该工程由 测速模块，输入控制模块模块，数码管显示模块，pwm 波形发生器模块组成。

1，测速模块

光电测速传感器在传感器草有障碍物遮挡时，输出高电平，在没有障碍物遮挡时，输出低电平。电机马达每转一圈，遮挡一次传感器。

在 fpga 端，检测传感器每两个高电平的时间间隔，即可换算出电机马达的速度。

具体实现原理为，利用上升沿检测原理，检测出传感器每次输出电平由低变为高的时间，并利用计数器记录两个上升沿之间的时间间隔。具体代码如下：

（1）上升沿检测。信号 speed_rise 为上升沿标志信号。当 speed_rise 为高时，说明存在 speed 信号的上升沿。信号 speed 为测速模块的输出信号。用两级寄存器接收传感器信号，不仅达到了消抖的专用，同时也达到了上升沿检测的目的。

```
always@(posedge clk,negedge rst_n)
begin
    if(!rst_n)
        begin
            speed_reg1  <=  1'b0;
            speed_reg2  <=  1'b0;
        end
    else
        begin
            speed_reg1  <=  speed;
            speed_reg2  <=  speed_reg1;
        end

end

always@(posedge clk,negedge rst_n)
begin
    if(!rst_n)
        speed_rise  <=  1'b0;
    else if( !speed_reg2 && speed_reg1  )
        speed_rise  <=  1'b1;
    else
        speed_rise  <=  1'b0;
end
```

（2）利用计数器记录每两个上升沿之间的时间。speed_count 为速度计数值，speed_count 本质记录的是两个上升沿之间的时钟计数值。

```
always@(posedge clk,negedge rst_n)
begin
    if(!rst_n)
        speed_count <=  'b0;
    else if( speed_rise )
        speed_count <=  'b0;
    else
        speed_count <=  speed_count +   1'b1;
end
```

（3）1 秒更新一次计数值，用于数码管的显示。time_cnt **用于 1 秒的计数，当时间大于 1 秒且遇到传感器输出信号的上升沿时，接收一次速度计数值。**speed_Average **用于缓存每一秒接收到的速度计数值。**

```
always@(posedge clk,negedge rst_n)
begin
    if(!rst_n)
        time_cnt    <=  'b0;
    else if( speed_rise && (  time_cnt >= time_ones))
        time_cnt    <=  'b0;
    else
        time_cnt    <=  time_cnt    +   1'b1;
end

always@(posedge clk,negedge rst_n)
begin
    if(!rst_n)
        speed_Average   <=  'b0;
    else if( speed_rise && (  time_cnt >= time_ones))
        speed_Average   <=  speed_count[40:14];
    else
        speed_Average   <=  speed_Average;
end
```

（3）提取速度计数值的每一位的值，用于传输到数码管显示模块中去显示。

seg0，seg1，seg2，seg3，seg4，seg5。分别对应 6 个数码管所显示的内容。将 **speed_Average 对应的 10 进制的每一位提取到** seg0，seg1，seg2，seg3，seg4，seg5 中，送到数码管显示模块中进行显示。

```
always@(posedge clk,negedge rst_n)
begin
    if(!rst_n)
        begin
            seg0    <=  4'b0;
            seg1    <=  4'b0;
            seg2    <=  4'b0;
            seg3    <=  4'b0;
            seg4    <=  4'b0;
            seg5    <=  4'b0;
        end
    else
        begin
            seg5    <=  ( speed_Average %1000000 ) / 100000 ;
            seg4    <=  ( speed_Average %100000 ) / 10000   ;
            seg3    <=  ( speed_Average %10000 ) / 1000     ;
            seg2    <=  ( speed_Average %1000 ) / 100       ;
            seg1    <=  ( speed_Average %100 ) / 10         ;
            seg0    <=  ( speed_Average %10 ) / 1           ;
        end
end
```

2，数码管显示模块

数码管显示模块用于将接收到的数字解码为数码管每个段位的亮灭情况，达到是数码管显示数字的目的。

```
//实现数码管的解码
always @ (posedge clk,negedge rst_n)
begin
    if( !rst_n )
        SEG_HEX <=  7'b0111111;
    else
        begin
            case(seg_num)
                0  : SEG_HEX[6:0] = 7'b1000000      ;           //显示数字 "0"
                1  : SEG_HEX[6:0] = 7'b1111001      ;       //显示数字 "1"
                2  : SEG_HEX[6:0] = 7'b0100100      ;           //显示数字 "2"
                3  : SEG_HEX[6:0] = 7'b0110000      ;           //显示数字 "3"
                4  : SEG_HEX[6:0] = 7'b0011001      ;       //显示数字 "4"
                5  : SEG_HEX[6:0] = 7'b0010010      ;           //显示数字 "5"
                6  : SEG_HEX[6:0] = 7'b0000010      ;           //显示数字 "6"
                7  : SEG_HEX[6:0] = 7'b1111000      ;           //显示数字 "7"
                8  : SEG_HEX[6:0] = 7'b0000000      ;           //显示数字 "8"
                9  : SEG_HEX[6:0] = 7'b0010000      ;       //显示数字 "9"
                10 : SEG_HEX[6:0] = 7'b0001000      ;           //显示数字 "A"
                11 : SEG_HEX[6:0] = 7'b0000011      ;           //显示数字 "B"
                12 : SEG_HEX[6:0] = 7'b0100111      ;           //显示数字 "C"
                13 : SEG_HEX[6:0] = 7'b0100001      ;           //显示数字 "D"
                14 : SEG_HEX[6:0] = 7'b0000110      ;           //显示数字 "E"
                15 : SEG_HEX[6:0] = 7'b0001110      ;           //显示数字 "F"
                default :SEG_HEX[6:0] = 7'b0111111; //显示数字 "0"
            endcase
        end
end
```

3，输入控制模块

输入控制模块负责控制 pwm 波的占空比，电机转向等

（1）按键消抖模块

用一个 1ms 的计数器，用来 1ms 接收一次按键的输入值，达到按键消抖的目的。key_cnt 是 1ms 计数器。key_out 为消抖后的按键的输出值。

```
always@(posedge clk,negedge rst_n)
begin
    if(!rst_n)
        key_cnt <=  'b0;
    else if( key_cnt == 50_000 )
        key_cnt <=  'b0;
    else
        key_cnt <=  key_cnt +   1'b1;
end

always@(posedge clk,negedge rst_n)
begin
    if(!rst_n)
        key_reg1    <=  'b0;
    else if( key_cnt == 50_000 )
        key_reg1    <=  key_in;
    else
        key_reg1    <=  key_reg1;
end

always@(posedge clk,negedge rst_n)
begin
    if(!rst_n)
        key_reg2    <=  'b0;
    else
        key_reg2    <=  key_reg1;
end

assign  key_out =   key_reg2 & ( ~key_reg1 );
```

（2）电机速度调节

默认占空比为 0.9，当 key3 按下时，减少 pwm 波的占空比，达到使电机减速的目的。当按下 key2 时，增加 pwm 波的占空比，达到使电机加速的目的。

```
always@(posedge clk,negedge rst_n)
begin
    if(!rst_n)
        clock_high  <=  90000;
    else if( ( key_out == 3'b010 ) && ( clock_high > 55000) )
        clock_high  <=  clock_high  -   5000;
    else if( ( key_out == 3'b001 ) && ( clock_high < 95000) )
        clock_high  <=  clock_high  +   5000;
    else
        clock_high  <=  clock_high  ;
end
```

（3）电机转动方向调节

Key1 用于调节电机的转动方向。原理是切换 pwm 波输出的引脚。每按下一次 key1 按键，电机的转向发生一次改变。

```
always@(posedge clk,negedge rst_n)
begin
    if(!rst_n)
        pwm_out_sel <=  'b0;
    else if(key_out == 3'b100)
        pwm_out_sel <=  ~pwm_out_sel;
    else
        pwm_out_sel <=  pwm_out_sel;
end

assign  pwm_out1    =   pwm_out_sel ?   pwm_out :   1'b0    ;
assign  pwm_out2    =   pwm_out_sel ?   1'b0    :   pwm_out ;
```

4，pwm 波形产生模块

（1）硬件设计

每路寄存器都有独立的寄存器用于参数设置和模块的控制。波形占空比控制是基于时钟数来控制的。

基于计数器的 PWM 波形发生器，在不工作时，该模块要停止计数，用以降低功耗。

（2）参数寄存器：

波形周期数寄存器

波形高电平周期数寄存器

（3）控制寄存器：

总的使能信号，该信号无效时，整个模块不工作。

单路使能寄存器，用于控制每一路有效。信号有效时，启动该路模块，否则关闭该路模块。

具体的代码实现:

(1)接收占空比参数

```
always@(posedge clk,negedge rst_n)
begin
    if(!rst_n)
        begin
            clock_total_reg <=  'b0 ;
            clock_high_reg  <=  'b0 ;
        end
    else
        begin
            clock_total_reg <=  clock_total ;
            clock_high_reg  <=  clock_high  ;
        end
end
```

（2）计数器，用于记录 pwm 周期。并产生 pwm 波形。

```
//pwm clock count
always@(posedge clk,negedge rst_n)
begin
    if(!rst_n)
        clock_count     <=  'b0 ;
    else if( !pwm_en )
        clock_count     <=  'b0 ;
    else if(    clock_count ==  clock_total_reg - 1 )
        clock_count     <=  'b0 ;
    else
        clock_count     <=  clock_count +   1'b1    ;
end

assign  pwm_wire    =   (   clock_count <   clock_high_reg )    ?   1'b1    :   1'b0    ;
```

（4）pwm 波输出使能

当输出使能信号 pwm_out_en 为 1 时，输出 pwm 波形信号。当该是能信号为 0 时，pwm 输出一直为 0.

```
always@(posedge clk,negedge rst_n)
begin
    if(!rst_n)
        pwm_out <=  'b0 ;
    else if(pwm_out_en)
        pwm_out <=  pwm_wire    ;
    else
        pwm_out <=  'b0 ;
end
```