中国地质大学（武汉）自动化学院

嵌入式系统技术实习报告

课　　程：嵌入式系统技术实习

学　　号：20221003169

20221000983

班　　级：231223

姓　　名：董佳颖、陈子逸

指导老师：刘玮、张莉君、彭健

陆承达、张盼

二〇二四年十二月十二日

目 录

第一部分：智趣多媒体娱乐系统设计及实现---------- 1
1. 系统功能介绍----------1
1.1 系统基础功能----------1
1.2 多媒体功能----------1
1.3 创意工具功能----------2
1.4 游戏功能----------3
2. 图像显示----------3
2.1 图像数据存储----------3
2.2 显示函数调用----------4
3. 用户输入获取----------4
3.1 触摸屏输入----------4
3.2 输入处理----------5
4. 系统基础功能的实现---------- 5
4.1 开机动画与音效逻辑---------- 5
4.2 综合功能切换----------5
4.3 实时时钟显示逻辑---------- 6
4.4 基础功能综合流程图---------- 6
5. 相册功能的实现---------- 6
5.1 图片存储结构----------6
5.2 图片切换与显示---------- 7
5.3 图片切换逻辑----------7
5.4 视频播放逻辑----------8
5.5 程序流程图----------8
6. 音乐播放功能的实现---------- 8
6.1 音乐存储结构----------8
6.2 音乐播放控制与切换逻辑----------9
6.3 音量控制----------9
6.4 程序流程图---------- 10
7. 电子书阅读功能的实现----------10
7.1 字库存储结构---------- 11
7.2 书籍/页面显示与切换逻辑---------- 11
7.3 程序流程图---------- 11
8. 绘图功能的实现---------- 12
8.1 区域选择检测与绘制----------12
8.2 清屏与退出操作处理----------12
8.3 圆形填充函数逻辑----------12
8.3 程序流程图---------- 12
9. 灯光控制功能的实现----------13
9.1 硬件初始化与背景绘制---------- 13
9.2 触摸检测与亮灯操作---------- 13
9.3 程序流程图----------13
10. 贪吃蛇游戏功能的实现---------- 14

10.1 蛇的位置和方向---------- 14
10.2 游戏状态更新---------- 14
10.3 游戏画面绘制与显示---------- 14
10.4 游戏循环与流程控制---------- 15
10.5 程序流程图----------15
11. 音乐游戏功能的实现---------- 16
11.1 整体架构----------16
11.2 画面更新----------16
11.3 游戏物体参量变化---------- 16
11.4 得分逻辑---------- 17
11.5 音乐与游戏触发结合----------17
11.6 程序实现流程---------- 18
11.7 游戏设计流程图---------- 19
11.8 游戏实现效果---------- 19
12. 总结---------- 19
13. 心得体会----------20
第二部分：Linux 系统体验实习----------22
1. 对 Linux 系统的理解---------- 22
2. 对移植的理解---------- 23
3. 驱动的开发---------- 23
3.1 实现驱动移植----------23
3.2 编写驱动程序---------- 24
4. 基于 Linux 系统的应用开发与裸机应用开发的区别---------- 24
4.1 裸机应用开发----------24
4.2 基于 Linux 的应用开发----------24
4.3 硬启动与软启动的区别---------- 25
5. 总结----------25
第三部分：实习意见和建议----------25

第一部分：智趣多媒体娱乐系统设计及实现

1. 系统功能介绍

本智趣多媒体娱乐系统功能独特丰富，涵盖系统基础、多媒体、创意工具及游戏等多方面。基础功能确保操作便捷与时间掌控；多媒体功能提供优质视听体验；创意与游戏功能增添趣味与挑战，旨在为用户打造全方位沉浸式娱乐体验。

1.1 系统基础功能

（1）开机动画功能：营造出一种具有动态视觉效果和音频配合的开机展示过程。完成开机初始化流程后进入系统主界面，为用户开机使用系统增添了一份趣味性和仪式感。如图 1.1 所示。

（2）模式切换功能：在系统主界面，通过触屏操作，依据触屏坐标处于不同范围来切换进入相册、音乐播放、画图、时钟显示、阅读、游戏、灯光控制等不同功能模块，实现综合性的功能调度。

（3）实时时钟功能：能够实时地将系统的 RTC 时间展示在屏幕的左下方区域，并提供触屏操作来决定何时停止显示时间信息。如图 1.2 所示。

图 1.1 系统开机动画

图 1.2 系统主界面与实时时钟显示

1.2 多媒体功能

（1）相册功能：提供了图片与视频浏览体验。进入后显示相册和视频分类界面，用户触屏选择进入。相册内置缩略图，点击屏幕中间区域能够切换其显示状态，隐藏时显示全屏图，显示时点击缩略图可切换全屏图。视频功能则通过循环展示简易的模拟动画，点击特定退出区域可返回分类界面。如图 1.3 所示。

（2）音乐播放功能：能够对音乐播放过程全面交互控制。在使用时，用户能够借助触屏点击屏幕不同区域完成诸如切换前后歌曲、暂停播放与音量控制操作，也可以通过按键来调节音量大小，带来丰富的音乐聆听体验。如图 1.4 所示。

（3）电子书阅读功能：书架及电子书阅读功能为用户提供了便捷的阅读体验。书架界面展示精美的书籍封面，用户触屏点击特定书籍区域，即可进入该书

籍的阅读，并能通过触屏左右区域实现翻页与返回上级菜单操作。如图 1.5 所示。

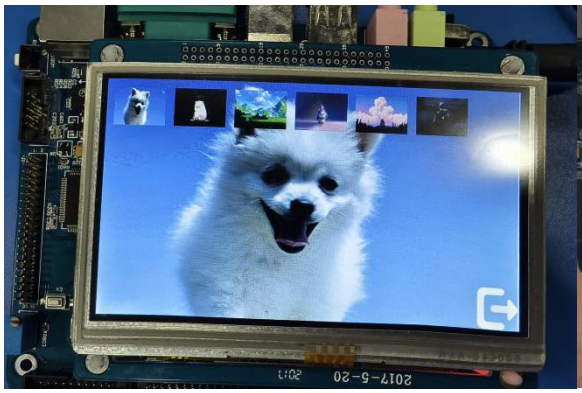

图 1.3 系统相册

图 1.4 系统音乐播放器

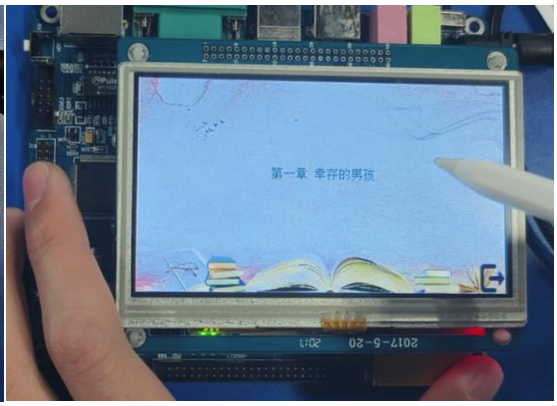

图 1.5 系统电子书书架及内页

1.3 创意工具功能

（1）绘图功能：提供了多种绘图方式，如画直线、画矩形（包括填充矩形）、画圆等，用户能在操作界面选择绘图颜色，并通过触屏操作在屏幕相应区域进行绘图创作，还支持清屏以及退出绘图操作。如图 1.6 所示。

（2）灯光控制功能：可实现对 LED 灯的多样化显示控制。具备点触区域亮灯功能，模拟对单个灯具的控制，用户点击相应区域，对应的灯亮起；还包括流水灯模式实现有序的灯光变化。可满足不同场景下对灯光的需求。如图 1.7 所示。

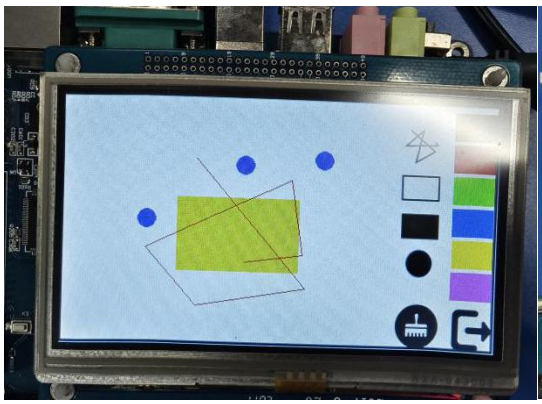

图 1.6 系统绘图界面

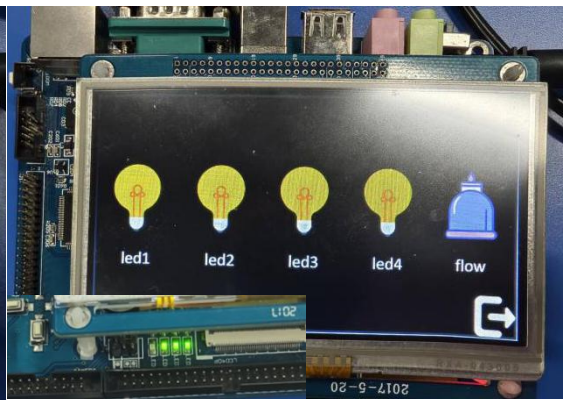

图 1.7 系统灯光控制界面

1.4 游戏功能

（1）音乐游戏：为玩家带来音乐与游戏操作融合的独特体验。依据音符出现的时间和位置，玩家跟随游戏音乐节奏点通过触屏操作判定点击音符，根据击中情况计算分数与切换人物形态，随着游戏进行不断更新音符状态并实时展示游戏画面，结束后显示总得分等信息。如图 1.8 所示。

（2）贪吃蛇游戏：作为一款经典游戏，玩家可通过触屏操作控制蛇的移动方向，贪吃蛇会自动移动并判断是否吃到食物，吃到食物后蛇身增长、得分增加且游戏速度加快难度提升，碰到边界或自身则游戏结束并展示得分情况。如图 1.9 所示。

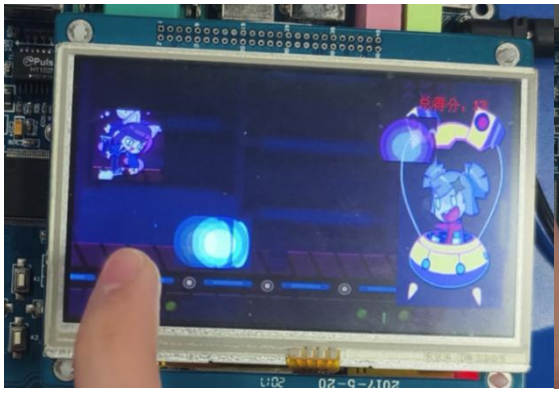

图 1.6 系统音乐游戏

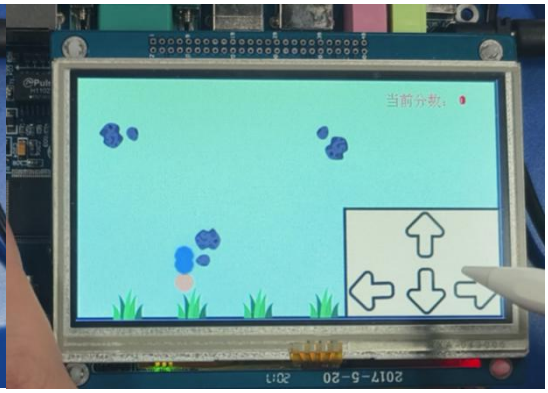

图 1.7 系统贪吃蛇游戏

2. 图像显示

整个系统设计围绕图像显示展开，在该方面有多个具体应用部分。

2.1 图像数据存储

当需要在系统中显示自定义图像时，需要进行一系列操作来实现图像数据的正确存储。首先在画图软件中将图片像素转换为宽高 480×272，格式为 24 位真彩色位图 bmp，接着使用特定的“bmp 转换工具”完成图像文件到数组的转换，将图像转换为可供显示的数组形式，其中，数据按照一定的顺序依次表示图片中各个像素的颜色值，每个像素的颜色由 R、G、B 三个颜色通道的值共同决定。转换完成后，会在图像同目录下生成×××.h 和×××.c 文件，×××.c 文件文件在使用前需要进行一些必要的修改。

具体修改内容包括：

①将头文件“base.h”修改为“LCD_TFT.h”，为了使其与系统的 LCD 显示相关头文件设置保持一致；

②加入宏定义“#define WIN32”，以满足系统环境的要求；

③删除 ALIGN4 const，因为在当前的使用场景下不需要这种对齐方式；

④删除 image header 部分内容，LCD 显示这些数组不需要图像文件头信息。

通过这些修改，使得自定义图像数据能够正确地存储并适配系统的显示要求。对于存储的图像数据，最终是通过 LCD_BUFFER 数组来进行管理和使用的。例如，Paint_Bmp 函数会将转换后的图像数据写入到 LCD_BUFFER 数组中，从而实现图像在屏幕上的显示。

通过特定的语句在 Main.c 文件起始处声明存储图片的数组，形式为：

```
extern unsigned char ×××[]
```

接着调用 Paint_Bmp 函数并传入图片数组文件来即可显示自定义图片。

2.2 显示函数调用

系统通过特定的显示函数 Paint_Bmp 来实现图像在 LCD 屏幕上的显示，它能够根据提供的参数在指定的坐标位置显示相应图像。根据 Paint_Bmp 函数定义可看到，函数通过循环遍历图像数据数组 bmp，并将每个像素的颜色值赋给 LCD_BUFFER 数组中对应的位置，从而实现图像的显示。

例如 Paint_Bmp(0, 0, 480, 272, picture)这样的函数调用，它实现在指定坐标点(0,0)位置上显示宽度为 480 像素，高度为 272 像素的图像，其位图数据存放在数组 picture[]内。该函数的实现原理是通过往 LCD_BUFFER[x][y]不断赋值来实现图像显示。具体来说，它是一个双层嵌套的循逐行扫描的过程，对 LCD 的每个像素点进行写入不同像素点不同颜色值的操作。

还有其他一些辅助函数也参与了图像显示过程。例如，PutPixel 函数用于在指定的坐标位置设置单个像素的颜色值，它被 Paint_Bmp 函数以及其他一些绘制图形的函数（如 Glib_Line、Glib_Rectangle 等）所调用；Lcd_ClearScr 函数用于全屏填充特定颜色单元或清屏。它们通过循环遍历 LCD_BUFFER 数组并将所有元素设置为指定的颜色值来实现画线、清屏或填充矩形的功能。

3. 用户输入获取

触摸屏是实现系统多种功能的关键输入方式。本部分将展开介绍。

3.1 触摸屏输入

开发板采用四线电阻式触摸屏，其具有 4 层透明复合结构。外面两层为玻璃或有机玻璃基层，中间两层为金属导电层，导电层间有透明隔离点绝缘。两个金属导电层是工作面，横向和纵向两端涂有导电银胶，通过四根导线连接银胶到外加电压和芯片 ADC 转换引脚。当触笔接触屏幕时，ADC 管脚测量 X、Y 方向电阻分压，形成触摸屏输入信号，构成硬件系统。

在软件层面，触摸屏测试采用先进入等待中断转换模式，中断产生后进入中断程序，首先判断是否为 “触笔点击”中断，若是则设置 ADC 进入自动连续 X/Y 转换模式，设定转换延时值并启动 ADC 转换，转换结束后读取触点坐标值，最

后判断是否为“触笔提起”中断，若是则退出中断程序。此过程中 ADC 模块产生触屏被按压时的 INT_ TC 中断和 X/Y 坐标转换后的 INT_ADC 中断，每次在“触笔点击”中断时从寄存器 ADCDAT0 和 ADCDAT1 中分别获取 xdata 和 ydata 的值。

电阻式触摸屏结构图与 ADC 模块和触摸屏接口如图 3.1 所示。

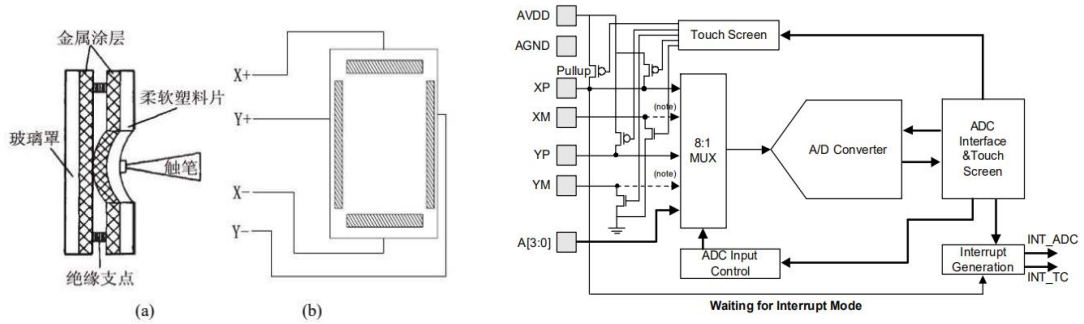

图 3.1 电阻式触摸屏结构与 ADC 模块和触摸屏接口图

3.2 输入处理

在触摸屏测试程序中，多次点击屏幕，电脑串口能够采集到一系列坐标数据。从这些数据可分析出 XP 和 YP 值反映触点坐标信息，其值来自于触屏 ADC 中断程序 void _irq AdcTsAuto(void)中的变量 xdata 和 ydata，xdata=(rADCDAT0& 0x3ff)；ydata=(rADCDAT1&0x3ff)。

由于电阻式触摸屏电阻分布非线性，显示屏坐标与笔触点实际位置有偏差，需进行触摸校正。校正过程包括数值采集分析，在主程序中编写程序段画黑色网格，用触摸笔点击横点和竖点并记录坐标值。采用最小二乘法进行校准，设拟合线性方程 $y^* = bx + a$，通过计算得到参数 a、b 和 x 与 xdata、y 与 ydata 间的校正函数，在触屏中断函数中插入相关语句实现校正。

4. 系统基础功能的实现

本部分主要介绍系统几个基础功能的实现方式，包括开机动画与音效逻辑、综合功能切换以及实时时钟显示逻辑。

4.1 开机动画与音效逻辑

系统依据音乐音符数据顺序播放相应音调，同时按照音符数据索引所处的不同范围，分阶段切换显示不同的开机画面图片。代码实现中，通过条件判断语句检测音符数据索引的值，然后根据不同的范围调用 Paint_Bmp 函数加载相应的图片资源并进行显示。每次切换后会利用提前计算好的音乐时间变量 timechange 设置相应延时，以确保画面切换与音乐播放节奏相匹配。直至音乐播放完毕且展示完最后一幅画面，便完成开机初始化流程，进入系统主界面。

4.2 综合功能切换

通过分支结构判断 xdata 和 ydata 的值来确定用户点击的屏幕区域，分别调

用不同函数进入相应功能模块，从而实现综合功能的切换。每调用一个功能之后重置 xdata 和 ydata，为下一次用户操作的判断作准备。

4.3 实时时钟显示逻辑

（1）时间获取与显示初始化：RTC_Display 函数首先调用 RTC_Time_Set 函数对 RTC 时间进行设置，接着进入一个循环，在循环内先判断触屏操作（当触屏横坐标和纵坐标处于特定范围时改变循环结束标志 RTCesc 的值），然后使能 RTC 的读写功能，读取年、月、日、小时、分、秒等时间信息对应的寄存器值，并进行相应的处理（如对年份加上 0x2000）。

（2）时间信息展示与循环更新：通过 Lcd_printf 函数将获取到的格式化后的时间信息显示在屏幕指定位置上，之后延时 900 毫秒，并利用 Paint_Bmp 函数重新绘制桌面背景图片，如此循环，不断更新并展示实时的 RTC 时间，直到满足触屏操作退出条件（通过改变 RTCesc 的值），循环结束，不再显示时间信息。

4.4 基础功能综合流程图

系统主程序，Start 函数以及 RTC_Display 函数综合流程图如图 4.1 所示。

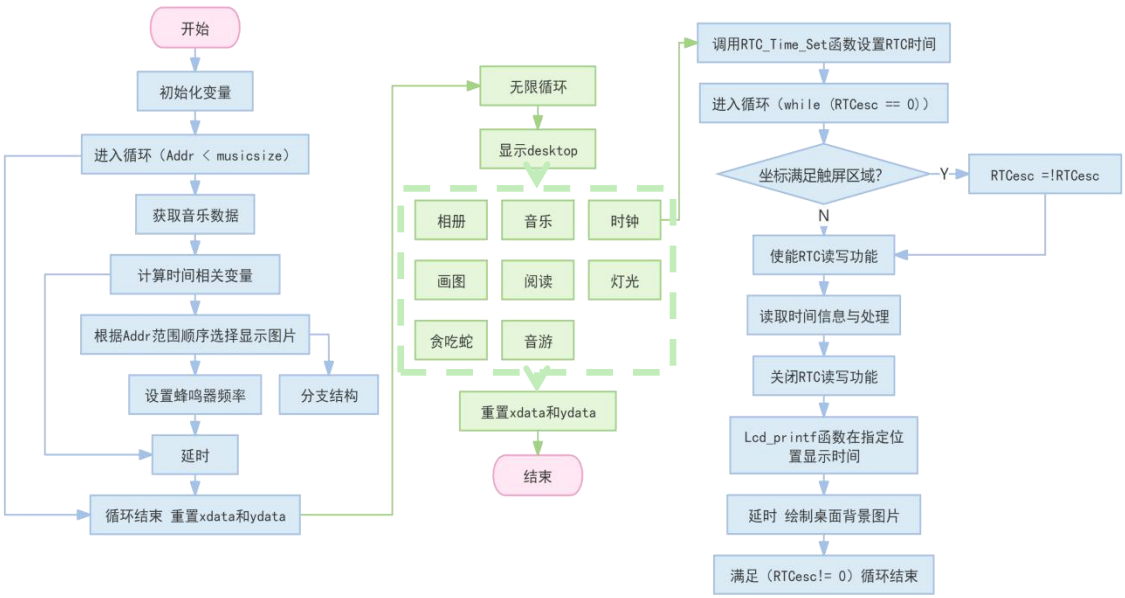

图 4.1 系统基础功能综合流程图

5. 相册功能的实现

本部分主要介绍相册功能中图片的存储结构，图片显示与切换、视频播放程序实现逻辑。

5.1 图片存储结构

从 bmp 文件转换而来的 picture.c 文件中定义一个 unsigned char 类型的数组

picture[]，用于存储图片数据。数据以十六进制的特定顺序排列，按照逐行扫描的方式存储像素数据，且每个十六进制值依次表示图片中各个像素的颜色值。

5.2 图片切换与显示

（1）图片显示：相册功能通过 xiangce 函数实现图片的切换与显示。进入相册后，首先绘制相册的背景图片，为用户提供图片与视频模式选择。图片模式下，会在屏幕上特定位置绘制不同的图片。例如，当进入该模式时，绘制一张大图片 picture1_F 以及六张小图片 picture1 到 picture6，这些图片按照一定的布局显示在屏幕上，为用户提供图片浏览界面。

（2）图片切换：图片切换操作主要通过判断 xdata 和 ydata 的值来实现。当用户点击屏幕缩略图区域，会触发图片切换操作，全屏显示所选图片。当用户点击全屏图中央区域，能够通过改变 show_small_pictures 以及相关标记变量（如 flag 和 flag_1）的值来实现缩略图的隐藏或显示。

5.3 图片切换逻辑

（1）变量定义与初始化：xiangce 函数开始时定义并初始化了一系列变量，包括 case_xiangce（用于标记相册不同模式，初始化为 0）、i、ii（用于循环计数）、show_small_pictures（用于控制小图片显示，初始化为 1）以及 flag 和 flag_1（用于标记图片相关状态，初始化为 0），同时将 xdata 和 ydata 重置为 0。

（2）主循环与模式选择：接着进入一个 while (1)的主循环，绘制相册背景图片 inxiangce。通过判断 xdata 和 ydata 的值确定用户选择的模式，如果 xdata、ydata 在左半区域，将 case_ xiangce 设置为 1，表示进入图片模式；如果 xdata、ydata 位于右半区域，将 case_xiangce 设置为 2，表示进入视频模式。每次模式选择后都将 xdata 和 ydata 重置为 0，防止变量仍然保留上一次操作的值陷入循环。

当 case_xiangce = 1 时，首先将其重置为 0，然后绘制一张大图片 picture1_F 以及六张小图片 picture1 到 picture6 在屏幕上的特定位置。接着进入一个内层 while (1)循环，用于处理图片的交互操作。

（3）缩略图显示隐藏逻辑：如果 xdata 和 ydata 位于中央区域，改变 show_ small _pictures 的值来切换小图片的显示状态，然后重置 xdata 和 ydata。

（4）全屏图显示切换逻辑：如果 show_small_pictures 为 false，根据 flag 的值绘制对应的大图片（如 flag = 1 时绘制 picture1_F），并更新 flag_1 的值，保证下次再次点击中央区域时缩略图能够显示出来，同时将 flag 重置为 0。

如果 show_small_pictures 为 true，通过判断 xdata 和 ydata 的值以及 flag_1 的值来确定绘制的图片。如果 xdata、ydata 在特定小图范围内或者 flag_1 等于相应的值，绘制相应的大图片和六张小图片，并更新 flag 和 flag_1 的值，形成互锁逻辑，同时重置 xdata 和 ydata。

（5）图片模式退出：如果 xdata、ydata 位于屏幕退出区域，将 case_xiangce 重置为 0 并重置 xdata 和 ydata 退出图片模式，跳出内层循环，回到主循环。

5.4 视频播放逻辑

视频播放逻辑通过 case_xiangce = 2 的条件分支来实现。通过判断 xdata 和 ydata 的值进入该模式后，会进入一个特定的视频播放循环。

（1）视频画面绘制：设置两层嵌套循环，外层循环次数为 16 次，用于控制视频的帧数。在每次外层循环中，又有一个内层循环来调整图片的显示位置，通过计算 ii = 253 - 3 * i 来改变 huojian 图片在垂直方向上的位置，以营造动态效果。迭代过程中，都会先绘制背景图片 diqiu，然后在特定位置绘制 huojian 图片并且设置一定的延时，模拟出视频播放的视觉效果。

（2）视频播放退出：与图片模式退出逻辑相同。

5.5 程序流程图

xiangce 函数程序流程图如图 5.1 所示。

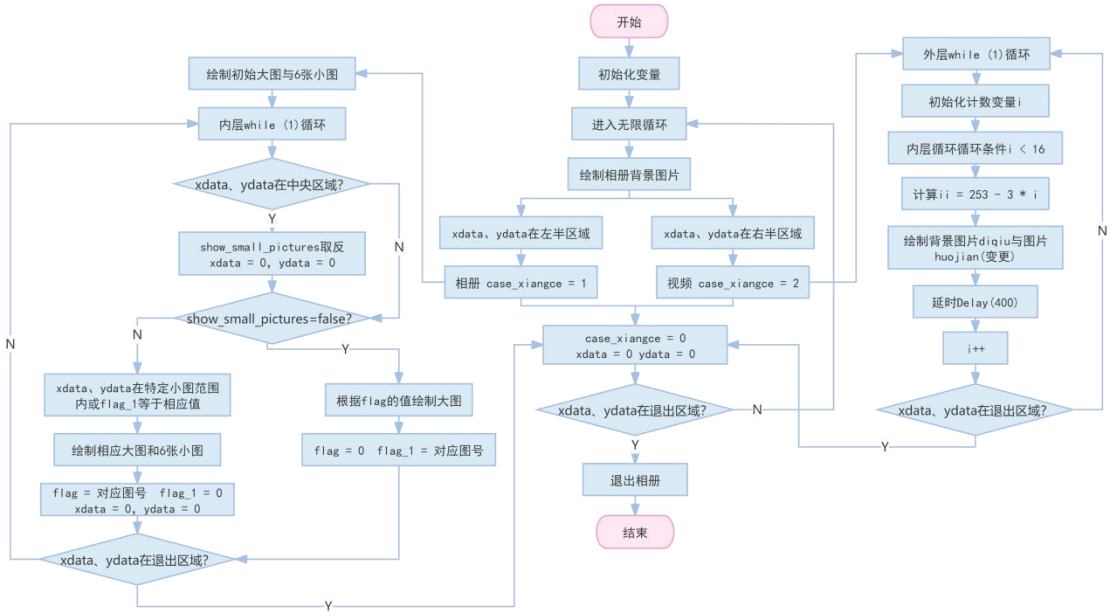

图 5.1 xiangce 函数程序流程图

6. 音乐播放功能的实现

本部分主要介绍音乐播放功能中音乐的存储结构，播放控制、音量控制与音乐切换程序实现逻辑。

6.1 音乐存储结构

在该系统中音乐是以数组形式存储。music 函数中定义了三首歌曲的音乐数据，分别存储在 SONG1、SONG2 和 SONG3 三个无符号字符数组中。数组中的每个元素代表音乐中的一个音符或节拍信息。

（1）音符与节拍的编码方式

①音符频率编码：数组中的奇数位数据用于表示音符频率。通过特定的十六进制数值与不同音区的音符相对应。例如，0x60,0x55,0x4c,0x48,0x40,0x39,0x33 等数值分别对应低音区的 do、re、mi、fa、sol、la、si；0x30,0x2b,0x26,0x24,0x20,0x1c,0x19 对应中音区；0x18,0x15,0x13,0x12,0x10,0x0e,0x0d 对应高音区。这种编码方式使得程序能够根据数组中的数据设置蜂鸣器发声频率，还原出歌曲中的音符。

②节拍延时编码：偶数位数据负责控制每个音频的延时时长，即对应乐谱中的节拍数。如 1/2 拍延时为 0x20，1/4 拍延时为 0x10。通过此方式呈现音乐节奏。

（2）通过 PWM 演奏歌曲的实现方式

在 music 函数中通过循环读取数组 SONG[]的数据来演奏歌曲。每次循环读取两个数据，分别为音符频率数据 Temp1 和节拍延时数据 Temp2。

通过公式 freq = 25000/Temp1 计算出频率，然后使用 Buzzer_Freq_Set0(freq)设置蜂鸣器频率，再使用 Delay(12*Temp2)进行相应的延时，以实现歌曲的演奏。如果 Addr（数组索引，用于记录播放进度）大于一定值，则重置 Addr 为 0，重新开始演奏。

6.2 音乐播放控制与切换逻辑

函数使用 while(1)作为主循环，不断检查用户的操作和播放状态。

（1）播放/继续/退出逻辑：当 xdata 和 ydata 的值在播放/继续按钮范围内，startmusic 被设置为 1/0，利用条件结构判断是否播放，重置 xdata 和 ydata 为 0。当 xdata 和 ydata 位于退出区域，while 循环将被 break 跳出，音乐播放功能停止。

（2）歌曲切换逻辑：当 xdata 和 ydata 的值在下/上一首按钮范围内，song change 变量会更新为下/上一首歌曲的索引。如果 songchange 达到最大值（此处为 2），则循环回到 0；达到最小值（此处为 0）则循环回到 2。同时，musicsize 更新为新歌曲的长度，Addr 重置为 0，xdata 和 ydata 也重置为 0。

（3）播放完成切换逻辑：在播放音乐的内层 while 循环中，有一个条件判断 if(Addr>=musicsize)。当当前歌曲播放到结尾（Addr 达到 musicsize）时，将切换至下一首。与点击下一首按钮类似，会更新 songchange、musicsize 和 Addr 的值。

6.3 音量控制

（1）按键音量控制

①Key_Scan 函数与硬件连接

K1 - K4 按键连接到 GPF0、GPF1、GPF2、GPF4 引脚，通过读取 rGPFDAT 判断按键状态；LED1 - LED4 连接 GPB5 - GPB8 引脚，通过修改 rGPBDAT 实现亮灭控制。

Key_Scan 函数主要负责检测开发板上的四个用户按键状态，按下时返回对应按键值（K1 - K4 分别对应 1 - 4），无按键按下则返回 0xff，且按键按下时相应 LED

灯亮起提供反馈。

先执行 Delay(80)以消除按键抖动。接着依次检测按键：

检测 K4（GPF0），若 rGPFDAT&(1<<0)==0，则 rGPBDAT = rGPBDAT & ~(LED4)点亮 LED4 并返回 4。

检测 K3（GPF2），若 rGPFDAT&(1<<2)==0，则 rGPBDAT = rGPBDAT & ~(LED3)点亮 LED3 并返回 3。

……

若都未检测到按键按下，执行 rGPBDAT = rGPBDAT & ~0x1e0|0x1e0 熄灭 LED[8:5]并返回 0xff。

②调用 Key_Scan 实现音量控制

当检测到按键操作（key 变量）为 3 时，表示用户按下了增加音量的按键，此时 dot1 变量增加 0.01，从而增加音量。当 key 为 4 时，表示用户按下了减小音量的按键，dot1 变量减小 0.01，从而减小音量。

（2）触屏音量控制：当 xdata 和 ydata 的值在音量增加/减小按钮范围内，dot1 变量会增加/减小 0.01，并且 xdata 和 ydata 重置为 0。

6.4 程序流程图

music 函数程序流程图如图 6.1 所示。

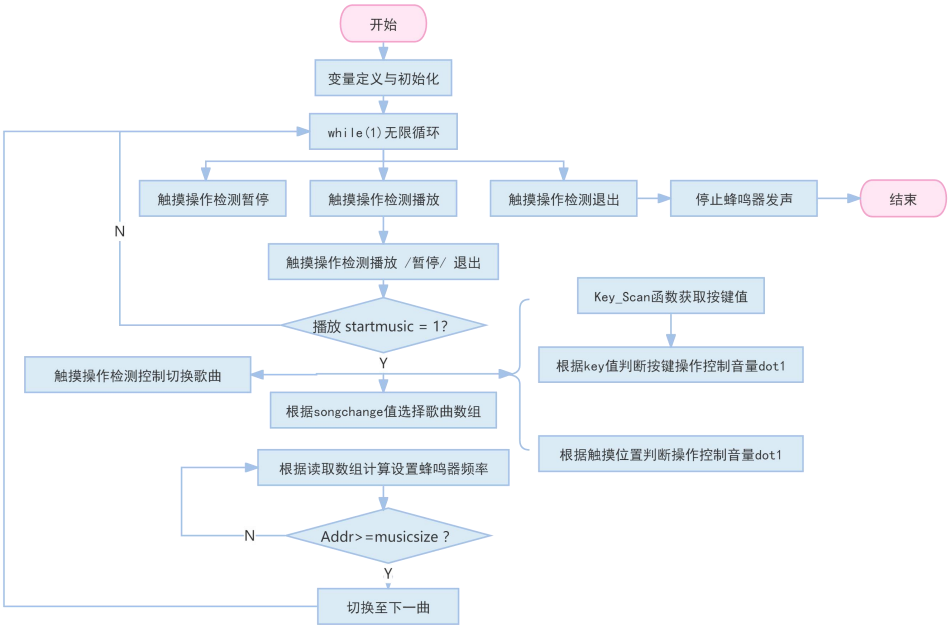

图 6.1 music 函数程序流程图

7. 电子书阅读功能的实现

本部分简要介绍中文字库存储结构与电子书阅读功能实现逻辑。

7.1 字库存储结构

中文字库存储在 Font_Libs.c 文件中的__CHS[]数组，包含了所有可显示汉字的点阵信息。

（1）点阵表示：汉字采用 16*16 的点阵表示。每个点阵位的取值只有 0 和 1 两种情况。其中，1 代表该位置的像素在显示汉字时应被点亮，通常使用前景色来绘制；0 代表该像素不被点亮，可能显示为背景色或者保持透明。通过这 16*16 个点阵位的不同组合，可描绘出各种复杂的汉字形状。

（2）字符编码与索引：中文字库使用 GB2312 编码标准。GB2312 将汉字分为 94 个区，每个区有 94 个位，汉字从 0xb0a1 开始，结束于 0xf7fe。在程序中，要显示一个汉字时，先通过汉字的 GB2312 编码计算出其区位码，然后根据区位码计算出在__CHS[]数组中的偏移量，从而获取对应的点阵数据。这种编码方式使得字库能够高效地存储和检索大量汉字，满足中文显示的需求。

（3）中文字符显示函数调用：中文字的显示通过 Lcd_printf 函数实现，该函数内部先使用 vsprintf 函数将格式化后的字符串存储到 LCD_Printf_Buf 缓冲区中，然后逐个字符进行处理。

7.2 书籍/页面显示与切换逻辑

电子书功能通过 ebook 函数实现。函数内首先初始化页面变量 page 为 0，并绘制阅读背景。然后在循环中，依据触摸位置判断翻页操作，左侧区域点击 page--，右侧区域点击 page++，同时限制 page 范围。接着根据 page 值用 switch 语句显示不同页面内容。循环中持续检测是否点击返回书架区域，若满足条件则调用 shujia 函数切换到书架界面。shujia 函数起始重置 xdata 和 ydata，绘制书架背景。接着通过检测触摸位置判断点击某本书籍区域，若在特定区域点击则调用 ebook 函数进入阅读；若点击返回区域，则跳出循环返回上级界面。

7.3 程序流程图

ebook 函数与 shujia 函数程序流程图如图 7.1 所示。

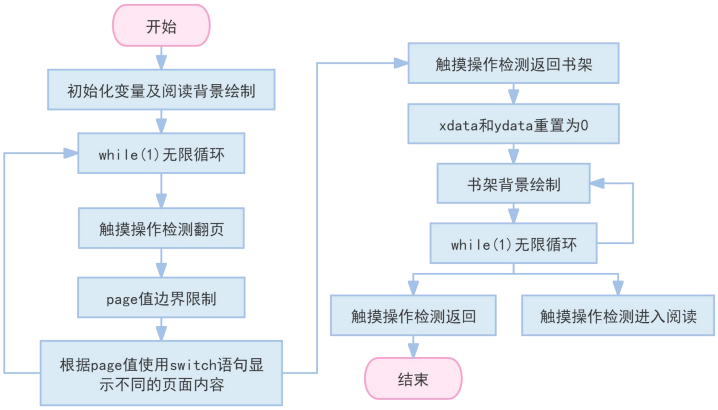

图 7.1 ebook 函数与 shujia 函数程序流程图

8. 绘图功能的实现

本部分结合 LCD_TFT.c 中部分函数与自行编写形状函数，实现简易画图功能。

8.1 区域选择检测与绘制

由于绘图背景的不变性，在主程序绘画分支下绘制初始背景。draw 函数开始定义颜色变量 color_1 并初始化为 0，在 while(esc==0)循环中开启绘画操作。

（1）颜色区域：使用 Glib_FilledRectangle 函数绘制了多个矩形区域分布于屏幕特定位置，用于显示不同的颜色选项。每个矩形对应一种颜色，通过坐标和颜色值来定义。当通过 xdata 和 ydata 检测到用户点击了某颜色区域，设置 color_1 为相应的颜色值。color_1=0，表示默认初始化选择黑色。

（2）图形区域：在选择颜色后，进入一个内层 while(1)循环，再次检测用户操作。如果用户点击了预定的图形区域，则调用相应的绘制函数。包括 huaxian 函数、huajuxing 函数，filljuxing 函数和 fillyuan 函数。

8.2 清屏与退出操作处理

在执行绘制函数过程中，还会通过条件判断持续检测用户操作。如果用户点击了颜色选择区域或清屏、退出区域则会跳出内层循环，根据新的操作进行相应处理。清屏操作包括重新绘制背景图片，重置 xdata 和 ydata 为 0，以及 Glib_FilledRectangle 重新绘制颜色选择区域，以清除之前的绘图内容，准备新的绘图操作。退出操作设置 esc 为 1，跳出主循环，结束画图功能。

8.3 圆形填充函数逻辑

首先定义画水平线函数 LCD_DrawHLine。使用 while 循环，只要 x0 小于等于 x1，就会执行循环体。首先调用 PutPixel 函数用于在指定的坐标(x0, y0)处绘制一个像素点。然后将 x0 的值自增 1，准备绘制下一个像素点。通过不断重复这个过程，从而逐像素绘制出一条水平直线。

在 Fill_Circle 函数中，首先计算与圆形算法相关辅助变量 imax = ((int)((int)r *707))/1000+1 和 sqmax = (int)r*(int)r+(int)r/2，用于确定圆边界。接着将 x 初始化为半径 r，调用 LCD_DrawHLine(x0-r,y0,x0+r,color)绘制穿过圆心、长度为直径的水平直线，作为圆的初始轮廓。for 循环中，i 从 1 增至 imax，循环内先判断(i*i +x*x) >sqmax，若满足且 x>imax，绘制圆上、下半部分水平直线（随 i 逼近边界）并通过 x--调整边界点；无论是否满足该条件，都会绘制圆左、右两侧水平直线（长度和位置由 x、i 确定）。通过不断调整 x、i 并绘制水平直线，利用圆对称性，实现绘制效果。

8.3 程序流程图

draw 函数程序流程图如图 8.1 所示。

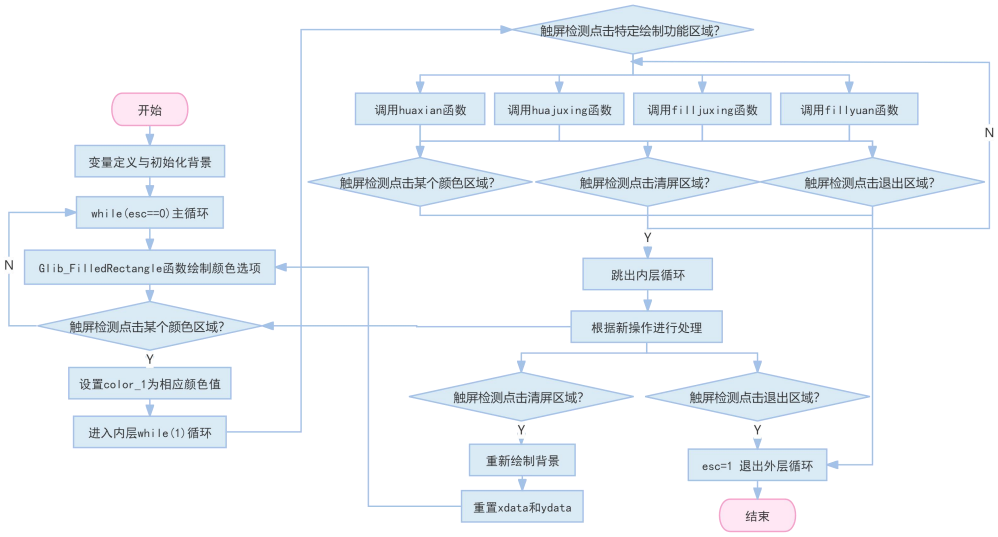

图 8.1 draw 函数程序流程图

9. 灯光控制功能的实现

本部分结合嵌入式实验中 LED 灯控制逻辑实现灯光控制功能。

9.1 硬件初始化与背景绘制

通过 rGPBCON&=(1<<10)|(1<<12)|(1<<14)|(1<<16)语句对 GPBCON 寄存器进行操作，配置与 LED 灯相关的引脚功能为输出模式。rGPBUP|=0xffff 语句设置 GPB 端口上拉电阻禁止。然后将 rGPBDAT 初始化为 0x1e0，设置初始状态下 LED 灯的熄灭状态（高电平熄灭）。

使用 Paint_Bmp 绘制 led 的背景图片，为亮灯操作提供可视化背景界面。

9.2 触摸检测与亮灯操作

进入 while(1)无限循环，程序持续检测触摸屏输入。如果触发至亮灯区域，通过异或操作来切换连接到 GPBx 引脚的 LED 灯状态，然后将 xdata 和 ydata 重置为 0。当触发至流水灯区域，调用 Led_Display 函数并通过延时实现流水灯效果。

9.3 程序流程图

light 函数程序流程图如图 9.1 所示。

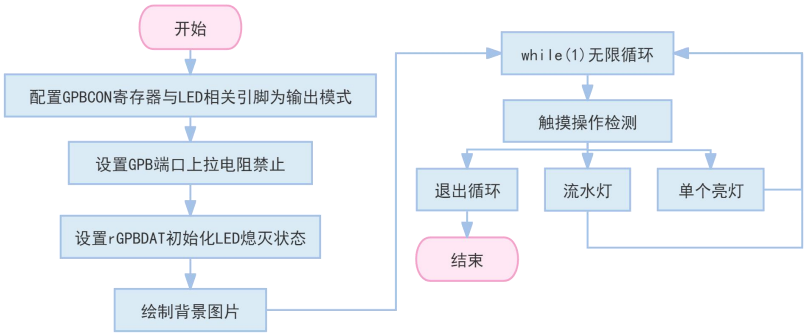

图 9.1 light 函数程序流程图

10. 贪吃蛇游戏功能的实现

本部分主要介绍贪吃蛇游戏状态更新、图像绘制与流程控制的实现逻辑。

10.1 蛇的位置和方向

（1）初始位置与变量定义：游戏开始时，蛇的初始位置通过变量 x 和 y 定义，表示蛇头在屏幕上的坐标。定义了 a 和 b 变量表示食物的位置，初始化为一定数值。数组 array_pos[10000][2]用于存储蛇身每个节点的位置，其中 array_pos [start][0]和 array_pos[start][1]分别记录蛇头的 x 和 y 坐标，随着游戏进行，蛇身节点位置依次记录在该数组中。

（2）用户输入处理与方向控制移动逻辑：通过直接判断 xdata 和 ydata 的值来获取用户输入。利用变量 key 记录蛇的移动方向，如 1 表示向上，3 表示向左，其值与触摸屏上特定按钮位置相对应。如果用户输入的方向与蛇当前的移动方向相反（例如，蛇正在向右移动，用户输入向左的方向键），则不会改变方向，这是为了防止蛇直接反向移动导致不合理的游戏行为。

在游戏循环中，根据 key 的值更新蛇头的位置。例如，当 key = 4 时，x += 10 持续使蛇头向右移动；当 key = 2 时，y += 10 持续使蛇头向下移动。每次移动后，将当前蛇头位置存入 array_pos 数组中，用于后续绘制蛇身。

10.2 游戏状态更新

（1）蛇身绘制与更新：在每次游戏循环中，通过 Fill_Circle 函数绘制蛇身。首先，根据 array_pos 数组中的坐标，从蛇头到蛇尾依次绘制蛇身节点，使蛇呈现出移动的效果。当蛇吃到食物（即 x == a && y == b）时，蛇身长度增加，length++。通过在 array_pos 数组中添加新的蛇头位置来实现蛇身的增长。

（2）食物生成与处理：食物初始位置由 a 和 b 定义，通过 Fill_Circle 函数绘制在屏幕上。当蛇头与食物位置重合时，蛇吃到食物，执行一系列操作，如增加分数 score_1++，调整游戏速度 speed--，并重新生成食物位置。重新生成食物位置时，使用 rand 函数生成随机坐标，并通过 do - while 循环确保食物不在蛇身或障碍物上。

（3）碰撞检测与游戏结束判断：游戏中存在多种碰撞情况会导致游戏结束。当蛇头碰撞到游戏区域边界或障碍物时（通过判断蛇头位置是否与区域边界坐标重合）游戏结束。同时，当蛇头撞到自己的身体（通过遍历 array_pos 数组，判断蛇头位置是否与蛇身节点位置重合，需保证蛇身长度大于 0），游戏也会结束。

10.3 游戏画面绘制与显示

（1）背景绘制与实时分数显示：游戏开始时，通过 Paint_Bmp 绘制游戏草地背景图片以及装饰、障碍物等。使用 Lcd_printf 函数与计分变量将实际分数显

示在屏幕右上方位置。

（2）动态效果绘制：随着游戏的进行，不断更新蛇和食物的位置，并通过绘制和擦除相应图形来实现动态效果。例如，每次蛇移动时，先擦除蛇尾的图形（调用 Fill_Circle 用背景颜色覆盖），然后在新的蛇头位置绘制蛇头图形，使蛇看起来像在移动。食物被吃掉后重新生成的过程中，也会利用蛇头节点覆盖擦除原来食物的图形，再在新的位置重新绘制。

10.4 游戏循环与流程控制

（1）游戏主循环：游戏的核心逻辑在一个 while 循环中实现，只要蛇头位置满足游戏继续的条件，循环就会持续执行。

在循环中，依次处理用户输入、更新游戏状态（蛇头位置、食物状态、分数等）、绘制游戏画面，并通过 Delay(speed)函数控制游戏的帧率，使游戏以合适的速度运行。且速度变量 speed 会随着游戏进行而变化，增加挑战性。

（2）游戏结束处理：当游戏结束条件满足时，循环结束。绘制游戏结束画面，并调用 score_tanchishe 函数显示总得分，等待用户点击任意位置退出（通过另一个 while 循环判断用户是否有新的触屏操作）。然后将分数重置为 0，准备下一次游戏。整个游戏流程通过这样的循环和各类条件判断实现游戏的开始、进行、结束和的完整控制。

10.5 程序流程图

tanchishe 函数程序流程图如图 10.1 所示。

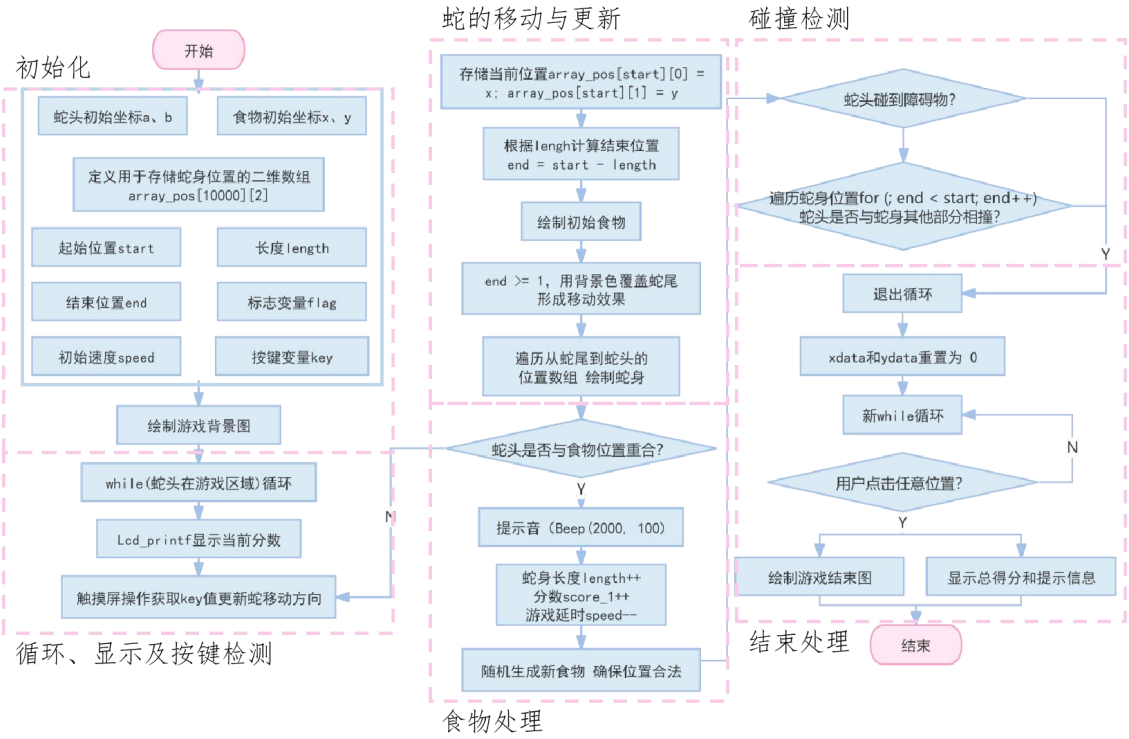

图 10.1 tanchishe 函数程序流程图

11. 音乐游戏功能的实现

受游戏功能和音乐功能的启发，整合实习中所学的 lcd、触摸屏、蜂鸣器、图片文字显示、动画制作等知识，并通过自主设计游戏逻辑，实现一个音游爱好者喜闻乐见的音乐游戏。

11.1 整体架构

采用模块化设计，将程序划分为画面更新、游戏物体参量变化、得分逻辑、音乐与游戏触发结合等几个主要部分，每个部分负责特定的功能，易于开发。游戏主体循环及其调用相关函数如下：

```
void game_loop() {
    int jump=0;
    init_notes();
    background_set();
    Paint_Bmp(20, 50, 75, 75, down2);
    while (jump==0) {
     time_count();
        update_notes();
        draw_scene();
        if(notes[MAX_NOTES-1].hit==1){
        dash_game_over(scores);
        while(xdata==0&&ydata==0){
        jump =1;}
        } } }
```

11.2 画面更新

运用 paint_bmp 函数通过不断覆写来实现动画效果，模拟一帧一帧的画面更新。先使用 Lcd_ClearScr 清除总体画面，再绘制背景图、角色、音符等元素，循环此过程以达到动态效果。

在实际操作过程中，paint_bmp 函数在绘画全景背景图时计算量大，耗时较多，在游戏过程中造成刷新缓慢。因此优化画面更新逻辑，只在元素改变时覆写，对于移动的音符仅覆写需要更新的一小段区域。

11.3 游戏物体参量变化

在游戏中，音符分为两行分别从右到左移动，人物依靠点击实现上下移动。

（1）音符管理：由于音符众多，而且需要每个单独控制，并且拥有较多属性，因此使用结构体 struct Note 来进行管理：

```
struct Note {
    int x;
    int y;
    int hit; //是否被击打
```

```
    int start_time;  //开始时间
};
```

通过一个数组 struct Note notes[MAX_NOTES]来存储多个音符。

音符每次更新时，根据其开始时间判断是否移动（x -= NOTE_SPEED），并进行一系列是否被击中的判断逻辑。当音符到达判定区域且满足其他条件（如触击位置等）时，标记为击中，同时更新相关变量（如得分、连击数等）。

（2）人物动画：通过定义多个变量来控制人物上下动画与击打动画，以给玩家正向反馈。使用如下变量：

```
int muse_pose = 0;  //用于确定上下位置
int muse_hit =0;     //用于确定是否击打
int muse_stand =0;   //用于计算击打动画存在的时间
int hit_stand = 0;  //用于设置判定区间
```

根据触击位置和是否击中音符来选择绘制不同的角色动作，并且在一定时间后重置击打动画。

11.4 得分逻辑

设置全局变量 scores 作为总分数，在每次击中音符的判定中 scores++，同时 combo++（连击数增加）。通过触摸屏中断获取触击位置 xdata、ydata，当音符到达判定区域且触击符合条件时实现得分。

11.5 音乐与游戏触发结合

既然是音游，自然需要有乐谱与所有音符对应。参考之前的音乐播放逻辑，通过蜂鸣器的不同频率形成音高，乐谱是由不同频率的 16 进制数组成的数组，用数组位置 addr 不断增加以逐步读取乐谱。由此只需要将音符变量与乐谱结合起来即可。

（1）乐谱设计：创建一个乐谱的二维数组：

```
int piece[MAX_NOTES][2]
={{1,1},{1,1.5},{0,3},{1,4},{0,5},{0,5.5},{0,7},{0,7.5},{1,9},
{0,10.5},{1,11},{1,12},{0,13},{0,14},{1,15},
{1,16},{0,17},{1,18},{0,19},{1,20},{0,21},{0,22},{1,23},{1,24},
{0,25},{1,26},{1,27},{0,28},{0,29},{1,30},{1,31},{0,32}};
```

包含音符位置上下及出现时间，决定了音符的时值。同时有一个音符频率数组 unsigned char GAME_THEME[]存储不同音符对应的频率数据。

（2）音乐播放与音符关联：在音符被击中的判定逻辑中，当满足击中条件时，只需根据音符在乐谱数组中相对应的位置获取频率数据，通过蜂鸣器播放相应乐谱中的音高，便能有不同的旋律，实现音乐与游戏操作的同步。

11.6 程序实现流程

（1）初始化阶段

在 init_notes 函数中，初始化游戏相关变量。将音符数组 notes 中的每个音符的初始位置 x 设为 430，y 根据乐谱数组 piece 中的位置信息计算得出，hit 初始化为 0，start_time 根据乐谱中的时间信息设置。同时初始化得分 scores、连击数 combo、人物状态变量（muse_pose、muse_hit、muse_stand、hit_stand 等）以及时间变量 time1 等。

调用 background_set 函数，绘制游戏背景图（先清屏再绘制 gamebeijing 和 boss 图片），并调用 Paint_Bmp 函数绘制初始角色位置（如 down2 图片）。

（2）游戏循环阶段

在 game_loop 函数中，进入主循环。每次循环执行以下操作：

①调用 time_count 函数，更新时间相关变量（time1 增加、hit_stand 和 muse_stand 增加，同时进行一定延时 Delay(50)）。

②调用 update_notes 函数，更新音符状态。包括根据触摸屏输入更新人物位置（通过 ydata 判断并更新 muse_pose 等变量），移动音符（根据 start_time 判断并改变 x 坐标），判断音符是否击中（根据位置和触击条件），如果击中则更新得分、连击数，播放相应音符音高，同时处理音符未击中或移出屏幕等情况。

③调用 draw_scene 函数，根据音符状态绘制音符。对于未击中且在屏幕内的音符，根据其位置和方向绘制不同的音符图形（如 ball1 或 ball2 及拖尾 fugai）。

④检查游戏是否结束，即判断最后一个音符 notes[MAX_NOTES - 1]是否被击中，如果击中则调用 dash_game_over 函数。

（3）画面绘制与更新

①背景绘制：在 background_set 函数中，先使用 Lcd_ClearScr 清除屏幕，然后绘制游戏背景图片 gamebeijing 和 boss 图片。

②角色绘制：在 character_set 函数中，根据人物状态变量（muse_pose 和 muse_hit）选择绘制不同的角色图片（如 down1、down2、up1、up2 等），同时调用 score_display 函数显示当前得分。

③音符绘制：在 draw_scene 函数中，通过循环遍历音符数组，对于未击中且在屏幕内的音符，根据其位置和方向绘制相应的音符图形，并处理音符移动时的拖尾覆写逻辑，以实现音符的动态显示效果。

（4）得分与音乐触发

①得分计算：在 update_notes 函数中，当音符击中判定成功时，scores++和 combo++实现得分和连击数增加。

②音乐播放：同样在 update_notes 函数中，当音符击中时，根据音符在乐谱中的位置获取频率数据（从 GAME_THEME 数组中），通过 Buzzer_Freq_Set0 函

数设置蜂鸣器频率播放相应音符音高。

（5）游戏结束处理

dash_game_over 函数绘制游戏结束画面 gameover，设置蜂鸣器相关参数（rTCMPB0 = 0），显示总得分和提示信息，并等待玩家点击屏幕任意位置退出，通过 xdata 和 ydata 判断玩家操作，当玩家点击后结束游戏循环。

11.7 游戏设计流程图

本游戏整体设计思路流程如图 11.1 所示。

图 11.1 音乐游戏设计流程图

11.8 游戏实现效果

图 11.2 展示了部分游戏截图。

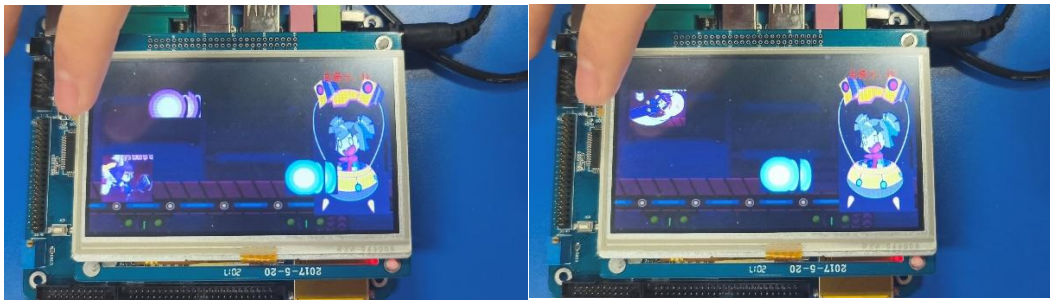

图 11.2 游戏呈现效果

12. 总结

本次实习中，我们深入学习实践了嵌入式系统开发，从初期的笔记本环境搭

建到各功能模块的测试与修改，包括触摸屏、LCD、PWM、RTC 等模块的熟悉与图像显示、音频输出、字库移植等操作实现。仔细学习第一阶段的实验指导书并实际操作修改代码后，对系统整体与各功能模块函数的原理和实现逻辑、调用方法有了初步的掌握，进而将各部分迁移至 TQ2440_Test 测试程序中，开展第二阶段智能平板的综合项目开发，不仅提升了编程技术能力，还培养了解决问题的能力与创新思维。但实际操作过程中，也遇到了诸多问题。

工程环境配置、编译以及图片显示问题，包括：nand flash 启动后出现“白屏”不工作状态；从自己笔记本工程中将文件拷贝至实验室后电脑后路径不对；编译时未勾选全部文件导致报错；第 n 张图片显示过后出现紫色等纯色图像；显示的图像不正确，有彩色的雪花点……我们均遇到过，尝试多次无法解决后，发现上述问题的解决方案大都出现在 FAQ 文件中。

软件层面遇到问题最多的还是音乐游戏的逻辑实现上。多次调试后总结出了如下几个值得注意的方面：

①在 game_loop 函数里按照 time_count、update_notes、draw_scene 的顺序依次调用相关函数来推进游戏进程，这几个函数之间存在一定的数据依赖和先后顺序要求。如果调用顺序有误，比如先绘制场景再更新音符状态，就会导致画面显示的是上一时刻的音符状态，出现画面与逻辑不同步的问题。

②在音符进入击中判定范围（notes[i].x < 60+HIT_RANGE）后的具体击中条件判断中，涉及多个条件的组合，如音符的 x、y 坐标、角色的姿势以及 hit_stand 的值等共同来决定是否击中音符。这里逻辑相对复杂，如果在条件的边界值处理上不准确，比如 HIT_RANGE 的设置不合理，或者 hit_stand 的更新和使用有误，都可能导致音符击中判定不准确。

③在 update_notes 函数中根据 y_record、muse_stand、normal 等变量的值来调用 character_set 函数更新角色画面显示并重置相关状态变量。当这些变量的更新和判断条件设置不准确，会导致角色画面不更新，与实际游戏状态不匹配。

其它功能模块的实现过程也并非一帆风顺，好在我们反复检查代码逻辑以及与同学积极讨论交流，逐一攻克了难关。最终呈现了一个功能完整且精细、用户体验感良好的嵌入式系统项目。

13. 心得体会

以往编写代码实现功能时，多局限于命令行窗口显示，用户体验单一。此次实习引入了 LCD 与触摸屏，这让一个个想法得以通过直观的人机交互界面 HMI 展现出来，也让我们格外细致用心，不断追求更好的使用体验。

回顾嵌入式开发的学习历程，我们在过去几个学期中已陆续有所接触，本次实习的第一阶段更是深入学习了各个功能模块。这些知识相对零散，尚未形成完

整的体系。而通过本次综合系统的开发实践，我们能够从整体上把握一个程序的内部结构。看到各个部分是如何协同工作，共同实现复杂功能的。这种对系统整体与局部的认识，让我们明白了在嵌入式系统开发中，首先要关注单个功能的实现，还需要注重各个功能之间的协调与整合，以构建一个完整高效的系统。

在项目开发过程中，对于资源的整合能力得到了充分锻炼。整个实习不仅考验主程序以及各函数的逻辑编程能力，还需要了解工程中的其他文件，包括与底层寄存器相关的部分，这些对诸如按键、LED、PWM 模块等的实现较为重要。LCD_TFT.c 中内置了很多函数，学会如何正确调用这些函数并与其他模块综合运用是实现复杂功能的关键。这也让我们再次感受到嵌入式系统开发是一个涉及硬件和软件多方面知识的综合性工作。

为节约实习时间，我们在自己的笔记本上搭建了开发环境。这使我们能够提前编写代码并进行编译调错，提高了开发效率。之后在实验室进行烧录程序时，通过软硬件结合，根据运行结果排查逻辑错误。从实验现象出发，观察到不合预期的结果反馈时，需要沉下心来，仔细梳理冗长复杂的代码逻辑，逐步修改调试。这一过程极大地提升了我们的编程能力，让我们在解决问题的过程中不断成长。

通过本次实习，我们深刻认识到嵌入式系统开发是一个综合性强、技术难度较高的领域，需要我们具备扎实的理论基础、严谨的编程思维和较强的问题解决能力。在遇到问题时，不能轻易放弃，要善于利用各种资源，从不同的角度去思考和分析问题，通过不断尝试和改进，最终找到解决问题的方法。同时，团队合作和交流也至关重要，与同学分享经验和思路，能够加快问题解决的速度，提高开发效率。

缺憾的部分：

在使用开发板时，我们查阅了 TQ2440 的开发手册，发现还有耳机功能、录音功能和摄像头功能可以使用，并且找到了例程。经过测试后，耳机功能可以使用，并且由于音频解码的逻辑不同，音质出色；录音功能并不能正常使用；cmos 摄像头则需要另外购买。

所以最开始音游的设想是使用耳机播放既成音乐，并跟着节奏设置音符的音值的。但是在音乐转数组过程中，发现不同比特率、频道、音频采样频率、采样大小都会影响文件头几位数值：

比特率	1024kbps
频道	2 (立体声)
音频采样频率	32.000 kHz
音频采样大小	16 位

0x52, 0x49, 0x46, 0x46, 0x72, 0x01, 0x3e, 0x01, 0x57, 0x41, 0x56, 0x45, 0x66, 0x6d, 0x74,0x20,

0x10, 0x00, 0x00, 0x00, 0x01, 0x00, 0x02, 0x00, 0x00, 0x7d, 0x00, 0x00, 0x00, 0xf4, 0x01, 0x00,

0x04, 0x00, 0x10, 0x00, 0x4c, 0x49, 0x53, 0x54, 0x1a, 0x00, 0x00, 0x00, 0x49, 0x4e, 0x46, 0x4f,

只有设置对了参数才能正常播放，否则会有尖锐的蜂鸣声。

即便如此，由于该音频与简单的蜂鸣器音频不同，包含了太多信息，导致文件极大，无法全部写入开发板中，故没法正常播放较长的音乐，权衡之下采用了蜂鸣器播放的方案。

第二部分：Linux 系统体验实习

1. 对 Linux 系统的理解

Linux 是一种完全免费、自由且开放源码的类 Unix 操作系统，遍布全球的 Linux 爱好者是其开发的强大技术后盾，是一个真正的多任务、多用户的操作系统。提供了先进的网络支持，符合 IEEE POSIX 标准，支持数十种文件系统格式，采用先进的内存管理机制，更加有效地利用物理内存；具有内核小、系统健壮、效率高、功能强大、资源丰富以及易于移植等特点。

通常我们使用的 Linux 操作系统是一个集 Linux 内核、工具集、各种库、桌面管理器、应用程序等于一体的一个发布包；主要由 4 部分组成，即：Linux 内核、SHELL、文件系统以及应用系统。 在 Linux 系统中“一切皆文件”，对 Linux 系统的操作本质上是对文件进行操作，这点与 Windows 具有很大差别。在本次实习中，对 Linux 系统的操作是通过各种命令来进行的。

基于它的应用场景可划分为桌面 Linux 操作系统和嵌入式 Linux 操作系统。其中，嵌入式 Linux 操作系统是经小型化裁剪后应用于特定场合的专用操作系统；它位于硬件抽象层之上，应用程序之下，向下管理硬件资源，向上为应用程序提供 接口，广泛应用于手机、机顶盒、掌上电脑、车载盒及工业控制等智能信息产品中。

在嵌入式开发中，Linux 系统作为一种开源、稳定且高度可定制的平台，广泛应用于各类硬件设备中。通过这次实习，我们初步体验了 Linux 系统在嵌入式领域的应用，尤其是它的内核、驱动移植以及简化应用开发。

在嵌入式 Linux 系统中，操作系统的内核提供了对硬件的精细控制。通过内核模块、设备驱动以及文件系统等多个子系统，开发者可以高效地管理系统资源。在本次实习中，我们使用了 VMware Workstation 16 虚拟机在 PC 端搭建了 Redhat 6.0 系统作为开发环境。为了能够在 ARM 架构的开发板上运行 Linux，我们配置了交叉编译工具链 arm-linux-gcc 4.3.3，确保能为目标平台编译内核和应用程序。

Linux 系统为嵌入式开发提供了高度的灵活性和可定制性。通过修改内核配置、选择合适的驱动程序、编译系统镜像，能够针对不同硬件平台进行个性化调整。嵌入式设备的开发需要处理具体硬件的外设接口（如 LCD、触摸屏、USB 设备），而 Linux 内核提供了丰富的硬件抽象层，使得这些硬件能够通过统一的接口进行访问和操作。

2. 对移植的理解

移植是指将软件从一种硬件平台或操作系统迁移到另一种平台的过程。Linux 内核的移植主要涉及以下几个方面：获取适用于目标硬件的 Linux 源码、交叉编译工具链的配置、硬件特性的支持以及驱动程序的适配。

在本次实习中，我们对内核移植有了更深的理解。首先，需要获取 Linux 内核的源码，并将其解压至指定目录。接着，对内核配置文件进行修改，确保内核能够支持 ARM 架构的开发板目标平台。这包括修改 Makefile 文件中的 ARCH 和 CROSS_COMPILE 变量，以便为目标架构生成可执行的内核映像。

内核移植的难点在于与硬件的兼容性问题。每个开发板的硬件配置不同，因此需要根据具体的硬件平台修改内核配置中的时钟频率、内存地址以及设备驱动等。通过修改 arch/arm/mach-s3c2440/mach-smdk2440.c 文件中的时钟频率和机器码，能够确保内核能够与开发板上的硬件相兼容。

移植过程较为繁琐，但移植 Linux 的内核及驱动就是不断的发现错误，解决错误的过程。需要根据命令行窗口提示的错误信息，细致的修改内核配置，逐步解决。

3. 驱动的开发

驱动程序是操作系统与硬件之间的桥梁，负责将操作系统的指令转换为硬件能理解的操作。在嵌入式 Linux 开发中，设备驱动的移植和编写是非常关键的部分，涉及到硬件的初始化、数据传输以及硬件资源的管理。

3.1 实现驱动移植

在本次实习中，我们移植了多个硬件驱动，包括 LCD 驱动、触摸屏驱动和 USB 设备驱动。LCD 驱动的移植主要涉及对时钟频率和显示参数的配置。通过修改内核源码中的 drivers/video/s3c2410fb.c 文件，调整 LCD 的时钟计算公式，并根据开发板的显示屏参数修改了配置文件，从而实现了 LCD 显示。

触摸屏驱动的移植则需要处理与硬件的通信，并确保触摸事件能正确传递给应用程序。通过添加触摸屏的驱动补丁和修改内核配置，我们成功将触摸屏集成到 Linux 系统中。此外，在移植过程中，还需要根据开发板的硬件特性修改文件系统的配置，以确保触摸屏能够正常工作。

对于 USB 设备驱动的移植，Linux 内核已经提供了对 USB 存储设备（如 U 盘）的支持。我们只需要通过修改内核配置文件，启用 USB 存储支持，即可使得 U 盘能够在开发板上被自动识别和挂载。

3.2 编写驱动程序

（1）Hello 驱动开发：在内核源码的 drivers/char/目录下创建 EmbedSky_hello.c 文件，实现加载和卸载时的简单信息输出。修改 Makefile 文件添加编译规则，进入内核配置界面将 TQ2440/SKY2440 Hello Driver 配置为模块，保存配置后，使用 make SUBDIR=drivers/char/modules 命令编译出驱动模块 EmbedSky_hello.ko。将该驱动模块复制到 U 盘中，插入开发板后，通过 insmod 和 rmmod 命令成功加载和卸载驱动，

（2）LED 驱动开发与应用：修改 Kconfig 和 Makefile 文件，添加 LED 驱动配置和编译规则，通过 make menuconfig 将 LED 驱动配置为模块，编译出驱动模块 EmbedSky_led.ko。利用现有框架编写 leds.c 应用程序实现 LED 灯控制，通过命令行参数指定亮灭状态，编译后拷贝至/sbin/目录。改写/etc/init.d/rcS 文件，使系统启动时自动运行 LED 应用程序，还提供 led-player 实现流水灯效果。

4. 基于 Linux 系统的应用开发与裸机应用开发的区别

在嵌入式开发中，裸机应用开发和基于 Linux 的应用开发有很大的区别。裸机应用开发是在没有操作系统支持的情况下直接与硬件交互，而基于 Linux 的应用开发则通过操作系统提供的服务进行资源管理和硬件交互。

4.1 裸机应用开发

裸机应用开发指的是在没有操作系统的情况下开发应用程序，开发者需要直接操作硬件。裸机程序通常由 main 函数中的 while 循环和中断服务程序组成，CPU 通常会一直执行 while 循环中的代码，只有在外部事件（如中断）发生时，才会跳转到相应的中断服务程序进行处理。在裸机开发中，没有多任务和线程的概念，所有任务都在主循环中依次执行。裸机开发最适合对实时性要求较高的应用，但它的开发复杂度较高，开发者需要掌握硬件细节并进行低级编程。

4.2 基于 Linux 的应用开发

与裸机开发不同，基于 Linux 的应用开发使用操作系统来管理硬件资源，是一种通过 Linux 提供的 API 和库函数来访问系统资源和实现应用逻辑。在 Linux 系统上，程序被分解成多个任务，并由操作系统的调度器根据任务的优先级进行分时调度。通过操作系统提供的系统调用，程序能够以一种抽象化的方式访问硬件资源，而不需要直接操作硬件寄存器。

基于 Linux 的开发使得应用程序的开发更加简便，可以利用操作系统提供的

多任务管理、内存保护、文件系统和 I/O 设备管理等功能。在 Linux 上开发应用程序时，开发者不需要担心硬件的具体实现细节，只需专注于业务逻辑的实现。这大大简化了开发过程，提高了开发效率。例如，在开发网络服务、图形界面应用程序、命令行工具等各类应用时，Linux 系统提供的丰富功能和强大生态能够助力开发者快速构建出功能完善、性能稳定的应用产品。

4.3 硬启动与软启动的区别

在嵌入式系统中，硬启动和软启动是两种常见的启动方式，它们在启动的过程和方式上有很大区别。

硬启动通常是通过开发板的 Bootloader（如 U-Boot）直接加载内核映像到系统内存中，启动 Linux 操作系统。在硬启动过程中，开发板会从预先配置的存储介质（如 NAND Flash）中读取操作系统内核并执行。

软启动则是通过串口调试或其他接口工具加载内核镜像。在软启动过程中，内核会通过调试工具进行加载，而不依赖于开发板的 Bootloader。软启动通常用于调试阶段，方便开发者查看内核启动过程中的调试信息，定位问题。

在本次实习中，我们首先通过软启动加载内核镜像到开发板上，并通过串口终端观察内核的启动信息。软启动过程中能够实时查看系统启动过程中的输出。软启动成功后，我们切换到硬启动模式，最终通过开发板的 Bootloader 加载内核，完成了 Linux 系统的最终启动。

5. 总结

通过本次 Linux 系统体验实习，我们不仅加深了对嵌入式 Linux 系统的理解，还实践了 Linux 内核的移植、驱动开发和应用程序的开发。学会了如何在嵌入式平台上配置和编译 Linux 内核，并通过驱动程序与硬件交互，完成了多个硬件设备的移植。在未来的嵌入式系统开发中，我们将继续深入学习 Linux 内核，提升自己的开发能力，并将所学知识应用到更复杂的项目中。

第三部分：实习意见和建议

本次实习的第二阶段中，我们在学习 TQ2440 开发板更多功能时遇到了很大的困难。由于该板子年代较为久远，在官网上已无法找到相关信息，并且开发者论坛上关于这款板子的讨论基本集中在 09 至 12 年，网站的更新和维护都已停止。此外，开发手册中提供的例程也十分有限。我们尝试了耳机、录音、摄像头等功能的例程测试，结果只有耳机能够正常运行，其他外设（如串口、摄像头等）都没有具体的使用说明或驱动支持。

在串口功能的使用上，缺乏有效的教程和资料让我在结合硬件进行开发时步履维艰。由于串口的使用方法不明确，无法与硬件进行联动开发，只能转向类似平板电脑功能的简单软件开发，限制了我们对硬件与软件结合开发的深入学习和实践。这个问题不仅浪费了大量时间，还使我们对系统开发的全貌和技术细节了解得不够深入。如果可以更换成更新的开发板，比如主流的嵌入式开发板，不仅可以从网络上获取更丰富的资料和支持，还能通过学习更多实时更新的教程提升技能，这样的实习体验无疑会更加高效和实用。

在第三部分的 Linux 系统移植任务中，我们大部分时间都花在了按照指导文件逐步配置系统上。由于任务设置比较机械化，没有为我们提供自主思考和尝试的空间，整个过程更像是照葫芦画瓢，而不是基于理解的操作。这导致我们对系统移植的原理和细节缺乏深入认知，无法明白为什么要进行某些配置以及是否存在更好的解决方案。整个学习过程中，系统移植的知识获取显得浅尝辄止，更别提从中得到实践创新的机会。因此，我们认为如果能够在指导文件中适当增加原理讲解和灵活性任务，鼓励学生对移植过程进行探索和改进，将有助于提升我们的实际工程能力和问题解决能力。