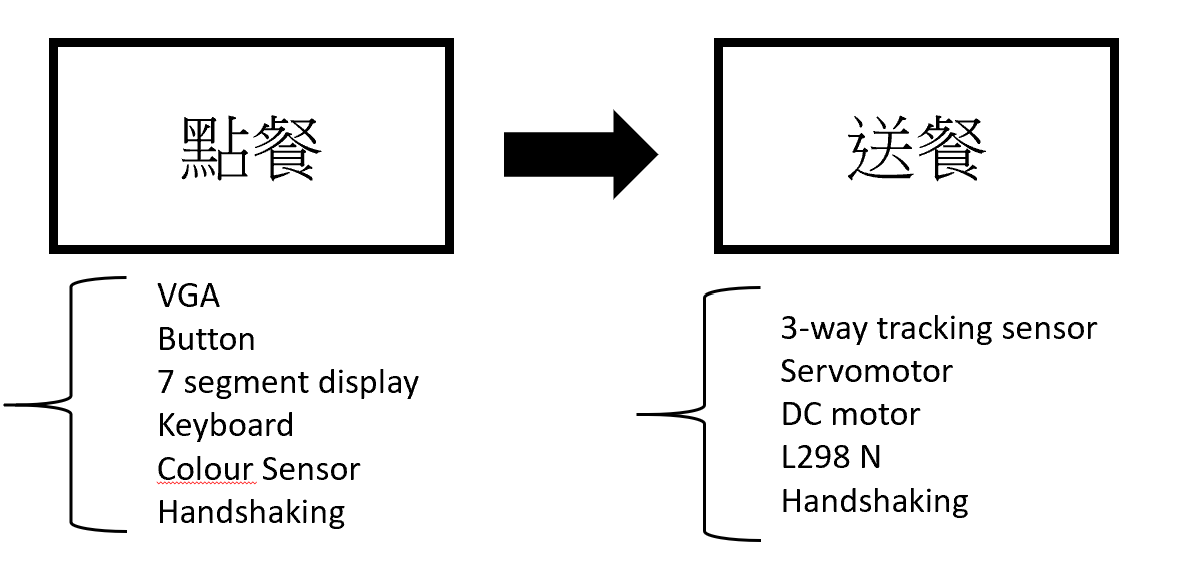
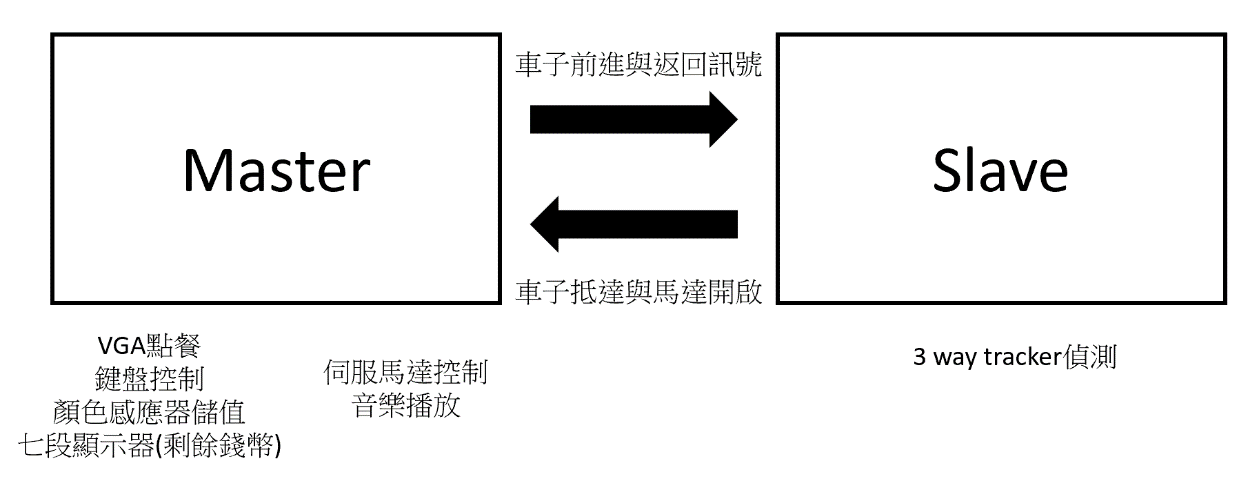
|  |  |
| --- | --- |
| **group 37 : Final project**  **是食物者為俊傑** | |
| 112062139 劉韋呈  112062117 王政傑 |  |

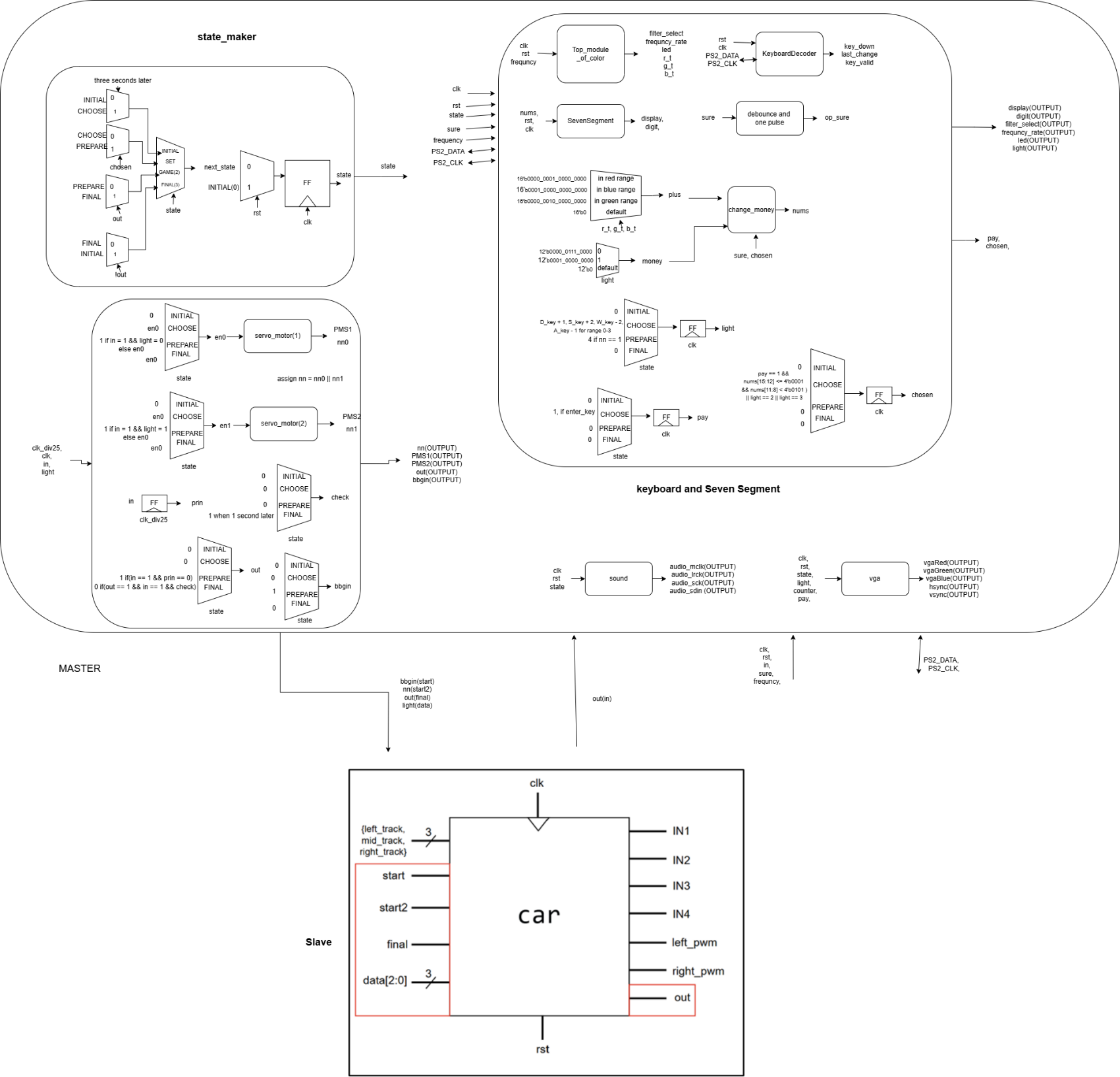
1. **硬體設計概念**

本次project模擬自動點餐機，先透過master端呈現音樂與點餐畫面，透過鍵盤選取餐點。此外，利用TCS3200顏色感應器可以獲取當前物體的RGB，當按下右邊的Button時，會根據相應假鈔的顏色，增加對應的金額，並顯示於七段顯示器。

**** 接著，進入準備狀態，master端handshaking位子給slave端，並抬起接受訊號，車子利用3-way tracking sensor保持直線運動，並且當接受全黑的訊號時，表示跨過一個站點。當抵達目的地時，slave端抬起抵達訊號，此時master端操控伺服馬達轉動，物體掉落於車上的漏斗後，master端抬起回去的訊號，車子到回原點後抬起結束訊號，slave短暫進入final狀態後，回到initial狀態。

****專題器材與技巧關聯圖

主從關係示意圖

1. **架構細節與方塊圖**
2. **Block diagram**
3. **VGA與七段顯示器**

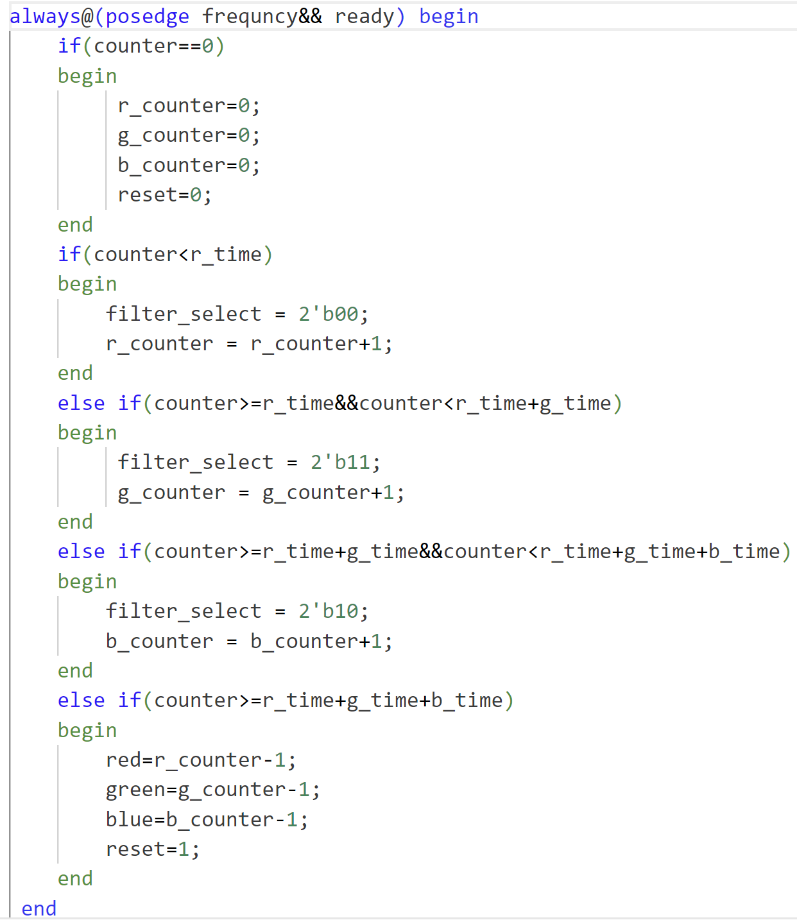
七段顯示器會顯示目前玩家擁有的金額，在CHOOSE state時，你可以選擇你要點的餐點，螢幕上會有四張圖，用light來記錄目前你所選的餐點，被選的圖上會有淡淡的藍色，表示他正在被選，當確定要選的東西時，按下enter後會跑到付費介面，你可以按N表示不要付費，按enter會進入PREPARE state並付費，七段顯示器的值會改變。

1. **顏色感應器儲值**

先進行白平衡，找出各種顏色的光強度上限，並控制在256以下。ready變數代表白平衡已經完成。

白平衡程式碼

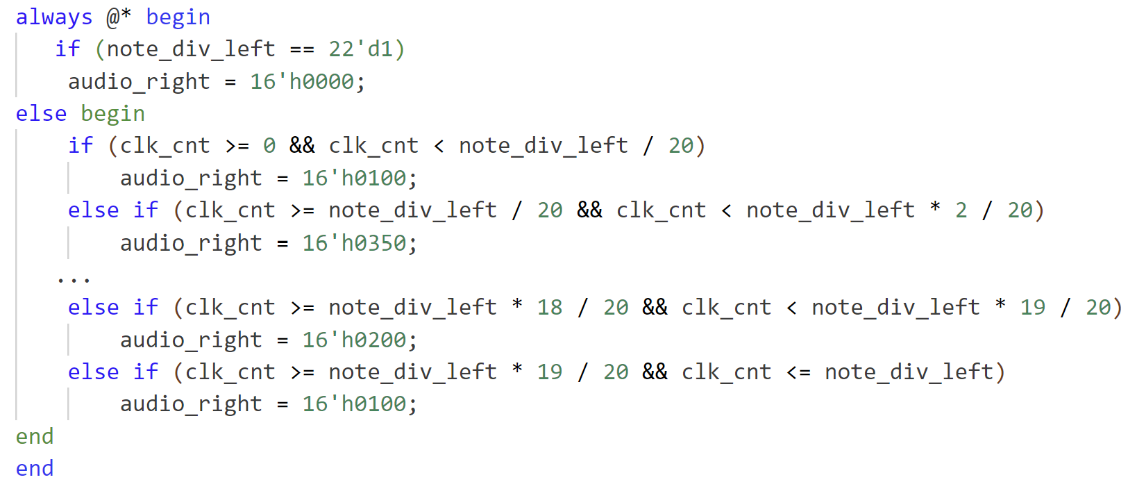
frequncy變數來源於顏色感應器，透過接受不一樣的光強度，會產出對應的頻率。而filter\_select代表遮色片的模式，當為2’b00時，此時只有紅光通過，2’b10時，則只有藍光通過，最後當2’b11時，為綠光通過。借助這種方法，可以有效得知RGB數值。

顏色判斷程式碼

1. **伺服馬達控制**

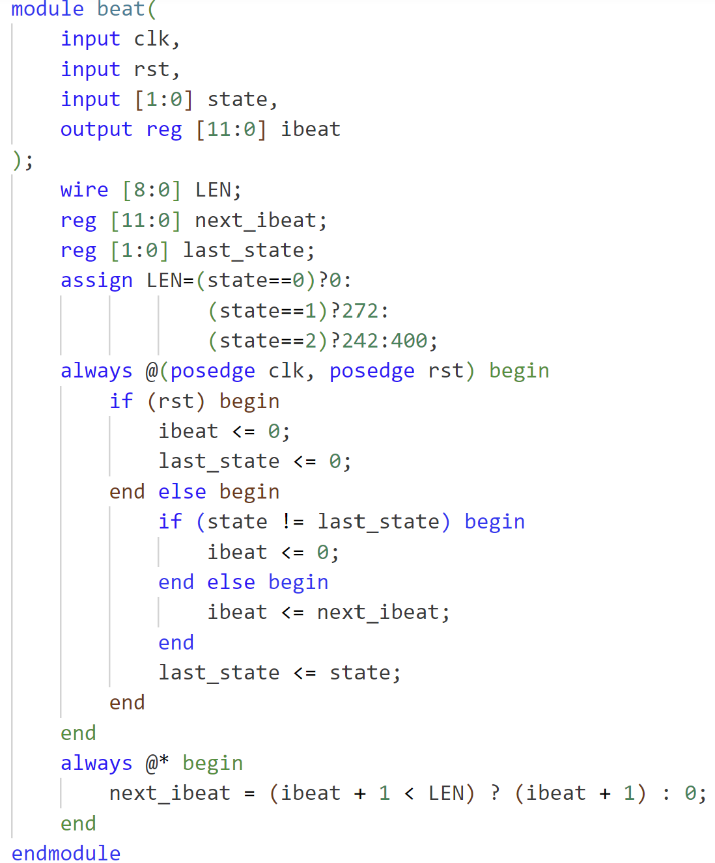
當進入PREPARE state後，bbgin(slave的start) = 1代表車子可以開始動了，如果slave傳入的in(slave的out) = 1，根據light的值改變哪個en，其對應的馬達轉360度，我用的方法是讓伺服馬達一直轉，大概估算一下轉完一圈的時間，時間到了就停下，轉完後輸出nn(slave的start2)給slave，代表車子可以倒回原點。等到倒回原點後，slave會傳入in(slave的out)=1，如果prin(0.25秒前的in)也為0，out變為1，輸出給slave的同時，也利用out == 1，使進入FINAL state。

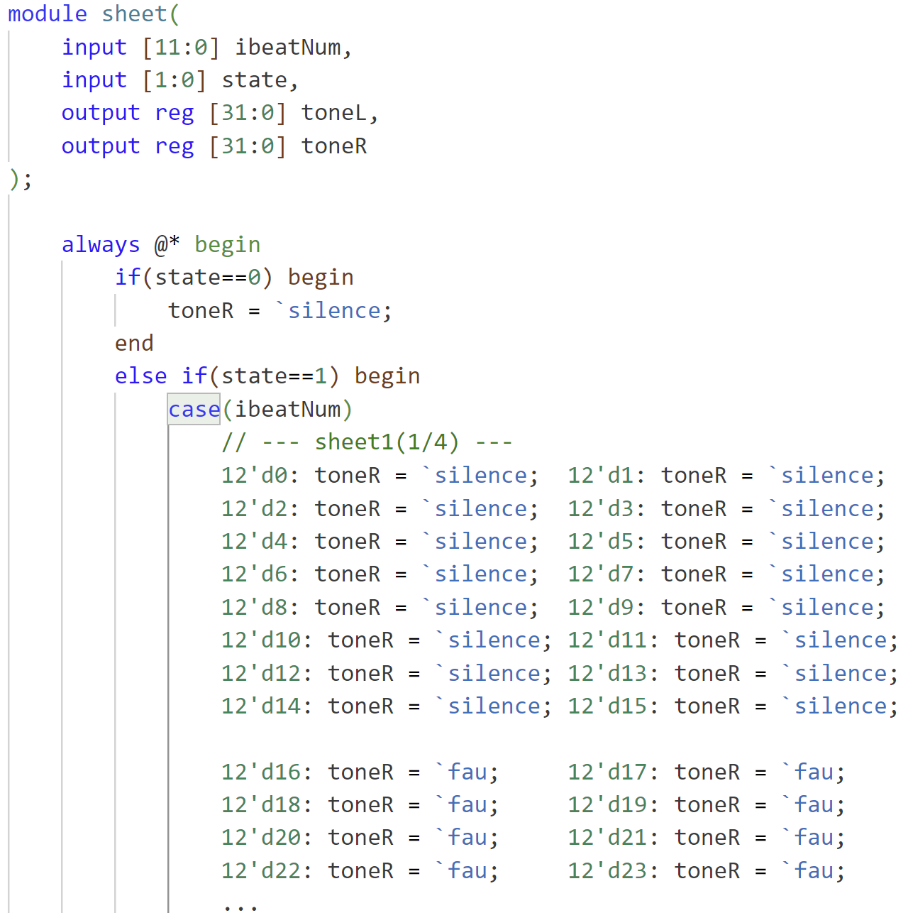
1. **音樂控制**

在note\_gen module中，原本波型為方波圖，透過描點的方式，將波型轉成類似鋼琴的波型，呈現的效果更加平滑順耳。

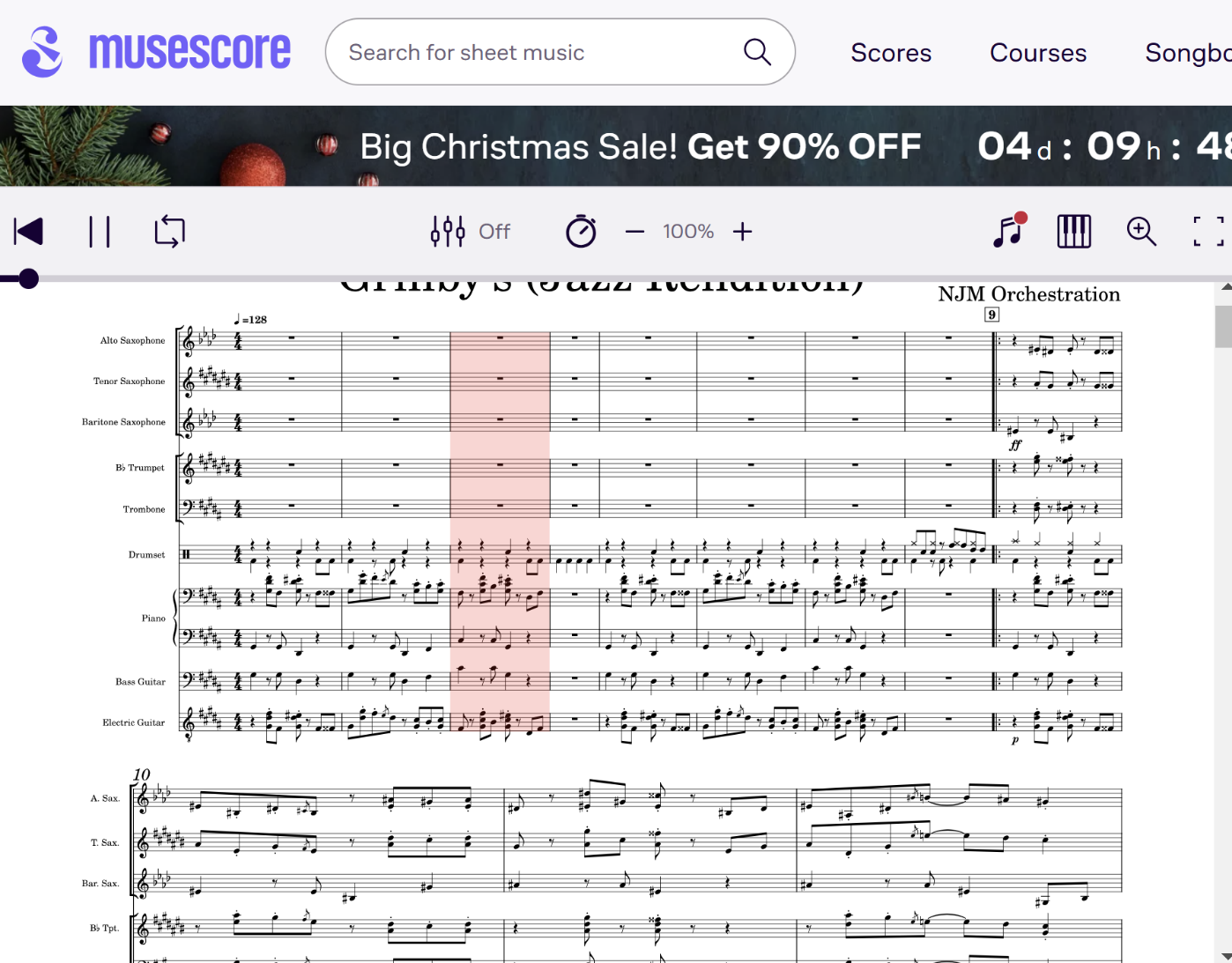
震幅控制程式碼

利用beat module，根據不同狀態來控制歌曲的時間長度，當播完之後，再次循環播放。

beat module程式碼

**** 利用sheet module，將網路上的五線譜轉成音調，然後以四分音符為佔十六個為基礎，去分配節奏於程式碼中，並同音調連音時要隔一個silence。此次專題總共三首曲子，每首曲大致有八個小節，右聲道處理高音部，而左聲道則處理低音部。

sheet module程式碼

**** 五線譜皆來自musescore這個網站，選擇較為簡化的譜來實做。

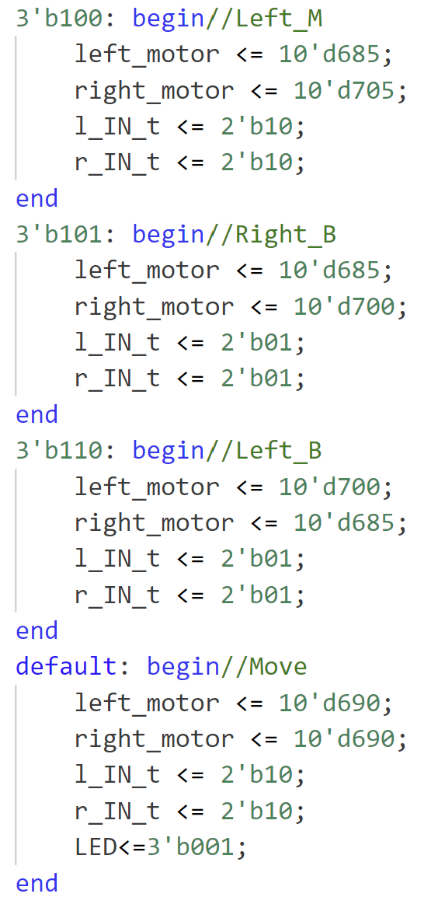
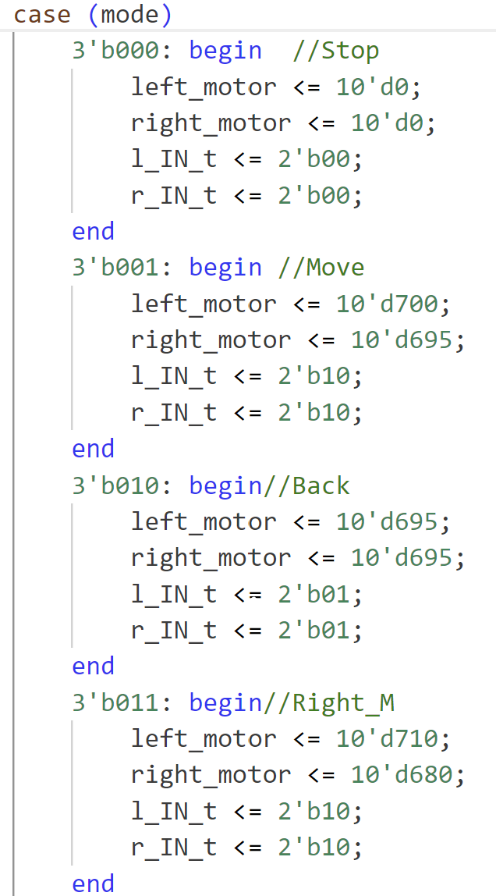
1. **自走車控制**

自走車在master端傳遞位子並接收到start或start2的訊號後，透過3-way tracking sensor來直進，並且當收到全黑訊號時，會記錄過一個站點。抵達目的地後，輸出out值用於後續的馬達控制，或是進入final狀態。

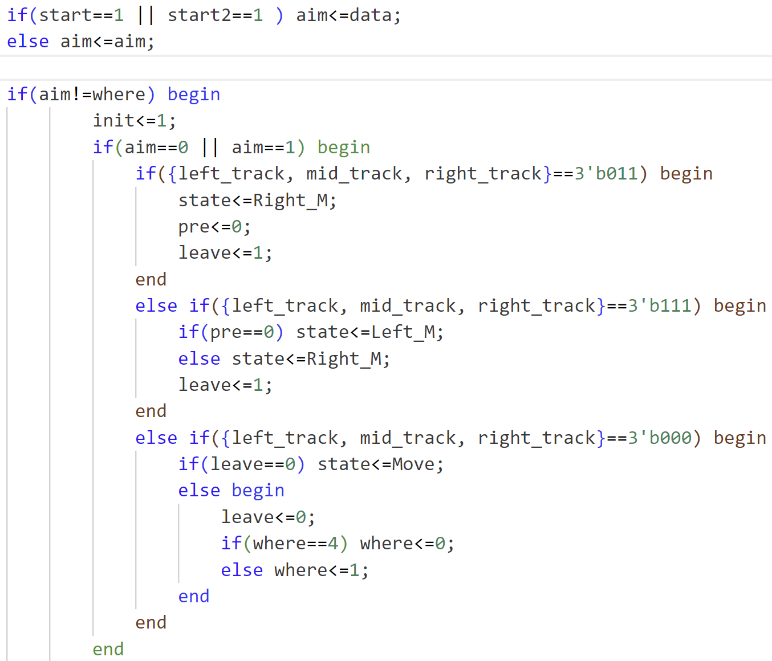
在motor module中，根據不同情況設定不同的轉速差異，以維持自走車的直線前進。因為初始情況下，黑色直線軌道會位在車子的左側，故通過讓車子往右偏的方式，並在檢測到全白的情況後，偏回左邊，以此達到車頭保持向前的狀態。

可以注意到，Move情況的輪胎轉速偏向右側，也是為了校正車頭而設計。但是因為倒退走的情況很難掌握，所以依舊維持兩輪轉速一致的情況。此外，IN值為2’b10是往前轉，而2’b01則是往後轉。

轉速控制程式碼

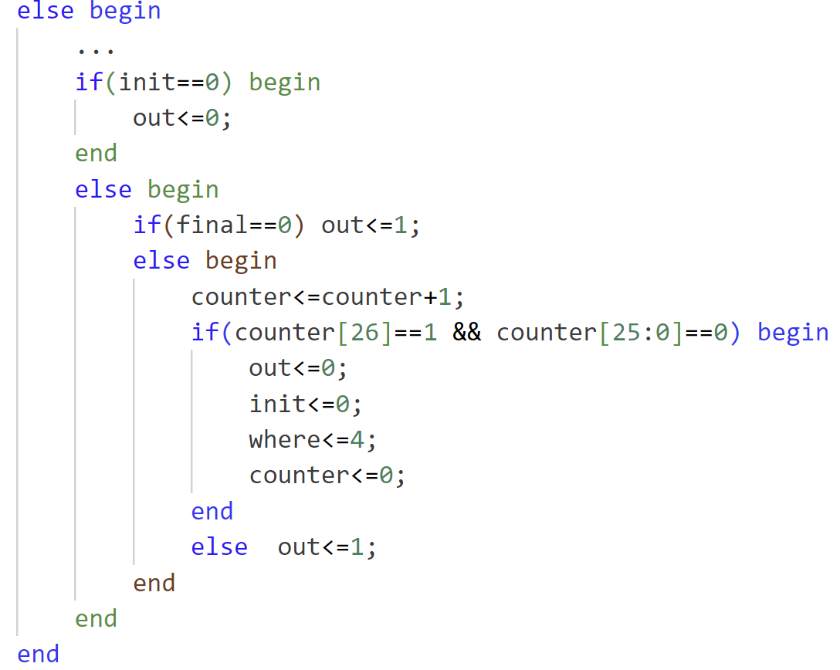


在tracker\_sensor module中，start代表接受前進訊號，而start2代表接收返回訊號。當偵測到位置訊號改變後，根據地板黑線判斷方向，當偵測到三黑代表過一站。而init變數是避免將起始三黑訊號算入一站，而leave變數是避免重複將一站誤認成多站。



自走車前進程式碼(1)

當當前位置與目標位置一致時，傳送抵達訊號。並且，當master端傳送final訊號時，代表已經抵達終點，經過一秒後，初始化所用的變數。

****自走車前進程式碼(2)

1. **實作完成度**

本次專題大致有三個部分未實作。

第一，我們將藍芽控制改成直接用handshaking，因為藍芽在操作上有一定難度。

第二，將食物掉落改成物品掉落，因為可能會造成不必要的浪費。

第三，我們將超聲波感應器刪除，因為容器固定於車子上，所以不用實作拿走容器的判斷。

總結，我們認為有80%的完成度。

1. **難易度說明**

困難的地方在於我們動用了所有課堂學到的技巧，所以相互配合起來有點吃力，加上大量實體物件需要手工製作。

VGA控制需要思考容量上的問題，而音樂則需要思考波形方面的設計，以及理解並將音符放置於程式碼中，需要很多時間。自走車方面，可能光線等因素，3-way tracking sensor有時候會失靈，需要不斷設計新的地圖。

最後，伺服馬達與顏色感應器皆屬於課堂外的技巧，所以蒐集了很多資料去理解。

1. **困難與解決**
2. **VGA與七段顯示器**

如果載太多圖Bram會有過載的問題，因此我有的畫面要用刻的，在不同h\_cnt和v\_cnt輸入對應的顏色，使得螢幕顯示出你想要的圖案或字。

1. **顏色感應器儲值**

原本使用tcs34725顏色感應器，但發現I2C通訊難以實作於FPGA板上，後面就改成TCS3200。此外，一開始沒有白平衡，導致數值極其不穩定。

1. **伺服馬達控制**

因為FPGA的接孔的電流會變小，因此如果伺服馬達的GND和VCC接FPGA的孔時，馬達轉不起來，因此我們使用了Pmod con3，這使我們可以用變壓器接Pmod con3的GND和VCC來供電。

1. **音樂控制**

方波音聽起來很刺耳，將波形後改善很多。在找譜方面，需要找到簡化但是不偏離原曲的譜，並不容易，甚至有去找除了鋼琴以外的譜。

1. **自走車控制**

由於自走車整體的構造，所以倒退走花了非常多時間，解決方式是改短倒退的移動路徑，這樣直線前進時，可以再度修正回來。

1. **心得討論**

這個專題可謂是整學期課程的總結，透過不同技巧的搭配，有時會出現意想不到的效果。雖然中間製作的過程很辛苦，但看到demo能順利進行，成就感滿滿。

若想延伸此專題，或許能將紙板改成木板比較牢固，然後如果能優化密封性，或許能改成液體的流動控制等。

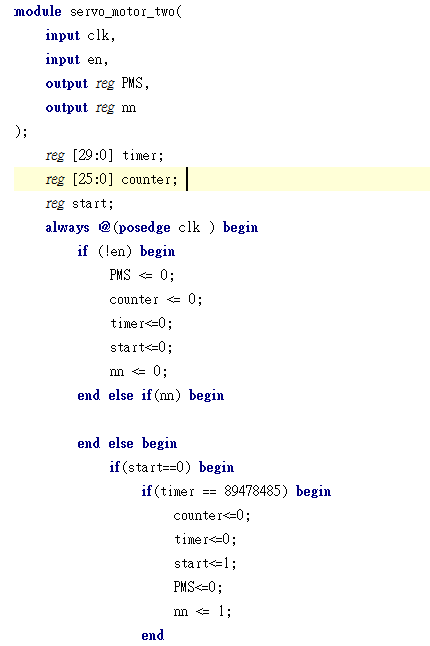
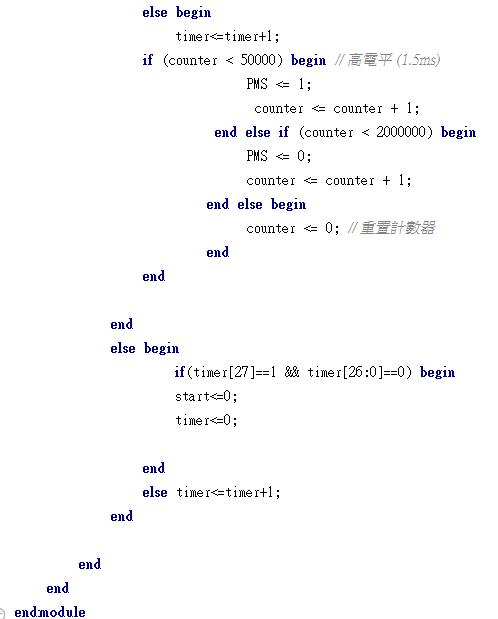
1. **詳細分工**
   * 1. **王政傑**

* **VGA控制(master端)**
* **伺服馬達控制**
* **七段顯示器**
  + 1. **劉韋呈**
* **自走車控制(slave端)**
* **顏色感應器控制**
* **聲音控制**

1. **課程外部分**
2. **伺服馬達**

我把伺服馬達的脈衝寬度固定，讓馬達一直轉，我們用計數器來計算甚麼時候馬達要停，這樣能使伺服馬達轉到我們想要的角度。

以下是轉360度後停止的code:



伺服馬達控制程式碼

1. **顏色感應器**

在顏色感應器的控制上，遇到很多困難，最後在一個github上找到相關的例子，所以有RGB部分程式碼參考於這個網站，屬於此專題未完整實作的部分。

Github: <https://github.com/zhangyiwen599/VGA_game>