《信息技术与工程开放实验B》

实 验 指 导 书

计算机与通信学院

目　录

第一章 基础知识

1.1 HarmonyOS 应用开发

1.1.1 HarmonyOS 简介

HarmonyOS 是一款面向万物互联时代的、全新的分布式操作系统。

在传统的单设备系统能力基础上，HarmonyOS 提出了基于同一套系统能力、适配多种终端形态的分布式理念，能够支持手机、平板、智能穿戴、智慧屏、车机等多种终端设备，提供全场景（移动办公、运动健康、社交通信、媒体娱乐等）业务能力。

HarmonyOS 有三大特征：

1、搭载该操作系统的设备在系统层面融为一体、形成超级终端，让设备的硬件能力可以弹性扩展，实现设备之间硬件互助，资源共享。

2、对消费者而言，HarmonyOS 能够将生活场景中的各类终端进行能力整合，实现不同终端设备之间的快速连接、能力互助、资源共享，匹配合适的设备、提供流畅的全场景体验。

3、面向开发者，实现一次开发，多端部署。

对应用开发者而言，HarmonyOS 采用了多种分布式技术，使应用开发与不同终端设备的形态差异无关，从而让开发者能够聚焦上层业务逻辑，更加便捷、高效地开发应用。

1.1.2 应用程序包

HarmonyOS 的用户应用程序包以 APP Pack（Application Package）形式发布，它是由一个或多个 HAP（HarmonyOS Ability Package）以及描述每个 HAP 属性的 pack.info 组成。HAP 是 Ability 的部署包，HarmonyOS 应用代码围绕 Ability 组件展开。

一个 HAP 是由代码、资源、第三方库及应用配置文件组成的模块包，可分为 entry 和 feature 两种模块类型。Entry 是应用的主模块。

1.1.2.1 Ability

Ability 是应用所具备的能力的抽象，一个应用可以包含一个或多个 Ability。Ability 分为两种类型：FA（Feature Ability）和 PA（Particle Ability）。FA 有 UI 界面，提供与用户交互的能力，而 PA 无 UI 界面，提供后台运行任务的能力以及统一的数据访问抽象。

1.1.2.2 库文件

库文件是应用依赖的第三方代码（例如 so、jar、bin、har 等二进制文件），存放在 libs 目录。

1.1.2.3 资源文件

应用的资源文件（字符串、图片、音频等）存放于 resources 目录下，便于开发者使用和维护。

1.1.2.4 配置文件

配置文件 (config.json)用于声明应用的 Ability，以及应用所需权限等信息。

1.1.3 开发工具

开发者可以使用 HUAWEI DevEco Studio 开发 HarmonyOS 用户应用程序。HUAWEI DevEco Studio 是面向华为终端全场景多设备的一站式集成开发环境（IDE）。

HarmonyOS 提供了两种 FA（Feature Ability）的 UI 开发框架：Java UI 框架和方舟开发框架。

Java UI 框架提供了细粒度的 UI 编程接口，UI 元素更丰富，使应用开发更加灵活。

方舟开发框架提供了相对高层的 UI 描述，使应用开发更加简单。

比较项	Java UI 框架	方舟开发框架	
语言生态	Java	JS	eTS
接口方式	命令式	类 Web 范式	声明式
执行方式	开发者处理，基于 API 驱动的 UI 变更	框架层处理，基于数据驱动的 UI 自动变更	框架层处理，基于数据驱动的 UI 自动变更
相对优势	UI 元素更丰富，开发更灵活	轻量化，开发更简便	极简开发，内存占用更少、运行性能更高

1.2 传感器开发步骤

1.2.1 配置权限

使用某些传感器（比如位置传感器），需要请求相应的权限，开发者才能获取到传感器数据。

使用位置传感器，需要在 config.json 中，在 "abilities"同级下面添加：

```
"reqPermissions": [ { "name": "ohos.permission.LOCATION" } ]
```

1.2.2 检查权限

由于敏感权限需要用户授权，因此，开发者在应用启动时或者调用订阅数据接口前，需要进行权限检查和请求用户授权。下面以位置权限为例：

```
@Override
public void onStart(Intent intent) {
  super.onStart(intent);
  super.setMainRoute(MainAbilitySlice.class.getName());

  // 测试应用没有被授予权限
  if (verifySelfPermission("ohos.permission.LOCATION") !=
          IBundleManager.PERMISSION_GRANTED) {
    // 测试是否能申请弹框授权(首次申请或者前面申请用户未选择禁止且不再提示)
```

```
        if (canRequestPermission("ohos.permission.LOCATION")) {
          //请求用户授权
          requestPermissionsFromUser(new String[]{ "ohos.permission.LOCATION" },
             MY_PERMISSIONS_REQUEST_GPS);
        }
     }
}
@Override
public void onRequestPermissionsFromUserResult (
   int requestCode, String[] permissions,int[] grantResults) {
       switch (requestCode) {
          case MY_PERMISSIONS_REQUEST_GPS: {
            if (grantResults.length > 0 && grantResults[0] ==
               IBundleManager.PERMISSION_GRANTED) { //已授权 }
             else {
               // 权限被拒绝
               new ToastDialog(getContext()).setText("位置权限被用户拒绝！").show();
            }
            return;
       }
   }
}
```

1.2.3 使用传感器

1、获取待订阅数据的传感器（以方向传感器为例）。

```
//新建方向代理类
categoryOrientationAgent = new CategoryOrientationAgent();
//获得方向传感器的实例
CategoryOrientation categoryOrientation =
categoryOrientationAgent.getSingleSensor(CategoryOrientation.SENSOR_TYPE_ORIENTATION);
```

2、创建传感器回调函数。

```
categoryOrientationDataCallback = new ICategoryOrientationDataCallback() {
  @Override
  public void onSensorDataModified(CategoryOrientationData categoryOrientationData){
   //获得方向角
   degree = categoryOrientationData.getValues()[0];
    ...
   }
   ...
```

```
};
```

3、订阅传感器数据。

```
categoryOrientationAgent.setSensorDataCallback(
categoryOrientationDataCallback, categoryOrientation,SAMPLING_INTERVAL_NANOSECONDS);
```

4、在回调函数的 onSensorDataModified 方法中接收并处理传感器数据。

5、取消订阅传感器数据。

```
categoryOrientationAgent.releaseSensorDataCallback(categoryOrientationDataCallback);
```

1.3 常见问题

1.3.1 vp、fp、px

1、虚拟像素单位：vp (virtual pixel)

以屏幕相对像素为单位，是一台设备针对应用而言所具有的虚拟尺寸（区别于屏幕硬件本身的像素单位）。它提供了一种灵活的方式来适应不同屏幕密度的显示效果,使用虚拟像素，使元素在不同密度的设备上具有一致的视觉体验。

2、字体像素单位：fp (font pixel)

字体像素(font pixel) 默认情况下与 vp 相同，即默认情况下 1 fp = 1vp。如果用户在设置中选择了更大的字体，字体的实际显示大小就会在 vp 的基础上乘以 scale 系数，即 1 fp = 1 vp * scale。

3、屏幕像素单位：px (pixel)

屏幕上的实际像素，1px 代表手机屏幕上的一个像素点，如果这个不怎么好理解,看下常见的手机比如:1080×1920，这个数值的单位都是 px,由于 px 在不同手机上的大小不同，差别较大，适配性太差，不建议使用,所以无论是 android 还是 HarmonyOS ，无论写距离大小还是字体大小,都不建议使用 px。

一般来说组件间的距离使用 vp (virtual pixel)，字体大小使用 fp (font pixel)。

1.3.2 本地模拟器启动报错的一种解决方案

本地模拟器启动报错，错误信息为：

错误

模拟器启动失败，查看处理指导。

首先关闭 Hyper-V，然后启用 CPU 的虚拟化选项，操作如下：

（1）关闭 Hyper-V

打开 “设置”>“应用”，点击“相关设置”文字下的“程序和功能”，然后点击“启动或关闭 Winodows 功能” ，找到并取消勾选“Hyper-V”，点击确定并重启电脑即可。

（2）启用 CPU 的虚拟化选项

进入电脑的 BIOS 中，将 CPU 的“Intel Virtualization Technology”选项开启。

1.3.3 事件使用步骤

（1）创建 EventHandler 的子类如 GPSEventHandler，在子类中重写方法 processEvent()来处理事件

（2）定义事件对象

```
InnerEvent event = InnerEvent.get(事件 ID, 参数, 附加数据 1);
```

（3）获取当前应用的主线程

```
EventRunner eventRunner = EventRunner.getMainEventRunner();
```

（4）定义事件处理对象，并绑定到主线程中

```
EventHandler eventHandler = new GPSEventHandler(eventRunner, 附加数据 2);
```

（5）把事件发送到事件处理对象中，其中方法 processEvent()进行事件处理

```
eventHandler.sendEvent(event);
```

1.3.4 线程

在启动应用时，系统会为该应用创建一个称为“主线程”的执行线程，是应用的核心线程。UI 界面的显示和更新等操作，都是在主线程上进行。主线程又称 UI 线程，默认情况下，所有的操作都是在主线程上执行。如果需要执行比较耗时的任务（如下载文件、查询数据库），可创建其他线程来处理。

UITaskDispatcher：绑定到 UI 线程的专有任务分发器， 由该分发器分发的所有的任务都是在主线程上按顺序执行，它在应用程序结束时被销毁。

1、使用 UITaskDispatcher 同步派发任务：

（1）创建 TaskDispatcher

```
TaskDispatcher uiTaskDispatcher = getUITaskDispatcher();
```

（2）同步派发任务

```
uiTaskDispatcher.syncDispatch(() ->{//要执行的任务语句});
```

2、使用线程池执行任务：

（1）参数信息：

```
int corePoolSize      核心线程数
int maximumPoolSize   线程池最大容量
long keepAliveTime    线程空闲时，线程存活的时间
TimeUnit unit         时间单位：TimeUnit.MILLISECONDS，TimeUnit.SECONDS
int QUEUE_SIZE        队列大小
BlockingQueue<Runnable> workQueue   任务队列如：new ArrayBlockingQueue<>(QUEUE_SIZE)
CommonThreadFactory threadFactory        线程工厂
```

（2）创建线程池：

```
ThreadPoolExecutor pool = new ThreadPoolExecutor(corePoolSize, maximumPoolSize,
      keepAliveTime, unit, workQueue, threadFactory);
```

（3）使用线程池执行任务：

```
ThreadPoolUtil.submit(() -> {//要执行的任务语句})
```

1.3.5 注册账号

进入华为开发者联盟官网 https://developer.huawei.com/consumer/cn/，点击“注册”进入注册页面，您可以通过电子邮箱或手机号码注册华为开发者联盟帐号，注册成功后将显示实名认证页面，推荐使用个人银行卡进行实名认证，认证成功后才可以使用远程模拟器或远程真机进行调试。具体操作可参看 https://developer.huawei.com/consumer/cn/doc/start/ibca-0000001062388135 的说明。

第二章 开放创新实验

2.1 运动健康监控实验

1、实验目的

使用 DevEco Studio 开发一个简单的运动健康监控程序。

2、实验设备

硬件：HarmonyOS 系统轻量级智能手表（可选）

软件：DevEco Studio 集成开发环境（一套）

3、实验内容

利用 DevEco Studio 开发一个简单的运动健康监控程序，实时获得运动步数、心率数据，然后将程序部署至模拟器或本地真机中。

4、实验预习要求

（1）获得运动步数的步骤：

```
//回调函数中的 this 不是现在程序所在位置的 this 对象，所以首先保存 this 为 that，
//以便在回调函数中通过 that 使用原 this
var that = this
//订阅计步传感器
sensor.subscribeStepCounter({
          //步数改变时的回调函数
            success: function(ret) {
                that.steps = ret.steps;
                console.log('get steps value:' + ret.steps);
            },
            //订阅失败时的回调函数
            fail: function(data, code) {
                console.log('Subscription failed. Code: ' + code + '; Data: ' + data);
            },
});
```

（2）获得心率的步骤和获得运动步数的步骤类似。

5、实验步骤

（1）启动 DevEco Studio。

（2）选择“Create Project”-> 选择”[Lite] Empty Ability”， 创建一个新项目“Demo1”，如下图：

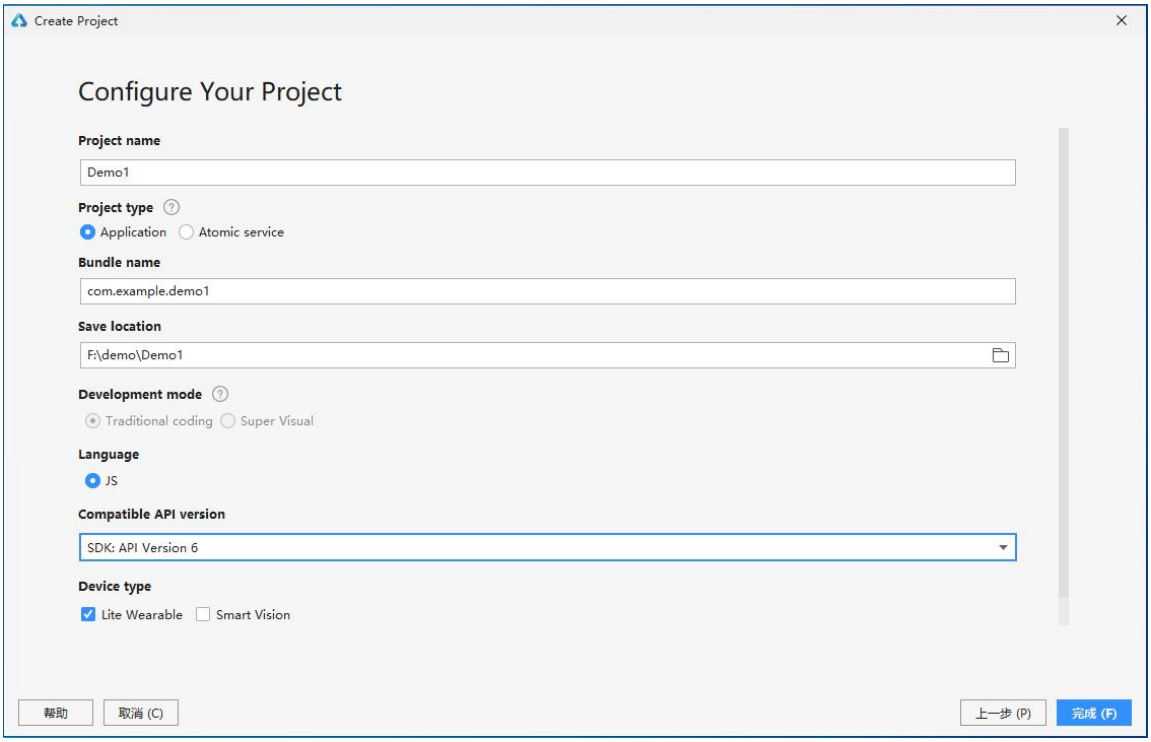

图 2.1.1 创建工程

(3) 程序目录结构如下：

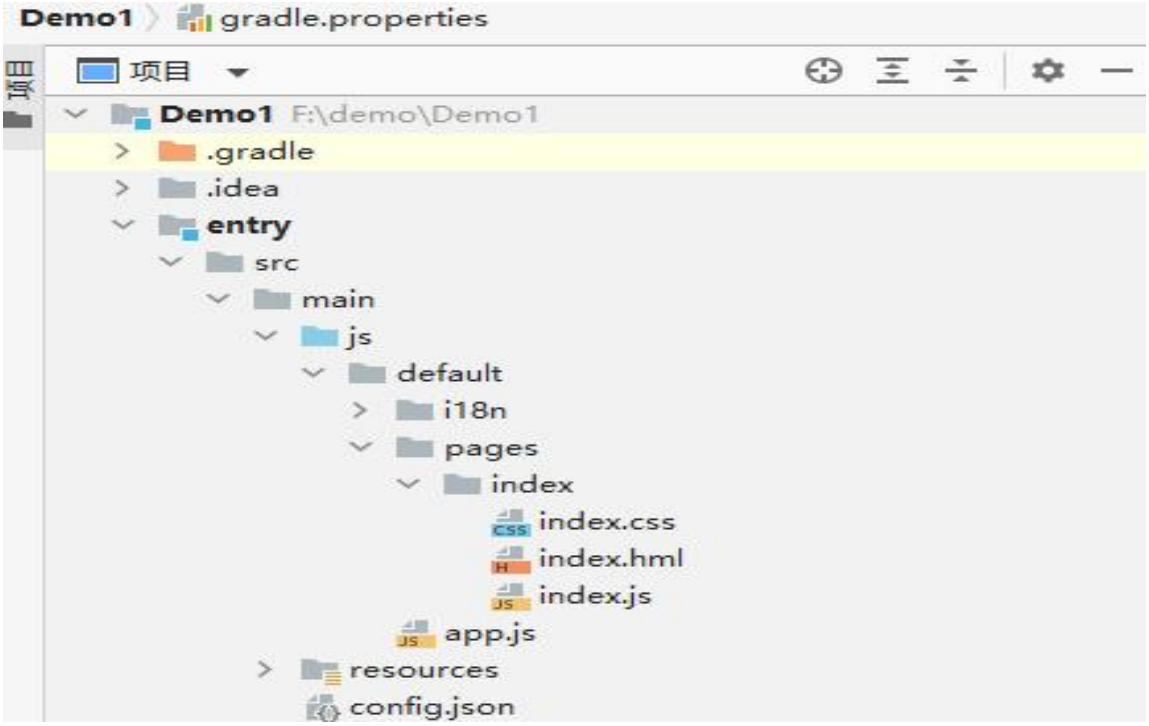

图 2.1.2 目录结构

(4) Index.hml 内容如下：

```
<div class="container">
  <text class="text">
    步数：{{ steps }}
```

```
    </text>
    <text class="text">
      心率：{{ heartRate }}
    </text>
  </div>
```

(5) Index.css 内容如下：

```
.container {
    flex-direction: column;
    width: 100%;
    height: 100%;
    justify-content: center;
    align-items: center;
}
.text {
   width: 400px;
   font-size: 30px;
   text-align: center;
}
```

(6) Index.js 内容如下：

```
import sensor from '@system.sensor';
export default {
    data: {
        steps: 0,
        heartRate: 0
    },
    onInit(){
        var that = this//回调函数中的 this 不是变量 steps 所在的 this 对象
        sensor.subscribeStepCounter({//订阅计步传感器
            success: function(ret) {//步数改变时的回调函数
                that.steps = ret.steps;
            },
            fail: function(data, code) {//订阅失败时的回调函数
                console.log('Subscription failed. Code: ' + code + '; Data: ' + data);
            },
        });
        sensor.subscribeHeartRate({//订阅心率传感器
            success: function(ret) {//心率改变时的回调函数
                that.heartRate = ret.heartRate;
            },
```

```
            fail: function(data, code) {//订阅失败时的回调函数
                console.log('Subscription failed. Code: ' + code + '; Data: ' + data);
            },
        });
    },
    onDestroy(){
        sensor.unsubscribeStepCounter();//取消订阅计步传感器
        sensor.unsubscribeHeartRate();//取消订阅心率传感器
    }
}
```

(7) 启动本地模拟器，运行结果如下：

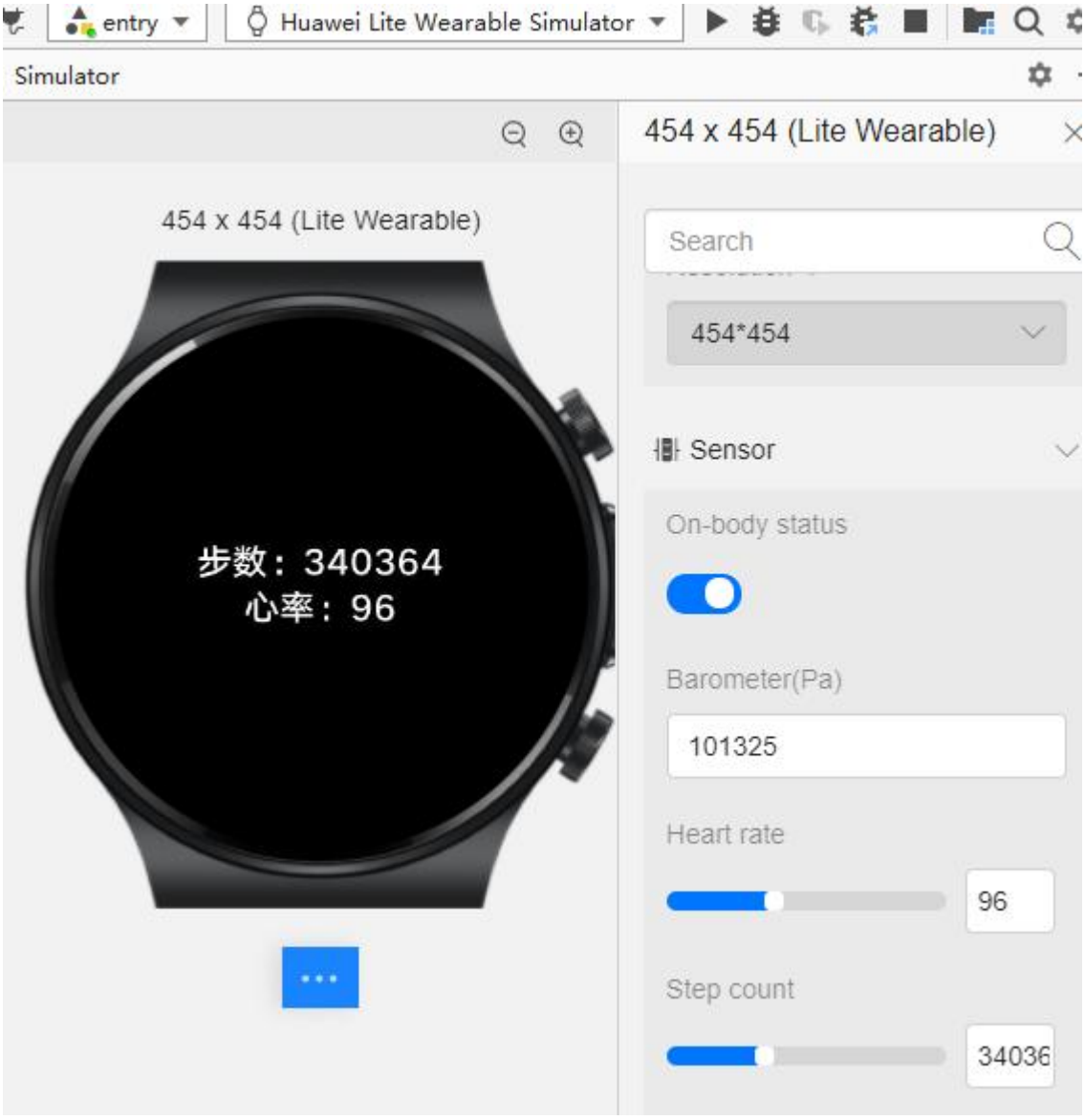

图 2.1.3 运动健康监控程序结果

2.2 指南针实验

1、实验目的

使用 DevEco Studio 开发一个简单的指南针程序。

2、实验设备

硬件：HarmonyOS 系统手机、智能手表（可选）

软件：DevEco Studio 集成开发环境（一套）

3、实验内容

利用 DevEco Studio 开发一个简单的指南针程序，获得方向角，转动指南针图像，然后将程序部署至模拟器或本地真机中。

4、实验预习要求

熟悉 DevEco Studio 进行 HarmonyOS 应用程序的开发过程，熟悉本地模拟器、远程模拟器或本地真机的使用。

使用方向传感器的步骤：

（1）实例化方向传感器代理并获得方向传感器

```
categoryOrientationAgent = new CategoryOrientationAgent();
CategoryOrientation categoryOrientation =
categoryOrientationAgent.
  getSingleSensor(CategoryOrientation.SENSOR_TYPE_ORIENTATION);
```

（2）定义方向传感器回调函数

```
categoryOrientationDataCallback = new ICategoryOrientationDataCallback() {
@Override
public void onSensorDataModified(CategoryOrientationData categoryOrientationData){
  degree = categoryOrientationData.getValues()[0];//获得方向角
  handler.sendEvent(0);//发送事件，用事件处理可增强界面交互的流畅度
}
@Override
public void onAccuracyDataModified(CategoryOrientation categoryOrientation, int i)
{}
@Override
public void onCommandCompleted(CategoryOrientation categoryOrientation) {}
};
```

（3）为方向传感器设置回调函数

```
  categoryOrientationAgent.setSensorDataCallback(
    categoryOrientationDataCallback, categoryOrientation,
              SAMPLING_INTERVAL_NANOSECONDS);
  }
```

（4）取消方向传感器的回调函数

```
categoryOrientationAgent.releaseSensorDataCallback(
    categoryOrientationDataCallback);
```

5、实验步骤

（1）启动 DevEco Studio。

（2）选择“Create Project”-> 选择”Empty Ability”， 创建一个新项目“Demo2”，如下图：

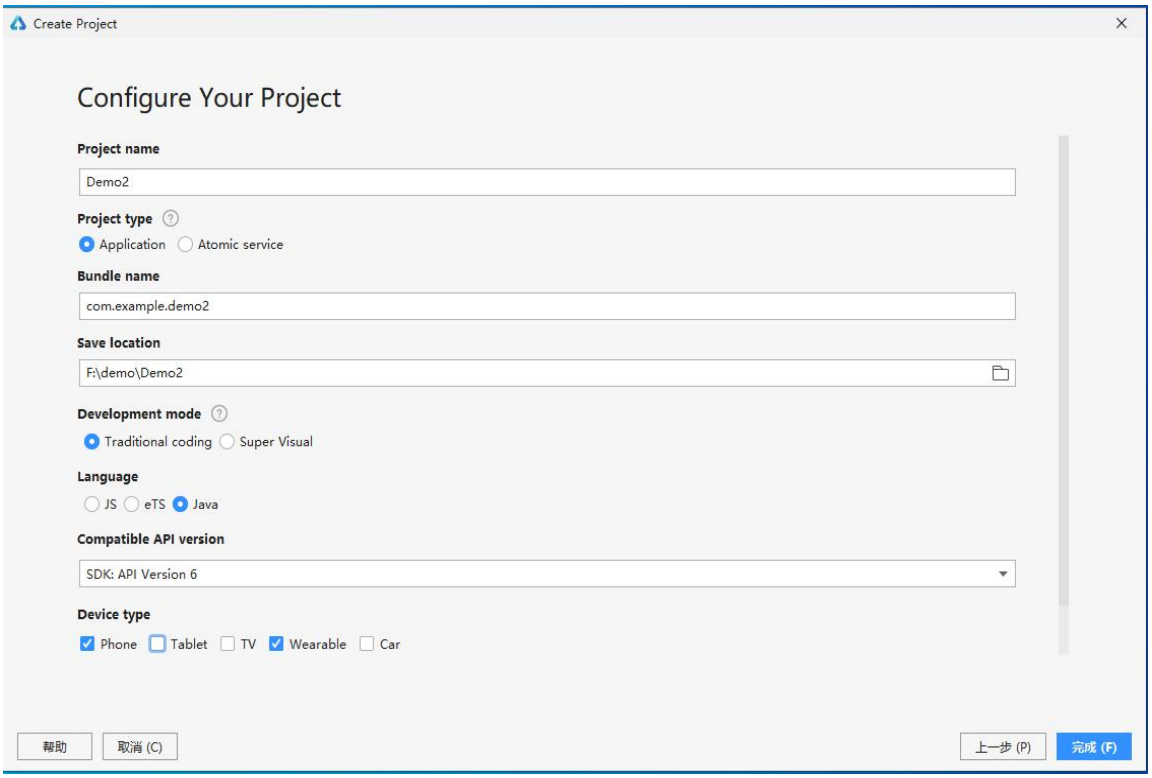

图 2.2.1 创建工程

（3）程序目录结构如下：

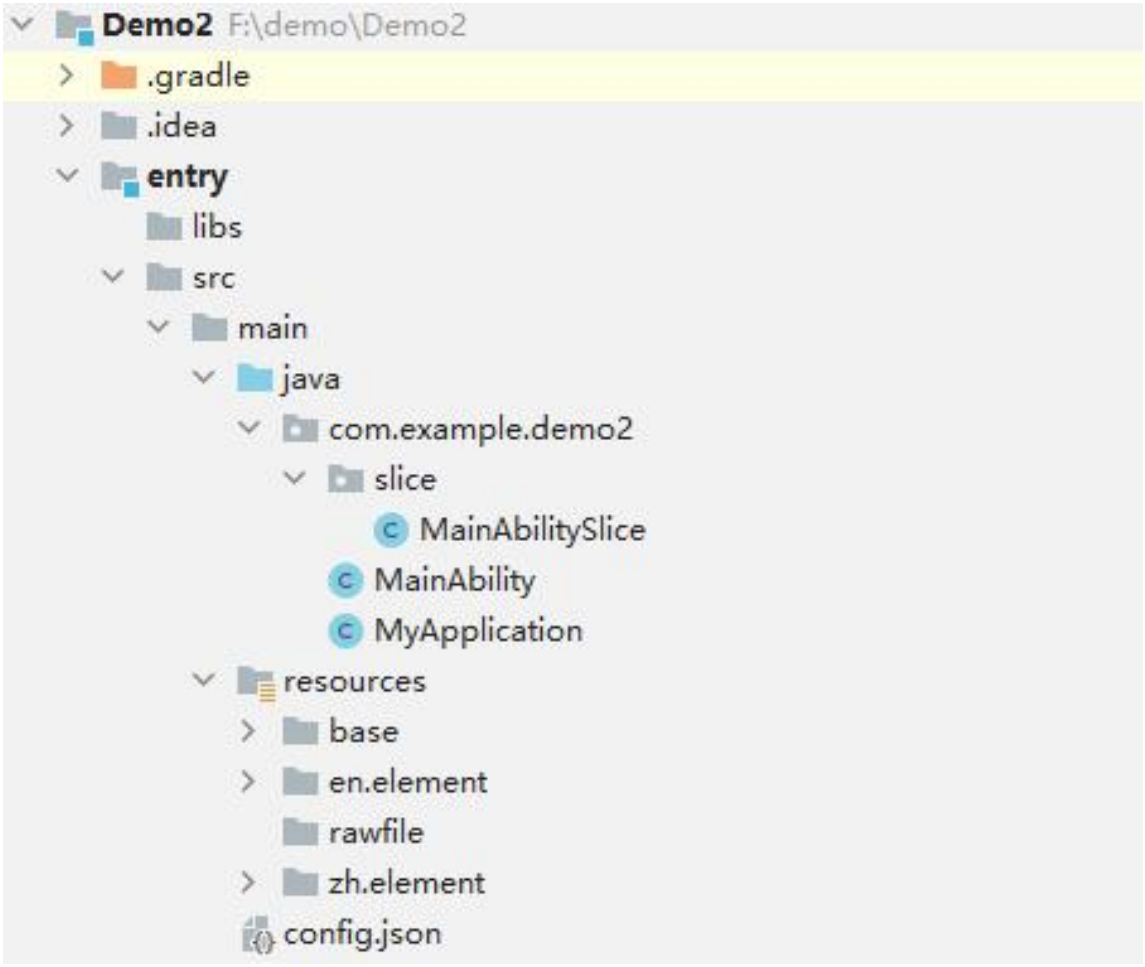

图 2.2.2 目录结构

（4）为了编程方便，删除 en.element 和 zh.element（为了国际化）（这一步可忽略）。

（5）把指南针图像 compass.png 放在 resources/base/media 下。

（6）修改 resources/base/layout/ability_main.xml，只保留显示指南针图像的组件

```
<Image
    ohos:id="$+id:compass_icon_img"
    ohos:height="300vp"
    ohos:width="300vp"
    ohos:layout_alignment="horizontal_center"
    ohos:image_src="$media:compass"/>
```

（7）在 com.example.demo1 包下的 slice/MainAbilitySlice.java 文件中输入程序代码。

（8）启动远程模拟器，运行结果如下：

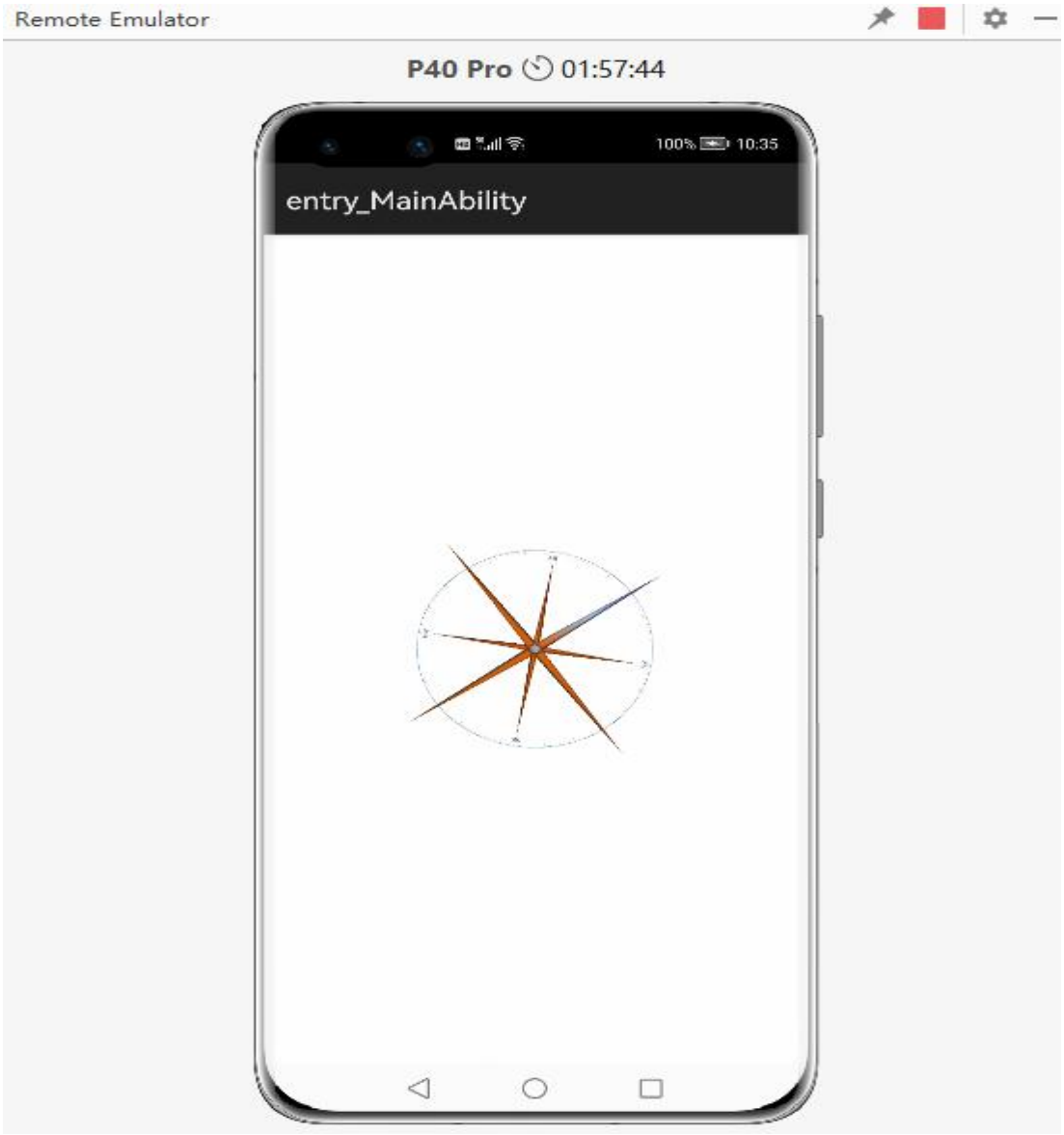

图 2.2.3 模拟器运行结果

6、实验参考程序

```
package com.example.demo2.slice;
import ohos.aafwk.ability.AbilitySlice;
import ohos.aafwk.content.Intent;
import ohos.agp.components.Image;
import ohos.eventhandler.EventHandler;
import ohos.eventhandler.EventRunner;
import ohos.eventhandler.InnerEvent;
import com.example.demo2.ResourceTable;
import ohos.sensor.agent.CategoryOrientationAgent;
import ohos.sensor.bean.CategoryOrientation;
import ohos.sensor.data.CategoryOrientationData;
```

```
import ohos.sensor.listener.ICategoryOrientationDataCallback;

public class MainAbilitySlice extends AbilitySlice {
    //回调函数使用，间隔 50ms
    private static final long SAMPLING_INTERVAL_NANOSECONDS = 500000000L;
    private static final float DEFLECTION_FLAG = -1.0f;
    private CategoryOrientationAgent categoryOrientationAgent;//方向代理类
    private Image compassImg;//指南针图片
    private float degree;//方向角
    //回调函数
    private ICategoryOrientationDataCallback categoryOrientationDataCallback;
    //用事件来刷新界面
    private final EventHandler handler = new EventHandler(EventRunner.current())
    {
        @Override
        protected void processEvent(InnerEvent event) {
            //根据方向角转动指南针图片
            compassImg.setRotation(DEFLECTION_FLAG * degree);
        }
    };
    @Override
    public void onStart(Intent intent) {
        super.onStart(intent);
        super.setUIContent(ResourceTable.Layout_ability_main);
        compassImg = (Image) findComponentById(ResourceTable.Id_compass_icon_img);
        categoryOrientationAgent = new CategoryOrientationAgent();
        //获得方向传感器
        CategoryOrientation categoryOrientation =
          categoryOrientationAgent.getSingleSensor(
                CategoryOrientation.SENSOR_TYPE_ORIENTATION);
        //方向传感器回调函数
        categoryOrientationDataCallback = new ICategoryOrientationDataCallback() {
        @Override
        public void onSensorDataModified(CategoryOrientationData
                        categoryOrientationData) {
                degree = categoryOrientationData.getValues()[0];//获得方向角
                handler.sendEvent(0);//发送事件，用事件来刷新界面
        }
        @Override
        public void onAccuracyDataModified(CategoryOrientation categoryOrientation,
```

```
            int i) {}
        @Override
        public void onCommandCompleted(CategoryOrientation categoryOrientation) {}
      };
      //为方向传感器设置回调函数
      categoryOrientationAgent.setSensorDataCallback(
        categoryOrientationDataCallback, categoryOrientation,
                SAMPLING_INTERVAL_NANOSECONDS);
    }
    @Override
    protected void onStop() {
        super.onStop();
        //取消方向传感器的回调函数
        categoryOrientationAgent.releaseSensorDataCallback(
          categoryOrientationDataCallback);
        //移除所有事件
        handler.removeAllEvent();
    }
}
```

2.3 位置服务实验

1、实验目的

使用 DevEco Studio 开发一个简单的位置服务程序。

2、实验设备

硬件：HarmonyOS 系统手机（可选）

软件：DevEco Studio 集成开发环境（一套）

3、实验内容

使用位置传感器获得设备所在的经度、维度数据，并根据经纬度数据获得设备所在的地理位置信息。

4、实验预习要求

权限申请步骤：

（1）声明所需要的权限

在 entry\src\main\config.json 中，在"abilities"的下面（同级）添加"reqPermissions"字段

```
"reqPermissions": [
      {"name": "ohos.permission.LOCATION"}
    ]
```

（2）请求所需的权限

在 MainAbility.java 的 onStart 方法中添加以下代码：

```
// 测试应用没有被授予权限
if (verifySelfPermission("ohos.permission.LOCATION") !=
    IBundleManager.PERMISSION_GRANTED) {
  // 测试是否能申请弹框授权(首次申请或者前面申请用户未选择禁止且不再提示)
  if (canRequestPermission("ohos.permission.LOCATION")) {
    //请求用户授权
    requestPermissionsFromUser(new String[]{ "ohos.permission.LOCATION" },
      MY_PERMISSIONS_REQUEST_GPS);
  }
 }
```

使用定位传感器步骤：

（1）在 MainAbilitySlice.java 的 onStart 方法中，实例化 locator 对象。

```
Locator locator = new Locator(getContext());
```

（2）实例化定位服务场景

```
RequestParam requestParam = new RequestParam(RequestParam.SCENE_NAVIGATION);
```

（3）实例化定位服务的回调对象

```
CurrentLocatorCallback locatorCallback = new CurrentLocatorCallback();
```

（4）启动一次定位请求

```
locator.requestOnce(requestParam, locatorCallback);
```

5、实验步骤

（1）启动 DevEco Studio。

（2）选择“Create Project”-> 选择”Empty Ability”， 创建一个新项目“Demo3”，Device type 选择 phone。

（3）声明所需要的权限

在 entry\src\main\config.json 中，在"abilities"的下面（同级）添加"reqPermissions"字段

```
"reqPermissions": [
      {"name": "ohos.permission.LOCATION"}
  ]
```

（4）在 resources/base/layout/ability_main.xml 文件中添加 3 个 Text 组件

```
<Text
ohos:id="$+id:text_x"
ohos:height="match_content"
ohos:width="match_content"
ohos:layout_alignment="horizontal_center"
ohos:padding="5vp"
ohos:bottom_margin="10vp"
ohos:text=""
ohos:text_size="50"
/>
```

```
<Text
ohos:id="$+id:text_y"
ohos:height="match_content"
ohos:width="match_content"
ohos:layout_alignment="horizontal_center"
ohos:padding="5vp"
ohos:bottom_margin="10vp"
ohos:text=""
ohos:text_size="50"
/>
<Text
ohos:id="$+id:text_address"
ohos:height="match_content"
ohos:width="match_content"
ohos:layout_alignment="horizontal_center"
ohos:padding="5vp"
ohos:bottom_margin="10vp"
ohos:text_alignment="horizontal_center"
ohos:multiple_lines="true"
ohos:text=""
ohos:text_size="50"
/>
```

（5）在 java/com.example.demo3 下新建 GPSEventHandler.java 类，功能为根据参数传递来的位置信息刷新界面上的经纬度和位置描述。

（6）在 MainAbility.java 的 onStart 中编写程序，功能为如果本程序没有获得位置的权限，则请求用户授权，在 onRequestPermissionsFromUserResult 中编写程序，功能为如果用户没授权，则显示提示信息。

（7）在 slice/MainAbilitySlice.java 中编写代码，向传感器进行一次定位请求，然后通过传感器的回调函数把位置信息和界面的 3 个 Text 作为事件的参数发送给事件处理对象，让它刷新到界面上（避免出现没响应现象）。

图 2.3.1 位置服务程序结果

6、实验参考程序

（1）GPSEventHandler.java 文件如下：

```
package com.example.demo3;
import ohos.agp.components.Text;
import ohos.eventhandler.EventHandler;
import ohos.eventhandler.EventRunner;
import ohos.eventhandler.InnerEvent;
```

```
import ohos.location.GeoAddress;
import ohos.location.GeoConvert;
import ohos.location.Location;
import java.io.IOException;
public class GPSEventHandler extends EventHandler {
    private Text txtX;
    private Text txtY;
    private Text txtAddress;

    public GPSEventHandler(EventRunner eventRunner,
                           Text txtX, Text txtY,
                           Text txtAddress) {
        super(eventRunner);
        this.txtX = txtX;
        this.txtY = txtY;
        this.txtAddress = txtAddress;
    }

    @Override
    protected void processEvent(InnerEvent event) {
        super.processEvent(event);
        //获取事件传递过来的位置信息
        Location location = (Location) event.object;
        //获取位置的纬度
        double x = location.getLatitude();
        //获取位置的经度
        double y = location.getLongitude();
        String addr = "";
        //定义地理解码对象
        GeoConvert geoConvert = new GeoConvert();
        try {
            //通过坐标位置获取地理位置
            GeoAddress ga = geoConvert.getAddressFromLocation
                    (x, y, 1).get(0);
            //获取地理位置的描述
            addr = ga.getDescriptions(0);
        } catch (IOException e) {
            e.printStackTrace();
        }
        //设置文本显示内容
```

```
        txtX.setText("纬度为：" + x);
        txtY.setText("经度为：" + y);
        txtAddress.setText("位置为：" + addr);
    }
}
```

（2）MainAbility.java 文件如下：

```
package com.example.demo3;
import com.example.demo3.slice.MainAbilitySlice;
import ohos.aafwk.ability.Ability;
import ohos.aafwk.content.Intent;
import ohos.agp.window.dialog.ToastDialog;
import ohos.bundle.IBundleManager;
public class MainAbility extends Ability {
    final int  MY_PERMISSIONS_REQUEST_GPS=1;
    @Override
    public void onStart(Intent intent) {
        super.onStart(intent);
        super.setMainRoute(MainAbilitySlice.class.getName());
        if (verifySelfPermission("ohos.permission.LOCATION") !=
                IBundleManager.PERMISSION_GRANTED) {
            // 应用未被授予权限
            if (canRequestPermission("ohos.permission.LOCATION")) {
              // 是否可以申请弹框授权(首次申请或者前面申请用户未选择禁止且不再提示)
                requestPermissionsFromUser(new String[]
                        { "ohos.permission.LOCATION" } ,
                        MY_PERMISSIONS_REQUEST_GPS);
            }
        }
    }
    @Override
    public void onRequestPermissionsFromUserResult (
            int requestCode, String[] permissions,int[] grantResults) {
        switch (requestCode) {
            case MY_PERMISSIONS_REQUEST_GPS: {
                // 匹配 requestPermissions 的 requestCode
                if (grantResults.length > 0&& grantResults[0] ==
                        IBundleManager.PERMISSION_GRANTED) { }
                else {
```

```
                    // 权限被拒绝
                    new ToastDialog(getContext()).
                            setText("位置服务被拒绝！").show();
                }
                return;
            }
        }
    }
}
```

（3）MainAbilitySlice.java 文件如下：

```
package com.example.demo3.slice;
import com.example.demo3.GPSEventHandler;
import com.example.demo3.ResourceTable;
import ohos.aafwk.ability.AbilitySlice;
import ohos.aafwk.content.Intent;
import ohos.agp.components.Text;
import ohos.eventhandler.EventHandler;
import ohos.eventhandler.EventRunner;
import ohos.eventhandler.InnerEvent;
import ohos.location.Location;
import ohos.location.Locator;
import ohos.location.LocatorCallback;
import ohos.location.RequestParam;
public class MainAbilitySlice extends AbilitySlice {
    //用来显示纬度
    private Text txtX;
    //用来显示经度
    private Text txtY;
    //用来显示当前位置
    private Text txtAddress;

    @Override
    public void onStart(Intent intent) {
        super.onStart(intent);
        super.setUIContent(ResourceTable.Layout_ability_main);
        //初始化纬度文本控件
        txtX = (Text) findComponentById(ResourceTable.Id_text_x);
        //初始化经度文本控件
```

```
    txtY = (Text) findComponentById(ResourceTable.Id_text_y);
    //初始化位置文本控件
    txtAddress = (Text) findComponentById(ResourceTable.Id_text_address);
    //实例化 locator 对象
    Locator locator = new Locator(getContext());
    if (locator.isLocationSwitchOn()) {
        //实例化定位服务场景
        RequestParam requestParam = new RequestParam(
                RequestParam.SCENE_NAVIGATION);
        //实例化定位服务的回调对象
        CurrentLocatorCallback locatorCallback = new
                CurrentLocatorCallback();
        //启动一次定位请求
        locator.requestOnce(requestParam, locatorCallback);
    }else {
        txtAddress.setText("请打开位置信息开关！");
    }
}
@Override
public void onActive() {
    super.onActive();
}
@Override
public void onForeground(Intent intent) {
    super.onForeground(intent);
}
/**
 * 定位回调类，实现 LocatorCallback 接口
 */
class CurrentLocatorCallback implements LocatorCallback {
    @Override
    public void onLocationReport(Location location) {
        int eventId = 0;
        long param = 0;
        //定义事件对象
        InnerEvent event = InnerEvent.get(eventId, param, location);
        //获取当前应用的主线程
        EventRunner eventRunner = EventRunner.getMainEventRunner();
        //定义事件处理对象，并绑定到主线程中
        EventHandler eventHandler = new GPSEventHandler
```

```
                (eventRunner, txtX, txtY, txtAddress);
        //把事件发送到事件处理对象中
        eventHandler.sendEvent(event);
    }
    @Override
    public void onStatusChanged(int type) {
        switch (type){
            case Locator.SESSION_START:
                //会话开始
                break;
            case Locator.SESSION_STOP:
                //会话停止
                break;
        }
    }
    @Override
    public void onErrorReport(int type) {
        switch (type){
            case Locator.ERROR_PERMISSION_NOT_GRANTED:
                //没权限
                break;
            case Locator.ERROR_SWITCH_UNOPEN:
                //位置开关没开
                break;
        }     }    } }
```

2.4 设备通信实验

1、实验目的

使用 DevEco Studio 开发一个设备间简单通信的程序。

2、实验设备

硬件：HarmonyOS 系统手机（可选）

软件：DevEco Studio 集成开发环境（一套）

3、实验内容

编写客户端和服务端程序，客户端发送信息，服务端显示客户端发来的信息。

4、实验预习要求

客户端使用当前网络通过 Socket 发送数据：

（1）调用 NetManager.getInstance(Context)获取网络管理的实例对象。

```
NetManager netManager = NetManager.getInstance(context);
```

（2）调用 netManager.hasDefaultNet()查询是否已有默认的数据网络。

```
if (!netManager.hasDefaultNet()) {return;}
```

（3）调用 NetManager.getDefaultNet()获取默认的数据网络。

```
NetHandle netHandle = netManager.getDefaultNet();
```

（4）调用 NetHandle.bindSocket()绑定网络。

```
socket = new DatagramSocket();
netHandle.bindSocket(socket);
```

（5）使用 socket 发送数据。

```
InetAddress address = netHandle.getByName("www.EXAMPLE.com");
byte[] buffer = new byte[1024];
// port 为连接 UDP Socket 时自行指定的端口，buffer 数据
DatagramPacket request = new DatagramPacket(buffer, buffer.length, address, port);
// 发送数据
socket.send(request);
```

服务端通过 Socket 接收数据：

（1）获取 Socket 的实例对象。

```
DatagramSocket socket = new DatagramSocket(PORT)
```

（2）获取 Packet 的实例对象。

```
DatagramPacket packet = new DatagramPacket(new byte[255], 255);
```

（2）接收数据。

```
socket.receive(packet);
```

5、实验步骤

（1）启动 DevEco Studio。

（2）选择“Create Project”-> 选择”Empty Ability”， 创建一个新项目“Demo4Client”，Device type 选择 phone。

（3）在 resources/base/layout/ability_main.xml 文件中添加 2 个组件

```
<TextField
    ohos:id="$+id:input_text"
    ohos:height="match_content"
    ohos:width="match_parent"
    ohos:hint="请输入服务器 IP"
    ohos:start_margin="10vp"
    ohos:padding="10vp"
    ohos:end_margin="10vp"
    ohos:text_alignment="center"
```

```
        ohos:text_size="20fp"/>
    <Button
        ohos:id="$+id:start_button"
        ohos:height="35vp"
        ohos:width="240vp"
        ohos:margin="20vp"
        ohos:text="发送"
        ohos:text_size="20fp"/>
```

（4）在 java\com.example.demo4client\utils 下新建 ThreadPoolUtil.java 类，功能为用新线程执行相应的程序。

（5）在 entry\src\main\config.json 中声明所需的权限 GET_NETWORK_INFO、INTERNET、SET_NETWORK_INFO、MANAGE_WIFI_CONNECTION、SET_WIFI_INFO、GET_WIFI_INFO

```
"reqPermissions": [
  {"name": "ohos.permission.GET_NETWORK_INFO"},
  {"name": "ohos.permission.INTERNET"},
  {"name": "ohos.permission.SET_NETWORK_INFO"},
  {"name": "ohos.permission.MANAGE_WIFI_CONNECTION"},
  {"name": "ohos.permission.SET_WIFI_INFO"},
  {"name": "ohos.permission.GET_WIFI_INFO"}
 ]
```

（6）在 slice/MainAbilitySlice.java 中编写代码，功能为输入服务端 IP，然后用发送按钮发送数据。

（7）选择“Create Project”-> 选择”Empty Ability”， 创建一个新项目“Demo4Server”，Device type 选择 phone（注意：选择在新窗口创建新项目）。

（8）在 resources/base/layout/ability_main.xml 文件中添加 3 个组件

```
    <Text
        ohos:id="$+id:input_text"
        ohos:height="match_content"
        ohos:width="match_parent"
        ohos:start_margin="10vp"
        ohos:padding="10vp"
        ohos:end_margin="10vp"
        ohos:text_alignment="center"
        ohos:text_size="20fp"/>
    <Button
```

```
        ohos:id="$+id:start_button"
        ohos:height="35vp"
        ohos:width="240vp"
        ohos:margin="20vp"
        ohos:text="开始服务"
        ohos:text_size="20fp"/>
    <Text
        ohos:id="$+id:out_text"
        ohos:height="match_content"
        ohos:width="match_parent"
        ohos:multiple_lines="true"
        ohos:padding="10vp"
        ohos:text_alignment="center"
        ohos:text_size="20fp"/>
```

（9）在 java\com.example.demo4server\utils 下新建 ThreadPoolUtil.java 类，功能为用新线程执行相应的程序（与客户端相同）。

（10）在 entry\src\main\config.json 中声明所需的权限 GET_NETWORK_INFO、INTERNET、SET_NETWORK_INFO、MANAGE_WIFI_CONNECTION、SET_WIFI_INFO、GET_WIFI_INFO

（11）在 slice/MainAbilitySlice.java 中编写代码，功能为点击开始服务按钮接收数据。

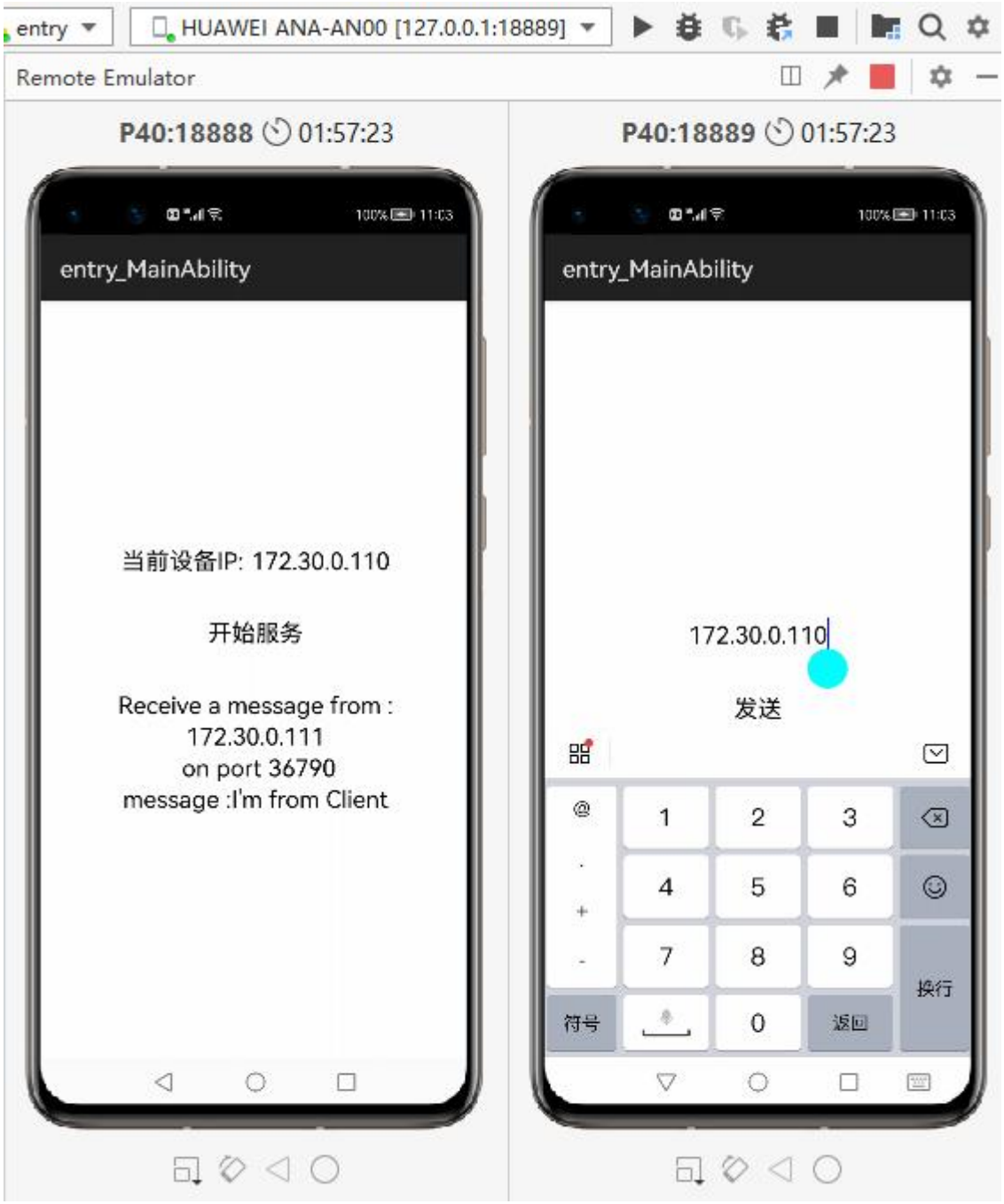

图 2.4.1 设备间通信程序结果

6、实验参考程序

（1）ThreadPoolUtil.java 文件如下：

```
package com.example.demo4client.utils;
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.ThreadFactory;
import java.util.concurrent.ThreadPoolExecutor;
import java.util.concurrent.TimeUnit;
```

```
import java.util.concurrent.atomic.AtomicInteger;
public class ThreadPoolUtil {
    private static final int CORE_COUNT = 10;
    private static final int THREAD_COUNT = 20;
    private static final int WORK_QUEUE_SIZE = 50;
    private static final long KEEP_ALIVE = 10L;
    private static final AtomicInteger THREAD_ID = new AtomicInteger(1);
    private static  final ThreadPoolExecutor executor =
      new ThreadPoolExecutor(CORE_COUNT, THREAD_COUNT, KEEP_ALIVE,
        TimeUnit.SECONDS, new ArrayBlockingQueue<>(WORK_QUEUE_SIZE),
        new CommonThreadFactory());
    private ThreadPoolUtil() {  }
    public static void submit(Runnable task) {
        executor.submit(task);    }
    static class CommonThreadFactory implements ThreadFactory {
        @Override
        public Thread newThread(Runnable runnable) {
            String threadName;
            if (THREAD_ID.get() == Integer.MAX_VALUE) {
                threadName = "threadpool-common-" + THREAD_ID.getAndSet(1);
            } else {
                threadName = "threadpool-common-" + THREAD_ID.incrementAndGet();
            }
            return new Thread(runnable, threadName);
        }
    } }
```

（2）客户端 MainAbilitySlice.java 文件如下：

```
package com.example.demo4client.slice;
import com.example.demo4client.ResourceTable;
import com.example.demo4client.utils.ThreadPoolUtil;
import ohos.aafwk.ability.AbilitySlice;
import ohos.aafwk.content.Intent;
import ohos.agp.components.Component;
import ohos.agp.components.Text;
import ohos.hiviewdfx.HiLog;
import ohos.net.NetHandle;
import ohos.net.NetManager;
import java.io.IOException;
```

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
public class MainAbilitySlice extends AbilitySlice {
    private static final int PORT = 8888;
    private Text inputText;
    @Override
    public void onStart(Intent intent) {
        super.onStart(intent);
        super.setUIContent(ResourceTable.Layout_ability_main);
        initComponents();
    }
    private void initComponents() {
        Component startButton = findComponentById(ResourceTable.Id_start_button);
        inputText = (Text) findComponentById(ResourceTable.Id_input_text);
        startButton.setClickedListener(this::netRequest);
    }
    private void netRequest(Component component) {
        ThreadPoolUtil.submit(() -> {
            NetManager netManager = NetManager.getInstance(null);
            if (!netManager.hasDefaultNet()) { return; }
            try (DatagramSocket socket = new DatagramSocket()) {
                NetHandle netHandle = netManager.getDefaultNet();
                InetAddress address = netHandle.getByName(inputText.getText());
                netHandle.bindSocket(socket);
                byte[] buffer = "I'm from Client".getBytes();
                DatagramPacket request =
                  new DatagramPacket(buffer, buffer.length, address, PORT);
                socket.send(request);
            } catch (IOException e) {
                //String s1=e.toString();
            }
        });
    }
    @Override
    public void onActive() {  super.onActive();   }
    @Override
    public void onForeground(Intent intent) { super.onForeground(intent); }
}
```

（3）服务端 MainAbilitySlice.java 文件如下：

```
package com.example.demo4server.slice;
import com.example.demo4server.ResourceTable;
import com.example.demo4server.utils.ThreadPoolUtil;
import ohos.aafwk.ability.AbilitySlice;
import ohos.aafwk.content.Intent;
import ohos.agp.components.Component;
import ohos.agp.components.Text;
import ohos.hiviewdfx.HiLog;
import ohos.wifi.WifiDevice;
import ohos.wifi.WifiLinkedInfo;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.util.Optional;
public class MainAbilitySlice extends AbilitySlice {
    private static final int PORT = 8888;
    private Text outText;
    @Override
    public void onStart(Intent intent) {
        super.onStart(intent);
        super.setUIContent(ResourceTable.Layout_ability_main);
        initComponents();
    }
    private void initComponents() {
        Component startButton = findComponentById(ResourceTable.Id_start_button);
        Text inputText = (Text) findComponentById(ResourceTable.Id_input_text);
        outText = (Text) findComponentById(ResourceTable.Id_out_text);
        startButton.setClickedListener(this::startServer);
        inputText.setText("当前设备 IP:" +
          System.lineSeparator() + getLocationIpAddress());
    }
    private void startServer(Component component) {
        ThreadPoolUtil.submit(() -> {
            try (DatagramSocket socket = new DatagramSocket(PORT)) {
                DatagramPacket packet = new DatagramPacket(new byte[255], 255);
                while (true) {
                    socket.receive(packet);
                    //用主线程显示接收的数据
```

```
                    getUITaskDispatcher().syncDispatch(() -> outText.setText(
                            "Receive a message from :" +
                    packet.getAddress().getHostAddress() + System.lineSeparator()+
                    " on port " + packet.getPort() +
                    System.lineSeparator() + "message :" +
                    new String(packet.getData()).substring(0, packet.getLength())));

                    packet.setLength(255);
                    Thread.sleep(1000);
                }
            } catch (IOException | InterruptedException e) {
                //StartServer  IOException | InterruptedException
            }
        });
    }
    private String getLocationIpAddress() {
        WifiDevice wifiDevice = WifiDevice.getInstance(this);
        Optional<WifiLinkedInfo> linkedInfo = wifiDevice.getLinkedInfo();
        int ip = linkedInfo.get().getIpAddress();
        return (ip & 0xFF) + "." + ((ip >> 8) & 0xFF) + "." +
                ((ip >> 16) & 0xFF) + "." + (ip >> 24 & 0xFF);
    }
    @Override
    public void onActive() { super.onActive();}
    @Override
    public void onForeground(Intent intent) {super.onForeground(intent); }
}
```

2.5 系统设置监控实验

1、实验目的

使用 DevEco Studio 开发一个简单的系统设置监控程序。

2、实验设备

硬件：HarmonyOS 系统手机（可选）

软件：DevEco Studio 集成开发环境（一套）

3、实验内容

监控设备的无线网络状态、蓝牙状态、飞行模式的改变，并实时显示出来。

4、实验预习要求

系统设置监控步骤：

（1）新建 dataAbilityHelper 对象

```
dataAbilityHelper = DataAbilityHelper.creator(this);
```

（2）为系统设置（如 WIFI_STATUS）登记观察者 dataAbilityObserver

```
dataAbilityHelper.registerObserver(
    SystemSettings.getUri(SystemSettings.Wireless.WIFI_STATUS),
    dataAbilityObserver );
```

（3）当系统设置（如 WIFI_STATUS）改变时，自动调用观察者 dataAbilityObserver 的 onChange 方法，得到当前的系统设置的值，再修改显示的系统设置的值。

```
String wifiFormat =
SystemSettings.getValue(dataAbilityHelper, SystemSettings.Wireless.WIFI_STATUS);
```

（4）当程序结束时为系统设置（如 WIFI_STATUS）取消登记的观察者 dataAbilityObserver

```
dataAbilityHelper.unregisterObserver(SystemSettings.getUri(
    SystemSettings.Wireless.WIFI_STATUS),dataAbilityObserver);
```

5．实验步骤

（1）启动 DevEco Studio。

（2）选择“Create Project”-> 选择”Empty Ability”， 创建一个新项目“Demo5”，Device type 选择 phone。

（3）在 resources/base/layout/ability_main.xml 文件中添加组件

```
<Text
    ohos:height="35vp"
    ohos:width="match_parent"
    ohos:text="系统配置"
    ohos:text_alignment="center"
    ohos:text_size="22fp"
    ohos:top_margin="30vp"/>
<DirectionalLayout
    ohos:height="40vp"
    ohos:width="match_parent"
    ohos:orientation="horizontal"
    ohos:top_margin="15vp">
    <Text
        ohos:height="match_content"
        ohos:width="match_content"
        ohos:end_padding="10vp"
        ohos:start_padding="15vp"
```

```
        ohos:text="无线网络:"
        ohos:text_size="16fp"/>
    <Text
        ohos:id="$+id:wifi_status"
        ohos:height="match_content"
        ohos:width="match_content"
        ohos:text_color="#708095"
        ohos:text_size="16fp"/>
</DirectionalLayout>
```

蓝牙状态 bluetooth_status、飞行模式 airplane_mode_status 除 id 和提示外和无线网络一致。

（5）在 slice/MainAbilitySlice.java 中编写代码，功能为监视系统设置（无线网络状态、蓝牙状态、飞行模式）的改变，并实时显示出来。

图 2.5.1 系统设置监控实验程序结果

6、实验参考程序

```
package com.example.demo5.slice;
import com.example.demo5.ResourceTable;
import ohos.aafwk.ability.AbilitySlice;
import ohos.aafwk.content.Intent;
import ohos.aafwk.ability.DataAbilityHelper;
import ohos.aafwk.ability.IDataAbilityObserver;
import ohos.agp.components.Text;
import ohos.sysappcomponents.settings.SystemSettings;
public class MainAbilitySlice extends AbilitySlice {
    private Text wifiStatusText;
    private Text bluetoothText;
    private Text airplaneModeStatusText;
    private DataAbilityHelper dataAbilityHelper;
    private final IDataAbilityObserver dataAbilityObserver = new IDataAbilityObserver()
    {
        @Override
        public void onChange() {
            String wifiFormat = SystemSettings.getValue(dataAbilityHelper,
                                SystemSettings.Wireless.WIFI_STATUS);
            String airplaneModeStatus = SystemSettings.getValue(dataAbilityHelper,
                                SystemSettings.General.AIRPLANE_MODE_STATUS);
            String bluetoothFormat = SystemSettings.getValue(dataAbilityHelper,
                                SystemSettings.Wireless.BLUETOOTH_STATUS);
            setWifiStatus(wifiFormat);
            setAirplaneModeStatus(airplaneModeStatus);
            setBluetoothStatus(bluetoothFormat);
        }
     };
    @Override
    public void onStart(Intent intent) {
        super.onStart(intent);
        super.setUIContent(ResourceTable.Layout_ability_main);
        initDataAbilityHelper();
        initComponents();
    }
    private void initDataAbilityHelper() {
        dataAbilityHelper = DataAbilityHelper.creator(this);
        dataAbilityHelper.registerObserver(SystemSettings.getUri(
```

```
        SystemSettings.Wireless.WIFI_STATUS),dataAbilityObserver);
    dataAbilityHelper.registerObserver(SystemSettings.getUri(
        SystemSettings.General.AIRPLANE_MODE_STATUS),dataAbilityObserver);
    dataAbilityHelper.registerObserver(SystemSettings.getUri(
        SystemSettings.Wireless.BLUETOOTH_STATUS),dataAbilityObserver);
}
private void initComponents() {
    if (findComponentById(ResourceTable.Id_wifi_status) instanceof Text) {
        wifiStatusText = findComponentById(ResourceTable.Id_wifi_status);
    }
    setWifiStatus(SystemSettings.getValue(dataAbilityHelper,
         SystemSettings.Wireless.WIFI_STATUS));
    if (findComponentById(ResourceTable.Id_bluetooth_status) instanceof Text) {
        bluetoothText = findComponentById(ResourceTable.Id_bluetooth_status);
    }
    setBluetoothStatus(SystemSettings.getValue(dataAbilityHelper,
       SystemSettings.Wireless.BLUETOOTH_STATUS));
    if (findComponentById(ResourceTable.Id_airplane_mode_status)
    instanceof Text){
        airplaneModeStatusText=
        findComponentById(ResourceTable.Id_airplane_mode_status);
    }
    setAirplaneModeStatus(SystemSettings.getValue(dataAbilityHelper,
        SystemSettings.General.AIRPLANE_MODE_STATUS));
}
private void setWifiStatus(String wifiStatus) {
    if ("1".equals(wifiStatus)) {
        wifiStatusText.setText("open");
    } else {
        wifiStatusText.setText("close");
    }
}
private void setAirplaneModeStatus(String airplaneModeStatus) {
    if ("1".equals(airplaneModeStatus)) {
        airplaneModeStatusText.setText("open");
    } else {
        airplaneModeStatusText.setText("close");
    }
}
private void setBluetoothStatus(String blueToothStatus) {
```

```
            if ("1".equals(blueToothStatus)) {
                bluetoothText.setText("open");
            } else {
                bluetoothText.setText("close");
            }
        }
        @Override
        protected void onStop() {
            super.onStop();
            unregisterObserver();
        }
        private void unregisterObserver() {
          dataAbilityHelper.unregisterObserver(SystemSettings.getUri(
            SystemSettings.Wireless.WIFI_STATUS),dataAbilityObserver);
          dataAbilityHelper.unregisterObserver(SystemSettings.getUri(
            SystemSettings.General.AIRPLANE_MODE_STATUS),dataAbilityObserver);
          dataAbilityHelper.unregisterObserver(SystemSettings.getUri(
            SystemSettings.Wireless.BLUETOOTH_STATUS),dataAbilityObserver);
        }
    }
```