

HARMONYOS CODELABS

软件应用示例操作指南

电子发烧友运营战队整理

<https://bbs.elecfans.com/>

目录

一、视频播放.....	1
1. 介绍.....	1
本篇 Codelab 将实现的内容.....	1
您将建立什么.....	1
您将会学到什么.....	1
2. 您需要什么.....	1
硬件要求.....	1
软件要求.....	2
需要的知识点.....	2
3. 能力接入准备.....	2
4. 代码片段.....	2
5. 总结.....	8
二、基本控件.....	9
1. 介绍.....	9
本篇 Codelab 将实现的内容.....	9
您将建立什么.....	9
您将会学到什么.....	9
2. 您需要什么.....	9
硬件要求.....	9
软件要求.....	9
需要的知识点.....	10
3. 能力接入准备.....	10
4. 代码片段.....	10
5. 总结.....	13
三、应用偏好数据读写.....	13
1. 介绍.....	13
本篇 Codelab 将实现的内容.....	13
您将建立什么.....	14
您将会学到什么.....	14
2. 您需要什么.....	14
硬件要求.....	14
软件要求.....	14
需要的知识点.....	14
3. 能力接入准备.....	15
4. 代码片段.....	15
5. 恭喜你.....	20
四、剪切板.....	21
1. 介绍.....	21

本篇 CodeLab 将实现的内容.....	21
您将建立什么.....	21
您将会学到什么.....	21
2. 您需要什么.....	21
硬件要求.....	21
软件要求.....	21
需要的知识点.....	22
3. 能力接入准备.....	22
4. 代码编写.....	22
实现"复制文本"功能，代码片段如下：	22
实现"粘贴文本"功能，代码片段如下：	23
5. 编译运行.....	23
通过 hdc 连接大屏设备.....	23
运行.....	23
6. 恭喜您.....	24
五、分布式任务调度.....	24
1. 介绍.....	24
本篇 CodeLab 将实现的内容.....	24
您将建立什么.....	24
您将会学到什么.....	24
2. 您需要什么.....	24
硬件要求.....	24
软件要求.....	25
需要的知识点.....	25
3. 能力接入准备.....	25
4. 代码编写.....	25
5. 编译运行.....	26
通过 hdc 连接大屏设备.....	26
运行.....	26
6. 恭喜您.....	27
六、元程序交互.....	27
1. 介绍.....	27
本篇 CodeLab 将实现的内容.....	27
您将建立什么.....	27
您将会学到什么.....	27
2. 您需要什么.....	28
硬件要求.....	28
软件要求.....	28
需要的知识点.....	28
3. 能力接入准备.....	28
4. 代码编写.....	28
5. 编译运行.....	29

通过 hdc 连接大屏设备.....	29
运行.....	29
6. 恭喜您.....	29
七、UI 设计开发与预览.....	30
1. 介绍.....	30
您将建立什么.....	30
您将学到什么.....	30
硬件要求.....	30
软件要求.....	30
需要的知识点.....	31
2. 代码开发.....	31
1.打开本地 Demo Project（harmony-todo）.....	31
2.点击 Previewer 按钮，实时预览 Demo Project（harmony-todo）.....	32
3.为 index 页面（index.html）添加布局信息.....	33
3.1 添加今日待办事项的列表.....	33
3.2 添加明日待办事项的列表.....	34
3.3 添加即将来临待办事项的列表.....	35
4.添加逻辑代码(index.js).....	35
4.1 添加待办事项完成的逻辑代码.....	35
4.2 添加删除待办事项的逻辑代码.....	36
3. 编译构建.....	36
4. 恭喜您.....	37
八、HelloWorld 应用开发 E2E 体验.....	38
1. 介绍.....	38
您将建立什么.....	38
您将学到什么.....	38
硬件要求.....	38
软件要求.....	38
技能要求.....	39
2. 代码开发.....	39
1. 工程创建.....	39
2. 编码.....	42
3. 编译构建.....	45
4. 部署并运行.....	46
1. 启动模拟器.....	46
2. 部署应用.....	47
5. 恭喜您.....	48
九、设备虚拟化特性开发.....	49
1. 介绍.....	49
您将建立什么.....	49
您将学到什么.....	49
2. 您需要什么.....	49

硬件要求.....	49
软件要求.....	49
需要的知识点.....	49
3. 操作步骤.....	50
1. 访问站点 https://hpm.harmonyos.com/#/home ，如下图，选择摄像头类产品.....	50
2. 进入页面后点击"直接下载"下载文件到 Linux 虚拟机映射的网络驱动器路径并解压。.....	50
3. 打开 vscode 编辑器，在 HUAWEI DevEco Device Tool 插件 welcome 首页，选择 Import Project，如下图：.....	51
4. 打开 TERMINAL 窗口.....	51
5. 在 TERMINAL 窗口中，输入命令 hpm install。如下图：.....	53
6. hpm install 命令行执行完毕，源码工程就准备就绪了。.....	53
7. 添加代码片段.....	53
8. 安全检查.....	54
9. HarmonyOS Demo 源码编译.....	54
10. HarmonyOS 镜像烧录.....	56
11. HarmonyOS 源码单步调试.....	58
12. 摄像头虚拟化特性验证.....	61
4. 恭喜您.....	66

更多最新的鸿蒙相关技术文章、课程、直播等信息，欢迎关注 **HarmonyOS社区**



HarmonyOS Codelabs 软件应用示例操作指南

一、视频播放

1. 介绍

本篇 Codelab 将实现的内容

HarmonyOS 是面向全场景多终端的分布式操作系统，使得应用程序的开发打破了智能终端互通的性能和数据壁垒，业务逻辑原子化开发，适配多端。通过一个简单应用开发，体验 HarmonyOS 的视频播放能力

您将建立什么

在这个 Codelab 中，你将创建 Demo Project，并将 Demo 编译成 Hap，此示例应用程序展示了如何播放视频。

您将会学到什么

- 如何创建一个 HarmonyOS Demo Project
- 如何构建一个 Hap 并且将其部署到智慧屏真机
- 通过此示例应用体验如何播放本地或者在线视频

2. 您需要什么

硬件要求

- 操作系统：Windows10 64 位
- 内存：8G 及以上。
- 硬盘：100G 及以上。
- 分辨率：1280*800 及以上

软件要求

- DevEco Studio: 需手动下载安装, 详细步骤请参考《DevEco Studio 使用指南》2.1.2
- JDK: DevEco Studio 自动安装。
- Node.js: 请手动下载安装, 详细步骤请参考《DevEco Studio 使用指南》2.1.3 下载和安装 Node.js。
- HarmonyOS SDK: 待 DevEco Studio 安装完成后, 利用 DevEco Studio 来加载 HarmonyOS SDK。详细步骤请参考《DevEco Studio 使用指南》2.1.6 加载 HarmonyOS SDK。
- Maven 库依赖包: 如需手动拷贝和配置, 详细步骤请参考《DevEco Studio 使用指南》2.3 离线方式配置 Maven 库。

需要的知识点

- Java 基础开发能力。

3. 能力接入准备

实现 HarmonyOS 应用开发, 需要完成以下准备工作:

- 环境准备。
- 环境搭建。
- 创建项目
- 申请调试证书
- 应用开发

具体操作, 请按照《DevEco Studio 使用指南》中详细说明来完成。

提示: 需要通过注册成开发者才能完成集成准备中的操作。

4. 代码片段

1. 布局:

- 创建播放视频的 Ability

- `public class VedioPlayAbilitySlice extends AbilitySlice implements SurfaceOps.Callback`

- 布局截图

-



- 布局代码:

```

//设置页面背景透明
WindowManager windowManager = WindowManager.getInstance();
Window window = windowManager.getTopWindow().get();
window.setTransparent(true);

//页面父布局
DependentLayout myLayout = new DependentLayout(this);
DependentLayout.LayoutParams params = new
DependentLayout.LayoutParams(MATCH_PARENT,
MATCH_PARENT);
myLayout.setLayoutConfig(params);
//显示视频的自定义 videoView
DependentLayout.LayoutParams lpVideo = new
DependentLayout.LayoutParams(MATCH_PARENT,
MATCH_PARENT);
videoView = new VideoView(this, this);
videoView.setHandler(handler);
myLayout.addComponent(videoView, lpVideo);

DependentLayout rlParent = new DependentLayout(this);
DependentLayout.LayoutParams lpParent = new
DependentLayout.LayoutParams(MATCH_PARENT,
WRAP_CONTENT);
lpParent.addRule(DependentLayout.ALIGN_PARENT_BOTTOM);

```



```
• lpParent.leftMargin = ConvertUtils.dp2Px(40);
• lpParent.rightMargin = ConvertUtils.dp2Px(40);
• lpParent.bottomMargin = ConvertUtils.dp2Px(40);
• myLayout.addComponent(rParent, lpParent);
• //显示播放暂停按钮
• playBtn = new Image(this);
• DependentLayout.LayoutConfig lpPlayBtn = new
DependentLayout.LayoutConfig(ConvertUtils.dp2Px(40),
ConvertUtils.dp2Px(40));
• lpPlayBtn.addRule(DependentLayout.ALIGN_PARENT_RIGHT);
• playBtn.setLayoutConfig(lpPlayBtn);
• playBtn.setId(1112);
• playBtn.setPixelMap(ResourceTable.Media_pause);
• playBtn.setScaleType(Image.ScaleType.SCALE_TO_FULL);
• playBtn.invalidate();
• playBtn.setClickListener(new Component.ClickedListener() {
• @Override
• public void onClick(Component component) {
•     if (videoView.getPlayState() == VideoView.STATE_PLAYING) {
•         videoView.pause();
•         playBtn.setPixelMap(ResourceTable.Media_play);
•     } else {
•         videoView.start();
•         playBtn.setPixelMap(ResourceTable.Media_pause);
•     }
• }
• });
• rParent.addComponent(playBtn);
• //显示当前视频播放时间
• timePlay = new Text(this);
• DependentLayout.LayoutConfig lpTimePlay = new
DependentLayout.LayoutConfig(WRAP_CONTENT,
WRAP_CONTENT);
• lpTimePlay.addRule(DependentLayout.CENTER_VERTICAL);
• timePlay.setLayoutConfig(lpTimePlay);
• timePlay.setId(1111);
• timePlay.setText("00:00/00:00");
• timePlay.setTextSize(40);
• timePlay.setTextColor(Color.WHITE);
• rParent.addComponent(timePlay);
• // 显示播放进度条
• roundProgressBar = new ProgressBar(this);
```

```

• roundProgressBar.setProgressWidth(ConvertUtils.dp2Px(5));
• roundProgressBar.setProgressColor(Color.RED);
• roundProgressBar.setMax(100);
• roundProgressBar.setProgress(0);
• DependentLayout.LayoutConfig lpProgressBar = new
  DependentLayout.LayoutConfig(MATCH_PARENT,
  ConvertUtils.dp2Px(40));
• lpProgressBar.addRule(DependentLayout.RIGHT_OF,
  timePlay.getId());
• lpProgressBar.leftMargin = ConvertUtils.dp2Px(20);
• lpProgressBar.rightMargin = ConvertUtils.dp2Px(60);
• rlParent.addComponent(roundProgressBar, lpProgressBar);

```

2. 自定义 VideoView

- 继承父类 SurfaceProvider

```

• public class VideoView extends SurfaceProvider implements
  Player.IPlayerCallback

```

- 初始化:

```

• public VideoView(Context context, SurfaceOps.Callback callback) {
•   super(context);
•   player = new Player(getContext());
•   player.setPlayerCallback(this);
•   Optional<SurfaceOps> surfaceHolderOptional = getSurfaceOps();
•   SurfaceOps surfaceHolder = surfaceHolderOptional.get();
•   surfaceHolder.addCallback(callback);
•   setZOrderOnTop(false);
•   state = STATE_INIT;
• }

```

- 播放视频

如果播放的是在线视频，需要申请网络权限：

```

"reqPermissions": [
  {
    "name": "harmonyos.permission.INTERNET"
  }
]

```

```
public void playAssets(String fileName, boolean isLooping, SurfaceOps
holder) {
    try {
        //播放本地视频:
        //player.setSource(getContext().getResourceManager().getRawFileEntr
y(fileName).openRawFileDescriptor());
        //播放在线视频:
        player.setSource(new
Source("https://media.w3.org/2010/05/sintel/trailer.mp4"));
        player.setVideoSurface(holder.getSurface());
        player.enableSingleLooping(isLooping);
        player.enableScreenOn(true);
        player.prepare();
        initPlayViewSize();
        player.play();
        if (state != STATE_INIT) {
            player.rewindTo(currentTime * 1000);
        }
        state = STATE_PLAYING;

        handler.sendEvent(InnerEvent.get(MESSAGE_UPDATE_PLAY_TIME))
        ;
    } catch (Exception e) {
        e.printStackTrace();
        Log.hiLog(e.toString());
    }
}

private void initPlayViewSize() {
    int videoWidth = player.getVideoWidth();
    int videoHeight = player.getVideoHeight();
    Log.hiLog("videoWidth:" + videoWidth + ", videoHeight:" +
videoHeight);
    if (videoWidth < videoHeight) {
        double scale = screenHeight * 1.f / videoHeight;
        double currHeight = videoHeight * scale;
        double currWidth = videoWidth * scale;
        setHeight(((int) currHeight));
        setWidth(((int) currWidth));
    } else {
        double scale = screenWidth * 1.f / videoWidth;
        double currHeight = videoHeight * scale;
```

```
double currWidth = videoWidth * scale;
setHeight(((int) currHeight));
setWidth(((int) currWidth));
}
}
```

- 暂停:

```
public void pause() {
    if (state == STATE_PLAYING) {
        player.pause();
        state = STATE_PAUSE;
    }
}
```

- 在 AbilitySlice 启动自动播放

```
@Override
public void surfaceCreated(SurfaceOps surfaceOps) {
    Log.hiLog("surfaceCreated");
    videoView.playAssets("resources/rawfile/VID_20200613_204240.mp4", true, surfaceOps);
}

@Override
public void surfaceDestroyed(SurfaceOps surfaceOps) {
    Log.hiLog("surfaceDestroyed");
    videoView.stop();
}
```

3. 响应遥控器点击

```
@Override
public boolean onKeyUp(int keyCode, KeyEvent keyEvent) {
    switch (keyCode) {
        case KeyEvent.KEY_DPAD_CENTER:
        case KeyEvent.KEY_ENTER:
            playBtn.performClick();
            return true;
        default:
            break;
    }
    return false;
}
```

4. 编译运行该应用

- 通过 `hdc` 连接大屏设备

先查看智慧屏 IP:

大屏设置->"网络与连接"->"网络"->"有线网络"

在 `cmd` 或者 IDE 的 Terminal 输入命令:

```
hdc tconn 192.168.3.9:5555
```

- 运行 `hap`



5. 总结

你已经成功完成了 **HarmonyOS** 应用开发 **E2E** 体验，学到了:

- 如何创建一个 **HarmonyOS Demo Project**
- 如何构建一个 **Hap** 并且将其部署到真机
- 在 **HarmonyOS** 上如何使用 **HarmonyOS** 的视频播放的能力

二、基本控件

1. 介绍

本篇 Codelab 将实现的内容

HarmonyOS 是面向全场景多终端的分布式操作系统，使得应用程序的开发打破了智能终端互通的性能和数据壁垒，业务逻辑原子化开发，适配多端。通过一个简单应用开发，体验 HarmonyOS 的基本控件使用。

您将建立什么

在这个 Codelab 中，你将创建 Demo Project，并将 Demo 编译成 Hap，部署到智慧屏上，以实现 HarmonyOS 基本控件使用。

您将会学到什么

- 如何创建一个 HarmonyOS Demo Project
- 如何构建一个 Hap 并且将其部署到智慧屏远程模拟器上
- 通过此示例应用体验在 HarmonyOS 上如何使用基本控件，包括文本输入框，日期选择控件，单选按钮，下拉菜单和按钮。

2. 您需要什么

硬件要求

- 操作系统：Windows10 64 位
- 内存：8G 及以上。
- 硬盘：100G 及以上。
- 分辨率：1280*800 及以上

软件要求

- DevEco Studio: 需手动下载安装，详细步骤请参考《DevEco Studio 使用指南》2.1.2

- JDK: DevEco Studio 自动安装。
- Node.js: 请手动下载安装, 详细步骤请参考《DevEco Studio 使用指南》2.1.3 下载和安装 Node.js。
- HarmonyOS SDK: 待 DevEco Studio 安装完成后, 利用 DevEco Studio 来加载 HarmonyOS SDK。详细步骤请参考《DevEco Studio 使用指南》2.1.6 加载 HarmonyOS SDK。
- Maven 库依赖包: 如需手动拷贝和配置, 详细步骤请参考《DevEco Studio 使用指南》2.3 离线方式配置 Maven 库。

需要的知识点

- Java 基础开发能力。

3. 能力接入准备

实现 HarmonyOS 应用开发, 需要完成以下准备工作:

- 环境准备。
- 环境搭建。
- 创建项目
- 申请调试证书
- 应用开发

具体操作, 请按照《DevEco Studio 使用指南》中详细说明来完成。

提示: 需要通过注册成开发者才能完成集成准备中的操作。

4. 代码片段

1. 在 index.html 文件中写入以下代码:

```
// input (text) 文本输入, 输入用户名。  
<div class="item-content">  
  <input class="input-text" placeholder="{{userNamePromt}}"  
  onchange="getName" type="text"></input> // 设置占位内容: userNamePromt,  
  设置 input 类型为 text  
</div>
```

```
// input (radio) 单选按钮，输入性别。
<div class="item-content">
  <label target="radio1">{{ $t('Strings.male') }}:</label>
  <input id="radio1" type="radio" name="radio"
value="{{ $t('Strings.male') }}" onchange="getGender"></input>
  <label target="radio2">{{ $t('Strings.female') }}:</label>
  <input id="radio2" type="radio" name="radio"
value="{{ $t('Strings.female') }}" onchange="getGender"></input>
  <label target="radio3">{{ $t('Strings.secret') }}:</label>
  <input id="radio3" type="radio" name="radio"
value="{{ $t('Strings.secret') }}" onchange="getGender"></input>
</div>

// picker 日期选择控件，输入日期。
<div class="item-content">
  <picker type="date" end="2020-01-01" selected="1990-01-01"
value="{{today}}" onchange="getDate"></picker>
</div>

// select 下拉菜单，输入学历。
<select class="select">
  <option value=" "> </option>
  <option
value="{{ $t('Strings.graduated') }}">{{ $t('Strings.graduated') }}</option>
  <option
value="{{ $t('Strings.bachelor') }}">{{ $t('Strings.bachelor') }}</option>
  <option value="{{ $t('Strings.master') }}">{{ $t('Strings.master') }}</option>
  <option value="{{ $t('Strings.doctor') }}">{{ $t('Strings.doctor') }}</option>
</select>
```

2. 在 index.js 文件中写入以下代码：

```
data: {
  userNamePromt:'这是占位符',
  today: "2020-08-26"
},
getName(e){
  console.info("用户名是" + e.value);
},
getGender(e){
  console.info("性别是" + e.value);
},
```



```
getDate(e){  
    console.info("选择日期是" + e.value);  
},
```

3. 在 js/default/i18n/ 文件中写入以下代码:

在 zh-CN. json 文件中写入:

```
"Strings": {  
    "hello": "您好",  
        "world": "世界",  
        "male": "男",  
        "female": "女",  
        "secret": "保密",  
        "graduated": "本科",  
        "bachelor": "学士",  
        "master": "硕士",  
        "doctor": "博士"  
},
```

相应的, 在 en-US. json 文件中写入:

```
"Strings": {  
    "hello": "Hello",  
        "world": "World",  
        "male": "male",  
        "female": "female",  
        "secret": "secret",  
        "graduated": "graduated",  
        "bachelor": "bachelor",  
        "master": "master",  
        "doctor": "doctor"  
},
```

4. 编译运行该应用

- 通过 hdc 连接大屏设备

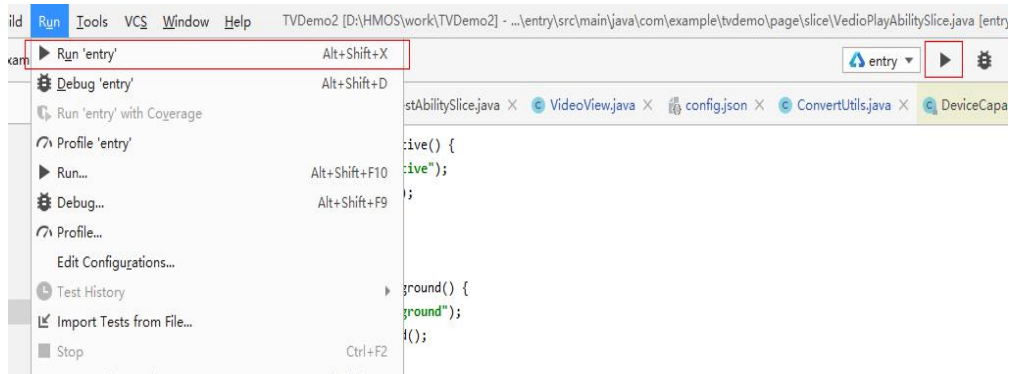
先查看智慧屏 IP:

大屏设置->"网络与连接"->"网络"->"有线网络"

在 cmd 或者 IDE 的 Terminal 输入命令:

```
hdc tconn 192.168.3.9:5555
```

- 运行 hap



5. 总结

您已经成功完成了 HarmonyOS 应用开发 E2E 体验，学到了：

- 如何创建一个 HarmonyOS Demo Project
- 如何构建一个 Hap 并且将其部署到真机
- 在 HarmonyOS 上如何使用基本控件，包括文本输入框，日期选择控件，单选按钮，下拉菜单和按钮。

三、应用偏好数据读写

1. 介绍

本篇 Codelab 将实现的内容

HarmonyOS 是面向全场景多终端的分布式操作系统，使得应用程序的开发打破了智能终端互通的性能和数据壁垒，业务逻辑原子化开发，适配多端。通过一个简单应用开发，体验 HarmonyOS 的轻量级偏好数据库能力

您将建立什么

在这个 Codelab 中，你将创建 Demo Project，并将 Demo 编译成 Hap，此示例应用程序展示了如何使用轻量级偏好数据库。

您将会学到什么

- 如何创建一个 HarmonyOS Demo Project
- 如何构建一个 Hap 并且将其部署到智慧屏真机
- 通过此示例应用体验如何使用轻量级偏好数据库

2. 您需要什么

硬件要求

- 操作系统：Windows10 64 位
- 内存：8G 及以上。
- 硬盘：100G 及以上。
- 分辨率：1280*800 及以上

软件要求

- DevEco Studio: 需手动下载安装，详细步骤请参考《DevEco Studio 使用指南》2.1.2
- JDK: DevEco Studio 自动安装。
- Node.js: 请手动下载安装，详细步骤请参考《DevEco Studio 使用指南》2.1.3 下载和安装 Node.js。
- HarmonyOS SDK: 待 DevEco Studio 安装完成后，利用 DevEco Studio 来加载 HarmonyOS SDK。详细步骤请参考《DevEco Studio 使用指南》2.1.6 加载 HarmonyOS SDK。
- Maven 库依赖包: 如需手动拷贝和配置，详细步骤请参考《DevEco Studio 使用指南》2.3 离线方式配置 Maven 库。

需要的知识点

- Java 基础开发能力。

3. 能力接入准备

实现 HarmonyOS 应用开发，需要完成以下准备工作：

- 环境准备。
- 环境搭建。
- 创建项目
- 申请调试证书
- 应用开发

具体操作，请按照《DevEco Studio 使用指南》中详细说明来完成。

提示：需要通过注册成开发者才能完成集成准备中的操作。

4. 代码片段

1. 布局：

- 布局代码：

```
•   LayoutConfig config = new
LayoutConfig(LayoutConfig.MATCH_PARENT,
LayoutConfig.MATCH_PARENT);
•   myLayout.setLayoutConfig(config);
•   myLayout.setOrientation(Component.VERTICAL);
•   ShapeElement element = new ShapeElement();
•   element.setRgbColor(new RgbColor(255, 255, 255));
•   myLayout.setBackground(element);
•   log = createText("日志信息");
•   myLayout.addComponent(log);
•   writeBtn = createBtn("写入 preferences 数据", new RgbColor(0, 0,
255), 1002);
•   readBtn = createBtn("读取 preferences 数据", new RgbColor(0, 0, 255),
1003);
•   addObserver = createBtn("注册观察者", new RgbColor(255, 0, 0),
1004);
•   private Text createText(String title) {
•   Text text = new Text(this);
```

```
• DirectionalLayout.LayoutConfig config = new
  DirectionalLayout.LayoutConfig(DirectionalLayout.LayoutConfig.MATC
    H_CONTENT, DirectionalLayout.LayoutConfig.MATCH_CONTENT);
• text.setLayoutConfig(config);
• text.setText(title);
• text.setTextSize(48);
• text.setTextColor(new Color(0xFF0000FF));
• return text;
• }
• private Button createBtn(String title, RgbColor color, int id) {
•   Button btn = new Button(this);
•   LayoutConfig configBtn = new LayoutConfig(500, 100);
•   configBtn.topMargin = 30;
•   btn.setLayoutConfig(configBtn);
•   btn.setText(title);
•   btn.setId(id);
•   btn.setTextSize(48);
•   btn.setTextColor(new Color(0xFFFFFFFF));
•   ShapeElement elementBtn = new ShapeElement();
•   elementBtn.setRgbColor(color);
•   elementBtn.setCornerRadius(12);
•   btn.setBackground(elementBtn);
•   myLayout.addComponent(btn);
•   return btn;
• }
```

2. Preferences 使用

- Preferences 初始化

```
• private void initPreferences() {
•   DatabaseHelper databaseHelper = new DatabaseHelper(this);
•   String fileName = "user_info";
•   preferences = databaseHelper.getPreferences(fileName);
• }
```

- 写文件:

```
• preferences.putInt("age", Integer.parseInt(age.getText()));
• preferences.putString("name", name.getText());
• preferences.flushSync();
```

- 读文件:

```
• int age = preferences.getInt("age", 0);
• String name = preferences.getString("name", "");
```

- `AlertDialog toastDialog = new AlertDialog(PreferencesAbilitySlice.this);`
- `toastDialog.setText("read user data from preferences name:" + name + ", age:" + age);`
- `toastDialog.show();`

- 观察者:

注册:

```
counter = new PreferencesChangeCounter();
preferences.registerObserver(counter);
private class PreferencesChangeCounter implements
Preferences.PreferencesObserver {
    @Override
    public void onChange(Preferences preferences, String key) {
        if ("name".equals(key)) {
            String name = preferences.getString("name", "");
            log.setText("user data name is edit:" + name);
        }
        if ("age".equals(key)) {
            int age = preferences.getInt("age", 0);
            log.setText("user data age is edit:" + age);
        }
    }
}
```

删除:

```
preferences.unregisterObserver(counter);
```

2. 响应遥控器点击

```
private void addFocusChangeListener(Component view) {
    view.setFocusChangeListener(new Component.FocusChangeListener() {
        @Override
        public void onFocusChange(Component component, boolean b) {
            ShapeElement shapeElement = (ShapeElement)
view.getBackgroundElement();
            if (b) {
                shapeElement.setStroke(10, new RgbColor(0, 0, 0));
                focusView = view;
            } else {

```

```

        shapeElement.setStroke(0, new RgbColor(0, 0, 0));
    }
}

});
}

@Override
public boolean onKeyUp(int keyCode, KeyEvent keyEvent) {
    switch (keyCode) {
        case KeyEvent.KEY_DPAD_CENTER:
        case KeyEvent.KEY_ENTER:
            if(focusView == writeBtn) {

                preferences.putInt("age", index++);
                preferences.putString("name", "张三");
                preferences.flushSync();
            }
            if(focusView == readBtn) {
                int age = preferences.getInt("age", 0);
                String name = preferences.getString("name", "");
                log.setText("read user data from preferences name:" +
name + ", age:" + age);
            }
            if(focusView == addObserver) {
                if (addObserver.getText().equals("注册观察者")) {
                    addObserver.setText("删除观察者");
                    // 向 preferences 实例注册观察者
                    counter = new PreferencesChangeCounter();
                    preferences.registerObserver(counter);

                } else {
                    addObserver.setText("注册观察者");
                    // 向 preferences 实例注销观察者
                    preferences.unregisterObserver(counter);
                }
            }
            return true;
        case KeyEvent.KEY_DPAD_UP:
            int position = views.indexOf(focusView.getId());
            if (position > 0) {
                switch (position) {
                    case 1:

```

```
        writeBtn.requestFocus();
        break;
    case 2:
        readBtn.requestFocus();
        break;
    default:
        break;
    }
}
return true;
case KeyEvent.KEY_DPAD_DOWN:
    position = views.indexOf(focusView.getId());
    if (position < 3) {
        switch (position) {
            case 0:
                readBtn.requestFocus();
                break;
            case 1:
                addObserver.requestFocus();
                break;
            default:
                break;
        }
    }
    return true;
}
return false;
}
```

3.编译运行该应用

- 通过 hdc 连接大屏设备

先查看智慧屏 IP:

大屏设置->"网络与连接"->"网络"->"有线网络"

在 cmd 或者 IDE 的 Terminal 输入命令:

hdc tconn 192.168.3.9:5555

- 运行 hap



5. 恭喜你

你已经成功完成了 HarmonyOS 应用开发 E2E 体验，学到了：

- 如何创建一个 HarmonyOS Demo Project
- 如何构建一个 Hap 并且将其部署到真机上
- 在 HarmonyOS 上如何使用 HarmonyOS 的轻量级偏好数据库

四、剪切板

1. 介绍

本篇 **CodeLab** 将实现的内容

HarmonyOS 是面向全场景多终端的分布式操作系统，使得应用程序的开发打破了智能终端互通的性能和数据壁垒，业务逻辑原子化开发，适配多端。通过一个简单应用开发，体验 **HarmonyOS** 的剪切板能力。

您将建立什么

在这个 **CodeLab** 中，你将创建 **Demo Project**，并将 **Demo** 编译成 **HAP**，此示例应用程序展示了如何使用 **HarmonyOS** 剪切板复制文字。

您将会学到什么

- 如何创建一个 **HarmonyOS Demo Project**
- 如何构建一个 **HAP** 并且将其部署到智慧屏上
- 通过此示例应用体验剪切板复制粘贴文本。

2. 您需要什么

硬件要求

- 操作系统：Windows10 64 位
- 内存：8G 及以上
- 硬盘：100G 及以上
- 分辨率：1280*800 及以上

软件要求

- 安装 DevEco Studio 和 Node.js，详情请参考[下载和安装软件](#)

- 设置 DevEco Studio 开发环境，DevEco Studio 开发环境需要依赖于网络环境，需要连接上网络才能确保工具的正常使用，可以根据如下两种情况来配置开发环境
 1. 如果可以访问 Internet，只需进行[下载 HarmonyOS SDK](#) 操作
 2. 如果网络不能直接访问 Internet，需要通过代理服务器才可以访问，请参考[配置开发环境](#)

需要的知识点

- Java 基础开发能力。

3. 能力接入准备

实现 HarmonyOS 应用开发，需要完成以下准备工作：

- [创建 TV 的工程](#)
- [准备密钥和证书请求文件](#)
- [申请调试证书](#)
- 应用开发

具体操作，请按照《DevEco Studio 使用指南》中详细说明来完成。

4. 代码编写

实现"复制文本"功能，代码片段如下：

```
private static final String ADDITON_KEY = "ADDITION_KEY";

mPasteboard = SystemPasteboard.getSystemPasteboard(this);
PasteData pasteData = new PasteData();
pasteData.addTextRecord("copyText");
PacMap pacMap = new PacMap();
pacMap.putString(ADDITION_KEY, "ADDITION_VALUE_OF_TEXT");
pasteData.getProperty().setAdditions(pacMap);
pasteData.getProperty().setTag("USER_TAG");
pasteData.getProperty().setLocalOnly(true);
mPasteboard.setPasteData(pasteData);
mShowText.setText("copy text succeed");
```

实现"粘贴文本"功能，代码片段如下：

```
private static final String ADDITON_KEY = "ADDITION_KEY";

mPasteboard = SystemPasteboard.getSystemPasteboard(this);
mPasteData = mPasteboard.getPasteData();
mRecord = mPasteData.getRecordAt(0);
mShowText.append(mRecord.getPlainText().toString());
mShowText.append(mRecord.getMimeType());
mShowText.append(mPasteData.getProperty().getTag().toString());
PacMap pacMap = mPasteData.getProperty().getAdditions();
String extraInfo = pacMap.getString(ADDITON_KEY);
if ((extraInfo != null && !extraInfo.isEmpty())) {
    mShowText.setText("value is " + extraInfo);
}
```

提示：以上代码仅是 demo 演示参考使用

5. 编译运行

通过 hdc 连接大屏设备

先查看智慧屏 IP：大屏设置->"网络与连接"->"网络"->"无线网络"

在 cmd 或者 IDE 的 Terminal 输入命令：

```
hdc tconn 192.168.xxx.xxx:5555
```

运行



6. 恭喜您

您已经成功完成了 **HarmonyOS** 应用开发体验，学到了：

- 如何创建一个 HarmonyOS Demo Project
- 如何构建一个 HAP 并且将其部署到真机上
- 在 HarmonyOS 上如何使用剪切板复制粘贴文本

五、分布式任务调度

1. 介绍

本篇 **CodeLab** 将实现的内容

HarmonyOS 是面向全场景多终端的分布式操作系统，使得应用程序的开发打破了智能终端互通的性能和数据壁垒，业务逻辑原子化开发，适配多端。通过一个简单应用开发，体验 **HarmonyOS** 的分布式任务调度。

您将建立什么

在这个 **CodeLab** 中，你将创建 **Demo Project**，并将 **Demo** 编译成 **HAP**，此示例应用程序展示了如何使用分布式任务调度。

您将会学到什么

- 如何创建一个 HarmonyOS Demo Project
- 如何构建一个 HAP 并且将其部署到智慧屏真机
- 通过此示例应用体验如何使用分布式任务调度

2. 您需要什么

硬件要求

- 操作系统：Windows10 64 位

- 内存：8G 及以上
- 硬盘：100G 及以上
- 分辨率：1280*800 及以上

软件要求

- 安装 DevEco Studio 和 Node.js，详情请参考[下载和安装软件](#)
- 设置 DevEco Studio 开发环境，DevEco Studio 开发环境需要依赖于网络环境，需要连接上网络才能确保工具的正常使用，可以根据如下两种情况来配置开发环境
 1. 如果可以直接访问 Internet，只需进行[下载 HarmonyOS SDK](#) 操作
 2. 如果网络不能直接访问 Internet，需要通过代理服务器才可以访问，请参考[配置开发环境](#)

需要的知识点

- Java 基础开发能力。

3. 能力接入准备

实现 HarmonyOS 应用开发，需要完成以下准备工作：

- [创建 TV 的工程](#)
- [准备密钥和证书请求文件](#)
- [申请调试证书](#)
- 应用开发

具体操作，请按照《DevEco Studio 使用指南》中详细说明来完成。

4. 代码编写

分布式任务调用代码参考：

```
Intent intent = new Intent();  
// BUNDLE_NAME 和 ABILITY_NAME 对应开发者需要进行分布式任务调度的  
Ability 信息  
Operation operation = new Intent.OperationBuilder()  
    .withDeviceId(info.getDeviceId())  
    .withBundleName(BUNDLE_NAME)
```

```

        .withAbilityName(ABILITY_NAME)
        .withFlags(Intent.FLAG_ABILITYSLICE_MULTI_DEVICE)
        .build();
intent.setOperation(operation);
try {
    // FLAGS_TO_QUERY 和 USERID_TO_QUERY 分别对应查询 ability 的
    // flags 和 userid，具体可以参考 API-DOC
    List<AbilityInfo> abilityInfos =
        getBundleManager().queryAbilityByIntent(intent, 0, 0);
    if (abilityInfos != null && !abilityInfos.isEmpty()) {
        startAbility(intent);
    }
} catch (RemoteException re) {
    HiLog.error(TAG, "RemoteException");
}

```

提示：以上代码仅 demo 演示参考使用

5. 编译运行

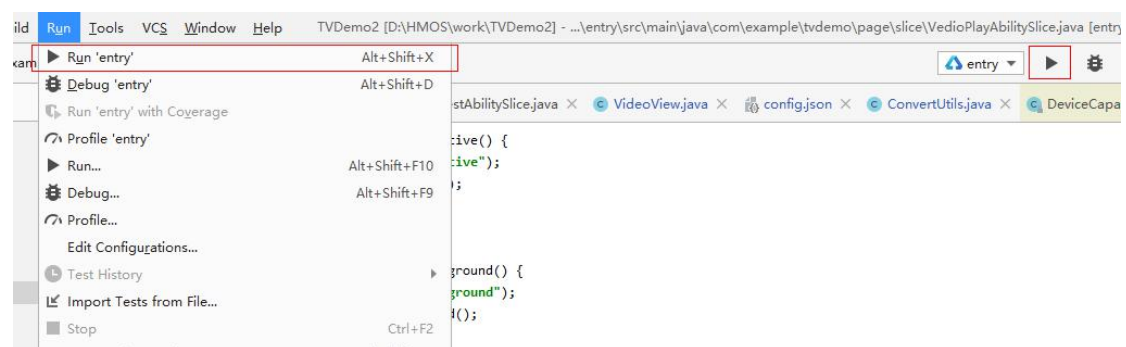
通过 hdc 连接大屏设备

先查看智慧屏 IP：大屏设置->"网络与连接"->"网络"->"无线网络"

在 cmd 或者 IDE 的 Terminal 输入命令：

```
hdc tconn 192.168.xxx.xxx:5555
```

运行



6. 恭喜您

您已经成功完成了 **HarmonyOS** 应用开发体验，学到了：

- 如何创建一个 **HarmonyOS Demo Project**
- 如何构建一个 **HAP** 并且将其部署到真机上
- 在 **HarmonyOS** 上如何使用分布式任务调度能力

六、元程序交互

1. 介绍

本篇 **CodeLab** 将实现的内容

HarmonyOS 是面向全场景多终端的分布式操作系统，使得应用程序的开发打破了智能终端互通的性能和数据壁垒，业务逻辑原子化开发，适配多端。通过一个简单应用开发，体验 **HarmonyOS** 的元程序调度能力。

- 有界面元程序 **A** 拉起另外一个有界面元程序 **B**，在元程序 **B** 上可以读到的意图。
- 元程序 **B** 可以回数据给元程序 **A**，元程序 **A** 收到的返回信息。

您将建立什么

在这个 **CodeLab** 中，你将创建 **Demo Project**，并将 **Demo** 编译成 **HAP**，此示例应用程序展示了如何使用有界面元程序。

您将会学到什么

- 如何创建一个 **HarmonyOS Demo Project**
- 如何通过实现界面跳转以及数据传递

2. 您需要什么

硬件要求

- 操作系统：Windows10 64 位
- 内存：8G 及以上
- 硬盘：100G 及以上
- 分辨率：1280*800 及以上

软件要求

- 安装 DevEco Studio 和 Node.js，详情请参考[下载和安装软件](#)
- 设置 DevEco Studio 开发环境，DevEco Studio 开发环境需要依赖于网络环境，需要连接上网络才能确保工具的正常使用，可以根据如下两种情况来配置开发环境
 1. 如果可以直接访问 Internet，只需进行[下载 HarmonyOS SDK](#) 操作
 2. 如果网络不能直接访问 Internet，需要通过代理服务器才可以访问，请参考[配置开发环境](#)

需要的知识点

- Java 基础开发能力。

3. 能力接入准备

实现 HarmonyOS 应用开发，需要完成以下准备工作：

- [创建 TV 的工程](#)
- [准备密钥和证书请求文件](#)
- [申请调试证书](#)
- 应用开发

具体操作，请按照《DevEco Studio 使用指南》中详细说明来完成。

4. 代码编写

核心代码参考

```
Operation operation = new Intent.OperationBuilder()
    .withDeviceld("")
    .withBundleName("com.huawei.codelab")
    .withAbilityName("com.huawei.codelab.CalleeAbility")
    .build();
Intent intent = new Intent();
intent.setOperation(operation);
startAbility(intent);
```

提示：以上代码仅 demo 演示参考使用

5. 编译运行

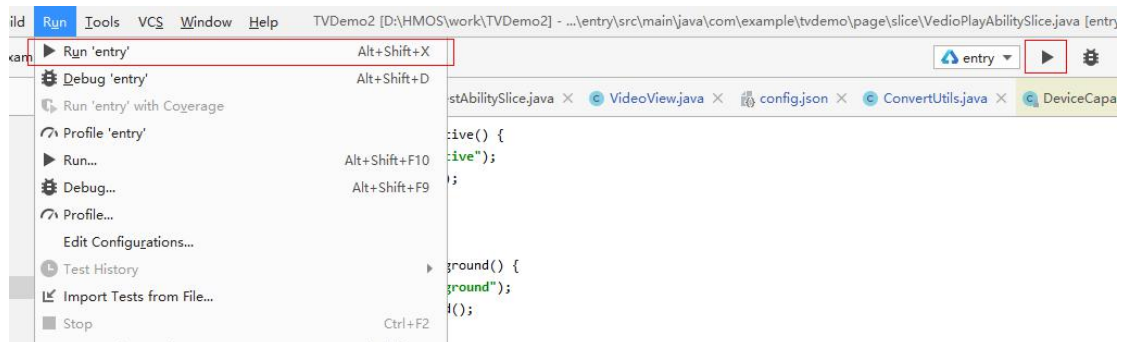
通过 hdc 连接大屏设备

先查看智慧屏 IP：大屏设置->"网络与连接"->"网络"->"无线网络"

在 cmd 或者 IDE 的 Terminal 输入命令：

```
hdc tconn 192.168.xxx.xxx:5555
```

运行



6. 恭喜您

您已经成功完成了 **HarmonyOS** 应用开发体验，学到了：

- 如何创建一个 HarmonyOS Demo Project
- 如何构建一个 HAP 并且将其部署到真机上
- 如何实现元能力界面跳转

七、UI 设计开发与预览

1. 介绍

通过智能表待办应用开发，让开发者了解智能表 **HarmonyOS** 应用开发的全流程，实现从工程创建到界面预览全过程。使用 **HUAWEI DevEco Studio** 开发 **HarmonyOS** 待办应用，完成工程创建、代码编辑，界面预览等开发过程。

您将建立什么

在这个 **CodeLab** 中，你将创建一个智能表的 **Demo Project (harmony-todo)**，以及完成一个待办应用的页面的搭建和预览。

您将学到什么

- 如何搭建一个 **APP** 并添加页面布局
- 如何实时预览创建的页面布局信息
- 完成智能表应用的页面搭建和预览

硬件要求

- 操作系统：Windows10 64 位
- 内存：8G 及以上
- 硬盘：100G 及以上
- 分辨率：1280*800 及以上

软件要求

- 安装 DevEco Studio 和 Node.js，详情请参考[下载和安装软件](#)

提示：智能表 **UI** 开发的预览功能将在 **Beta2** 版本上线，当前只能在 **CodeLab** 现场体验尝鲜

- 设置 DevEco Studio 开发环境，DevEco Studio 开发环境需要依赖于网络环境，需要连接上网络才能确保工具的正常使用的，可以根据如下两种情况来配置开发环境
- 如果可以直接访问 Internet，只需进行[下载 HarmonyOS SDK](#) 操作

- 如果网络不能直接访问 Internet，需要通过代理服务器才可以访问，请参考[配置开发环境](#)

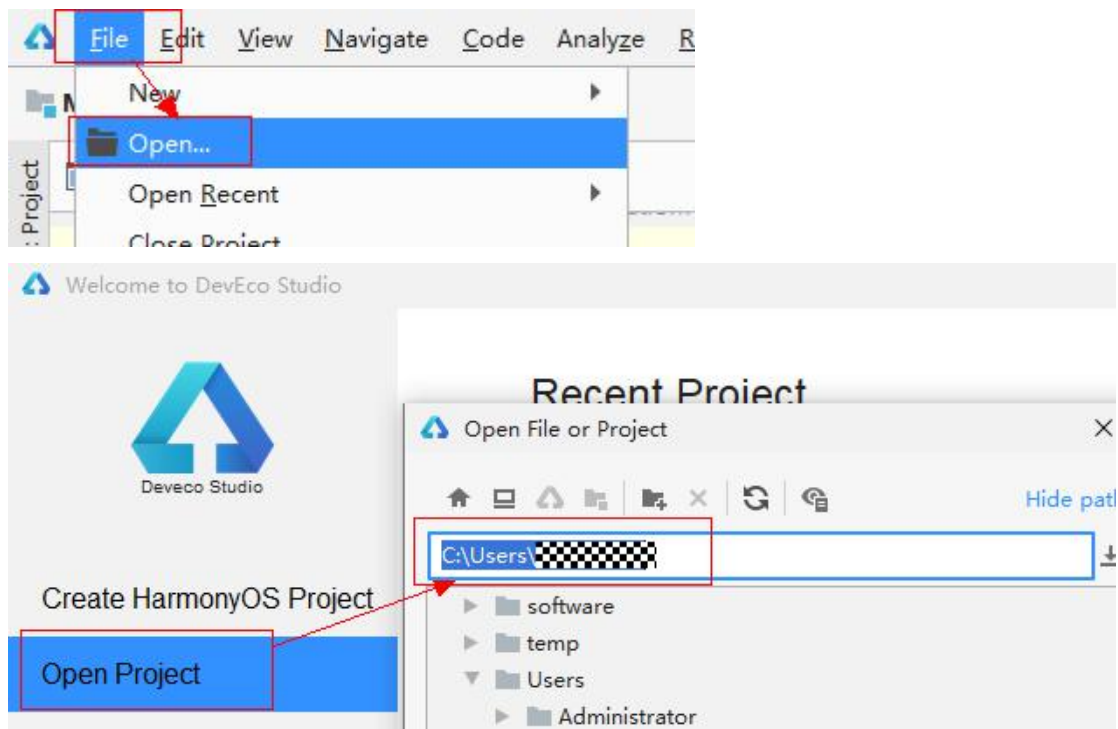
提示：下载 HarmonyOS SDK 时，需要下载 JS SDK 和 SDK Tools 中的 Previewer

需要的知识点

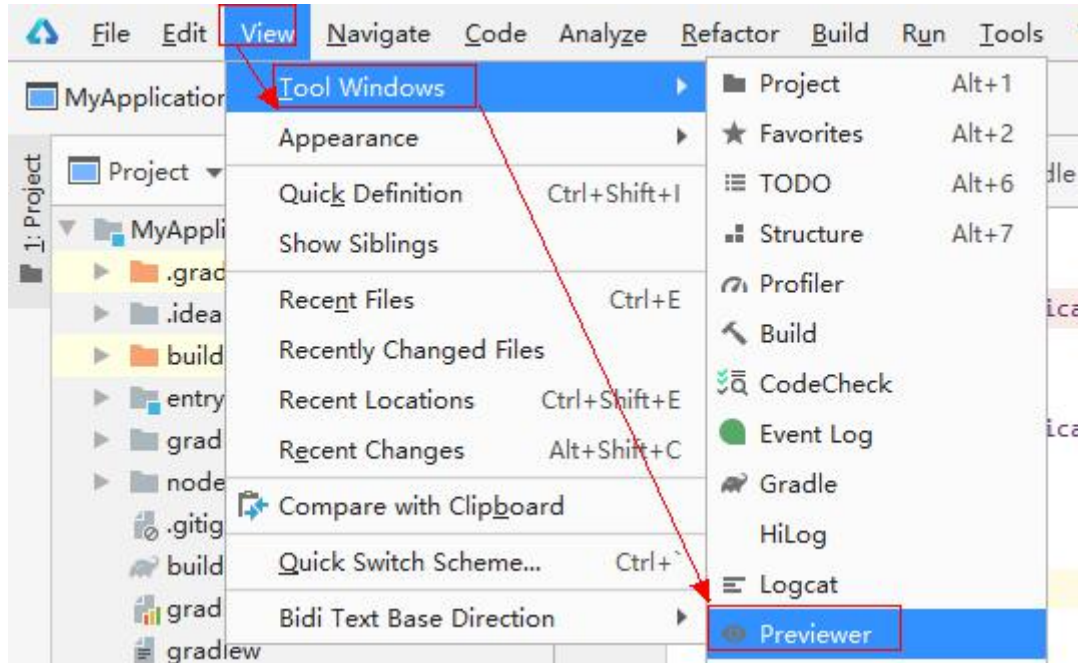
- html、CSS、JavaScript 开发基础能力

2. 代码开发

1.打开本地 Demo Project (harmony-todo)

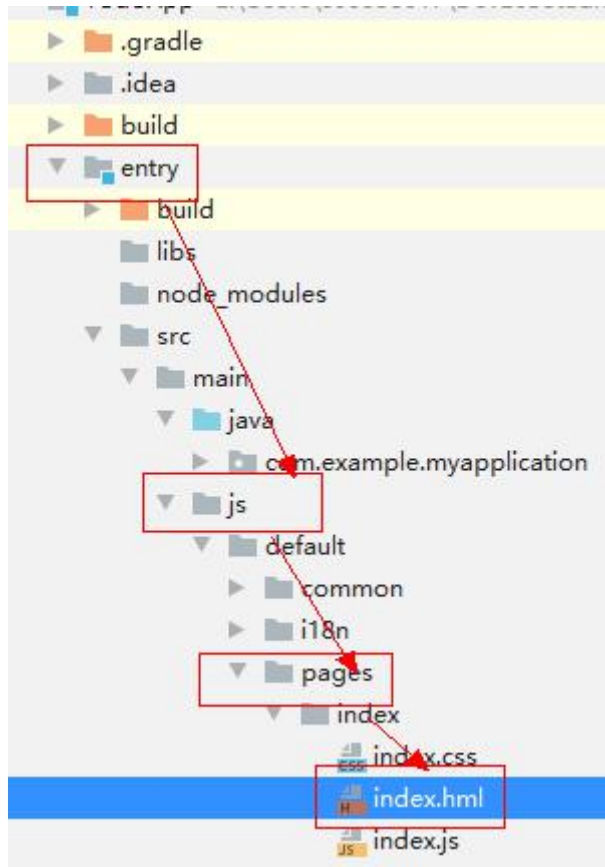


2. 点击 **Previewer** 按钮，实时预览 **Demo Project** (harmony-todo)



提示：开发过程完成每一步点击保存之后即可在预览界面实时预览效果

3.为 index 页面（index.html）添加布局信息



3.1 添加今日待办事项的列表

```
<todo-header title="今日待办" type=""
@add-event="toAddEvent"></todo-header>
<list-item for="{{todayList}}" class="tag-list-item"
clickeffect="false">
    <todo-list @complete-event="completeEvent"
@delete-event="deleteEvent" todo="{{item}}"
        todos="{{todayList}}" index="{{idx}}"></todo-list>
</list-item>
```

提示：以上代码只是 **demo** 演示，产品化的代码需要使用国际化

预览效果如下所示：



3.2 添加明日待办事项的列表

```
<todo-header title="明日待办" type=""
@add-event="toAddEvent"></todo-header>
<list-item for="{{tomorrowList}}" class="tag-list-item"
clickeffect="false">
    <todo-list @complete-event="completeEvent"
@delete-event="deleteEvent" todo="{{${item}}"
        todos="{{tomorrowList}}" index="{{${idx}}}"></todo-list>
</list-item>
```

提示：以上代码只是 **demo** 演示，产品化的代码需要使用国际化

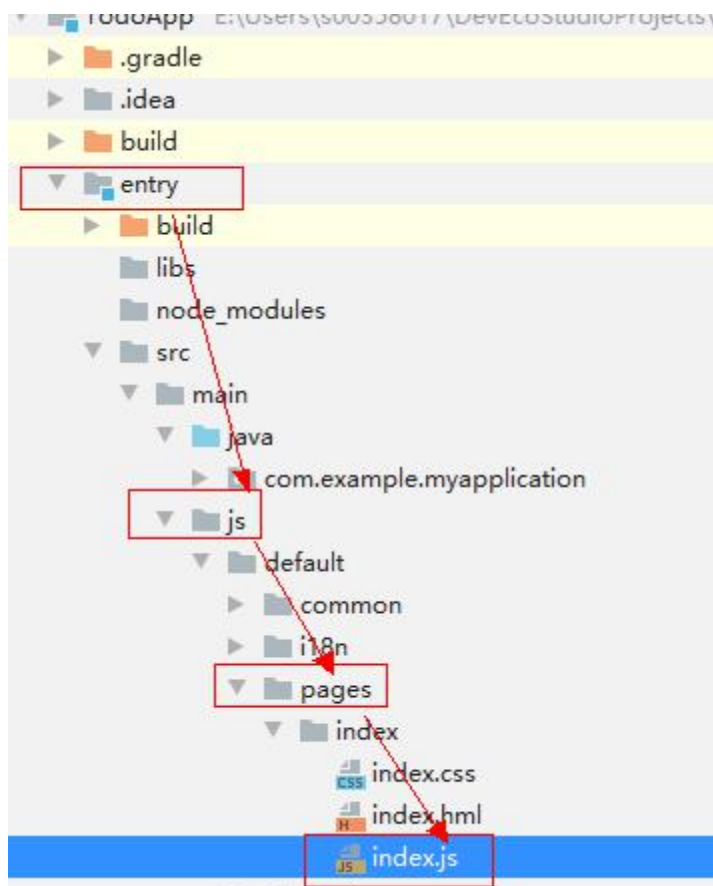


3.3 添加即将来临待办事项的列表

```
<todo-header title="即将来临" type=""
@add-event="toAddEvent"></todo-header>
<list-item for="{{laterList}}" class="tag-list-item"
clickeffect="false">
    <todo-list @complete-event="completeEvent"
@delete-event="deleteEvent" todo="{{item}}"
        todos="{{laterList}}" index="{{idx}}"></todo-list>
</list-item>
```

提示：以上代码只是 **demo** 演示，产品化的代码需要使用国际化

4.添加逻辑代码(index.js)



4.1 添加待办事项完成的逻辑代码

```
completeEvent(clicked) {
```



```
var lists = clicked.detail.lists;
var eid = clicked.detail.id;
datahelper.completeEvent(eid, lists);
},
```

4.2 添加删除待办事项的逻辑代码

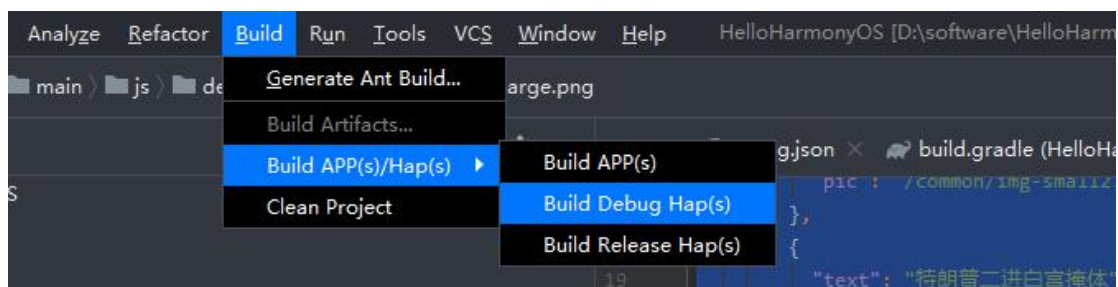
```
deleteEvent(clicked) {
    var index = clicked.detail.index;
    var lists = clicked.detail.lists;
    lists.splice(index, 1);
},
```

预览效果如下图所示：



3. 编译构建

点击 Build > Build APP(s)/Hap(s) > Build Debug Hap(s)，打包生成 **hap** 包



4. 恭喜您

您已经成功完成了 **CodeLab**，并学到了：

- 如何开发一个智能表应用
- 如何实时预览开发的 UI 界面
- 完成 UI 设计开发与预览的整体流程。

八、HelloWorld 应用开发 E2E 体验

1. 介绍

智慧屏是首个搭载 **HarmonyOS** 的终端产品，通过 **HarmonyOS** 智慧屏多页签应用开发模板，让开发者了解 **HarmonyOS** 应用开发的全流程，20 分钟快速上手，实现从工程创建到应用运行全过程。

您将建立什么

在这个 **CodeLab** 中，您将使用 **HUAWEI DevEco Studio** 开发 **HarmonyOS** 智慧屏多页签应用，完成工程创建、编译构建，并实现 **HarmonyOS** 智慧屏部署和运行。

您将学到什么

- 如何创建一个 **HarmonyOS Project**
- 编译构建 **hap** 包
- 将 **hap** 包部署到智慧屏远程模拟器上，并运行

硬件要求

- 操作系统：Windows10 64 位
- 内存：8G 及以上
- 硬盘：100G 及以上
- 分辨率：1280*800 及以上

软件要求

- 安装 **DevEco Studio** 和 **Node.js**，详情请参考[下载和安装软件](#)
- 设置 **DevEco Studio** 开发环境，**DevEco Studio** 开发环境需要依赖于网络环境，需要连接上网络才能确保工具的正常使用，可以根据如下两种情况来配置开发环境
 - 1.如果可以直接访问 **Internet**，只需进行[下载 HarmonyOS SDK](#) 操作
 - 2.如果网络不能直接访问 **Internet**，需要通过代理服务器才可以访问，请参考[配置开发环境](#)

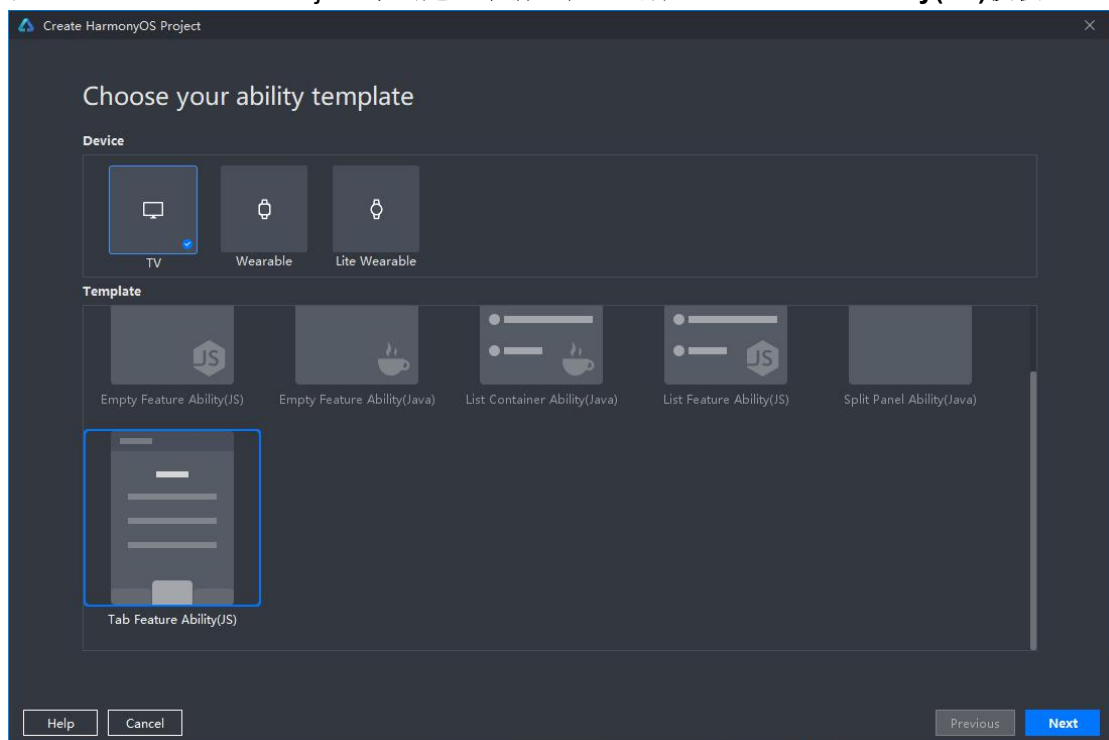
技能要求

- Java 基础开发能力
- JavaScript/HML(HarmonyOS Markup Language) /CSS 基础开发能力

2. 代码开发

1. 工程创建

在 File > New > New Project 来创建一个新工程，选择 **Tab Feature Ability(JS)**模板：



填写工程基本信息，如命名工程名和包名：

Create HarmonyOS Project

Configure your project

Project Name
HelloHarmonyOS

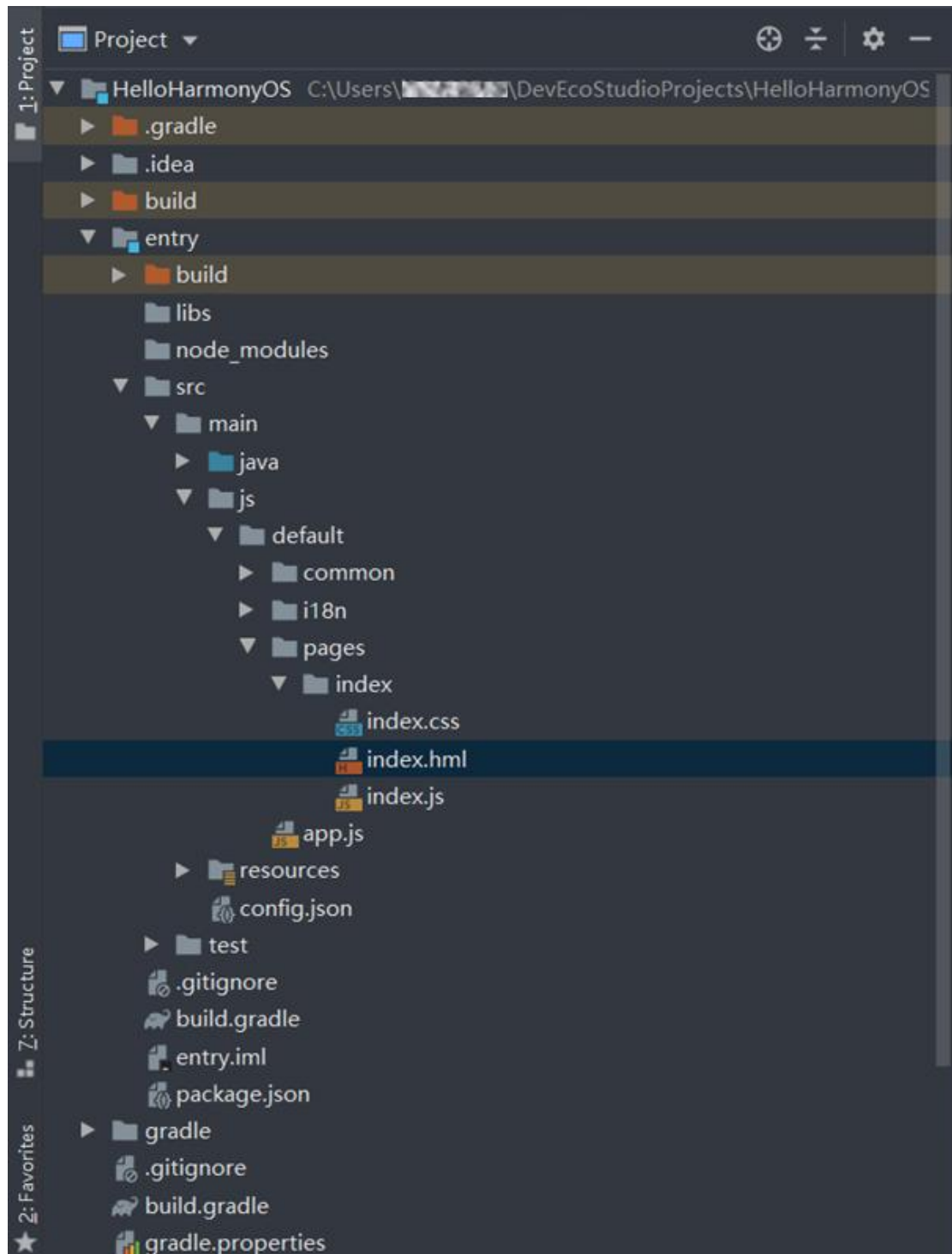
Package name
com.example.helloharmonyos

Save location
C:\Users\user\DevEcoStudioProjects\HelloHarmonyOS

Compatible SDK
SDK: API Version 3

Help Cancel Previous Finish

工程创建完成后，目录结构如下：



目录结构中文件分类及作用：

- .html 文件布局结构描述文件
- .css 页面样式描述文件
- .js 页面显示和用户交互文件
- app.js 用于全局应用生命周期管理

- **pages** 用于存放组件页面
- **common** 用于存放公共资源文件。如：媒体资源、自定义组件和 JS 文件
- **resources** 用于存放资源配置文件。如：全局样式、多分辨率加载等配置文件
- **i18n** 用于存放全球化资源

2. 编码

本次 **CodeLab** 需要编写布局文件、添加图片资源以及修改模块配置。

2.1 开发多页签界面布局及全球化资源

1、编辑布局文件 entry/src/main/js/default/pages/index/index.html

```
<div class="container">
  <div class="tv_box">
    <div class="title_box">
      <text class="title">{{ $t(' Strings.title' ) }}
    </text>
    <button type="circle" icon="{{ icon_src }}"
class="setting_box" onfocus="iconFocusFunc"
      onblur="iconBlurFunc"></button>
    </div>
    <tabs class="tab_box">
      <tab-bar mode="scrollable" class="bar_box">
        <block for="{{ item in $t(' Strings.tab' ) }}">
          <text class="tab_text">{{ item }}
        </text>
        </block>
      </tab-bar>
      <tab-content>
        <block for="[1, 2, 3, 4, 5, 6, 7, 8, 9]">
          <div class="content_box">
            <list class="content_img">
              <block for="{{ imgIndex in
$t(' Strings.images' ) }}">
                <list-item type="listItem"
class="list_img">
                  <image focusable="true"
class="tab_img" src="{{ imgIndex }}"></image>
                </list-item>
              </block>
            </list>
          </div>
        </block>
      </tab-content>
    </tabs>
  </div>
</div>
```

```

        </list>
        <div class="subtitle_box">
            <text
class="subtitle">{{ $t(' Strings.subtitle' ) }}
            </text>
        </div>
        <list class="img_list">
            <block for="{{detailItem in
$t(' Strings.details' ) }}">
                <list-item type="listItem"
class="list_box">
                    <image focusable="true"
class="img_img" src="{{detailItem.pic }}"></image>
                    <text
class="img_text">{{detailItem.text}}
                    </text>
                </list-item>
            </block>
        </list>
    </div>
</block>
</tab-content>
</tabs>
</div>
</div>

```

2、编辑全球化资源文件 entry/src/main/js/default/i18n/zh-CN. json

```

{
  "Strings": {
    "title": "华为智慧屏",
    "subtitle": "历史观看",
    "tab": [
      "首页",
      "电影",
      "电视剧",
      "购物",
      "溪村风景",
      "图册",
      "少儿",
      "VIP"
    ],
    "images": [

```



```

        "/common/img-large1.png",
        "/common/img-large2.png",
        "/common/img-large3.png",
        "/common/img-large4.png"
    ],
    "text": "文本内容",
    "details": [
        {
            "text": "花园",
            "pic": "/common/img-small1.png"
        },
        {
            "text": "风景一角",
            "pic": "/common/img-small2.png"
        },
        {
            "text": "蓝天白云",
            "pic": "/common/img-small3.png"
        },
        {
            "text": "池塘",
            "pic": "/common/img-small4.png"
        },
        {
            "text": "办公一角",
            "pic": "/common/img-small5.png"
        }
    ]
},
"Files": {}
}

```

2.2 添加图片资源

将 **CodeLab PC** 桌面 CodeLab/common 目录下的资源拷贝到 entry/src/main/js/default/commom。

注：可以在 common 目录点击右键，通过 **Show in Explorer** 快速进入目录。

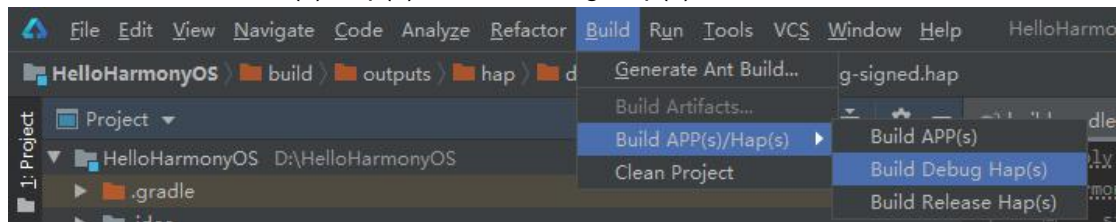
2.3 修改模块配置文件

修改 entry/src/main/config.json 文件，将 **designWidth** 的值修改为 **1024**，并将 **autoDesignWidth** 的值修改为 **false**。

```
"js": [  
  {  
    "pages": [  
      "pages/index/index"  
    ],  
    "name": "default",  
    "window": {  
      "designWidth": 1024,  
      "autoDesignWidth": false  
    }  
  }  
]
```

3. 编译构建

点击 Build > Build APP(s)/Hap(s) > Build Debug Hap(s) 进行代码编译构建，

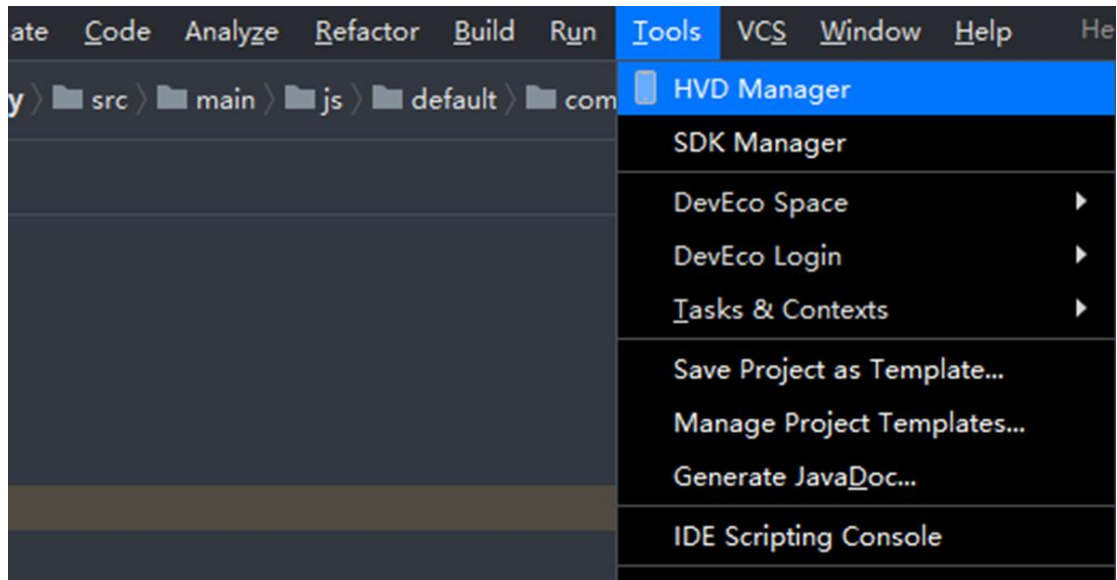


等待系统编译，在控制台看到编译成功提示信息即可。

4. 部署并运行

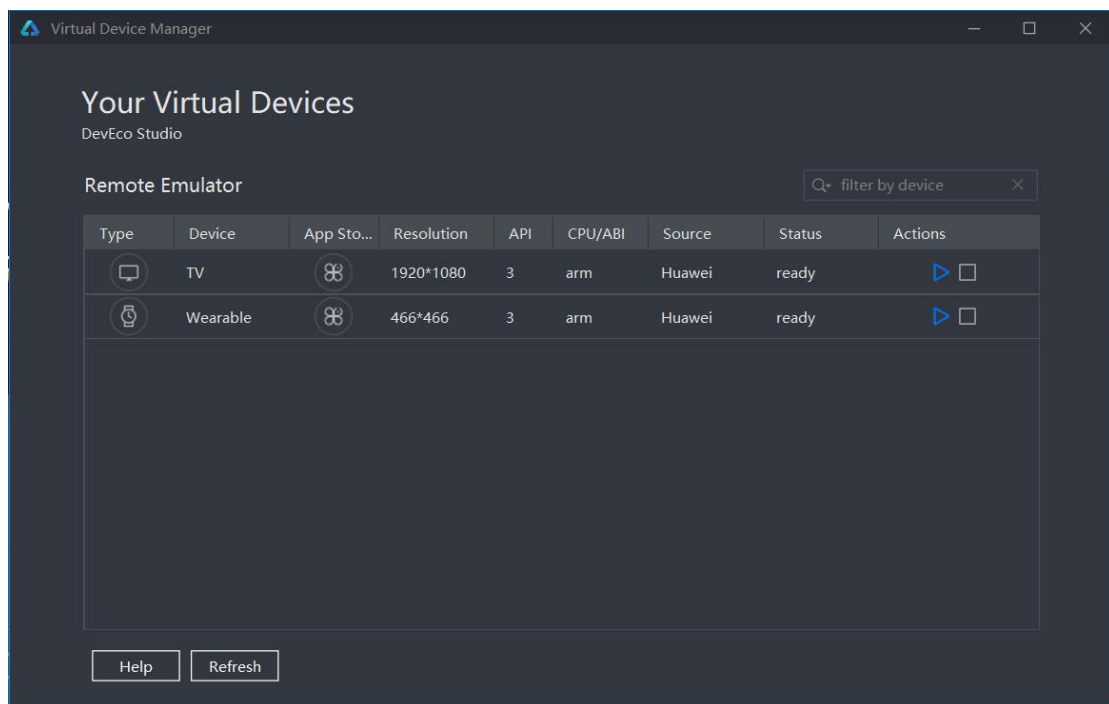
1. 启动模拟器

点击 **Tools>HVD Manager** 启动模拟器。

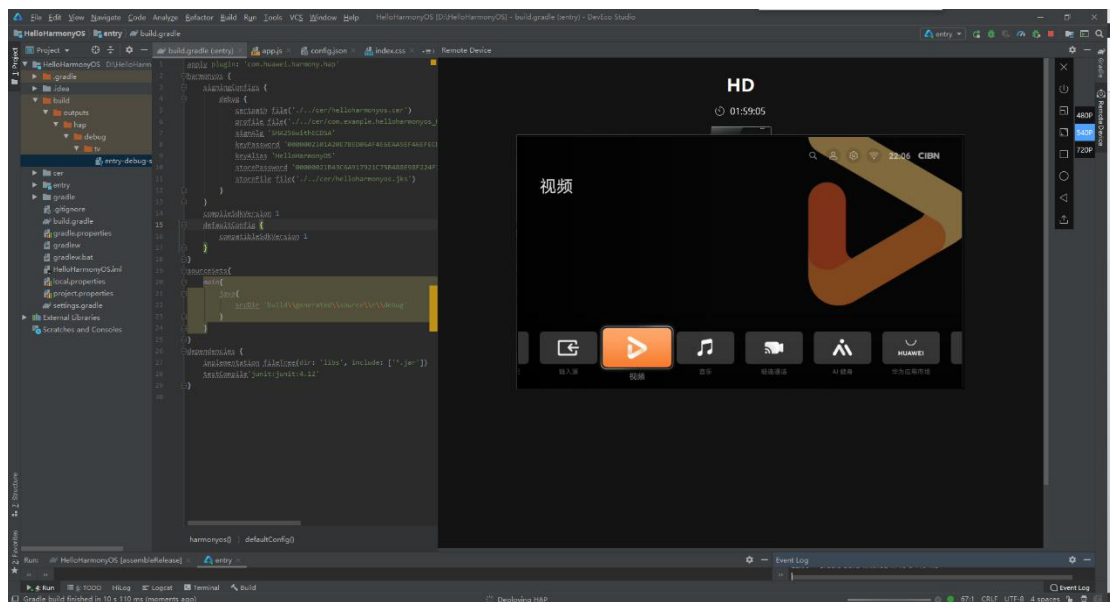


注：使用模拟器需要登录实名认证的华为开发者账号，请按照提示完成登录及授权。

选择智慧屏模拟器（TV），点击启动按钮，如下图所示。

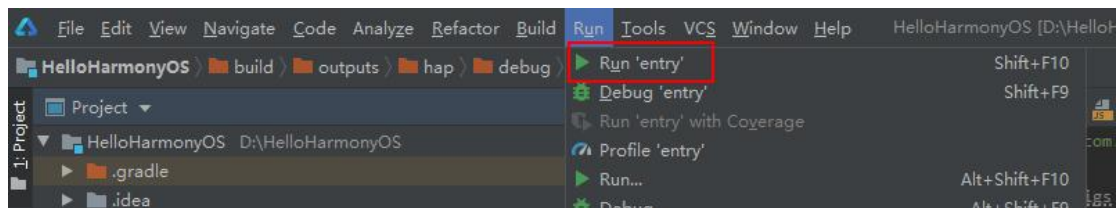


待模拟器启动成功后，出现下图所示的界面。

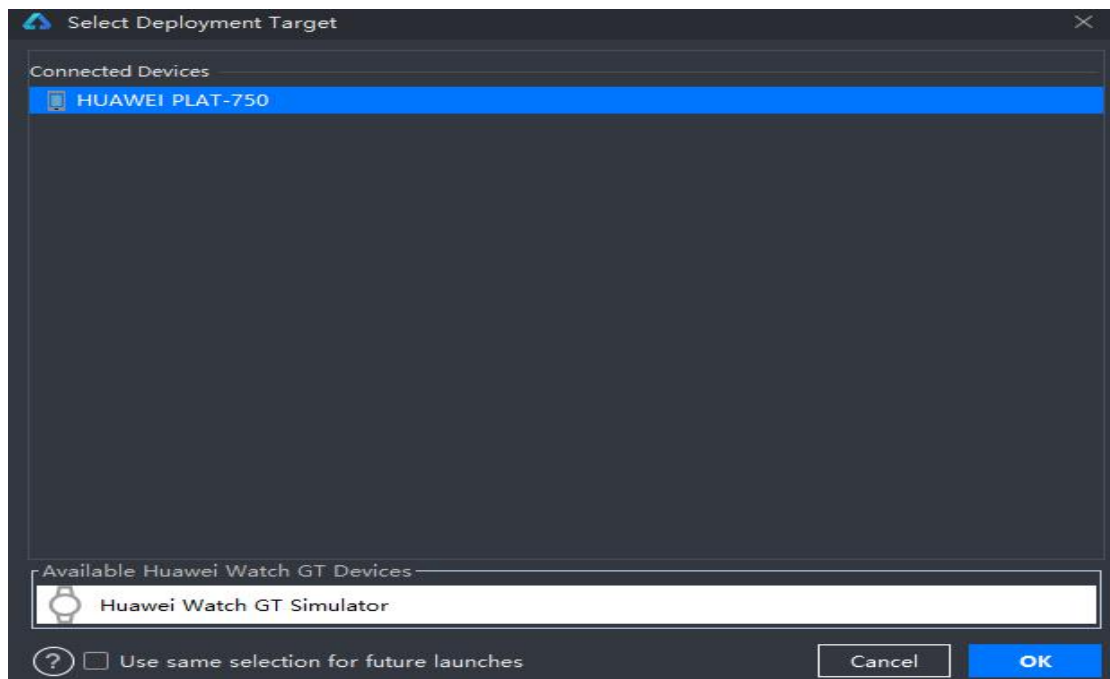


2. 部署应用

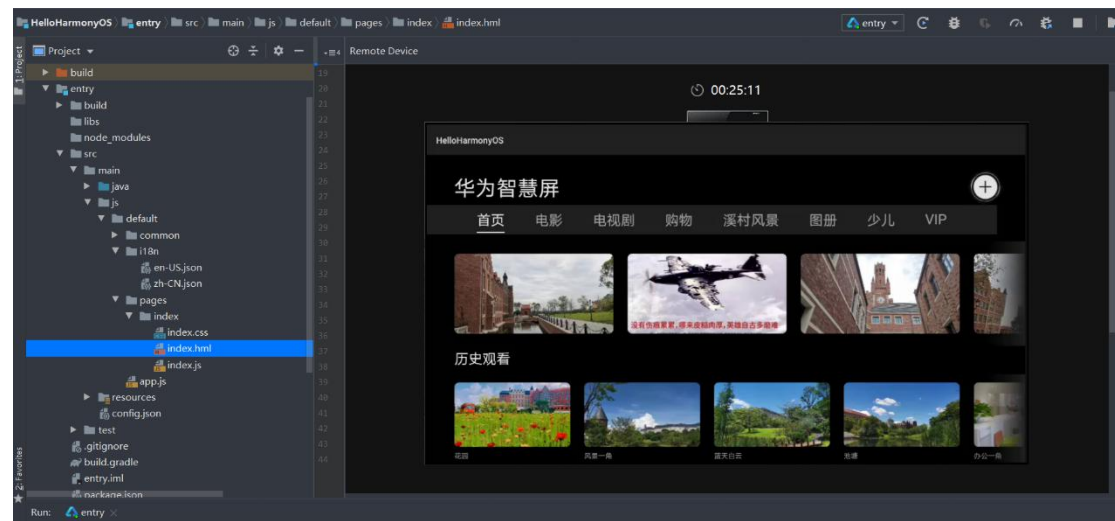
点击 **Run > Run 'Entry'**，部署应用。



选择模拟器设备。



应用程序运行如下



至此，您已经成功开发出第一个 **HarmonyOS** 应用，欢迎进入 **HarmonyOS** 世界！

5. 恭喜您

您已经成功完成了 **HelloWorld** 应用开发 **E2E** 体验，并学到了：

- 如何创建一个 **HarmonyOS** Project
- 编译构建 **hap** 包
- 将 **hap** 包部署到智慧屏远程模拟器上并运行

九、设备虚拟化特性开发

1. 介绍

您将建立什么

在这个 **Codelab** 中,您将能够使用 **HUAWEI DevEco Device Tool** 完成摄像头设备的开发,达到以下效果:在使用华为手机畅连通话的过程中,可以将开发板摄像头虚拟化为本端手机摄像头,在对端手机上展示摄像头设备拍摄的画面。

您将学到什么

1. 如何通过 HPM 软件包管理器获取基于 HarmonyOS 的具有设备虚拟化能力的摄像头解决方案。
2. 使用 HUAWEI DevEco Device Tool 开发工具完成代码开发、代码安全检查、编译、调试、烧录全流程。

2. 您需要什么

硬件要求

- HI3518EV300+HI3881WiFi 模组的开发板、配套摄像头
- 华为手机 (EMUI 10.1)
- PC 电脑

软件要求

- VS Code
- HUAWEI DevEco Device Tool 插件
- Linux 虚拟机

需要的知识点

- 硬件基础开发能力

- C 语言基础能力

提示：本次代码编译构建环境均为 Linux。所有软硬件环境, 需要使用的用户名和密码已经在 CodeLab 现场为大家准备好了。

3. 操作步骤

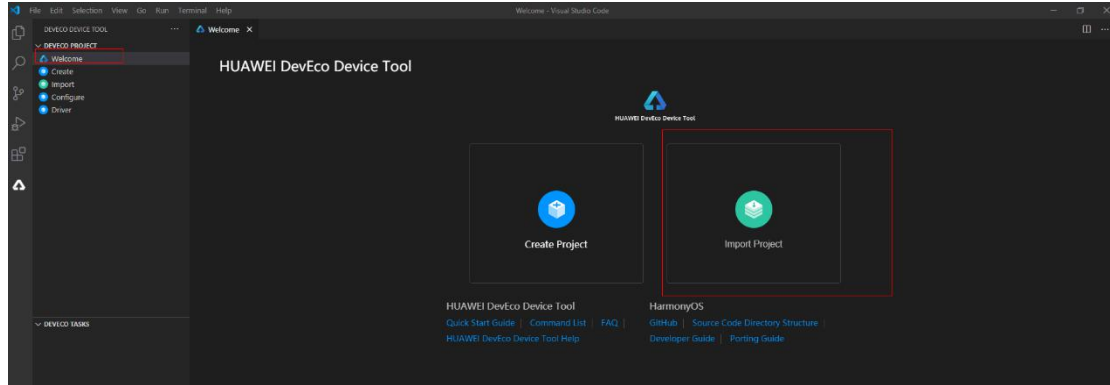
1. 访问站点 <https://hpm.harmonyos.com/#/home>，如下图，选择摄像头类产品



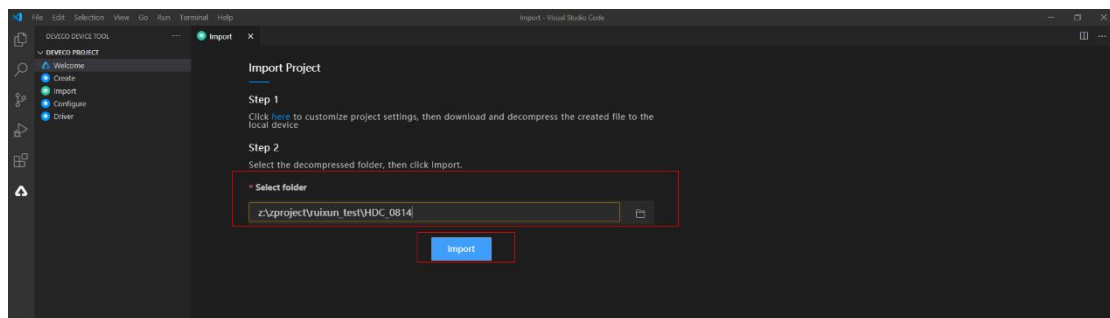
2. 进入页面后点击"直接下载"下载文件到 Linux 虚拟机映射的网络驱动器路径并解压。



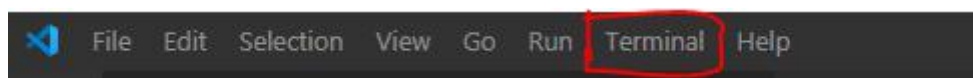
3. 打开 **vscode** 编辑器，在 **HUAWEI DevEco Device Tool** 插件 **welcome** 首页，选择 **Import Project**，如下图：



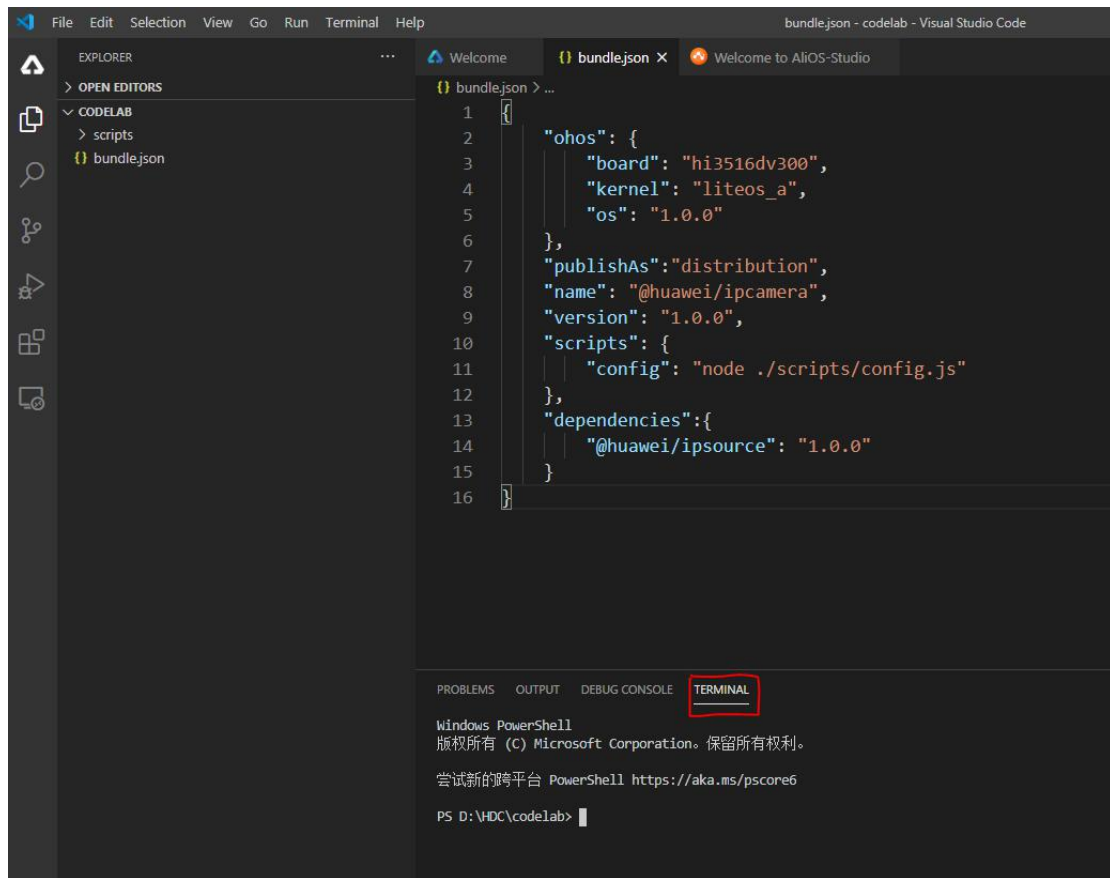
出现下面窗口，选择上一步解压文件所在路径，点击 **Import**：



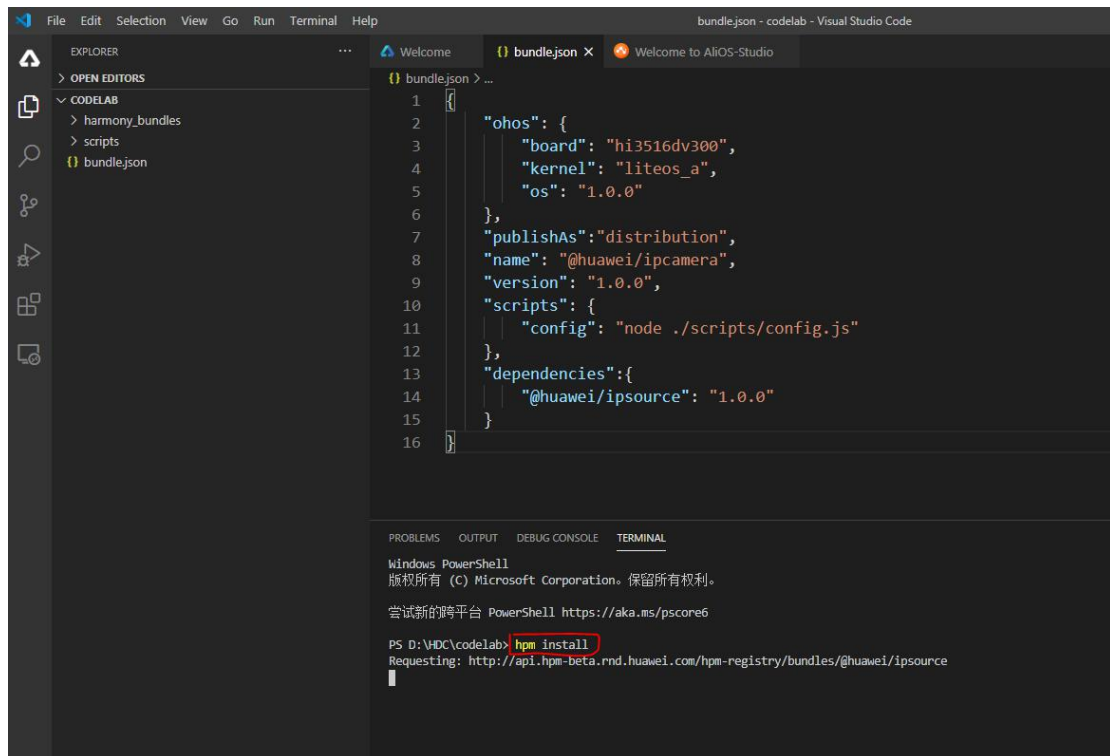
4. 打开 **TERMINAL** 窗口



如下图:



5. 在 **TERMINAL** 窗口中，输入命令 **hpm install**。如下图：



6. **hpm install** 命令行执行完毕，源码工程就准备就绪了。

7. 添加代码片段

打

开 `./vendor/hisi/camera/dvkit_product/sample/ohos3518ev300/dvkit/dvkit_camera/dmsdp_camera_sdk.c`:

将 **OpenCamera** 接口函数参照如下代码片段补充完整，即可以实现 **camera** 的虚拟化功能，代码如下：

```
int32_t OpenCamera(const DMSDPCameraParam *param)
{
    if (param == NULL) {
        return DMSDP_ERR_INVALID_PARAMETER;
    }

    if (!IsCameraIdValid(param->id, param->idLen)) {
        LOGD("OpenCamera camera id invalid");
        return DMSDP_ERR_INVALID_PARAMETER;
    }
}
```

```

}

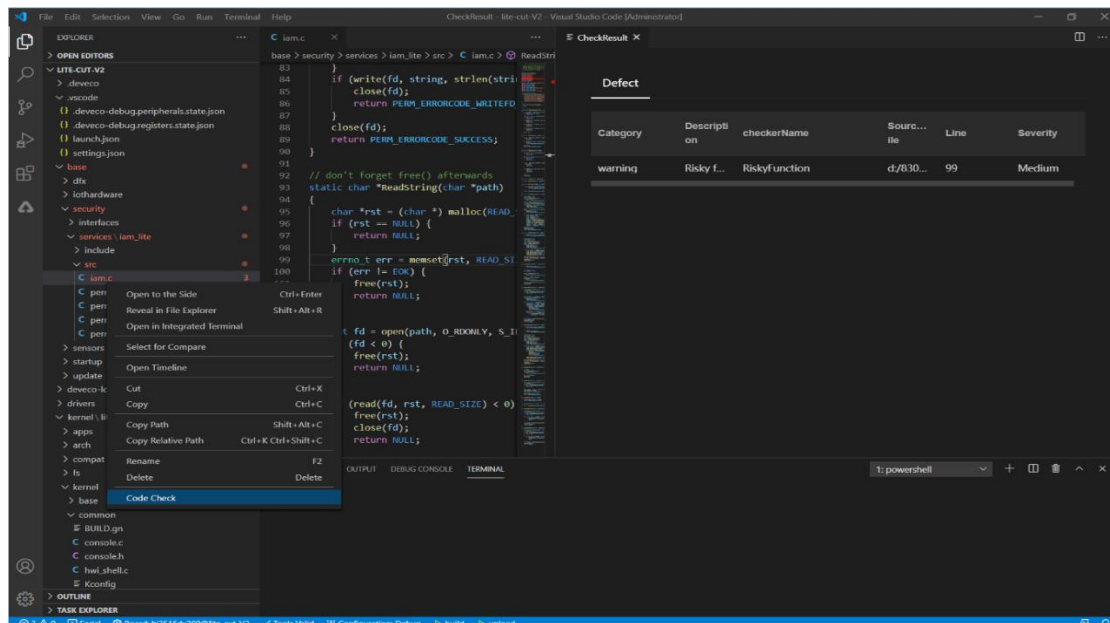
/* camera paramter convert */
LOGD("OpenCamera sdk
camera=%s,width=%d,height=%d,fps=%d,dataType=%d", param->id,
param->width, param->height,
    param->fps, param->dataFormat);

int32_t ret = VideoStartPIPE(param->dataFormat);
return ret;
}

```

8. 安全检查

选中需要安全检查的代码文件或者目录，点击右键，选择**"Code Check"**进行安全检查



检查完毕后，点击具体的错误项，跳转到对应文件的对应代码行。

9. HarmonyOS Demo 源码编译

通过 **Terminal > New Terminal** 打开终端窗口，其中

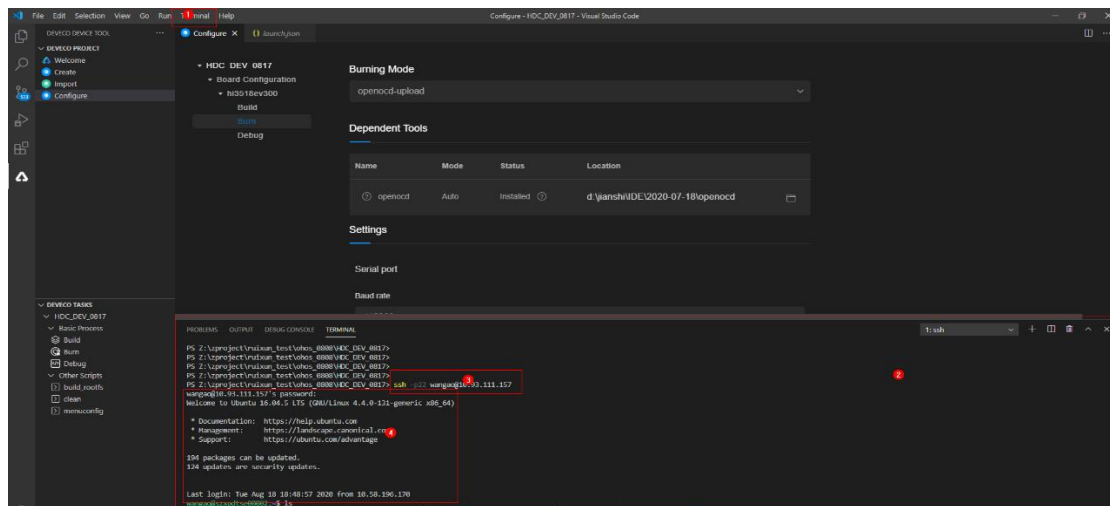
方框 1 表示新建终端窗口按钮；

方框 2 表示打开终端窗口后的界面；

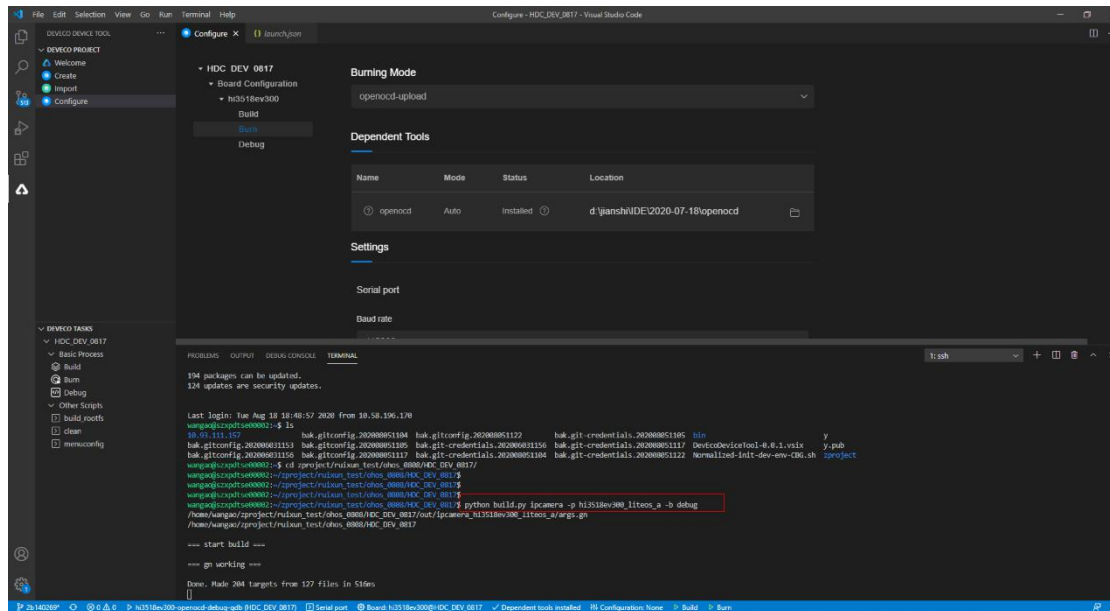
方框 3 在终端窗口中通过 **SSH** 连接 **Linux** 机器，并输入 **Linux** 机器的账号和 **IP** 地址：

ssh-p22 account@IP;

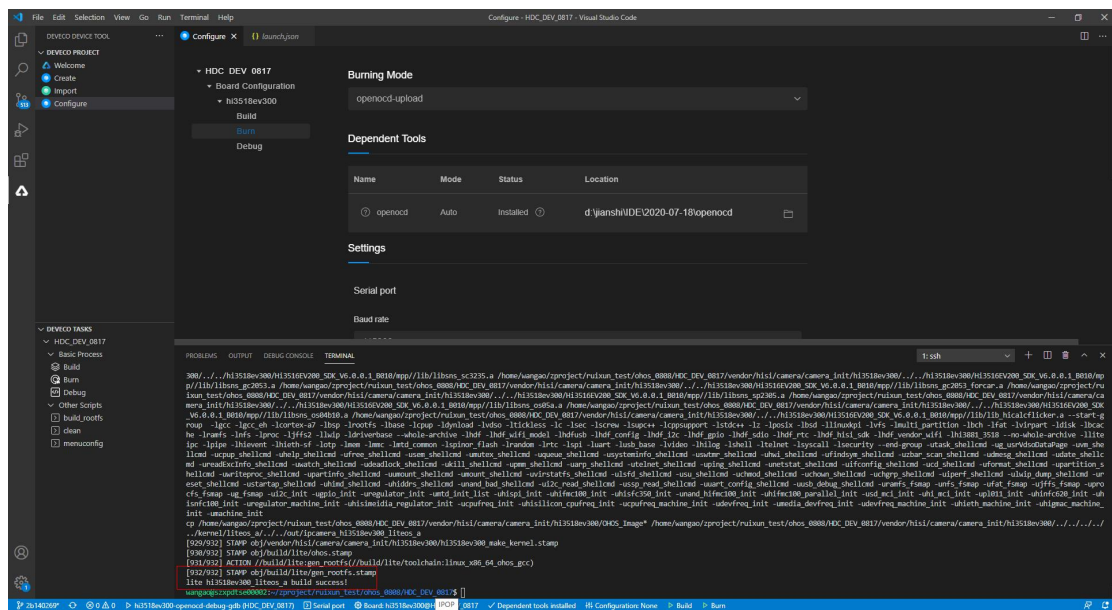
方框 4 输入 **Linux** 机器的密码，成功连上 **Linux** 机器；



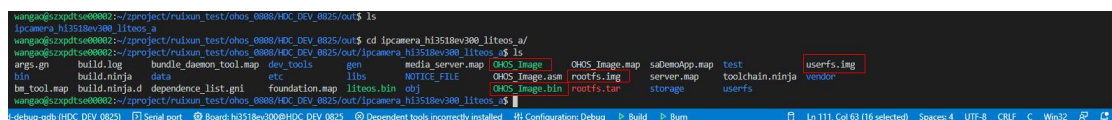
在终端窗口执行 `cd` 进入源码路径，输入编译脚本命令：**`python build.py ipcamera -p hi3518ev300_liteos_a -b debug`**，见下图：



成功编译完成如下图：



编译结果位于 out 目录。待烧录文件：OHOS_Image.bin、rootfs.img、userfs.img；调试可执行文件：OHOS_Image，图示如下：

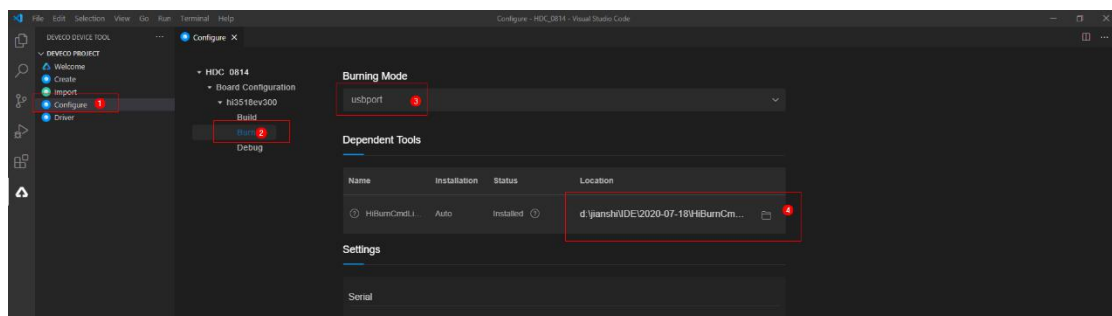


如果 out 目录下不能找到对应的文件，则从./vendor/hisi/i35xx/hi35xx_init/hi3518ev300 目录中查找 OHOS_Image.bin、OHOS_Image。

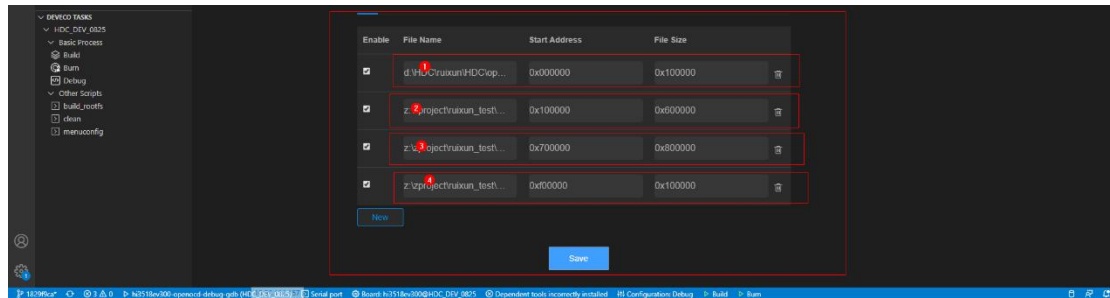
10. HarmonyOS 镜像烧录

目前烧录方法主要是 USB 烧录，具体步骤如下：

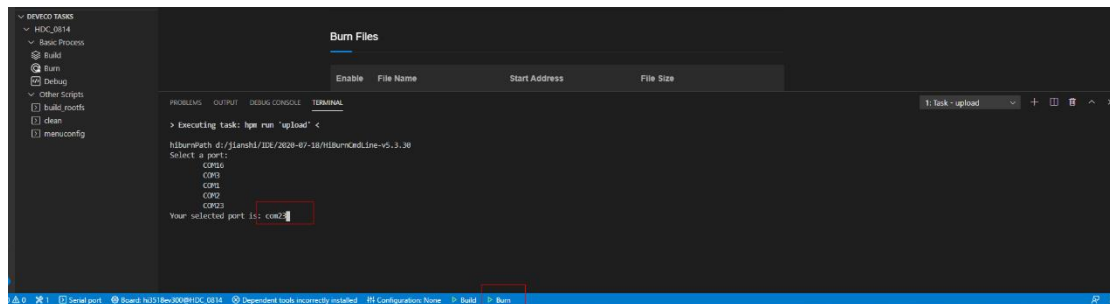
- 1) 打开插件，找到方框 1 的按钮 **Configure**，左键点击；
- 2) 弹出的窗口，找到方框 2 的按钮 **Burn**，左键点击；
- 3) 右侧窗口找 **Burning Mode** 选择 **usbport**；
- 4) 下方的 **Dependent Tools** 的依赖文件，需要选择依赖文件的存放目录 **HiBurnCmdLine-v5.3.30**，依赖文件单独提供；



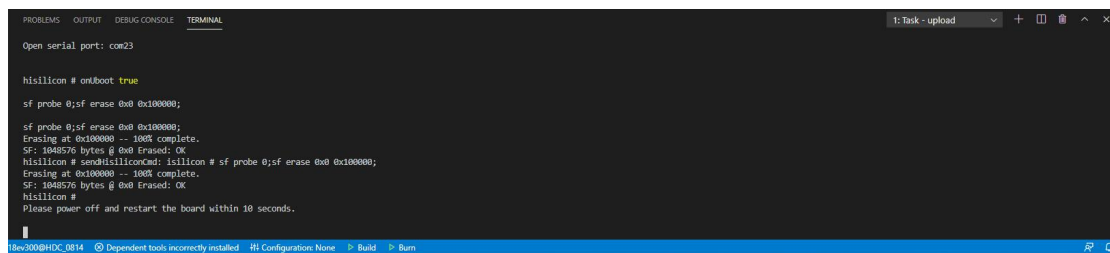
5) 在窗口下方的 **Burn Files** 中选择待烧录文件的路径，分别是 **u-boot-hi3518ev300.bin**(./vendor/hisi/camera/hi3518ev300/Hi3516EV200_SDK_V6.0.0.1_B010/osdrv/pub/u-boot-hi3518ev300.bin)、**OHOS_Image.bin**、**rootfs.img**、**userfs.img**，烧录的起始地址如下图所示，最后点击保存按钮，USB 烧录的配置完成。如下图：其中标记 1 对应 **u-boot-hi3518ev300.bin**，标记 2 对应 **OHOS_Image.bin**，标记 3 对应 **rootfs.img**，标记 4 对应 **userfs.img**：



6) 点击 **Burn** 按钮，输入开发板对应的串口号：

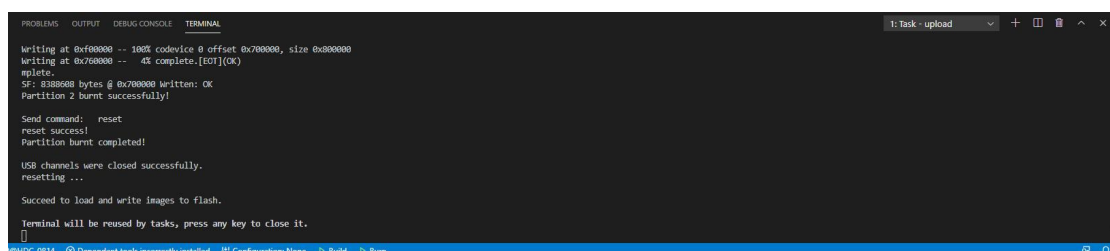


烧录过程中出现**"Please power off and restart the board within 15 seconds"**如下图所示，则复位单板：

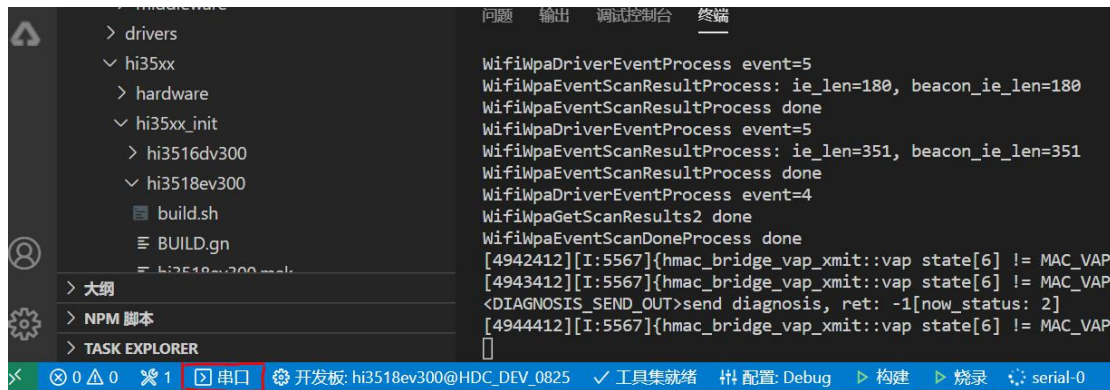


显示擦除完 **uboot** 后如果还有上图提示，则再次复位单板。

7) 烧录完成：



烧录完成后，关闭当前终端窗口。点击下图所示"串口"按钮，打开串口终端，



输入如下命令：

```
setenv bootargs 'console=ttyAMA0,115200n8 root=flash fstype=jffs2 rw
rootaddr=7M rootsize=8M'
setenv bootcmd 'sf probe 0;sf read 0x40000000 0x100000 0x600000;mw
0x112c0048 1a04 1;mw 0x112c004c 1004 1;mw 0x112c0064 1004 1;mw
0x112c0060 1004 1;mw 0x112c005c 1004 1;mw 0x112c0058 1004 1;mw
0x120C0020 0430 1;go 0x40000000'
saveenv
reset
```

最后出现下面的结果，则表示系统已经正常启动：



11. HarmonyOS 源码单步调试

目前调试支持 GDB 调试和 LLDB 调试。如下以 GDB 调试为例来介绍单步调试：

通过 **Configure > Debug** 打开调试配置页，将方框 4 中调试类型选为 **"openocd-debug-gdb"**；

方框 5 GDB 调试过程中需要依赖的软件名称、安装状态、配置路径；

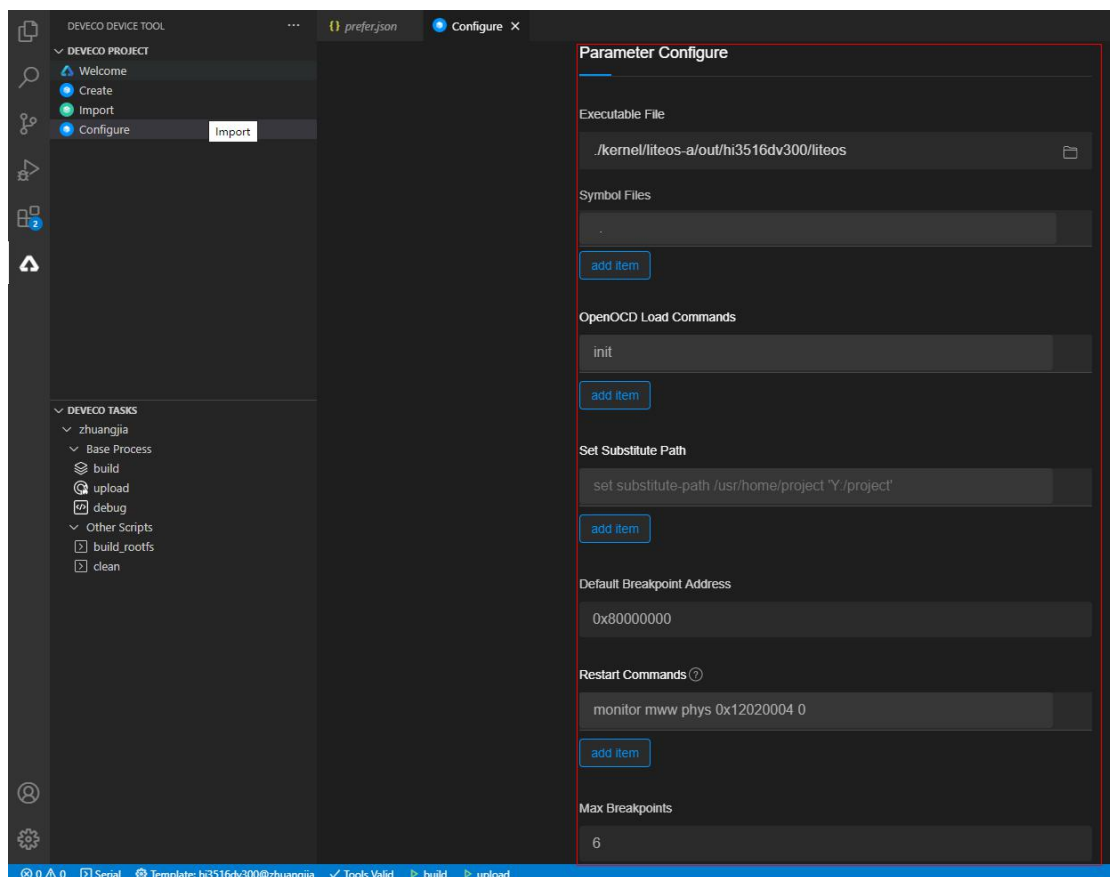
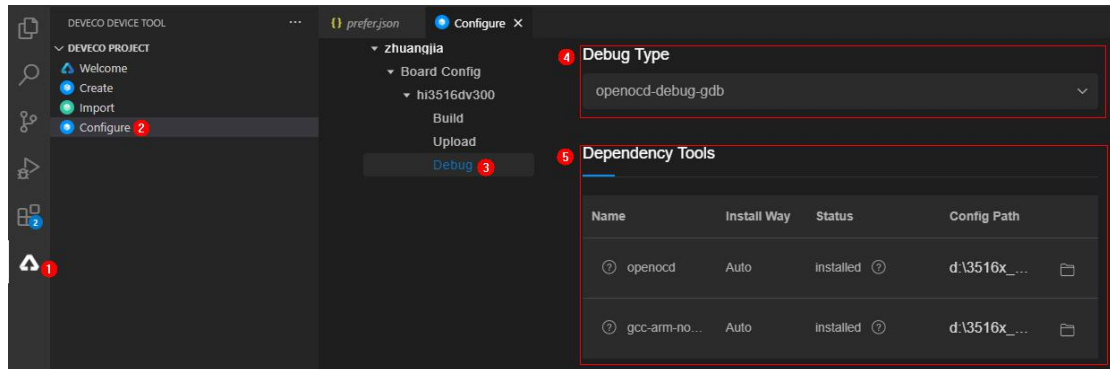
方框 6 调试过程中经常涉及的一些参数设置，包括：

- 1) Executable File: 选择内核文件 OHOS_Image ；
- 2) OpenOCD Load Commands: OpenOCD 加载命令，保持默认值"init"即可；
- 3) Set Substitute Path: 当前源码在 Linux 环境下进行编译，借助 IDE 进行烧录、调试时，设置映射路径，把 Linux 路径映射到 Windows，格式参考提示 "set substitute-path {Linux 路径} {映射的后的 Windows 路径}"；

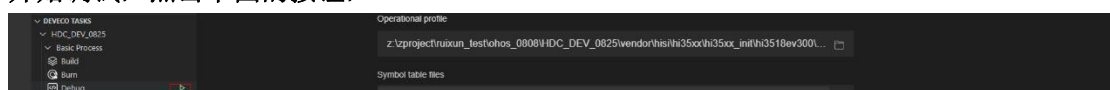
4) **Default Breakpoint Address:** 默认的断点位置，启动调试功能后，程序会自动在该位置停止；

5) **Restart Commands:** 重启调试任务时，工具发给单板的命令，保持默认值即可；

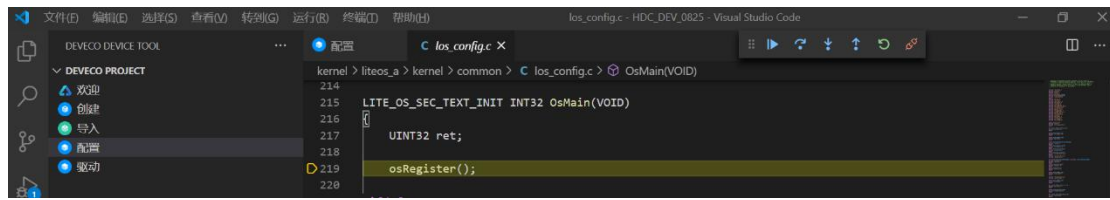
6) **Max Breakpoints:** 设置断点的最大数量(主要是硬件断点，不同的硬件可能存在差异)，当用户在调试过程中添加的断点数过多，可能导致调试功能崩溃；



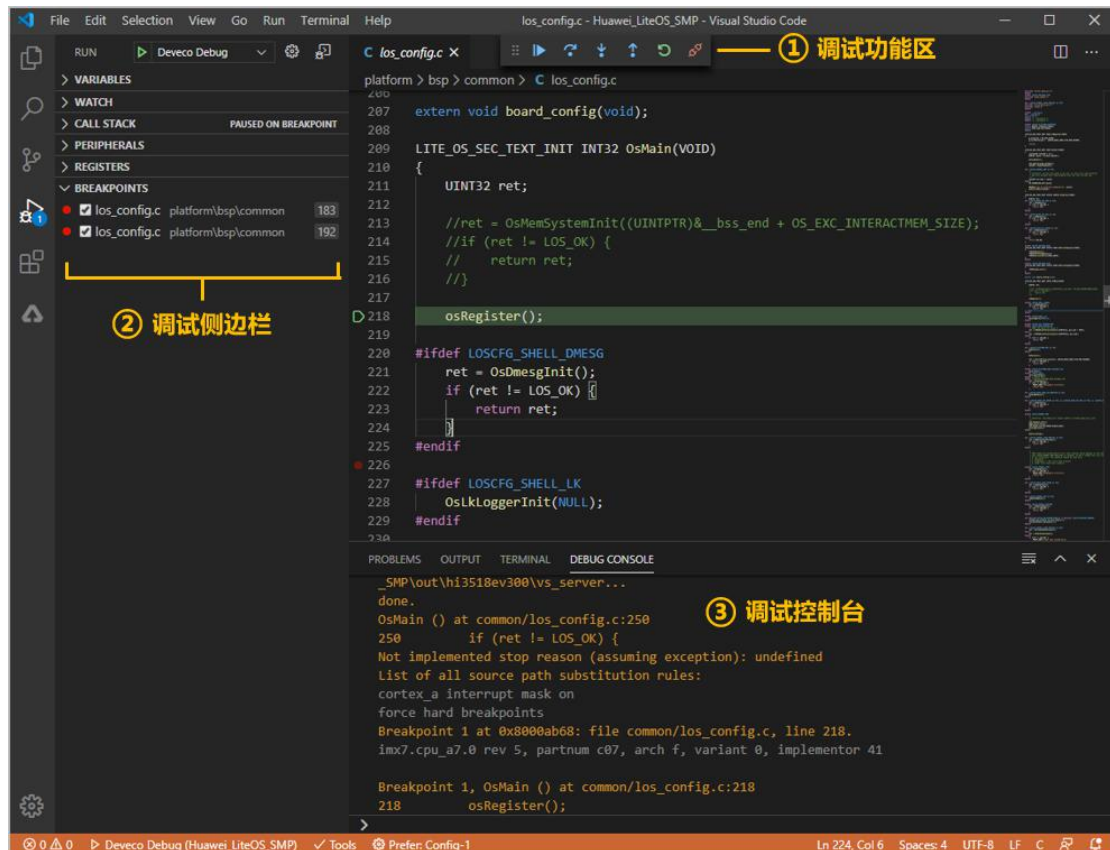
开始调试，点击下图的按钮，






正式进入调试阶段：






调试工作界面如下图所示，主要分为调试功能区、调试侧边栏和调试控制台。

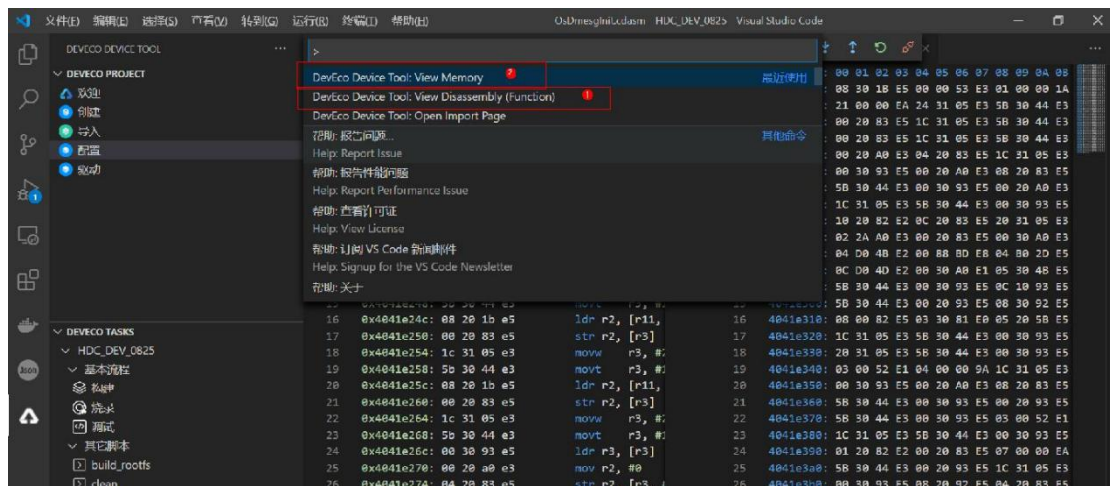


启动调试功能后，当代码执行到设置的断点时，程序会暂停，你可以根据调试功能区的按钮进行代码的调试。

- : Continue/Pause ("F5")，当程序执行到断点时暂停，点击此按钮程序继续执行。
- : Step Over ("F10")，在单步调试时，直接前进到下一行（如果在函数中存在子函数时，不会进入子函数内单步执行，而是将整个子函数当作一步执行）。
- : Step Into ("F11")，在单步调试时，遇到子函数后，进入子函数并继续单步执行。

- : Step Out ("Shift+F11"), 在单步调试执行到子函数内时, 点击 Step Out 会执行完子函数剩余部分, 并跳出返回到上一层函数。
- : Restart ("Ctrl+Shift+F5"), 重新启动调试。
- : Stop ("Shift+F5"), 停止调试任务。

调试过程中可以使用 IDE 查看函数的反汇编和内存的内容, 使用快捷键 **ctrl+shift+p**, 在弹出的窗口中输入 **DevEco Device Tool:View Disassembly(Function)**, 然后再输入函数名字, 即可查看当前函数的对应的汇编。输入 **DevEco Device Tool:View Memory**, 输入内存地址和要显示的内存长度, 即可显示内存内容:



12. 摄像头虚拟化特性验证

1) 请使用华为手机 A, 打开智慧生活 APP, 通过添加设备, 把刚开发完成的智能设备添到智慧生活 APP 中。

详细步骤如下:

a) 设备启动后，会默认启动一个 **AP** 等待手机连接，此时打开手机上的智慧生活 **APP**，下图为智慧生活 **APP** 界面：



b) 开始添加设备，点击右上角"+", "添加设备", 进行设备搜索，添加设备界面如下：



c) 搜索设备，搜索设备界面如下：



d) 搜索到的所要连接的设备后点击"连接", 此时需要输入当前 WiFi 信息, 输入 WIFI SSID 和密码后, 点击"下一步". 出现下图点击手动输入 12345678 后, 然后点击"确定"后开始连接:



e) 连接后出现下面的界面，选择房间号，点击"完成"，即完成了设备添加：



2) 用 B 手机的畅连通话 APP 拨打 A 手机。接通后，在 A 手机的屏幕左上角点击更多设备，可以发现开发完的智能设备，点击该设备，选择把摄像头虚拟化成 A 手机的外设，B 手机视频通话能看到的界面切换成了智能设备拍摄到的画面。

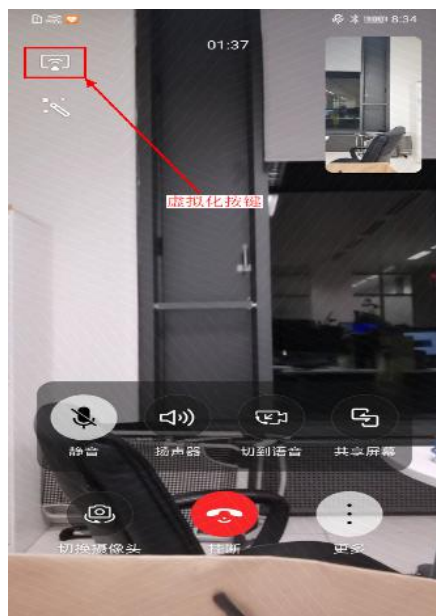
具体步骤如下：

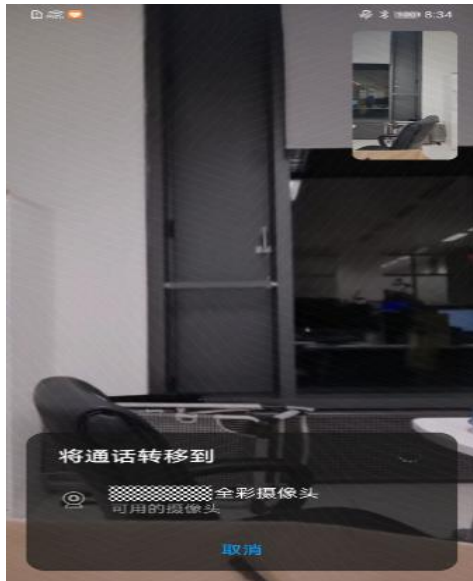
a) 打开手机通话(EMUI10.0 之后的版本)切换至畅连通话页或打开畅连通话 APP 如下图，

输入被呼叫手机号，拨通畅连通话：

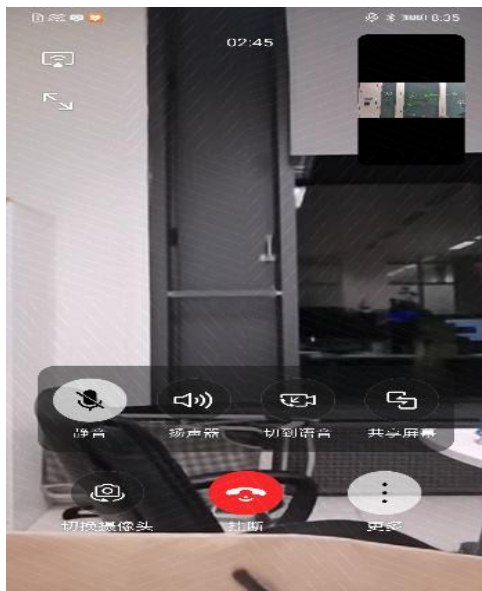


b) 点击测试手机左上角虚拟化按键，进行搜索，如图搜索到"XX 全景摄像头"：





c) 点击搜索到的设备，虚拟化摄像头成功：



4. 恭喜您

您已经成功完成了 **Codelab** 并学到了：

1. 如何通过 HPM 软件包管理器获取基于 HarmonyOS 的具有设备虚拟化能力的摄像头解决方案。
2. 使用 HUAWEI DevEco Device Tool 开发环境完成代码开发、代码安全检查、编译、调试、烧录全流程。

注：文档和视频中的所有图片及代码截图皆为示意图，具体以 HarmonyOS 官网发布内容为准。