


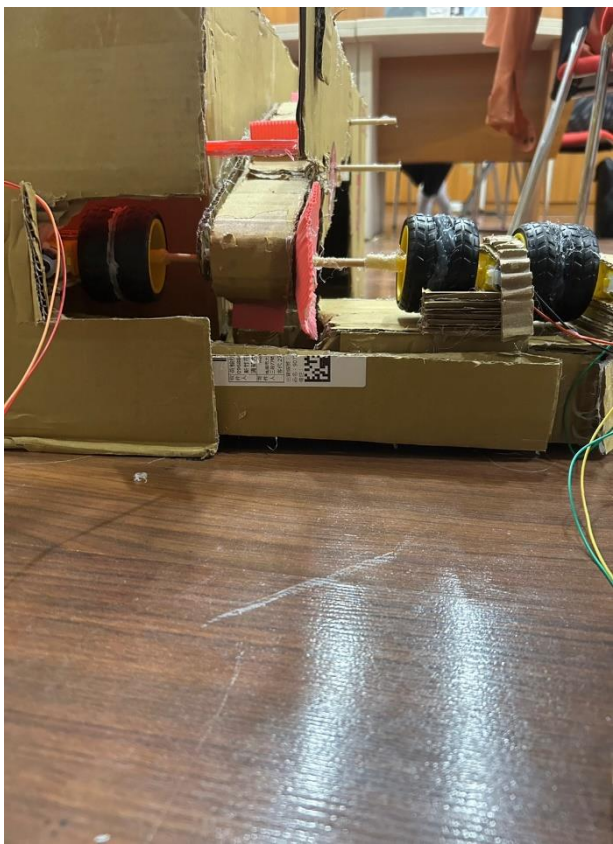
Team No: 27	Team Name: CTH 就是帥
Project Title: 寶0球	
Name: 蕭以勝	ID: 110070016
Name: 楊立慈	ID: 110011138

## A. 設計概念

我們做出了迷你保齡球機，以兩人競賽的方式進行，總共比三局，總分高的獲勝，當第一位玩家投完的時候，要按按鍵，換第二位玩家進行，同時我們將把競賽資訊顯示於螢幕上。保齡球投出時會有音效，擊倒保齡球會有音效並且可以計分，當按下切換玩家後，球瓶會自動歸位，我們會想在各個球瓶下接線，要歸位的時候，用馬達將線回收，這樣就可以讓球瓶重新站起，我們也希望利用輸送帶做出自動偵測球並自動回收球的功能，讓球可以自動回到投球的地方。

## B. 架構細節與方塊圖

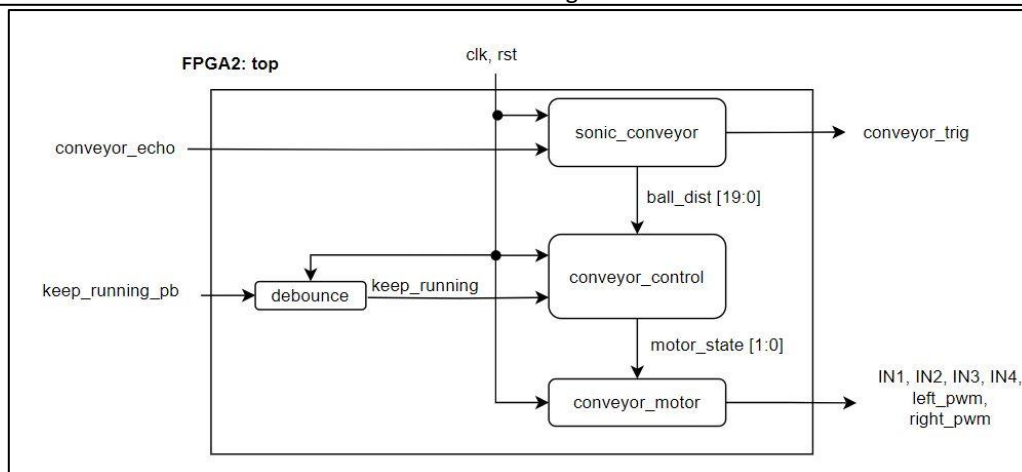
### (a) FPGA2: 輸送帶

輸送帶	馬達
	
上保護	下保護



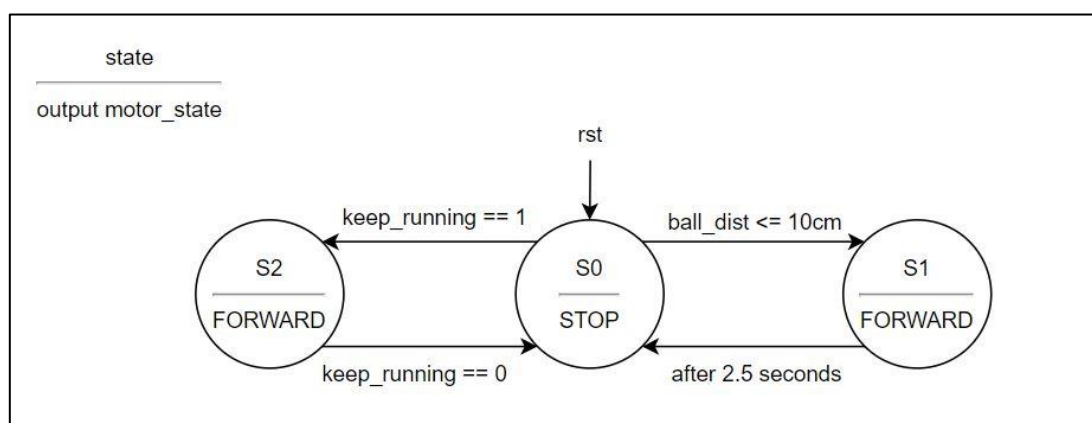
我們在輸送帶上方架設超音波去偵測有沒有需要運送。輸送帶的部分我們使用紙板製作，輸送帶上下有防護履帶跑出輸送帶的裝置，上面的防護裝置是熱熔膠，下面的是塑膠片，比較特別的是下面的塑膠片，我們將塑膠片的外緣往外搬，避免外緣會卡到履帶，防護裝置都要上潤滑劑避免摩擦力的問題，動力的部分我們使用三個馬達串聯才可以提供足夠的扭力，電源使用變壓器提供。

下圖為 FPGA2 top module 的 block diagram，其中所有 module 都是由 100MHZ 的 clk trigger。Sonic\_conveyor module 會輸出 conveyor\_trig 訊號去 trigger sonic sensor，並獲得回傳的 conveyor\_echo 作為輸入，並根據其計算目前球的距離 ball\_dist。Conveyor\_control module 的輸入為 ball\_dist、debounced 過的 keep\_running 按鈕訊號，並根據此計算輸出目前所需的馬達狀態 motor\_state。Conveyor\_motor module 再根據 motor\_state 去輸出相對應要給左右馬達的訊號: IN1, IN2 控制左馬達的旋轉方向、left\_pwm 控制左馬達的轉速、IN3, IN4 控制右馬達旋轉方向、right\_pwm 控制右馬達轉速。(code 中使用的 7-segment、LED 訊號為 debug 用的，因此沒有畫於 block\_diagram 內)



Sonic\_conveyor、conveyor\_motor modules 都是使用 lab8 寫過的 code，我們修改的部分為在 sonic\_conveyor 內將 ball\_dist 的 reset 值改為較大的值以避免一開始 reset 後會誤啟動輸送帶、conveyor\_motor 在 FORWARD motor\_state 的速度設為 100%。

Conveyor\_control 基本上為一個 Moore Machine，其 state diagram 如下圖。在 reset 後會來到 S0 等待進一步指示，在此 state 馬達要停止因此輸出 STOP。在 S0 如果檢測到球落到輸送帶上，即 ball\_dist 小於等於 10cm (球落到輸送帶上時，距離超音波模組的最大距離)，則會來到 S1 並讓輸送帶運轉、輸出 FORWARD，啟動過了 2.5 秒(把球送到頂端大約所需時間)後再回到 S0 等待。在 S0 如果檢測到玩家按下 keep\_running button 則會進入 S2 並讓輸送帶運轉、輸出 FORWARD，直到玩家放開 keep\_running button 再回到 S0。S2 這個 state 主要是給我們輸送帶的 boundary case 用的，當輸送帶上的檔板不小心轉到會卡住球滾落的位置時，我們可以按住 FPGA 的 btnUP 按鈕來運轉輸送帶直到我們放開。



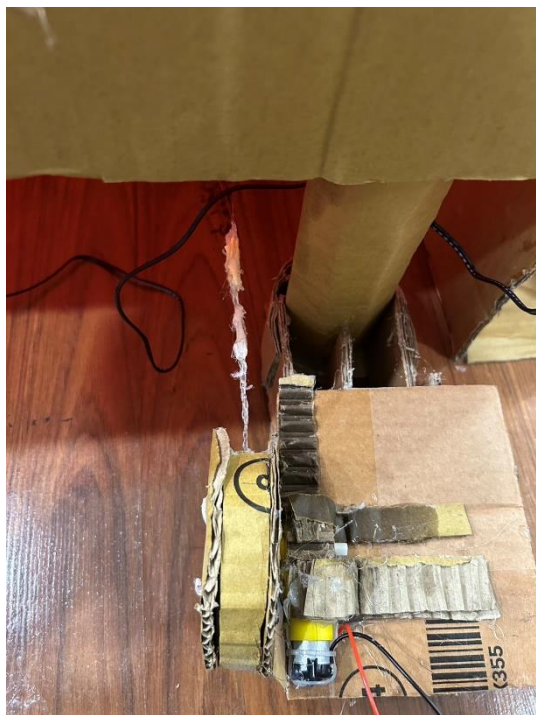
## (b) FPGA1: 球瓶自動歸位、計分

### 捲線機

捲線機鬆開

捲線機拉緊

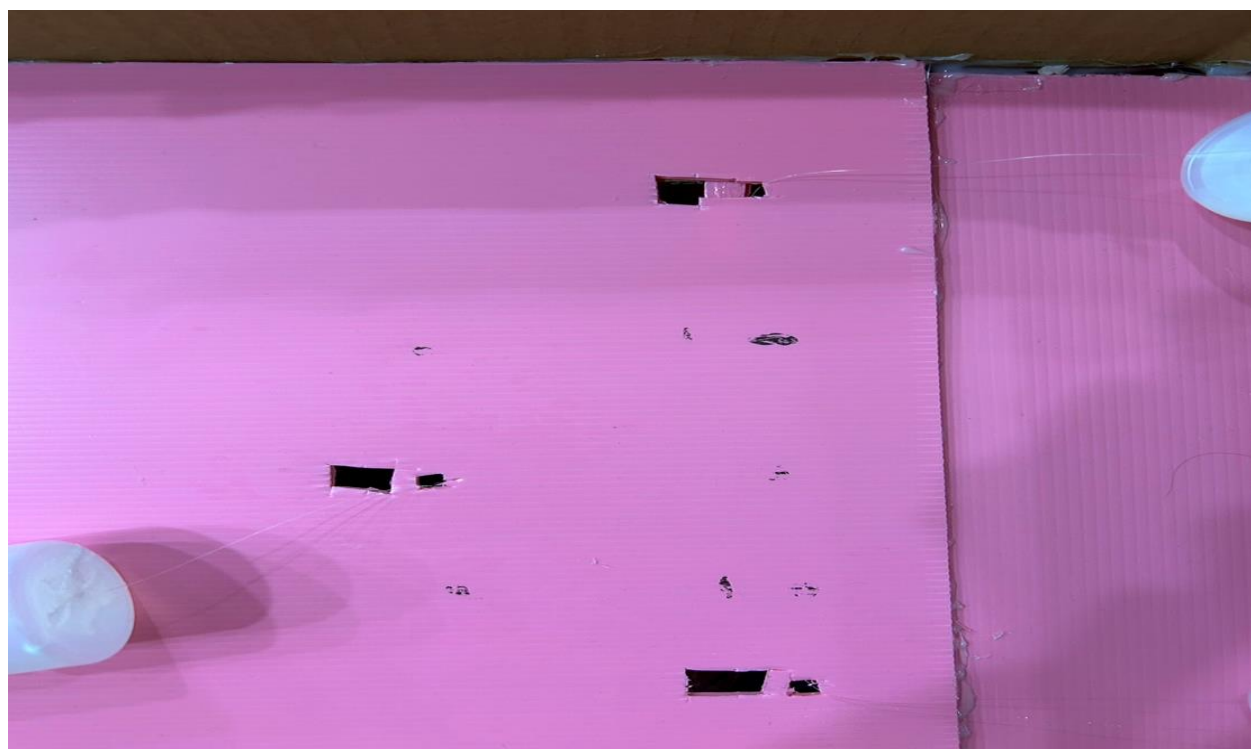




我們製作出自動捲線機當切換玩家玩家，動力是由馬達控制，電源使用變壓器提供，每個瓶子底下都有黏尼龍線，並將三條線用熱熔膠黏在一起，確保捲線機可以同時拉緊三個瓶子。

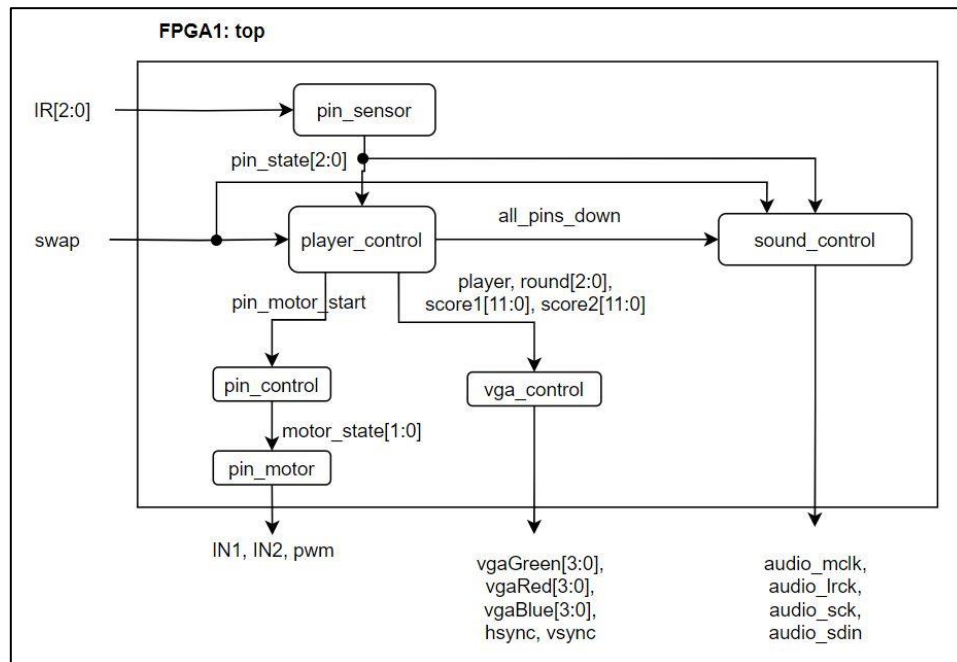
### 紅外線計分

#### 紅外線



我們將紅外線貼在上圖的三個洞裡，去檢測球瓶有沒有倒。

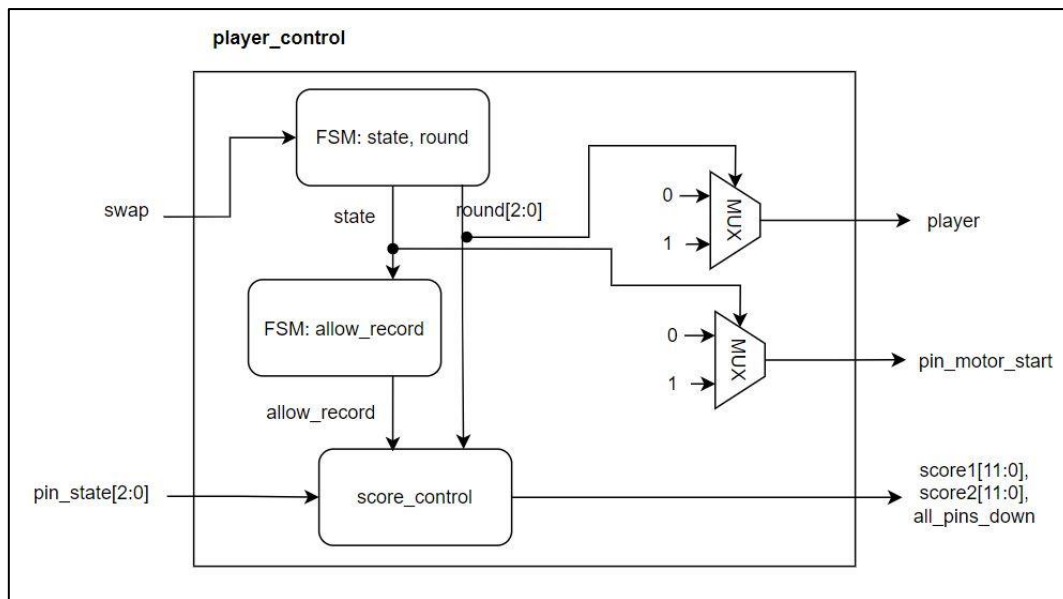
下圖為 FPGA1 top module 的 block diagram，圖中所有方塊都有 clk、rst 訊號為求簡潔省略沒畫，7-segment、LED 主要是 debug 用因此也省略，swap 為經過 debounce、one\_pulse 處理後的訊號。



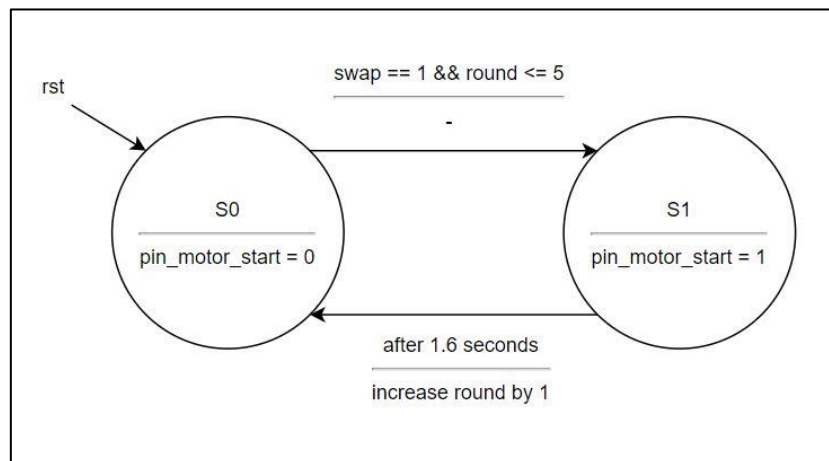
首先紅外線的三個輸入會經過 pin\_sensor module 處理過後輸出成 pin\_state[2:0] 代表 3 個 pin 的狀態，某個 pin 是 1 代表該球瓶還沒被撞倒、0 代表該球瓶倒下。

Player\_control module 是遊戲最主要的控制 module，負責記錄回合數、目前玩家、玩家各自 3 個 round 的分數、trigger pin\_control module 去將球瓶歸位。Player\_control 的輸入為切換玩家的 swap button、球瓶狀態 pin\_state，並根據此計算並輸出目前玩家 player、回合數 round(值為 0~5，偶數代表玩家 1 回合)、玩家 1 分數 score1、玩家 2 分數 score2 (NOTE: score1[11:0] 其實是 3 個 4-bit 的分數接起來的，即 {P1 round1 score, P1 round2 score, P1 round3 score}，score2 也一樣)、驅動收線機的訊號 pin\_motor\_start。

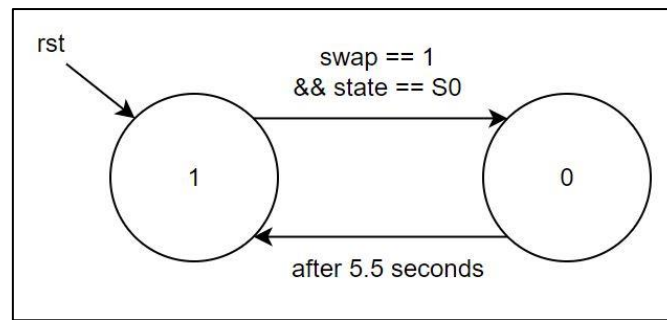
Pin\_control module 會根據 pin\_motor\_start 訊號去計算目前的馬達狀態 motor\_state，並傳給 pin\_motor 去輸出控制馬達訊號 IN1, IN2, pwm。Vga\_control module 會將輸入的遊戲資訊顯示到螢幕上。Sound\_control module 則會根據球瓶的擊倒狀態去輸出聲音。



上圖為 `player_control` 的 block diagram，其中各個長方形方塊都有 `clk`、`rst` 訊號。主要有 2 個 FSM: `state`、`allow_record` 的，`round` 為跟著 `state` 去更新的 counter 因此寫在一起。`Allow_record` 訊號是給 `score_control` module 提示當前是否可以記錄分數的訊號，如果當前球瓶正在被拉起則 `allow_record` 是 0。輸出的 `player` 根據 `round` 來決定、控制球瓶馬達的訊號 `pin_motor_start` 由 `state` 決定、玩家分數由 `score_control` module 根據 `pin_state` 和 `allow_record` 計算、`all_pins_down` 是專門拉出來給 `sound_control` 的訊號。

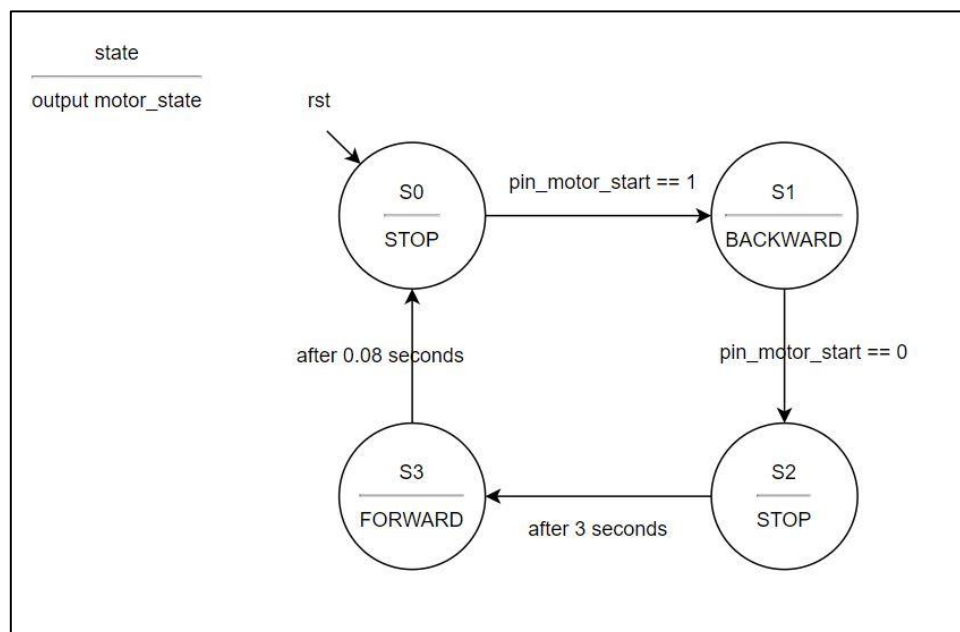


上圖為 `player_control` 裡 `state` 的 FSM state diagram。當 reset 後會來到 `S0` 並輸出 `pin_motor_start` 為 0 通知馬達不用啟動，若 `swap` 按鈕被按下且回合數  $\leq 5$ ，則進到 `S1` 並沒有額外動作。在 `S1` 會輸出 `pin_motor_start` 為 1 去通知馬達要運轉拉線收球瓶，過了 1.6 秒 (馬達拉球瓶持續 1.6 秒) 後則回到 `S0` 並將 `round` 增加 1。



上圖為 `player_control` 裡 `allow_record` 的 FSM state diagram。當 reset 後會是 1，代表目前可以記錄分數，如果 `swap` 按鈕按下且 `state` 在 `S0` (i.e., 要切換玩家)則 `allow_record` 會進到 0，代表目前要維持住原本的分數。在 0 時經過 5.5 秒後會回到 1，這是因為收線機在馬達拉了 1.6 秒後還要等 3 秒多讓球瓶穩定並鬆開綁線，在這期間紅外線會很不穩定因此不能紀錄分數，而且此時紀錄也沒意義。

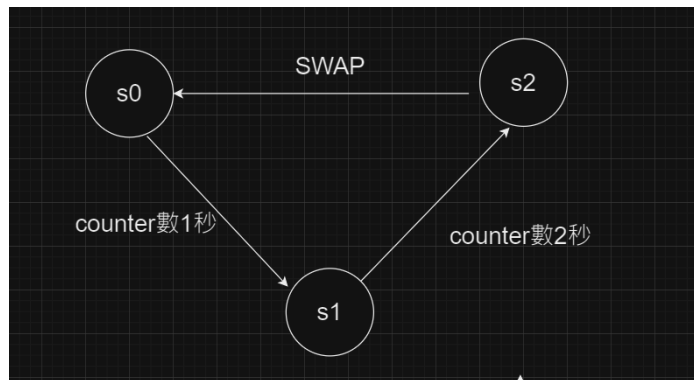
最後是 `pin_control` module，與 FPGA2 的 `conveyor_control` 類似，基本上為一個 Moore Machine，每個 state 對應的 output 為控制馬達的 `motor_state`，其 state diagram 如下圖。



Reset 後會回到 `S0` 並將馬達設為 `STOP`、不要運轉，若 `player_control` 傳來的訊號 `pin_motor_start` 叫我們要開始拉了，則會進到 `S1`。在 `S1` 會將馬達設為 `BACKWARD`，代表持續把線拉緊，直到 `player_control` 認為不用再拉了 (`pin_motor_start==0`)，則會進到 `S2`。在 `S2` 會將馬達設為 `STOP`，並等 3 秒後進到 `S3`，這段時間主要是球瓶被拉起後晃動到穩定所需時間。在 `S3` 則會將馬達設為 `FORWARD` 代表把線放鬆，以便球瓶下一回合不會因為拉太緊而無法倒下。在 `S3` 經過 0.08 秒後便會回到 `S0`，此段鬆線時間很短是因為馬達動力太大，且我們只需要把線放鬆一點點即可。

**(c) FPGA1: 音效**

FSM



聲音共有 3 個 state s0、s1、s2

s0:

```

se (state)
s0:begin
    if(counter>0&&counter<20_000_000)begin
        freqL=32'd784 ;
        freqR=32'd784 ;
    end else begin
        freqL=silence;
        freqR=silence;
    end
end
end
  
```

球被擊倒後 counter 才會數 1 秒進到 s1，0-0.2 秒會有聲音，從 s2 回到 s0 時使用 counter1 數 5.5 秒這期間因為球瓶恢復會晃動不會偵測有沒有球瓶倒下。

s1:

會判斷有沒有全倒有的話會撥音效，counter 數兩秒後進到 s2。

s2:等待 swap 回到 s1。

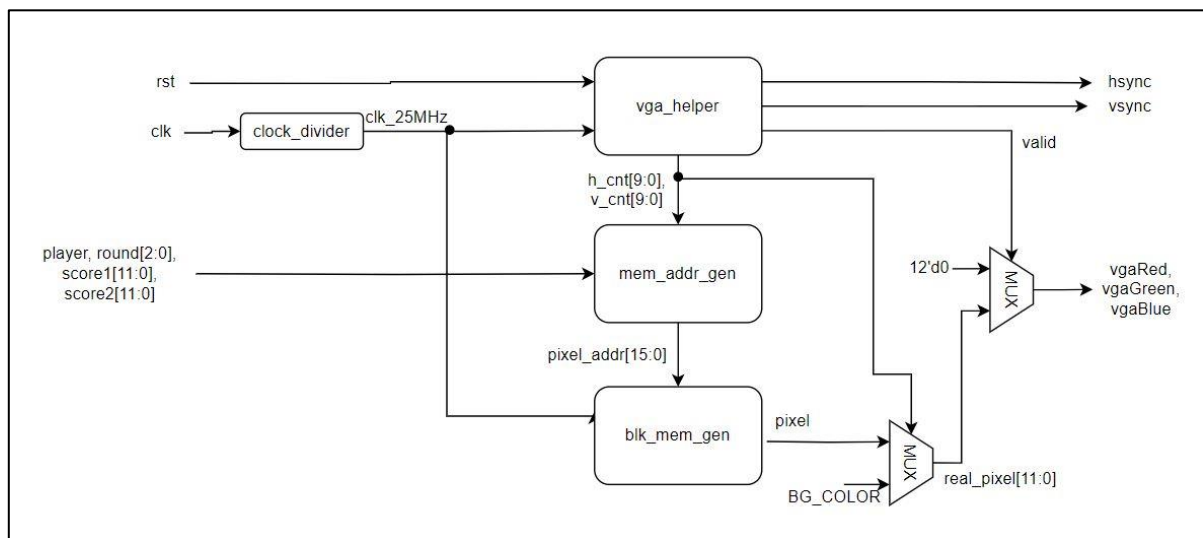
**(d) FPGA1: VGA 顯示**

下圖為螢幕顯示例子，圖中第一排的 1、2、3 代表回合數，各個回合底下的數字分別對應到每個玩家各個回合的分數，總計分數在最右邊的直列，目前正在丟球的玩家會顯示為藍色的方塊，尚未輪到的回合分數則以全灰方塊代表。以這張圖為例，目前是玩家 1 的第二回合。只要球瓶被擊倒或者玩家按下切換玩家按鈕，螢幕上的狀態便會立即更新。



	1	2	3	TOTAL
P1	2	0		2
P2	1			1

下圖為 `vga_control` 的 block diagram，基本上與 lab6 很類似，不同的是 `mem_addr_gen` 是根據玩家、回合數、玩家 1 分數、玩家 2 分數去計算目前螢幕的 pixel 要印的圖片的 pixel 位址。此外 `blk_mem_gen` 產生的 pixel 還要先經過 MUX 根據目前(h, v)去判斷目前位置是否要用背景顏色。



### C. 實作完成度&測試完整度&難易度說明

**輸送帶:** 完成度 95%。我們原本期望超音波偵測到球後，會自動啟動輸送帶把球送到球架。而實作出來大部分時間都可以完美做到，不過我們測試時大約 5%的機率球會被卡住，這是因為輸送帶上有架設檔板，當球從機台後方軌道滑落到輸送帶上時有可能輸送帶剛好轉到檔板卡住球落到輸送帶上。輸送帶也是整個 project 最困難的部分，在實作過程遇到許多硬體的問題，詳細於 F.討論。

**球瓶自動歸位:** 完成度 95%。我們測試時幾乎符合當初設計，即當按下切換玩家的按鈕後，倒下的球瓶會自動站起。沒完成的 5%主要為硬體的層面，像是由於要收球瓶需要綁線於瓶底因此瓶子有一定的倒下範圍、倒下的瓶子可能卡住球、球瓶拉起後需要等約 5 秒來讓球瓶穩定不再晃動等。

**計分:** 完成度 99%。我們期望每一回合只要球瓶一倒下就立即更新分數+1 分到該玩家該回合分

數，實際做出也符合此期望，未完成的 1% 是理論上有可能發生球瓶被撞移開 IR sensor 上方但仍舊沒被撞倒，規則上這樣不應算擊倒，但由於 IR 偵測到上方沒瓶子就會當作是擊倒，因此還是會算到分數。不過我們測試時完全沒出現此狀況過就是了，因為我們軌道設計很長使得球速較快不太會有被撞到但沒倒的情形。此項目較困難的部分主要是 code，由於球瓶倒下後可能蓋到其他球瓶的 IR sensor、球瓶拉起時可能會晃動影響 IR sensor 的訊號，因此 code 要想辦法避免這些情況去影響分數，而我們後來也確實做到了。

**擊倒保齡球音效:** 完成度 100%。問題自動歸位一樣，需要等球瓶站穩再去檢測有沒有球瓶有沒有被擊倒，需要等待 5 秒的時間。

**保齡球投出音效:** 未完成。我們原本設計放一個 sonic sensor 在球道上去檢測球滾出，然而如同先前我們與老師討論時老師所提到的速度太快會有檢測不到球的可能，我們實際測試後也發現確實 sensor 無論放在球道哪處都很難穩定檢測到滾動的球的距離，因此最後沒有實作出此項功能。

**VGA 顯示:** 完成度 80%。原先期望將雙方分數、目前回合數、目前玩家顯示於螢幕上，就功能來說都確實做到了。未完成的 20% 主要是美工的部分，原先期待顯示可以更好看一點，比如說加底圖、漸層、全倒動畫等，類似下圖。最後由於我們剩餘時間不夠去畫這些圖、思考 code 的實作方法，因此做出來的版本較為陽春。



## D. 分工

項目	負責人
硬體機台實作	蕭以勝
Code: 音效模組	蕭以勝
Code: 輸送帶、收線機、計分、VGA 顯示	楊立慈
Debug、整體機台設計、機台問題討論	一起

## E. 課程外部分比重

我們 project 使用了 4 個 TT 減速馬達、3 個 IR sensor、1 個 sonic sensor、1 個 VGA 顯示、1 個音

效模組，其餘的機台、收線器皆為自己手做，因此有使用的模組皆為課內的，課外比重為 0%。在 code 上，我們的 motor、sonic sensor、VGA 顯示、音效模組有使用過去 lab 提供的 template 來做修改，包含: clock\_divider、debounce、one\_pulse、SevenSegment、vga\_controller、note\_gen、sonic、motor\_pwm，使用比例約占全部 code 的 15%。

## F. 遇到的困難與解決方法

### (a) 輸送帶

#### 1. 履帶跑掉和摩擦力:

一開始我們先將輸送帶只做 1/2 的大小就會出現履帶跑出的問題，因此加了上保護裝置，而上保護裝置也會有磨擦力太大的問題，我們嘗試讓上保護裝置更平滑，就可以成功輸送。輸送帶加長後，一開始可以正常運行，但架到機台後，經過多次的測試後履帶會有磨損，導致下方的履帶也會跑出輸送帶，因此我們嘗試製作和上保護裝置一樣的保護裝置，但發現馬達轉軸不是平行的保護裝置的外緣會卡到履帶因此我們將下保護裝置的外緣往外搬以避免卡到的狀況，我們也在上下保護裝置都有噴潤滑劑減少摩擦力。

#### 2. 扭力不足:

一開始我們使用電池盒提供電力，但發現電池盒供電不穩定，所以我們改使用變壓器提供電源，輸送帶可以正常運行後我們將球放上去時會發現會有扭力不足的問題，我們因此多增加兩個馬達，就可以解決此問題。

#### 3. 帶子上的檔板卡到球落下

由於輸送帶要 sonic sensor 偵測到球在輸送帶上才會運轉，然而有可能出現檔板剛好運轉到卡在球落下軌道的位置，導致 sonic sensor 偵測不到。由於我們測試時大約 20 次才會出現 1 次此情形，且沒想到比較好的解法，因此最後決定加裝一顆操縱輸送帶的 keep\_running 按鈕，當球被卡住時就按鈕把檔板往前送。(demo 時不知為何發生了 2 次此情形，我們推測可能是我們 demo 前一小時大量測試導致輸送帶有磨損或馬達有磨損，因此原本 code 設定的參數不再合適)

### (b) 捲線機

1. 一開始我們三條線並沒黏在一起，但經過測試後收線時旁邊會有線卡到收線機外圍問題，我們後來黏在一起就解決了這些問題。

2. 馬達原本測試很多次都沒問題但在 demo 前兩三個小時前發現馬達內部扭力有點不夠，我們將拉緊的時間拉久一點，球瓶雖然無法像之前一樣快速站立，但至少 5 秒內會穩定站立，因此計分和音效也要一起調整。

3. 在馬達收線後，球瓶仍會晃動一段時間才穩定。我們經過多次測試後得出大約最多花 3 秒才會回到穩定，因此在 pin\_control 的 FSM 多加了一個 S2 去等 3 秒。

## G. 心得討論

我們發現實作機台硬體很多問題。首先架構要規劃的很嚴謹，並經過多次測試確認沒問題後再架設，否則若有一點誤差，很可能導致越偏越多，像是我們的輸送帶就是因為帶子不是直的導致會隨著運轉造成帶子偏右；還有我們的收線機的線也是因為長短不一造成線的鬆緊程度不同，導致有的瓶子比較難倒。

另外，我們發現機台不能用紙板，由於紙板較薄弱容易造成在測試、實際跑的過程中紙板便磨損掉，導致每次測試機台的穩定度不一，時常需要改 `code` 裡面的參數。除此之外，經過多次測試的過程中也可能造成硬體的磨損，像是收線機的馬達在我們測試多次後會發生卡卡的情況，導致動力不足每次放線的長短不一。此外，我們一開始想要先做小型的機組先試試看能不能正確運轉，再修改成大型的機組，然而隨著尺寸變大也出現更多問題，像是馬達動力不足、機台穩定度。

最重要的是我們學到實作硬體不像軟體一樣是理論上完美的、你 `code` 寫什麼它跑出來就會是什麼，實作硬體會需要考量很多 **real world** 的問題。舉例來說輸送帶摩擦力的問題需要試很多方法解決，像是加大馬達動力、多串聯更多馬達、加潤滑油、裝平滑的檔板在側面、剪掉輸送帶右側的牆壁等等。實際的硬體多了很多變數在裡面，造成往往實際跑出來的結果與 `code` 寫的預期結果有偏差。不過最有趣的部分卻也在此，有時候一些理論上的解法不一定能 **work**，反而是一些奇特的嘗試也許就能成功。最有成就感的莫過於在多次 **trial & error** 後，找到一個理論與實際的平衡點。