**华中科技大学**

**网络空间安全学院**

**《OpenHarmony操作系统原理》实验报告**

姓 名：

班 级：

学 号：

联系方式：

分 数：

**目 录**

**[一、](#_Toc168217198)****[实验概述](#_Toc168217198)** [1](#_Toc168217198)

**[1.1](#_Toc168217199)****[实验名称](#_Toc168217199)** [1](#_Toc168217199)

**[1.2 实验目的](#_Toc168217200)** [1](#_Toc168217200)

**[1.3](#_Toc168217201)****[实验环境](#_Toc168217201)** [1](#_Toc168217201)

**[1.4](#_Toc168217202)****[实验内容](#_Toc168217202)** [1](#_Toc168217202)

**[1.5](#_Toc168217203)****[实验报告要求](#_Toc168217203)** [1](#_Toc168217203)

**[二、实验一过程记录](#_Toc168217204)** [3](#_Toc168217204)

**[2.1 实验步骤记录](#_Toc168217205)** [3](#_Toc168217205)

**[2.2 遇到的问题](#_Toc168217206)** [4](#_Toc168217206)

**[三、实验二过程记录](#_Toc168217207)** [7](#_Toc168217207)

**[3.1](#_Toc168217208)****[实验步骤记录](#_Toc168217208)** [7](#_Toc168217208)

**[3.2 遇到的问题](#_Toc168217209)** [12](#_Toc168217209)

**[四、实验总结](#_Toc168217210)** [14](#_Toc168217210)

**[4.1 实验感想](#_Toc168217211)** [14](#_Toc168217211)

**[4.2 意见和建议](#_Toc168217212)** [14](#_Toc168217212)

**OpenHarmony操作系统原理实验**

## **实验概述**

## **实验名称**

OpenHarmony操作系统原理实验。

## **实验目的**

学习OpenHarmony1.1.3系统的基本框架，使用Pegasus智能家居开发套件进行简单开发，完成一个简单的温湿度实时监测屏显功能。

## **实验环境**

操作系统：Windows/Linux

编程语言：C，gn

操作环境：VSCode

## **实验内容**

完成虚拟机环境、VSCode环境、驱动程序的安装配置。

完成镜像烧录，学习端口调试的方法。

编写HelloWorld程序，了解OpenHarmony系统接口调用方法。

学习新增组件和产品解决方案。

学习Pegasus智能家居开发套件OLED屏幕的使用方法和操作逻辑。

学习Pegasus智能家居开发套件传感器的使用方法和操作逻辑。

学习Pegasus智能家居开发套件WIFI模块的使用方法和操作逻辑。

使用API从网络获取数据。

使用共享变量在进程间传递数据。

额外功能的实现，将视具体情况予以一定加分。

## **实验报告要求**

要对实验步骤进行图片展示和相应的文字说明，必须包含关键步骤和结果，文本中需要对图片进行引用（如图x.x所示）。

提交实验设计报告和源代码；实验设计报告必须包括程序流程图，源代码必须加详细注释。

实验设计报告需提交纸质档和电子档，源代码、编译说明需提交电子档。

基于自己的实验设计报告，通过实验课的上机试验，将源代码编译成功，运行演示给实验指导教师检查。

## **二、实验一过程记录**

## **2.1 实验步骤记录**

1. Hello World编译

用VSCode进入1.1.3目录下,使用终端-新建终端启动命令行。使用hb set命令指定编译板,上下键选中wifiiot\_hispark\_pegasus,Enter确定。

在applications\sample\wifi-iot\app\startup目录下,编写源码hello.c,使用SYS\_RUN()调用,输出Hello World!字符串。

1. #include <stdio.h>
2. #include "ohos\_init.h"
4. **void** hello(**void**)
5. {
6. printf("Hello world!\r\n");
7. }
8. SYS\_RUN(hello);

编写编译脚本

applications\sample\wifi-iot\app\startup\BUILD.gn

1. static\_library("hello\_world") {
2. sources = [ "hello.c" ]
3. include\_dirs = [
4. "//utils/native/lite/include",
5. ]
6. }

applications\sample\wifi-iot\app\BUILD.gn

1. import("//build/lite/config/component/lite\_component.gni")
3. lite\_component("app") {
4. features = [
5. "startup:hello\_world",
6. ]
7. }

使用hb build -f命令执行编译。

在VSCode中,找到输出文件out\hispark\_pegasus\wifiiot\_hispark\_pegasus\

Hi3861\_wifiiot\_app\_allinone.bin

使用HiBurn烧录到开发板上（如图2-1所示）。

使用串口工具连接调试开发板，查看开发板输出（如图2-2所示）。

开发板在启动时成功输出Hello World！字符串。

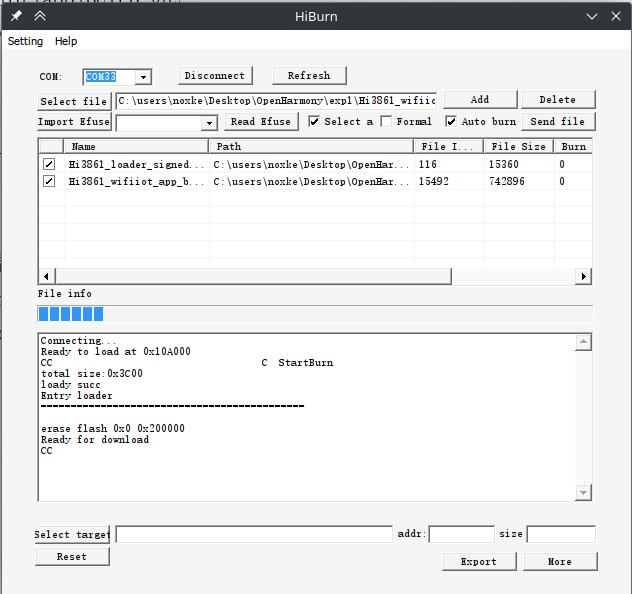


图2-1 HiBurn烧录镜像

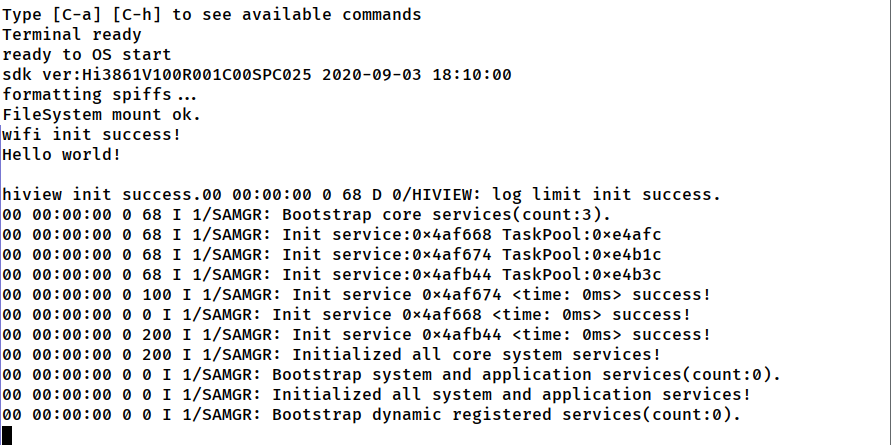


图2-2 串口工具调试开发板

1. 新增组件

创建目录applications\sample\component\_demo,在目录下编写demo.c、BUILD.gn文件。

demo.c

1. #include <stdio.h>
3. #include "ohos\_init.h"
5. **void** entry(**void**)
6. {
7. printf("I am a component.\r\n");
8. }
10. SYS\_RUN(entry);

BUILD.gn

1. static\_library("TestComponent") {
2. sources = [ "demo.c" ]
4. include\_dirs = [
5. "//utils/native/lite/include",
6. ]
7. }

使用hb build -f -T //applications/sample/component\_demo:TestComponent命令编译目标。

在build\lite\components\applications.json中添加组件定义。

1. {
2. "component": "TestComponent",
3. "description": "A test component",
4. "optional": "true",
5. "dirs": [
6. "applications/sample/component\_demo"
7. ],
8. "targets": [
9. "//applications/sample/component\_demo:TestComponent"
10. ],
11. "adapted\_kernel": [ "liteos\_m" ]
12. },

将组件TestComponent配置到产品,vendor\hisilicon\hispark\_pegasus\config.json

1. {
2. "subsystem": "applications",
3. "components": [
4. { "component": "wifi\_iot\_sample\_app", "features":[] },
5. { "component": "TestComponent", "features":[] }
6. ]
7. },

单独编译组件hb build -f TestComponent。

编译产品hb build -f。

烧录固件检查输出（如图2-3所示）。

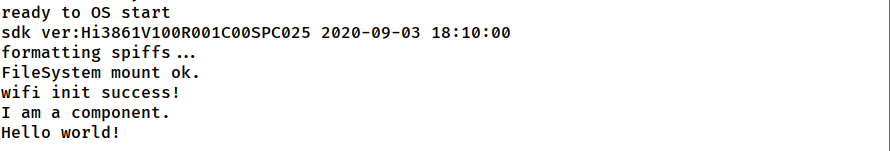


图2-3 新建组件调试输出

1. 新增产品解决方案

新建目录vendor\hust\product1

将目录product1中的文件传入虚拟机

切换到新产品hust\_product1

hb set

编译产品

hb build -f

将out/hispark\_pegasus/hust\_product1/Hi3861\_wifiiot\_app\_allinone.bin烧录到开发板上。

## **2.2 遇到的问题**

在配置BUILD.gn脚本时，对于include目录中的源码根目录、相对路径、绝对路径理解错误，导致在编译过程中发生错误，查看教程文档及示例代码了解到使用“//”表示源码根目录，而非使用绝对路径表示源码根目录，修改相应的错误后，成功通过编译。

新增组件的实验中，编写了组件源码、编译脚本，并且在build\lite\components\

applications.json中配置了组件，组件能够正常编译，但是在编译烧录后，发现新增组件并未成功运行，无预期输出。反复排查后发现没有按要求将组件配置到产品，在vendor\hisilicon\hispark\_pegasus\config.json中添加相应配置后，重新编译产品并烧录，新增组件成功按照预期运行和输出。

## **三、实验二过程记录**

## **实验步骤记录**

1. 环境及项目配置

修改 /device/hisilicon/hispark\_pegasus/sdk\_liteos/build/config/usr\_config.mk 文件。

将 # CONFIG\_I2C\_SUPPORT is not set 修改为 CONFIG\_I2C\_SUPPORT=y 。

开启I2C支持，否则编译过程中会产生错误报错:\*undefined reference to hi\_i2c\_\*\*。

本实验将实现一个简单的桌面摆件,展示时间、温度与湿度。需要实现三个模块,与项目模板对应。项目目录如下：

dragon@vmubuntu20:~/openharmony/1.1.3/applications/myapp$ tree

.

└── app

    ├── BUILD.gn

    ├── network   # 网络模块,连接wifi并获取时间

    │   ├── BUILD.gn

    │   ├── get\_online\_time.c

    │   └── wifi\_connect.c

    ├── oled   # 显示模块,在屏幕上绘制时间,温度与湿度

    │   ├── BUILD.gn

    │   ├── gif2code.py

    │   ├── madoka.gif

    │   ├── madoka.h

    │   └── oled.c

    ├── sensor   # 传感器模块,获取温度与湿度

    │   ├── aht20.c

    │   ├── aht20.h

    │   ├── BUILD.gn

    │   └── temp-humi\_sensor.c

    ├── shared\_variable   # 用于组件间共享变量

    │   └── shared\_variable.h

    ├── ssd1306\_3rd\_driver   # 显示器的驱动

    └── startup

        ├── BUILD.gn

        └── helloworld.c

项目编译脚本BUILD.gn

1. import("//build/lite/config/component/lite\_component.gni")
3. lite\_component("app") {
4. features = [
5. "startup",
6. "ssd1306\_3rd\_driver/ssd1306:oled\_ssd1306",
7. "ssd1306\_3rd\_driver/libm\_port:libm\_port",
8. "oled",
9. "network",
10. "sensor",
11. ]
12. }

在build\lite\components\applications.json中添加组件定义MyAPP。

1. {
2. "component": "MyAPP",
3. "description": "my openharmony application",
4. "optional": "true",
5. "dirs": [
6. "applications/myapp/app"
7. ],
8. "targets": [
9. "//applications/myapp/app"
10. ],
11. "adapted\_kernel": [ "liteos\_m" ]
12. },

将组件MyAPP配置到产品,vendor\dragon/product1/config.json。

1. {
2. "subsystem": "applications",
3. "components": [
4. { "component": "MyAPP", "features":[] }
5. ]
6. },

使用hb set选择产品dragon\_product1。

1. 共享变量

shared\_variable 文件夹下不需要 BUILD.gn 文件。

在C语言中可以使用 extern 关键字在不同的文件中共享变量。

1. 在一个头文件(例如 shared\_variable.h )中使用 extern 关键字声明这个变量。

2. 在一个文件(例如 get\_online\_time.c )中定义变量,设置初值或写入当前值。

3. 在需要使用这个变量的其他文件(例如 oled.c )中包含这个头文件,代码中直接使用即可。

1. // shared.h
2. #ifndef SHARED\_H
3. #define SHARED\_H
4. #include "time.h"
6. **extern** **time\_t** shared\_variable\_time\_offset;
7. **extern** **float** shared\_variable\_temperature;
8. **extern** **float** shared\_variable\_humidity;
10. #endif // SHARED\_H
11. 网络-network

创建WifiConnectTask线程，连接无线热点，成功连接后定期调用get\_online\_time从互联网更新时间。

1. strcpy(apConfig.ssid, "HarmonyOS 13");
2. strcpy(apConfig.preSharedKey, "1111111111");
3. apConfig.securityType = WIFI\_SEC\_TYPE\_PSK;
4. errCode = EnableWifi();；
5. errCode = AddDeviceConfig(&apConfig, &netId);
6. g\_connected = 0;
7. errCode = ConnectTo(netId);
9. // 等待连接成功
10. **while** (!g\_connected)
11. {
12. osDelay(10);
13. }
14. **struct** netif \*iface = netifapi\_netif\_find("wlan0");
16. // 获取在线时间
17. get\_online\_time();

编译产品并烧录，如图3-1所示，开发板成功连接到网络热点，并且从互联网成功获取时间更新。

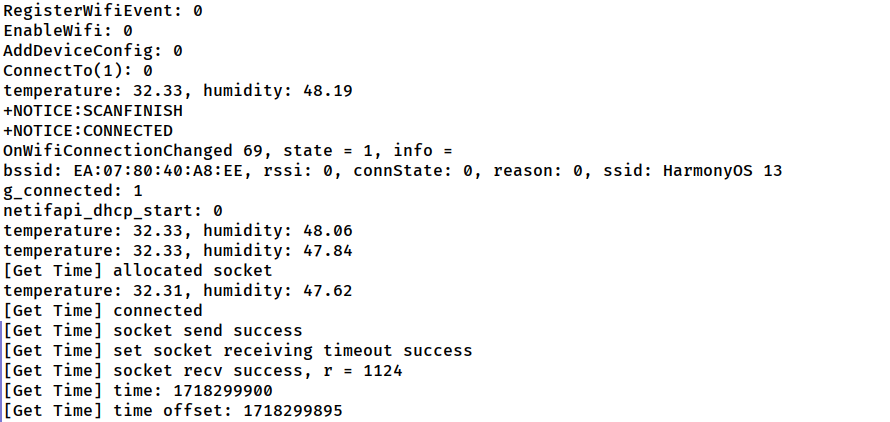


图3-1 网络连接及时间更新

1. 显示-oled

参考 案例程序\applications\sample\wifi-iot\app\ssd1306\_3rd\_driver\examples 。板载显示器为ssd1306,这是一块128x64像素的双色oled屏幕。

在显示线程Ssd1306Task中，循环执行屏幕刷新操作，使用DrawTme绘制时间到屏幕，使用DrawSensor绘制传感器数据。

DrawTime使用共享变量中的shared\_variable\_time\_offset与开发板运行时间计算得到当前的时间戳，再将时间格式化显示到屏幕指定位置。

1. **time\_t** time\_raw;
2. **char** time\_date[20];
3. **char** time\_time[20];
4. **struct** **tm** \*local\_time;
5. **void** DrawTime(**void**)
6. {
7. // 格式化时间 +8时区
8. time(&time\_raw);
9. time\_raw = time\_raw + shared\_variable\_time\_offset + 8 \* 3600;
10. local\_time = localtime(&time\_raw);
11. snprintf(time\_date, **sizeof**(time\_date),
12. "%04d-%02d-%02d %s",
13. local\_time->tm\_year + 1900,
14. local\_time->tm\_mon + 1,
15. local\_time->tm\_mday,
16. (local\_time->tm\_wday == 0) ? "Sun" :
17. (local\_time->tm\_wday == 1) ? "Mon" :
18. (local\_time->tm\_wday == 2) ? "Tue" :
19. (local\_time->tm\_wday == 3) ? "Wed" :
20. (local\_time->tm\_wday == 4) ? "Thu" :
21. (local\_time->tm\_wday == 5) ? "Fri" :
22. "Sat");
23. snprintf(time\_time, **sizeof**(time\_time),
24. "%02d:%02d:%02d",
25. local\_time->tm\_hour,
26. local\_time->tm\_min,
27. local\_time->tm\_sec);
28. // 设置显示位置
29. ssd1306\_SetCursor(20, 0);
30. ssd1306\_DrawString(time\_time, Font\_11x18, White);
31. ssd1306\_SetCursor(15, 20);
32. ssd1306\_DrawString(time\_date, Font\_7x10, White);
33. }

DrawSensor从共享变量中shared\_variable\_temperature和shared\_variable\_humidity获取环境温度与适度，并将温湿度信息格式化地绘制到屏幕。

1. **char** temperature\_str[20];
2. **char** humidity\_str[20];
3. **void** DrawSensor(**void**)
4. {
5. //需要补全
6. sprintf(temperature\_str, "Temperature: %.2f", shared\_variable\_temperature);
7. sprintf(humidity\_str, "Humidity: %.2f%", shared\_variable\_humidity);
8. // 设置显示位置
9. ssd1306\_SetCursor(0, 36);
10. ssd1306\_DrawString(temperature\_str, Font\_7x10, White);
11. ssd1306\_SetCursor(0, 48);
12. ssd1306\_DrawString(humidity\_str, Font\_7x10, White);
13. }

屏幕显示效果如图3-2所示。

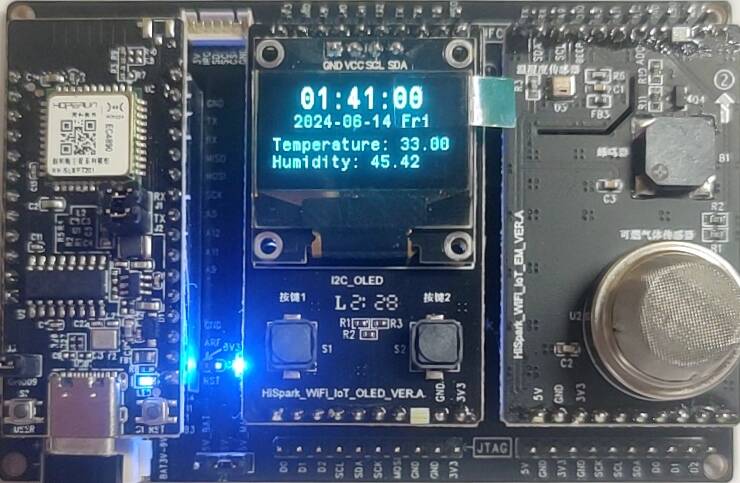


图3-2 屏幕显示效果

1. 传感器-sensor

参考 案例程序\applications\sample\wifi-iot\app\i2c\_demo 。

板载温湿度传感器为AHT20,使用I2C机制通信进行数字输出,使用该传感器获取温度和湿度以供显示。

在tempHumiTask线程中，从AHT20传感器获取温湿度信息，将温湿度储存在共享变量中，在显示线程中由DrawSensor将传感器数据绘制到屏幕。

1. 开机动画

将gif动图进行逐帧分割，将每一帧图像转换为对应的数组储存在代码中，利用ssd1306\_DrawBitmap实现将像素画绘制到屏幕。

在对图片进行处理时，使用PIL将gif动图拆分得到每一帧图像，利用cv2将图像缩放到适合屏幕显示的尺寸，再将图片转换为灰度图，根据每个像素的灰度信息将每8个像素组合为一个字节，编码为数组保持到源码中。

由于屏幕像素较低，且只支持一个灰阶，只进行简单的灰度处理时屏幕显示效果较差，因此对灰度图像再进行抖动处理，得到适合低像素屏幕显示的图像。

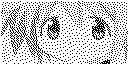


图3-3 图像处理

在显示线程Ssd1306TestTask启动时，首先运行DrawMadoka函数，实现开机动画的绘制，在DrawMadoka中，利用ssd1306\_DrawBitmap依次将动图每一帧对应的数组进行绘制，得到连续的动图播放效果。动图显示效果如图3-4所示。

1. #include "madoka.h"
2. // 画小圆
3. **const** unsigned **char** \*frameDatas[] = {frame0Data, frame1Data, frame2Data, frame3Data, frame4Data, frame5Data, frame6Data, frame7Data, frame8Data, frame9Data, frame10Data};
4. **const** **int** frameNum = 11;
5. **void** DrawMadoka(**void**)
6. {
7. **for** (**int** i = 0; i < 50; i++)
8. {
9. ssd1306\_DrawBitmap(frameDatas[i%frameNum], frameDatas[i%frameNum]);
10. ssd1306\_UpdateScreen();
11. osDelay(10);
12. }
13. }

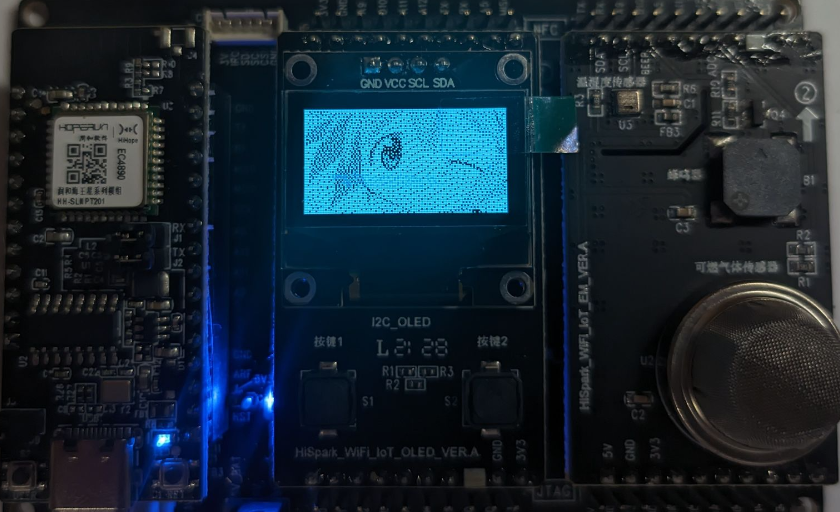


图3-4 动图显示效果

## **遇到的问题**

在第二次实验中，在网络时间获取时，错误的将shared\_variable\_time\_offset当作当前时间，单独创建了线程来刷新shared\_variable\_time\_offset时间，但测试发现每次调用get\_online\_time时，开发板时间都会回到开机时的时间，分析get\_online\_time后发现shared\_variable\_time\_offset指的是开发板时间time()与实际时间的差值，应该利用time()加上该差值获取准确时间。

在处理开发板开机动画时，在图像处理上最开始只使用了简单的灰度处理，导致图片显示不明显，无法准确分辨原图像，在增加抖动处理时，直接对原图灰度后进行抖动处理，抖动处理后再对图像进行缩放，导致缩放后抖动处理产生的黑点过于明显，显示效果仍然很差。调整处理顺序后，先对图像进行缩放，接着灰度，最后再进行抖动处理，多次调整参数后，得到了较为满意的显示表现。

## **四、实验总结**

## **4.1 实验感想**

通过此次OpenHarmony操作系统原理实验，我不仅深入了解了OpenHarmony系统的基本框架，还掌握了Pegasus智能家居开发套件的使用方法，成功实现了温湿度实时监测屏显功能。

实验的初期阶段，通过配置虚拟机环境、VSCode环境及相关驱动程序，完成了镜像烧录和端口调试，并编写了HelloWorld程序。熟悉了开发环境，学习了如何在OpenHarmony中新增组件和配置产品解决方案，并成功将新建的组件编译并烧录到开发板上。

通过学习OLED屏幕、传感器以及WIFI模块的操作方法，编写了相关代码，实现了数据的获取与显示。实现了无线热点的连接与时间的获取，并通过共享变量在进程间传递数据，确保了数据的同步与更新。在OLED屏幕的操作中，我掌握了屏幕刷新、时间显示以及传感器数据绘制的方法。

通过对图像处理方法的学习与实践，我实现了开机动画的效果展示。尽管在处理过程中遇到了一些问题，但通过不断调整参数和优化处理流程，最终得到了满意的显示效果。

在本次实验中，我深刻认识到独立思考和解决问题的重要性。在遇到问题时，我积极查阅相关资料，通过不断尝试和调整，最终找到了合适的解决方案。这不仅提升了我的技术能力，也增强了我的自信心和耐心。

通过此次实验，我不仅收获了丰富的技术经验，还在独立思考、问题解决和耐心方面得到了全面提升。这次实验不仅是对我技术能力的提升，更是对我综合素质的全面锻炼，为我未来的职业发展打下了坚实的基础。在此，我要特别感谢老师和助教的指导和帮助，让我在实验过程中少走了许多弯路，顺利完成了实验任务。

## **4.2 意见和建议**

此次OpenHarmony实验内容丰富，涵盖了操作系统框架、开发环境配置、组件添加、传感器使用等多个方面，极大地拓展了学生的知识面。然而，实验步骤较为复杂，特别是对于初学者而言，可能会在环境配置和调试过程中遇到较多困难。建议在实验指导中增加更多详细的操作步骤和常见问题的解决方案。此外，可以增加一些基础知识的讲解环节，帮助学生更好地理解每一步操作的原理和目的。总体而言，此次实验具有很高的教学价值，但在细节指导上还可以进一步完善。

|  |
| --- |
| 原创性声明 |
| 本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。  已阅读并同意以下内容。  判定为不合格的一些情形：  （1） 请人代做或冒名顶替者；  （2） 替人做且不听劝告者；  （3） 实验报告内容抄袭或雷同者；  （4） 实验报告内容与实际实验内容不一致者；  （5） 实验代码抄袭者。  **作者签名：** |