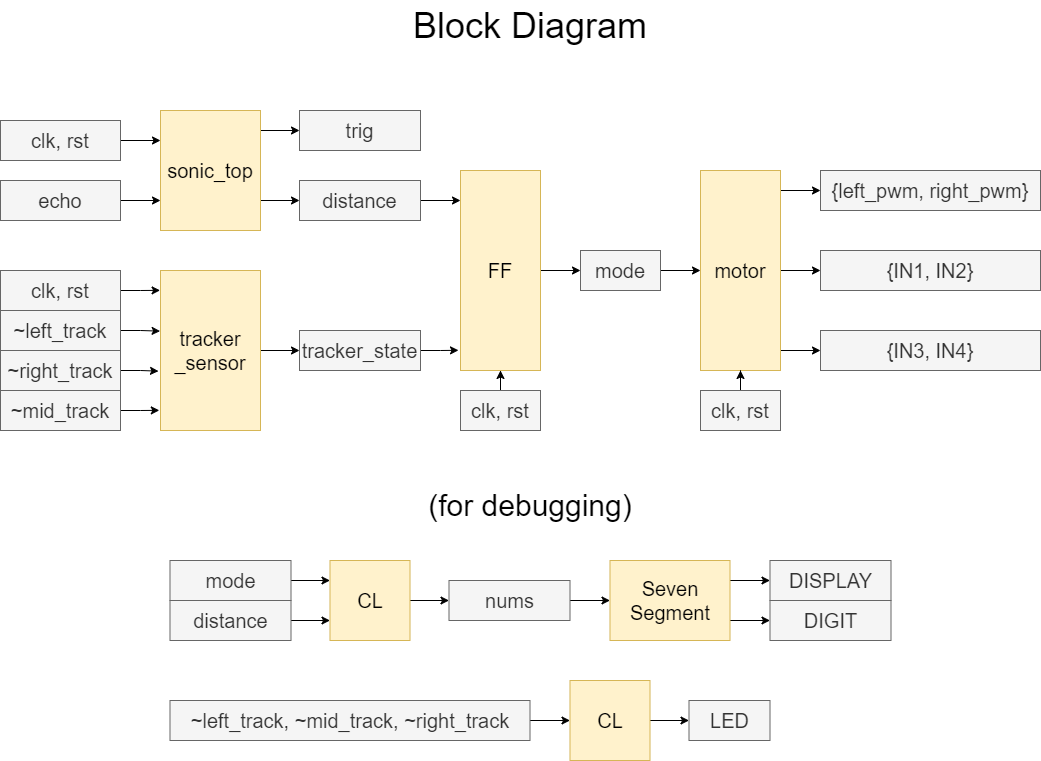
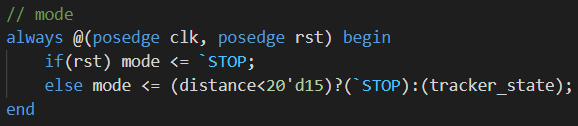
|  |  |
| --- | --- |
| **Lab 9** | |
| 學號: 108032053、109062174 | 姓名: 陳凱揚、謝承恩 |

1. 實作過程

下圖1為module lab9的block diagram，首先將module sonic\_top取得的distance和module tracker\_sensor取得的tracker\_state做運算輸出mode，如下圖2，在rst或是distance小於15公分時，mode設為停止，其他時候則直接設為tracker\_state。接著便將mode接至module motor中，控制馬達的轉速及轉向。此外，我們也另外加上SevenSegment和LED來幫助debug，其中第1位數字用來顯示mode狀態，有0~3等4種，第3~4位數字用來顯示distance，大於100公分時則不顯示；並用LED[2:0]分別顯示~right\_track、~mid\_track、~left\_track的偵測狀態，亮的時候代表黑色。

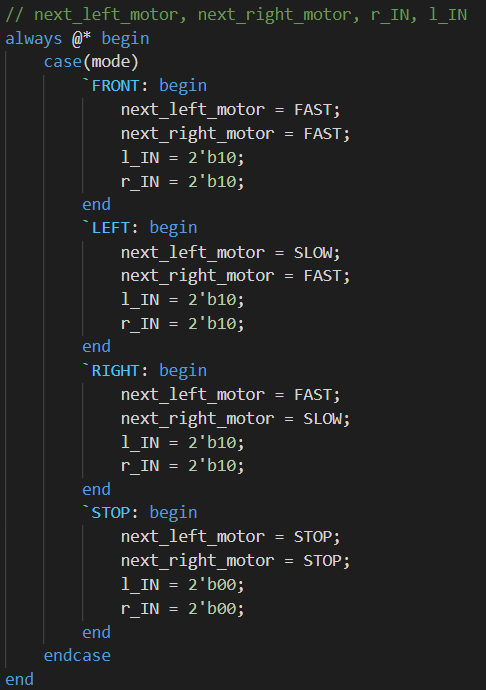


▲ 圖1



▲ 圖2

如下圖3，在module motor中，mode會有4種狀態，在FRONT時，左右輪的轉速都設為FAST(800)；在LEFT時，左輪設為SLOW(600)，右輪設為FAST；在RIGHT時則相反；在STOP時設為STOP(0)。而l\_IN和r\_IN除了停止時都設為向前轉，因為並不會有倒車的情況發生。



▲ 圖3

在module sonic\_top下，會有個module PosCounter負責輸出距離（單位為公分），我們所做的只是將已經數好的cycle數，根據cycle的週期和音速轉換成單位為公分的距離，如下圖4所示。

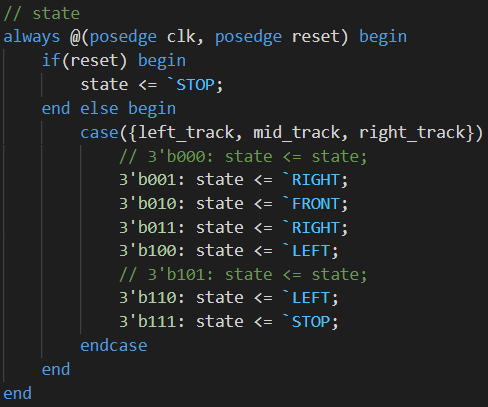


▲ 圖4

在module tracker\_sensor中，我們會根據3個track所偵測的結果組合來決定state

（下圖5）。

1. 當只有右邊或只有中間和右邊偵測到黑色時，設為向右。
2. 當只有左邊或只有中間和左邊偵測到黑色時，設為向左。
3. 當只有中間偵測到黑色時（通常在直線時偶爾發生），設為向前。
4. 當全部都是黑色時（通常發生在車子拿在空中時，在跑道時不太可能發生，因為跑道黑線較細），設為停止。
5. 當只有中間是白色時，延續前一次的狀態，因為此狀態不應該發生在跑道上，可能只有極少數時候意外發生，因此忽略此偵測結果。
6. 當全部都是白色時，延續前一次狀態，因為在實際測試時發現在大多數情況下，偵測結果皆為此種，可能是因為跑道線較細，因此讓車子延續前一次的狀態。



▲ 圖5

1. 學到的東西與遇到的困難

**學到的東西：**

(1) 紅外線感測器的使用：如果感應到黑色，回傳0，感應到白色，回傳1，不過

template code把它們顛倒過來了。

(2) 馬達的控制：  
 方向：

(IN1, IN2) = (0, 0) 或 (IN1, IN2) = (1, 1)：馬達停止  
 (IN1, IN2) = (1, 0) 馬達正轉

(IN1, IN2) = (0, 1) 馬達反轉

\*正反轉的部分，要實際用車子測試再修改code比較準確

速度：

原理是藉由控制PWM訊號(in module PWM\_gen)拉成1的時間佔整個cycle的比例，

來調整馬達的速度，簡單來說，我們只需要設定不同的duty值給PWM\_gen即可，

duty 的值越大(0~1023)，則電壓越高，馬達轉速越快

* PWM\_gen module

一張含有 文字 的圖片

自動產生的描述

(3) 超音波模組的使用：

* PosCounter module

初始狀態狀態為S0: 此時count的值維持0。當start訊號為1時，進入到S1: 此時開始計算經過的cycle數。當finish訊號為1時，進入到S2: 此時將經過的cycle數存到distance\_register裡，並將count歸0，回到S0。

一張含有 文字 的圖片

自動產生的描述

start, finish 訊號的產生：

echo\_reg1 為 1, echo\_reg2 為 0 => echo訊號的 posedge, 此時開始計時

echo\_reg1 為 0, echo\_reg2 為 1 => echo訊號的 negedge, 此時結束計時

一張含有 文字, 橙色 的圖片

自動產生的描述

* PosCounter module

每經過 100ms將 trig 設為1 10us

一張含有 文字 的圖片

自動產生的描述

(4) 寫一個FSM控制車子行進的方向

**遇到的困難：**

1. 車子的接線：組裝車子的時候超音波模組和感測器有接錯線，導致fpga用外部電源供電的時候吃不到電，一開始應該就要照著xdc檔接線，我貪圖方便照著pdf檔的照片去接線(用顏色去對)，最後反而浪費更多時間。
2. 沒注意到track的值被反轉：因為template code有將left\_track, mid\_track, right\_track的值進行反轉，我們一開始沒注意到，花了不少時間debug
3. 馬達轉向的控制：一開始我們的車子會在原地打轉(一顆輪子往前轉一顆輪子往後轉)，但我們的程式並沒有這樣的操作，後來就將往後轉的輪子改成往前轉(用程式控制)，發現車子變成倒著走(兩顆輪子都往後走)，我們才發現左右輪跟我們想的剛好相反。
4. 馬達速度的控制：我們一開始將duty設為500發現輪子不會動，花了蠻多時間檢查程式結果最後發現只是duty設太小。
5. 車子方向控制：我們原本想針對 {left\_track, mid\_track, right\_track} 所有的可能值(000~111)描述車子的行進方向，但是車子在跑的時候感測器的值很不穩定，當出現非預期的數值時(EX：101, 兩邊黑中間白)，我們沒辦法確定車子要往哪邊走，因此我們改成若出現未定義的結果時，就讓它維持先前的狀態，我們只定義 100, 110要左轉，001, 011要右轉, 010要直走，經過測試後發現使用這種寫法車子就能順利跑完跑道。
6. 轉彎幅度過大：車子在轉彎的時候迴轉半徑很大，雖然最後都轉的過去，但在轉彎的時候會有點偏離軌道，後來發現是我們在轉彎的時候輪子速度太慢，如果在轉彎的時候將車輪速度調快，轉彎幅度就會比較大，車子就可以貼著軌道跑。
7. debug困難：一開始我們沒有使用七段顯示器和LED，因此完全不知道是哪邊出了問題，後來將車子的狀態以及感測器的值輸分別輸出到七段顯示器和LED以後，debug速度就變快很多。
8. 超音波感測器偵測到不存在的東西：有發生前方明明沒有障礙物但超音波感測器的距離顯示小於15，於是車子停下來的情況，後來重新寫了一份code就沒有遇到這個問題，事後猜測當初可能是distance發生溢位所導致，印象中我有修改過distance 的 bit width。
9. 想對老師或助教說的話

**想問的問題：**

(1) 我們有自己去便利商店買9V的電池，結果裝上去發現沒辦法供電(馬達的燈有亮，

但fpga的power燈沒亮)，不太清楚為什麼會這樣。

(2) 想請問這次lab的分數會如何計算，因為lab9不在原本的課程大綱內。

**笑話：**



1. 分工

|  |  |
| --- | --- |
| 陳凱揚 | 1. 程式編寫 2. 車子測試 3. Report - 實作過程 |
| 謝承恩 | 1. 程式編寫 2. 車子測試 3. Report - 學到的東西與遇到的困難 4. Report - 想對老師或助教說的話 |