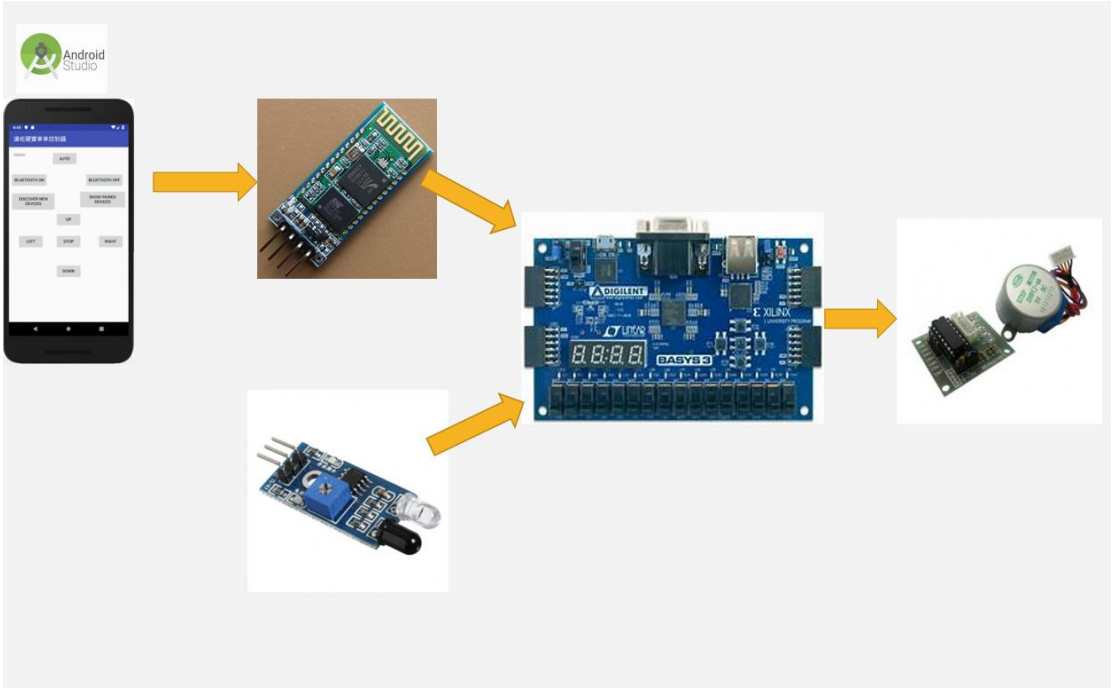
硬體實驗 final project

第七組
105030009 黎佑廷
105030015 郭家瑋

一、**標題**：FPGA 車

二、**目標**：使用 FPGA 與 verilog 做出一台可以由手機藍芽遙控，運用藍芽模組接收遙控訊號的車子，另外的模式是可以讓車子自動行徑而不靠遙控，車子運用紅外線避障模組來感測前方障礙物並且及時轉彎再前行。



其中的兩種模式詳述如下：

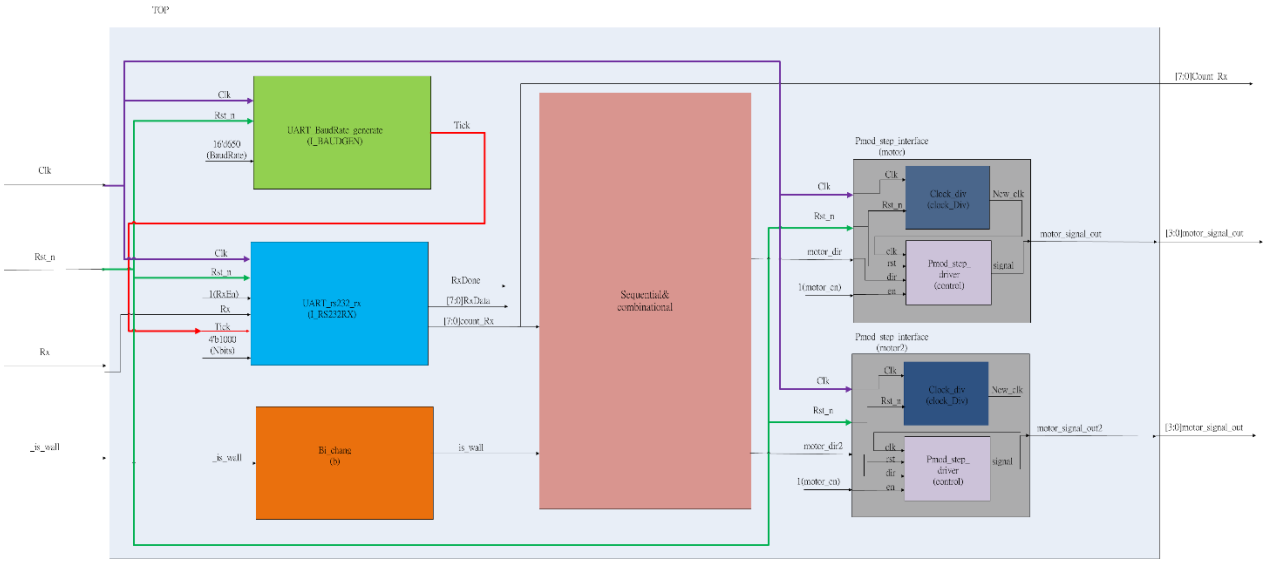
Manual: 於手機控制上、下、左、右

Auto: 車子自動前行，前方有障礙物時，後退、左轉、再前行

手機使用 android studio 寫 app，介面如下。

Android Studio

三、Block Diagram



以下將說明各個 module：

✓ **Bi_chang**

```
module bi_chang(is_wall, out_is_wall);
    input is_wall;
    output out_is_wall;

    //when close to the wall, is_wall=0
    //https://www.playrobot.com/infrared/1039-infrared-sensor.html

    assign out_is_wall = (!is_wall)? 1 : 0;
    //assign led = 1;
endmodule
```

這個 module 主要處理紅外線避障模組的訊號。我們的紅外線避障模組有三個 pin。兩個是電源 pin，另外一個是邏輯 pin，在前方三公分處有障礙物時，將 0 訊號傳回 FPGA 版，沒有障礙物時為 1。因為跟邏輯的 high、low 有點差距，因此在這個 module 將訊號反向。

✓ **Sequential & Combinational**

```
always@(posedge Clk) begin
    //if(!Rst_n) state <= state_direct;
    /*else */    state <= next_state;
    count_back <=  next_count_back;
    count_turn <=  next_count_turn;
end
```

在這個 module 主要是將來自紅外線、藍芽訊號收集並且給出最後的馬達訊號。

```
always@(*) begin
   if(count_Rx == 8'd6) begin
       case(state)
           state_direct: begin
                motor_en = 1;
                motor_dir = 0;
                motor_dir2 = 1;
                if(is_wall) next_state = state_back;
                else        next_state = state_direct;
           end
           state_back: begin
               motor_en = 1;
               motor_dir = 1;
               motor_dir2 = 0;
               next_count_back = (count_back == 28'd200_000_000)? 0 : count_back+1;
               next_state      = (count_back == 28'd200_000_000)? state_turn : state_back;
           end
           state_turn: begin
               motor_en = 1;
               motor_dir = 0;
               motor_dir2 = 0;
               next_count_turn = (count_turn == 32'd200_000_000)? 0 : count_turn+1;
               next_state      = (count_turn == 32'd200_000_000)? state_direct : state_turn;
           end
       endcase
```

Count_Rx 的來源是我們用手機傳送的藍芽訊號，若傳送的訊號是 6，車子進入自動模式，是一個包括三個 state 的 FSM。第一 state 是 state_direct，會讓車子直行。若前方有障礙物則會進入第二個 state，state_back，會倒退約兩秒。最後進入 state_turn，會向左轉約兩秒(實際操作接近轉角 90 度)，再回到 state_direct。

```
    else if(count_Rx == 8'd3)begin //up

        motor_en = 1;
        motor_dir = 0;
        motor_dir2 = 0;

    end
    else if(count_Rx == 8'd4)begin //down

        motor_en = 1;
        motor_dir = 1;
        motor_dir2 = 1;

    end
    else if(count_Rx == 8'd5)begin //stop
        motor_en = 0;
        motor_dir = 1;
        motor_dir2 = 1;
    end
    else if(count_Rx == 8'd7)begin //left

        motor_en = 1;
        motor_dir = 1;
        motor_dir2 = 0;

    end
    else if(count_Rx == 8'd9)begin //right

        motor_en = 1;
        motor_dir = 0;
        motor_dir2 = 1;

    end
```

若傳送的訊號是 3、4、5、7、9 則分別代表上、下、停、左、右，分別給出兩個馬達的 dir。

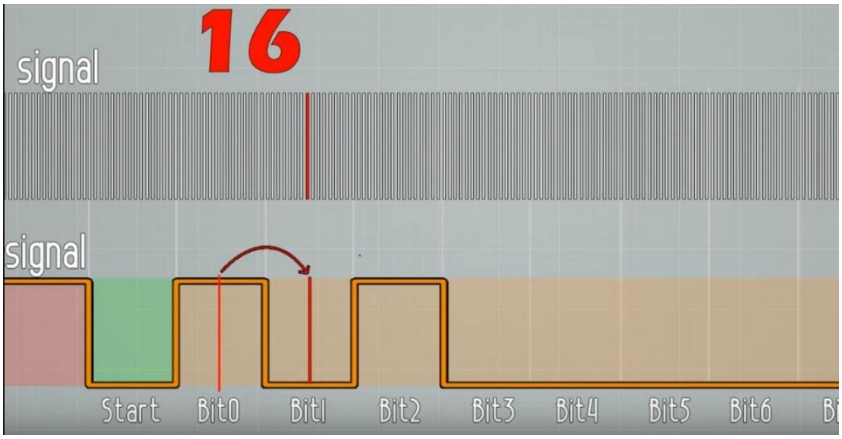
✓ UART_BaudRate_Generate

```
15  input           Clk             ; //Clock input
16  input           Rst_n           ; //Reset input
17  input [15:0]    BaudRate        ; //Value to divide the generator by
18  output          Tick            ; //Each "BaudRate" pulses we create a tick pulse
19  reg [15:0]      baudRateReg     ; //Register used to count
20
21
22  always @(posedge Clk or negedge Rst_n)//begin
23      if (!Rst_n) baudRateReg <= 16'b1;
24      else if (Tick) baudRateReg <= 16'b1;
25          else baudRateReg <= baudRateReg + 1'b1;
26  //end
27  assign Tick = (baudRateReg == BaudRate);
28  endmodule
```

這個 module 是產出 UART_rs232_rx module 的 Tick 訊號。基本上就是一個可以被 input BaudRate 的 clock divider。在 UART_rs232_rx module 的講解會說明 BaudRate 要帶入多少及 Tick 的功用。

✓ **UART_rs232_rx**

這個 module 較為複雜，但原理其實不難。原理示意圖如下：



簡單來說，手機藍牙與藍芽模組是透過 UART 序列平行轉換來傳送資料。我們的藍牙模組的 baud rate 為 9600，我們的 FPGA 時脈是 100MHz，如果我們想要 TICK(圖中的上方訊號)的頻率是 UART 訊號(途中下方訊號)的 16 倍(每個 UART bit 寬度是 16 個 TICK)，那我們會需要頻率是 16*9600HZ 的時脈。UART 的寬度是 1/9600，大約是 104us；FPGA 的時脈寬度是 10ns，因此我們會需要 FPGA 除頻 104us/10ns/16=650，因此，在 TOP module 中，設定 Baud rate=650。

為了取得穩定的訊號值，我們只在 UART 訊號中點讀值。藍牙模組在沒有接收訊號的 idle state 是 high，一接收到訊號後首先是 low 的 start 訊號。在開始是 low 後，我們數 8 個 TICK 以讀到 start bit 的中央，之後每過 16 個 TICK 就再讀一個 bit，直到讀完 8 個 bit 之後出現 high 的 idle state。

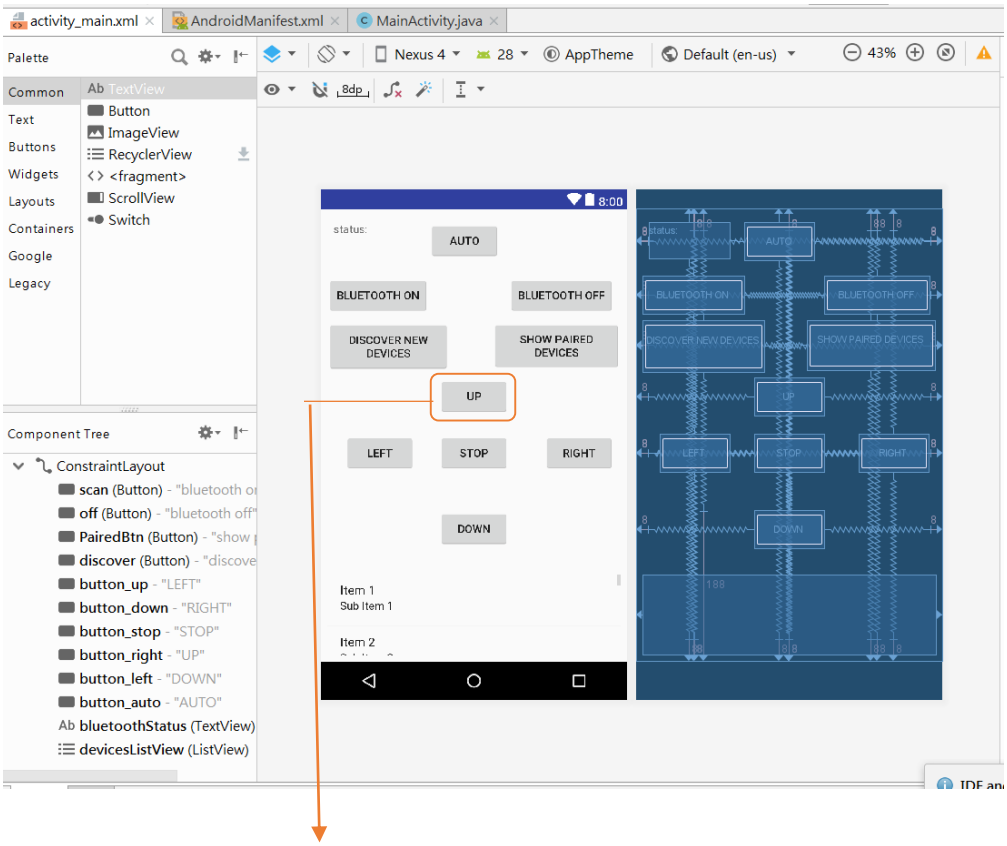
以上方法與圖示來源自
https://www.youtube.com/watch?v=QlscDcbKUV4
不過我們實際的操作結果經各種 debug 後，發現在 idle state 的時候 UART 事實上是 high，並且我們經由手機傳送過去的 char 轉換成數字型態也是不對的，並且，每次傳相同的東西過去，也會讀到不同的值。因為這一點，我們只能懷疑是時脈出了問題，可能沒對齊，造成每次讀的有長有短，讀出來的值也就不相同。這可能來自於我們在算要給定的 Baud rate 時，所做的近似有關。不過，我們沒辦法取得示波器，所以也沒辦法實際檢視這一點。

我們想出的解決方法是，盡量不要用到 TICK，雖然這樣就沒有用到這個 module 裡面的大部分 code 與原理。不過仍是一個方法。在每次有 Rx(讀進的每個 bit 叫 Rx)上升緣時，讓一個 counter+1，因此，在傳送完一筆資料後，就會有一定量的 counter。至於要怎麼再傳完一筆資料後把 counter 清空，則是判斷若 clock 經過一定的數量，就把 counter 清空，意思是，兩次發送資料的時間間隔不能太近，不然有可能出問題。這個時間間隔約是 0.01 秒，經測試，在手機上是非常難在 0.01 秒鐘連按兩次發送資料的，因此，算是解決了這個問題。

✓ APP (Android Studio)

在寫 APP 來控制我們的車子方面，我們使用了 Android studio 這個開發環境來編寫我們的 APP，我在這邊簡單介紹一下我們藍芽訊號是怎麼從手機到達 FPGA 版上的。

在 Android studio 中我們定義了一些 button，如下:



我以讓車子前進的 ”UP” 這個按鈕來舉例說明，當 UP 這個按鈕被按下時就會觸發下圖第 200 行的 onClick 這個 function，此外，在傳送資料前我們得先檢查藍芽是否已經接上裝置，不然有可能會出現 bug 使得我們的 APP 在執行時 shut down，如果藍芽沒有成功連接到裝置，那麼下圖第 201 行的 mConnectedThread 就會等於 null，當確認完藍芽

有正確連接到 FPGA 上的藍芽模組後，我們就用 write 這個 function 將傳送資料出去(當 UP 這個按鈕被按下時，我傳送字串”199” 到 fpga 版上的藍芽模組)，如下圖第 202 行所示:

```
198         button_up.setOnClickListener(new View.OnClickListener(){
199             @Override
200             public void onClick(View v){
201                 if(mConnectedThread != null){     //First check to make sure thread created
202                     mConnectedThread.write( input: "199"); //9
203                 }
204             }
205         });
```

至於我們為什麼要傳送 199 呢? 為什麼不是其他的字串? 其實我也不太清楚，199 這個字串是 trial & error 所得到的答案，我有查過 ascii code，不過也對不起來，而且很多字串都會傳送同一筆資料到 FPGA 上的藍芽模組，不過這並不影響我們的設計，因為我們只需要讓 FPGA 能夠分辨 6 種我所傳送出去的資料就足夠了(因為我們的 APP 有 6 個按鍵會送資料到 FPGA 版上，分別是 AUTO、UP、DOWN、LEFT、RIGHT、STOP 這 6 個按鈕)，trial & error 的結果如下表所示:

APP 傳送出去的資料(字串)	FPGA 版上藍芽模組接收到的資料(整數)
1、2、3、4、6、7、9、a	3
5、m	4
0、8	2
10、18、20、30、40	5
11、12、13、14、16、17、19、21、22、23、50	6
15、100	7
199	9
fuck	14

在 verilog 的 TOP.v 中，我們利用 count_Rx 這個變數來接收手機所傳送的資料，當我們在手機上按下 UP 這個按鈕時，我們從手機傳送字串”199”出去，根據上表，FPGA 接收到的信號會是整數 9，所以當 count_Rx 這個變數等於 9 時，我們就設置一些控制馬達轉動方向的訊號，讓車子向前行進，如下圖所示(motor_en 為 enable 訊號，如果等於 0 馬達就不會轉動，motor_dir 是控制第一個馬達轉動的方向，motor_dir2 是控制第二個馬達轉動的方向，至於為什麼一個是 0 一個

是 1 呢?，這個我在後面講到馬達時會多做說明)。

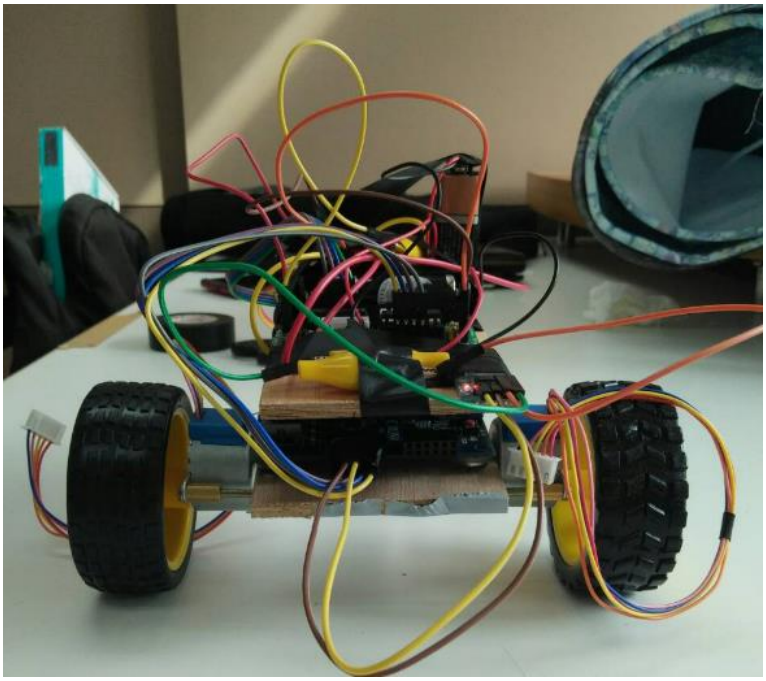
```
else if(count_Rx == 8'd9)begin //up

    motor_en = 1;
    motor_dir = 0;
    motor_dir2 = 1;

end
```

✓ **pmod_step_interface 與 TOP module 中關於馬達部分**

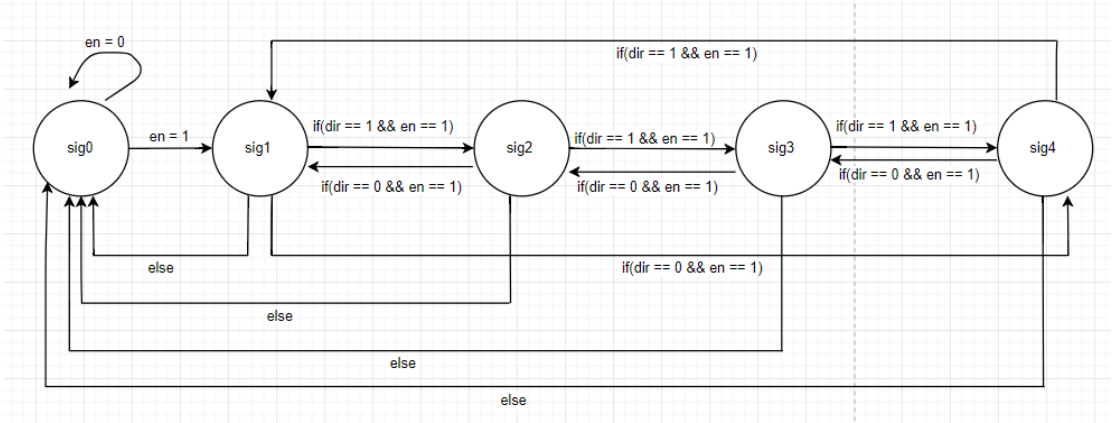
承上所述，在接收到藍芽不同的信號以後，我在 verilog(TOP.v)會設置不一樣的訊號給馬達以控制它的轉動，如下圖所示:

```
else if(count_Rx == 8'd3)begin //left

    motor_en = 1;
    motor_dir = 0;
    motor_dir2 = 0;

end
else if(count_Rx == 8'd4)begin //right

    motor_en = 1;
    motor_dir = 1;
    motor_dir2 = 1;

end
else if(count_Rx == 8'd5)begin //stop
    motor_en = 0;
    motor_dir = 1;
    motor_dir2 = 1;
end
else if(count_Rx == 8'd7)begin //down

    motor_en = 1;
    motor_dir = 1;
    motor_dir2 = 0;

end
else if(count_Rx == 8'd9)begin //up

    motor_en = 1;
    motor_dir = 0;
    motor_dir2 = 1;

end
```

motor_en 的運作原理相當簡單，等於 1 時馬達能動，等於 0 時馬達不能動，motor_dir、mortor_dir2 則是分別用來控制第一個馬達與第二個馬達的轉動方向。我以前進為例(上圖最後一個 else if，count_Rx = 8'd9)，為什麼在前進時兩個馬達的旋轉方向不一樣呢? 其實這個原理也相當簡單，因為兩個輪子安裝在相反的方向上，他們是面對面的(如下圖)，所以要前進時，轉動方向要不同，轉彎時轉動方向相同。



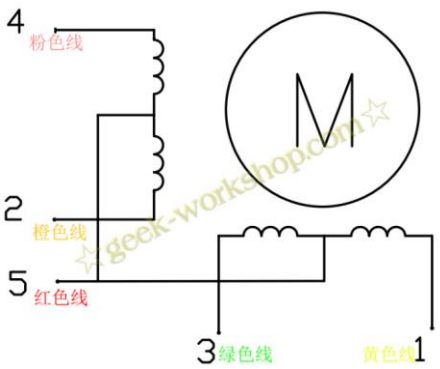
而 TOP.v 中 motor_en、motor_dir、motor_dir2 這三個變數會傳送給 pmod_step_interface 這個 module，然後這個 module 會把 clk 進行除頻後把新的 clk 以及 motor_en、motor_dir、motor_dir2 這三個訊號傳給 pmod_step_driver 這個 module，它是用來控制步進馬達旋轉的，其 state transistion diagram 如下:



上圖中 sig0 是當馬達停止時的 state，也就是當 en=0 時都會回到 sig0。

sig1 ~ sig4 是控制左下圖中這四個磁鐵的 state，如果 dir == 1 就是正轉，dir == 0 則是反轉。

這個 finite state machine 是一個 moore machine，會根據 state 給馬達訊號，如右下圖所示。




```
always @ (posedge clk)
begin
    if (present_state == sig4)
        signal = 4'b1000;
    else if (present_state == sig3)
        signal = 4'b0100;
    else if (present_state == sig2)
        signal = 4'b0010;
    else if (present_state == sig1)
        signal = 4'b0001;
    else
        signal = 4'b0000;
end
```

步進馬達結構圖

✓ 問題與心得：

在改題目之前我們是想做螞蟻養成遊戲，鑒於遊戲類型的 final project 已經寫過太多次，我們比較不感興趣，並且因為這堂課是硬體實驗，我們還是比較想做出一些實體、硬體的東西。做車子的過程中遇到很多問題，像是 ①接線問題：我們用的是杜邦線，有時候會需要和電源線相連接，因為沒辦法用插入的，所以常常斷路而不易察覺，有效的解決方式是用單頭的鱷魚夾夾住，或者可用麵包版，但我們沒有使用麵包版因為可能有額外支出。 ②電壓問題：因為我們買的紅外線模組、藍芽模組在網路上查詢 spec 後，很多都是需要 5V 電壓，但 FPGA 只有 output 3.3 V，因此，我們用了許多不同型號的電池進行串、並聯，但也因此加重了重量。在步進馬達上，因為需要 input 12V，但 12V 需要串聯非常多電池，於是我們買了一個 motor driver，可以 input 3V 左右電壓，輸出 12V 電壓。

總之，這次 project 是眾多 project 中，我們覺得最好玩的，雖然中途換了題目，自己摸索了馬達、藍牙、紅外線遇上不少困難，不過真的成長很多。感謝教授，助教。