

# 第八組- HELLO KITTY之聖誕

## 轉轉音樂盒

B10201054	陳妍合	B10201056	黃冠誌
B10201058	陳義芳	B10201068	王威仁
B10201070	張家榮	B10201072	陳苡秀
B10201100	李奕璋	B10201102	陳佑儒
B10201104	張達瑋	B10201106	王晟譯
B10201108	黃獻清		



# 主程式

```
+ module proje(clock,reset,out25ms,out,outh,outend,LED,Dout);
+ input clock,reset;
+ output out25ms;
+ output out,outh,outend,Dout;
+ output [6:0]LED;
+ wire k;
+ wire [7:0] q;
+ wire [7:0] SW;

+ mode555 U1( .clock(clock),
+ .reset(reset),
+ .OUT(k)
+ );

+ mode225 U2( .clock(k),
+ .reset(reset),
+ .q(q),
+ .OUT(out25ms)
+ );

+ mode15 U3 ( .clk(k),
+ .reset(reset),
+ .Q(SW)
+ );

+ cmp U4 ( .W(SW),
+ .X(q),
+ .OUT(out)
+ );

+ decoder3x8 U5 ( .clock(out),
+ .reset(reset),
+ .DEOUT(outh)
+ );

+ result U6 ( .x(out),
+ .y(outh),
+ .OUT(outend)
+ );

+ Mn U7 ( .F50M(clock),
+ .reset(reset),
+ .Dout(Dout),
+ .LED(LED)
+ );
+ endmodule
```

# 各個小程式-MODE555

```
+ module mode555(clock,reset,OUT);  
+ input clock,reset;  
+ output OUT;  
+ reg OUT;  
+ reg [9:0]q;  
+ always@(negedge reset,posedge clock)  
+ begin  
+     OUT = q[9];  
+     if(!reset);  
+     else if (q==10'd554) q<=0;  
+     else q<=q+1;  
+ end  
+ endmodule
```

# MODE225

```
+ module mode225(clock,reset,q,OUT);  
+ input clock,reset;  
+ output [7:0]q;  
+ reg [7:0]q;  
+ output OUT;  
+ reg OUT;  
+ always@(negedge reset,posedge clock)  
+ begin  
+     OUT = q[7];  
+     if(!reset);  
+     else if (q==8'd224) q<=0;  
+     else q<=q+1;  
+ end  
+ endmodule
```

# MODE15

```
+ module mode15(clk,reset,Q);
+ input clk,reset;
+ output [7:0]Q;
+ reg [7:0]Q;
+ reg [9:0]q;

+ always@(negedge reset,posedge clk)
+ begin
+   if (reset==0) q<=9'd0;
+   else
+     if(q==10'd999) q<=9'd0;
+     else q<=q+1;

+ end
+ always@(negedge reset,posedge q[9])
+ begin
+   if (reset==0) Q<=8'd0;
+   else
+     if(Q==8'd180) Q<=8'd0;
+     else Q<=Q+1;

+ end
+ endmodule
```

# CMP

```
+ module cmp(W,X,OUT);  
+ input [7:0]W;  
+ input [7:0]X;  
+ output OUT;  
+ reg OUT;  
+ reg [7:0]W1;  
+ always@(W,X)  
+ begin  
+     W1 = {W}+8'd45;  
+     if(W1>X) OUT=1;  
+     else OUT=0;  
+ end  
+ endmodule
```

# DECODER3X8

```
+ module decoder3x8(clock,reset,DEOUT);
+ input clock,reset;
+ output [7:0]DEOUT;
+ reg [7:0]DEOUT;
+ reg [2:0] q;

+ always@ (negedge reset,posedge clock)
+ begin
+   if(!reset) q<=0;
+   else q<=q+1;
+ end

+ always@(q)
+ begin
+   case(q)
+     3'd0 : DEOUT = 8'b00000001;
+     3'd1 : DEOUT = 8'b00000010;
+     3'd2 : DEOUT = 8'b00000100;
+     3'd3 : DEOUT = 8'b00001000;
+     3'd4 : DEOUT = 8'b00010000;
+     3'd5 : DEOUT = 8'b00100000;
+     3'd6 : DEOUT = 8'b01000000;
+     default : DEOUT=8'b10000000;
+   endcase
+ end
+ endmodule
```

# RESULT

```
+ module result (x,y,OUT);  
+ input x;  
+ input [7:0]y;  
+ output OUT;  
+ reg OUT;  
+ always@(x,y)  
+ begin  
+     if (y[0]==1&&x==1) OUT=1;  
+     else OUT=0;  
+ end  
+ endmodule
```



# 音樂

```
+ module Mn(F50M,reset,Dout,LED);
+ input F50M,reset;
+ output Dout;
+ reg Dout;
+ output [6:0]LED;
+ reg [6:0]LED;

+ reg [7:0]Do;
+ reg [7:0]Re;
+ reg [7:0]Mi;
+ reg [7:0]Fa;
+ reg [6:0]So;
+ reg [6:0]La;
+ reg [6:0]Si;

+ reg [6:0]Do2;
+ reg [6:0]Re2;
+ reg [6:0]Mi2;
+ reg [6:0]Fa2;
+ reg [5:0]So2;
```

# 音樂

```
+ reg [5:0]La2;

+ reg[9:0]p;
+ reg F4Hz;
+ reg F50KHz;
+ reg [7:0]w;
+ reg [23:0]Q;

+ always@(negedge reset,posedge F50M)
+ begin
+     F50KHz=p[9];
+     if (reset==0) p<=0;
+     else if (p==10'd999) p<=0;
+     else p<=p+1;
+ end

+ always@(posedge F50KHz,negedge reset)
+ begin
+     if (reset==0) Do<=0;
+     else if (Do==8'd190)Do<=0;
```

# 音樂

```
+         else Do<=Do+1;
+     end

+     always@(posedge F50KHz,negedge reset)
+     begin
+         if (reset==0) Re<=0;
+         else if (Re==8'd169)Re<=0;
+         else Re<=Re+1;
+     end

+     always@(posedge F50KHz,negedge reset)
+     begin
+         if (reset==0) Mi<=0;
+         else if (Mi==8'd151)Mi<=0;
+         else Mi<=Mi+1;
+     end

+     always@(posedge F50KHz,negedge reset)
+     begin
+         if (reset==0) Fa<=0;
+         else if (Fa==8'd142)Fa<=0;
```

# 音樂

```
+         else Fa<=Fa+1;
+     end

+     always@(posedge F50KHz,negedge reset)
+     begin
+         if (reset==0) So<=0;
+         else if (So==7'd127)So<=0;
+         else So<=So+1;
+     end

+     always@(posedge F50KHz,negedge reset)
+     begin
+         if (reset==0) La<=0;
+         else if (La==7'd113)La<=0;
+         else La<=La+1;
+     end

+     always@(posedge F50KHz,negedge reset)
+     begin
+         if (reset==0) Si<=0;
+         else if (Si==7'd100)Si<=0;
```

# 音樂

```
+         else Si<=Si+1;
+     end

+     always@(posedge F50KHz,negedge reset)
+     begin
+         if (reset==0) Do2<=0;
+         else if (Do2==7'd95)Do2<=0;
+         else Do2<=Do2+1;
+     end

+     always@(posedge F50KHz,negedge reset)
+     begin
+         if (reset==0) Re2<=0;
+         else if (Re2==7'd84)Re2<=0;
+         else Re2<=Re2+1;
+     end

+     always@(posedge F50KHz,negedge reset)
+     begin
+         if (reset==0) Mi2<=0;
```

# 音樂

```
+     else if (Mi2==7'd75)Mi2<=0;
+         else Mi2<=Mi2+1;
+     end

+ always@(posedge F50KHz,negedge reset)
+ begin
+     if (reset==0) Fa2<=0;
+     else if (Fa2==7'd71)Fa2<=0;
+         else Fa2<=Fa2+1;
+ end

+ always@(posedge F50KHz,negedge reset)
+ begin
+     if (reset==0) So2<=0;
+     else if (So2==6'd63)So2<=0;
+         else So2<=So2+1;
+ end

+ always@(posedge F50KHz,negedge reset)
+ begin
+     if (reset==0) La2<=0;
```

# 音樂

```
+     else if (La2==6'd56)La2<=0;
+         else La2<=La2+1;
+     end

+ always@(negedge reset,posedge F50M)
+ begin
+     F4Hz=Q[23];
+     if (reset==0) Q<=0;
+     else if (Q==24'd12499999) Q<=0;
+     else Q<=Q+1;
+ end

+ always@(posedge F4Hz,negedge reset)
+ begin
+     if (reset==0) w<=0;
+     else if (w==7'd88)w<=0;
+     else w<=w+1;
+ end

+ always@(w)
+ begin
```

# 音樂

```
+      case(w)
+
+      9'd1 : Dout=      0          ;
+      9'd2 : Dout=      0          ;
+      9'd3 : Dout=      La        [6] ;
+      9'd4 : Dout=      Si        [6] ;
+      9'd5 : Dout=      0          ;
+      9'd6 : Dout=      Do2       [6] ;
+      9'd7 : Dout=      Si        [6] ;
+      9'd8 : Dout=      Do2       [6] ;
+      9'd9 : Dout=      Mi2       [6] ;
+      9'd10 : Dout=     0          ;
+      9'd11 : Dout=     Si        [6] ;
+      9'd12 : Dout=     0          ;
+      9'd13 : Dout=     0          ;
+      9'd14 : Dout=     Mi        [7] ;
+      9'd15 : Dout=     0          ;
+      9'd16 : Dout=     La        [6] ;
+      9'd17 : Dout=     So        [6] ;
+      9'd18 : Dout=     La        [6] ;
+      9'd19 : Dout=     Do2       [6] ;
```



# 音樂

```
+ 9'd20 : Dout= 0 ;
+ 9'd21 : Dout= So [6] ;
+ 9'd22 : Dout= 0 ;
+ 9'd23 : Dout= 0 ;
+ 9'd24 : Dout= Mi [7] ;
+ 9'd25 : Dout= Mi [7] ;
+ 9'd26 : Dout= Fa [7] ;
+ 9'd27 : Dout= 0 ;
+ 9'd28 : Dout= Mi [7] ;
+ 9'd29 : Dout= Fa [7] ;
+ 9'd30 : Dout= Do2 [6] ;
+ 9'd31 : Dout= Do2 [6] ;
+ 9'd32 : Dout= 0 ;
+ 9'd33 : Dout= Mi [7] ;
+ 9'd34 : Dout= 0 ;
+ 9'd35 : Dout= 0 ;
+ 9'd36 : Dout= Do2 [6] ;
+ 9'd37 : Dout= Do2 [6] ;
+ 9'd38 : Dout= Do2 [6] ;
+ 9'd39 : Dout= 0 ;
```

# 音樂

+ 9'd40 : Dout= Si [6] ;  
+ 9'd41 : Dout= Fa [7] ;  
+ 9'd42 : Dout= Si [6] ;  
+ 9'd43 : Dout= Si [6] ;  
+ 9'd44 : Dout= 0 ;  
+ 9'd45 : Dout= 0 ;  
+ 9'd46 : Dout= 0 ;  
+ 9'd47 : Dout= La [6] ;  
+ 9'd48 : Dout= Si [6] ;  
+ 9'd49 : Dout= 0 ;  
+ 9'd50 : Dout= Do2 [6] ;  
+ 9'd51 : Dout= Si [6] ;  
+ 9'd52 : Dout= Do2 [6] ;  
+ 9'd53 : Dout= Mi2 [6] ;  
+ 9'd54 : Dout= 0 ;  
+ 9'd55 : Dout= Si [6] ;  
+ 9'd56 : Dout= 0 ;  
+ 9'd57 : Dout= 0 ;  
+ 9'd58 : Dout= Mi [7] ;  
+ 9'd59 : Dout= Mi [7] ;  
+ 9'd60 : Dout= 0 ;

# 音樂

+ 9'd61 : Dout= La [6] ;  
+ 9'd62 : Dout= So [6] ;  
+ 9'd63 : Dout= La [6] ;  
+ 9'd64 : Dout= Do2 [6] ;  
+ 9'd65 : Dout= 0 ;  
+ 9'd66 : Dout= So [6] ;  
+ 9'd67 : Dout= 0 ;  
+ 9'd68 : Dout= 0 ;  
+ 9'd69 : Dout= Mi [7] ;  
+ 9'd70 : Dout= 0 ;  
+ 9'd71 : Dout= Fa [7] ;  
+ 9'd72 : Dout= Do2 [6] ;  
+ 9'd73 : Dout= Si [6] ;  
+ 9'd74 : Dout= Do2 [6] ;  
+ 9'd75 : Dout= 0 ;  
+ 9'd76 : Dout= Re2 [6] ;  
+ 9'd77 : Dout= Mi2 [6] ;  
+ 9'd78 : Dout= Do2 [6] ;  
+ 9'd79 : Dout= 0 ;  
+ 9'd80 : Dout= 0 ;  
+ 9'd81 : Dout= 0 ;

# 音樂

```
+ 9'd82 : Dout=    Do2    [6]    ;
+ 9'd83 : Dout=    Si     [6]    ;
+ 9'd84 : Dout=    La     [6]    ;
+ 9'd85 : Dout=    Si     [6]    ;
+ 9'd86 : Dout=    So     [6]    ;
+ 9'd87 : Dout=    0      [6]    ;
+ 9'd88 : Dout=    La     [6]    ;
+ default : Dout=  0      ;

+   endcase
+ end

+ always@(w)
+ begin
+   case(w)
+     9'd0 : LED=7'b1111111;
+     9'd1 : LED=7'b0011101;
+     9'd2 : LED=7'b1111010;
+     9'd3 : LED=7'b1110111;
+     9'd4 : LED=7'b1101110;
+     9'd5 : LED=7'b1011111;
```

# 音樂

```
+ 9'd6 : LED=7'b0000001;  
+ 9'd7 : LED=7'b1111110;  
+ 9'd8 : LED=7'b0110001;  
+ 9'd9 : LED=7'b0111101;  
+ 9'd10 : LED=7'b0110111;  
+ 9'd11 : LED=7'b1011101;  
+ 9'd12 : LED=7'b0110010;  
+ 9'd13 : LED=7'b1100111;  
+ 9'd14 : LED=7'b0111100;  
+ 9'd15 : LED=7'b1101110;  
+ 9'd16 : LED=7'b0111101;  
+ 9'd17 : LED=7'b1111111;  
+ 9'd18 : LED=7'b0100100;  
+ 9'd19 : LED=7'b1110001;  
+ 9'd20 : LED=7'b0100011;  
+ 9'd21 : LED=7'b1111000;  
+ 9'd22 : LED=7'b0101101;  
+ 9'd23 : LED=7'b0101000;  
+ 9'd24 : LED=7'b1011001;  
+ 9'd25 : LED=7'b0111111;  
+ 9'd26 : LED=7'b1010110;
```

# 音樂

+ 9'd27 : LED=7'b1000110;  
+ 9'd28 : LED=7'b1100001;  
+ 9'd29 : LED=7'b0100100;  
+ 9'd30 : LED=7'b1101101;  
+ 9'd31 : LED=7'b0111000;  
+ 9'd32 : LED=7'b1101011;  
+ 9'd33 : LED=7'b0101000;  
+ 9'd34 : LED=7'b1111011;  
+ 9'd35 : LED=7'b0101000;  
+ 9'd36 : LED=7'b0111101;  
+ 9'd37 : LED=7'b1101000;  
+ 9'd38 : LED=7'b0100001;  
+ 9'd39 : LED=7'b1100000;  
+ 9'd40 : LED=7'b0101001;  
+ 9'd41 : LED=7'b1000100;  
+ 9'd42 : LED=7'b0101001;  
+ 9'd43 : LED=7'b1101100;  
+ 9'd44 : LED=7'b0010101;  
+ 9'd45 : LED=7'b1101001;  
+ 9'd46 : LED=7'b0101100;  
+ 9'd47 : LED=7'b0111001;

# 音樂

+ 9'd48 : LED=7'b1101001;  
+ 9'd49 : LED=7'b0101100;  
+ 9'd50 : LED=7'b0010001;  
+ 9'd51 : LED=7'b1100101;  
+ 9'd52 : LED=7'b0101001;  
+ 9'd53 : LED=7'b1000001;  
+ 9'd54 : LED=7'b0101001;  
+ 9'd55 : LED=7'b1100000;  
+ 9'd56 : LED=7'b0101001;  
+ 9'd57 : LED=7'b1111001;  
+ 9'd58 : LED=7'b0101101;  
+ 9'd59 : LED=7'b1100011;  
+ 9'd60 : LED=7'b0101001;  
+ 9'd61 : LED=7'b1000010;  
+ 9'd62 : LED=7'b0101101;  
+ 9'd63 : LED=7'b1101001;  
+ 9'd64 : LED=7'b0110001;  
+ 9'd65 : LED=7'b1101101;  
+ 9'd66 : LED=7'b0101001;  
+ 9'd67 : LED=7'b1010100;  
+ 9'd68 : LED=7'b0101001;

# 音樂

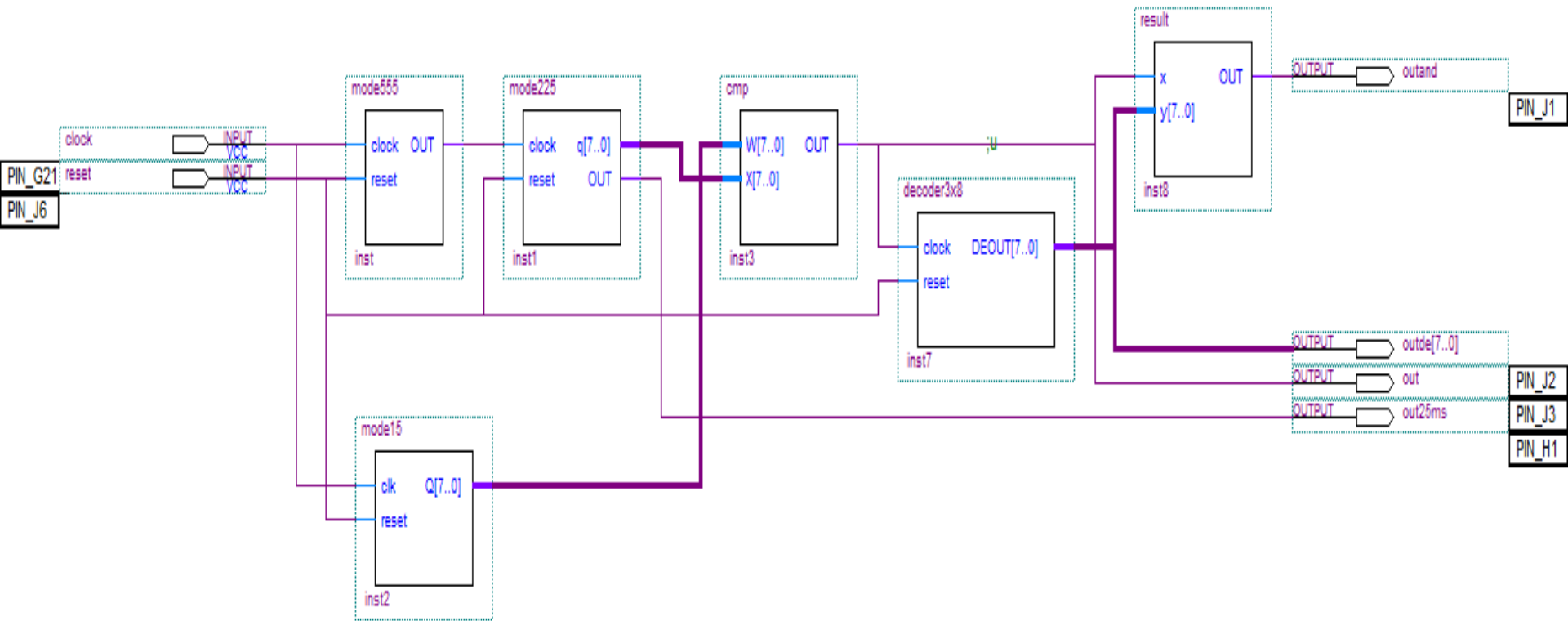
+ 9'd69 : LED=7'b1101001;  
+ 9'd70 : LED=7'b1101101;  
+ 9'd71 : LED=7'b0101001;  
+ 9'd72 : LED=7'b1110001;  
+ 9'd73 : LED=7'b0101101;  
+ 9'd74 : LED=7'b0110001;  
+ 9'd75 : LED=7'b1101001;  
+ 9'd76 : LED=7'b1010101;  
+ 9'd77 : LED=7'b1011010;  
+ 9'd78 : LED=7'b0101000;  
+ 9'd79 : LED=7'b1011111;  
+ 9'd80 : LED=7'b0000111;



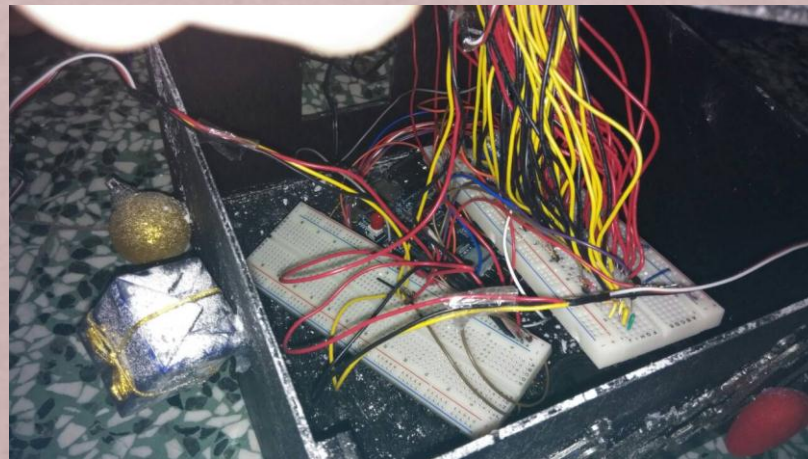
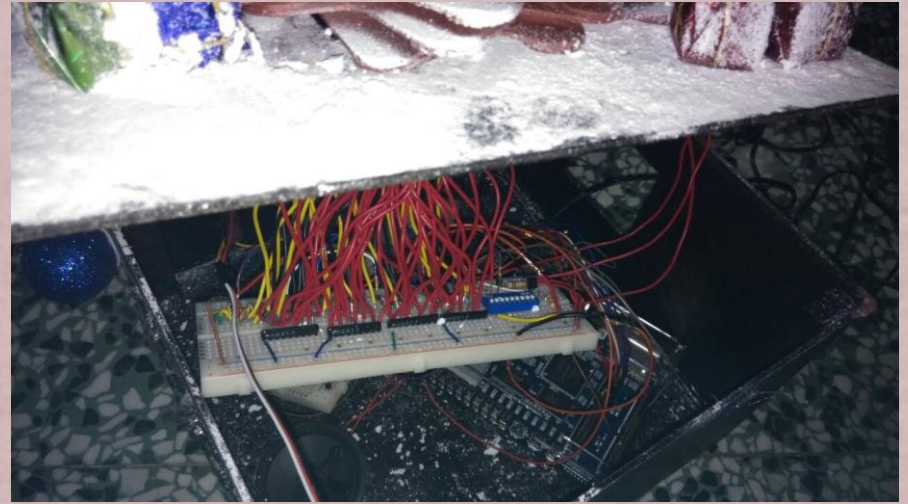
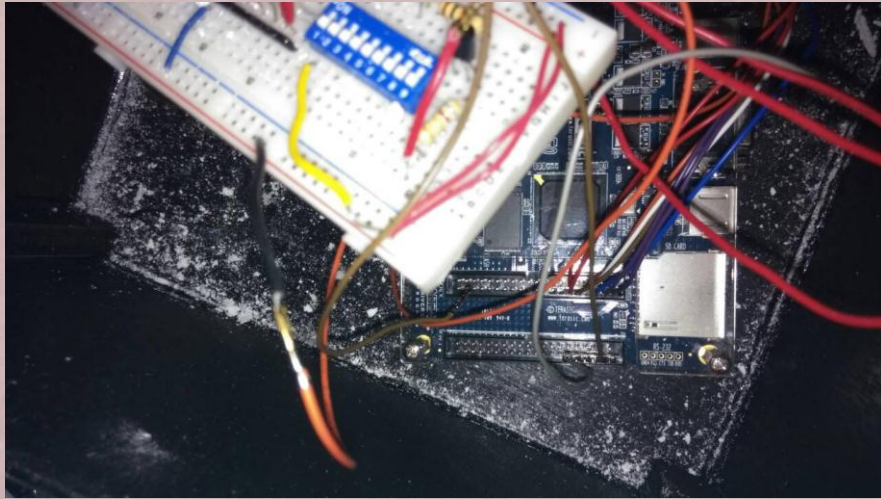
# 音樂

```
+ 9'd81 : LED=7'b1010100;  
+ 9'd82 : LED=7'b1111111;  
+ 9'd83 : LED=7'b0000001;  
+ 9'd84 : LED=7'b0111101;  
+ 9'd85 : LED=7'b1111100;  
+ 9'd86 : LED=7'b1101110;  
+ 9'd87 : LED=7'b1010010;  
+ 9'd88 : LED=7'b1011010;  
+ default LED<=7'bxxxxxxx;  
+ endcase  
+ end  
+ endmodule
```

# 電路圖



# 自己外接的電路圖



# 架構介紹

- + mode555 :  $555 > 512 = 2^9$
- + 位元為[9:0]
- + 因為要把起始值當為 0 所以  $555-1 = 554$
- + 555 的來源為藉由計算最後出來的 2.5ms，往前推所得的值
  
- + Mode225 :  $225 < 256 = 2^8$
- + 位元為[7:0]
- + 因為要算 0 所以  $225-1 = 224$
- + 225的來源為 :我們所要轉動的180度角，在加上為了維持 0 ms，所需的45度角，相加為 225

# 架構介紹

- + **mode 15** : 為代替 SW 輸入的計數器值，藉由在其中另設一個頻率，讓計數器的值，由 0 ~ 180 度轉動，依我們所設的頻率，來轉動
- + **cmp** : 讓 SW 指撥的數值，跟暫存器的值作比較。
- + 但是要維持零度的值，所以基本底給 45 的值，給 179 的話就是直接輸出 2.5ms

# 架構介紹

- + **decoder3x8** : 除8的部分是維持總週期20ms。  
下面的編碼是控制我們所需要的通道部分。  
這樣我們就可以控制8個不同馬達
- + **result** : 單純的AND閘，這樣的設計是方便架構，讓 **cmp** 和 **decoder3x8** 所輸出的值做AND，成為最後的輸出

# 製作過程

- + 某年某月的某一天，不知地球自轉了幾圈，繞太陽公轉了幾圈，我們的王志湖老師，希望我們用馬達做出機器人來，說真的，一點頭緒也沒有，也不知該如何動手，就這樣荒廢了幾天。但最後，我們想到，結合上學期所做的音樂盒加LED燈，在組合到馬達上，做出有音樂、燈光閃爍以及又會動的作品，這就是我們初期的構想。

# 製作過程

- + 接下來，我們就是打好我們需要的程式，燒到板子上面，然後連接到我們的馬達、音樂盒及燈，等一切都成功之後，就可以把它組裝起來了。所以我們就需要去買材料了，等把材料都買齊後，我們就把它組起來了。外型為聖誕樹，有天空之城的音樂，繽紛的LED燈，以及最主要的讓hello kitty 在其上面旋轉，雖然可能很普通，但這卻是我們這組所有人努力的結晶，



# 製作過程

- + 不管它被評論為怎樣，只要知道它是我們一起用心做出來的作品，那我們就心滿意足了。
- + 這就是我們這一組一起努力、汗血的結晶

# 影片

