华中科技大学

2025

嵌入式系统 课程实验报告

题　　目：	嵌入式系统实验报告
专　　业：	计算机科学与技术
班　　级：	CS2304
学　　号：	U202315653
姓　　名：	郝依凝
电　　话：	17743017029
邮　　件：	3481963967@qq.com

目 录

1 实验一 系统烧录

1.1 实验要求

烧录香橙派映像，远程登录开发板，配置开发板编译环境，简单 linux 应用程序开发，成功运行 lab1.

1.2 实验过程

1. 开发板与触摸屏连接

HDMI 接口，连接显示屏的 HDMI 接口，用于接收音频及视频信号；电源插口，DC 接口，连接电源；USB 接口，连接显示屏的 touch 接口，用于给显示屏供电以及发送触摸信号。

2. 烧录映像

下载映像压缩文件，解压映像 Orangepi4-lts_3.0.6_ubuntu_desktop_otg.img；将 SD 卡插入读卡器后，连接开发主机，在 Ubuntu 系统下选择映像进行烧录；烧录完成后在开发板断电的情况下插入 SD 卡，连接电源启动开发板。

3. 远程登录开发板

开发板 USB 连上键盘，打开开发板的终端，在开发板终端执行命令连接手机热点并获取开发板 IP；宿主机连接手机热点，然后打开 ubuntu 虚拟机，在终端中使用 ssh 命令进行连接，输入密码 orangepi 实现远程登录。

4. 开发板配置

ssh 登录开发板，禁用开发板桌面；在 ubuntu 虚拟机中下载交叉编译器和实验源代码，采用在开发主机上编辑并生成在 armv8 开发板上运行的程序，然后把程序通过网络拷贝到开发板上运行的配置交叉编译环境方式。

5. 程序运行

在开发主机上进入 lab1 文件夹，运行 make；运行命令 scp -r ./out

root@xx.xx.xx.xx:/root/把 lab1 上传到开发板上运行；ssh 登录到开发板上，进入 /root/out 目录， ./lab1 运行 lab1。

1.3 实验结果

```
Last login: Mon Dec  8 23:30:42 2025 from 192.168.136.102
root@orangepi4-lts:~# cd out
root@orangepi4-lts:~/out# ./lab1
Hello embedded linux!
root@orangepi4-lts:~/out#
```

图 1 lab1 运行结果

2 实验二 图形界面基础

2.1 实验要求

理解 Linux 下的 LCD 显示驱动接口：framebuffer 的使用原理；

实现基本图形的显示：点、线、矩形区域；

理解双缓冲机制。

2.2 实验过程

1. 实现画点函数 void fb_draw_pixel(int x, int y, int color)

实现给指定位置像素点填充颜色即可，即计算像素点位置坐标对应在缓冲区中的位置*(buf + y*SCREEN_WIDTH + x)并赋值为 color。

2. 实现画矩形块函数 void fb_draw_rect(int x, int y, int w, int h, int color)

双重循环实现起始位置为（x，y），宽度为 w 且高度为 h 的矩形的绘制，即以绘制高度为外层循环、绘制宽度为内层循环，计算在矩形范围内的每一个像素点在缓冲区中的位置逐个赋值为 color。

3. 实现画线函数 void fb_draw_line(int x1, int y1, int x2, int y2, int color)

计算线起始位置，即较小 x、y 值作为起始点，计算线高度宽度，初始化缓冲区，即*buf = _begin_draw(x_min, y_min, x_max - x_min + 1, y_max - y_min + 1);

采用 Bresenham 直线算法，只需要进行整数计算。

Bresenham 直线算法实现：

误差项初始化：err = dx - dy;

移动方向初始化：sx = (x1 < x2) ? 1 : -1； sy = (y1 < y2) ? 1 : -1;

单次循环计算像素点位置，直至起点与终点重合：

2err > -dy → 向 x 方向移动，更新误差项 err -= dy

2err < dx → 向 y 方向移动，更新误差项 err += dx

2.3 实验结果

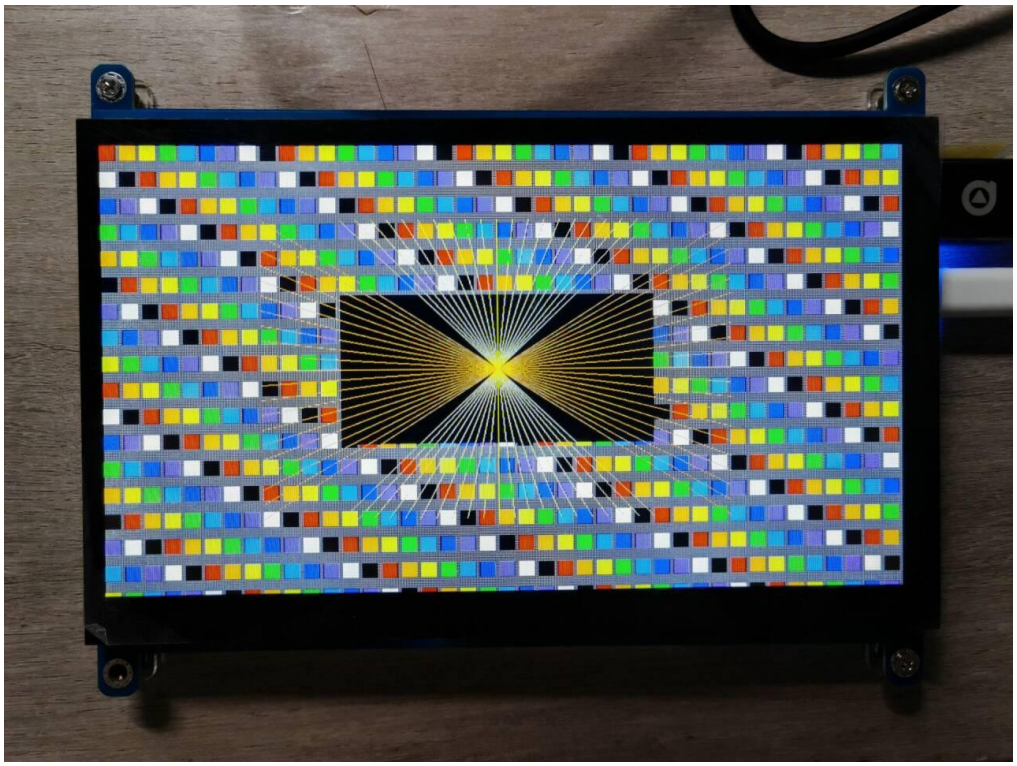


图 2 lab2 运行结果

3 实验三 图片文字显示

3.1 实验要求

实现 jpg 不透明图片显示；

实现 png 半透明图片显示；

实现矢量字体显示：字模的提取，字模的显示（只有 alpha 值的位图）；

优化代码，进行性能测试。

3.2 实验过程

实现画图函数 void fb_draw_image(int x, int y, fb_image *image, int color)，支持显示 jpg、png、font。

1. 不透明 jpg 图片 FB_COLOR_RGB_8880

无需每个像素透明度计算，简单拷贝图片内存到后缓冲即可。

2. 透明 png 图片 FB_COLOR_RGBA_8888

需要逐点计算，并且要单独计算每个颜色通道，因此采用双层循环对每个点进行计算，可得到每个点在缓冲区中起始位置为 pos_color = (char *)(buf + y_ * SCREEN_WIDTH + x_)；每个点在图片内存中的起始位置为 image_color = image->content + iy_ * image->pixel_w * 4 + ix_ * 4。根据透明 png 图片存储格式地址从高到低每四个字节存储内容依次为 alpha、r、g、b 即可提取透明度以及颜色信息。当透明度 alpha=0 时，图片完全透明，由于缓冲区初始内容为 0，因此无需赋值，跳过该点进入下一个即可；当透明度 alpha=255 时，图片完全不透明，只需将对应缓冲区内容写入 rgb 值即可；当图片半透明情况时，对于要绘制的每一个像素，使用如下方式对像素进行修正：

```
pos_color[0] += (((image_color[0] - pos_color[0]) * alpha) >> 8);
pos_color[1] += (((image_color[1] - pos_color[1]) * alpha) >> 8);
pos_color[2] += (((image_color[2] - pos_color[2]) * alpha) >> 8);
```

3. 矢量字体的字模图片 FB_COLOR_ALPHA_8

需要逐点计算，并且要单独计算每个颜色通道，因此采用双层循环对每个点进行计算，可得到每个点在缓冲区中起始位置为 pos_color = (char *)(buf + y_ * SCREEN_WIDTH + x_), 得到每个点透明度为 image_color = image->content + iy_ * image->pixel_w + ix_;。将输入 int color 拆分为 char 类型：

```
c_color[0] = (color & 0xff);
c_color[1] = (color & 0xff00) >> 8;
c_color[2] = (color & 0xff0000) >> 16;
```

对每个像素点处理方式与透明 png 图片相同。

3.3 实验结果

图 3 lab3 运行结果

```
gabriel@LAPTOP-CT8HO90O:~/lab-st$ ssh root@10.221.17.1
========== Start Test ===========

Lab2 test:
draw pixel: 13 ms
draw rect: 18 ms
draw line: 15 ms

Lab3 test:
    **jpeg:     72
    **png:      57
    **font:     3
draw image: 132 ms
draw text: 10 ms

Total time: 188 ms

================================
root@orangepi4-lts:~/out#
```

图 4 test 结果 188ms

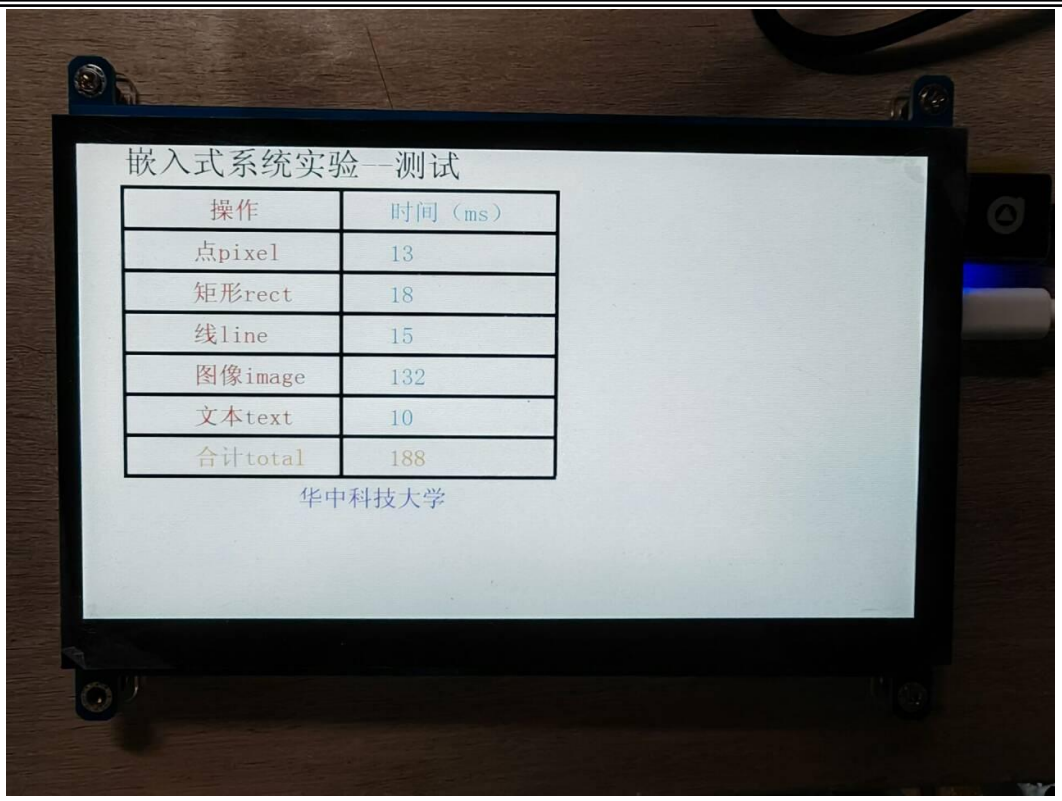

嵌入式系统实验—测试
操作
时间（ms）
点pixel
13
矩形rect
18
线line
15
图像image
132
文本text
10
合计total
188
华中科技大学

图 5 test 运行结果

4 实验四 多点触摸开发

4.1 实验要求

学习并理解 Linux 下的触摸屏驱动接口：Input event 的使用，多点触摸协议(Multi-touch Protocol)；

获取多点触摸的坐标；

实现在 LCD 上显示多点触摸轨迹；

绘制一个清除屏幕的按钮，点击后清除屏幕内容。

实现效果：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。

4.2 实验过程

1. 绘制粗点 void fb_draw_circle_filled(int x, int y, int radius, int color)

接收填充点半径，圆心位置，填充颜色。采用扫描填充算法，即计算横纵坐标边界后，双重循环判断圈定的矩形范围内的点是否在绘制圆区域内，即计算点到圆心距离是否小于等于半径 dx*dx + dy*dy <= radius*radius，若在则填充像素点*(buf + (y_min + j) * SCREEN_WIDTH + (x_min + i)) = color，否则不填充。

2. 绘制粗线段 void fb_draw_thick_line(int x1, int y1, int x2, int y2, int color, int thickness)

接收粗线段起始位置中心，终止位置中心，填充颜色，线条宽度。对于线条主体采用多线条叠加的方法，即利用已经实现的 fb_draw_line 画线函数从 -thickness/2 到 thickness/2 逐条绘制。优化绘制方法，在循环绘制以实现线条平移叠加宽度前，首先计算横纵方向的起止点差，若横坐标差值更大，采用纵向平移叠加 fb_draw_line(x1, y1 + t, x2, y2 + t, color)；否则采用横向平移叠加 fb_draw_line(x1 + t, y1, x2 + t, y2, color)。对于短点绘制实心粗点，其中点的半径为 thickness/2。

3. 绘制清除按钮 void draw_clear_button(void)

选定清除按钮边界，绘制矩形标识清除按钮区域，选定字绘制位置以实现在区域中间绘制矢量字 CLEAR 标识按钮作用。

4. 触摸事件处理逻辑

打开多点触摸设备文件，绘制清除按钮，把后缓冲中绘图区域内容拷贝到前缓冲，添加任务，当 touch_fd 文件可读时，调用 touch_event_cb 函数。

touch_event_cb 函数处理逻辑：读取触摸点、手指标号、触摸事件类型，当类型为 TOUCH_PRESS 时，优先判断是否在清除按钮区，若在则清除屏幕，其中清除屏幕包含将屏幕绘制为白色，重置手指状态，绘制清除按钮，把后缓冲中绘图区域内容拷贝到前缓冲。若不为清除操作，当类型为 TOUCH_PRESS 时，设置手指活跃状态、手指颜色信息、手指位置，绘制粗点即可；当类型为 TOUCH_MOVE 时，读取手指活跃状态，设置手指颜色信息，读取手指上次位置作为本次画线起始点、当前位置作为本次画线终点，调用绘制粗线段函数即可随手指移动绘制线条，移动过程中更新手指位置，即重置下一次绘制线段起始位置；当类型为 TOUCH_RELEASE 时，重置对应手指状态，即设置活跃状态为 0，坐标位置为无效位置（-1，-1）；当类型为 TOUCH_ERROR 时，关闭并调用函数 task_delete_file 删除错误文件。绘制结束调用函数 fb_update 刷新缓冲区即可。

4.3 实验结果

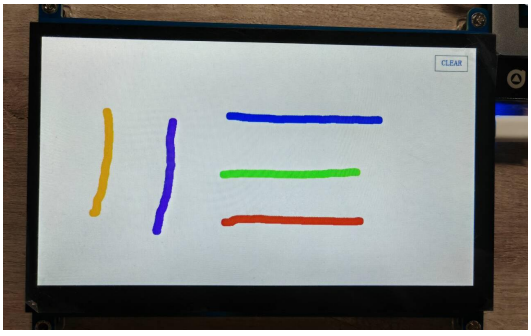


图 6 lab4 运行结果

5 实验五 蓝牙通讯

5.1 实验要求

正确配置并启动蓝牙服务；

熟练使用蓝牙的常用命令工具；

通过 RFCOMM 串口实现无线通讯，测试通过 lab5 实验代码。

5.2 实验过程

1. 配置并启动蓝牙服务

编辑开发板上的配置文件:nano /etc/systemd/system/dbus-org.bluez.service，确定下面启动 bluetoothd 添加了兼容性标志 -C;确认有下面 sdptool add SP 命令行，配置完文件重启开发板 reboot；使用 bluetoothctl 扫描蓝牙设备，扫描到对应设备后使用设备的 mac 地址进行配对。

2. 蓝牙串行通讯实现

一个开发板做服务器，运行 rfcomm -r watch 0 1

另一个开发板做客户端，运行 rfcomm -r connect 0 server_mac 1

3. 蓝牙串行通讯测试

服务端与客户端建立连接后，两个开发板上都运行 lab5 的测试程序进行测试；由于是触摸屏设备，需要将 lab5 中打开多点触摸设备文件修改为 event2，即 touch_init("/dev/input/event2")。

5.3 实验结果

```
root@orangepi4-lts:~/out# rfcomm -r connect 0 D0:A4:6F:CF:B7:68 1 &
[1] 2571
root@orangepi4-lts:~/out# Connected /dev/rfcomm0 to D0:A4:6F:CF:B7:68 on channel 1
Press CTRL-C for hangup
rfcomm -r connect 0 D0:A4:./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 "hello
"
bluetooth tty send hello
bluetooth tty receive "hello
"
bluetooth tty receive "hello
"
bluetooth tty receive "hello
"
bluetooth tty receive "hello
"
bluetooth tty send hello
bluetooth tty send hello
```

图 7 lab5 运行结果

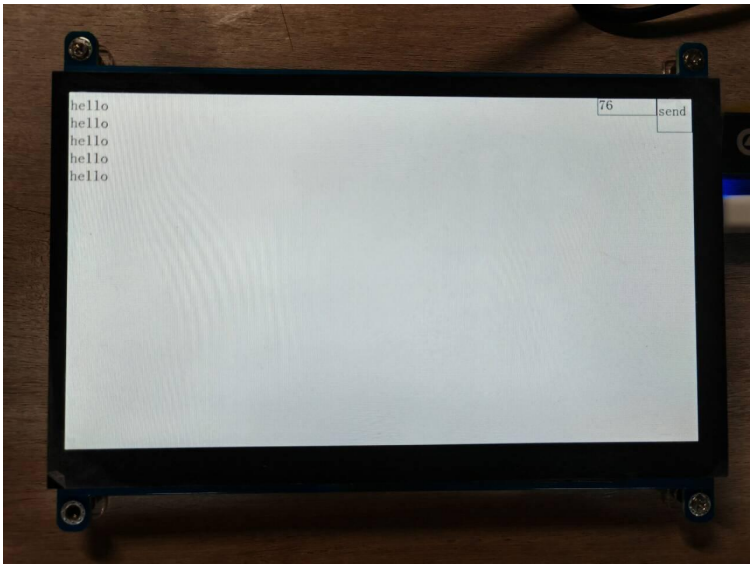


图 8 lab5 运行结果

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. 语音消息：两个开发板之间语音转文字并发送到对方屏幕
4. 联网游戏：比如五子棋等
5. 其他

单机程序:

1. 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.
2. 视频播放器
3. 语音识别程序：语音控制屏幕显示
4. 游戏或其他

6.2 实验过程

1. 游戏设计

游戏以五子棋对战游戏为基础，通过蓝牙实现两台设备之间的对战，支持触摸操作进行落子和技能预选择从而避免误触，设计了 5 种不同技能增强游戏策略性，增加倒计时系统增加游戏紧张感，使用不同的背景图片和界面设计。

Part1.技能系统设计

飞沙走石：可以移除对方一颗棋子；

拾金不昧：可以交换双方指定棋子；

静如止水：增加思考时间 60 秒；

时光倒流：回退一回合；

力拔山兮：清空棋盘（需要能量满 3 个）。

Part2.游戏界面设计

开始界面：显示游戏以及角色信息和准备按钮；

游戏界面：包含棋盘、技能按钮、状态显示（选中技能按钮或棋子位置高亮）；

结束界面：显示获胜信息和重新开始按钮。

2. 功能实现

变量名	类型	说明	初始值
board	int[15][15]	棋盘状态数组	全 0
current_player	int	当前玩家（1=黑，2=白）	1
game_state	int	游戏状态（0=等待，1=游戏中，2=结束）	0
game_over	int	游戏是否结束	0
winner	int	获胜者（0=无，1=黑，2=白）	0
is_my_turn	int	是否轮到己方	0
my_role	int	己方角色（1=黑，2=白）	1
role_initialized	int	角色是否已初始化	0
my_ready	int	己方是否准备好开始	0
opp_ready	int	对方是否准备好开始	0
my_restart	int	己方是否点击重新开始	0
opp_restart	int	对方是否点击重新开始	0
touch_fd	int	触摸屏文件描述符	-1
bluetooth_fd	int	蓝牙文件描述符	-1
skill_used	int[6]	技能使用状态数组	全 0
energy_count	int	能量果实计数	0
energy_fruits	int[5][2]	能量果实位置数组	-1,-1
energy_fruits_placed	int	已放置的能量果实数	0
selected_row/col	int	选中位置	-1
selected_skill	int	选中技能	SKILL_NONE
pick_gold_state	int	拾金不昧状态（0-2）	0
pick_gold_my_row/col	int	拾金不昧己方棋子	-1
pick_gold_opp_row/col	int	拾金不昧对方棋子	-1
history	结构体数组	历史记录（MAX_HISTORY=100）	-
history_count	int	历史记录计数	0

变量名	类型	说明	初始值
time_left	int	剩余时间（秒）	30
timer_period	int	定时器周期（毫秒）	1000

表 1 全局变量表

Part1.程序初始化 main

初始化帧缓冲设备，初始化字体系统，初始化游戏状态变量，初始化随机数种子（便于后续随机生成位置），初始化触摸屏，初始化蓝牙，添加定时器，进入主事件循环。

Part2.游戏界面绘制 static void draw_board(void)

判断游戏状态：等待开始状态，游戏中状态，游戏结束状态。

等待开始：绘制游戏背景图片，绘制游戏标题、角色信息、准备状态、连接状态，绘制开始按钮；

游戏中：绘制游戏背景图片、棋盘网格、棋子，绘制技能按钮、确定按钮，显示状态信息、倒计时信息、按钮状态、棋子状态；

游戏结束：绘制胜利方背景图片，绘制重新开始按钮，显示胜利方信息、游戏结束信息。

函数名	参数	返回值	功能描述
scale_image_to_screen	fb_image *src_img	fb_image*	缩放图片以适应屏幕大小
draw_thick_line	x1, y1, x2, y2, color, thickness	void	绘制加粗线条（用于棋盘网格）
screen_to_board	x, y, *row, *col	int	将屏幕坐标转换为棋盘坐标
check_button_click	x, y, btn_x, btn_y, btn_w, btn_h	int	检查是否在按钮区域内

表 2 界面绘制函数

Part3.触摸事件处理 static void touch_event_cb(int fd)

由于五子棋通过点击形式进行交互，因此主要处理并详述触摸事件类型

TOUCH_PRESS 逻辑。

当触摸事件类型为 TOUCH_PRESS，判断游戏状态。当前状态为开始前，若游戏初始化完成，点击开始按钮，发送 READY 信息。当前状态为游戏中，检查点击位置，点击位置为确定按钮，检查是否有选择技能，若有执行技能，否则执行落子；点击位置为技能按钮，检查技能是否可用，技能不可用则不做任何操作，技能可用则更改是否使用技能状态，发送 MOVE 信息；点击位置为棋盘区域，检查当前是否选中技能，若选中技能“拾金不昧”则选中交换的双方棋子，其中若已经选中一颗己方棋子，再次选中己方棋子则更新选中的己方棋子，若选中对方棋子则设置或更新选中的对方棋子，若选中技能“飞沙走石”则选中对方一颗棋子，其余情况为选择落子位置。当前状态为游戏结束，点击重新开始按钮，发送 RESTART 信息，等待再次开始游戏。

Part4.蓝牙消息处理 static void bluetooth_tty_event_cb(int fd)

接收 INIT：解析对手角色，确定己方角色，设置角色初始化标志，如果是后手，等待接收 SEED 消息，重绘界面；

接收 READY：设置对手准备标志，检查双方是否都准备好，都准备好则进入游戏状态，重绘界面；

接收 MOVE：解析行和列，在棋盘上放置对手的棋子，切换当前玩家到己方，重置倒计时，重绘界面；

接收 SEED： 解析随机种子，使用相同种子放置能量果实，重绘界面；

接收 SKILL：解析技能类型和参数，执行对应的技能效果，重绘界面；

接收 RESTART：设置对手重新开始标志， 检查双方是否都准备好重新开始，都准备好则重置游戏状态，重绘界面；

接收 RESET：完全重置游戏状态，重新放置能量果实，重绘界面。

消息类型	格式	发送方	接收方处理函数	说明
INIT	INIT <role>	先手方	bluetooth_tty_event_cb	初始化角色
READY	READY	点击准备按钮方	bluetooth_tty_event_cb	准备开始游戏
MOVE	MOVE <row> <col>	落子方	bluetooth_tty_event_cb	棋子移动
SEED	SEED <seed>	先手方	bluetooth_tty_event_cb	同步能量果实种子
SKILL	SKILL <type> [params]	使用技能方	bluetooth_tty_event_cb	技能使用通知
RESTART	RESTART	点击重新开始方	bluetooth_tty_event_cb	重新开始游戏
RESET	RESET	系统或手动发送	bluetooth_tty_event_cb	重置游戏状态

表 3 消息协议表

Part5.技能处理逻辑 static void execute_skill(int skill)

对五个附加技能进行处理以按预想效果实现。

其中为实现收集能量使用“力拔山兮”技能，使用 place_energy_fruits 函数随机选择五个位置放置能量果实。

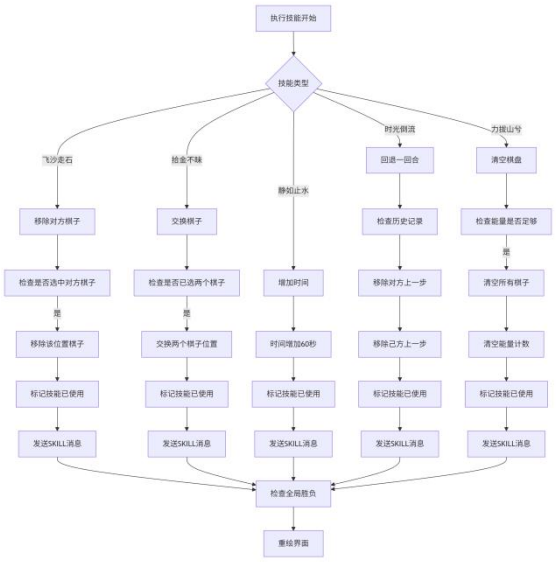


图 9 技能处理逻辑

Part6.定时器处理 static void timer_cb(int period)

检查是否在游戏中且轮到己方，任意条件不满足即不作操作；减少剩余时间，若时间到自动随机落子，即随机生成位置并尝试找到空位置，若成功找到则发送

MOVE 并重置计时器进入下一回合，重绘界面更新倒计时显示。

Part7.落子判定 static int place_stone(int row, int col, int player)

检查位置有效性；检查位置是否为空；检查是否是能量果实，其中只有己方获得能量果实时才增加能量条，且落子位置为能量果实位置时需要将此位置坐标从能量果实数组中移除；放置棋子，即维护全局变量棋盘状态数组 board；记录历史，即维护全局变量历史结构体数组 history；每轮落子结束需要调用函数 check_win 检查是否获胜，若获胜更新全局变量维护游戏状态。

Part8.胜负判定 static int check_win(int row, int col, int player)

每次落子后，只检查当前落子位置是否形成了五子连珠，不需要遍历整个棋盘，只检查与当前落子位置相关的四个方向，每个方向都进行双向检查（正向和反向），实现了每次落子只检查局部，提升胜负判定效率，具体代码如下：

```
static int check_win(int row, int col, int player)
{
    int directions[4][2] = {{0, 1}, {1, 0}, {1, 1}, {1, -1}};
    int i, j;

    for (i = 0; i < 4; i++) {
        int count = 1;
        int dx = directions[i][0];
        int dy = directions[i][1];

        for (j = 1; j < 5; j++) {
            int new_row = row + dy * j;
            int new_col = col + dx * j;
            if (new_row >= 0 && new_row < BOARD_SIZE &&
                new_col >= 0 && new_col < BOARD_SIZE &&
                board[new_row][new_col] == player) {
                count++;
            } else {
                break;
            }
        }
```

```
        for (j = 1; j < 5; j++) {
            int new_row = row - dy * j;
            int new_col = col - dx * j;
            if (new_row >= 0 && new_row < BOARD_SIZE &&
                new_col >= 0 && new_col < BOARD_SIZE &&
                board[new_row][new_col] == player) {
                count++;
            } else {
                break;
            }
        }
        if (count >= 5) {
            return 1;
        }
    }
    return 0;
}
```

由于加入技能后每次使用技能会对棋子排布产生影响，因此设置函数 check_global_win 进行对于整个棋盘的全局扫描判定是否存在获胜方，此函数只在使用技能后进行判定，由于每个技能双方仅有一次使用机会，因此不影响整体判定效率。

6.3 实验结果

```
[2]+  Done                    rfcomm -r connect 0 D0:A4:6F:CF:B7:68 1
root@orangepi4-lts:~/out# ./gomoku_game
framebuffer info: bits_per_pixel=32,size=(1024,600),virtual_pos_size=(0,0)(1024,600),line_length=4096,smem_len=2457600
Bluetooth connected
```

图 10 lab6 测试

图 11 启动页测试

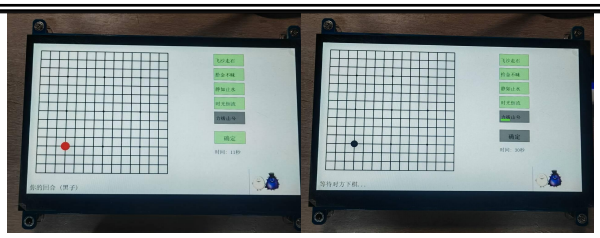
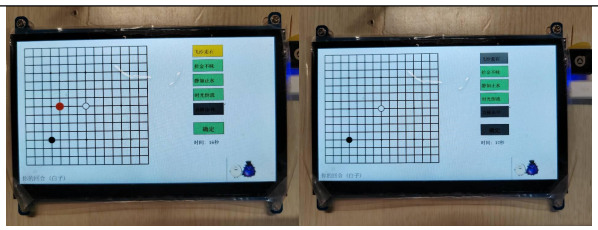
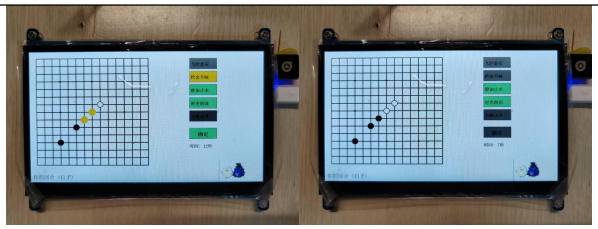
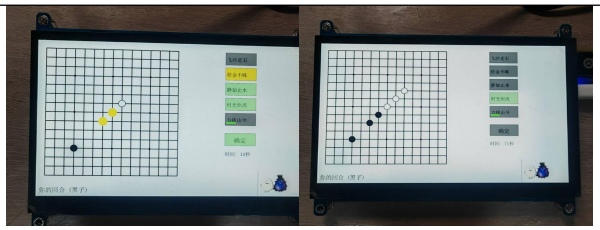
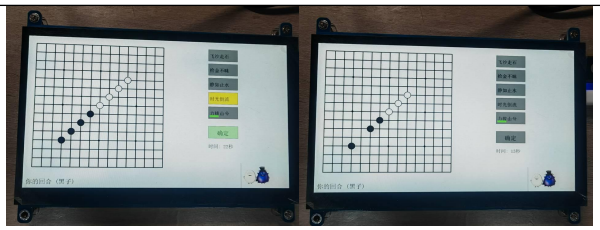
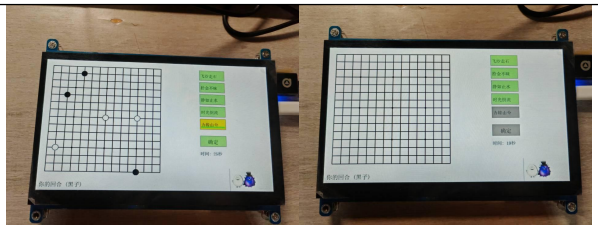
	**落子测试** 可预选位置，预选位置标识为红色 当按下确定后即选定位置并落子 避免操作失误
	**技能测试：飞沙走石** 可预选位置，预选位置标识为红色 当按下确定后即选定并将指定棋子删除 避免操作失误
	**技能测试：拾金不昧** 可预选双方棋子，预选位置标识为黄色 当按下确定后即选定并将指定棋子换位 避免操作失误
	**技能测试：静如止水** 使用后可延长本轮时间 1min 测试倒计时增加
	**技能测试：时光倒流** 使用后双方棋子均回退一轮 测试使用前后棋子状态 测试结果与预期相符
	**技能测试：力拔山兮** 需要积累能量条后使用 使用后清空棋盘 测试使用条件与效果与预期相符

表 4 游戏内容测试

	
白子胜利	**黑子胜利**

表 5 结束页测试

7 实验总结与建议

7.1 实验总结

通过本次嵌入式系统实验课程的学习与实践，我全面掌握了嵌入式系统开发的基本流程和关键技术以及从硬件连接到软件开发的完整流程。通过六个循序渐进的实验，我掌握了嵌入式系统从系统烧录、交叉编译环境配置到远程调试的完整搭建，深入理解 framebuffer 工作原理，学习了像素级图形绘制算法，实现触摸屏、蓝牙、语音识别等多种交互方式，掌握了蓝牙串口通信和简单的协议设计。

本次综合实验成功实现了基于蓝牙通讯的双人对战五子棋游戏。在界面绘制方面，根据不同游戏状态准确绘制了各类界面元素，确保了游戏界面的清晰与美观，各绘制函数功能稳定可靠，为游戏的流畅运行提供了基础保障。触摸事件处理逻辑完整，能够精准识别不同的触摸位置和游戏状态，并做出相应的正确操作，开始游戏、落子、使用技能、重新开始游戏等操作均准确响应，有效保障游戏的交互性和用户体验。

蓝牙消息处理机制有效地实现了两个开发板之间的信息交互，各种消息类型的解析和处理准确无误，确保了游戏双方能够同步进行游戏进程，保证了游戏的公平性和连贯性。技能处理逻辑为游戏增添了丰富的策略性和趣味性，五个附加技能实现效果符合预期，通过合理的技能设计和处理，使游戏玩法更加多样化。

定时器处理和落子判定功能稳定，能够准确计时并在时间结束时进行自动随机落子，胜负判定逻辑经过优化使得每次落子后只检查局部位置，提高了判定效率，同时设置的全局扫描判定函数也确保了在使用技能后能准确判断获胜方。

在游戏测试环节，各项测试内容均达到了预期效果。落子测试、技能测试以及胜负判定测试等结果都表明游戏功能完整且稳定，能够为玩家提供良好的游戏体验。然而，实验过程中也发现了一些可以改进的地方：在 UI 设计上可以进一步优化，增加动画效果和音效，提升游戏的视觉和听觉体验；在技能平衡性方面，可以对部分技能的效果和使用条件进行微调，使游戏更加公平公正。

7.2 实验建议

指导 PPT 内容清楚丰富，但对于开发板与手机连接进行蓝牙串行通讯测试部分，由于手机配置存在差异，需要更加详细的文档资料链接保证不同设备的兼容性以顺利完成测试，谢谢老师们的辛苦付出。