同济大学计算机系

数字逻辑课程综合实验报告

学　　号 **1651390**

姓　　名 刘思源

专　　业 信息安全

授课老师 郭玉臣

目录

一、 实验内容 3
二、 温湿度检测智能小车数字系统总框图 3
三、 系统控制器设计 5
四、 子系统模块建模 7
五、 测试模块建模 21
六、 实验结果 23
七、 遇到的问题及解决 26
八、 未完成的蓝牙模块 27
九、 心得体会及建议 33
附录 使用说明 35

一、实验内容

名称： 基于 FPGA 的检测温湿度红外循迹可变速小车

背景：

轨道式自动引导车 RGV 在流水化生产中起到了相当大的作用，在进行加工清洗作业的时候有时需要进行过热检测。本文实现的基于 FPGA 的检测温湿度红外循迹可变速小车作为原型机，可实时监测 CNC 机床的温度和清洗作业的湿度。

设备：

1、硬件：L298N 电机，HL-1 型小车底板，Nexys4 DDR 开发板

2、传感器：温度湿度模块 DHT22，红外反射传感器 TCRT5000。

功能：

1、实时监测当前环境温度湿度，并将温度和湿度以整数形式显示在数码管上；

2、可实现两种运动模式：手动控制运动和红外循迹自动运动：

a) 手动模式下，使用 Nexys4 DDR 上方向键控制小车运动；

b) 自动模式下，小车根据黑线（其他颜色也可，但应为深色）自动循迹移动；

3、可手动控制小车速度，共有三种速度可选

亮点：

1、可同时显示当前环境的温度湿度，精度可调；

2、可使用两种方式进行小车移动方式的控制；

3、三种速度可供用户选择，便于精细操作

使用说明：

见附录 1

二、温湿度检测智能小车数字系统总框图

（按由顶向下方法进行子系统的划分，给出包含各子系统相互关系及控制信号的总框图，并对各子系统功能及实现进行概述。具体可参考教材 183 页的相关描述方法。）

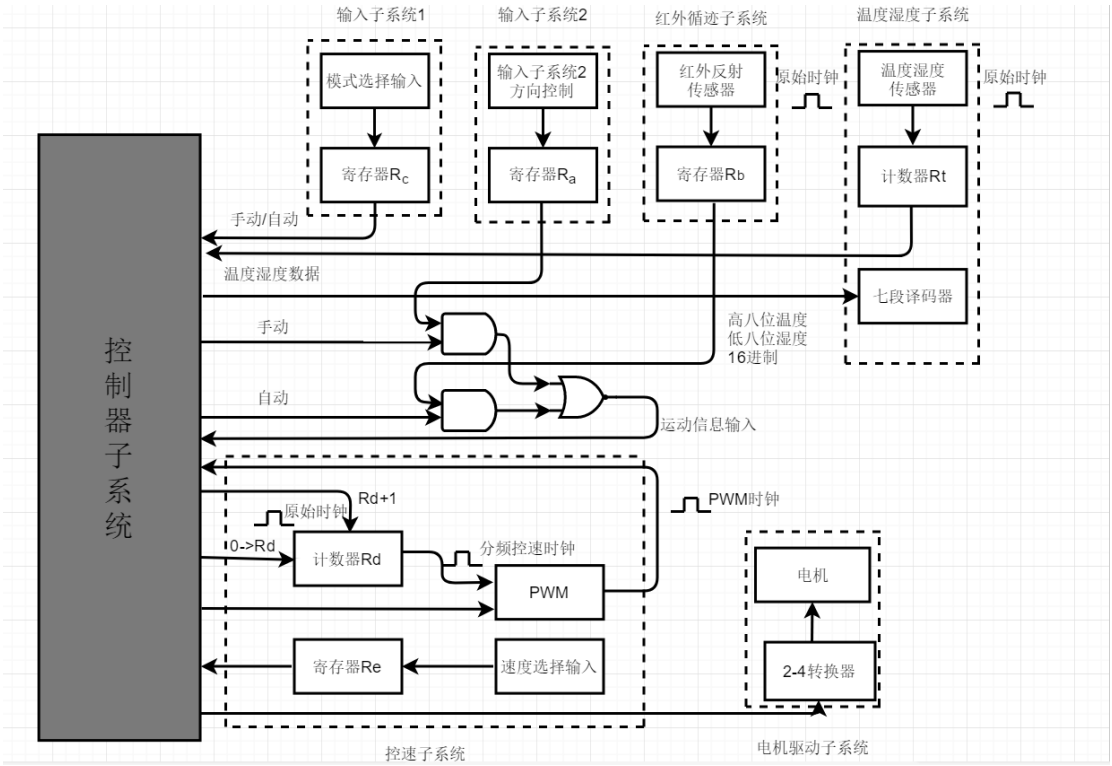


图 1 组成总框图

图 1 示出了检测温湿度红外循迹可变速小车系统的组成总框图，它可以划分为以下六个子系统：

1、 **输入子系统 1** 由 Nexys4 DDR 板上 V10，U11 两个开关组成。使用 V10 设置运动模式为自动模式，使用 U11 将运动模式设置为手动模式。

2、 **输入子系统 2** 由 Nexys4 DDR 板上方向按键组成。对应的方向按钮可设置小车手动运动模式下运动的方向。

3、 **红外循迹子系统** 设备搭载了两个 TCRT5000 红外反射模块，置于小车运动方向前端车底。当车底有黑线时，传感器返回 1；否则返回 0。根据传感器返回的数值存储在寄存器 b 中。若数据显示底部有黑线，则说明方向正确无需更改输出；若有一方显示无黑线，证明小车向无黑线一方偏移，应更改输出。

4、 **控速子系统** 控速子系统由 Nexys4 DDR 板上 L16、M13、R15 三个开关、一个分频器、一个 PWM 模块组成。通过三个开关用户选择慢、中、快三种速度模式，通过分频器对原始的时钟进行调整、再通过 PWM 模块调整输出信号的占空比，来达到控速的效果。占空比越高，速度越慢。

5、 **电机驱动子系统** 设备搭载了两个 L298N 电机，电压使用 6V 蓄电池进行供电。根据输入的运动方向信号调整输出的值，用一个 2-4 的转换将寄存器 b 的两项值转换为 4 管脚的输出。控制左右电机的前转后转。同时前进为小车前进，左电机前转右电机不动为右转；右电机前转左电机不动为左转。倒车同理。

6、 **温度湿度检测子系统** 设备搭载了一个 DHT22 模块，上接一个

AM2302 模块检测温度湿度。由于传感器传入的信号一共 40 位，前 32 位为有效数据，需要计数器进行计数，同时根据计数器判断 0/1 数据。采集的数据要根据七段译码器显示在数码管上，数码管左侧四位显示温度、右侧四位显示湿度（实际上每个都只是用了两个数码管）。皆为整数显示。

三、系统控制器设计

（要求画出所设计数字系统的 ASM 流程图，列出状态转移真值表。由状态转移真值表，求出系统控制器的次态激励函数表达式和控制命令逻辑表达式，并用 Logisim 画出系统控制器逻辑方案图。）

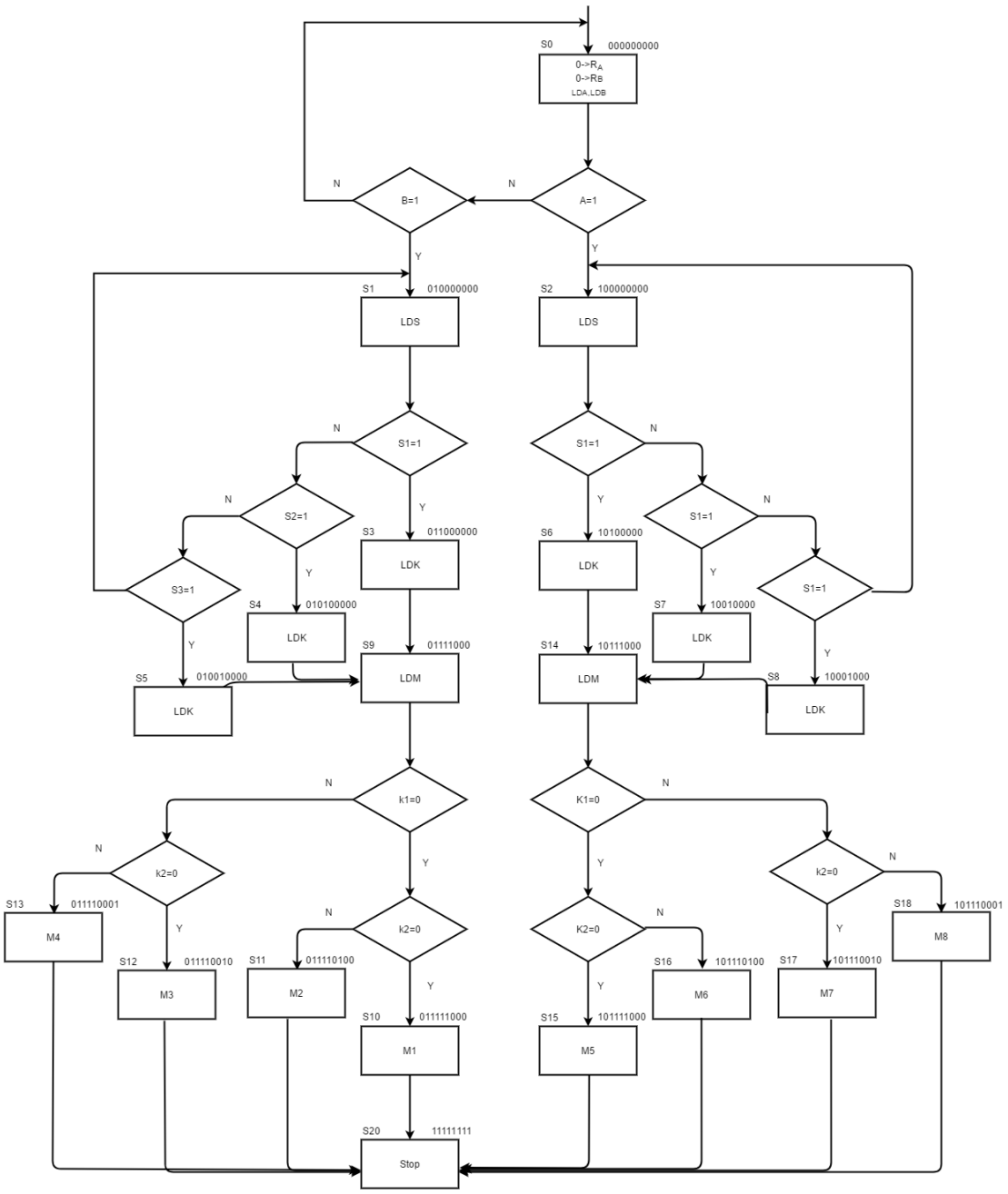


图 2 控制系统 ASM

根据 ASM 流程图，我们可以得到它的状态转移真值表。如图 3 所示：

状态转移真值表

PS									PS									
状态名	A	C	S1	S2	S3	K0	K1	K2	状态名	A(D)	C(D)	S1(D)	S2(D)	S3(D)	K0(D)	K1(D)	K2(D)	转移条件
S0	0	0	X	X	X	X	X	X	S1	0	1	0	0	0	0	0	0	A=1
S0	0	0	X	X	X	X	X	X	S2	1	0	0	0	0	0	0	0	A=0,B=1
S1	0	1	0	0	0	0	0	0	S3	0	1	1	0	0	0	0	0	S1=1，A=1
S1	0	1	0	0	0	0	0	0	S4	0	1	0	1	0	0	0	0	S2=1，A=1
S1	0	1	0	0	0	0	0	0	S5	0	1	0	0	1	0	0	0	S3=1，A=1
S2	1	0	0	0	0	0	0	0	S6	1	0	1	0	0	0	0	0	S1=1，B=1，A=0
S2	1	0	0	0	0	0	0	0	S7	1	0	0	1	0	0	0	0	S2=1，B=1，A=0
S2	1	0	0	0	0	0	0	0	S8	1	0	0	0	1	0	0	0	S3=1，B=1，A=0
S3	0	1	1	0	0	0	0	0	S9	0	1	1	1	1	0	0	0	S1+S2+S3=1
S4	0	1	0	1	0	0	0	0	S9	0	1	1	1	1	0	0	0	S1+S2+S3=1
S5	0	1	0	0	1	0	0	0	S9	0	1	1	1	1	0	0	0	S1+S2+S3=1
S6	1	0	1	0	0	0	0	0	S14	1	0	1	1	1	0	0	0	S1+S2+S3=1
S7	1	0	0	1	0	0	0	0	S14	1	0	1	1	1	0	0	0	S1+S2+S3=1
S8	1	0	0	0	1	0	0	0	S14	1	0	1	1	1	0	0	0	S1+S2+S3=1
S9	0	1	1	1	1	0	0	0	S10	0	1	1	1	1	1	0	0	K1=0,K2=0
S9	0	1	1	1	1	0	0	0	S11	0	1	1	1	1	1	1	0	K1=1,K2=0
S9	0	1	1	1	1	0	0	0	S12	0	1	1	1	1	1	0	1	K1=0,K2=1
S9	0	1	1	1	1	0	0	0	S13	0	1	1	1	1	1	1	1	K1=1,K2=1
S10	0	1	1	1	1	1	0	0	S19	1	1	1	1	1	1	1	1	
S11	0	1	1	1	1	1	1	0	S19	1	1	1	1	1	1	1	1	
S12	0	1	1	1	1	1	0	1	S19	1	1	1	1	1	1	1	1	
S13	0	1	1	1	1	1	1	1	S19	1	1	1	1	1	1	1	1	
S14	1	0	1	1	1	0	0	0	S15	1	0	1	1	1	1	0	0	K1=0,K2=0
S14	1	0	1	1	1	0	0	0	S16	1	0	1	1	1	1	1	0	K1=1,K2=0
S14	1	0	1	1	1	0	0	0	S17	1	0	1	1	1	1	0	1	K1=0,K2=1
S14	1	0	1	1	1	0	0	0	S18	1	0	1	1	1	1	1	1	K1=1,K2=1
S15	1	0	1	1	1	1	0	0	S19	1	1	1	1	1	1	1	1	
S16	1	0	1	1	1	1	1	0	S19	1	1	1	1	1	1	1	1	
S17	1	0	1	1	1	1	0	1	S19	1	1	1	1	1	1	1	1	
S18	1	0	1	1	1	1	1	1	S19	1	1	1	1	1	1	1	1	

图 3 状态转移真值表

由状态转移真值表，利用$NS = \sum PS \cdot C$公式，可求出次态激励函数表达式如下：

$$K1(D) = A(S_1 + S_2 + S_3)$$

$$K2(D) = AB(S_1 + S_2 + S_3) + K1$$

$$K3(D) = \overline{A}B(S_1 + S_2 + S_3) + K4$$

$$K4(D) = \overline{A}(S_1 + S_2 + S_3) + K3$$

控制命令的逻辑表达式为：

$$LDA = A + AB$$

$$LDB = B$$

$$LDS = (A + B)(S_1 + S_2 + S_3)$$

$$LDK = (A + B)((S_1 + S_2 + S_3))\left(\overline{K1} + \overline{K2}\right)\left(\overline{K3} + \overline{K4}\right)$$

控制器逻辑图如图 4：

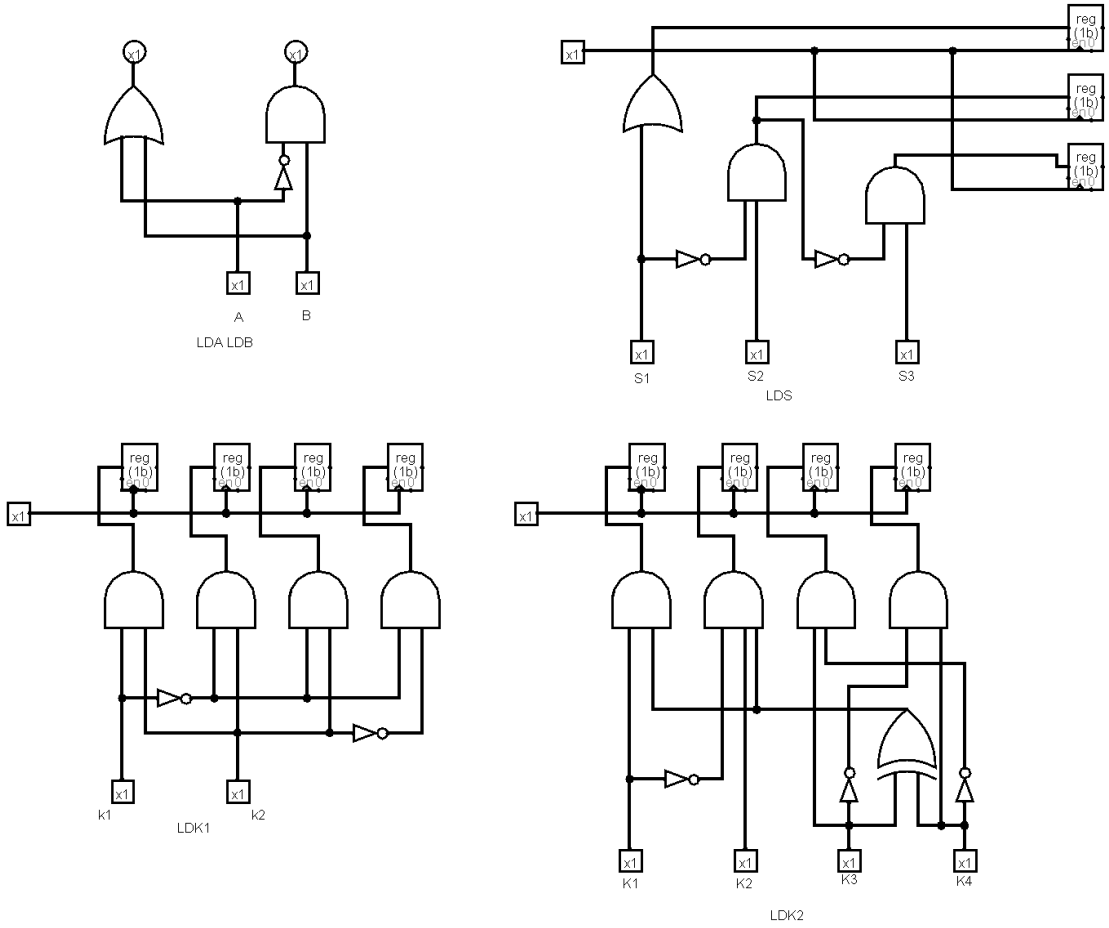


图 4 系统控制器逻辑方案图

四、子系统模块建模

（该部分要求对实验中的所有子系统模块进行描述，给出各子系统的功能框图及接口信号定义，并列出各模块建模的 verilog 代码）

1、 输入子系统 1

输入子系统 1 直接通过板上的开关按钮进行输入，结果储存在寄存器中。代码整合在电机驱动模块中。

2、 输入子系统 2

输入子系统 2 直接通过板上的开关按钮进行输入，结果储存在寄存器中。代码整合在电机驱动模块中。

3、 红外循迹子系统

红外线传感器包括光学系统、检测元件和转换电路。光学系统按结构不同可分为透射式和反射式两类。检测元件按工作原理可分为热敏检测元件和光电检测元件。热敏元件应用最多的是热敏电阻。热敏电阻受到红外线辐射时温度升高，电阻发生变化，通过转换电路变 成电信号输出。

包括变送电路：模拟量；数字量：将模拟量经过比较器输出开关量。在一般

设计等对循迹功能要求不高的场合，完全可以采用比较器输出开关量，这样编程简单，易于实现。2 路循迹模块则输出 2 路开关量，可以接开发板的普通输入 IO 口。

本段描述时以 0、1、x 表示传感器的状态，0 表示传感器下方为白，1 表示传感器下方为黑，x 表示不关心这一位传感器的状态。

在一般路况中只要有中间黑（00x11x00）、两边黑（1xxxxxx1，用以通过井字弯）、全黑（11111111，用以通过十字弯）的情况即前进；

小车左偏（0x0xxx11）即右转； 小车右偏（11xxx0x0）即左转；

代码整合在电机驱动模块中。

4、 温度湿度子系统

温度湿度子系统包含了三个子模块：传感器采集模块、时钟分频模块、七段数码管显示模块。

温度采集模块需要在采集数据前先拉低 800us，本文采用的做法是直接拉低 1ms 的时间。紧接着有两个 80us 的释放信号。在这些时间结束后，温度传感器开始向外界输送信息。

获取 DHT22 传感器的数据输入。本文采用的方法是，不论两种信号的低电平持续多长时间，只判断高电平。首先将时钟分为 10μs，0 的高电平持续时间为 26μs，1 的高电平持续时间为 72μs。判断如果高电平持续时间超过了 3 个时钟，则为 1，反之为 0。

最后显示在七段数码管上，前四位显示温度，后四位显示湿度。均已十六进制的整数显示。

接口定义：

```
1.  input     I_clk,    //原始时钟
2.  inout     I_O_sda ,  //数据输入输出接口
3.  output    [7:0]  O_seg_060,   //温度数码管
4.  output    [7:0]  O_sel_060     //湿度数码管
```

功能框图：

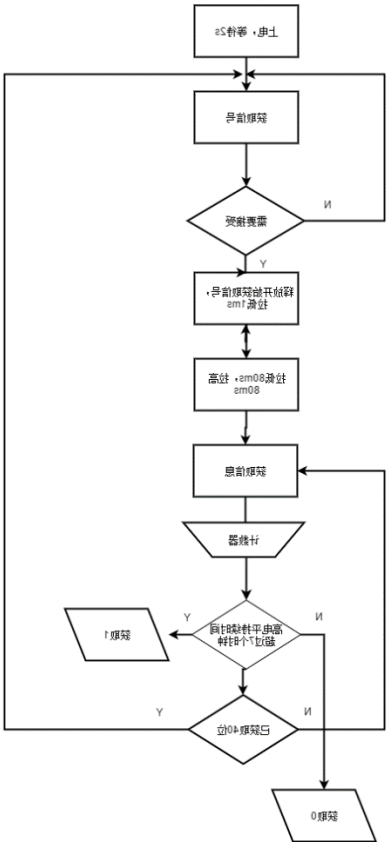


图 5 温度湿度框图

代码：

```
1. `timescale 1ns / 1ps
2. //////////////////////////////////////////////////////////////////////////////////
   //////
3. // Company:
4. // Engineer:
5. //
6. // Create Date: 2019/01/01 12:33:35
7. // Design Name:
8. // Module Name: process_060
9. // Project Name:
10. // Target Devices:
11. // Tool Versions:
12. // Description:
13. //
14. // Dependencies:
15. //
16. // Revision:
17. // Revision 0.01 - File Created
18. // Additional Comments:
19. //
```

```
////////////////////////////////////////////////////////////////////////////////
//////


module process_060(
    input               I_clk,
    inout               I_O_sda ,
    output    [7:0]  O_seg_060       ,
    output    [7:0]  O_sel_060
);

wire clk_1khz ;
wire clk_100khz;
wire      [3:0]   O_num1  ;
wire      [3:0]   O_num2  ;
wire      [3:0]   O_num3  ;
wire      [3:0]   O_num4  ;
wire      [3:0]   O_num5  ;
wire      [3:0]   O_num6  ;
wire      [3:0]   O_num7  ;
wire      [3:0]   O_num8  ;


det
det
(
    .I_clk   ( I_clk ),  //100khz
    .I_O_sda ( I_O_sda    ),
    .O_num1  ( O_num1     ),
    .O_num2  ( O_num2     ),
    .O_num3  ( O_num3     ),
    .O_num4  ( O_num4     ),
    .O_num5  ( O_num5     ),
    .O_num6  ( O_num6     ),
    .O_num7  ( O_num7     ),
    .O_num8  ( O_num8     )

);

divclk_060
divclk_060
(
    .inclk_060  ( I_clk       ),
    .outclk_060 ( clk_1khz    )
```

```
63. );
64. /*
65. wire clk1khz ;
66. 
67.    BUFG BUFG_inst1 (
68.       .O(clk1khz), // 1-bit output: Clock output
69.       .I(clk_1khz)  // 1-bit input: Clock input
70.    );
71. 
72. */
73. sel_det_060
74. u_sel
75. (
76.    .I_clk_div_060 ( clk_1khz ),
77.     //  .I_clk_div_060 ( clk_seg_060 ),
78.    .I_num1_060    ( O_num1      ),
79.    .I_num2_060    ( O_num2      ),
80.    .I_num3_060    ( O_num3      ),
81.    .I_num4_060    ( O_num4      ),
82.    .I_num5_060    ( O_num5      ),
83.    .I_num6_060    ( O_num6      ),
84.    .I_num7_060    ( O_num7      ),
85.    .I_num8_060    ( O_num8      ),
86.    .O_seg_060     ( O_seg_060  ),
87.    .O_sel_060     ( O_sel_060  )
88.     );
89. 
90. 
91. endmodule
92. module det(
93.     input               I_clk   ,  //100khz
94. (*mark_debug="true"*)    inout               I_O_sda ,
95. (*mark_debug="true"*)    output      [3:0]   O_num1  ,
96. (*mark_debug="true"*)    output      [3:0]   O_num2  ,
97. (*mark_debug="true"*)    output      [3:0]   O_num3  ,
98. (*mark_debug="true"*)    output      [3:0]   O_num4  ,
99. (*mark_debug="true"*)    output      [3:0]   O_num5  ,
100. (*mark_debug="true"*)    output      [3:0]   O_num6  ,
101. (*mark_debug="true"*)    output      [3:0]   O_num7  ,
102. (*mark_debug="true"*)    output      [3:0]   O_num8
103. 
104. );
105. 
106. parameter  total_mult   = 100_000_000   ;//total_mult   = 100_000   ;
```

```
107.
108. (*mark_debug="true"*)reg [31: 0]  clk_cnt = 0 ;
109.
110. always@(posedge I_clk )begin
111.     if( clk_cnt == total_mult - 1  )
112.         clk_cnt <= 0;
113.     else
114.         clk_cnt <= clk_cnt + 1'b1;
115. end
116.
117.
118. (*mark_debug="true"*)reg out_en;
119. (*mark_debug="true"*)reg sda_temp;
120.
121. always@(posedge I_clk )begin
122.     if(( clk_cnt >= 100_000  ) && ( clk_cnt < 200_000 ) )
123.         out_en <= 1;
124.     else
125.         out_en <= 0;
126. end
127.
128. always@(posedge I_clk )begin
129.     if(( clk_cnt >= 100_000  ) && ( clk_cnt < 200_000 ))
130.         sda_temp <= 0;
131.     else
132.         sda_temp <= 1;
133. end
134.
135. assign I_O_sda = out_en ? sda_temp : 1'hz;
136.
137. (*mark_debug="true"*)reg [1:0] sda_dly;
138.
139.
140. always@(posedge I_clk )begin
141.     if( !out_en )
142.         sda_dly <= {sda_dly , I_O_sda } ;
143. end
144.
145. (*mark_debug="true"*)reg [ 39 : 0 ] sda_reg ;
146.
147.
148. (*mark_debug="true"*)reg [31:0] sda_high;
149.
150. always@(posedge I_clk )begin
```

```
151.     if( sda_dly[1] )
152.         sda_high <= sda_high + 1;
153.     else
154.         sda_high <= 0;
155. end
156. 
157. always@(posedge I_clk )begin
158.     if( out_en_dly == 2'b01)
159.         sda_reg <= 40'h0;
160.     else
161.     if( ( sda_dly == 2'b10 ) && ( sda_high <= 3_000) )
162.         sda_reg <= {sda_reg , 1'h0 };
163.     else
164.     if( ( sda_dly == 2'b10 ) && ( sda_high > 5_000) )
165.         sda_reg <= {sda_reg , 1'h1 };
166. end
167. 
168. 
169. (*mark_debug="true"*)reg [ 15 :0] sda_dly_cnt ;
170. 
171. always@(posedge I_clk )begin
172.     if( out_en_dly == 2'b01)
173.         sda_dly_cnt <= 0;
174.     else
175.     if( sda_dly == 2'b10 )
176.         sda_dly_cnt <=  sda_dly_cnt + 1'h1;
177. end
178. 
179. (*mark_debug="true"*)reg [ 15 :0] temperature ;
180. (*mark_debug="true"*)reg [ 15 :0] humidity ;
181. 
182. (*mark_debug="true"*)reg [1:0]out_en_dly;
183. 
184. always@(posedge I_clk )begin
185.     out_en_dly <= { out_en_dly , out_en };
186. end
187. 
188. always@(posedge I_clk )begin
189.     if( out_en_dly == 2'b01) begin
190.         temperature <= sda_reg[23 -: 16];
191.         humidity <= sda_reg[39 -: 16];
192.     end
193. end
194. 
```

```
195.
196.
197. assign O_num1  = temperature/100 ;
198. assign O_num2  = temperature%100 ;
199. assign O_num3  = 0 ;
200. assign O_num4  = 0 ;
201. assign O_num5  = humidity/100  ;
202. assign O_num6  = humidity%100 ;
203. assign O_num7  = 0 ;
204. assign O_num8  = 0 ;
205.
206.
207. endmodule
208.
209. module sel_det_060(
210.     input             I_clk_div_060 ,
211.     input      [3:0]  I_num1_060    ,
212.     input      [3:0]  I_num2_060    ,
213.     input      [3:0]  I_num3_060    ,
214.     input      [3:0]  I_num4_060    ,
215.     input      [3:0]  I_num5_060    ,
216.     input      [3:0]  I_num6_060    ,
217.     input      [3:0]  I_num7_060    ,
218.     input      [3:0]  I_num8_060    ,
219.
220.     output     [7:0]  O_seg_060       ,
221.     output    [7:0]  O_sel_060
222.     );
223. reg [7:0] sel_060 = 8'b0000_0001  ;
224. reg [3:0] num_temp_060;
225.
226. always@(posedge I_clk_div_060 )
227.         sel_060 <= { sel_060[ 6 : 0 ] , sel_060[ 7 ] };
228.
229. assign O_sel_060 =  ~sel_060 ;
230.
231. always@(*)
232.     case( sel_060 )
233.         1<<0 :  num_temp_060 = I_num1_060;
234.         1<<1 :  num_temp_060 = I_num2_060;
235.         1<<2 :    num_temp_060 = I_num3_060;
236.         1<<3 :  num_temp_060 = I_num4_060;
237.         1<<4 :  num_temp_060 = I_num5_060;
238.         1<<5 :  num_temp_060 = I_num6_060;
```

```
        1<<6 :  num_temp_060 = I_num7_060;
        1<<7 :  num_temp_060 = I_num8_060;
    default :   num_temp_060 = num_temp_060;
    endcase

//wire [7:0]  seg_temp  ;

seg_num_060
seg_num
(
    .num_060(  num_temp_060 ) ,
    .seg_060(  O_seg_060 )
);


endmodule


module seg_num_060(
    input      [3:0]  num_060,
    output reg [7:0]  seg_060
    );

always@(*)
    case(num_060)
        0:   seg_060 = 8'hC0;
        1:   seg_060 = 8'hF9;
        2:   seg_060 = 8'hA4;
        3:   seg_060 = 8'hB0;
        4:   seg_060 = 8'h99;
        5:   seg_060 = 8'h92;
        6:   seg_060 = 8'h82;
        7:   seg_060 = 8'hF8;
        8:   seg_060 = 8'h80;
        9:   seg_060 = 8'h90;
        10:  seg_060 = 8'h88;
        11:  seg_060 = 8'h83;
        12:  seg_060 = 8'hC6;
        13:  seg_060 = 8'hA1;
        14:  seg_060 = 8'h86;
        15:  seg_060 = 8'h8E;
        default:seg_060 = 8'hFF;
    endcase

```

```
283. endmodule
```

5、 控速子系统

控诉子系统包含了一个分频模块、PWM 调整占空比。

直流电机的 PWM 调速原理与交流电机调速原理不同，它不是通过调频方式去调节电机的转速，而是通过调节驱动电压脉冲宽度的方式，并与电路中一些相应的储能元件配合，改变了输送到电枢电压的幅值，从而达到改变直流电机转速的目的。它的调制方式是调幅。

对于电机的转速调整，我们是采用脉宽调制 （PWM）办法，控制电机的时候，电源并非连续 地向电机供电，而是在一个特定的频率下以方波脉冲的形式提供电能。不同占空比的方波信号能对电机起到调速作用，这是因为电机实际上是一个大电感，它有阻碍输入电流和电压突变的能力，因此脉冲输入信号被平均分配到作用时间上，这样，改变在始能端 EN1 和 EN2 上输入方波的占空比就能改变加在电机两端的电压大小，从而改变了转速。

应当注意的是控速子系统有一个慢速下限。低于这个值的时候就无法正常的启动电机。在实验中这个值为$\frac{5}{16}$。

接口定义：

```
1.  input clk,
2.  input [3:0]pulse_width,
3.  output out
```

功能框图：

图 6 控速模块框图

代码：

```
module speed(
input key1,
input key2,
input key3,
output [3:0]mod
);

reg[3:0]speedmod;
assign mod=speedmod;

always@(*)
begin
    speedmod=4'd00;
    if(key1==1) speedmod=4'd10;
    if(key2==1) speedmod=4'd13;
    if(key3==1) speedmod=4'd15;
end
endmodule


module pwm(
input clk,
input [3:0]pulse_width,
output out
);

reg [3:0]cnt;
reg wave;
assign out = wave;
always @(posedge clk )
if(cnt<4'hF)
 cnt <= cnt + 1;
else
 cnt <= 0;

always @(posedge clk)
if(cnt<pulse_width)
 wave <= 1;
else
 wave <= 0;

endmodule
```

6、 电机驱动子系统

电机有两种运动的形式，前进和后退。本文将电机引出 4 个引脚，分别对应左右的前进高电压、后退高电压。当数据选择前进时，两个前进电压输出 pwm 波形；数据选择后退时，两个后退电压输出 pwm 波形；数据选择左转时，右轮前进电压输出 pwm 波形；数据选择右转时，左轮前进电压输出 pwm 波形。

接口定义：

```
1.   input StepEnable;
2.      input iS1;
3.      input iS2;
4.      input up;
5.      input down;
6.      input left;
7.      input right;
8.      input pwm;
9.      output reg IN1;
10.     output reg IN2;
11.     output reg IN3;
12.     output reg IN4;
```

功能框图：

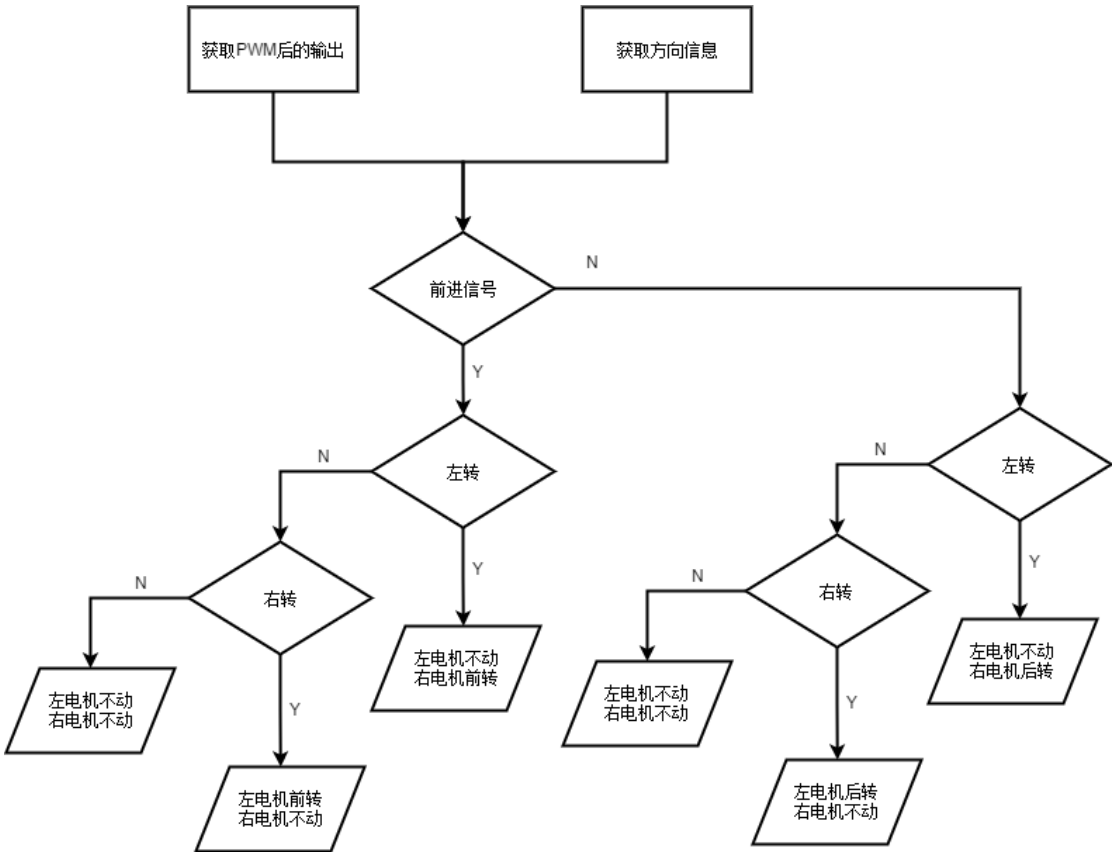


图 7 电机驱动模块框图

代码：

```
13. module sport(IN1,IN2,IN3,IN4,iS1,iS2,StepEnable,pwm,up,down,left,right);

14. // input clk;
15. // input hwx;
16.     input StepEnable;
17.     input iS1;
18.     input iS2;
19.     input up;
20.     input down;
21.     input left;
22.     input right;
23.     input pwm;
24.     output reg IN1;
25.     output reg IN2;
26.     output reg IN3;
27.     output reg IN4;
28.
29.
30.     always @(*)
31.     //always @(posedge clk or negedge hwx)
32.     //if (!hwx)
33.         //begin
34.          //IN1=0;
35.         //IN2=0;
36.          // IN3=0;
37.           // IN4=0;
38.       //  end
39.     //else
40.     begin
41.         IN1=0;
42.                     IN2=0;
43.                     IN3=0;
44.                     IN4=0;
45.         if(StepEnable==1)
46.         begin
47.         if(iS1==1&&iS2==1)
48.             begin
49.                 IN1=0;
50.                 IN2=pwm;
51.                 IN3=pwm;
52.                 IN4=0;
```

```
            end
            else if(iS1==0&&iS2==1)
            begin
                IN1=0;
                IN2=pwm;
                IN3=0;
                IN4=0;
            end
            else  if(iS1==1&&iS2==0)
            begin
                IN1=0;
                IN2=0;
                IN3=pwm;
                IN4=0;
            end
            else if(iS1==0&&iS2==0)
            begin
                IN1=0;
                IN2=0;
                IN3=0;
                IN4=0;
            end
        end
        else
        begin
            if(up==1&&down==0&&left==0&&right===0)
                    begin
                        IN1=0;
                        IN2=pwm;
                        IN3=pwm;
                        IN4=0;
                    end
            else   if(up==1&&down==0&&left==0&&right===1)
                     begin
                     IN1=0;
                      IN2=pwm;
                      IN3=0;
                      IN4=0;
                      end
             else   if(up==1&&down==0&&left==1&&right===0)
                                begin
```

```
                                IN1=0;
                                 IN2=0;
                                 IN3=pwm;
                                  IN4=0;
                                  end
            else   if(up==0&&down==1&&left==0&&right===0)
                                        begin
                                        IN1=pwm;
                                         IN2=0;
                                         IN3=0;
                                         IN4=pwm;
                                         end
             else   if(up==0&&down==1&&left==1&&right===0)
                                                                  begin
                                                                  IN1=0;
                                                                   IN2=0;
                                                                   IN3=0;
                                                                   IN4=pwm;

                                                                   end
    else   if(up==0&&down==1&&left==1&&right===0)
                              begin
                              IN1=pwm;
                                   IN2=0;
                                       IN3=0;
                                          IN4=0;

                                                          end

        end
    end
endmodule
```

五、测试模块建模

（要求列写各建模模块的 test bench 模块代码）

1、温度湿度模块 tb

```
2、 `timescale 1ns/1ns
3、 module  tb;
4、     reg                 I_clk      ;
5、     wire    [7:0]       O_seg_060  ;
6、     wire    [7:0]       O_sel_060  ;
```

```
7、
8、
9、 process_060
10、 process_060
11、 (
12、     .I_clk      (I_clk    )  ,
13、     .I_O_sda    (I_O_sda  )  ,
14、     .O_seg_060  (O_seg_060)  ,
15、     .O_sel_060  (O_sel_060)
16、 );
17、
18、
19、 initial
20、     begin
21、         I_clk = 0;
22、
23、     #100000 $stop;
24、     end
25、
26、 always #10  I_clk = ~I_clk;
27、
28、 initial
29、     begin
30、         I_clk = 0;
31、
32、     #100000 $stop;
33、     end
34、
35、 endmodule
```

2、 电机驱动模块 tb

```
1.  module carMove_tb;
2.  reg key1;
3.  reg key2;
4.  reg key3;
5.  reg CLK;
6.  reg iS1;
7.  reg iS2;
8.  reg StepEnable;
9.  wire IN1;
10. wire IN2;
11. wire IN3;
12. wire IN4;
13.
```

```
carMove uut(
.key1(key1),
.key2(key2),
.key3(key3),
.CLK(CLK),
.iS1(iS1),
.iS2(iS2),
.StepEnable(StepEnable),
.IN1(IN1),
.IN2(IN2),
.IN3(IN3),
.IN4(IN4)
);

initial  CLK=0;
always #5 CLK=~CLK;

initial
begin
    key1=0;
    key2=0;
    key3=0;

    iS1=0;
    iS2=0;
    StepEnable=1;


    #200   key3=1;
    #200   iS1=1;
    #200   iS2=1;

end


endmodule
```

六、实验结果

（该部分可截图说明，可包含 logisim 逻辑验证图、modelsim 仿真波形图、以及下板后的实验结果贴图）

温度湿度模块 debug 结果

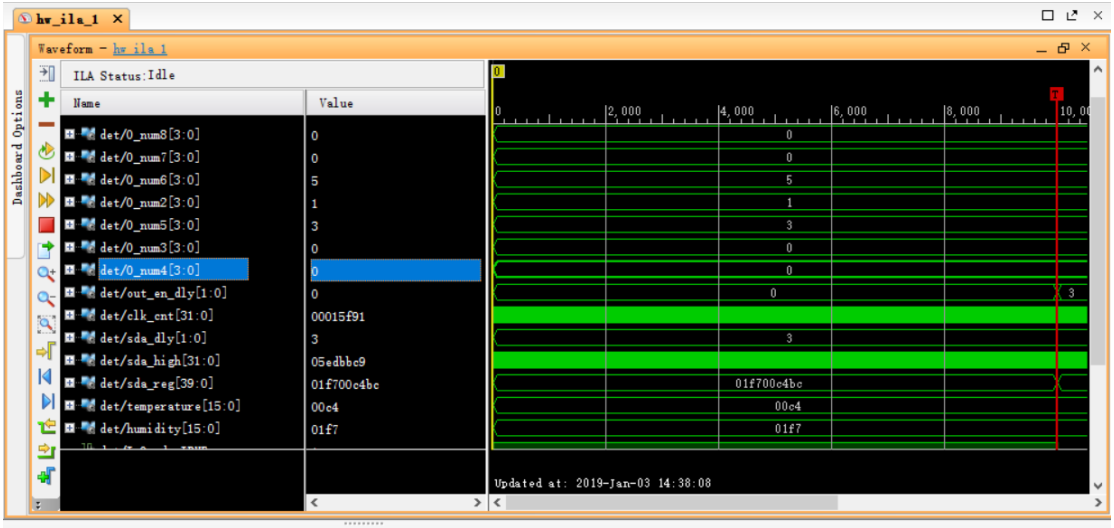


图 8 温度湿度 debug 图像

温度湿度模块下板结果

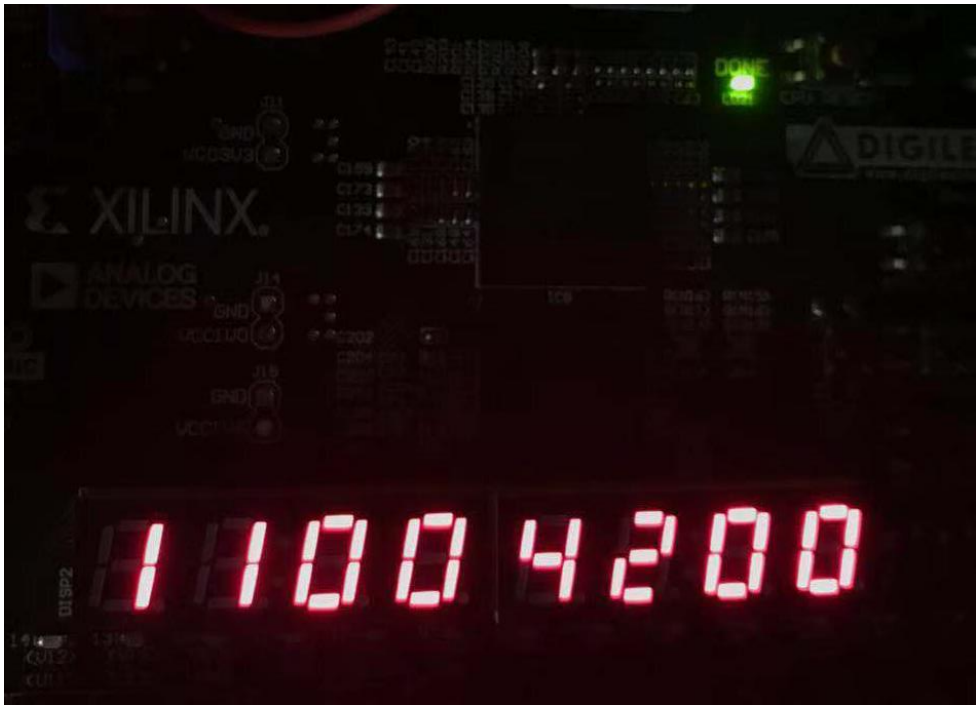
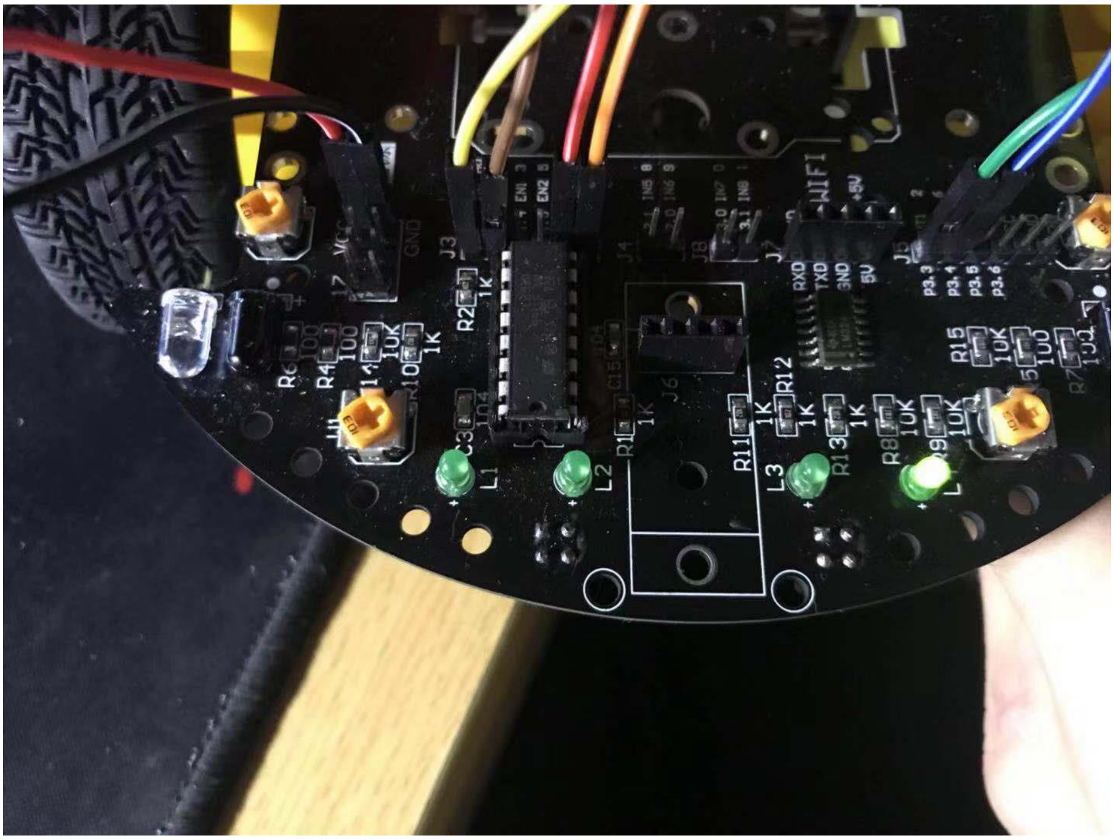
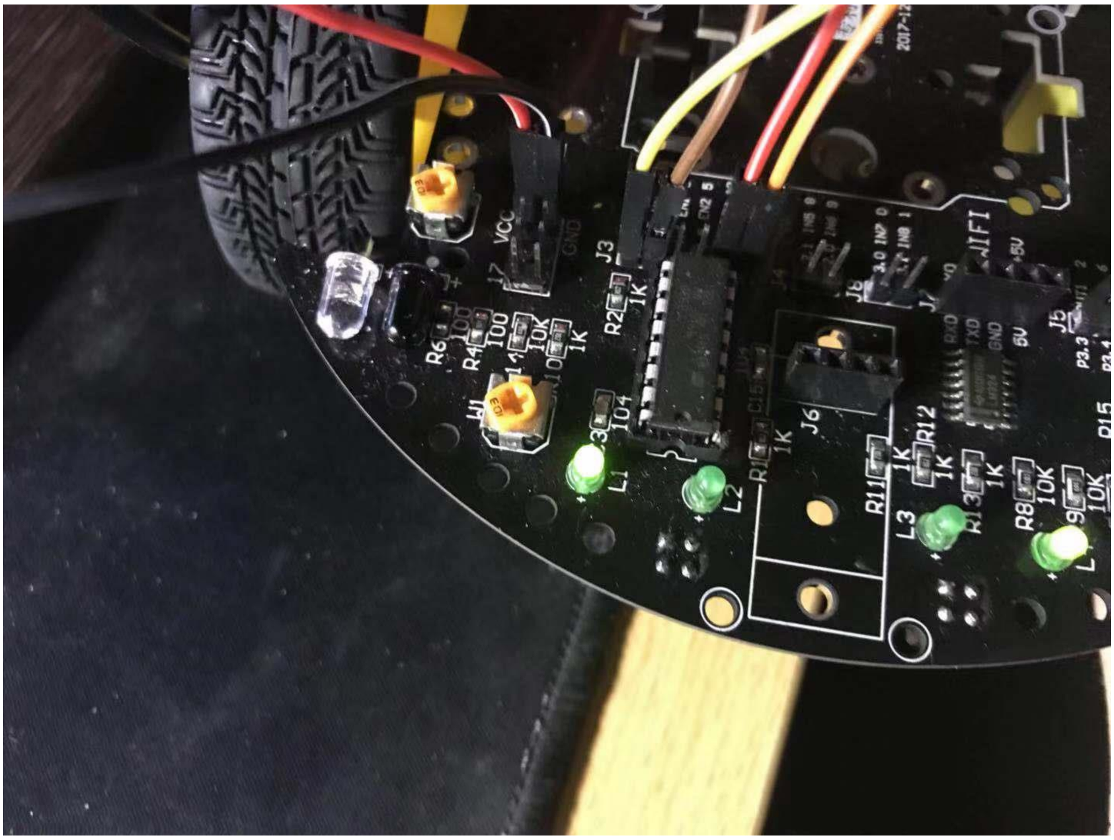


图 9 温度湿度实时监测图像

当前室内温度 11 摄氏度、湿度 42%。

红外反射模块下板结果：


WIFI
RXD
TXD
GND
5V
VCC
J3
J6
L1
L2
L3


WIFI
RXD
TXD
GND
5V
VCC
J3
J6
L1
L2
L3

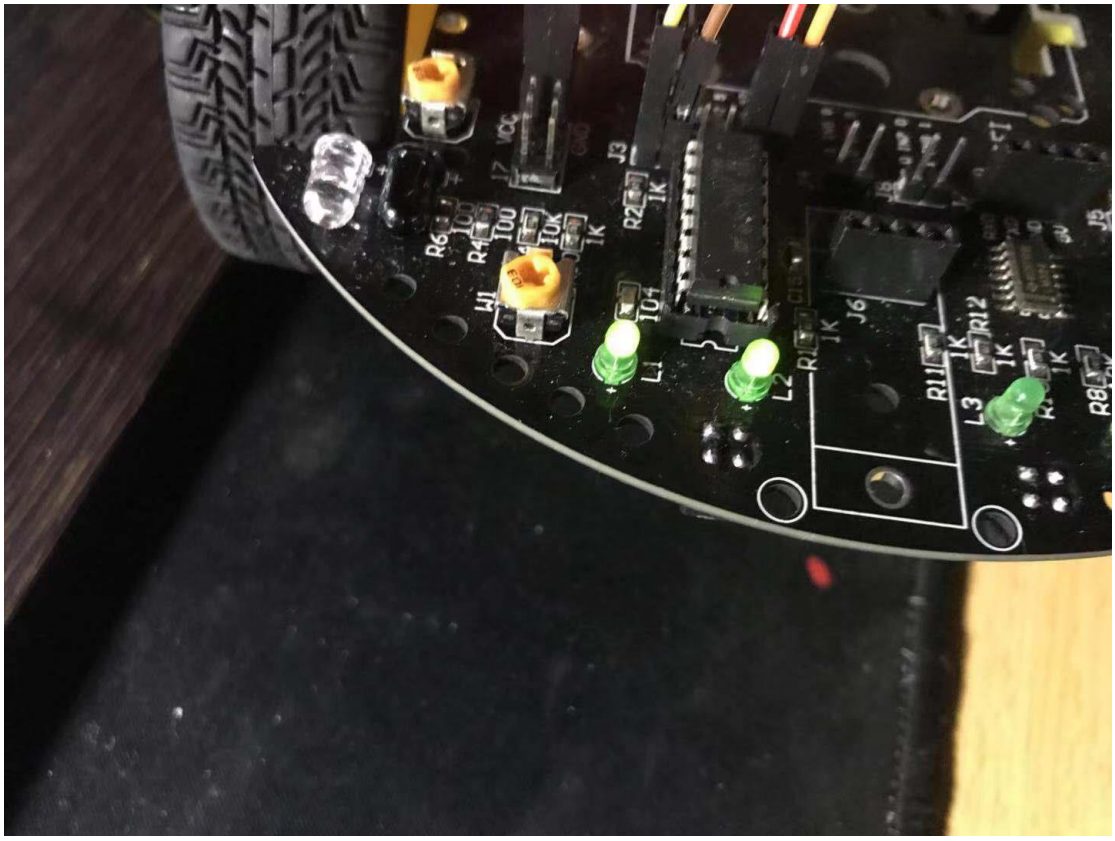


图 10-12 红外反射模块实时图像

如图，底部有黑色区域时灯全亮、无黑色区域时灯灭。
具体演示请见附录视频链接。

七、遇到的问题及解决

1、 温度湿度模块无法采集到数据

一开始设定了时钟频率后仍然无法正确的采集到数据，判断可能是 DHT22 本身输入的 0，1 值并不是简单的高电平低电平。通过阅读说明书后得知，0/1 的判断源于高低电平不同的持续时间。

本文采用的方法是，不论两种信号的低电平持续多长时间，只判断高电平。首先将时钟分为$10\mu s$，0 的高电平持续时间为 $26\mu s$，1 的高电平持续时间为 $72\mu s$。判断如果高电平持续时间超过了 3 个时钟，则为 1，反之为 0。

2、 控速模块有下限

控速模块通过调整输出波形的占空比来达到控速的效果。本文使用 6v 的电源，3.3v 的电机输出。在实际操作中，当占空比小于 5/16 的时候就无法正常输出工作波形。判断是电压过小，无法驱动电机。

3、 数码管显示不正确

本文在一开始进行湿度温度显示的时候总是乱码。后来判断为程序逻辑错误，

Nexys4 DDR 的数码管为阴极有效，本文一开始的逻辑为阳极有效。

八、未完成的蓝牙模块

本文最初设计中是以蓝牙模块作为多个模块之间的整合，但由于不可知的原因无法在规定时间内完成调试（代码可在其他机器上运行，但无法在自己的电脑上运行）。Verilog 代码已通过仿真检测，在下板时无法正确采集到数据。

接口定义：

```
1.      input   rx;
2.      input   fpga_clk;
3.      input   rst_n;
4.      output  tx;
```

代码：

```
5.  module uarttop(rx,fpga_clk,rst_n,tx);
6.      input   rx;
7.      input   fpga_clk;
8.      input   rst_n;
9.      output  tx;
10.
11.     wire [7:0] data;
12.     wire       rdsig;
13.     wire       clk;
14.
15.
16.     uartclk uartclk_inst
17.     (
18.         .fpga_clk(fpga_clk),
19.         .rst_n(rst_n),
20.         .clk(clk)
21.     );
22.
23.     uartrx uartrx_inst
24.     (
25.         .rx(rx),
26.         .clk(clk),
27.         .rst_n(rst_n),
28.         .dataout(data),
29.         .rdsig(rdsig)
30.     );
31.
```

```
32.
33.     uarttx uarttx_inst
34.     (
35.         .clk(clk),
36.         .rdsig(rdsig),
37.         .datain(dataout),
38.         .tx(tx)
39.     );
40.
41. endmodule
42.
43.
44. //分频
45. module uartclk(fpga_clk,rst_n,clk);
46.     input   fpga_clk;
47.     input   rst_n;
48.     output  clk;
49.
50.     reg [15:0]  cnt;
51.     reg         clk;
52.
53.     always @(posedge fpga_clk or negedge rst_n)begin
54.             if(!rst_n)begin
55.                 cnt <= 0;
56.                 clk <= 0;
57.             end
58.             else if(cnt == 16'd325)begin
59.                 clk <= 1;
60.                 cnt <= cnt + 1;
61.             end
62.             else if(cnt == 16'd650)begin
63.                 clk <= 0;
64.                 cnt <= 0;
65.             end
66.             else begin
67.                 cnt <= cnt + 1;
68.             end
69.
70.     end
71.
72. endmodule
73.
74.
75. //接受
```

```
module uartrx(rx,clk,rst_n,dataout,rdsig,receive,rxfall,idle,cnt);
    input   rx;
    input   clk;
    input   rst_n;
    output  dataout;
    output  rdsig;
    output  receive;
    output  rxfall;
    output  idle;
    output  cnt;

    reg [7:0]   dataout;
    reg         rdsig;
    reg         rxbuf;
    reg         rxfall;
    reg [7:0]   cnt;
    reg         receive;
    reg         idle;

    wire [7:0]  dataout_wire;
    assign dataout_wire = 32;
    assign dataout_wire = dataout;

   always @(posedge clk) begin
        rxbuf <= rx;
        rxfall <= rxbuf & (~rx);
    end
always @(posedge clk) begin
        if(rxfall == 1 && idle == 1) begin
            receive <= 1;
        end
        else if(cnt == 152) begin
            receive <= 0;
        end
    end
    always @(posedge clk) begin
            if(receive ==1) begin
            case (cnt)
                8'd24: begin
                    dataout[0] <= rx;
                    idle <= 0;
                    cnt <= cnt + 1;
                    rdsig <= 0;
                end
```

```
                8'd40: begin
                    dataout[1] <= rx;
                    idle <= 0;
                    cnt <= cnt + 1;
                    rdsig <= 0;
                end
                8'd56: begin
                    dataout[2] <= rx;
                    idle <= 0;
                    cnt <= cnt + 1;
                    rdsig <= 0;
                end
                8'd72: begin
                    dataout[3] <= rx;
                    idle <= 0;
                    cnt <= cnt + 1;
                    rdsig <= 0;
                end
                8'd88: begin
                    dataout[4] <= rx;
                    idle <= 0;
                    cnt <= cnt + 1;
                    rdsig <= 0;
                end
                8'd104: begin
                    dataout[5] <= rx;
                    idle <= 0;
                    cnt <= cnt + 1;
                    rdsig <= 0;
                end
                8'd120: begin
                    dataout[6] <= rx;
                    idle <= 0;
                    cnt <= cnt + 1;
                    rdsig <= 0;
                end
                8'd136: begin
                    dataout[7] <= rx;
                    idle <= 0;
                    cnt <= cnt + 1;
                    rdsig <= 0;
                end
                8'd152: begin
                    idle <= 1;
```

```
                    cnt <= 0;
                    rdsig <= 1;
                end
                default: begin
                    cnt <= cnt + 8'd1;
                end
            endcase
        end
        else begin
            idle <= 1;
            cnt <= 0;
            rdsig <= 0;
        end
      end


endmodule


//发出
module uarttx(clk,rdsig,datain,tx);
    input            clk;
    input            rdsig;
    input    [7:0]   datain;
    output           tx;

    reg         tx;
    reg         send;
    reg         idle;
    reg [7:0]   cnt;

    always @(posedge clk) begin
        if(rdsig == 1 && idle == 1) begin
            send <= 1;
        end
        else if(cnt == 152) begin
            send <= 0;
        end
    end

    always @(posedge clk) begin
            if(send ==1) begin
            case (cnt)
                8'd8: begin
```

```
208.                    tx <= 1'b0;
209.                    idle <= 0;
210.                    cnt <= cnt + 1;
211.                end
212.                8'd24: begin
213.                    tx <= datain[0];
214.                    idle <= 0;
215.                    cnt <= cnt + 1;
216.                end
217.                8'd40: begin
218.                    tx <= datain[1];
219.                    idle <= 0;
220.                    cnt <= cnt + 1;
221.                end
222.                8'd56: begin
223.                    tx <= datain[2];
224.                    idle <= 0;
225.                    cnt <= cnt + 1;
226.                end
227.                8'd72: begin
228.                    tx <= datain[3];
229.                    idle <= 0;
230.                    cnt <= cnt + 1;
231.                end
232.                8'd88: begin
233.                    tx <= datain[4];
234.                    idle <= 0;
235.                    cnt <= cnt + 1;
236.                end
237.                8'd104: begin
238.                    tx <= datain[5];
239.                    idle <= 0;
240.                    cnt <= cnt + 1;
241.                end
242.                8'd120: begin
243.                    tx <= datain[6];
244.                    idle <= 0;
245.                    cnt <= cnt + 1;
246.                end
247.                8'd136: begin
248.                    tx <= datain[7];
249.                    idle <= 0;
250.                    cnt <= cnt + 1;
251.                end
```

```
                8'd152: begin
                    idle <= 1;
                    cnt <= 0;
                end
                default: begin
                    cnt <= cnt + 8'd1;
                end
            endcase
        end
        else begin
            idle <= 1;
            cnt <= 0;
            tx <= 8'd1;
        end
    end


endmodule
```

九、心得体会及建议

1、 仔细阅读模块的说明书可以带来极大的便利，很多说明书中就已经将模块工作的逻辑和ASM 描述了出来；
2、 设计思路最好自顶向下，本文初期采用的模式是自下而上，在后期综合的时候产生了极大的不便；
3、 不同的部件在高低电平有效的问题上总是有不统一，在进行逻辑的设计时一定要注意；
4、 由于是在课程作业中第一次进行有外设的设计，开始需要自己克服一些连接上的问题。建议以后的课程中可以增加一个外设的示例教程。

附录 1

基于 FPGA 的检测温湿度红外循迹可变速小车

使用说明

设备：

1、 硬件：L298N 电机，HL-1 型小车底板，Nexys4 DDR 开发板

2、 传感器：温度湿度模块 DHT22，红外反射传感器 TCRT5000。

功能：

1、 实时监测当前环境温度湿度，并将温度和湿度以整数形式显示在数码管上；

2、 可实现两种运动模式：手动控制运动和红外循迹自动运动：

a) 手动模式下，使用 Nexys4 DDR 上方向键控制小车运动；

b) 自动模式下，小车根据黑线（其他颜色也可，但应为深色）自动循迹移动；

3、 可手动控制小车速度，共有三种速度可选

按键说明：

模式选择

V10 开启自动循迹功能

V11 开启手动控制功能

速度选择

L16 低速运动

M13 中速运动

R15 高速运动

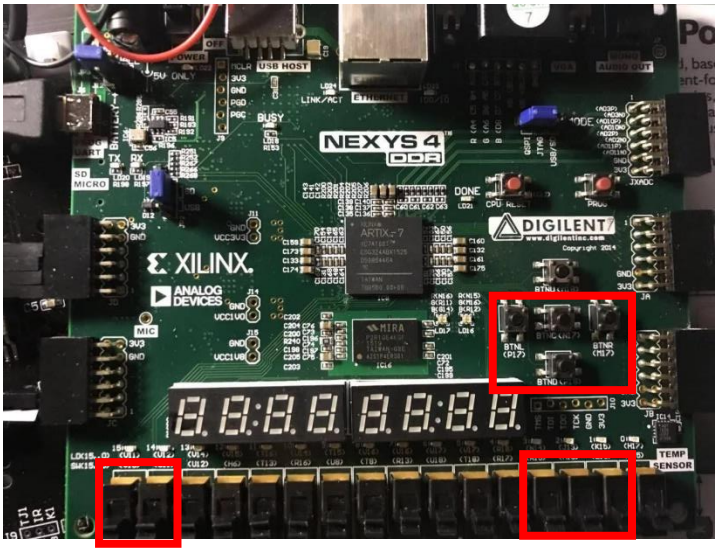
方向控制

N17 前进

P17 左转

M17 右转

P18 后退



注意事项：

1、模式选择 和 速度选择各自必须至少打开一个才可运动；

2、手动选择和自动选择同时开启时，默认为自动；

3、三种速度同时开启时，默认为高速；

4、驱动轮只有两个，后配一个万向轮。

演示视频 demo：https://pan.baidu.com/s/1b7P7sLbTkxkvhfnuURekRw