
101  );
```

伺服馬達有三條線，分別為Vcc,gnd跟signal。Signal的線我們就接到xdc檔案設定的JB port，(上圖是PWM這個signal，我記得設成JB9或10的樣子)。這樣就可以傳送訊號了！

最後附上整個Top module的block diagram



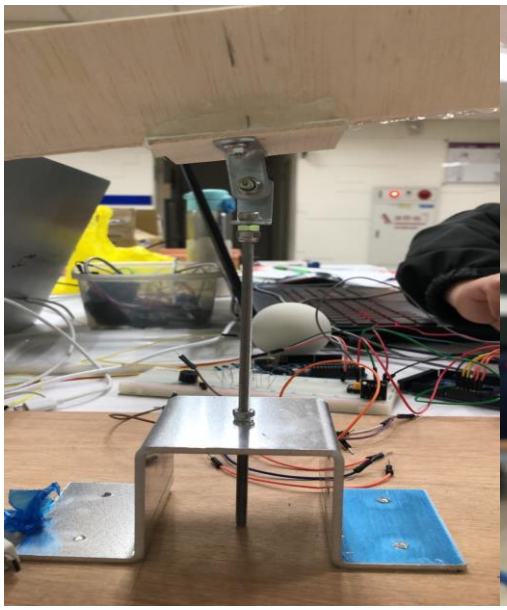
：

3. Mechanism & Schematics. (Block diagram)

Mechanism:

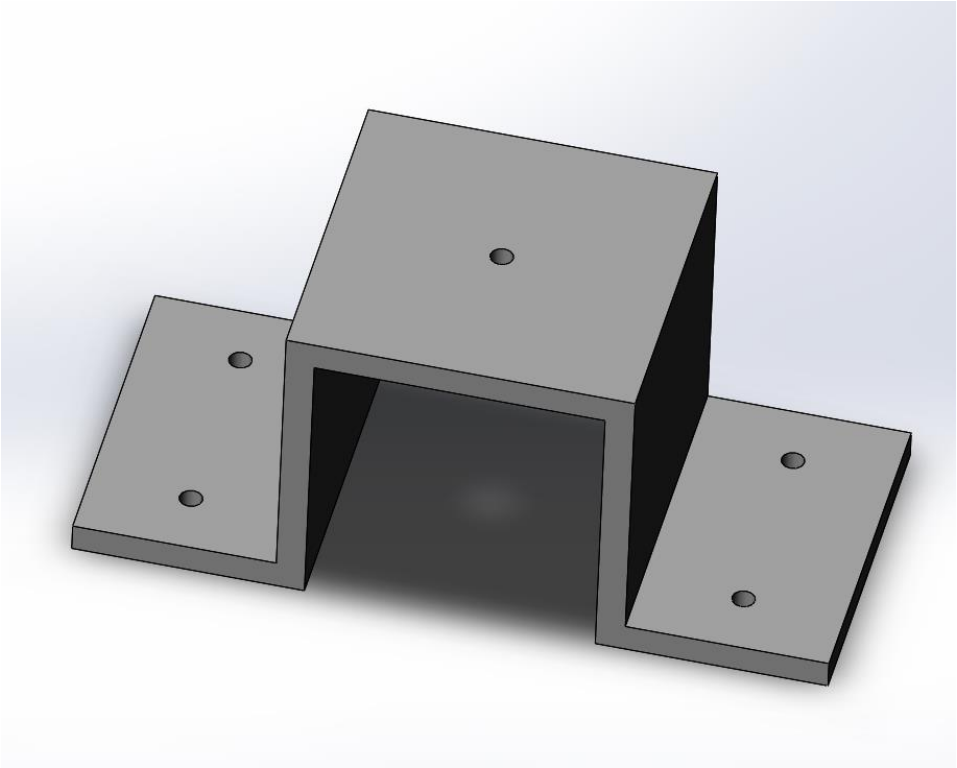
機構的設計是我跟葉蓁琢磨最久的部分，因為機構方面的問題不處理好，寫出再好的程式都會因機構的不穩定而白費。機構的部分主要分成支撐平衡系統和多連趕系統，以下就分這兩部分說明：

(1) 支撐系統：

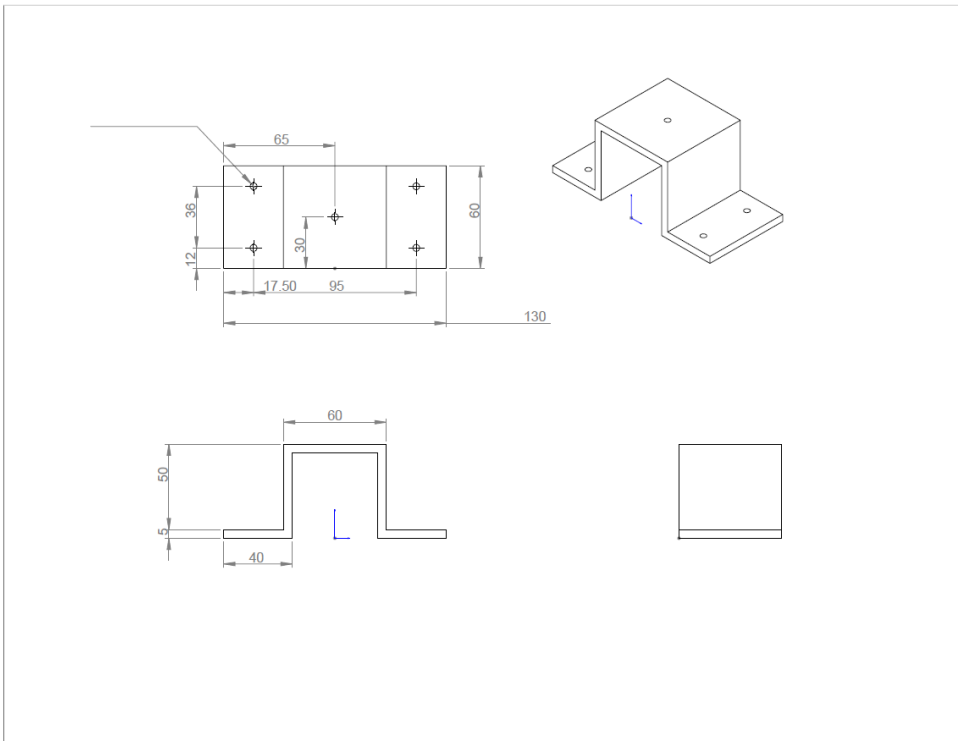



我們用了L型角鋼來讓平衡軌道保持水平，透過兩支螺絲建立系統的平衡軸。第一支短螺絲是作為搖擺桿，螺帽鎖的位置要很精確，太緊或太鬆都不行，太緊的話，系統左右搖擺就會受限制，太鬆的話，系統不只會左右搖擺還會前後搖擺，這在Control的Block Diagram裡面是雜訊的部分，會影響系統整體的穩定性。第二支螺絲是支撐桿，原本有想用其他的物品代替螺絲，後來想一想螺絲可以用鎖的還是最穩的方式。支撐桿底下連接的是冂型支撐架，這個支撐架是我們自己用solidworks畫的，再交由外面的加工廠代工。

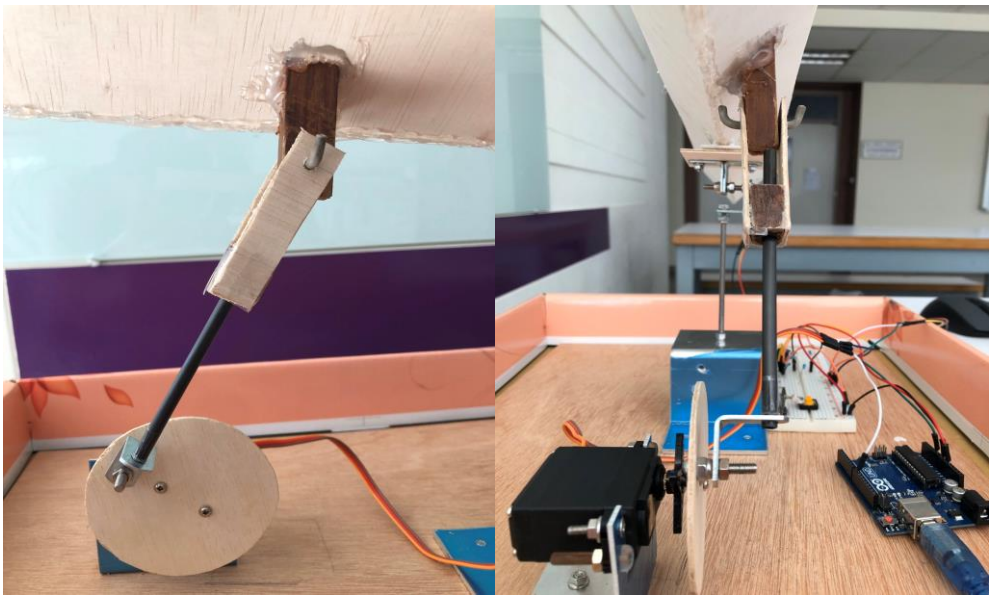
ㄇ型支撐架的3D圖:



ㄇ型支撐架的工程圖:

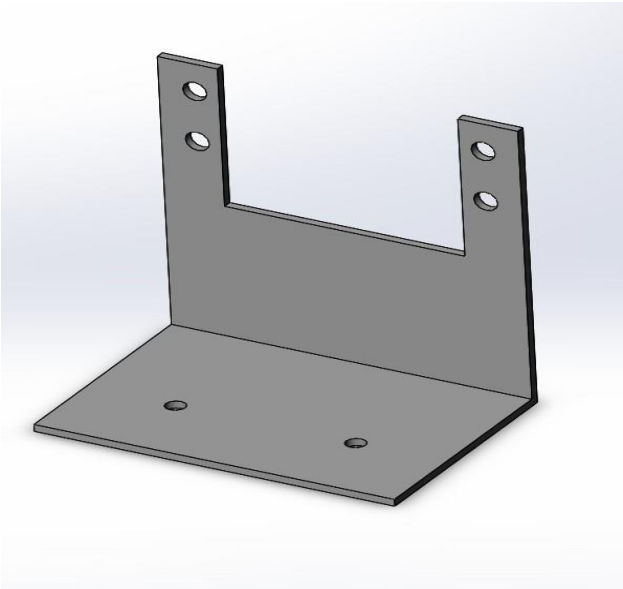


(2)　多連桿系統：

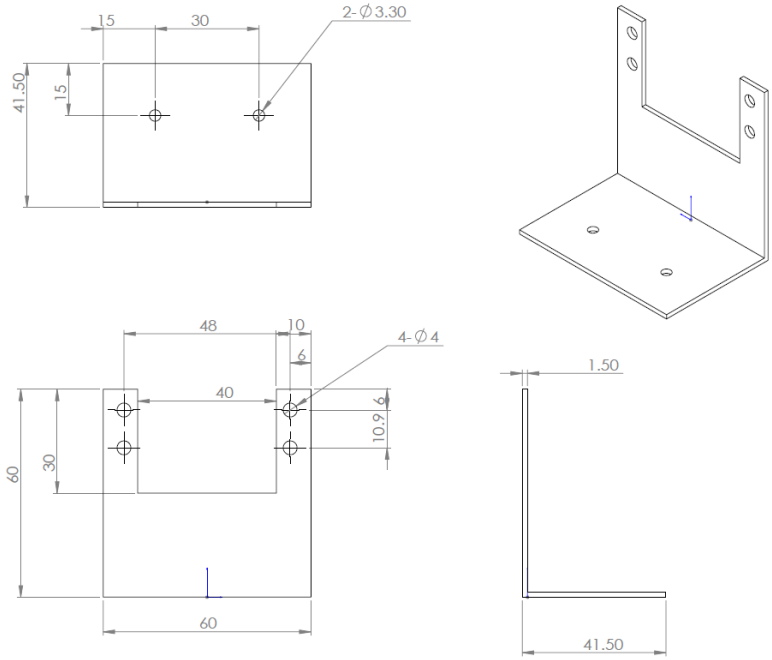


使用多連桿系統的目的：將馬達的旋轉運動，轉成碳纖棒的直線運動，再進而推蹺蹺板。

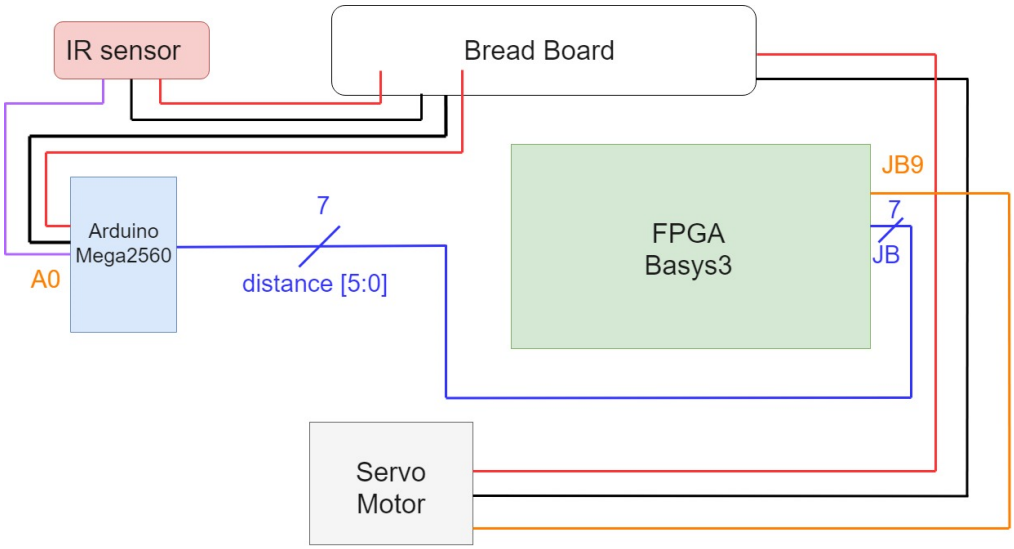
這個系統連接伺服馬達和軌道，但是馬達的輸出角度怎麼等比例的交給軌道，一直是我們必須去克服的難題。首先我們將一片圓形木片鎖在馬達的轉盤上，來放大達所轉的弧長，讓軌道的可變化角度更大。接下來鎖一個L型角鋼在轉盤上連接多連桿來控制軌道的擺動。軌道連下來有一塊木條，用兩片木片和一個軸來形成多連趕的轉軸系統。最後是馬達架的部分，我們原本要買現成的，但是太貴了而且大多不符合我們的使用需求。因此我們就稍微看了servo motor的data sheet之後自己幫他設計一個馬達架。馬達架的3D圖：



馬達架的工程圖：



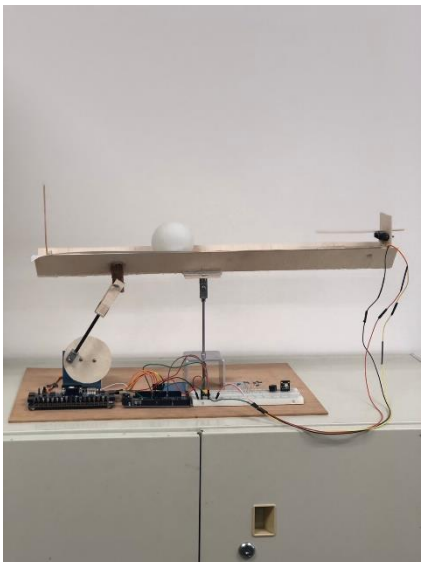
Schematics :

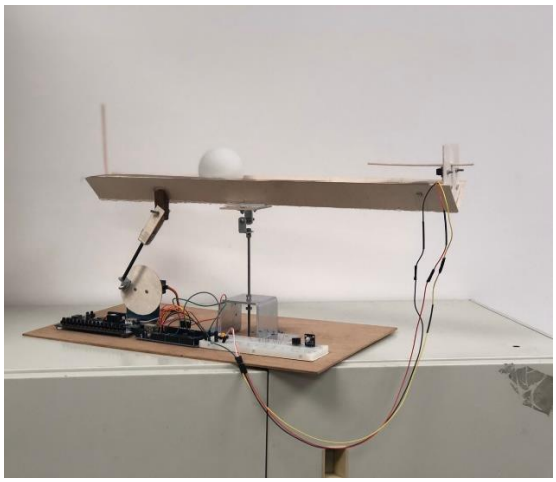


4. Results

(1) Final Assembly:

所有機構組裝完成後的總組合圖如下：(正視圖 & 等角視圖)





(2) Effectiveness Assesment:

最後試驗的結果：在不斷的調整參數以及修改code後，平衡架是可以將球平衡的，只是隨著球干擾的程度不同，需要的平衡時間不一樣。有時候不小心會把球停在偏差較大(約5公分)的位置，我們也還在想原因。因為時間不夠長，所以其實還有很大的進步空間。但至少基本的平衡功能有成功，這是最欣慰的QQ。

5. Discuss

迫於時間有限，demo的時候只能以目前的狀態，其實我有想到幾個問題，我們可以繼續改善的部分：

(1) 能否不用Arduino 傳資料，全用FPGA去讀取紅外線sensor的資料呢？ 應該是可以的，只是可能要完全了解那個sensor運作的原理，像超音波sensor那樣，才有可能寫出他的module去讀值。

(2) PID是平常廣為人知的控制理論，但它也有它的限制，我們的控制老師其實很不喜歡人家隨便用PID然後亂調kp,ki,kd三個參數的值去逼近理想結果(結果我們就再用這件事XD)也許未來有機會可以用到lead-lag control。今天聽到老師說可能要加個Q-learning，我沒有很了解reinforcement learning的理論，但有想過如果不只用控制，加入machine learning的方法，也許會讓成果更穩定。

6. 如果我們一維的做好，能否推展到2維呢？不過首先sensor就不能只靠紅外線了QQ，需要三軸加速陀螺儀等等的，然後機構方面也變複雜許多，再來code方面也需要好好思考。不過我覺得是相當有挑戰性又有趣的題目，未來有機會希望可以繼續研究!

7. 心得

葉蓁：

首先感謝這次的final project，讓我有機會實作一個PID控制的練習，也練習如何透過FPGA板子整合所有硬體和sensor。期末真的是太爆炸了，我這學期也才五門課，就有五個final project…(還有兩個還沒開始弄XD)，所以其實前兩周進度比我們proposal預期的大概再慢一半QQ。我們是星期三中午才考完所有的考試，才開始最後全力衝刺。主要的code之前有初步打好雛形，但星期三當天才正式決定要換測距的sensor，所以code方面真的滿趕的，加上硬體方面debug也de很久。這個project挑戰的地方真的滿多的，雖然看起來很普通，但光架設穩定的機構就修了滿多次的，剛開始用的超音波模組也花了不少時間debug才弄成功，興奮之餘卻馬上發現無法偵測球，宣告失敗(好挫折又可惜QQ但人生何嘗不是不停的失敗、檢討、再重新開始呢？)硬體部分確保它很robust後，大概星期四中午，我們開始全力衝刺code，從十進位換成二進位，計算pid的結果如

何mapping，到如何送給馬達控制。(其實中間有很多問題，初始pid合成的，馬達根本不動，不論球的位置多遠。不確定是因為sample period太長或甚麼原因。我們一直嘗試，直到demo前一天的晚上10點多，終於成功了!當下真的非常激動，就希望成功可以在當天demo給大家看。但demo當天不曉得為何，有點不太穩定QQ這是最遺憾的，加上我太緊張presentation講的有點亂，真是抱歉QQ
最後謝謝鈺智助教支援我們HDMI轉VGA接頭，否則熬夜趕的PPT就付諸流水了~謝謝助教們和老師看我們demo~雖然很爛QQ

劉多聞：

這次的final project一開始想不太到底要做什麼，因為怎麼樣想來想去都是一些遊戲和以前別人做過的東西。後來和葉蓁討論後就發現，原來我們可以把這學期學到的東西都結合到這次的final project裡面。也許我和葉蓁在純程式的部分贏不了別人，但我們可以展現我們雙專長的特色，把動機系所學到的東西結合進來。一開始遇到很大的困難是，我們參考的資料都是用arduino打的，要把它轉成verilog有一定的難度，所以我們的方法是先用arduino測試系統的穩定性，最後再把code換成verilog。這學期真的很爆炸，修了5門主科有5個final project，真的是快累死，到現在還有2個final project沒做完，好險剛好遇到投票，有些課的老師把final project的deadline往後延了很多，所以還算能分配好時間把事情做完。

這學期修邏輯設計實驗這門課的心得就是很充實且很實用，把過去很多不懂的觀念又重新學了一遍，把現在學得很抽象的東西都實做出來，謝謝老師的課程安排和用心教導，謝謝助教的用心解惑和耐心地等我們每次lab都demo完。

分工：

兩人一起把final project趕出來!