**同济大学计算机系**

**数字逻辑课程综合实验报告**

****

**学 号 2151130**

**姓 名 童海博**

**专 业 计算机科学与技术**

**授课老师 张冬冬**

目录

**一、实验内容**..............................................................................................................................................3

1.项目内容介绍.......................................................................................................................................3

2.设备介绍................................................................................................................................................3

3.界面说明................................................................................................................................................3

4.布线说明................................................................................................................................................4

5.操作说明................................................................................................................................................5

**二、项目总视图**..........................................................................................................................................6

**三、子系统模块建模**.................................................................................................................................9

1.顶层模块................................................................................................................................................9

2.蓝牙模块................................................................................................................................................11

3.七段译码管模块..................................................................................................................................13

4.寄存器初始化模块..............................................................................................................................14

5.SCCB模块..............................................................................................................................................22

6.摄像头初始化模块..............................................................................................................................24

7.摄像头模块............................................................................................................................................25

8.图像处理模块........................................................................................................................................26

9.VGA控制模块.......................................................................................................................................27

10.VGA模块..............................................................................................................................................28

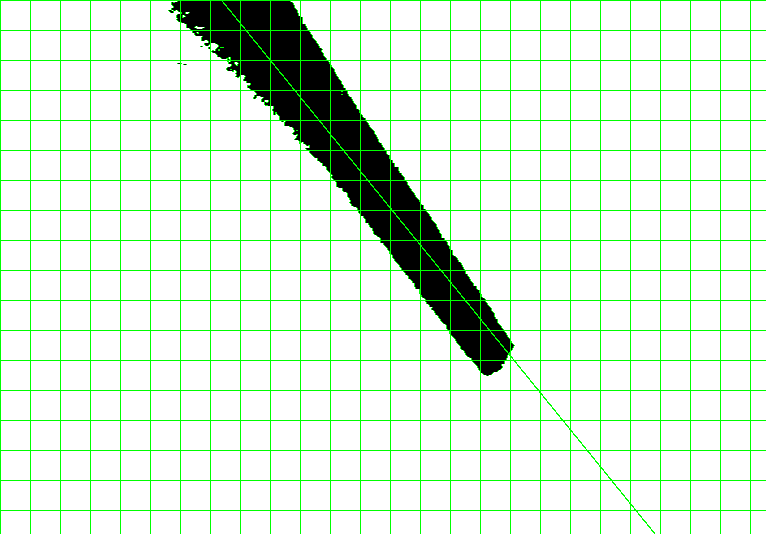
**四、版本迭代与算法设计**.......................................................................................................................33

**五、结果与总结**.........................................................................................................................................41

1. 实验内容
2. 项目内容介绍

本项目使用VGA显示器模块、蓝牙模块、摄像头模块和NEXYS4开发板实现线性拟合效果。本项目刚开始打算通过摄像头、蓝牙、vga、开发板模仿触摸屏效果，但由于刷新率较低、计算能力不足、储存量不足、环境噪声大等问题，导致触摸屏无法实现，故实现原本触摸屏中的一部分，也就是直线拟合的部分。

1. 设备介绍
2. NEXYS 4 DDR Atrix-7开发板：由Xilinx公司开发出的一款现场可编程门阵列(FPGA)开发板
3. 显示器：使用 VGA 接口，分辨率为 1024\*768，对应的时钟频率为 24Mhz
4. 摄像头：摄像头使用rgb565视频模式，对应时钟频率未25Mhz
5. 界面说明

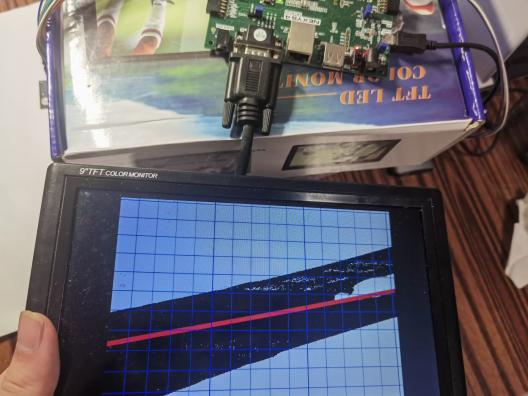


VGA上显示坐标网格和拟合的直线，开发板上七段数码管显示选择的模式，手机上在蓝牙调试器上编辑按键，可以更加方便的通过手机按键选择模式



1. 布线说明

VGA接线如下图，插座接电源



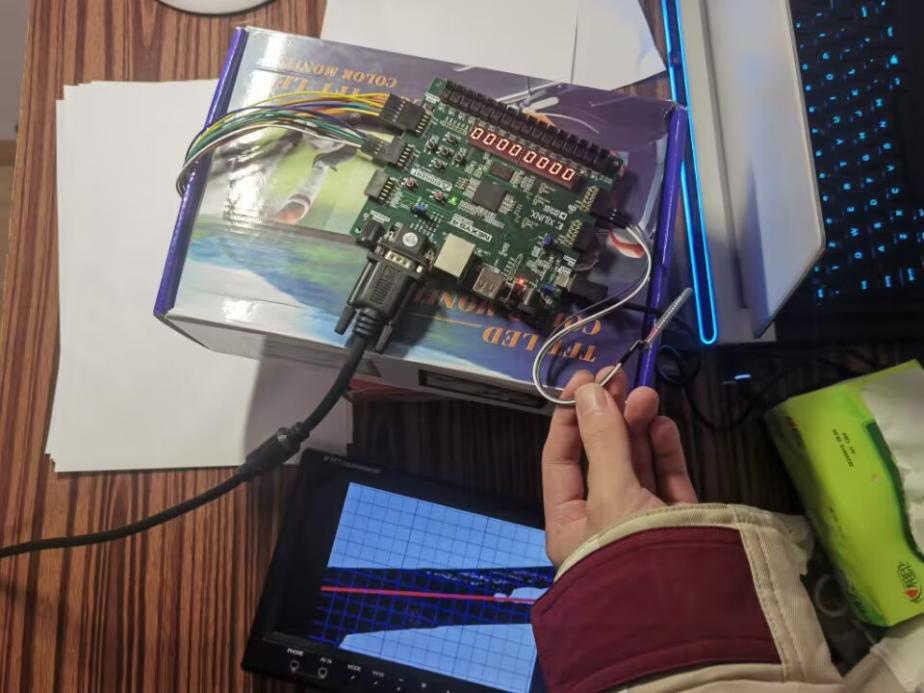
摄像头接线如下图，使用电工胶带固定在盒子上，方便拍摄稳定画面。



蓝牙接线如下图



整体效果如下图

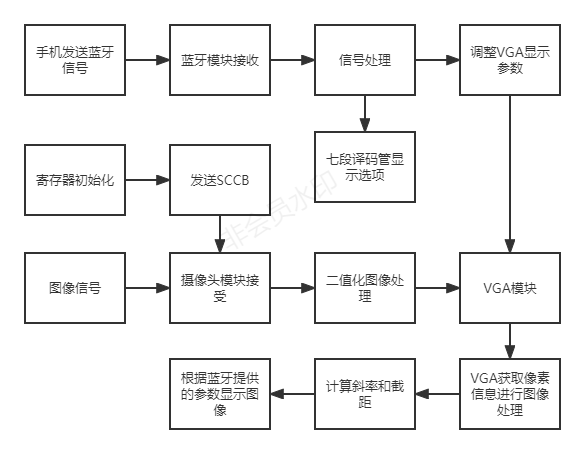


1. 操作说明

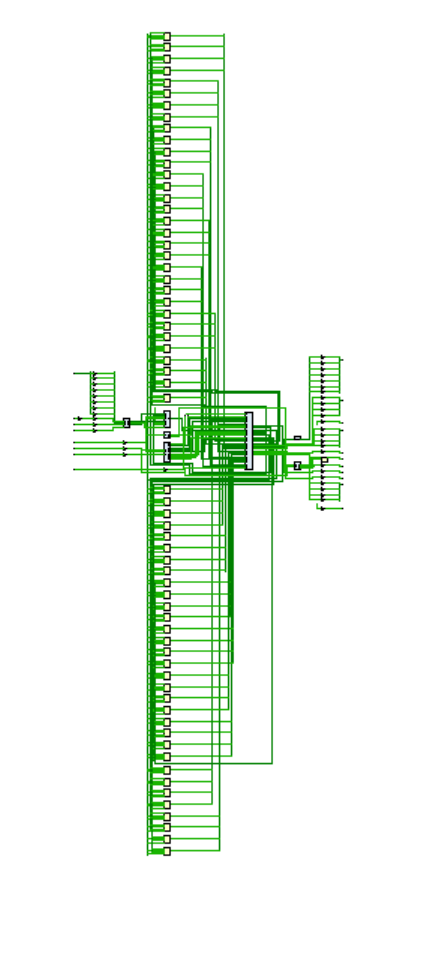
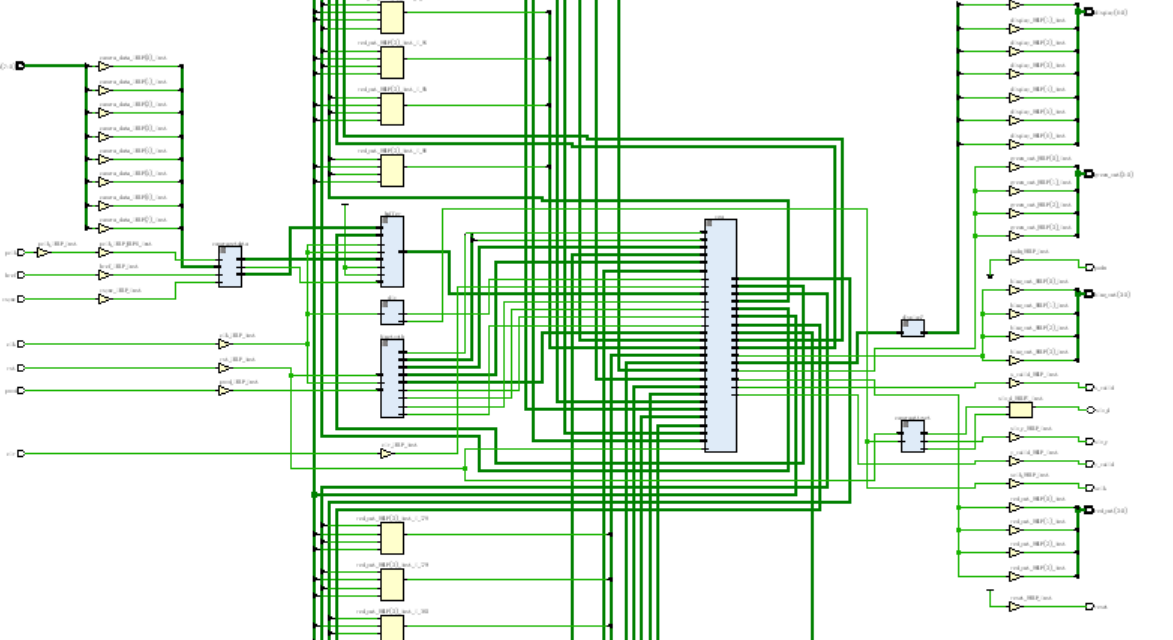
Program device之后大概vga显示器，将摄像头下方铺设白纸，并将颜色较深的物体放置于白纸上，vga上将会显示出物体的二值化图像，并拟合出直线。可以通过手机上的蓝牙调试器发送蓝牙型号并更改VGA显示器上的网格颜色和网格判定间距。通过开发板上的按键初始化或者重启系统。

1. 项目总视图

设计框图如下：

****

RTL分析：



各个模块说明：

1. 时钟分频模块：vivado自带的IP核，产生24Mhz和25Mhz两个时钟频率，分别用于寄存器初始化和vga时钟
2. RAM模块：vivado自带的IP核，用于储存摄像头传输的数据，方便VGA显示
3. 蓝牙模块(Bluetooth 和 UART)：通过实现 UART 协议，从手机端获取控制信号，获取的蓝牙和板载按键共同进行播放器的操作
4. 七段译码管模块：转化蓝牙信号传输的选项编号，并显示在开发板上的七段译码管上
5. 寄存器初始化模块：初始化寄存器，为SCCB提供信号，用于初始化摄像头
6. SCCB模块：实现SCCB协议，通过sio\_d核sio\_c控制信号发送，用于初始化摄像头
7. 摄像头初始化模块：使用摄像头之前，通过寄存器初始化和sccb协议传输信号设置摄像头的各项参数以及模式
8. 摄像头模块：获取像素信息，并将对应的信息传输到RAM模块，用于VGA显示
9. 图像处理模块：主要对摄像头模块提供的像素信息进行二值化处理，方便计算斜率等信息
10. VGA控制模块：在时钟控制下逐个扫描所有像素，将坐标信号传输给VGA用于确认像素显示位置，同时传输行列有效信号，用于启动显示
11. VGA模块：接受二值化处理后的图像信息并进行再处理，计算斜率截距。在显示原有二值化图像的同时，显示拟合的直线，并根据蓝牙信号显示对应的网格同时改变运算方法
12. 子系统模块建模
13. 顶层模块：

module LinearFitting(

//摄像头对外接口

output sio\_c,//摄像头sio\_c信号

inout sio\_d,//摄像头sio\_d信号

output reset,//reset信号，需要拉高，否则会重置寄存器

output pwdn,//pwdn信号，拉低，关闭耗电模式

output xclk,//xclk信号，可不接

input pclk,href,vsync,//用于控制图像数据传输的三组信号

input [7:0] camera\_data ,//图像数据信号

//VGA对外接口

output [3:0] red\_out,green\_out,blue\_out,//rgb像素信息

output x\_valid,//行时序信号

output y\_valid,//场时序信号

input rst,//复位

input clr,//vga初始化

//时钟

input clk,//接板内时钟，100mhz

//蓝牙端口

input pmod,//接pmod

//七位译码管

output [6:0]display//输出

);

//时钟

wire clk\_vga ;//vga时钟 24mhz

wire clk\_init\_reg;//初始化寄存器的时钟，25mhz

clk\_wiz\_0 div(.clk\_in1(clk),.clk\_out1(clk\_vga),.clk\_out2(clk\_init\_reg));

//蓝牙

wire [7:0]out\_bluetooth;//蓝牙数据输出

Bluetooth bluetooth(.clk(clk),.rst(rst),.get(pmod),.out(out\_bluetooth));

//摄像机初始化

wire [11:0] ramdata;//写数据

wire wr\_en;//缓存写有效

wire [18:0] ramaddr;//写地址

CameraGetData cameragetdata(.rst(rst),.pclk(pclk),.href(href),.vsync(vsync),.data\_in(camera\_data),.data\_out(ramdata),.wr\_en(wr\_en),.out\_addr(ramaddr));

CameraAttrSet cameraattrset(.clk(clk\_init\_reg),.sio\_c(sio\_c),.sio\_d(sio\_d),.reset(reset),.pwdn(pwdn),.rst(rst),.xclk(xclk));

//数据处理和显示

wire [11:0] rddata;//读数据

wire [18:0] rdaddr;//读地址

wire [11:0] binary\_data;//二值化结果

blk\_mem\_gen\_0 buffer(.clka(clk),.ena(1),.wea(wr\_en),.addra(ramaddr),.dina(ramdata),.clkb(clk),.enb(1),.addrb(rdaddr),.doutb(rddata));//数据储存

Binarization binarization(.clk(clk),.rgb\_data(rddata),.binary\_data(binary\_data));//二值化处理

wire [3:0] choice;//选项显示

VGA vga(.clk\_vga(clk\_vga),.rst(rst),.clr(clr),.color\_data\_in(binary\_data),.ram\_addr(rdaddr),.x\_valid(x\_valid),.y\_valid(y\_valid),.red(red\_out),.green(green\_out),.blue(blue\_out),.bluetooth\_data(out\_bluetooth),.choice(choice));

Display7 display7(.iData(choice),.oData(display));

endmodule

1. 蓝牙模块：

module Bluetooth(

input clk,//时钟

input rst,//复位

input get,//接受

output reg [7:0] out//输出

);

parameter bps=10417;//分频信号

reg [14:0] first\_count;//计数器1

reg [3:0] second\_count;//计数器2

reg buffer[2:0];//除去滤波

wire buffer\_en;//检测到边沿

reg add\_en;//加法使能信号

always @ (posedge clk)

begin

if(rst)

begin

buffer[0]<=1;

buffer[1]<=1;

buffer[2]<=1;

end

else

begin

buffer[0]<=get;

buffer[1]<=buffer[0];

buffer[2]<=buffer[1];

end

end

assign buffer\_en=buffer[2]&~buffer[1];

always @ (posedge clk)

begin

if(rst)

begin

first\_count<=0;

end

else if(add\_en)

begin

if(first\_count==bps-1)

begin

first\_count<=0;

end

else

begin

first\_count<=first\_count+1;

end

end

end

always @ (posedge clk)

begin

if(rst)

begin

second\_count<=0;

end

else if(add\_en&&first\_count==bps-1)//如果每一位加

begin

if(second\_count==8)

begin

second\_count<=0;

end

else

begin

second\_count<=second\_count+1;

end

end

end

always @ (posedge clk)

begin

if(rst)

begin

add\_en<=0;

end

else if(buffer\_en)

begin

add\_en<=1;

end

else if(add\_en&&second\_count==8&&first\_count==bps-1)

begin

add\_en<=0;

end

end

always @ (posedge clk)

begin

if(rst)

begin

out<=0;

end

else if(add\_en&&first\_count==bps/2-1&&second\_count!=0)

begin

out[second\_count-1]<=get;

end

end

endmodule

1. 七段译码管模块

module Display7(

input [3:0] iData,

output reg [6:0] oData

);

always@(\*)begin

case(iData)

0:oData=7'b1000000;

1:oData=7'b1111001;

2:oData=7'b0100100;

3:oData=7'b0110000;

4:oData=7'b0011001;

5:oData=7'b0010010;

6:oData=7'b0000010;

7:oData=7'b1111000;

8:oData=7'b0000000;

9:oData=7'b0010000;

endcase

end

Endmodule

1. 寄存器初始化模块

//reg\_state标注寄存器初始化状态

//sccb\_state标注sccb状态

module RegInit(

input clk,

input rst,

output reg [15:0]data\_out,

output reg\_state,

input sccb\_state

);

//初始化寄存器，用于配置模式

//寄存器储量

parameter number\_of\_reg= 176;

reg [10:0] count=0;

always @ (posedge clk)

begin

if(rst)

count<=0;

else if(reg\_state&&sccb\_state)

count<=count+1;

end

assign reg\_state=count<=number\_of\_reg;

always @ (posedge clk)

case (count)

8'h00:

data\_out <= 16'hFF01;

8'h01 :

data\_out <= 16'h1280;

8'h02 :

data\_out <= 16'hFF00;

8'h03 :

data\_out <= 16'h2CFF;

8'h04 :

data\_out <= 16'h2EDF;

8'h05 :

data\_out <= 16'hFF01;

8'h06 :

data\_out <= 16'h3C32;

8'h07 :

data\_out <= 16'h1101;

8'h08 :

data\_out <= 16'h0902;

8'h09 :

data\_out <= 16'h0420;

8'h0A :

data\_out <= 16'h13E5;

8'h0B :

data\_out <= 16'h1448;

8'h0C :

data\_out <= 16'h2C0C;

8'h0D :

data\_out <= 16'h3378;

8'h0E :

data\_out <= 16'h3A33;

8'h0F :

data\_out <= 16'h3BFB;

8'h10 :

data\_out <= 16'h3E00;

8'h11 :

data\_out <= 16'h4311;

8'h12 :

data\_out <= 16'h1610;

8'h13 :

data\_out <= 16'h3992;

8'h14 :

data\_out <= 16'h35DA;

8'h15 :

data\_out <= 16'h221A;

8'h16 :

data\_out <= 16'h37C3;

8'h17 :

data\_out <= 16'h2300;

8'h18 :

data\_out <= 16'h34C0;

8'h19 :

data\_out <= 16'h361A;

8'h1A :

data\_out <= 16'h0688;

8'h1B :

data\_out <= 16'h07C0;

8'h1C :

data\_out <= 16'h0D87;

8'h1D :

data\_out <= 16'h0E41;

8'h1E :

data\_out <= 16'h4C00;

8'h1F :

data\_out <= 16'h4800;

8'h20 :

data\_out <= 16'h5B00;

8'h21 :

data\_out <= 16'h4203;

8'h22 :

data\_out <= 16'h4A81;

8'h23 :

data\_out <= 16'h2199;

8'h24 :

data\_out <= 16'h2440;

8'h25 :

data\_out <= 16'h2538;

8'h26 :

data\_out <= 16'h2682;

8'h27 :

data\_out <= 16'h5C00;

8'h28 :

data\_out <= 16'h6300;

8'h29 :

data\_out <= 16'h4600;

8'h2A :

data\_out <= 16'h0C3C;

8'h2B :

data\_out <= 16'h6170;

8'h2C :

data\_out <= 16'h6280;

8'h2D :

data\_out <= 16'h7C05;

8'h2E :

data\_out <= 16'h2080;

8'h2F :

data\_out <= 16'h2830;

8'h30 :

data\_out <= 16'h6C00;

8'h31 :

data\_out <= 16'h6D80;

8'h32 :

data\_out <= 16'h6E00;

8'h33 :

data\_out <= 16'h7002;

8'h34 :

data\_out <= 16'h7194;

8'h35 :

data\_out <= 16'h73C1;

8'h36 :

data\_out <= 16'h1240;

8'h37 :

data\_out <= 16'h1711;

8'h38 :

data\_out <= 16'h1839;

8'h39 :

data\_out <= 16'h1900;

8'h3A :

data\_out <= 16'h1A3C;

8'h3B :

data\_out <= 16'h3209;

8'h3C :

data\_out <= 16'h37C0;

8'h3D :

data\_out <= 16'h4FCA;

8'h3E :

data\_out <= 16'h50A8;

8'h3F :

data\_out <= 16'h5A23;

8'h40 :

data\_out <= 16'h6D00;

8'h41 :

data\_out <= 16'h3D38;

8'h42 :

data\_out <= 16'hFF00;

8'h43 :

data\_out <= 16'hE57F;

8'h44 :

data\_out <= 16'hF9C0;

8'h45 :

data\_out <= 16'h4124;

8'h46 :

data\_out <= 16'hE014;

8'h47 :

data\_out <= 16'h76FF;

8'h48 :

data\_out <= 16'h33A0;

8'h49 :

data\_out <= 16'h4220;

8'h4A :

data\_out <= 16'h4318;

8'h4B :

data\_out <= 16'h4C00;

8'h4C :

data\_out <= 16'h87D5;

8'h4D :

data\_out <= 16'h883F;

8'h4E :

data\_out <= 16'hD703;

8'h4F :

data\_out <= 16'hD910;

8'h50 :

data\_out <= 16'hD382;

8'h51 :

data\_out <= 16'hC808;

8'h52 :

data\_out <= 16'hC980;

8'h53 :

data\_out <= 16'h7C00;//////

8'h54 :

data\_out <= 16'h7D00;///////

8'h55 :

data\_out <= 16'h7C03;////////

8'h56 :

data\_out <= 16'h7D48;/////////

8'h57 :

data\_out <= 16'h7D48;/////////

8'h58 :

data\_out <= 16'h7C08;////////

8'h59 :

data\_out <= 16'h7D20;

8'h5A :

data\_out <= 16'h7D10;

8'h5B :

data\_out <= 16'h7D0E;

8'h5C :

data\_out <= 16'h9000;

8'h5D :

data\_out <= 16'h910E;

8'h5E :

data\_out <= 16'h911A;

8'h5F :

data\_out <= 16'h9131;

8'h60 :

data\_out <= 16'h915A;

8'h61 :

data\_out <= 16'h9169;

8'h62 :

data\_out <= 16'h9175;

8'h63 :

data\_out <= 16'h917E;

8'h64 :

data\_out <= 16'h9188;

8'h65 :

data\_out <= 16'h918F;

8'h66 :

data\_out <= 16'h9196;

8'h67 :

data\_out <= 16'h91A3;

8'h68 :

data\_out <= 16'h91AF;

8'h69 :

data\_out <= 16'h91C4;

8'h6A :

data\_out <= 16'h91D7;

8'h6B :

data\_out <= 16'h91E8;

8'h6C :

data\_out <= 16'h9120;

8'h6D :

data\_out <= 16'h9200;

8'h6E :

data\_out <= 16'h9306;

8'h6F :

data\_out <= 16'h93E3;

8'h70 :

data\_out <= 16'h9305;

8'h71 :

data\_out <= 16'h9305;

8'h72 :

data\_out <= 16'h9300;

8'h73 :

data\_out <= 16'h9304;

8'h74 :

data\_out <= 16'h9300;

8'h75 :

data\_out <= 16'h9300;

8'h76 :

data\_out <= 16'h9300;

8'h77 :

data\_out <= 16'h9300;

8'h78 :

data\_out <= 16'h9300;

8'h79 :

data\_out <= 16'h9300;

8'h7A :

data\_out <= 16'h9300;

8'h7B :

data\_out <= 16'h9600;

8'h7C :

data\_out <= 16'h9708;

8'h7D :

data\_out <= 16'h9719;

8'h7E :

data\_out <= 16'h9702;

8'h7F :

data\_out <= 16'h970C;

8'h80 :

data\_out <= 16'h9724;

8'h81 :

data\_out <= 16'h9730;

8'h82 :

data\_out <= 16'h9728;

8'h83 :

data\_out <= 16'h9726;

8'h84 :

data\_out <= 16'h9702;

8'h85 :

data\_out <= 16'h9798;

8'h86 :

data\_out <= 16'h9780;

8'h87 :

data\_out <= 16'h9700;

8'h88 :

data\_out <= 16'h9700;

8'h89 :

data\_out <= 16'hC3ED;

8'h8A :

data\_out <= 16'hA400;

8'h8B :

data\_out <= 16'hA800;///////

8'h8C :

data\_out <= 16'hC511;////////

8'h8D :

data\_out <= 16'hC651;/////////

8'h8E :

data\_out <= 16'hBF80;/////////

8'h8F :

data\_out <= 16'hC710;////////

8'h90 :

data\_out <= 16'hB666;

8'h91 :

data\_out <= 16'hB8A5;

8'h92 :

data\_out <= 16'hB764;

8'h93 :

data\_out <= 16'hB97C;

8'h94 :

data\_out <= 16'hB3AF;

8'h95 :

data\_out <= 16'hB497;

8'h96 :

data\_out <= 16'hB5FF;

8'h97 :

data\_out <= 16'hB0C5;

8'h98 :

data\_out <= 16'hB194;

8'h99 :

data\_out <= 16'hB20F;

8'h9A :

data\_out <= 16'hC45C;

8'h9B :

data\_out <= 16'hC050;

8'h9C :

data\_out <= 16'hC13C;

8'h9D :

data\_out <= 16'h8C00;

8'h9E :

data\_out <= 16'h863D;

8'h9F :

data\_out <= 16'h5000;

8'hA0 :

data\_out <= 16'h51A0;

8'hA1 :

data\_out <= 16'h5278;

8'hA2 :

data\_out <= 16'h5300;

8'hA3 :

data\_out <= 16'h5400;

8'hA4 :

data\_out <= 16'h5500;

8'hA5 :

data\_out <= 16'h5AA0;

8'hA6 :

data\_out <= 16'h5B78;

8'hA7 :

data\_out <= 16'h5C00;

8'hA8 :

data\_out <= 16'hD382;

8'hA9 :

data\_out <= 16'hC3ED;

8'hAA :

data\_out <= 16'h7F00;

8'hAB :

data\_out <= 16'hDA08;

8'hAC :

data\_out <= 16'hE51F;

8'hAD :

data\_out <= 16'hE167;

8'hAE :

data\_out <= 16'hE000;

8'hAF :

data\_out <= 16'hDD7F;

8'hB0 :

data\_out <= 16'h0500;

endcase

endmodule

1. SCCB模块

module Sccb(

input clk,//时钟信号，传入25MHz,内部自行分频

input rst,//复位信号，高电平有效

inout sio\_d,

output reg sio\_c,

input reg\_state,//说明reg\_init模块已经完成，可以发送下一个数据

output reg sccb\_state=0,//向外传输，告诉reg\_init sccb已经发送完毕

input [7:0]slave\_id,

input [7:0]reg\_addr,

input [7:0]value

);

reg [20:0]count=0;//计数器

always @ (posedge clk)

begin

if(count==0)

count<=reg\_state;//当reg\_ok有效时，开始计数，否则为0，一直等待

else

if(count[20:11]==31)//31个周期

count<=0;

else

count<=count+1;

end

always @ (posedge clk)

begin

sccb\_state<=(count==0)&&(reg\_state==1);

end

reg sio\_d\_send;//发送信号

always @ (posedge clk)

begin

if(count[20:11]==0)

sio\_c<=1;

else if(count[20:11]==1)

begin

if(count[10:9]==2'b11)//设置缓冲延时

sio\_c<=0;

else

sio\_c<=1;

end

else if(count[20:11]==29)

begin

if(count[10:9]==2'b00)

sio\_c<=0;

else

sio\_c<=1;

end

else if(count[20:11]==30||count[20:11]==31)

sio\_c<=1;

else//产生分布均匀的时钟信号

begin

if(count[10:9]==2'b00)

sio\_c<=0;

else if(count[10:9]==2'b01)

sio\_c<=1;

else if(count[10:9]==2'b01)

sio\_c<=1;

else if(count[10:9]==2'b11)

sio\_c<=0;

end

end

always @ (posedge clk)

begin

if(count[20:11]==10||count[20:11]==19||count[20:11]==28)

sio\_d\_send<=0;//此时为ack时间，don't care位

else

sio\_d\_send<=1;

end

reg [31:0] data\_temp;

always @ (posedge clk)

begin

if(rst)

data\_temp<=32'hffffffff;

else

begin

if(count==0&&reg\_state==1)

data\_temp<={2'b10,slave\_id,1'bx,reg\_addr,1'bx,value,1'bx,3'b011};

else if(count[10:0]==0)

data\_temp<={data\_temp[30:0],1'b1};//串行赋值法

end

End

assign sio\_d=sio\_d\_send?data\_temp[31]:'bz;

endmodule

1. 摄像头初始化模块

module CameraAttrSet(

input clk,

input rst,

output sio\_c,

inout sio\_d,

output reset,

output pwdn,

output xclk

);

//拉低pwdn,拉高reset，表示正常启动

assign reset=1;

assign pwdn=0;

//拉高sio\_d的电阻

pullup up (sio\_d);

//赋给xclk时钟信号

assign xclk=clk;

//reg\_state和sccb\_state控制寄存器和恶sccb状态

wire [15:0] data\_send;

wire reg\_state,sccb\_state;

//初始化寄存器

RegInit reginit(.clk(clk),.rst(rst),.data\_out(data\_send),.reg\_state(reg\_state),.sccb\_state(sccb\_state));

//sccb传输配置信息

Sccb sccb(.clk(clk),.rst(rst),.sio\_d(sio\_d),.sio\_c(sio\_c),.reg\_state(reg\_state),.sccb\_state(sccb\_state),.slave\_id(8'h60),.reg\_addr(data\_send[15:8]),.value(data\_send[7:0]));

Endmodule

1. 摄像头模块

module CameraGetData (

input rst,

input pclk,

input href,

input vsync,

input [7:0]data\_in,//摄像头D[9]到D[2]

output reg[11:0]data\_out,//一个像素点的数据

output reg wr\_en,//处理完毕，写有效使能

output reg[18:0]out\_addr=0//缓存位置

);

reg [15:0] rgb565 = 0;

reg [18:0] next\_addr = 0;

reg [1:0] status = 0;//每两组数据组合成一个rgb信息，利用一个两位的status不断地移位运算

always@ (posedge pclk)

begin

if(vsync == 0)//高电平有效

begin

out\_addr <=0;

next\_addr <= 0;

status=0;

end

else

begin

data\_out <= {rgb565[15:12],rgb565[10:7],rgb565[4:1]};

out\_addr <= next\_addr;

wr\_en <= status[1];

status <= {status[0], (href && !status[0])};//都是高电平有效

rgb565 <= {rgb565[7:0], data\_in};

if(status[1] == 1)

begin

next\_addr <= next\_addr+1;

end

end

end

Endmodule

1. 图像处理模块

module Binarization(

input clk,

input [11:0] rgb\_data,//杈撳叆rgb鏁版嵁

output reg[11:0] binary\_data//浜屽?煎寲缁撴灉

);

reg [7:0]red,green,blue,avg;

always@(\*)

begin

red={rgb\_data[11:8],4'b0000};

green={rgb\_data[7:4],4'b0000};

blue={rgb\_data[3:0],4'b0000};

avg=(red+green+blue)/3;

if(avg>=100)

begin

red=255;

green=255;

blue=255;

end

else

begin

red=0;

green=0;

blue=0;

end

binary\_data[11:8]=red[7:4];

binary\_data[7:4]=green[7:4];

binary\_data[3:0]=blue[7:4];

end

endmodule

1. VGA控制模块

//用于控制当前的位置信号，保证信号可以正确显示到显示屏的正确位置

module VGAPosControl(

input vga\_clk,//时钟周期

input rst,//清零信号，高电平有效

output reg signed[11:0] x\_poi,//输出此时x的坐标

output reg signed[11:0] y\_poi,//输出此时y的坐标

output is\_display,//表征此时是否能够输出

output x\_valid,//行有效信号`

output y\_valid//列有效信号

);

//行参数

parameter x\_sync=11'd96;

parameter x\_before=11'd144;

parameter x\_beside\_after=11'd784;

parameter x\_all=11'd799;

//列参数

parameter y\_sync=11'd2;

parameter y\_before=11'd35;

parameter y\_beside\_after=11'd515;

parameter y\_all=11'd524;

assign is\_display=((x\_poi>=x\_before)&&(x\_poi<x\_beside\_after)

&&(y\_poi>=y\_before)&&(y\_poi<y\_beside\_after))?1:0;

assign x\_valid=(x\_poi<x\_sync)?0:1;//行同步信号拉低时段

assign y\_valid=(y\_poi<y\_sync)?0:1;//场同步信号拉低时段

always @ (posedge vga\_clk)//判断此时是否可以进行绘制图像

begin

if(rst)//清零信号

begin

x\_poi<=0;

y\_poi<=0;

end

else

begin

if(x\_poi==x\_all)

begin

x\_poi<=0;

if(y\_poi==y\_all)

begin

y\_poi<=0;

end

else

begin

y\_poi<=y\_poi+1;

end

end

else

begin

x\_poi<=x\_poi+1;

end

end

end

endmodule

1. VGA模块

module VGA(

input clk\_vga,//输入vga的时钟，频率为25.175MHz

input rst,//复位信号，高电平有效

input clr,

input [11:0] color\_data\_in,//从RAM中读取的像素信息

output reg[18:0] ram\_addr,//应该读取的RAM的图片地址，由vga\_control给出

output x\_valid,

output y\_valid,

output reg[3:0] red,

output reg[3:0] blue,

output reg[3:0] green,

input [7:0] bluetooth\_data,

output reg[3:0] choice

);

parameter x\_before=11'd144;

parameter y\_before=11'd35;

parameter x\_size\_pic=11'd640;

parameter y\_size\_pic=11'd480;

parameter x\_all=11'd799;

parameter y\_all=11'd524;

integer dis=40;

wire signed [11:0] x\_poi;//输出此时x的坐标

wire signed [11:0] y\_poi;//输出此时y的坐标

wire is\_display;//表征此时是否能够输出

integer x\_sum;//x的和

integer y\_sum;//y的和

integer xx\_sum;//x^2的和

integer num;//总数

integer xy\_sum;//xy的和

integer k\_up;//斜率分子

integer k\_down;//斜率分母

integer b\_up;//截距分子

integer b\_down;//截距分母

integer x;//x

integer y;//y

reg [3:0] cr,cg,cb;//填充的颜色

VGAPosControl vgaposcontrol(clk\_vga,rst,x\_poi,y\_poi,is\_display,x\_valid,y\_valid);

always@ (bluetooth\_data)

begin

if(bluetooth\_data==0)

begin

cr=0;

cg=0;

cb=15;

dis=40;

end

else if(bluetooth\_data==1)

begin

cr=cr;

cg=cg;

cb=cb;

dis=30;

end

else if(bluetooth\_data==2)

begin

cr=cr;

cg=cg;

cb=cb;

dis=40;

end

else if(bluetooth\_data==3)

begin

cr=cr;

cg=cg;

cb=cb;

dis=50;

end

else if(bluetooth\_data==4)

begin

cr=cr;

cg=cg;

cb=cb;

dis=60;

end

else if(bluetooth\_data==5)

begin

cr=cr;

cg=cg;

cb=cb;

dis=70;

end

else if(bluetooth\_data==6)

begin

cr=15;

cg=0;

cb=0;

dis=dis;

end

else if(bluetooth\_data==7)

begin

cr=0;

cg=15;

cb=0;

dis=dis;

end

else if(bluetooth\_data==8)

begin

cr=0;

cg=0;

cb=15;

dis=dis;

end

else

begin

cr=cr;

cg=cg;

cb=cb;

dis=dis;

end

end

always@ (posedge clk\_vga)

begin

if(clr==1)

begin

x\_sum=0;

y\_sum=0;

xx\_sum=0;

xy\_sum=0;

num=0;

x=0;

y=0;

k\_up=1;

k\_down=1;

b\_up=0;

b\_down=0;

end

else

begin

x=x\_poi;

y=y\_poi;

if((x!=x\_all) || (y!=y\_all))

begin

if(color\_data\_in[3:0]==0 && x%dis==0 && y%dis==0)

begin

x\_sum=x\_sum+x/dis;

y\_sum=y\_sum+y/dis;

xx\_sum=xx\_sum+(x/dis)\*(x/dis);

xy\_sum=xy\_sum+(x/dis)\*(y/dis);

num=num+1;

end

end

else

begin

k\_up=xy\_sum\*num-x\_sum\*y\_sum;

k\_down=xx\_sum\*num-x\_sum\*x\_sum;

b\_up=(y\_sum\*xx\_sum-x\_sum\*xy\_sum)\*dis;

b\_down=xx\_sum\*num-x\_sum\*x\_sum;

choice=bluetooth\_data;

num=0;

x\_sum=0;

y\_sum=0;

xx\_sum=0;

xy\_sum=0;

end

end

end

always@ (\*)

begin

red=0;

blue=0;

green=0;

if(is\_display)

begin

if(x\_poi-x\_before<=x\_size\_pic&&y\_poi-y\_before<=y\_size\_pic)

begin

red=color\_data\_in[11:8];

green=color\_data\_in[7:4];

blue=color\_data\_in[3:0];

ram\_addr=(y\_poi-y\_before)\*x\_size\_pic+(x\_poi-x\_before);

x=x\_poi;

y=y\_poi;

if(k\_down>0)

begin

if(x\*k\_up+b\_up<=(y+5)\*k\_down && x\*k\_up+b\_up>=(y-5)\*k\_down)

begin

red=15;

green=0;

blue=0;

end

end

else

begin

if(x\*k\_up+b\_up<=(y-5)\*k\_down && x\*k\_up+b\_up>=(y+5)\*k\_down)

begin

red=15;

green=0;

blue=0;

end

end

if(x%dis==0 || y%dis==0)

begin

red=cr;

green=cg;

blue=cb;

end

end

else

begin

red=0;

green=0;

blue=0;

end

end

end

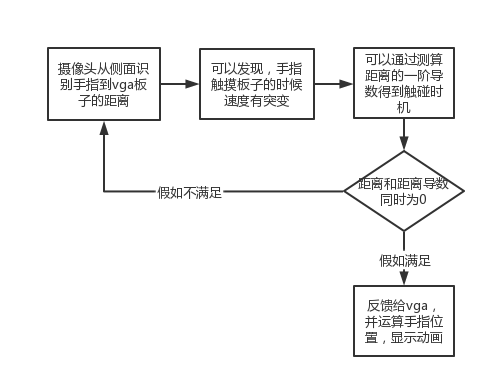
endmodule

1. 算法设计与实现

本项目较为困难的部分体现在直线拟合的算法设计上。

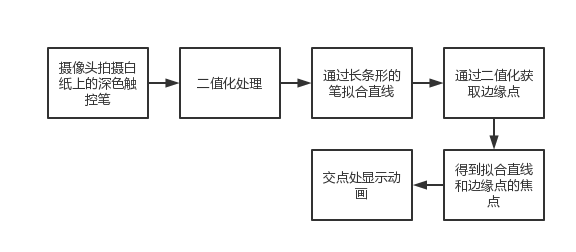
1. 外部设备的设计：

第一版实现设计（实现触摸屏）：



这一版设计的问题在于，外部干扰因素太多，且摄像头对着vga拍摄会出现大量噪点，导致无法识别手指位置，故最终舍弃。

第二版设计（实现触控笔笔头跟踪）：

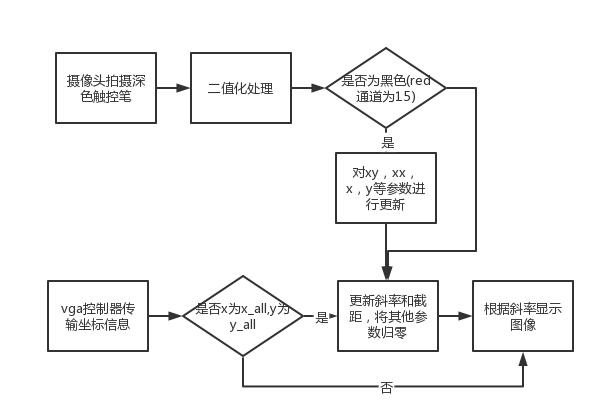


该设计的问题在于，就算是白纸和黑笔，二值化处理后的噪点任然太多，并且由于内存过小，无法对于完整的图形进行高斯降噪处理，对于边缘检测的影响过大，故最终舍弃。

第三版设计：

最终决定实现直线拟合的部分。最初设计时，打算获取屏幕上所有的像素点再一起计算，但发现二维数组存不下，所以考虑一便读取像素一边计算。

直线拟合的初版设计：

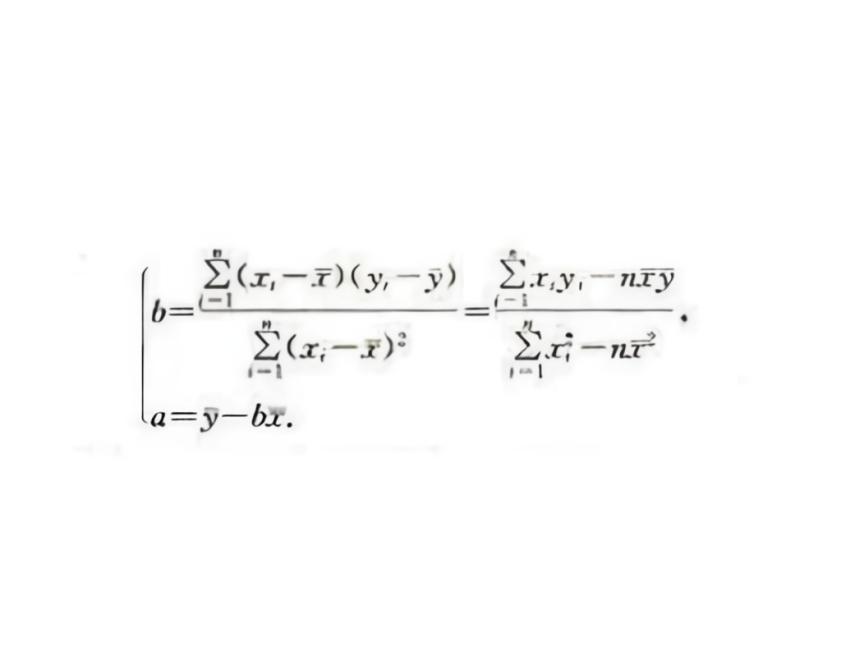


后序所有设计的实现思路和主要算法均没有变化。

算法设计：

对于直线拟合使用最小二乘法实现。最小二乘法公式为:

b为斜率，a为截距：



通过这个公式可以计算斜率。

设置以下变量：

x\_sum为x的和

y\_sum为y的和

xx\_sum为x^2的和

xy\_sum为xy的和

num为黑色点的数量

每次摄像头传输的像素经过二值化处理为0,0,0即为黑色点的时候更新这几个变量

x\_sum=x\_sum+x;

y\_sum=y\_sum+y;

xx\_sum=xx\_sum+x\*x;

xy\_sum=xy\_sum+x\*y;

num=num+1;

当vga\_control读取的x和y坐标信息刚好为x\_all,y\_all即扫描完完整的一遍的时候，计算k和b。这样就统计到了屏幕上的所有点

这样就可以计算出k和b

之后通过判定点是否在y=k\*x+b上，若在该直线上，则将点标记为红色，连起来就是一条直线。

以上算法实现会出现如下问题:

1. verilog没有可综合的浮点数，使用real无法进行综合，所以无法进行浮点数运算，使用整除会导致k和b的进度大幅下降，甚至会被噪声点完全覆盖
2. 上述参数在计算时的值过大，需要巨大的寄存器

这里以xx\_sum为例子

限制屏幕在[0-799][0-524]，取x的均值为400计算，全屏幕总共400000个点

xx\_sum可以达到400\*400\*400000=64,000,000,000，远远超过了int的范围，需要使用36位寄存器储存。

1. 综合考虑以上两点，需要改变算法来更好的提高精度，并减小变量大小。

首先考虑精度问题：

观察最小二乘法的公式，可以发现k和b均可以表示为分数。我们尽量避免除法运算，一切从乘法运算出发。

可以将k分解为两部部分：

这样k\_up/k\_down的小数运算就是精确的k值

同样，我们变换公式，将b表述为分数形式

可以化为很简洁的模式

表述为

测试点的公式为y=k\*x+b

转化为关于k\_up,k\_down,b\_up,b\_down,y,x的等式

得到

使用该公式判定点是否在直线上，可以得到最终结果

但是，注意到

左右均为小数，但是在vga上x和y只能是整数，需要用最靠经y，x值的整数代替

使用公式

将发现屏幕上没有产生直线

我们改变公式为

这样才应该是正确的直线，但是verilog无法进行小数运算，我们变换公式为

此处经过观察得：

和

是一样的

变化原有公式为

:

:

这样就可以显示出完整的直线，线的粗细将随着width的增大而增大。

其次是数值过大的问题：

观察到最终公式

出现的最大值为b\_up数量级

取x，y的均值为400和250

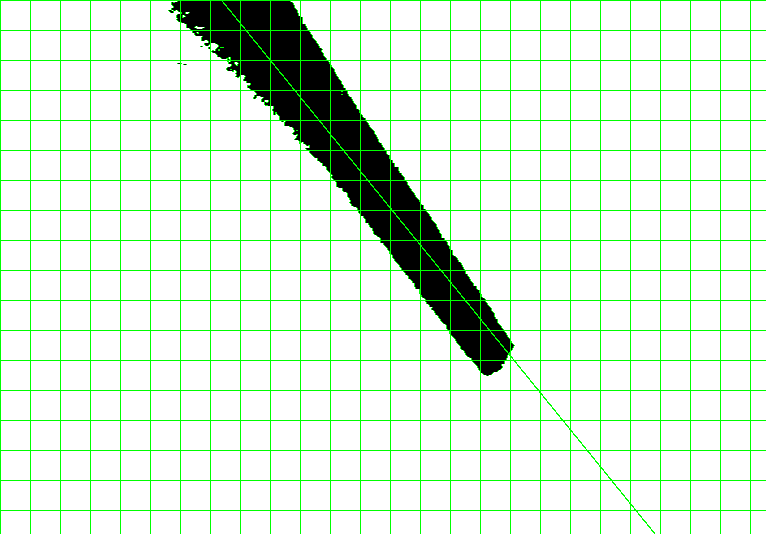
可以发现b\_up的最大值为：250\*400000\*400\*400\*400000=6.4e+18

该数值需要大小为60的寄存器能储存的下

发现使用大小为60的寄存器会无法综合，所以必须缩小数值，解决精度问题。

这里使用网格划分整个屏幕，计算时，并不取所有的点，仅仅取处于网格交点上的点。

如下图：



假设x方向和y方向网格间距均为dis

仅考察x%dis==0 && y%dis==0的点

观察原有k的公式：

可以发现两点：

1. k的分子分母上x，y因子的总数是相等的
2. 由于x，y只在x%dis==0 && y%dis==0处取，所以x/dis==int(x/dis)，y同理，对于dis做除法不会降低精度

这样可以将变量更新的公式变化为：

x\_sum=x\_sum+x/dis;

y\_sum=y\_sum+y/dis;

xx\_sum=xx\_sum+(x/dis)\*(x/dis);

xy\_sum=xy\_sum+(x/dis)\*(y/dis);

num=num+1;

观察b

可以发现b分子比分母多一个x因子，因此将b\_up的公式改为：

b\_down的公式不变

同样的，计算此时可能出现的最大值，取dis为40：

此时的num大致为：400000/40/40=250

y\_sum大致为：250\*250/40=1600

xx\_sum大致为：250\*400/40\*400/40=25000

b\_up的最大值为：1,600,000,000，这个值使用integer变量记录即可。

在python上实现代码后测试，发现拟合效果和没有处理时的效果几乎一样，该方案完全可行

Vivado上实现的代码为：

计算斜率部分：

x=x\_poi;

y=y\_poi;

if((x!=x\_all) || (y!=y\_all))

begin

if(color\_data\_in[3:0]==0 && x%dis==0 && y%dis==0)

begin

x\_sum=x\_sum+x/dis;

y\_sum=y\_sum+y/dis;

xx\_sum=xx\_sum+(x/dis)\*(x/dis);

xy\_sum=xy\_sum+(x/dis)\*(y/dis);

num=num+1;

end

end

else

begin

k\_up=xy\_sum\*num-x\_sum\*y\_sum;

k\_down=xx\_sum\*num-x\_sum\*x\_sum;

b\_up=(y\_sum\*xx\_sum-x\_sum\*xy\_sum)\*dis;

b\_down=xx\_sum\*num-x\_sum\*x\_sum;

choice=bluetooth\_data;

num=0;

x\_sum=0;

y\_sum=0;

xx\_sum=0;

xy\_sum=0;

End

判断直线部分：

if(k\_down>0)

begin

if(x\*k\_up+b\_up<=(y+5)\*k\_down && x\*k\_up+b\_up>=(y-5)\*k\_down)

begin

red=15;

green=0;

blue=0;

end

end

else

begin

if(x\*k\_up+b\_up<=(y-5)\*k\_down && x\*k\_up+b\_up>=(y+5)\*k\_down)

begin

red=15;

green=0;

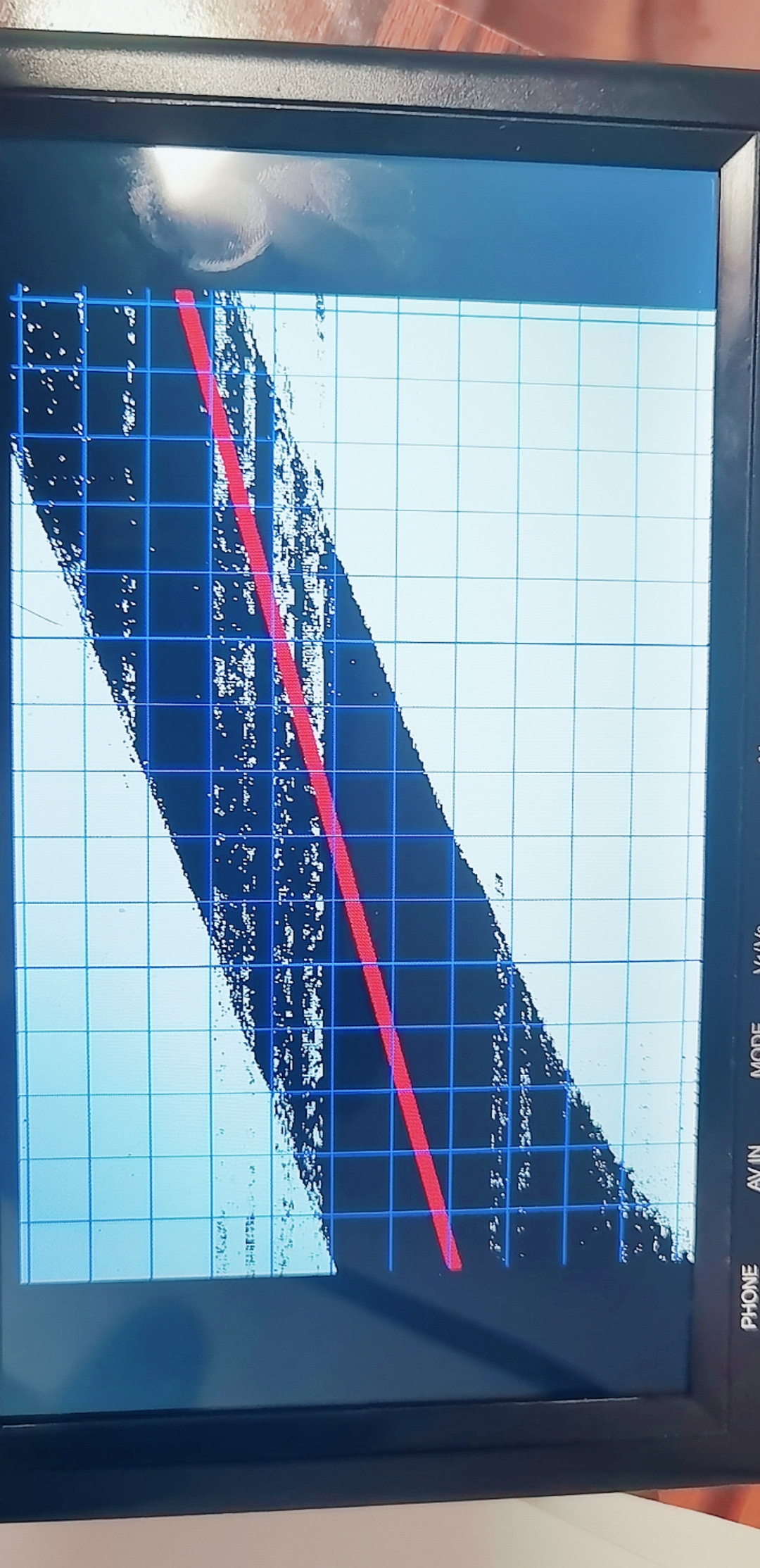
blue=0;

end

end

下板后可以正确显示直线。

但是仍然存在部分问题：



以上图为例

1. 由于此时我们只考虑处于网格上的点，导致采样的点会比以前少很多，这样噪点对于直线拟合的影响会很大。
2. 倘若使用较粗的物体拟合直线来避免噪点问题，会导致出现图上红圈部分的多余部分，该部分会导致直线拟合不准确。

为了解决该问题，提供蓝牙信号控制，可以调整网格间距，得出最适合此时环境的网格间距，提高计算准确度。

同时也可以用提高环境亮度，减少阴影干扰等办法减小噪点干扰。

1. 结果与总结
2. 结果：使用蓝牙模块，摄像头模块，vga模块以及开发板，实现了对于直线的拟合。由于硬件条件不足，导致成品对于环境干扰较为敏感。
3. 总结：本次大作业让我熟悉了各个模块的协议以及实现方式。可以发现，协议的本质是打包信号，接受之后解码。该设计思路同样可以用在网页设计前后端数据传输、错误码等领域。该设计思路对于信息传输的安全性，便利性都有很大帮助，在以后的项目中可以使用。其次各个模块之间的配合设计让我对于设计逻辑有了更清晰的认识。图像处理，以及各种算法的优化，则是对于verilog语言理解的提升。