

华中科技大学

2022

嵌入式系统

课程实验报告

题 目:	香橙派相关实验
专 业:	计算机科学与技术
班 级:	CS2009
学 号:	
姓 名:	raindaydream
电 话:	187xxxx5392
邮 件:	daydream@hust.edu.cn

华中科技大学课程设计报告

目 录

1 实验一 系统烧录	2
1.1 实验要求	2
1.2 实验过程	2
1.3 实验结果	4
2 实验二 图形界面基础	6
2.1 实验要求	6
2.2 实验过程	6
2.3 实验结果	8
3 实验三 图片文字显示	9
3.1 实验要求	9
3.2 实验过程	9
3.3 实验结果	12
4 实验四 多点触摸开发	16
4.1 实验要求	16
4.2 实验过程	16
4.3 实验结果	18
5 实验五 蓝牙通讯	20
5.1 实验要求	20
5.2 实验过程	20
5.3 实验结果	21
6 实验六 综合实验	23
6.1 实验要求	23
6.2 实验过程	23
6.3 实验结果	24
7 实验总结与建议	25
7.1 实验总结	25
7.2 实验建议	25

1 实验一 系统烧录

1.1 实验要求

对 OrangePi4 进行开发，对其实现镜像烧录并启动，在此基础上远程登录开发板并配置，完成一个简单程序在开发板上的编译和运行。

1.2 实验过程

1.2.1 烧录

不同于教程所提供的 Linux 下的烧录方式，我采用了 Windows 下烧录的方法完成了烧录。

步骤：

1. 在网上搜索烧录软件 balenaEtcher 并下载安装 Windows 版本；
2. 在 Windows 下解压 OrangePi4-lts_3.0.6_ubuntu_jammy_desktop_xfce_linux5.18.5.zip 得到镜像文件；
3. 将 SD 卡插入读卡器中并连接电脑主机；
4. 打开 balenaEtcher 选择解压的镜像文件；
5. balenaEtcher 中选择 SD 卡对应的路径；
6. 点击 flash 进行烧录。

将镜像文件烧录到 SD 卡之后将该 SD 卡插入 orangepi4 对应的 SD 卡插口处，并正确连接屏幕，对 orangepi4 接入电源后显示了启动界面即表示烧录完成。

1.2.2 远程连接

课堂所提供的教程为 Linux 下通过网线连接自动分配 IP 地址的方式完成的，在实验过程中我发现了一种更为简单的方案连接 orangepi 开发板。

远程连接的工作有两步：分配 IP 地址和 ssh 连接。

1. 分配 IP 地址

打开电脑的热点，在香橙派桌面尚未禁用的时候连接电脑热点所提供的 WiFi。在连接成功后将会为开发板分配一个 IP 地址，在设置中可以直接查找到该 IP 地

址。

在之后每次打开 orangepi4 开发板的时候，打开电脑热点，开发板都会自动连接该 WIFI，从而可以获取到 IP 地址，一般来讲该 IP 地址不会改变。

2. ssh 连接

在获取到 IP 地址后，使用 mobarxterm 管理工具采用 ssh 连接，连接到该 IP 地址并绑定用户 root，成功后便可以远程登陆开发板的 root 用户。

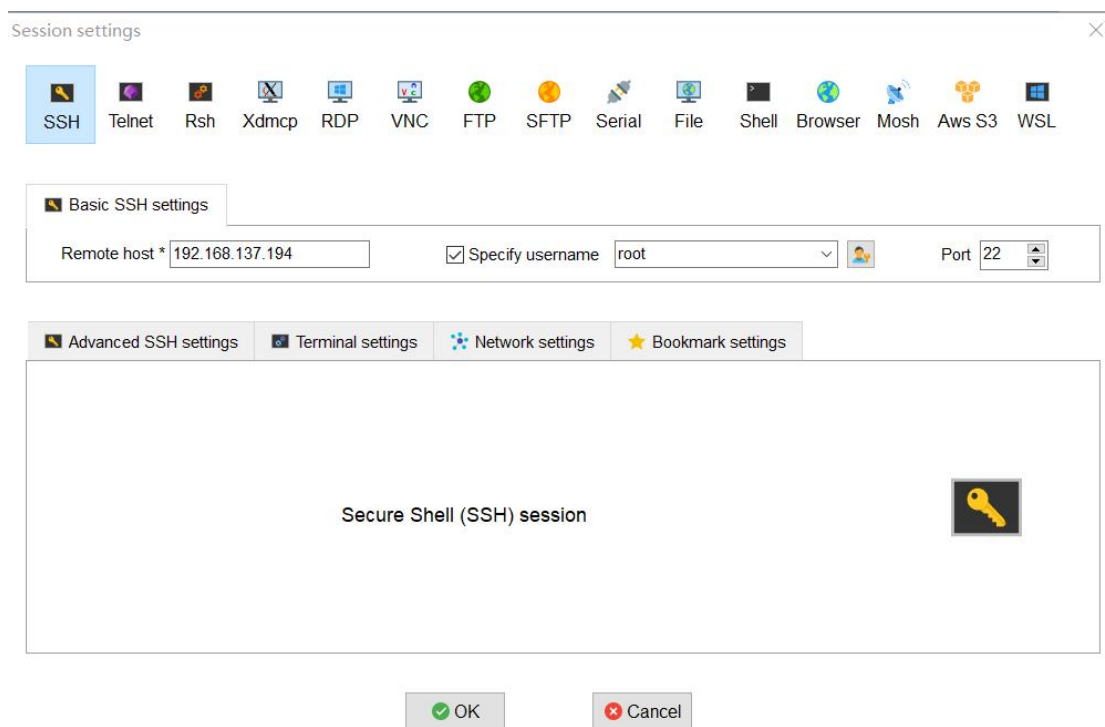


图 1.1 连接开发板

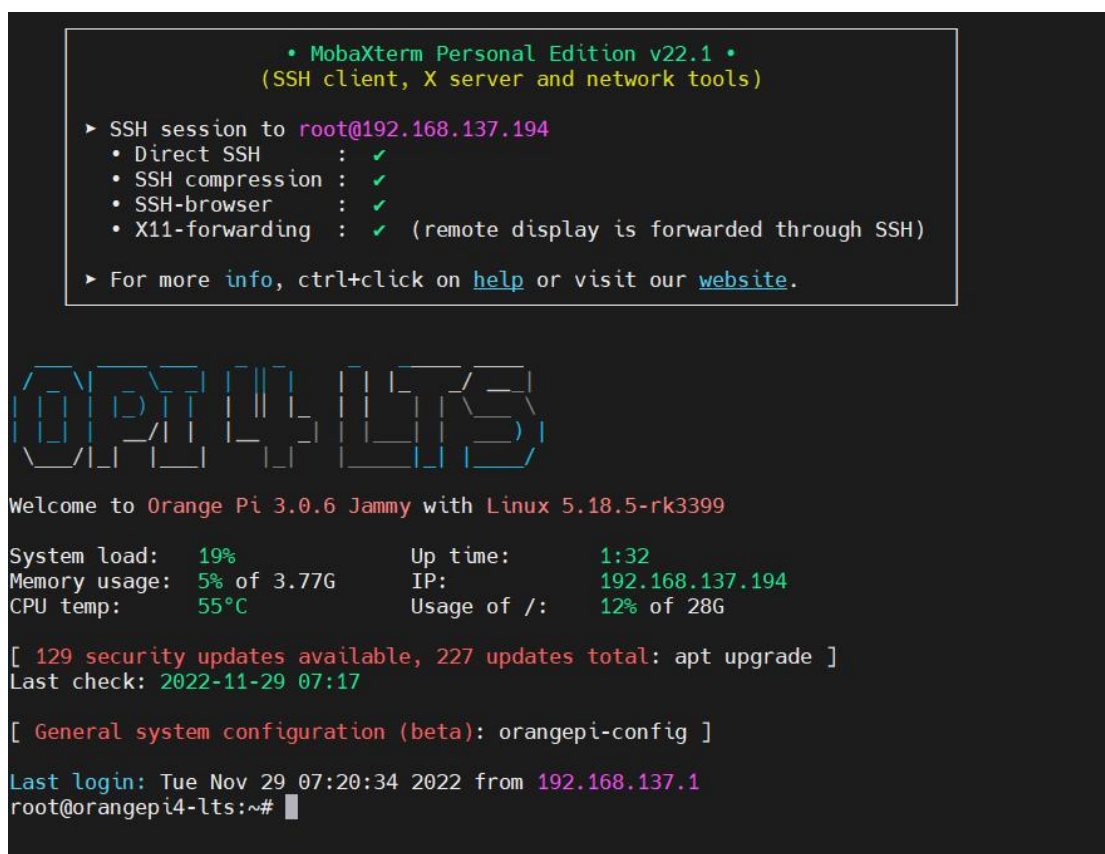


图 1.2 连接成功并打开

1.2.3 配置编译环境

由于我使用的是 ssh 连接远程登录开发板来运行代码,因此需要配置 orangepi 开发板的本地编译环境来编译运行代码。

Mobarxterm 工具支持直接从 Windows 下拖动文件到连接的主机上,因此可以直接将实验所需要的代码文件拖动到该管理工具中。

本地编译需要编辑编译器配置文件 rules.mk 文件删除 gcc 交叉编译的编译代码并保存,便可正常的对实验文件 make 编译并运行。

1.3 实验结果

在完成了 1.2 中的烧录、连接、编译环境配置之后便可以编译运行实验一的简单代码。

代码:

华中科技大学课程设计报告

```
#include <stdio.h>
int main(int argc, char* argv[])
{
    printf("Hello embedded linux!\n");
    return 0;
}
```

运行结果:

```
Last login: Tue Nov 29 07:20:34 2022 from 192.168.137.1
root@orange4-lts:~# cd lab-2022-st/
root@orange4-lts:~/lab-2022-st# cd lab1
root@orange4-lts:~/lab-2022-st/lab1# make
cc -Wall -o lab1 ../common/graphic.o ../common/task.o main.o -L../common/external/lib -ljpeg -lfreetype -lpng -lasound -lz -lc -lm
mv lab1 ../out/
root@orange4-lts:~/lab-2022-st/lab1# ../out/lab1
Hello embedded linux!
```

图 1.3 运行成功

2 实验二 图形界面基础

2.1 实验要求

1. 掌握 Linux 下的 LCD 显示驱动接口：framebuffer 的使用原理。
2. 掌握基本图形的显示：点、线、矩形区域。
3. 掌握双缓冲机制。

2.2 实验过程

2.2.1 画线函数 fb_draw_line

对于 (x_1, y_1) 为起点， (x_2, y_2) 为终点的线，需要计算斜率并根据斜率判断线中间点的位置。

以 (x_1, y_1) 为起点，需要判断线上显示的点的个数、斜率和点的伸展方向。

点的个数：起点和终点之间有宽与高，由于需要不间断显示点，所以将以 $\max(\text{width}, \text{height})$ 作为点的个数；

斜率：确定了点的个数以后，斜率 $= \min(\text{width}, \text{height}) / \max(\text{width}, \text{height})$ ；

点的伸展方向：对于一条 360° 都可以伸展的线来说，点的伸展不能按照自增的方式计算。这里我按照 (x_1, y_1) 为起点，因此需要知道在循环过程中当前的点与上一个点的关系，该关系可以通过两点之间的 x 和 y 轴的坐标关系确定，即设定两个 flag 为伸展方向的标志，根据 1 \rightarrow 2 的 x 和 y 的大小标志 flag 为 1 或 -1，每次循环都可以通过 flag 乘上该方向的自增单位来更新坐标值。。

因此代码为：

```
void fb_draw_line(int x1, int y1, int x2, int y2, int color)
{
    /*-----*/
    int width,height,num,flag_x,flag_y;
    float k;
    if(x2>x1) flag_x = 1,width = x2-x1+1;
    else flag_x = -1,width = x1-x2+1;
    if(y2>y1) flag_y = 1,height = y2-y1+1;
    else flag_y = -1,height = y1-y2+1;
    if(width<height){
        num = height;
```

```
k = 1.0*width/height;//斜率
for(int i=0;i<num;i++)
{
    int cur_x = x1+k*i*flag_x;
    int cur_y = y1+i*flag_y;
    fb_draw_pixel(cur_x,cur_y,color);//找线上的下一个点，调用画点函数画点
}
}
else{
    num=width;
    k = 1.0*height/width;
    for(int i=0;i<num;i++)
    {
        int cur_x = x1+i*flag_x;
        int cur_y = y1+k*i*flag_y;
        fb_draw_pixel(cur_x,cur_y,color);
    }
}
/*-----*/
return;
}
```

2.2.2 画矩形函数 fb_draw_rect

对于从 (x, y) 开始的宽为 w, 高为 h 的矩形, 需要对 w*h 个点进行 color 赋值。但是由于要画的矩形位置可能会越界导致点的位置赋值错误, 因此需要先将一些信息进行判断并更新为合法位置信息, 需要更新的信息有起始位置和长和宽的值, 放置其输出的点的位置超出屏幕。

实验代码为:

```
//画矩形的函数, 实心矩形
void fb_draw_rect(int x, int y, int w, int h, int color)
{
    if(x < 0) { w += x; x = 0; }
    if(x+w > SCREEN_WIDTH) { w = SCREEN_WIDTH-x; } //更新起始点的位置和宽度
    if(y < 0) { h += y; y = 0; }
    if(y+h > SCREEN_HEIGHT) { h = SCREEN_HEIGHT-y; } //更新起始点的位置和高度
    if(w<=0 || h<=0) return;
    int *buf = _begin_draw(x,y,w,h); //buf 指向保存了图像所有像素点的颜色值的开始位置
    /*-----*/
}
```



```
int i,j;
for(i=0;i<w;i++)
    for(j=0;j<h;j++)
        *(buf + x+i + (y+j)*SCREEN_WIDTH) = color;//对矩形对应屏幕的位
置赋像素点 color 值
/*-----*/
return;
}
```

2.3 实验结果

根据实验提供的 test 对所写的代码进行测试，运行结果为：

```
root@orangepi4-lts:~/lab-2022-st/lab1# cd ..
root@orangepi4-lts:~/lab-2022-st# cd lab2
root@orangepi4-lts:~/lab-2022-st/lab2# make
cc -Wall -o lab2 ../common/graphic.o ../common/touch.o ../common/image.o ../common/task.o main.o -L../common/external/lib -ljpeg -lfreetype -lpng -lasound -lz -lc -lm
mv lab2 ../out/
root@orangepi4-lts:~/lab-2022-st/lab2# ../out/lab2
framebuffer info: bits_per_pixel=32,size=(1024,768),virtual_pos_size=(0,0)(1024,768),line_length=4096,smem_len=3145728
===== Start Test =====
draw pixel: 6 ms
draw rect: 6 ms
draw line: 2 ms
root@orangepi4-lts:~/lab-2022-st/lab2#
```

图 2.1 lab2 运行结果成功图

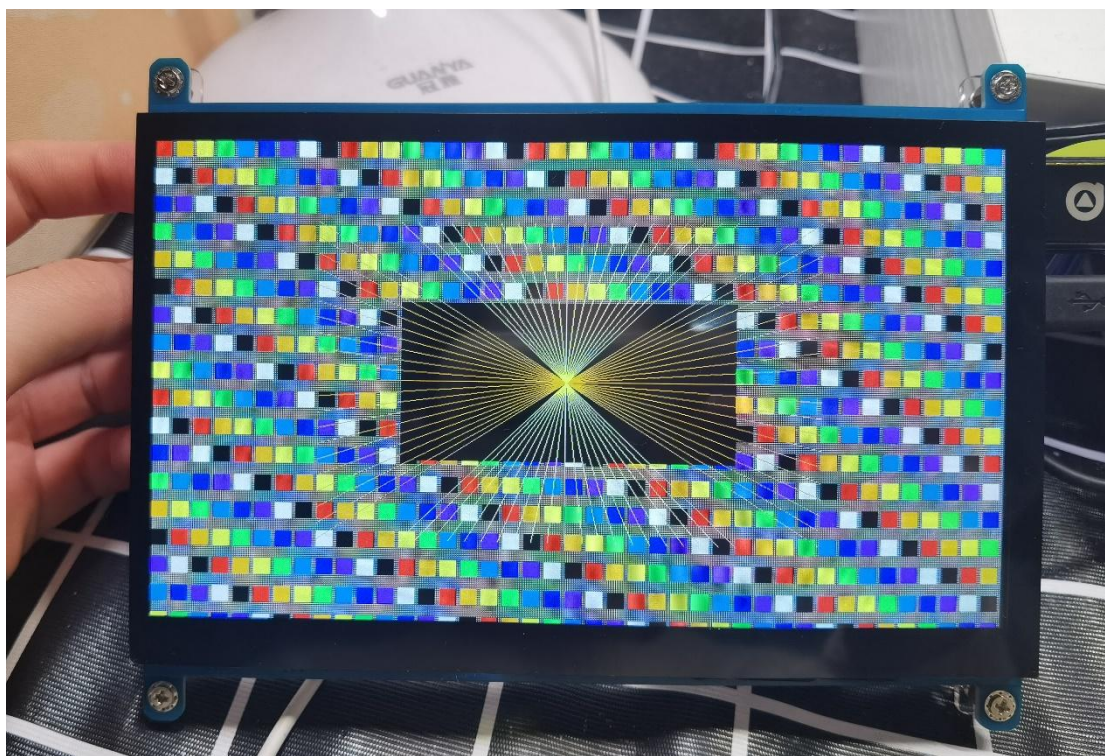


图 2.2 lab2 实验效果图

3 实验三 图片文字显示

3.1 实验要求

本实验需要做到：

1. jpg 不透明图片显示；
2. png 半透明图片显示；
3. 矢量字体显示：字模的提取和显示。

3.2 实验过程

一张图片显示在屏幕上需要知道其起始位置信息、长宽信息和每一个点的 color 信息，在图片显示中，每一个像素点所占的字节为 4 个，包含透明度和 RGB 值。

在开始画图时，需要先根据图片是否超出屏幕范围对其更新上面所说的几个信息。如果起始位置在屏幕左上部分的屏幕外，那么需要将可以显示的长宽减少一部分，同时图像可显示的起始位置也会向右下移动相应部分。如果右下部分超出屏幕右下，需要将可以显示的长宽更新。

画图这里需要两个指针，一个 dst 指向屏幕显示的第一个位置，该位置由起始位置 (x, y) 确定；另一个 src 指向图片的像素点颜色信息中开始显示的第一个位置，该位置由计算得到的图像可显示的起始位置确定。

接下来根据图片的格式对屏幕像素点的内存进行 color 赋值。三种图片的 color 赋值循环方式都相同，根据上面所说的信息更新方式得到两个指针正确的起始位置和长宽，接下来只需要采用类似于矩形画图的方式对指针进行移动即可，但这里每次要移动的位置为 4 个。

不同的是每个像素点四个字节的赋值计算方式不同。对于 jpg 图片，将四个字节直接赋值即可；对于 png 图片，需要根据第四个字节的透明度计算真实显示的 RGB 的值： $color = color1 * alpha1 + color2 * (1 - alpha1)$ ；对于矢量字体，其计算方式和 png 的计算方法相同，但是不同的是矢量字体中只保存每一个像素点的透明度，因此每次更新 src 位置的时候只自增 1。

因此代码为：

```
void fb_draw_image(int x, int y, fb_image *image, int color)
{
    if(image == NULL) return;

    int ix = 0; //image x
    int iy = 0; //image y
    //ix,iy 表示该图像从哪个位置开始显示,因为如果起始位置超出了屏幕,图像显示的
    起始位置有所更新
    int w = image->pixel_w; //draw width
    int h = image->pixel_h; //draw height
    if(x<0) {w+=x; ix-=x; x=0;}
    if(y<0) {h+=y; iy-=y; y=0;}//更新图像显示起始位置和屏幕显示起始位置
    if(x+w > SCREEN_WIDTH) {
        w = SCREEN_WIDTH - x;
    }
    if(y+h > SCREEN_HEIGHT) {
        h = SCREEN_HEIGHT - y;
    }
    if((w <= 0)|| (h <= 0)) return;
    int *buf = _begin_draw(x,y,w,h);
    /*-----*/
    char *dst = (char *) (buf + y*SCREEN_WIDTH + x); //指向屏幕显示位置的第一个
    像素点
    char *src=image->content + iy*image->line_byte + ix*4; //指向图片显示
    位置的第一个像素点
    //不同的图像颜色格式定位不同
    /*-----*/
    int i,j;
    if(image->color_type == FB_COLOR_RGB_8880) /*lab3: jpg*/
    {
        for(i = 0; i < h ; i++)//每一行
        {
            for(j=0;j<w;j++)//每一列
            {
                *(dst+j*4+0)=*(src+j*4+0);
                *(dst+j*4+1)=*(src+j*4+1);
                *(dst+j*4+2)=*(src+j*4+2);
                *(dst+j*4+3)=*(src+j*4+3); //jpg 的一个像素点由 4 byte 表示
            }
            dst += 4*SCREEN_WIDTH; //指向下一行屏幕起始显示位置
            src += image->line_byte; //指向下一行图像起始显示位置
        }
        return;
    }
```

```
}
else if(image->color_type == FB_COLOR_RGBA_8888) /*lab3: png*/
{
    for(int i=0;i<h;i++)
    {
        for(int j=0;j<w;j++)
        {
            //像素点由透明度和像素值共同决定
            dst[0]+=((src[0]-dst[0])*src[3])>>8;
            dst[1]+=((src[1]-dst[1])*src[3])>>8;
            dst[2]+=((src[2]-dst[2])*src[3])>>8; //png 的一个像素点由 4
byte 表示
            dst+=4;
            src+=4; //下一个像素点
        }
        dst += 4*SCREEN_WIDTH-4*w;
        src += image->line_byte-4*w; //下一行像素点
    }
    return;
}
else if(image->color_type == FB_COLOR_ALPHA_8) /*lab3: font*/
{
    char *pcolor=(char*)&color;
    for(int i=0;i<h;i++)
    {
        for(int j=0;j<w;j++)
        {
            //像素点由透明度和 color 共同决定
            dst[0]+=((pcolor[0]-dst[0])*src[0])>>8;
            dst[1]+=((pcolor[1]-dst[1])*src[0])>>8;
            dst[2]+=((pcolor[2]-dst[2])*src[0])>>8;
            dst[3]+=((pcolor[3]-dst[3])*src[0])>>8;
            dst+=4;
            src+=1;
        }
        dst += 4*SCREEN_WIDTH-4*w;
        src += image->line_byte-w;
    }
    return;
}
/*-----*/
return;
}
```


3.3 实验结果

在该实验中，需要结合实验 2 的内容一起进行 test 测试，在本实验还有一个优化环节，这里我采用了系统调度方面的优化，采用了指令：`taskset -c 5 nice -n -20 ./test` 进行了优化。

实验过程和最终的结果为：

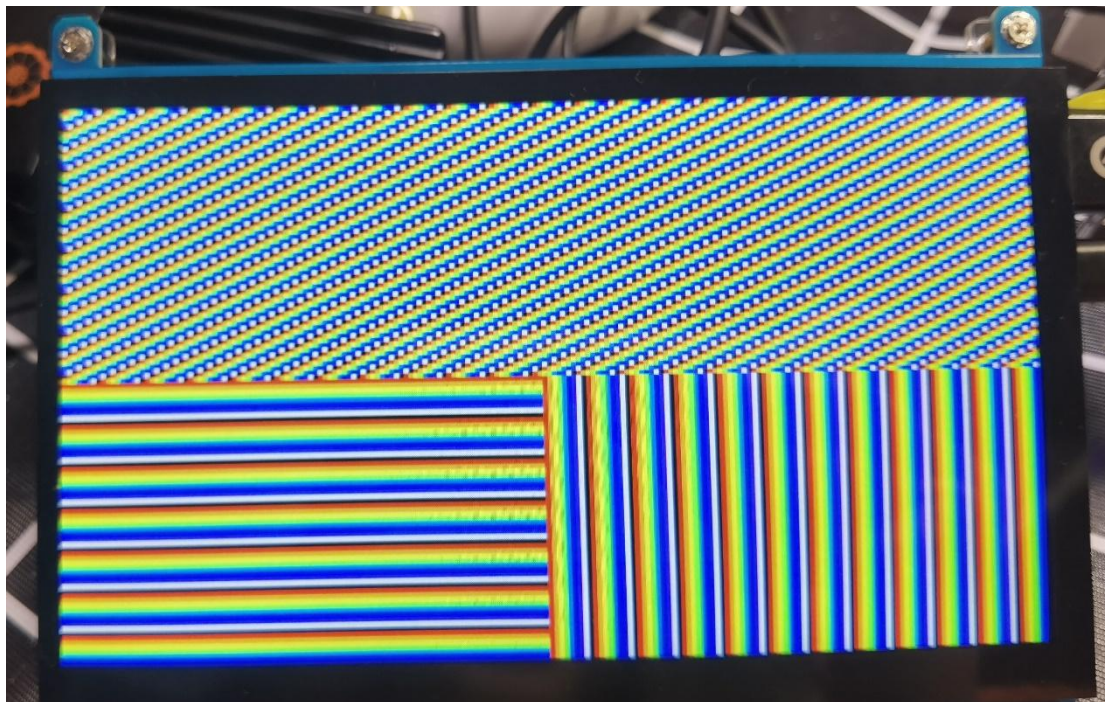


图 2.3 lab3 实验过程效果图

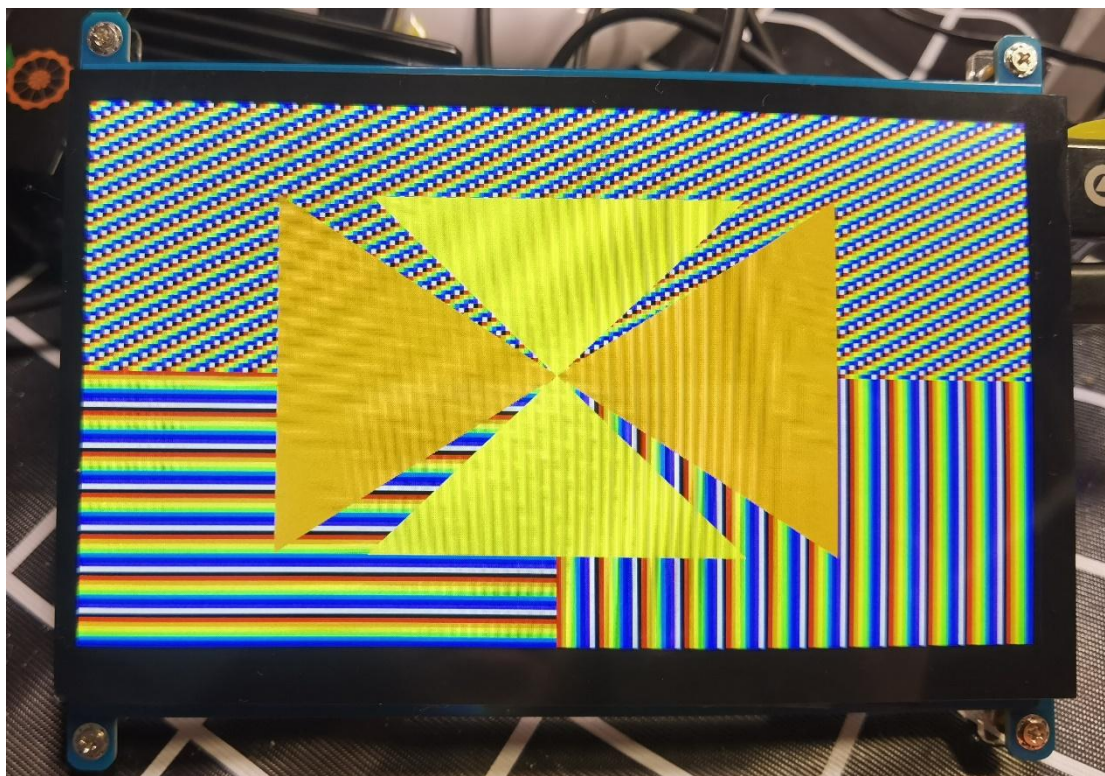


图 2.4 lab3 实验过程效果图



图 2.5 lab3 实验过程效果图

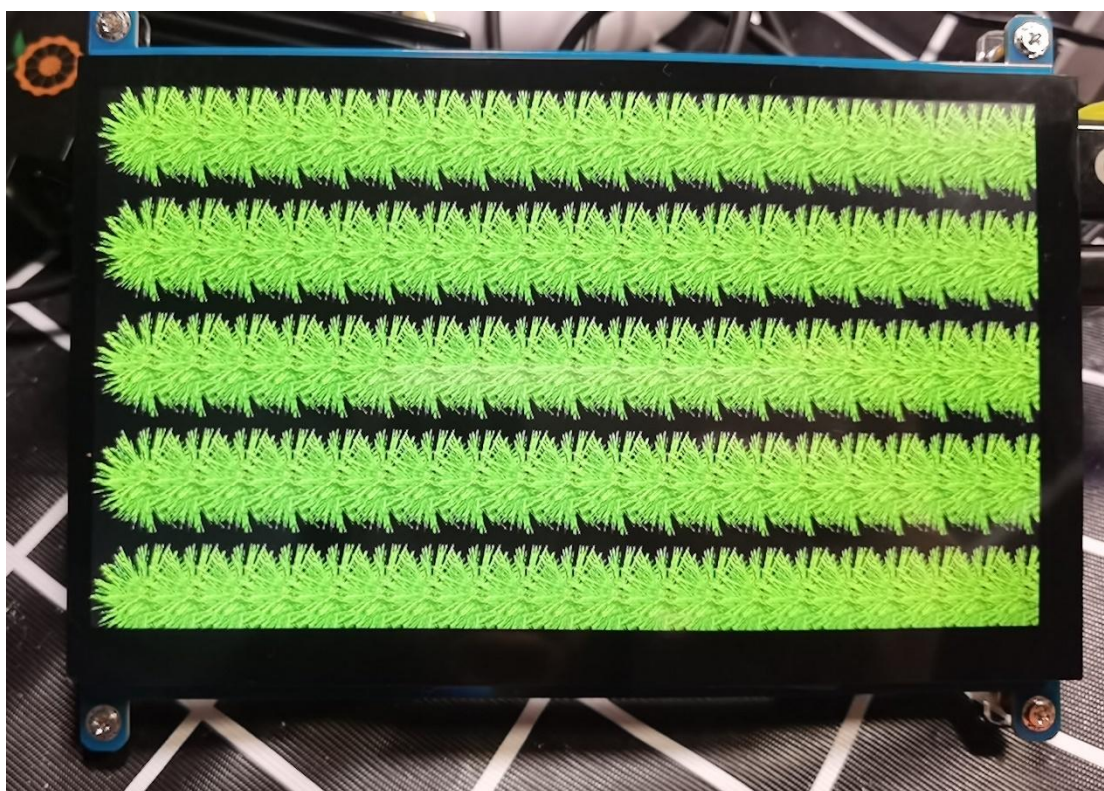


图 2.6 lab3 实验过程效果图

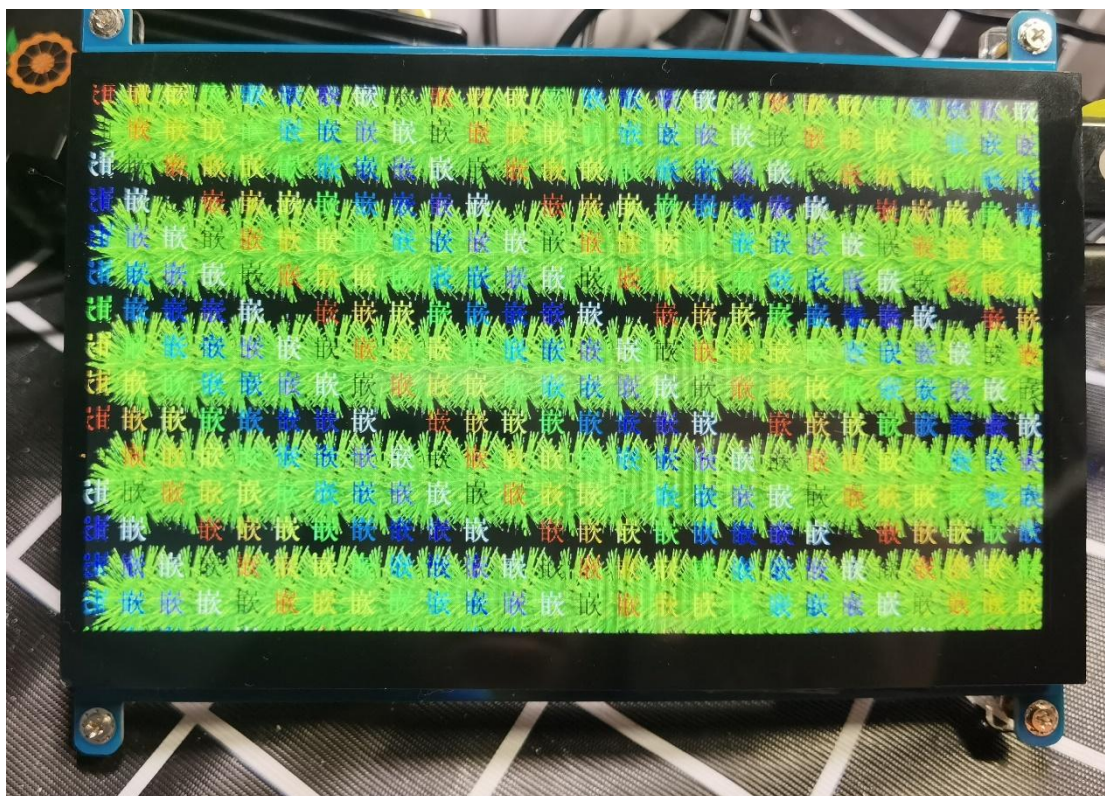


图 2.7 lab3 实验过程效果图

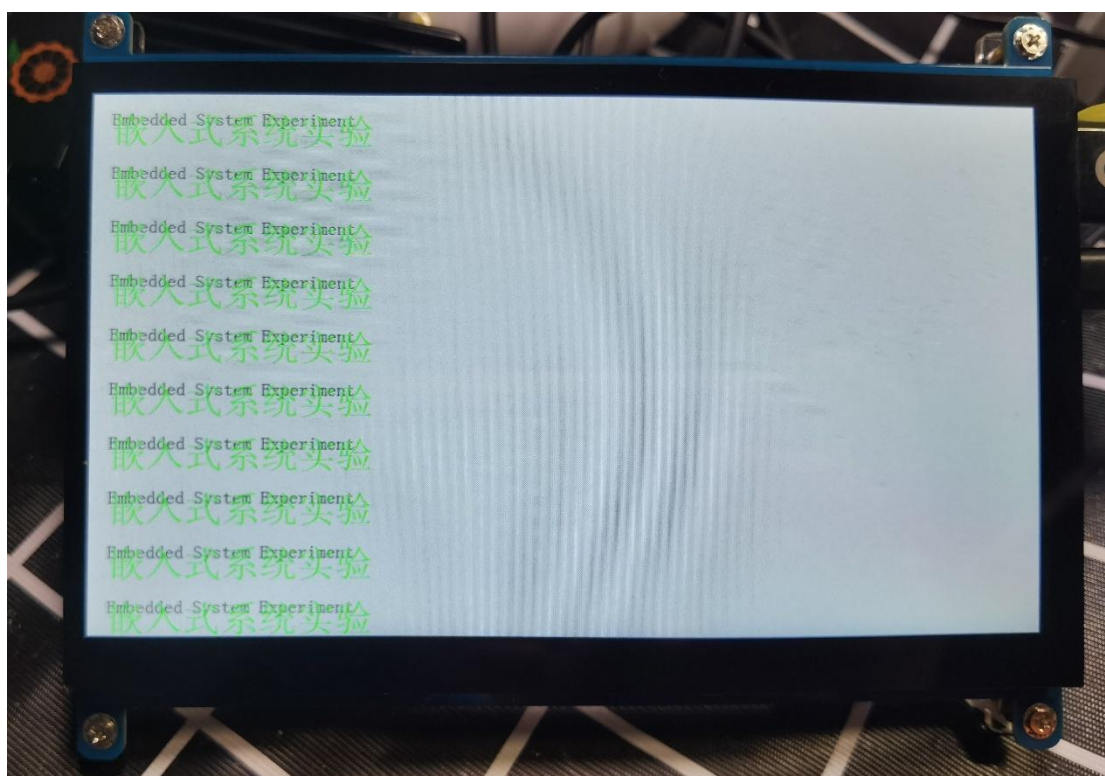
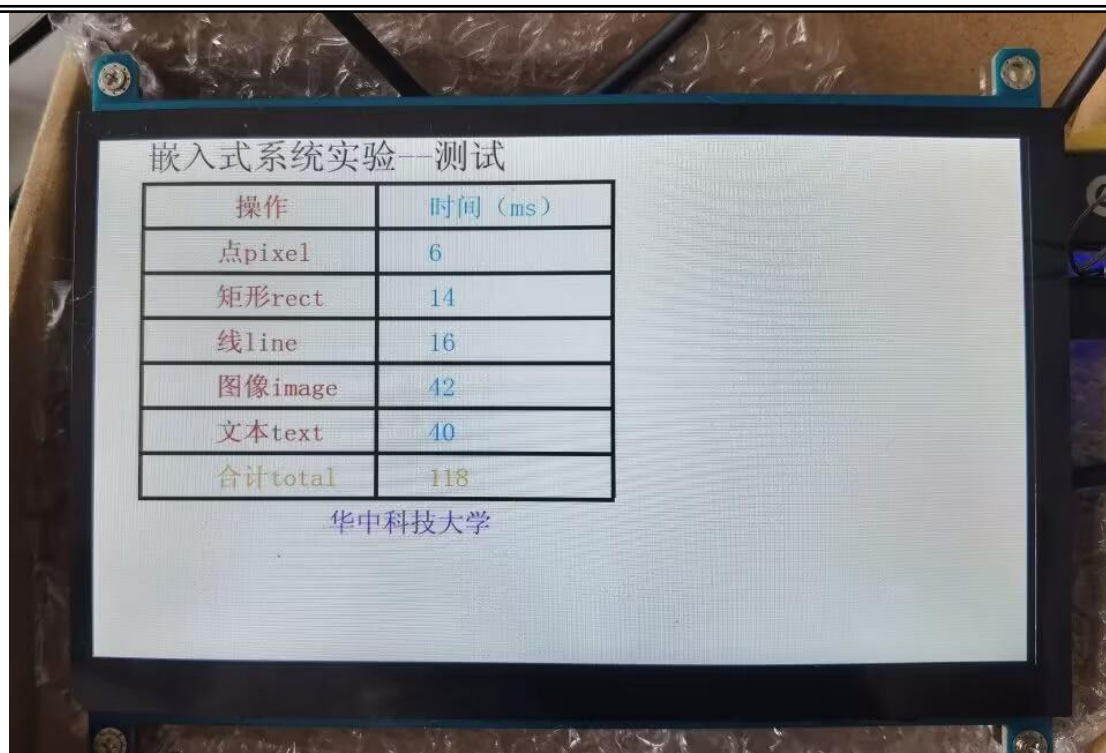


图 2.8 lab3 实验过程效果图



嵌入式系统实验—测试

操作	时间 (ms)
点pixel	6
矩形rect	14
线line	16
图像image	42
文本text	40
合计total	118

华中科技大学

图 2.9 lab3 实验最终效果图

4 实验四 多点触摸开发

4.1 实验要求

本实验要求掌握并实现：

1. Linux 下的触摸屏驱动接口:Input event 的使用和多点触摸协议(Multi-touch Protocol);
2. 获取多点触摸的坐标;
3. 在 LCD 上显示多点触摸轨迹;
4. 绘制一个清除屏幕的按钮, 点击后清除屏幕内容;
5. 二选一实现画板/画圆, 要求为:
 - (1) 每个手指轨迹的颜色不同/对应每个手指触点的圆的颜色不同;
 - (2) 轨迹是连贯的, 不能断断续续/实时跟踪触点, 只显示当前位置;
 - (3) 轨迹要比较粗, 线宽不能是 1 个点/之前位置的圆要清除掉;
 - (4) 绘制一个清除屏幕的按钮, 点击后清除屏幕内容/屏幕不能明显闪烁。

4.2 实验过程

在本实验中, 我选择了画圆的任务。针对主函数, 这里没有修改的内容, 主函数完成了打开设备并初始化、初始化一个触摸设备接口并传递一个任务无限循环的任务。但需要注意的是, 触摸设备接口需要根据连接屏幕的接口确定, 一个较为方便的方法是确定接口的编号并每次将屏幕插在文件中打开的接口中。

函数 `touch_event_cb` 用来根据触摸屏幕采取相应的措施。该函数会调用 `touch_read` 函数获取当前屏幕的触摸信息, 根据函数返回信息对其进行相应处理。

①当前触摸为按压: 对该位置进行画圆, 该操作通过调用封装好的函数完成, 并将该位置保存下来, 保存到数组对应相应的手指位置中;

②当前触摸为移动: 将该手指保存的对应位置提取出来, 将该位置的圆删除, 该操作通过封装好的删除函数完成, 同时将保存的上一个圆的位置更新;

③当前触摸为释放: 将该圆删除, 并将对应的保存上一个圆的位置删除;

④当前触摸为错误操作：直接关闭触摸并删除任务返回。

在整个过程中，需要有画圆函数和删除函数并将其封装，这两个函数可以通过遍历圆所在的点并调用画点函数将圆上的每一个点显示相应的颜色，画圆函数和删除函数的区别就是画点的颜色是否为背景颜色。

代码为：

```
void fb_draw_circle(int x,int y,int color,int r)
{
    if(x<0 || y<0 || x>=SCREEN_WIDTH || y>=SCREEN_HEIGHT) return;
    int i,j;
    for(i=x-r;i<x+r;i++)
    {
        for(j=y-r;j<y+r;j++)
        {
            if(i<0 || j<0 || i>=SCREEN_WIDTH || j>=SCREEN_HEIGHT)continue;
            if((i-x)*(i-x)+(j-y)*(j-y)<=r*r)
                fb_draw_pixel(i,j,color);
        }
    }
}

void fb_delete_circle(int x,int y,int r)
{
    if(x<0 || y<0 || x>=SCREEN_WIDTH || y>=SCREEN_HEIGHT) return;
    int i,j;
    for(i=x-r;i<x+r;i++)
    {
        for(j=y-r;j<y+r;j++)
        {
            if(i<0 || j<0 || i>=SCREEN_WIDTH || j>=SCREEN_HEIGHT)continue;
            if((i-x)*(i-x)+(j-y)*(j-y)<=r*r)
                fb_draw_pixel(i,j,COLOR_BACKGROUND);
        }
    }
}

static void touch_event_cb(int fd)
{
    int type,x,y,finger;
    type = touch_read(fd, &x,&y,&finger);
    switch(type){
        case TOUCH_PRESS:
            fb_draw_circle(x,y,finger_color[finger],cr);
            pre_x[finger]=x;
            pre_y[finger]=y;
```

```
printf("TOUCH_PRESS: x=%d,y=%d,finger=%d\n",x,y,finger);
break;
case TOUCH_MOVE:
    fb_delete_circle(pre_x[finger],pre_y[finger],cr);
    fb_draw_circle(x,y,finger_color[finger],cr);
    pre_x[finger]=x;
    pre_y[finger]=y;
    printf("TOUCH_MOVE: x=%d,y=%d,finger=%d\n",x,y,finger);
    break;
case TOUCH_RELEASE:
    fb_delete_circle(x,y,cr);
    pre_x[finger]=-1;
    pre_y[finger]=-1;
    printf("TOUCH_RELEASE: x=%d,y=%d,finger=%d\n",x,y,finger);
    break;
case TOUCH_ERROR:
    printf("close touch fd\n");
    close(fd);
    task_delete_file(fd);
    break;
default:
    return;
}
fb_update();
return;
}
```

4.3 实验结果

```
root@orange4-lts:~/lab-2022-st/lab4# make
cc -Wall -o lab4 ../common/graphic.o ../common/touch.o ../common/image.o ../common/task.o main.o -L../common/external/lib -ljpeg -lfreetype -lpng -lasound -lz -lc -lm
mv lab4 ../out/
root@orange4-lts:~/lab-2022-st/lab4#
```

图 2.10 代码运行成功截图

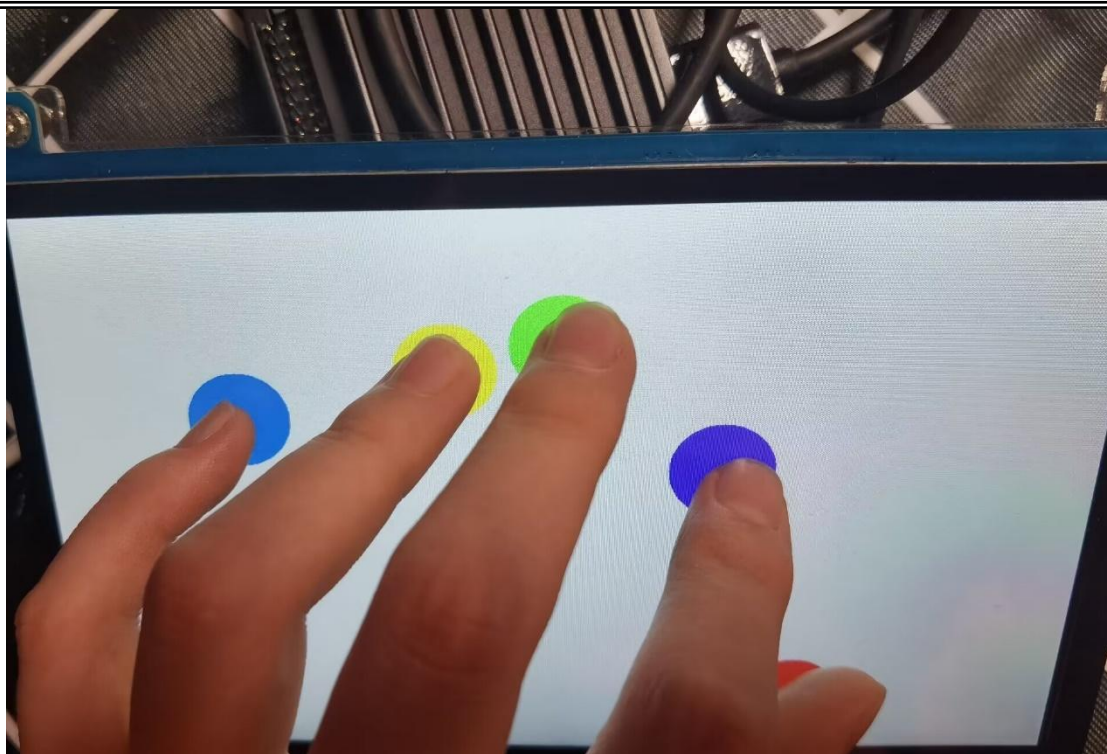


图 2.11 运行效果

5 实验五 蓝牙通讯

5.1 实验要求

1. 正确配置并启动蓝牙服务；
2. 熟练使用蓝牙的常用命令工具；
3. 通过 RFCOMM 串口（蓝牙串行通讯）实现；
4. 无线通讯，测试通过 lab5 实验代码。

5.2 实验过程

该实验不需要编写代码，而是通过命令行对蓝牙服务进行操作从而使得手机串口工具与 orangepi 的蓝牙连接可以通讯。

根据课程所提供的教程，需要：

①编辑配置文件：增加兼容性标志并添加一行新的代码，配置后重启生效；

②打开蓝牙：采用 bluetoothctl 工具，使用 power on 打开蓝牙；

③与手机配对：orangepi 在 bluetoothctl 工具中采用 scan on 命令进行扫描，手机打开蓝牙，寻找 orangepi4-lts，此时有两种配对方式。第一种配对方式由手机扫描到 orangepi4-lts，从手机发送配对请求，板子收到请求后在命令行中选择 yes/no 进行配对；另一种方法是从板子扫描找到手机的 mac 地址，找到了以后通过 pair 指令与手机进行配对请求，手机同意是否配对；

④打开服务端：在 orangepi4-lts 中通过命令行 rfcomm -r listen/watch 0 1 打开服务端；

⑤连接蓝牙：打开手机上的串口工具（此处我使用的是 SPP 蓝牙串口），与 orangepi4-lts 进行连接；

⑥通信：保持服务端打开，并打开 orangepi4-lts 的另一个终端，在该终端编译运行 lab5 的代码，运行后，在手机串口工具上发送信息给服务端，orangepi4-lts 的屏幕上就会显示出发送的信息。

该实验主要是了解蓝牙的连接使用，蓝牙只需要在一开始使用工具进行配对即可，在之后的使用中，直接连接即可，只要这两个主机不改变自己的主机名称。由于我在这次的实验中是使用手机和板子进行通讯，而手机的一般蓝牙不能发送信息，只能共享信息，因此需要一个串口工具进行连接并从手机传送一些可以自

华中科技大学课程设计报告

已编辑的文本。

两个 orangepi4-lts 之间进行蓝牙通讯的步骤与手机与板子进行通讯的步骤差不多，只需要：

①对两个 orangepi4-lts 进行编辑配置文件并打开蓝牙；

②配对：两个板子在蓝牙工具中 scan on 不断扫描，扫描到另一个板子的时候，该板子采用 pair 进行配对请求，另一个板子命令行输入 yes 进行配对；、

③连接：打开一个板子的客户端，打开另一个板子的服务端，两板子连接；

④通讯：两个板子都打开另一个终端运行 lab5，在客户端发送信息，服务端的板子屏幕上显示发送的信息。

5.3 实验结果

```
Request confirmation
[agent] Confirm passkey 012206 (yes/no): [NEW] Device 68:71:EF:4B:E3:B2 68-71-EF-4B-E3-B2
[agent] Confirm passkey 012206 (yes/no): [CHG] Device 9C:74:03:90:0B:1C RSSI: -79
[agent] Confirm passkey 012206 (yes/no): [CHG] Device 9C:74:03:90:0B:1C RSSI: -71
[agent] Confirm passkey 012206 (yes/no): yes
[CHG] Device 5A:BD:2C:0D:5C:44 RSSI: -73
```

图 2.12 手机申请蓝牙配对

```
root@orangepi4-lts:~/lab-2022-st/out# rfcomm -r listen/watch 0 1
Waiting for connection on channel 1
Connection from B4:A8:98:D4:A8:BD to /dev/rfcomm0
Press CTRL-C for hangup
```

图 2.13 打开服务端



图 2.14 手机连接板子并发送信息

```
root@orange pi4-lts:~/lab-2022-st/out# ./lab5
framebuffer info: bits_per_pixel=32,size=(1024,600),virtual_pos_size=(0,0)(1024,
600),line_length=4096,smem_len=2457600
bluetooth tty receive "华中科技大学70周年"
bluetooth tty receive "嵌入式实验第五关"
bluetooth tty receive "测试成功"
bluetooth tty receive "444 555 666"
bluetooth tty receive "hello world!"
```

图 2.15 服务端收到信息

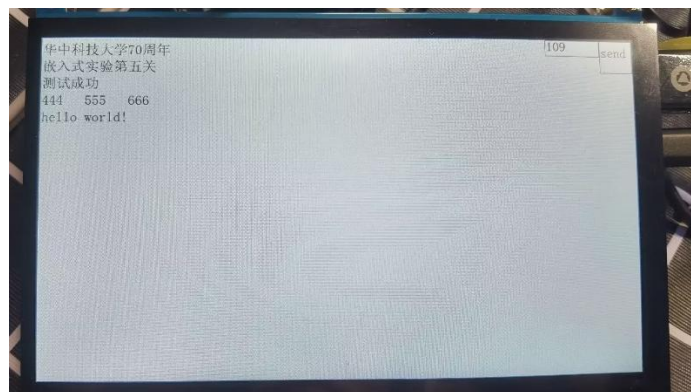


图 2.16 信息显示在屏幕上

6 实验六 综合实验

6.1 实验要求

综合运用所学内容,设计并实现一个在多个开发板之间进行蓝牙互联协作的程序或一个功能比较复杂的单机程序。例如:

1. 共享白板: 两个开发板之间共享屏幕手绘内容;
2. 共享文件: 显示远程开发板上的目录文件列表,选中指定文件,显示指定文件内容(文本和 png/jpg 图片);
3. 联网游戏: 比如五子棋等;
4. 单机程序:
 - (1) 视频播放器;
 - (2) 图片浏览器: 图片放大、缩小、图片尺寸超过显示区域时手指拖动显示;
 - (3) 游戏或其他应用程序。

6.2 实验过程

综合实验选择**共享白板**功能进行开发,使用蓝牙互联协作的方式实现多个开发板的协作,完成共享屏幕的功能。本次实验由我和周xx同学共同完成。

1. 本次实验基于实验四的屏幕绘图和实验五的蓝牙连接,主要实现 `touch_event_cb` 多点触摸事件的处理,以及 `bluetooth_tty_event_cb` 蓝牙通讯事件的处理。

2. 程序运行:

两个开发板都进行实验五相同配置,一个开发板做服务器,运行 `rfcomm -r watch 0 1`。另一个做客户端,运行 `rfcomm -r connect 0 server_mac 1`。连接成功后同时运行 `lab6`,即可实现共享白板。

3. 分工情况:

徐xx: 编写 `touch_event_cb` 手指触摸事件处理,对 4 类主要的触摸事件进行处理,实现多个手指在屏幕绘制不同颜色曲线;以及编写实现清除轨迹的按钮模块;

周xx: 编写蓝牙通讯事件的处理,将当前轨迹坐标、触摸类型、手指颜色等信息传输到远端,以及将传输的数据读出,使用已实现的方式绘制曲线。

最后我们共同完成了整个代码的调试与优化,并进行测试,记录结果。

6.3 实验结果

```
root@orange4-lts:~/out# rfcomm -r listen/watch 0 1
Waiting for connection on channel 1
Connection from D0:A4:6F:CF:B5:9C to /dev/rfcomm0
Press CTRL-C for hangup
```

图 6.1 服务端蓝牙连接

```
root@orange4-lts:~/lab-2022-st/lab6# rfcomm -r connect 0 D0:A4:6F:CF:B9:AC 1
Connected /dev/rfcomm0 to D0:A4:6F:CF:B9:AC on channel 1
Press CTRL-C for hangup
```

图 6.2 客户端蓝牙连接

```
root@orange4-lts:~/lab-2022-st/out# ./lab6
framebuffer info: bits_per_pixel=32,size=(1024,768),virtual_pos_size=(0,0)(1024,768),line_length=4096,smem_len=3145728
```

图 6.3 lab6 运行初始情况

```
root@orange4-lts:~/out# ./lab6
framebuffer info: bits_per_pixel=32,size=(1024,600),virtual_pos_size=(0,0)(1024,600),line_length=4096,smem_len=2457600
No.1 Get: status=0,x=303,y=164,finger=0,color=-12490271
No.2 Get: status=1,x=308,y=164,finger=0,color=-12490271
No.3 Get: status=1,x=314,y=165,finger=0,color=-12490271
No.4 Get: status=1,x=325,y=167,finger=0,color=-12490271
```

图 6.4 lab6 手指绘制时输出情况

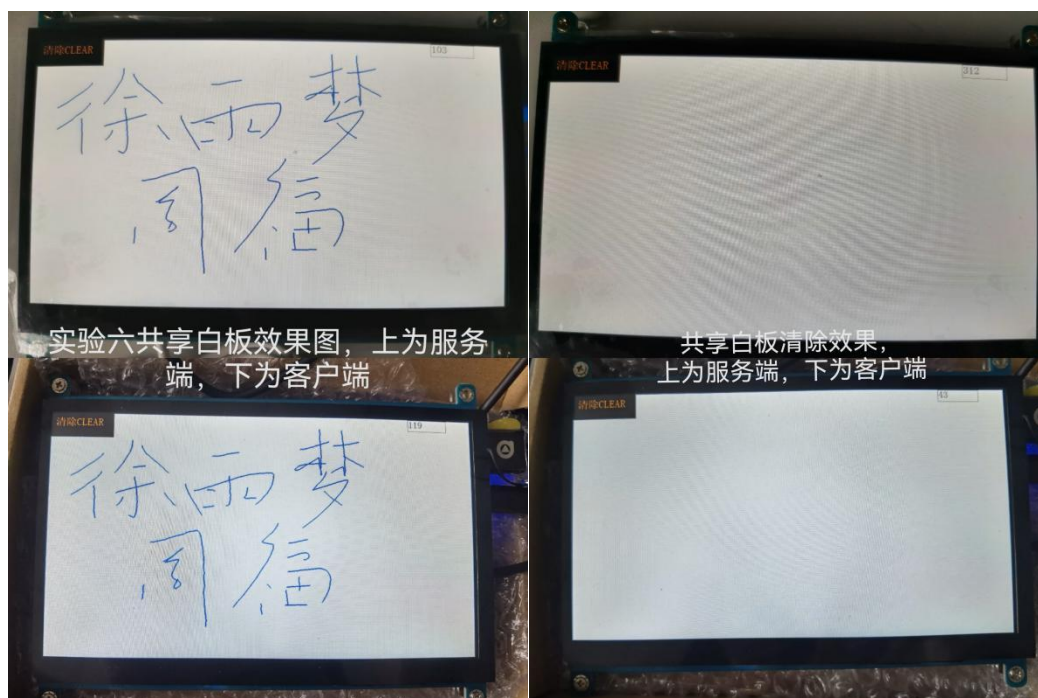


图 6.5 共享白板运行效果图与清除效果图

7 实验总结与建议

7.1 实验总结

在整个实验课的过程中，我完成了六个实验：

1. 简单程序开发：对系统进行烧录并配置环境运行了一个简单的程序；
2. 界面显示开发：点、线、矩阵区域显示；
3. 图片显示和文本显示；
4. 多点触摸开发：触摸屏幕获得反应；
5. 蓝牙无线互联通讯：实现手机与开发板通讯和开发板之间互相通讯；
6. 共享白板实现。

在完成这六个实验的过程中，一开始是困难的，因为一开始对 Linux 并不熟悉并且在一开始对实验内容没有理解到位，因此一开始花费了很多时间而收获不大。于是后来便开始与同学、朋友一起交流学习，得到了一些同学的指导和答疑，也为一些同学提供了帮助，大家一起探讨完成了实验，还总结了很多经验和教训。

这个实验也让我学到了很多，在学习了计组和操作系统之后又学习了这门实验课，让我对计算机的底层知识有了更深的理解，对其他的操作系统有了一些浅薄的了解。

7.2 实验建议

在课堂讲解该实验的时候，老师一直推荐使用 Linux 完成所有实验，但是综合我在实验过程中遇到的问题和身边的同学遇到的问题来说，我发现使用 Linux 来完成该实验在一开始的配置过程是有些麻烦的，对于不熟悉 Linux 的同学来说，这个配置过程更是痛苦的。

在这次的实验中，我使用了 ssh 连接的方式进行远程登录，发现这种方式对于新手来说更加方便上手，不需要使用双系统或者是子系统，只需要一个远程登录的工具即可，非常的方便上手，但这个方法唯一不方便的是需要在开发板打开的时候才能进行操作，但是该问题也有解决方法，在编写代码的时候，可以将需要编程的文件拖出来在本地根据功能进行编辑，编辑好在打开开发板的时候拖进去即可，也十分的方便。

整个实验是非常友好的，相信大家在老师的帮助下对这门课程也有了很好的学习。