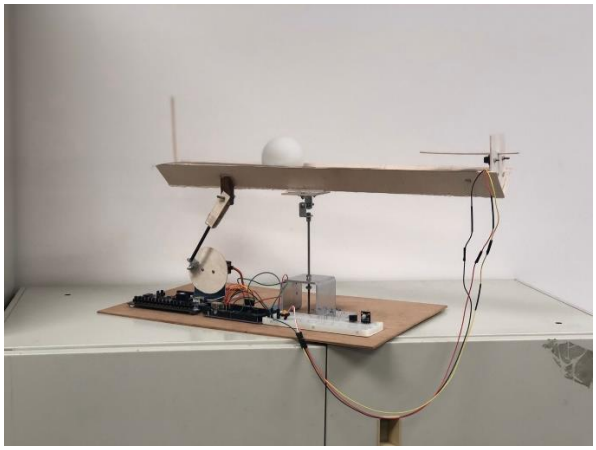
Embedded System Design

Final Project Report

PID Balance Ball and Beam

106030009 葉蓁



1. Goal

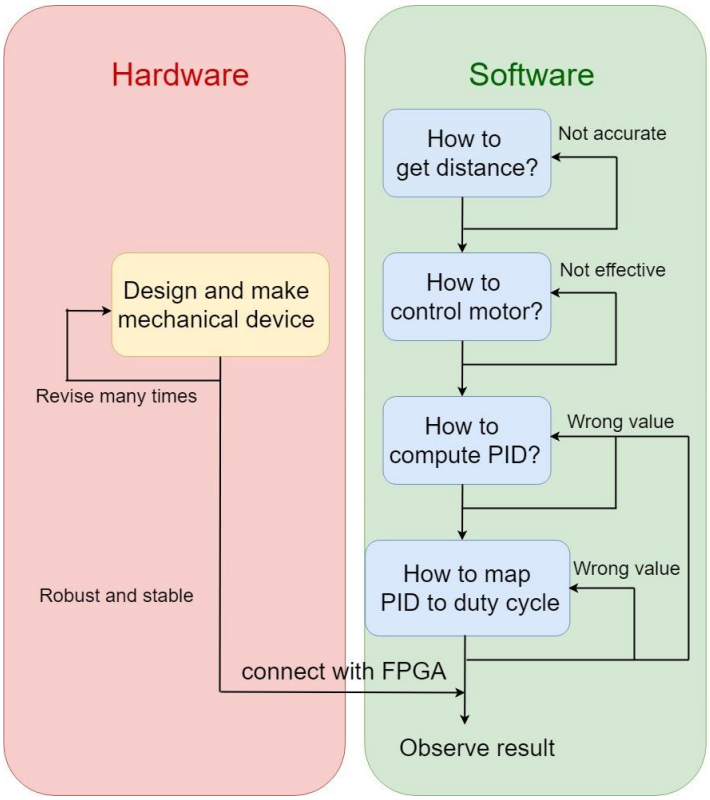
我製作一個一維的平衡球和支架，目的是：

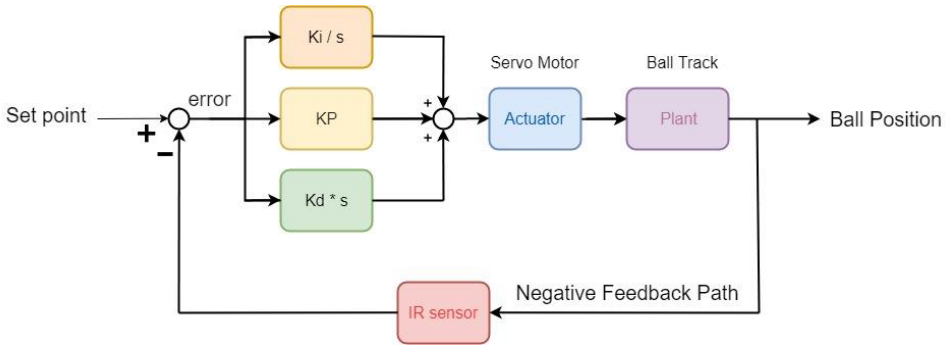
平衡樑要能夠讓球停在指定的 setpoint。

Operating Philosophy:

Distance sensor get ball position(Input) -> Tell Arduino and compute the PID value-> Tell servo motor to move and try to stabilize the ball.(Output)

My design flow:



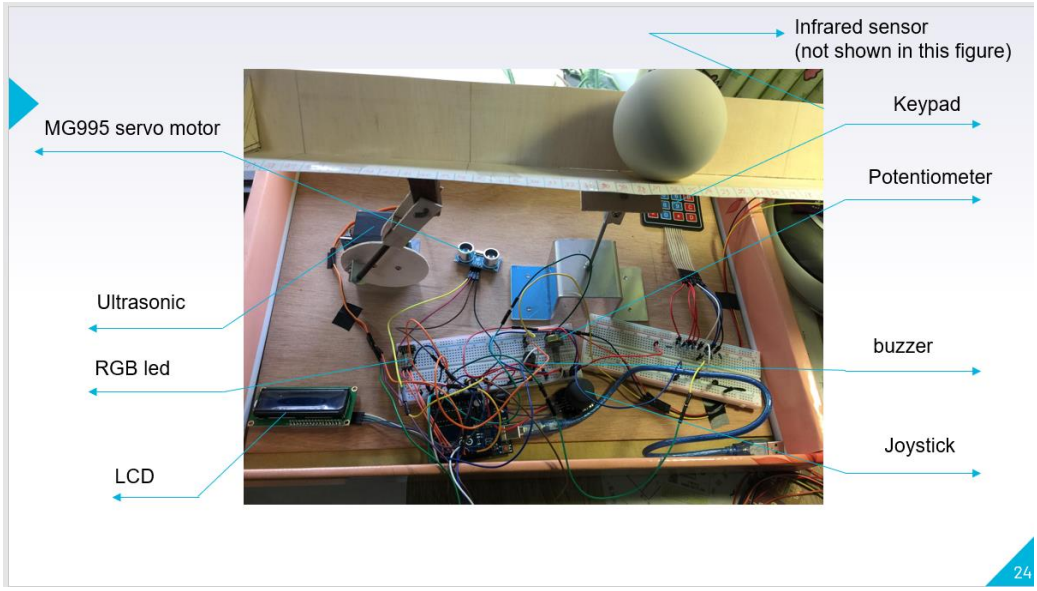


簡單解釋一下：受控場(plant)是平衡架，由伺服馬達(actuator)驅動它，得到的 output 是球的位置(Ball Position)。藉由一個 IR sensor 去讀取 output，然後回授到 input 固定的位置(set point)，兩者相減得到一個誤差值。這個誤差值如果最簡單的控制方法，就是用一個單純的 P gain 去放大然後送給驅動器，讓他去追命令，減少誤差。這種只有一個放大倍率的控制叫做 P control。但這種簡易的控制器有它的限制，可能會有太多的過衝量(overshoot)，或是因為 system type 是 0 的原因，即使 input 只是一個 step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

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。

2. Component Used (Schematics)



(1) Component:

a. Electrical components.

Sensors :

SHARP GP2Y0A21 YK0F infrared distance sensor(10-80cm)

Joysticks

10KOhm potentiometer

Actuators:

MG995 Servo motor

Liquid crystal display

DAC:

RGB led light

Buzzer

MCU:

Arduino UNO R3 *2 (one for program, one for power supply for servo)

MSP430G2553S *1

b. Mechanical components.

Several M3, M4 screws and nuts.

一些鋁的沖孔版(支架用)

碳纖棒

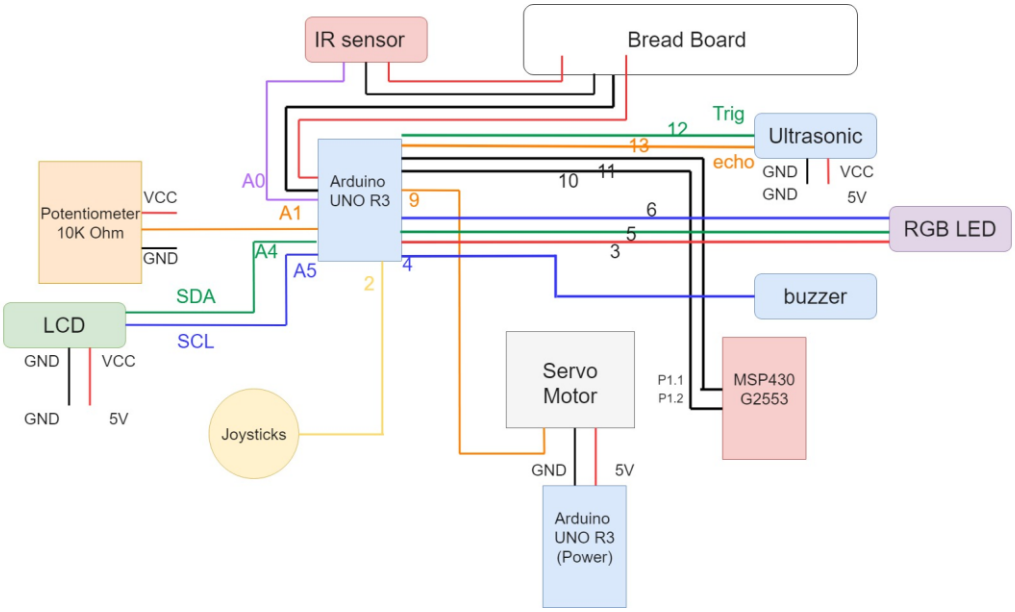
木片、木頭少許

冰棒棍

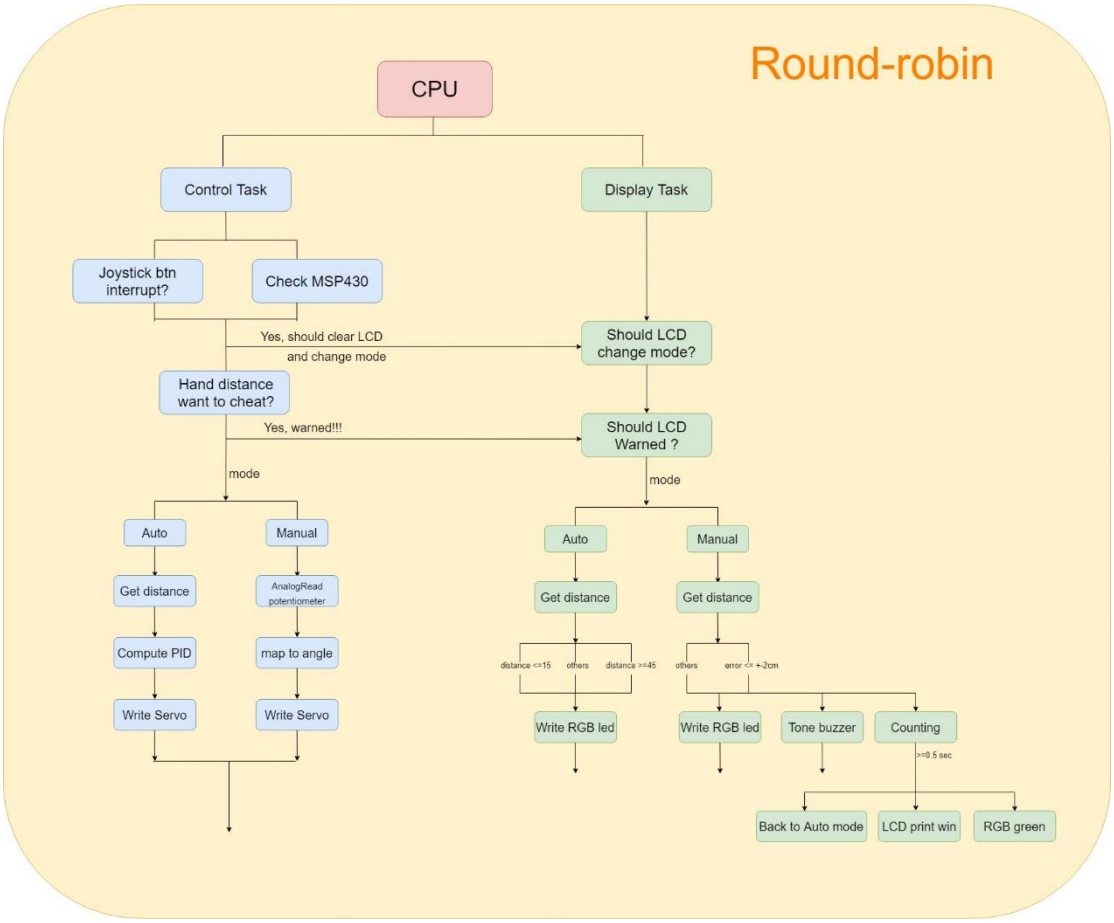
膠帶、電火布、熱熔槍、焊錫

Components	Type	Name	Pupose
Electrical	Sensor	Infrared sensor	Get ball distance
	Sensor	Ultrasonic	Detect user hand
	Sensor	Joystick	Switch mode
	Sensor	Potentiometer	Manual mode control motor
	Sensor	MSP430	Switch mode
	Actuator	MG995 Servo motor	Adjust beam angle
	LCD	LCD	Mode/Distance detail / Warning
	DAC	RGB LED	distance hint
	DAC	Buzzer	Warning / Win signal
	MCU	Arduino UNO R3	Main program
	MCU	Arduino UNO R3	Power supply for motor
Mechanical	Fasteners	M3 screws	Stabilize the structure
	Fasteners	M4 screws	Stabilize the structure
	Fasteners	M3 nuts	Stabilize the structure
	Fasteners	M4 nuts	Stabilize the structure
	Fasteners	Aluminum sheet meta	Motor and beam support
	Mechanism	wood	Beam structure

(2) Schematics: (How I connect everything with Arduino board)



3. Program Flow



我用 FreeRTOS 寫，宣告兩個 Task,分別是 controlTask 跟 displayTask。任何跟控制 Beam 相關的程式都寫在於 controlTask 裡面，跟 display 相關的都放到

displayTask 裡面。兩個 Task 的 priority 都宣告成 1，CPU 排程是用 Round-robin scheduling。記憶體分配分別是 160 跟 128byte。

(1)Control Task： 每次進來這個 Task，都會先跑兩個 function：

```
Check_Btn_from_MSP430();
Your_hand = Hand_distance();
```

第一個 function 是確認 MSP430 有沒有按按鈕，如果有按會換 mode。

第二個 function 是讀取距離超音波感測器的距離，如果手很靠近(想作弊)就會出現 Warning 的訊息，叫你不可以作弊(buzzer 會響)。

跑完這兩個 function 後，有個 if_else 判斷進哪個 mode。所以我

#define Auto 0

#define Manual 1

如果是自動(Auto)模式，平橫樑會自己依照球的誤差去平衡，使球回到 setpoint。

如果是手動(Manual)模式，使用者用 10K 歐姆的可變電阻去平衡球的位置。如果在可接受誤差內(正負 2cm)維持超過一定秒數，機器會告訴使用者你成功了，然後回到自動模式。

至於如何切換模式，有兩種方法，可以按 Joystick 的 sw，也可以按 MSP430 的 Button。可以各別按一下回到本來的 mode。E.x.: 預設 Auto mode，按一下 Msp 進入 Manual mode，再按一下 Joystick 變回 Auto mode。兩個按鈕不會衝突。Joystick 的 btn 偵測是用 Interrupt 寫的。

說明 PID 實作方式：附上部分 code 截圖：

```
119     if(mode == Auto){
120             if (millis() > time+period)
121             {
122            time = millis();
123             distance = get_dist();
124               //Modify minor distance error.
125 //              if(distance>=35 && distance <38){distance = distance-3;}
126 //              if(distance>=38 && distance <44){distance = distance-4;}
127 //              else if(distance>=44){distance = distance-10;}
128
129             distance_error = distance_setpoint - distance;
130             if(distance_error>=-2 && distance_error <= 2 ){
131               PID_total = 80; //Stop.
132             }
133             else{
134                 PID_p = kp * distance_error;
135
136                 dist_difference = distance_error - distance_previous_error;
137                 PID_d = kd*((dist_difference)/period);
```

如果是 Auto mode，準備要進來平衡。當超過 sample 的週期(period 我設

50ms)重新跑一次流程：

(1) 取得距離(get_dist)

(2) 可能要修正距離(因為我的 sensor 在大於 30 公分後會有不定的誤差，我用人工修正方法讓他更貼近真實距離

(3) 誤差 = SetPoint – distance

(4) 如果誤差小於正負 2 公分，視為平衡，讓馬達維持在水平靜止

(5) 如果誤差大於正負 2 公分：計算三個 PID 值：

P : Kp * error

D : Kd * (error – last_error)/period

I : 如果距離 setpoint 小於 10 公分或大於 5 公分(這部分要由做實驗得到你球跟馬達配合的結果來決定)，讓 Integral 項作用。

Ki * error + 上一次的 I term。(積分效果)

否則 I 項為零。

(6) 將三項加起來得到總 PID_total

(7) 將有效的 PID 值 mapping 到馬達轉的角度(0~150 度)

Mapping 這邊也要靠 trial and errror 來嘗試，最為困難。

(8) 剔除極端值後，讓馬達轉

(9) 更新上次的 error 為現在的 error，準備下一次的 sampling。

```
136                 dist_difference = distance_error - distance_previous_error;
137                 PID_d = kd*((dist_difference)/period);
138
139                 if(-5 < distance_error && distance_error < 10)
140                 {
141                   PID_i = PID_i + (ki * distance_error);
142                 }
143                 else
144                 {
145                   PID_i = 0;
146                 }
147
148                 PID_total = PID_p + PID_i + PID_d;
149                 Serial.print("PID befor mappeds: ");
150                 Serial.print(PID_total);
151                 Serial.print(" ");
152
153                 PID_total = map(PID_total, -150,150,0,150);
154             }
155
156             if(PID_total < 20){PID_total = 20;}
```

```
                Serial.print(PID_total);
                Serial.print(" ");

                PID_total = map(PID_total, -150,150,0,150);
            }

            if(PID_total < 20){PID_total = 20;}
            if(PID_total >150) {PID_total = 130; }

            myservo.write(PID_total);
            Serial.print("PID: ");
            Serial.print(PID_total);
            Serial.print(" ");
            Serial.print("Distance: ");
            Serial.println(distance);
            distance_previous_error = distance_error;
          }
    }
```

(2) Display Task :

任何跟顯示有關的(LCD, RGB,buzzer)都用這個 Task 控制。以下分用這三個元件說明：

LCD：用 mode 分要印甚麼。第一行會印是甚麼 mode(Auto mode：)，第　二行會印距離幾公分(e.x. 22 cm)。如果有換 mode 的話，才會把 lcd.clear()，第一行會洗掉重印，這樣才不會太頻繁的 clear 導致螢幕閃爍。

另外如果有 cheat 行為，Lcd 會警告，印出”Warning! Do not Cheat”來告訴使用者。在 Manual mode 如果你進入平衡範圍，他會告訴你”You balance”，成功平衡後會印出”Congratulation”，然後回到 Auto mode。

RGB：用 mode 來分，

Auto mode :

球大於 45 公分的話，會亮紅燈；小於 15 公分的話，會熄滅。中間距離想用成比例關係亮燈，如下：

$$analogWrite(redPin, distance * 5);$$
$$analogWrite(bluePin, distance * 5);$$
$$analogWrite(greenPin, distance * 5);$$

Manual mode:

球在 balance 區內會亮藍燈，這個區間以外則不亮燈。

Buzzer：不論在哪個 mode，如果超音波偵測到人手想要 cheat，buzzer 會教出高頻的警告聲。

另外在 Manual mode，如果玩家讓球停在平衡區範圍內，buzzer 會發區低頻聲表示現在是在平衡區，超過指定時間，會發出一段旋律表示你成功了！然後又回到 Auto mode。

兩個 Task 寫好之後，loop 沒有東西，記憶體調整一下，原本我都只给 128，結果 controlTask 不太夠，所以調大一點，就可以了!

4. Difficulties

我整理以下幾個困難點：

(1) 機構設計：這個機構最困難的地方是：如何將伺服馬達旋轉的運動，轉為直線運動，再轉成推動蹺蹺板(木軌道)的運動。這會需要一個多連桿系統。從側面看過去，碳纖棒盡量要垂直於木片，才不會有側向剪應力，造成應力集中機構毀損，或是馬達做無效的力量傳送。

(2) 調整 PID 參數：PID 有三個最重要的參數：KP、KI 跟 KD。調整的順序為，先調 KP，再加 KD，最後加 KI。但除了這三個參數之外，計算後的 PID_total 如何 mapping 到讓伺服馬達動的 Servo.write()裡面也是一大挑戰。要先觀察自己馬達尚未 mapping 前的初始值，PID_total 的範圍是多少，剔除極端值後，選出有效範圍，然後做 mapping。Arduino 很方便，有內建的 map API 可以叫，不過你的 Domain 和 Range 要設有效範圍，才會讓平衡裝置成功。我光調整這個 mapping 就花了不少時間。調完還要回去看 KP, KI, KD 三個值可不可以，頗花心力的。

(3) 讀取距離的精準度：由於這是一個一維的平衡裝置，單靠距離就可以做到平衡，但也因為這樣距離的精準度變得相當重要。原本我是用常見的超音波感測器，但對於球狀構造，波可能會繞射或產生其他非預期行為，我後來查了資料發現紅外線比較精準，但解析度沒有超音波那麼好(只能讀到 cm 等級)。有時候如果接地線鬆開，量測距離的 sensor 會瞬間壞掉(把 50 多公分顯示成 9 公分等)讓我非常緊張。這是我實驗過非常多次後得到的結論。未來如果這個 project 要做的更強健，可能不能用杜邦線，要直接焊接以減少接觸不良的問題。

p.s.這個紅外線模組初始沒有杜邦接頭，也是我自己焊接的 XD 生平第一次焊電線 QQ

5. Discuss & Feedback

關於我的 project 我想到了三個問題：

(1) 可以動態(不用改 code)就改變 setpoint 位置嗎?

我有想到用 keypad 去寫入，但不曉得因為我用 FreeRTOS 的方式寫，memory 會有不夠的問題，導致這個目前還沒做出來。如果用 mega 或更大記憶體的板子，也許可以成功。

(2) 可以展延成 2D 的平衡板嗎？

是可以，但一來機構上會變得很複雜(二維彼此不能干涉)，軟體上也變得相當複雜，因為首先：Sensor 不只是用測距離就可以解決，必須用三軸陀螺儀去偵測，這個模組的 code 應該也相當複雜，需要一些時間研究。不過既然網路上查的到相關成功的影片，表示一定做得出來。只是我時

間跟經費不夠。

(3) 可以用更好的控制方法嗎？

如果我得到精準的 Transfer function，也許我可以嘗試用 lead-lag 控制方式。不過這目前超出我的能力範圍 XD 期許未來自己有機會可以實現更好的控制器。

這學期辛苦各位助教了，我真的感謝無比，把我手把手從超級廢帶起來，沒想到自己可以修完這門課! 雖然現在還是很廢，期許自己可以帶著嵌入式學到的 Coding Style 和 Debug method 在未來任何地方解決問題! 謝謝助教，辛苦了!