

华中科技大学

2023

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：计卓 2101

学号：U202115285

姓名：高僖

电话：18518472108

邮件：761768812@qq.com

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.....	5
2.1 实验要求	5
2.2 实验过程	5
2.3 实验结果	7
3 实验三 图片文字显示.....	8
3.1 实验要求	8
3.2 实验过程	8
3.3 实验结果	9
4 实验四 多点触摸开发.....	11
4.1 实验要求	11
4.2 实验过程	11
4.3 实验结果	12
5 实验五 蓝牙通讯	13
5.1 实验要求	13
5.2 实验过程	13
5.3 实验结果	14
6 实验六 综合实验	15
6.1 实验要求	15
6.2 实验过程	15
6.3 实验结果	18
7 实验总结与建议	20
7.1 实验总结	20
7.2 实验建议	21

1 实验一 系统烧录

1.1 实验要求

对香橙派 OrangePi4-LTS 进行系统的烧录,并且成功运行第一个实验的代码。

1.2 实验过程

1.系统烧录

首先准备一个读卡器,插入提供好的 32GB 内存卡,使用 etcher 程序将实验文件中的 ubuntu 的 iso 文件刻录到内存卡中。拔出内存卡,断电的前提下插入到 orangepi-lts 的卡槽内,按照指导 PPT 上的说明连接好各种线材。

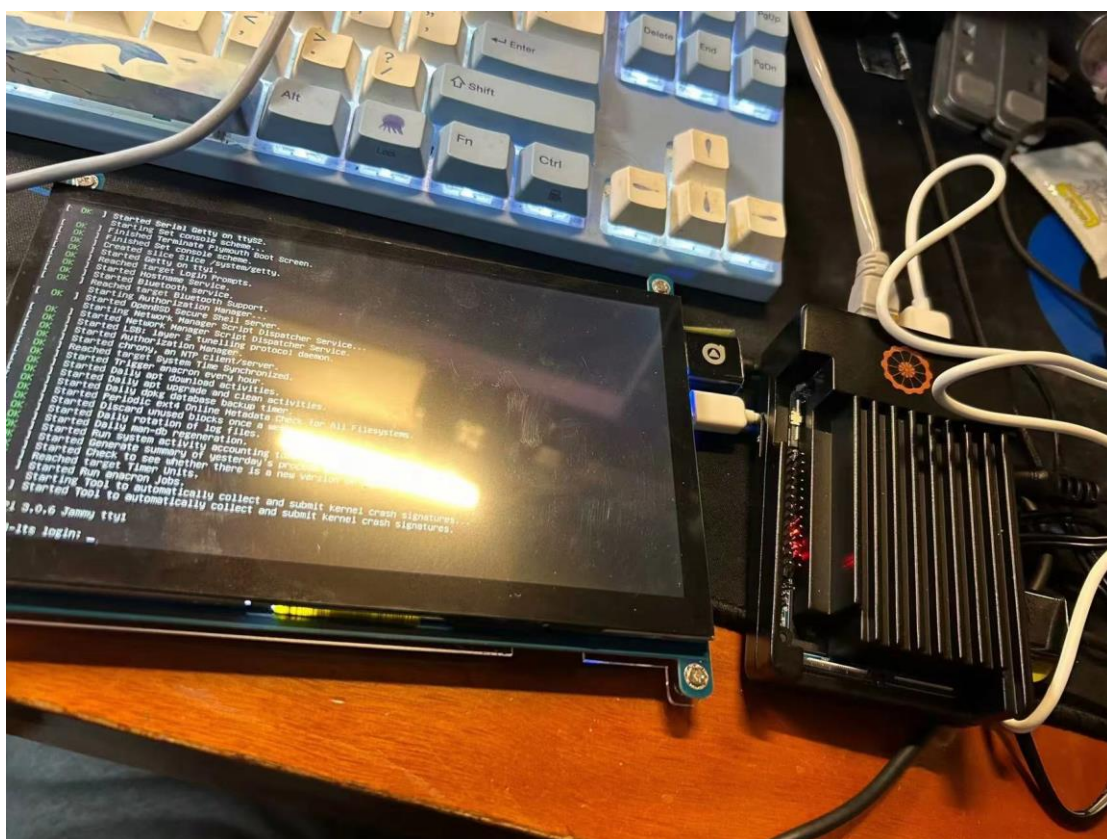


图 1-1 连接线材

2. 连接开发板

接通电源,我的 PC 使用的是 Windows 的环境,通过 MobaXterm 的 SSH 进行访问。对于开发板的设置,我是使用键盘连接开发板后通过 `ip addr show eth0` 的命令获得了 orangepi 的 ip 地址。除此之外,在 PC 中我打开了网络适配器的设置,将我与因特网进行连接的无线适配器中的共享设置中如图进行设置。如此,开发板就可以通过以太网口通过 PC 机接入因特网。(可以通过命令 `ping`

华中科技大学课程设计报告

www.baidu.com 来测试连接性)

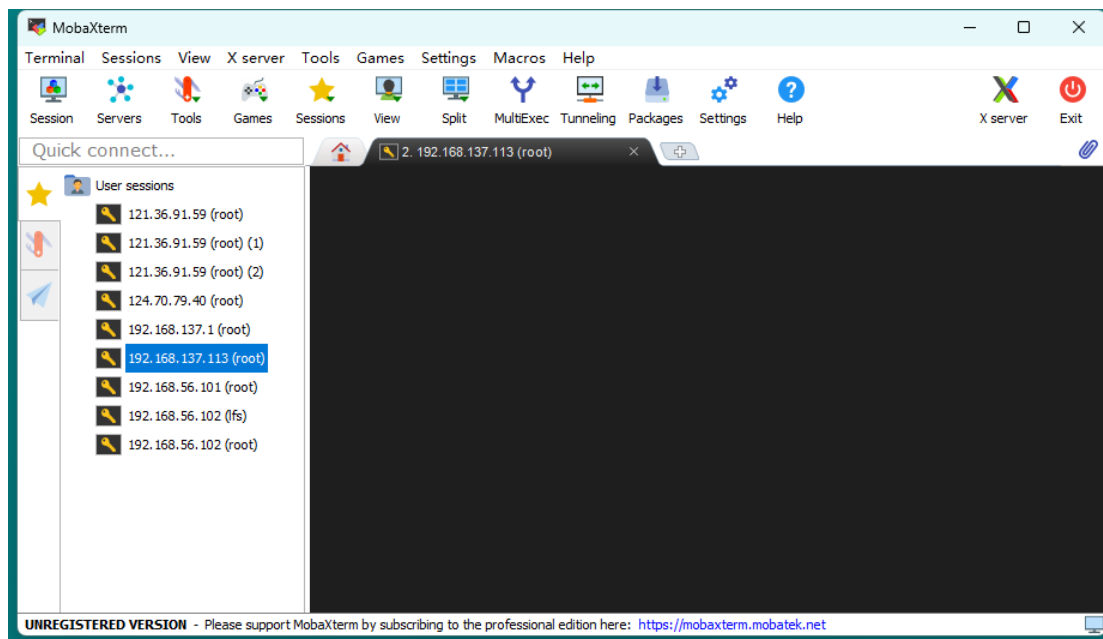


图 1-2 通过 Windows 下的 MobaXTerm 访问开发板

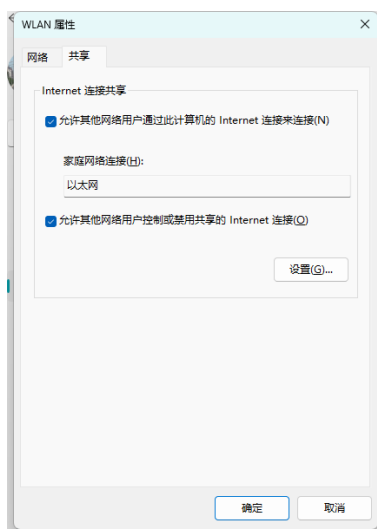


图 1-3 PC 的无线适配器的设置

3. 配置本地编译环境

鉴于 MobaXterm 中集合了 SSH-browser 的功能且香橙派的 Linux 环境支持这一功能，我选择在本次实验及后续的实验中采用本地编译的方式进行——SSH-browser 提供了文件浏览器，对于文件的复制、编辑、创建等操作十分方便，可以直接通过该客户端进行，因此本地编译是一个不错的选择。

我直接在网盘上下载了实验源代码并通过该客户端的文件浏览器上传到了相应的位置，并进行解压。同时修改 common/rules.mk 的编译选项为 `CC:= gcc`，以配置本地的编译环境

4. 编译及链接

在相应的目录下运行如下命令

```
Cd ../lab1
```

```
Make
```

```
Cd ../out
```

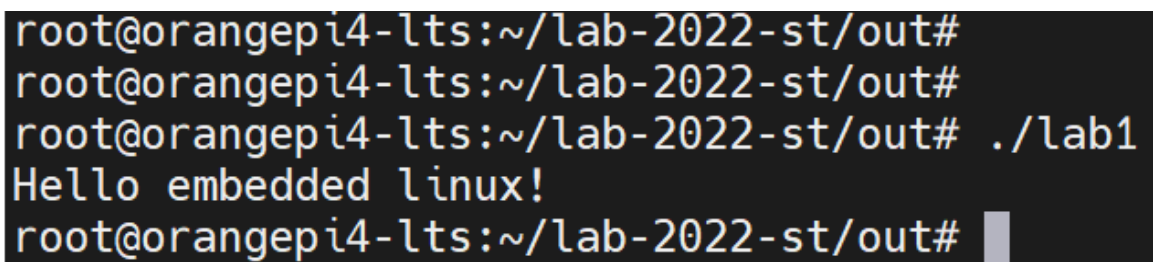
```
./lab1
```

1.3 实验结果

Lab1 的 main.c 代码如下：

```
#include <stdio.h>

int main(int argc, char* argv[])
{
    printf("Hello embedded linux!\n");
    return 0;
}
```



```
root@orange pi4-lts:~/lab-2022-st/out#
root@orange pi4-lts:~/lab-2022-st/out#
root@orange pi4-lts:~/lab-2022-st/out# ./lab1
Hello embedded linux!
root@orange pi4-lts:~/lab-2022-st/out#
```

图 1-4 Lab1 的运行结果

2 实验二 图形界面基础

2.1 实验要求

1. 理解 Linux 下的 LCD 显示驱动接口 framebuffer 的使用原理；
2. 理解双缓冲机制；
3. 理解基本图形的显示原理：点、线以及矩形区域绘制方式，并完成画点函数 `fb_draw_pixel`，画矩形函数 `fb_draw_rect` 以及画线函数 `fb_draw_line`；
4. 在开发板上绘制出所要求的图形，并且记录画图时间。

2.2 实验过程

本次实验是对 `common/graphic.c` 中相应函数的编写。

1. `fb_draw_pixel` 实现

画点函数非常简单，只需添加一行：

```
*(buf + y*SCREEN_WIDTH + x) = color;
```

就能够实现把屏幕上相应的点对应的数组元素赋值成功。

2. `fb_draw_rect` 实现

注意到存储屏幕像素信息的数组是逐行存储，所以采用双层 `for` 循环，以 `y` 为外侧循环，`x` 为内层循环，逐点赋值为 `color`。

3. `fb_draw_line` 实现

检查起始和结束点坐标是否在屏幕范围内，如果不在则返回。

根据起始和结束点的坐标，判断直线的走向（水平、垂直、斜线）。

根据直线走向计算打点距离，选择较大的值。

使用 `Bresenham` 算法，在两点之间打点，绘制直线。

关于 `Bresenham` 算法的一些理解：

该算法是一种用于在离散的网格上绘制直线的算法。该算法主要解决的问题是如何在网格中找到最接近直线路径的像素点，以保持直线在离散空间中的连续性。

算法基于以下观察：对于一条直线上的任意一点 (x, y) ，它与直线的距离（误差）可以通过一个累积的误差项来表示。该算法利用这个误差项来决定每一步在 `x` 或 `y` 方向上的移动，并在每一步上选择最接近实际直线的像素。

华中科技大学课程设计报告

以下是算法的步骤：

- 1) 初始化 dx 和 dy ，它们分别表示两个端点在 x 和 y 方向上的差值。
计算 sx 和 sy ，它们表示 x 和 y 方向上的步进方向（正方向或负方向）。
- 2) 计算初始误差项 $err = dx - dy$ 。
- 3) 在一个循环中，对于每个步骤：
 - a. 在当前点 (x, y) 绘制像素。
 - b. 计算 $e2 = 2 * err$ 。
 - c. 如果 $e2 > -dy$ ，则减小 err 并向 x 方向移动一步。
 - d. 如果 $e2 < dx$ ，则增加 err 并向 y 方向移动一步。
- 4) 重复步骤 3)，直到终点 (x_2, y_2) 被达到。

这样，通过逐步的像素点选择，Bresenham 算法能够以高效的方式在两个给定点之间绘制连续的直线。

4. 编译运行检查

执行如下指令：

```
Cd ../lab2
```

```
Make
```

```
Cd ../out
```

```
./lab2
```

观察屏幕上的图案，若不符合则继续修改。

2.3 实验结果

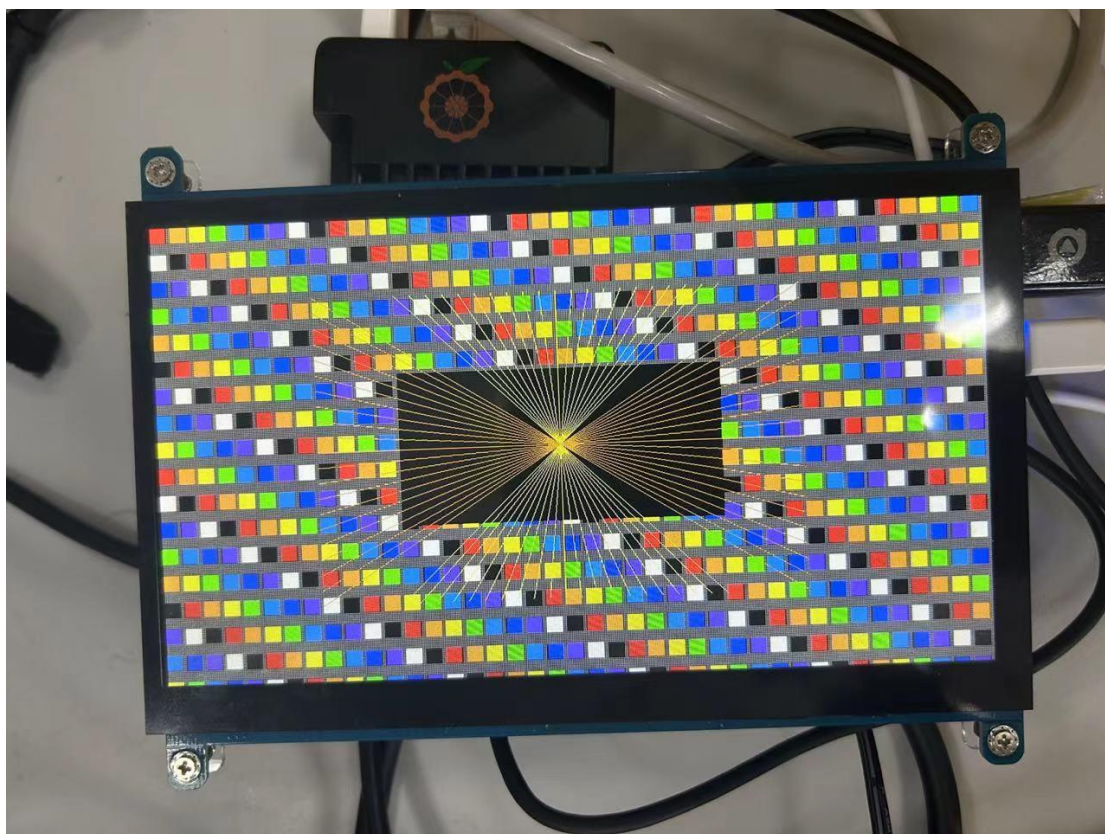


图 2-1 Lab2 实验结果

3 实验三 图片文字显示

3.1 实验要求

1. 实现 jpg 格式不透明图片的显示;
2. 实现 png 格式半透明图片的显示;
3. 实现矢量字体显示, 需要完成字模的提取以及字模的显示。
4. 在尽可能短的时间内实现以上要求

3.2 实验过程

实现 jpg 格式不透明图片、png 格式半透明图片以及矢量字体的显示, 需要编写函数 `fb_draw_image` 对不同类型的图片进行显示。

1. JPG 图片显示

当图片的像素颜色类型为“`FB_COLOR_RGB_8880`”时, 表示显示的图片类型为 jpg 图片。由于 jpg 图片无需处理透明度, 我们可以简单地使用 `memcpy` 函数将图片数据直接复制到缓冲区。

2. PNG 图片显示

对于颜色类型为“`FB_COLOR_RGBA_8888`”的 png 图片, 需要考虑透明度。透明图片中的像素内容从高位到低位分别为 `alpha`、`R1`、`G1` 以及 `B1`。假设将缓冲区 `framebuffer` 的目标地址设置为 `p`, 对于每一个要绘制的像素, 我们使用如下方式对像素进行修正:

```
p[0] += (((B1 - p[0]) * alpha) >> 8);
```

```
p[1] += (((G1 - p[1]) * alpha) >> 8);
```

```
p[2] += (((R1 - p[2]) * alpha) >> 8);
```

3. 矢量字体显示

当图片的像素颜色类型为“`FB_COLOR_ALPHA_8`”时, 表示显示的图片类型为矢量字体。矢量字体只包含透明度, 颜色根据函数的参数设置。在进行颜色赋值时, 需要考虑到 `alpha` 值在 0-255 之间的情况, 以确保文字边缘平滑。这一过程与处理 png 图像类似, 可以借鉴相似的方法。

在完成了 `fb_draw_image` 函数的编写之后, 我对整个 `graphic.c` 文件进行了一些删改, 以优化性能。

部分优化如下:

华中科技大学课程设计报告

1. 使用内存映射 (mmap) 进行 framebuffer 操作：在初始化阶段，通过使用 `mmap` 将 framebuffer 的内存映射到用户空间，避免了频繁的内核空间和用户空间之间的数据拷贝，提高了内存访问效率。

```
void *addr = mmap(NULL, fb_fix.smem_len, PROT_READ|PROT_WRITE,
MAP_SHARED, fd, 0);
```

2. 使用字节操作代替像素级别的操作：在图像绘制中，使用字节级别的操作，而不是逐像素的操作。这可以通过直接操作内存缓冲区来减少函数调用和迭代次数，提高了图像绘制的效率。

```
char *dst = (char *) (buf + y * SCREEN_WIDTH + x);
```

3. 避免重复绘制和无效操作：在绘制之前，进行了一系列的边界检查，避免了越界访问和不必要的绘制操作，减少了不必要的计算和内存访问。例如：

```
if (x < 0 || y < 0 || x >= SCREEN_WIDTH || y >= SCREEN_HEIGHT)
return;
```

除此之外，我也修改了 rules.mk 文件的编译和链接选项，均使用 O3 优化：

```
CFLAGS:=-Wall -O3
LDFLAGS:=-Wall -O3
```

图 3-1 开启 O3 优化

在完成上述内容后，使用 make 命令，并运行 lab3 和 test 程序

3.3 实验结果

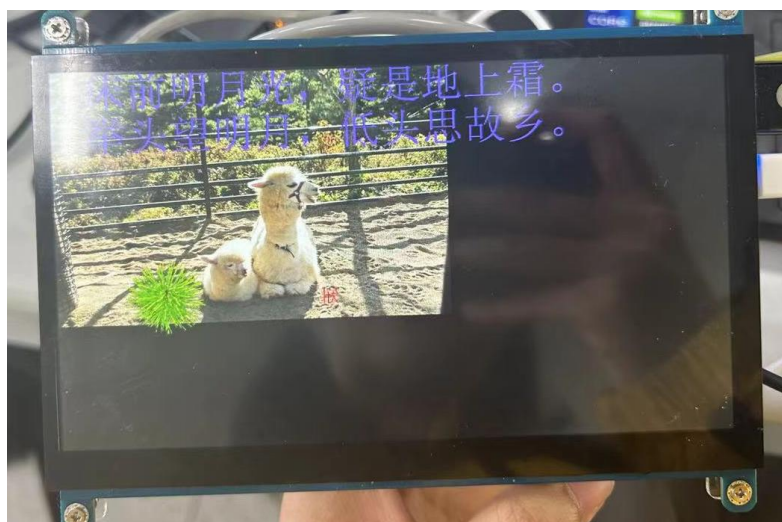
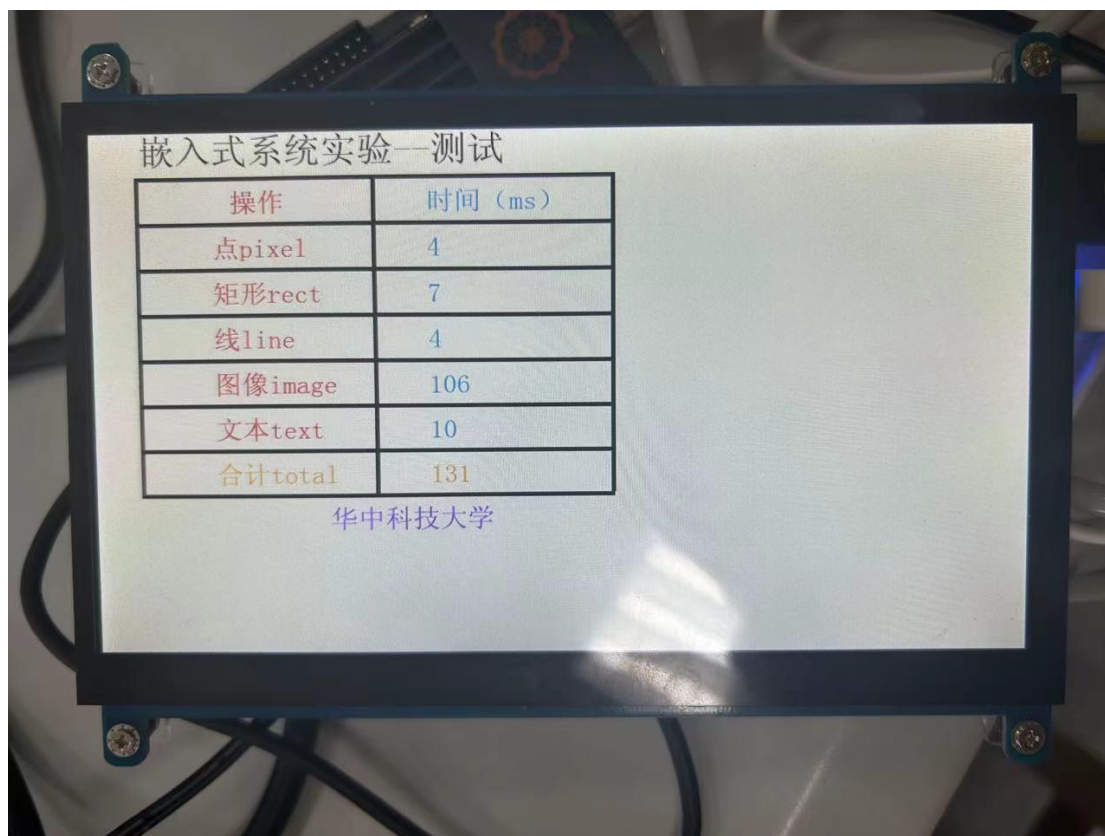


图 3-2 Lab3 运行结果

华中科技大学课程设计报告

优化后运行 test 基本上结果可以稳定在 130ms 左右，如图为 test 运行的结果：



操作	时间 (ms)
点pixel	4
矩形rect	7
线line	4
图像image	106
文本text	10
合计total	131

华中科技大学

图 3-3 test 的运行结果

4 实验四 多点触摸开发

4.1 实验要求

1. 了解 Linux 下的触摸屏驱动接口，学习 Input event 的使用，了解多点触摸协议（Multi-touch Protocol）的工作流程；

2. 使用函数获取多点触摸的坐标，并在 LCD 上显示多点触摸轨迹；

3. 完成基于多点触摸协议的应用程序，选择屏幕绘图或者触点追踪进行应用开发，具体的应用要求如下：

（1）屏幕绘图：使用多个手指在屏幕绘制曲线，要求每个手指轨迹的颜色不同；轨迹是连贯的，不能断断续续；轨迹要比较粗，线宽不能是 1 个点；绘制一个清除屏幕的按钮，点击后清除屏幕内容。

（2）触点追踪：实时跟踪触点，只显示当前位置，要求对应每个手指触点的圆的颜色不同；之前位置的圆要清除掉；屏幕不能明显闪烁。

4.2 实验过程

本次实验选择触点追踪进行开发。

1. 颜色定义：

- 使用 `FB_COLOR` 宏定义了几种常见颜色，如红色、橙色、黄色、绿色、青色、蓝色、紫色和黑色。

- `COLOR_BACKGROUND` 定义了背景颜色，这里是白色。

2. 触摸屏和绘图参数：

- `touch_fd` 用于存储触摸屏设备文件的文件描述符。

- `radius` 数组定义了每个触摸点的初始半径。

3. 触摸事件回调函数：

- `touch_event_cb` 是触摸事件的回调函数，通过调用 `touch_read` 读取触摸事件信息。

- 在 `TOUCH_PRESS` 事件中，根据触摸点的序号选择颜色，然后在屏幕上绘制一个圆形。

- 在 `TOUCH_MOVE` 事件中，先擦除之前的圆形，然后在新位置绘制一个圆形。这样实现了触摸移动的效果。

- 在 `TOUCH_RELEASE` 事件中，擦除之前的圆形。

- ``COLOR_BACKGROUND`` 用于擦除之前的绘制。

5. 性能优化：使用字体绘制圆形：

尽管之前关于圆形的绘制函数也进行了优化，比如用整型代替浮点运算以及用乘方代替开方运算（判断圆的绘制边界），但是效果仍然不尽人意，最终我选择的方法如下：

- 使用 ``font_init`` 初始化字体。

- ``fb_draw_text`` 函数用于在屏幕上绘制文本，这里绘制了一个文本 “

●” 代表触摸点。

4.3 实验结果

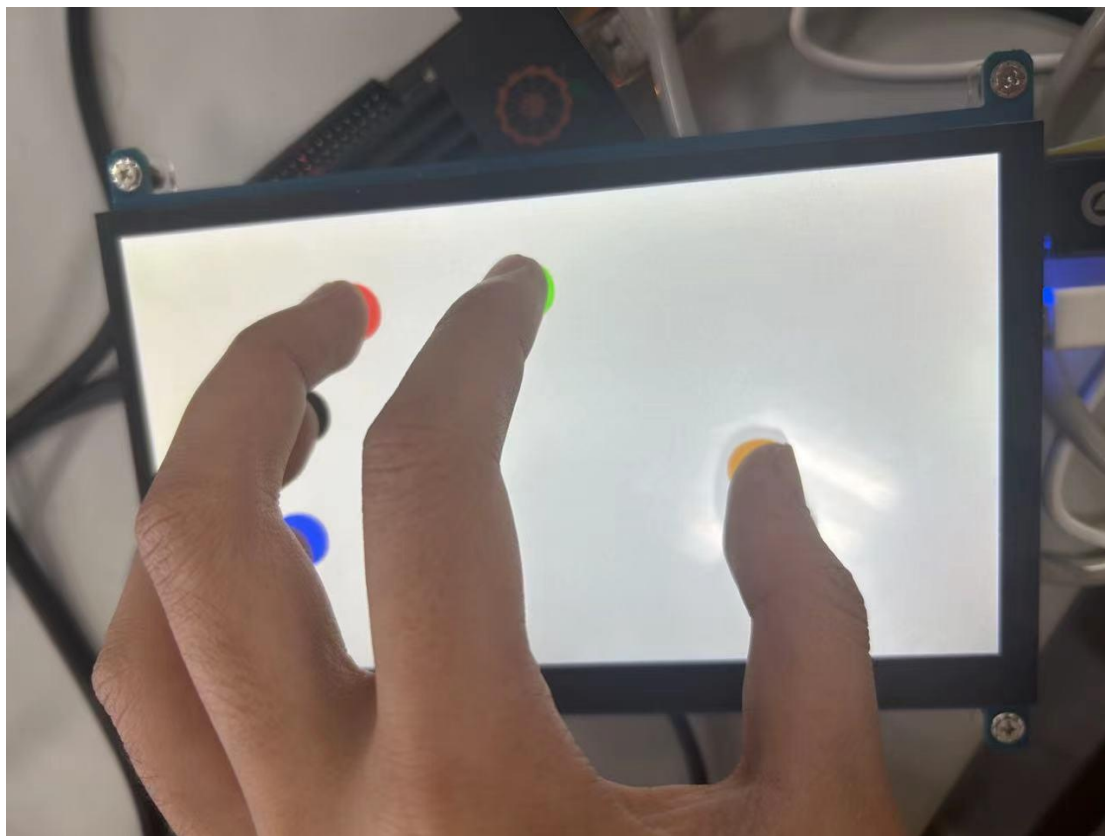


图 4-1 Lab4 实验结果

5 实验五 蓝牙通讯

5.1 实验要求

1. 了解并学习蓝牙无线技术，使用命令正确配置并启动蓝牙服务；
2. 学习并熟练使用蓝牙的常用命令工具如 `hciconfig`、`hcitool` 以及 `bluetoothctl` 等；
3. 通过 RFCOMM 协议（蓝牙串行通讯）实现无线通讯，测试通过 lab5 实验代码。

5.2 实验过程

1. 首先，需要编辑开发板上的配置文件：

```
nano /etc/systemd/system/dbus-org.bluez.service
```

```
ExecStart=/usr/lib/bluetooth/bluetoothd -C
```

```
ExecStartPost=/usr/bin/sdptool add SP
```

```
reboot
```

2. 使用 `bluetoothctl` 扫描蓝牙设备：

```
bluetoothctl
```

```
进入蓝牙 shell 会显示[bluetooth]#
```

```
power on
```

```
discoverable on
```

```
pairable on
```

```
开始扫描周围的蓝牙设备
```

```
scan on
```

扫描到想连接的蓝牙设备后就可以关闭扫描了，然后记下蓝牙设备的 mac 地址

```
scan off
```

扫描到想配对的设备后就可以进行配对了，配对需要使用设备的 mac 地址

```
pair 蓝牙 mac
```

配对成功后，手机蓝牙界面会有所提示（或者通过手机发起配对）

3. 开发板做配置并启动蓝牙服务，运行：

```
rfcomm -r watch 0 1
```

在手机上安装“SPP 蓝牙串口”。

在手机上搜索开发板的蓝牙设备，需要先配对；在手机上进入蓝牙串口程序，进行连接。建立连接后，在开发板上运行 lab5 的测试程序。

5.3 实验结果

蓝牙连接成功之后，运行 lab5 程序可以在开发板屏幕上看到手机 app 发来的消息，点击屏幕上的 send 按钮可以向手机发送“hello”字样。

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，可参考功能如下：

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：如五子棋等小型游戏
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；
5. 程序已经给出 lab6/main.c，同学们在成功编译和运行之后可以考虑如何利用该程序实现一个有图形界面的语音交互的软件或游戏。

6.2 实验过程

本次实验选择设计一个基于蓝牙的五子棋对战小游戏。

1. 宏定义

颜色宏定义：

FB_COLOR(r, g, b): 接受红、绿、蓝三个颜色分量，返回对应的颜色值。

例如：FB_COLOR(255, 0, 0) 表示红色。

背景色和画笔颜色宏定义：

COLOR_BACKGROUND: 背景色，用于清屏。

LINE_COLOR: 画笔颜色，用于绘制棋盘线条。

棋盘状态宏定义：

grid_count: 棋盘的网格数目。

WHITE_SIZE: 留白的大小。

半径和边框宏定义：

R: 棋子的半径。

BORDER: 棋子的边框宽度。

2. draw_chessboard 函数:

绘制五子棋棋盘的网格线。

定义了屏幕上的交点坐标，并通过 `fb_draw_line` 绘制水平和垂直线。

3. draw_menu 函数:

绘制游戏菜单，包括重新开始按钮和历史记录按钮。

`handle_menu_click` 函数处理菜单按钮的点击事件。这个函数首先判断用户点击的坐标是否在重新开始按钮的范围内，如果是，则发送信号给对方，并重新开始游戏。接着，它判断用户是否点击了历史记录按钮，如果是，则在屏幕上显示历史记录（或隐藏历史记录）。

4. 游戏逻辑方面的处理:

play 函数:

在交点附近落子，根据当前玩家绘制黑白棋子。

check_winner 函数:

判断游戏是否有胜者，检查水平、垂直、对角线方向是否有五颗相同颜色的棋子。

check_valid 函数:

检查触摸点是否在交点附近，以确定落子的合法性。

touch_event_cb 函数:

触摸事件回调函数，处理触摸屏幕的操作。

在按下事件中，检查菜单按钮的点击。

如果在交点附近合法落子，则执行 `play` 函数。

5. 处理蓝牙事件

首先尝试从蓝牙设备中非阻塞地读取信息。如果读取成功，表示接收到了信息。接着，函数调用 `play` 函数更新对面的棋局，然后判断是否有胜者。如果有胜者，将在屏幕上显示相应的胜利信息，并更新胜利方的次数。最后，如果对方发送了重新开始的信号（`t[3] == 1`），则调用 `start_game` 函数重新开始游戏。这个函数的目的是在收到蓝牙消息时，根据消息内容执行相应的操作。

于此同时，在 `main` 函数中初始化两个蓝牙设备，并将它们的文件描述符添加到任务循环中。

1) 蓝牙设备 1 初始化:

```
bluetooth_fd1 = bluetooth_tty_init("/dev/rfcomm0");
```

```
if (bluetooth_fd1 == -1) return 0;
```

```
task_add_file(bluetooth_fd1, bluetooth_tty_event_cb);
```

- bluetooth_tty_init("/dev/rfcomm0"): 调用 bluetooth_tty_init 函数打开 /dev/rfcomm0 设备, 返回蓝牙设备的文件描述符 bluetooth_fd1。

- task_add_file(bluetooth_fd1, bluetooth_tty_event_cb);: 将 bluetooth_fd1 添加到任务循环中, 当 bluetooth_fd1 文件可读时, 将调用 bluetooth_tty_event_cb 函数。

2) 蓝牙设备 2 初始化:

```
bluetooth_fd2 = bluetooth_tty_init("/dev/rfcomm1");
```

```
if (bluetooth_fd2 == -1) return 0;
```

- bluetooth_tty_init("/dev/rfcomm1"): 调用 bluetooth_tty_init 函数打开 /dev/rfcomm1 设备, 返回蓝牙设备的文件描述符 bluetooth_fd2。

这些代码确保了两个蓝牙设备的正常初始化, 并将它们的文件描述符添加到任务循环中, 以便在有数据到达时执行相应的回调函数。

6.运行:

在编译通过并完成 debug 后使用两个开发板运行该程序。

1) 根据实验指导手册完成两个开发板的配对

2) 两个开发板分别使用 rfcomm 建立蓝牙 socket 并监听, 且连接对方的信道

开发板 A:

```
rfcomm -r watch 0 1
```

```
rfcomm -r connect 1 xx:xx:xx:xx:xx:xx 2
```

开发板 B:

```
rfcomm -r watch 1 2
```

```
rfcomm -r connect 0 xx:xx:xx:xx:xx:xx 1
```

3) 开发板 A 和 B 分别运行该程序

6.3 实验结果

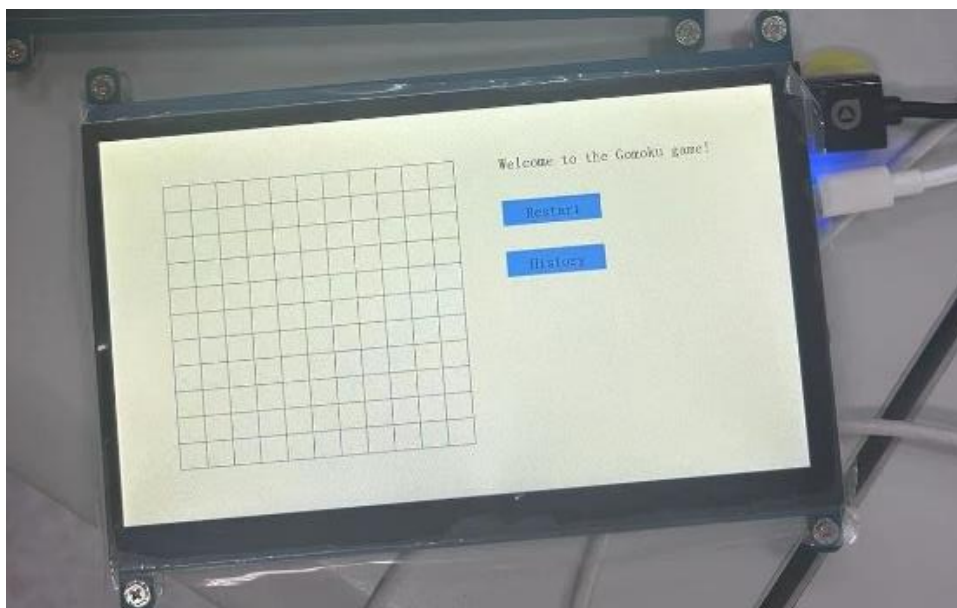


图 6-1 开始游戏

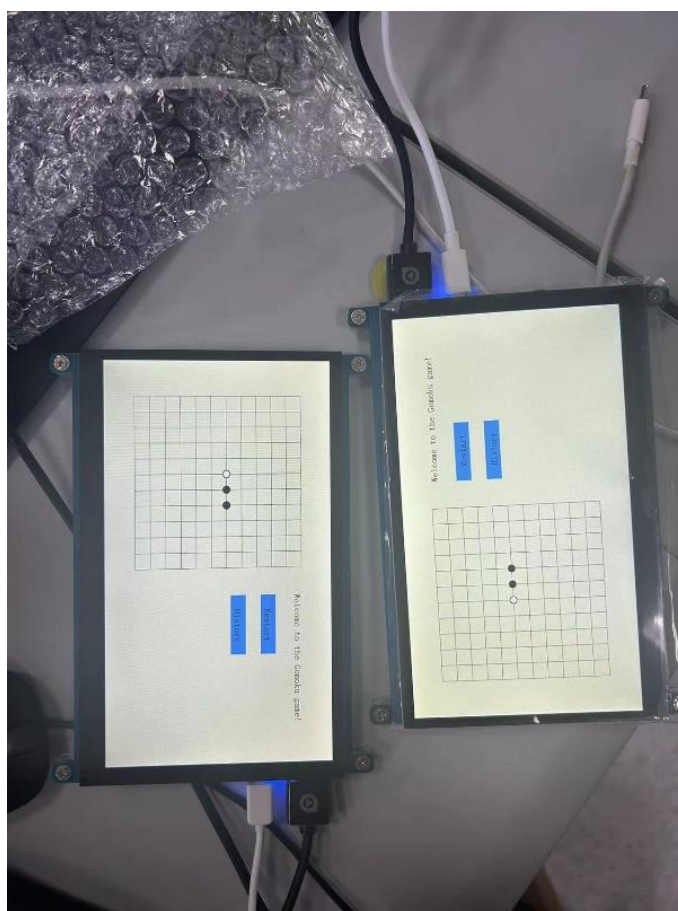


图 6-2 落子

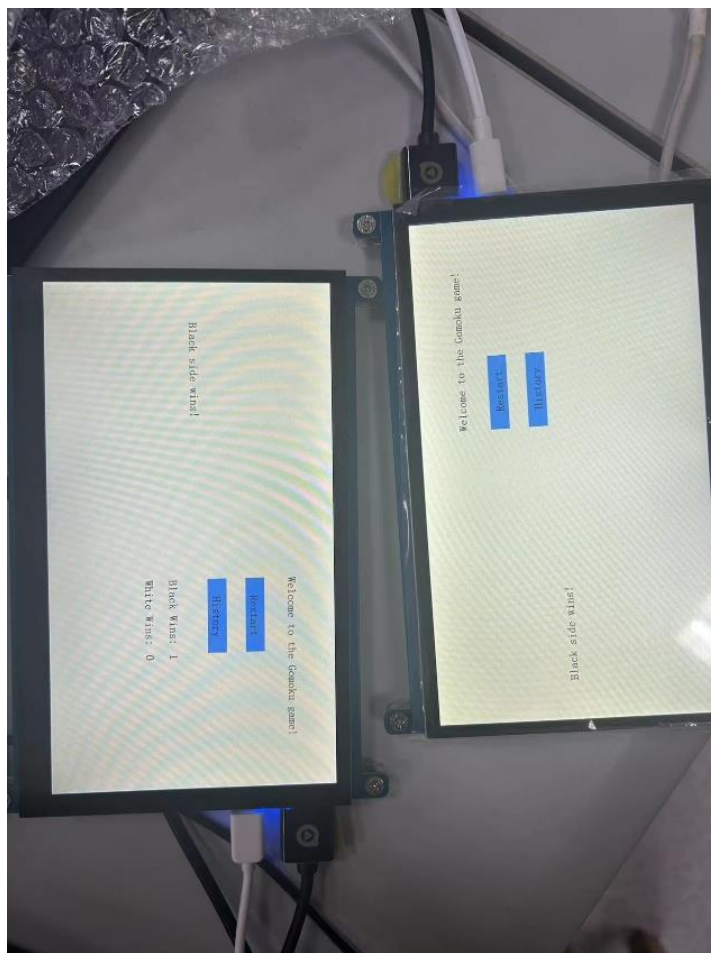


图 6-3 黑方获胜，历史记录中黑方胜利次数+1

7 实验总结与建议

7.1 实验总结

本次课程的实验旨在帮助我们掌握嵌入式系统开发的基本流程、步骤，以及学习嵌入式开发的方式和常用工具。以下是对每个实验的简要总结：

1. 实验 1：内核编译与烧写

主要目标是了解 Linux 系统开发流程，学习使用其他设备将系统烧写至嵌入式设备。这一实验使我们能够将嵌入式设备从裸机状态转变为具有可用性的设备，深入了解嵌入式设备的基本开发方式。

2. 实验 2：LCD 驱动与图像显示

重点在于基于 Linux 下 LCD 驱动接口 `framebuffer` 的图像显示功能开发。通过学习双缓冲机制的工作流程，以及编写画点、画线和画矩形函数，我们进一步熟悉了嵌入式系统开发的流程，掌握了基于 `framebuffer` 的图像显示设计与实现。

3. 实验 3：图片显示

此实验是实验 2 的延伸，要求将图片显示到屏幕上。通过此实验，我们注意到嵌入式设备对段错误非常敏感，因此需要谨慎处理这一问题。

4. 实验 4：多点触摸

学习了多点触摸协议的使用方式，了解了触摸屏驱动接口的应用，并成功实现了触点追踪的功能。这一实验为我们提供了对触摸屏开发的实际经验。

然而，在触控开发中，我留有一点点遗憾。就是我并没有运用到 Bresenham 画圆的算法，希望将来有机会可以用上。后来查阅资料，我了解到 Bresenham 画圆算法是一种经典的画圆方法。该算法通过逐步逼近圆形路径，以递归的方式绘制圆上的像素点，避免了直接计算圆上的点坐标。我想在我的实验总结这里简要记录一下该算法的大体思路：

- 1) 初始化参数： 设置圆心坐标为 (x_0, y_0) ，半径为 r 。
- 2) 计算初始决策参数： 初始化决策参数 d ，一般为 $3 - 2 * r$ 。
- 3) 绘制八分圆： 利用对称性，仅绘制圆的八分之一部分，从第一象限开始。
- 4) 递归绘制： 从圆的第一象限出发，利用对称性逐步绘制其他七个象限。
在第一象限，从 $(0, r)$ 开始，按照决策参数的变化，选择下一个像素点的位置。
根据决策参数的符号和大小，更新决策参数，并选择下一个像素点。

逐步迭代，直到到达 45 度对应的 $x=y$ 为止。

5. 实验 5：蓝牙应用开发

首次接触蓝牙开发，遇到了启动和连接蓝牙设备时的错误和问题。通过一系列的调整，最终成功实现了蓝牙应用的功能，为我们提供了解蓝牙开发的实际经验。

6. 实验 6：蓝牙五子棋对战小游戏

作为综合实验，通过实现一个简单的蓝牙五子棋对战小游戏，综合运用了绘图和蓝牙功能。这个小游戏不仅有趣，还允许查看对战记录和重新开始游戏，展现了实际应用的多方面性。

通过以上实验，我基本掌握了嵌入式系统的基础开发流程，了解了嵌入式系统开发的编译和调试工具，提升了代码编写能力，加深了对理论知识的理解，同时深刻体会到 Linux 在嵌入式系统中的重要地位。这次实验经历对我的职业发展将有着积极的影响。

7.2 实验建议

据我了解，我这次在 Windows 平台下使用 MobaXterm 对开发板进行开发的方法受到了好几个同学的采纳。自认为这个方法是比在 Linux 环境下进行交叉编译方便快捷的，在后续的实验指导手册中或许可以加上这个方法。

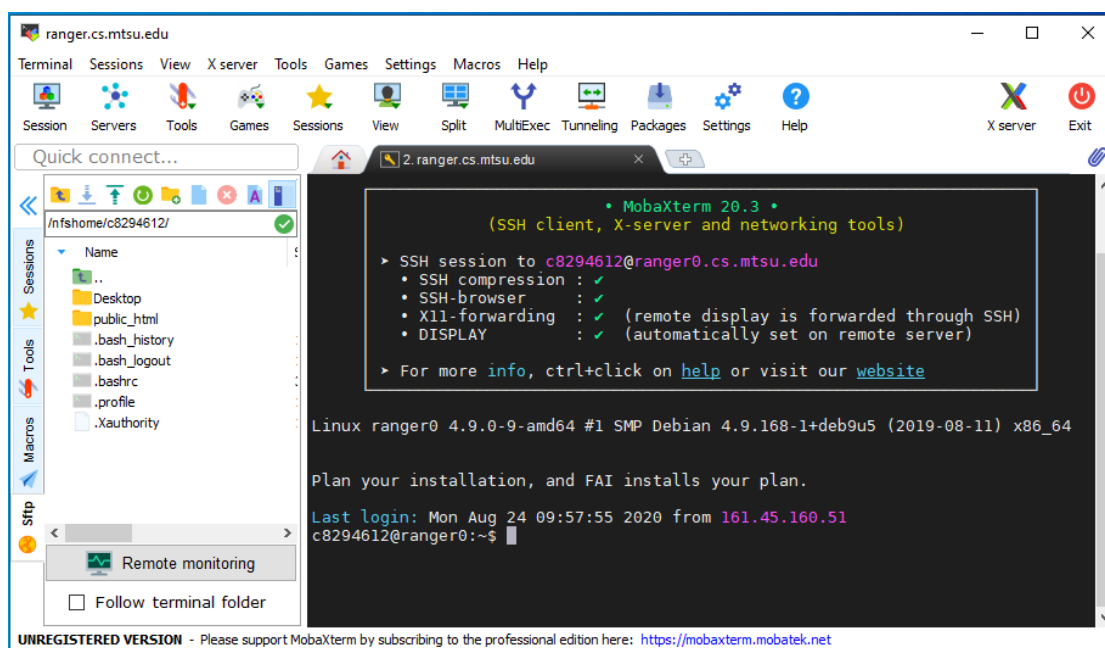


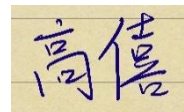
图 7-1 左侧有文件浏览器，右侧为命令行

华中科技大学课程设计报告

原创性声明

本人郑重声明本报告内容，是由本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明的内容外，本报告不包含任何其他个人或集体已经公开的发表的作品成果，不存在剽窃、抄袭行为。

特此声明！

A rectangular stamp with a yellow background containing the handwritten signature '高僊' in blue ink.

作者签字： 高僊