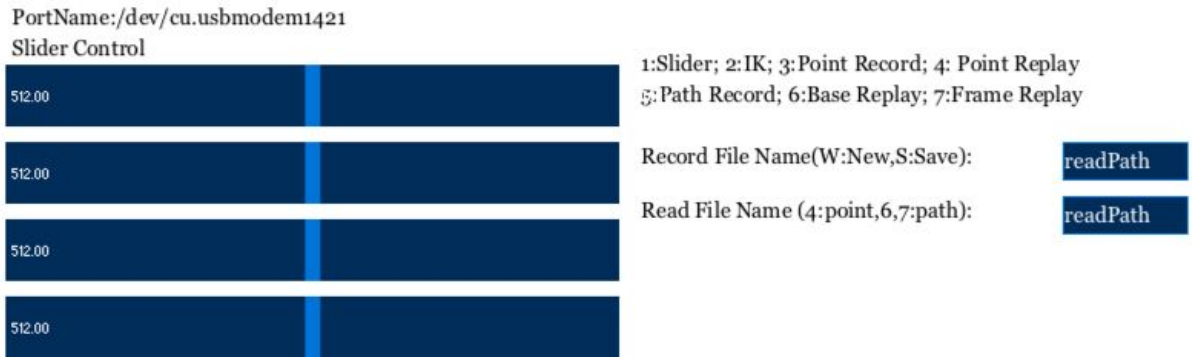
Stylus Assistant Instructions

Modes

在硬體鏈接無誤后，Arduino的Code也上傳到Mega后，打開Processing。
這份Code包括7種不同的控制馬達的模式，通過按鍵盤上數字的1-7來切換：

1. Slider Control，左右移動滑塊可控制單個馬達，用以最初的Debug;
2. IK Control, 這時候按住滑鼠左鍵，移動滑鼠就可以控制馬達移動;
3. Points Record Control, 記錄單個點,一般用于實驗；
4. Points Replay Control, 回放單個點；
5. Path Record Control，記錄畫筆的路徑；
6. DynaBase Path Replay, 用於DynaBase 路徑回放；
7. DynaFrame Path Replay, 用於Dyna Frame 路徑回放。



Processing Setting

1. 首先確認Processing 的PortName為Arduino Mega (比如dev/cu.usbmodem1421)，否則到Processing中的portName = Serial.list()[3] 中修改數字，數字參考Arduino菜單欄Tools->Port， 從0開始數；
2. PortName 下方為7重模式的狀態欄，切換數字鍵1-7來更改；
3. 右邊兩個輸入框為文件名填寫，以保存與讀取相對應的路徑。

Points Record and Replay

=== Points Record

1. 更改記錄文件的名稱 Record File Name；
2. 按數字‘3’，進入Points Record Mode；
3. 按‘w’新建一個一個txt檔案；
4. 點擊滑鼠到相對應的位置；按'n'記錄下一個點，重複該步驟；
5. 記錄結束后按's'保存文件，切記，否則文件為空。

=== Points Replay

1. 更改讀取文件的名稱Read File Name；
2. 按數字‘4’，進入Points Replay Mode；
3. 這時候馬達會直接開始跑到第一個點；
4. 按字母‘n’，可以走到下一個點；
5. 按字母’b’, 可以走回上一個點。

Path Record and Replay

=== Path Record

1. 更改紀錄文件的名稱，Record File Name；
2. 按數字’5’, 進入Path Record Mode；
3. 按字母’w’，新建一個txt檔案；
4. 按住滑鼠左鍵，移動滑鼠來畫出所需要的路徑，在记录Frame的路徑都適合，每一個筆劃結束和開始的時候，按住鼠標停頓1秒；
5. 畫完後按字母’s’，保存文件，切記，否則文件唯恐。

=== Points Replay

DynaBase與DynaFrame的回放操作步驟一致，但由於兩者的後3 4顆馬達的角度計算不同，故回放模式不同，Base為6，Frame為7，不可混餚；

1. 更改回放文件的名稱，Read File Name;
2. 按下’6/7’後，開始回放剛才的路徑。

Hardware Setting

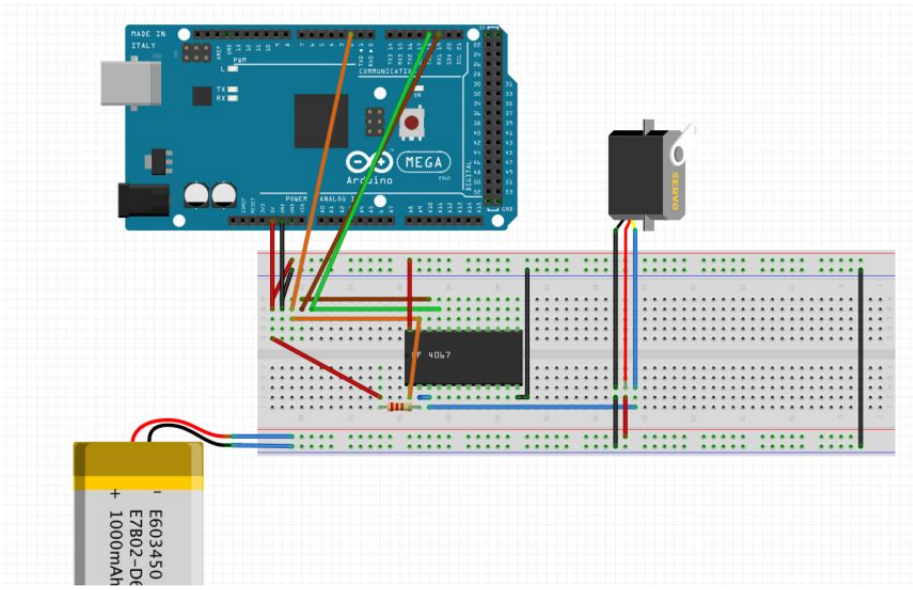
Reference

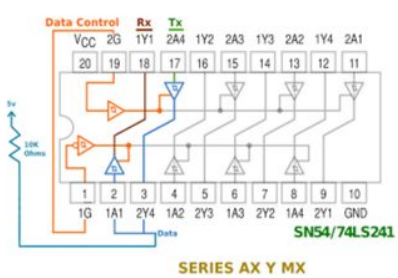
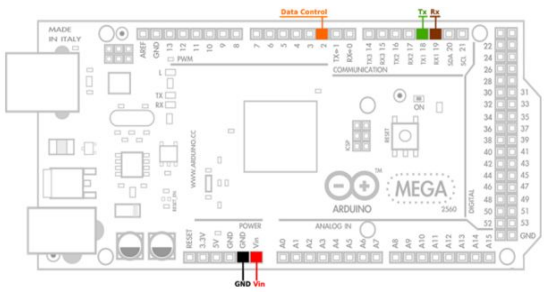
硬體的設置最早的參考是來源於這個鏈接，AX12A的原理與AX18A基本相同，這裡我們使用的鏈接方式為Dynamixel Serial1
http://savageelectronics.blogspot.tw/2011/01/arduino-y-dynamixel-ax-12.html

Hardware Lists

以下為物品清單，以及各個組件之間的連接方式，
如果是很多個馬達，就直接並聯馬達的pin腳。

- Arduino Mega 2560
- Dynamile AX18A
- Multiplexer 74LS241
- Resistance 10kΩ



Multiplexer 74LS241	Mega
	

Software Setting

Arduino

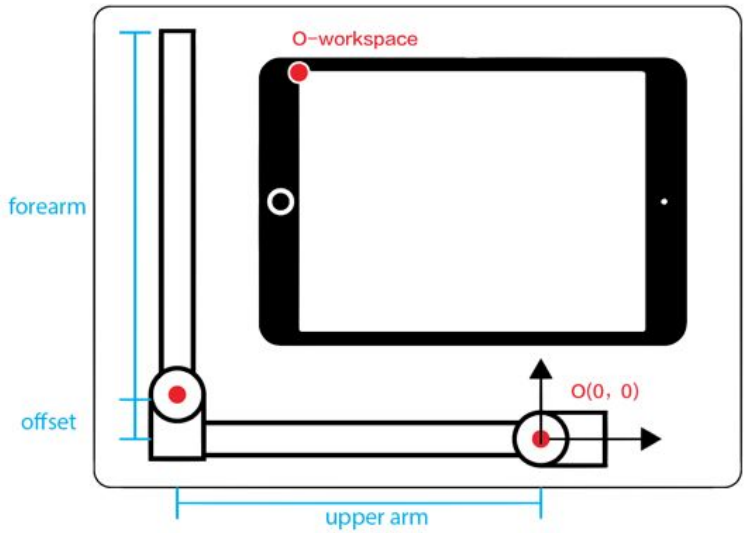
== Dynamixel Arduino Library Document
http://austinlpalmer.com/Projects/Documentr/#/home
== Library Download, [DynamixelSerial1.zip]
https://sourceforge.net/projects/dynamixelforarduino/files/?source=navbars

以下為Arduino控制Dyanmixel的Library，根據Document顯示，比較常用的幾個函數分別為

- setId (ID , newID)；因為會控制很多馬達，所以需要設置各個馬達的ID，出廠ID為1。這裡需要注意的是，在為馬達設置完ID后，請再馬達上做上相對的記號，免得之後忘記；
- ping(ID) 用來找馬達是否鏈接正確，1為找到，0為未找到；
- Dynamixel.moveSpeed(ID , Position, Speed) : ID為對應馬達的ID，Postion為馬達要轉到的位置[0,1023]，Speed為馬達轉動的速度[0,1023];
- Dynamixel.move (ID , Position) : ID為對應馬達的ID，Postion為馬達要轉到的位置[0,1023]
- var = Dynamixel.readPosition(ID) ; 讀取馬達ID的當前位置

== Code Review
MOTORNUM 馬達的個數
motorRange[MOTORNUM][2] 馬達角度保護區間，放置數據傳輸錯誤，防止馬達角度超過一定值，互相干擾卡在一起，保護馬達不會損壞
readEvent(Pos)：接受從Processing 那邊傳來的馬達角度
sendEvent(readPos): 將馬達當前的位置發送給Processing

Processing



===Parameters Review
PVector screen = new PVector(19.7, 14.9); //螢幕的長寬長度
在計算IK的時候，會把Processing的座標系原點轉換到機械手臂根部，如圖O點，並且水平方向做鏡像翻轉，從而得到新的座標系；
PVector wsOrig = new PVector(-12.4, 20.3); // workspace original（cm)
float uppL = 19.3; //上臂的長度(upper arm)
float mdL = 2.55; //中間的長度(offset)
float downL = 20.3; //前臂的長度(forearm)
byte motorNum = 4; //馬達個數，應該與Arduino一致
float cmTopx = 25; //單位轉換 Cm to Px
PVector orignal = new PVector(600, 750); //新座標系原點在Processing座標系中的位置

===Fuctions Review

Processing 中分為7個主要的文檔，以控制不同的模塊：

- DyanUI為主函數；
- Gui 為Processing上面的UI；
- GuiScript 為Processing控制UI組件的函數，以及快捷鍵的控制；
- IKFuc 為IK計算過程中的函數；
  - IKControl(x, y, upperArm, offset, foreArme)，輸入Processing內的座標系，得出馬達1與馬達2的角度float[-300,300]
  - angleMap，將角度轉化為馬達可以讀取的數值int[0,1023]
- TangentCalculate 用於計算Base的曲線切線角度與Frame的z軸高度；
  - TangentAngle()，計算出DynaBase第三個馬達的角度值；
  - AddZDimension()，插入出DynaFrame第三第四顆馬達的高度值
- TxtIO 為Txt文檔的讀取；
- serial_IO 為Processing與Arduino溝通的函數。