

# HarmonyOS 设备开发 实验手册



华为技术有限公司

版权所有 © 华为技术有限公司 2021。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

## 商标声明



HUAWEI 和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

## 注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼

邮编： 518129

网址： <http://e.huawei.com>

# 目录

<b>目录</b>	<b>3</b>
<b>1 实验介绍</b>	<b>4</b>
1.1 实验说明	4
1.1.1 关于本实验	4
1.1.2 实验目的	4
1.1.3 实验规划	4
1.2 开发环境简介	4
1.3 实验设备介绍	5
1.5.1 Windows 工作平台	5
1.5.2 Linux 编译服务器	5
1.5.3 HarmonyOS 设备	5
<b>2 Wifi 模组设备的 HarmonyOS 系统开发</b>	<b>8</b>
2.1 任务说明	8
2.2 HarmonyOS 系统代码获取	8
2.3 开发一个指示灯点亮应用	9
2.4 系统编译和镜像生成	10
2.5 镜像烧写	11
<b>3 IPCamera 设备的 HarmonyOS 系统开发</b>	<b>17</b>
3.1 任务说明	17
3.2 HarmonyOS 系统代码获取	17
3.3 开发一个“helloworld”系统应用	17
3.4 系统编译与镜像生成	20
3.5 镜像烧写	21

# 1 实验介绍

## 1.1 实验说明

### 1.1.1 关于本实验

本实验通过两个设备的系统开发场景实践演练，帮助读者理解掌握 HarmonyOS 设备开发的开发过程。

### 1.1.2 实验目的

- 掌握 HarmonyOS 设备开发基础知识
- 掌握 HarmonyOS 在 Wifi 模组设备上的系统开发
- 掌握 HarmonyOS 在 IPCamera 设备上的系统开发

### 1.1.3 实验规划

1. 开发环境配置（参考开班前，HarmonyOS 设备开发环境搭建指导手册）。
2. 演练场景一：Wifi 模组设备的 HarmonyOS 系统开发。

核心要点：开发环境配置，代码获取，组件配置，系统编译，编译运行，系统烧写等。

3. 演练场景二：IPCamera 设备的 HarmonyOS 系统开发

核心要点：组件配置，三方应用

## 1.2 开发环境简介

设备开发的开发环境，如下图所示：

- Linux 编译服务器: HarmonyOS 代码的下载以及编译，镜像打包

Windows 工作平台: HarmonyOS 镜像的开发、调试、烧写



图 1-1 设备开发环境总体图

## 1.3 实验设备介绍

### 1.5.1 Windows 工作平台

windows 10 64 位系统

### 1.5.2 Linux 编译服务器

服务器的实现有两种可选方式（在环境搭建指导手册中使用了第一种方式）：

1. Linux 虚拟机，可安装在 Windows 系统上运行，与 Windows 工作平台使用同一台 PC
2. 真实的 Linux 服务器

### 1.5.3 HarmonyOS 设备

搭载 HarmonyOS 的设备。

- 场景一：本实验中为 Hi3861 芯片的开发板。

Hi3861 WLAN 模组是一片大约 2cm\*5cm 大小的开发板，是一款高度集成的 2.4GHz WLAN SoC 芯片，集成 IEEE 802.11b/g/n 基带和 RF（Radio Frequency）电路。支持 HarmonyOS，并配套提供开放、易用的开发和调试运行环境。

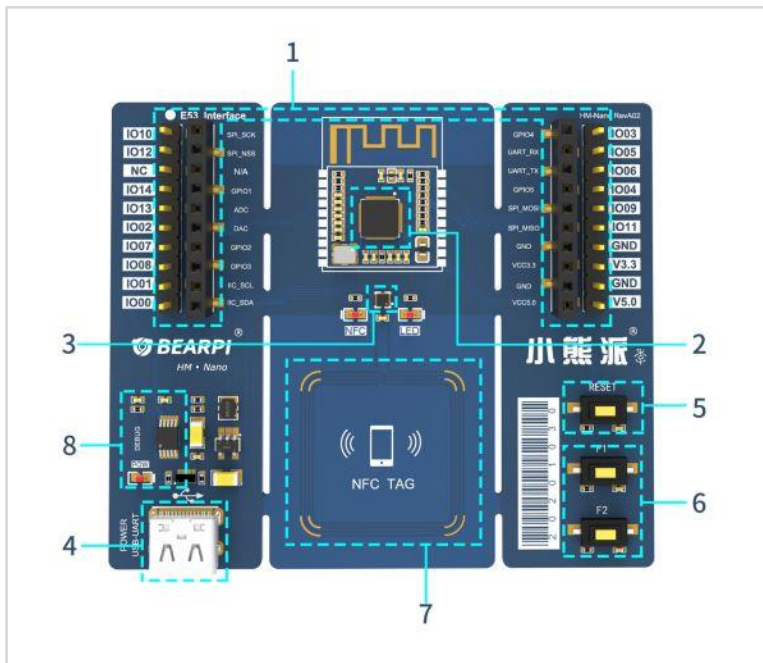


图1-2 Wifi 模组开发板

1. E53 Interface; 2. Hi3861RNIV100 2.4G Wi-Fi Soc 芯片; 3. \*NT3H1x01W0FHKH NFC 标签; 4. USB Type-C 5V 电源接口; 5. Reset 复位按键; 6. KEY1、KEY2 用户按键; 7. NFC 射频天线; 8. CH340 串口转换电路。

- 场景二：本实验中为 Hi3516 芯片的开发板

Hi3516DV300 作为新一代行业专用 Smart HD IP 摄像机 SOC，集成新一代 ISP、H.265 视频压缩编码器，同时集成高性能 NNIE 引擎，使得 Hi3516DV300 在低码率、高画质、智能处理和分析、低功耗等方面引领行业水平。

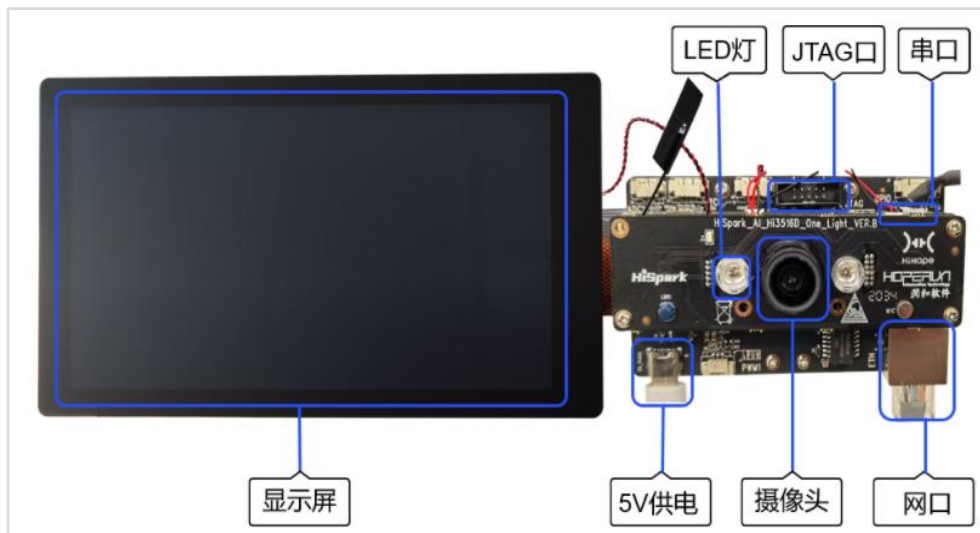


图1-3 IPCamera 开发板

Hi3516 开发板规格清单：

规格类型	规格清单
处理器及内部存储	Hi3516DV300芯片 DDR3 1GB eMMC4.5, 8GB容量
外部器件	以太网口 音频视频 1路语音输入 1路单声道(AC_L)输出，接3W功放(LM4871) MicroHDMI ( 1路HDMI 1.4 ) 摄像头 传感器IMX335 镜头M12，焦距4mm，光圈1.8

	<p>显示屏</p> <p>LCD连接器（2.35寸）</p> <p>LCD连接器（5.5寸）</p> <p>外部器件及接口</p> <p>SD卡接口</p> <p>JTAG/I2S 接口</p> <p>ADC接口</p> <p>舵机接口</p> <p>Grove连接器</p> <p>USB2.0(Type C)</p> <p>功能按键3个，2个用户自定义按键，1个升级按键</p> <p>LED指示灯，绿灯，红灯</p>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

# 2 Wifi 模组设备的 HarmonyOS 系统开发

## 2.1 任务说明

- 下载 Hi3861 对应开发板设备的代码，
- 在 Linux 服务器上编译
- 使用 DevECO Devicetool 建立 Hi3861 开发板的工程并烧写镜像
- 启动设备

## 2.2 HarmonyOS 系统代码获取

我们通过代码仓库方式获取 HarmonyOS 系统源码。

打开 windows 命令窗口，使用 docker 命令进入 Linux 虚拟机容器

```
docker run -it -v d:\harmonyos:/home/openharmony swr.cn-south-1.myhuaweicloud.com/openharmony-docker/openharmony-docker:0.0.3
```

在容器命令行输入命令：

```
cd /home/openharmony;mkdir demo1;cd demo1
hpm init -t dist
hpm i @ohos/hispark_pegasus
```

执行命令后，进入代码下载和解压阶段，可能会耗费几分钟时间，出现如下界面表示代码下载完成：



```

% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
   100    925k   100    925k    0     0  1796k      0  --:--:--  --:--:--  --:--:--  1793k
gn
gn
gn : install success
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
   100   3430k   100   3430k    0     0  2624k      0  0:00:01  0:00:01  --:--:--  2624k
ninja/
ninja/ninja
ninja/COPYING
ninja/
ninja/ninja
ninja/COPYING
ninja : install success
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
   100   110k   100   110k    0     0   478k      0  --:--:--  --:--:--  --:--:--  480k
hc-gen/
hc-gen/hc-gen
hc-gen/LICENSE
hc-gen/
hc-gen/hc-gen
hc-gen/LICENSE
hc-gen : install success
Installed.
root@7b33b871b0c1:/home/openharmony#

```

代码下载完成后，目录如图：

```

root@a3199725df2d:/home/openharmony/demol# ll
total 20
drwxr-xr-x 1 root root 512 May 27 06:58 ./
drwxrwxrwx 1 root root 512 May 27 06:34 ../
drwxr-xr-x 1 root root 512 May 27 06:37 applications/
drwxr-xr-x 1 root root 512 May 27 06:36 base/
drwxr-xr-x 1 root root 512 May 27 06:35 build/
-rw-r--r-- 1 root root 202 May 27 08:07 bundle.json
-rw-r--r-- 1 root root 19792 May 27 06:35 bundle-lock.json
drwxr-xr-x 1 root root 512 May 27 06:35 device/
drwxr-xr-x 1 root root 512 May 27 06:36 domains/
drwxr-xr-x 1 root root 512 May 27 06:37 foundation/
drwxr-xr-x 1 root root 512 May 27 06:36 kernel/
-rw-r--r-- 1 root root 29 Sep 5 2020 LICENSE
drwxr-xr-x 1 root root 512 May 27 06:35 ohos_bundles/
-rw-r--r-- 1 root root 304 May 28 01:10 ohos_config.json
drwxr-xr-x 1 root root 512 May 27 06:58 out/
drwxr-xr-x 1 root root 512 May 27 06:37 prebuilts/
-rw-r--r-- 1 root root 150 Sep 5 2020 README.md
drwxr-xr-x 1 root root 512 May 27 06:37 test/
drwxr-xr-x 1 root root 512 May 27 06:37 third_party/
drwxr-xr-x 1 root root 512 May 27 06:37 utils/
drwxr-xr-x 1 root root 512 May 27 06:37 vendor/
root@a3199725df2d:/home/openharmony/demol#

```

## 2.3 开发一个指示灯点亮应用

当前发行版的代码仓中已存在 3861 Led 指示灯 demo，只需要将该 demo 编译运行即可，步骤如下：

### 步骤 1 安装 vim 文本编辑器

```
apt-get install vim -y
```

步骤 2 修改 applications/sample/wifi-iot/app/iothardware/led\_example.c 中 Led 指示灯 GPIO 管脚值为 2（本实验中使用的开发板 Led 对应的管脚实际值）

修改后：#define LED\_TEST\_GPIO 2

```
#define LED_INTERVAL TIME_US 300000
#define LED_TASK_STACK_SIZE 512
#define LED_TASK_PRIO 25
#define LED_TEST_GPIO 2 // for hispark_pegasus
enum LedState {
    LED_ON = 0,
    LED_OFF,
    LED_SPARK,
};
```

步骤 3 修改 applications/sample/wifi-iot/app/iothardware/BUILD.gn 文件，此处为代码为同步为最新的不匹配的问题

```
static_library("led_example") {
    sources = [
        "led_example.c"
    ]
    include_dirs = [
        "../utils/native/lite/include",
        "../kernel/liteos_m/components/cmsis/2.0",
        "../base/iot_hardware/peripheral/interfaces/kits",
    ]
}
```

步骤 4 修改 applications/sample/wifi-iot/app/BUILD.gn 文件，将 demo 链接进来

```
import("../build/lite/config/component/lite_component.gni")

lite_component("app") {
    features = [
        "startup",
        "iothardware:led_example",
    ]
}
```

步骤 5 重新编译（方法见 2.4），生成二进制文件，烧录（方法见 2.5）后运行即可看到 3861 设备指示灯闪烁效果。

## 2.4 系统编译和镜像生成

在容器命令行输入命令：

```
hb set
```

出现的界面中输入。（选择当前路径）并回车

出现的界面通过键盘方向键，选择 **wifi-iot\_hispark\_pegasus** 并回车，出现如下界面：

```
root@a3199725df2d:/home/openharmony/demo1# hb set
[OHOS INFO] Input code path:
OHOS Which product do you need? wifiiot_hispark_pegasus
root@a3199725df2d:/home/openharmony/demo1#
```

在容器命令行输入命令：

```
hb build -f
```

最后，镜像就会编译出来了，镜像所在在工程目录（demo1）下的  
out/hispark\_pegasus/wifiiot\_hispark\_pegasus：

```
root@a3199725df2d:/home/openharmony/demo1# ll out/hispark_pegasus/wifiiot_hispark_pegasus
total 31232
drwxr-xr-x 1 root root    512 May 28 01:27 ./
drwxr-xr-x 1 root root    512 May 28 01:26 ../
-rw-r--r-- 1 root root    281 May 28 01:26 args.gn
-rw-r--r-- 1 root root  47605 May 28 01:27 build.log
-rw-r--r-- 1 root root 25056 May 28 01:26 build.ninja
-rw-r--r-- 1 root root   5244 May 28 01:26 build.ninja.d
drwxr-xr-x 1 root root    512 May 28 01:26 gen/
-rw-r--r-- 1 root root  24576 May 28 01:27 Hi3861_boot_signed.B.bin
-rw-r--r-- 1 root root  24416 May 28 01:27 Hi3861_boot_signed.bin
-rw-r--r-- 1 root root  15360 May 28 01:27 Hi3861_loader_signed.bin
-rw-r--r-- 1 root root 1216360 May 28 01:27 Hi3861_wifiiot_app_allinone.bin
-rw-r--r-- 1 root root 22949079 May 28 01:27 Hi3861_wifiiot_app.asm
-rw-r--r-- 1 root root  1176208 May 28 01:27 Hi3861_wifiiot_app_burn.bin
-rw-r--r-- 1 root root   25376 May 28 01:27 Hi3861_wifiiot_app_flash_boot_ota.bin
-rw-r--r-- 1 root root 3590592 May 28 01:27 Hi3861_wifiiot_app.map
-rw-r--r-- 1 root root  581456 May 28 01:27 Hi3861_wifiiot_app_ota.bin
-rwxr-xr-x 1 root root 2205884 May 28 01:27 Hi3861_wifiiot_app.out*
-rw-r--r-- 1 root root     8 May 28 01:27 Hi3861_wifiiot_app_vercfg.bin
drwxr-xr-x 1 root root    512 May 28 01:26 libs/
-rw-r--r-- 1 root root    16 May 28 01:26 .ninja_deps
-rw-r--r-- 1 root root  36985 May 28 01:27 .ninja_log
drwxr-xr-x 1 root root    512 May 28 01:26 NOTICE_FILE/
drwx----- 1 root root    512 May 28 01:26 obj/
drwxr-xr-x 1 root root    512 May 28 01:26 suites/
-rw-r--r-- 1 root root  30940 May 28 01:26 toolchain.ninja
```

## 2.5 镜像烧写

### 步骤 1 镜像所在的位置

对应执行 docker run 时候映射的文件目录，从虚拟机内的\home\openharmony 映射到  
windows 工作台的 D:\harmonyos 下，所以生成的镜像对应的位置

D:\harmonyos\demo1\out\hispark\_pegasus\wifiiot\_hispark\_pegasus\Hi3861\_wifiiot\_app\_a  
llinone.bin

### 步骤 2 安装 usb 转串口驱动

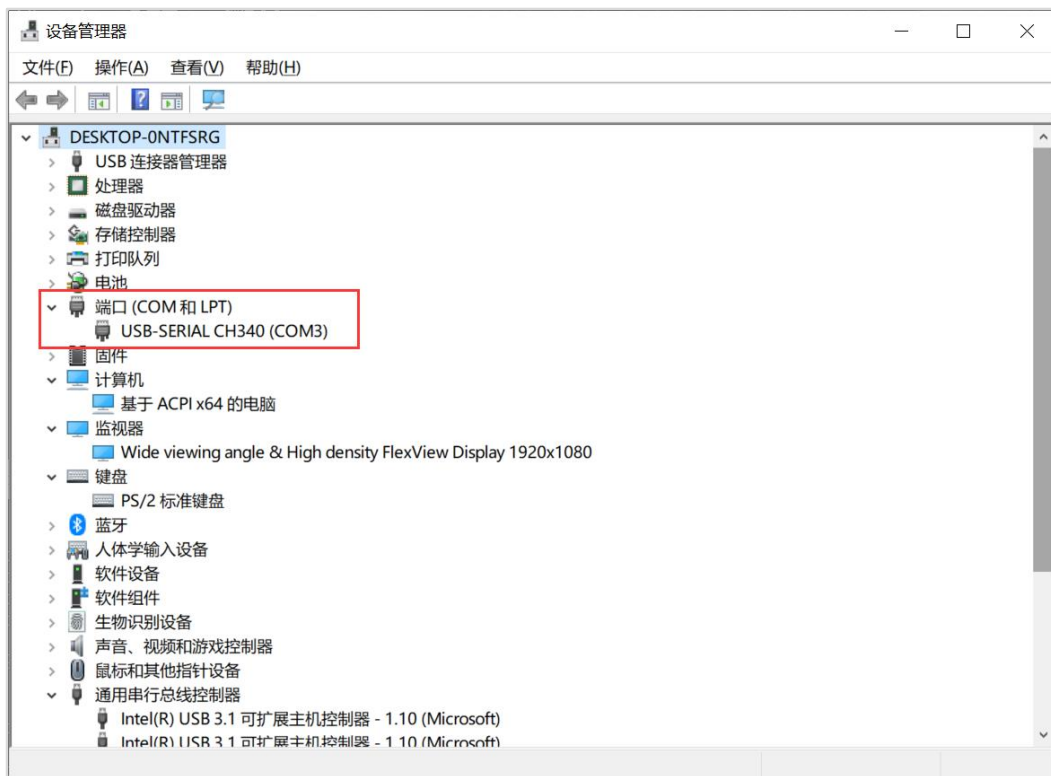
1) 点击链接下载 CH341SER USB 转串口驱动程序。

驱动下载链接：<http://www.wch.cn/downloads/file/65.html?time=2021-05-25%2020:45:01&code=GE04y9mMFqKAbQYgVyd9qVYb8qenr1hNlrGegU2s>

2) 使用 USB 线，将 3861 开发板接入 PC，并点击安装包，安装驱动程序。



3) 驱动安装完成后，重新插拔 USB 接口，串口信息显示如下图所示。



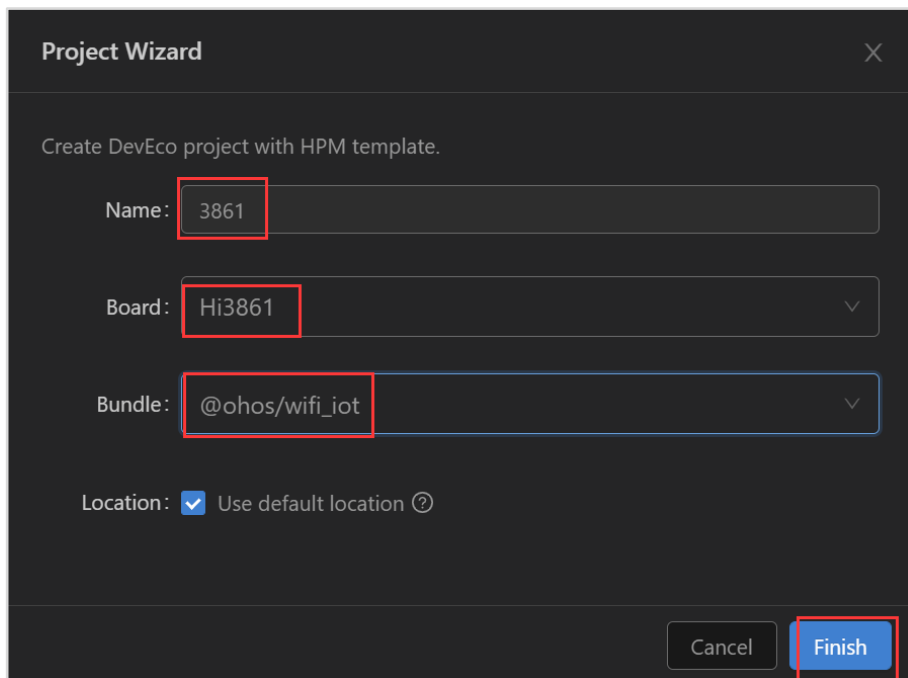
### 步骤 3 在 deveco 中新建工程

打开 visual code 默认会进入 DevEco Device Tool 首页：



点击新建 DevEco 工程，创建新工程。

跳出的配置项向导中，进行如下配置（工程名可自己配置）后创建：



创建后，进入工程配置页面。

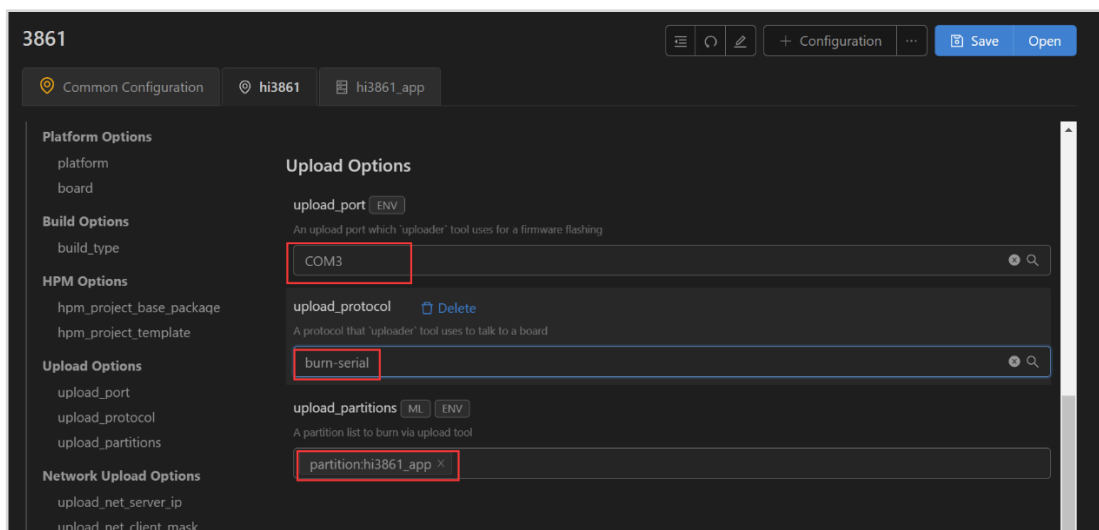
#### 步骤 4 设置烧录选项

在“hi3861”页签，设置烧录选项，包括 upload\_port、upload\_protocol 和 upload\_partitions。

upload\_port：选择步骤 2 中查询的串口号。

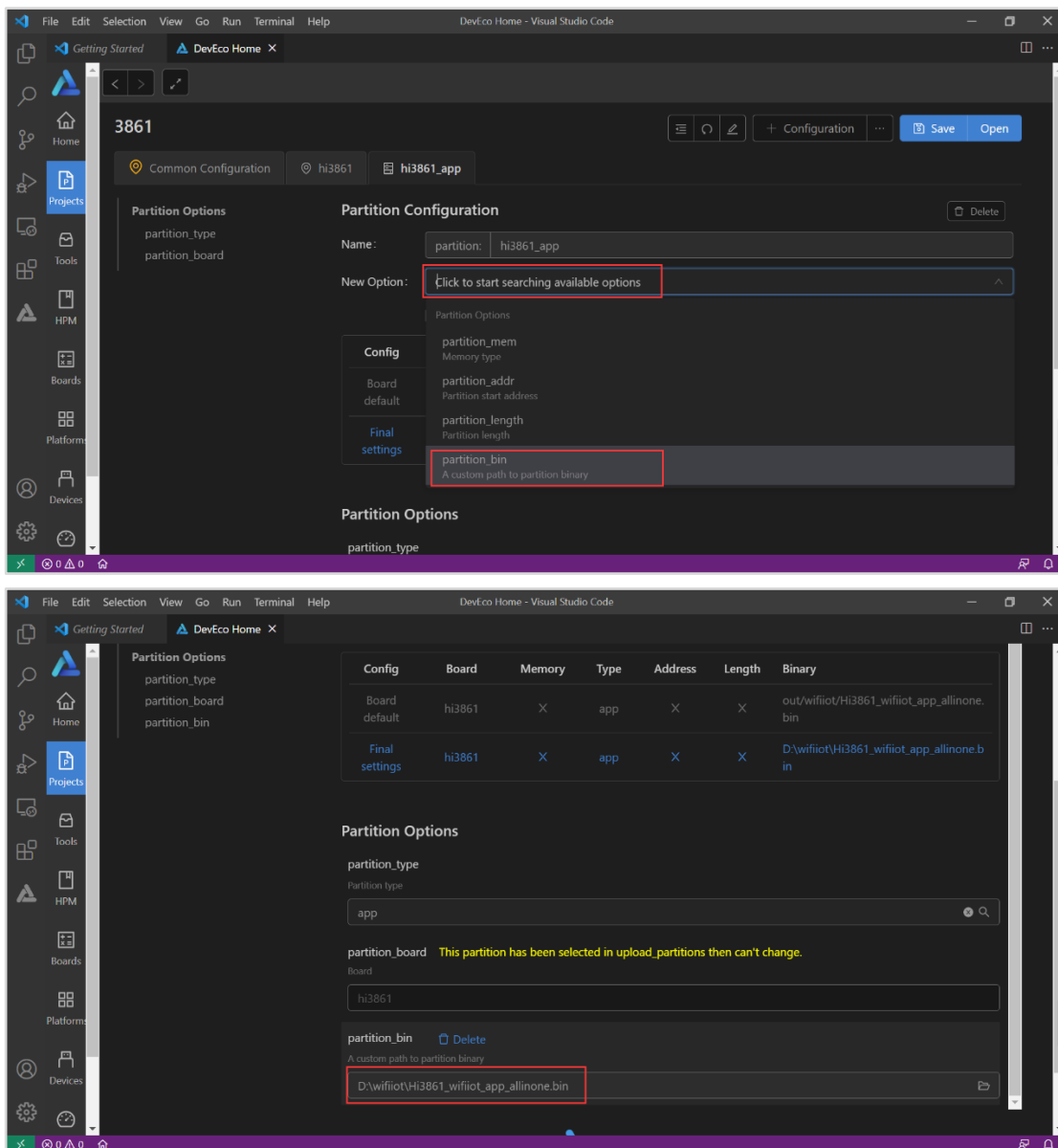
upload\_protocol：选择烧录协议，固定选择“hiburn-serial”。

upload\_partitions：选择待烧录的文件，固定选择“partition:hi3861\_app”。

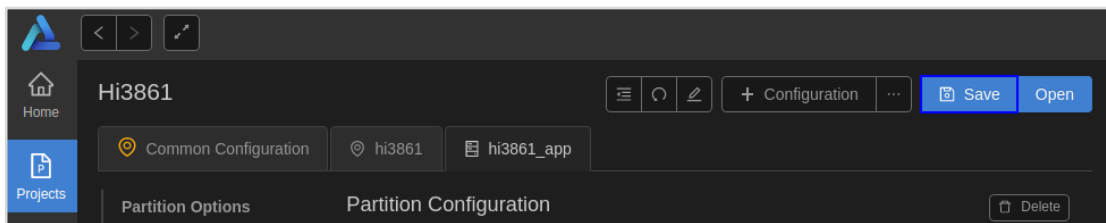


在“hi3861\_app”页签，设置烧录二进制文件。

点击“New Option”选择框，在页签中选择“partition\_bin”，并设置对应二进制文件。

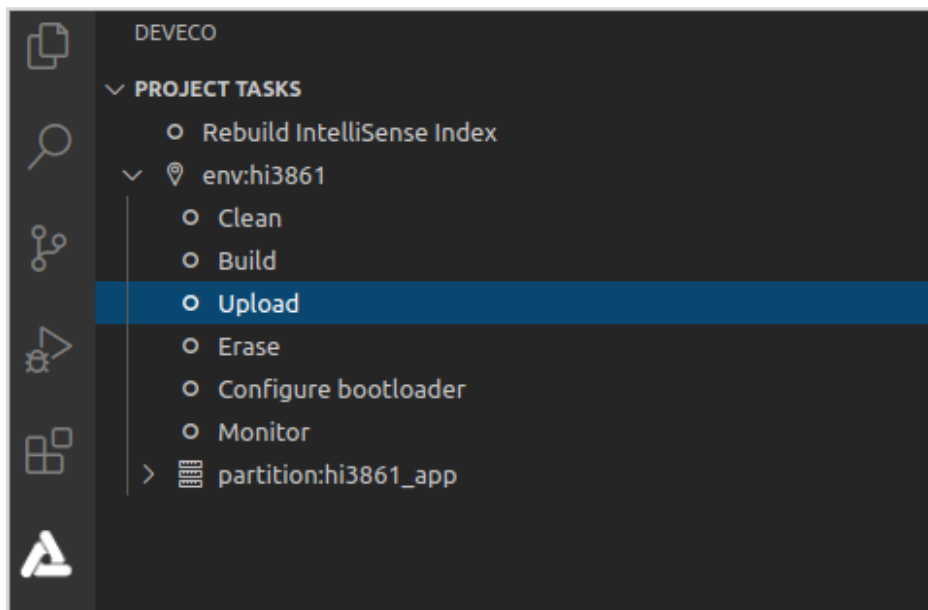


设置完成，点击“Save”按钮，然后打击“Open”按钮打开工程。

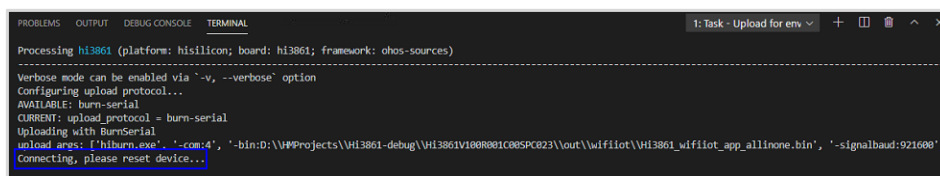


## 步骤 5 烧录

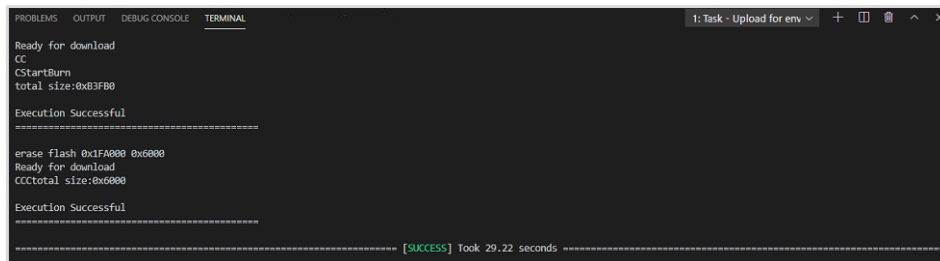
打开工程文件，在 DevEco Device Tool 界面的“PROJECT TASKS”中，点击 env:hi3861 下的 Upload 按钮，启动烧录



启动烧录后，显示如下提示信息时，请按开发板上的 RST 按钮重启开发板。

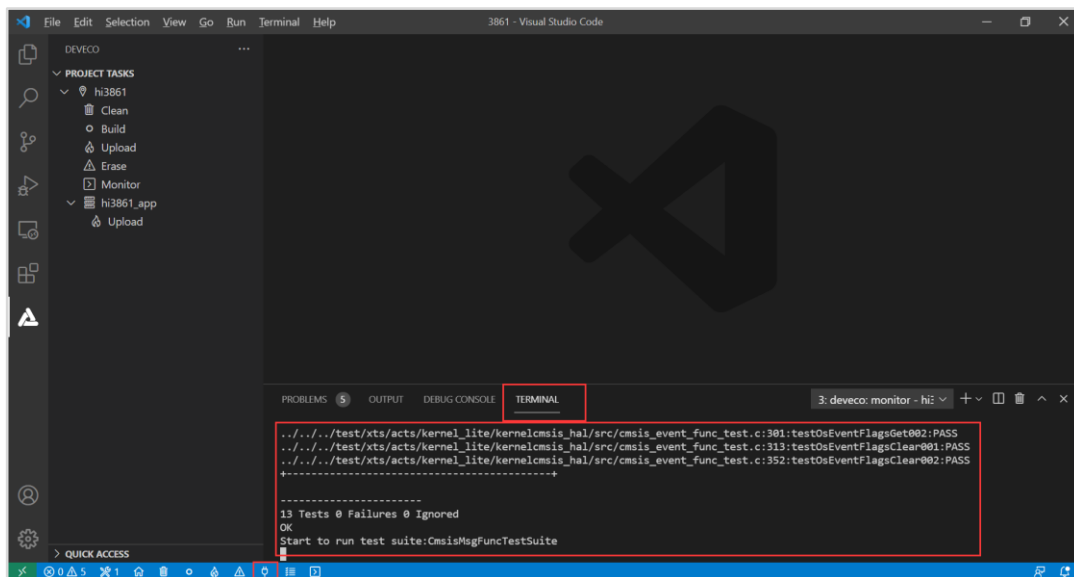


重新上电后，启动烧录，界面提示如下信息时，表示烧录成功。



## 步骤 6 单板启动

Visual Studio Cod 主界面，点击“Serial Monitor”工具按钮，即可连接 3861 设备串口。





# 3 IPCamera 设备的 HarmonyOS 系统开发

## 3.1 任务说明

- 下载 IPCamera 开发板设备代码，
- 在 Linux 服务器上编译
- 使用 DevECO Devicetool 建立 Hi3516 开发板的工程并烧写镜像
- 启动使用设备

## 3.2 HarmonyOS 系统代码获取

在第二章中我们已通过代码仓库方式获取 HarmonyOS 系统源码，目录如图：

```
root@299aa47592ce:/home/openharmony# ls -al
total 116
drwxr-xr-x 16 root root 4096 May 25 07:10 .
drwxr-xr-x  1 root root 4096 Apr 23 08:55 ..
drwxr-xr-x  3 root root 4096 May 25 07:06 applications
drwxr-xr-x  9 root root 4096 May 25 07:06 base
drwxr-xr-x  3 root root 4096 May 25 07:06 build
-rw-r--r--  1 root root  182 May 25 07:10 bundle.json
-rw-r--r--  1 root root 39280 May 25 07:05 bundle-lock.json
drwxr-xr-x  3 root root 4096 May 25 07:10 developeptools
drwxr-xr-x  3 root root 4096 May 25 07:06 device
drwxr-xr-x  6 root root 4096 May 25 07:06 drivers
drwxr-xr-x 10 root root 4096 May 25 07:06 foundation
drwxr-xr-x  3 root root 4096 May 25 07:06 kernel
-rw-r--r--  1 root root   29 Sep  5 2020 LICENSE
drwxr-xr-x  3 root root 4096 May 25 07:05 ohos_bundles
drwxr-xr-x  4 root root 4096 May 25 07:06 prebuilts
-rw-r--r--  1 root root  150 Sep  5 2020 README.md
drwxr-xr-x  5 root root 4096 May 25 07:06 test
drwxr-xr-x 26 root root 4096 May 25 07:10 third_party
drwxr-xr-x  3 root root 4096 May 25 07:06 utils
drwxr-xr-x  3 root root 4096 May 25 07:06 vendor
```

如未实践第二章 Wifi 模组设备的 HarmonyOS 系统开发，则可参照 2.2 步骤下载 HarmonyOS 系统源码。

## 3.3 开发一个“helloworld”系统应用

### 步骤 1 新建目录及源码

直接在 d:\harmonyos\demo2\applications\sample\camera 下创建 hello 目录，并在 hello 目录下创建 src\helloworld.c、hello/BUILD.gn，目录结构如下：

```

+---hello
|   BUILD.gn
|   \---src
|       helloworld.c

```

hello/src/helloworld.c 如下：

```

#include <stdio.h>
int main(int argc, char **argv)
{
    printf("\n*****\n");
    printf("\n\t\tHello OHOS!\n");
    printf("\n*****\n\n");

    return 0;
}

```

hello/BUILD.gn 如下：

```

import("//build/lite/config/component/lite_component.gni")
lite_component("hello-OHOS") {
    features = [ ":helloworld" ]
}
executable("helloworld") {
    output_name = "helloworld"
    sources = [ "src/helloworld.c" ]
    include_dirs = []
    defines = []
    cflags_c = []
    ldflags = []
}

```

## 步骤 2 添加新组件

在 windows 下修改 d:\harmonyos\demo2\build\lite\components\applications.json，添加组件 hello\_world\_app 的配置，如下所示为 applications.json 文件片段，"##start##"和"##end##"之间为新增组件配置（"##start##"和"##end##"仅用来标识位置，添加完配置后删除这两行）：

```

{
  "components": [
    {
      "component": "camera_sample_communication",
      "description": "Communication related samples.",
      "optional": "true",
      "dirs": [
        "applications/sample/camera/communication"
      ]
    }
  ]
}

```

```
    ],
    "targets": [
        "//applications/sample/camera/communication:sample"
    ],
    "rom": "",
    "ram": "",
    "output": [],
    "adapted_kernel": [ "liteos_a" ],
    "features": [],
    "deps": {
        "components": [],
        "third_party": []
    }
},
##start##
{
    "component": "hello_world_app",
    "description": "Communication related samples.",
    "optional": "true",
    "dirs": [
        "applications/sample/camera/hello"
    ],
    "targets": [
        "//applications/sample/camera/hello:hello-OHOS"
    ],
    "rom": "",
    "ram": "",
    "output": [],
    "adapted_kernel": [ "liteos_a" ],
    "features": [],
    "deps": {
        "components": [],
        "third_party": []
    }
},
##end##
{
    "component": "camera_sample_app",
    "description": "Camera related samples.",
    "optional": "true",
    "dirs": [
        "applications/sample/camera/launcher",
        "applications/sample/camera/cameraApp",
        "applications/sample/camera/setting",
        "applications/sample/camera/gallery",
        "applications/sample/camera/media"
    ],
    "targets": [
        "//applications/sample/camera/launcher:launcher-OHOS",
        "//applications/sample/camera/cameraApp:cameraApp-OHOS",
        "//applications/sample/camera/setting:setting-OHOS",
        "//applications/sample/camera/gallery:gallery-OHOS",
        "//applications/sample/camera/media:media-OHOS"
    ],
    "rom": "",
    "ram": "",
    "output": [],
    "adapted_kernel": [ "liteos_a" ],
    "features": [],
    "deps": {
        "components": [],
        "third_party": []
    }
}
```

### 步骤 3 修改单板配置文件

修改文件修改 d:\harmonyos\demo2\vendor\hisilicon\hispark\_taurus\config.json，新增 hello\_world\_app 组件的条目，如下所示代码片段为 applications 子系统配置，"##start##" 和 "##end##" 之间为新增条目（"##start##" 和 "##end##" 仅用来标识位置，添加完配置后删除这两行）：

```
{
  "subsystem": "applications",
  "components": [
    { "component": "camera_sample_app", "features":[] },
    { "component": "camera_sample_ai", "features":[] },
    ##start##
    { "component": "hello_world_app", "features":[] },
    ##end##
    { "component": "camera_screensaver_app", "features":[] }
  ]
},
```

### 步骤 4 编译和镜像烧录

详见 3.4 的编译章节和 3.5 的镜像烧录章节。

### 步骤 5 执行应用程序

根目录下，在命令行输入指令“./bin/helloworld”执行写入的 demo 程序，显示成功结果如下图所示。

```
OHOS # ./bin/helloworld
OHOS #
*****
                        Hello OHOS!
*****
```

## 3.4 系统编译与镜像生成

在容器命令行输入命令：

```
hb set
```

出现的界面中输入.（选择当前路径）并回车

出现的界面通过键盘方向键，选择 ipcamera\_hispark\_taurus 并回车，出现如下界面：

```
root@e315a2bb63bc:/home/openharmony# hb set
[OHOS INFO] Input code path: .
OHOS Which product do you need?  ipcamera_hispark_taurus
```

在容器命令行输入命令：

```
hb build -f
```

最后，镜像就会编译出来了，镜像所在目录 out/hispark\_taurus/ipcamera\_hispark\_taurus:

```
root@0e4be02414fc:/home/openharmony# cd out/hispark_taurus/ipcamera_hispark_taurus/
root@0e4be02414fc:/home/openharmony/out/hispark_taurus/ipcamera_hispark_taurus# ls -al
total 188904
drwxr-xr-x 17 root root    12288 May 23 08:01 .
drwxr-xr-x  3 root root     4096 May 23 07:56 ..
-rw-r--r--  1 root root      409 May 23 07:56 args.gn
drwxr-xr-x  2 root root    12288 May 23 08:01 bin
-rw-r--r--  1 root root   19717 May 23 07:57 bm_tool.map
-rw-r--r--  1 root root  382042 May 23 08:01 build.log
-rw-r--r--  1 root root  159069 May 23 07:56 build.ninja
-rw-r--r--  1 root root   21366 May 23 07:56 build.ninja.d
-rw-r--r--  1 root root   55866 May 23 07:57 bundle_daemon_tool.map
drwxr-xr-x  2 root root     4096 May 23 07:58 config
drwxr-xr-x  2 root root     4096 May 23 07:56 data
drwxr-xr-x  3 root root     4096 May 23 07:57 dev_tools
drwxr-xr-x  2 root root     4096 May 23 07:56 etc
-rw-r--r--  1 root root    6469 May 23 08:00 foundation.map
drwxr-xr-x  3 root root     4096 May 23 08:01 gen
drwxr-xr-x  3 root root     4096 May 23 08:01 libs
-rwxr-xr-x  1 root root  1986112 May 23 07:58 liteos.bin
-rw-r--r--  1 root root    8735 May 23 08:00 media_server.map
-rw-r--r--  1 root root   110712 May 23 08:01 .ninja_deps
-rw-r--r--  1 root root   313247 May 23 08:01 .ninja_log
drwxr-xr-x  7 root root     4096 May 23 07:56 NOTICE_FILE
drwx----- 13 root root     4096 May 23 07:56 obj
-rwxr-xr-x  1 root root  7826368 May 23 08:00 OHOS_Image
-rw-r--r--  1 root root  86398333 May 23 08:00 OHOS_Image.asm
-rwxr-xr-x  1 root root   7179804 May 23 08:00 OHOS_Image.bin
-rw-r--r--  1 root root   2048605 May 23 08:00 OHOS_Image.map
-rw-r--r--  1 root root    13411 May 23 07:57 provider_proc.map
-rw-r--r--  1 root root  16619520 May 23 08:01 rootfs.tar
-rw-r--r--  1 root root  16817152 May 23 08:01 rootfs_vfat.img
-rw-r--r--  1 root root   146301 May 23 07:57 server.map
drwxr-xr-x  3 root root     4096 May 23 08:01 suites
drwxr-xr-x  3 root root     4096 May 23 08:00 system
drwxr-xr-x  3 root root     4096 May 23 07:57 test
drwxr-xr-x  4 root root     4096 May 23 07:56 test_info
-rw-r--r--  1 root root   447023 May 23 08:01 toggleButtonTest.map
-rw-r--r--  1 root root   318242 May 23 07:56 toolchain.ninja
drwxr-xr-x  5 root root     4096 May 23 08:01 userfs
-rw-r--r--  1 root root  52428800 May 23 08:01 userfs_vfat.img
drwxr-xr-x  3 root root     4096 May 23 07:56 vendor
```

## 3.5 镜像烧写

本实验通过网口烧录的方式。

### 步骤 1 镜像所在位置

对应执行 docker run 时候映射的文件目录，从虚拟机内的 \home\openharmony 映射到 windows 工作台的 D:\harmonyos 下，所以生成的镜像对应的位置：

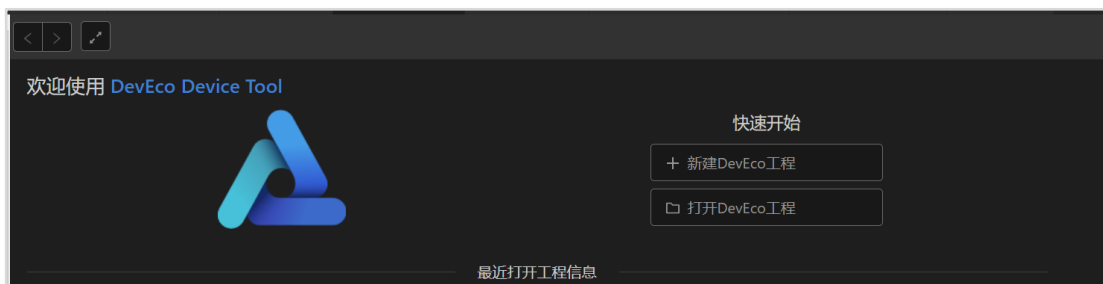
D:\harmonyos\demo2\out\hispark\_taurus\ipcamera\_hispark\_taurus

### 步骤 2 安装 usb 转串口驱动

如已在第二章安装可跳过此步骤，尚未安装的可参照 2.5 节步骤 2。

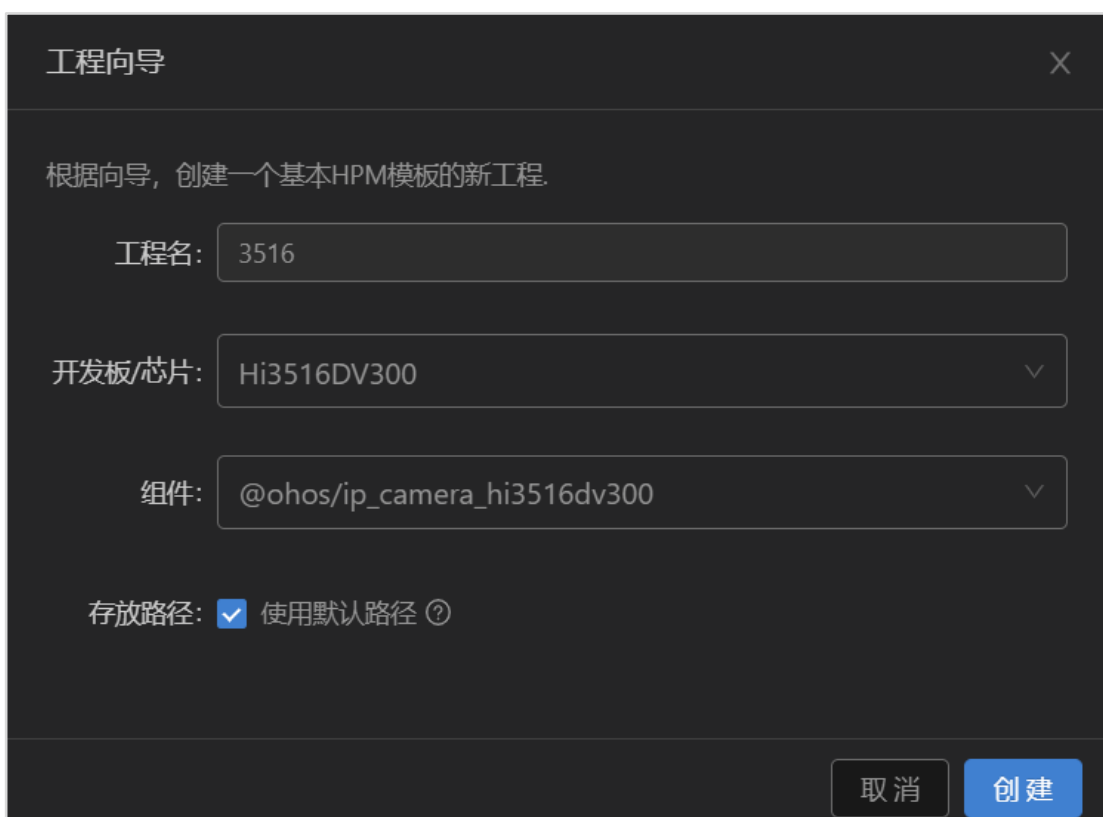
### 步骤 3 在 deveco 中新建工程

打开 visual code 默认会进入 DevEco Device Tool 首页：

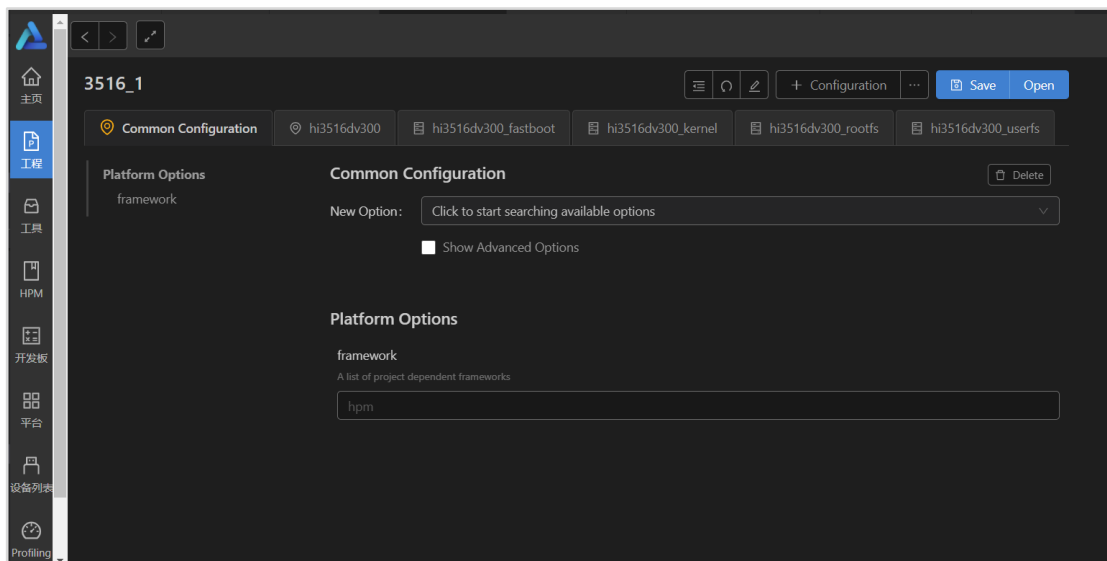


点击新建 DevEco 工程，创建新工程。

跳出的配置项向导中，进行如下配置（工程名可自己配置）后创建：



创建后，进入工程配置页面：



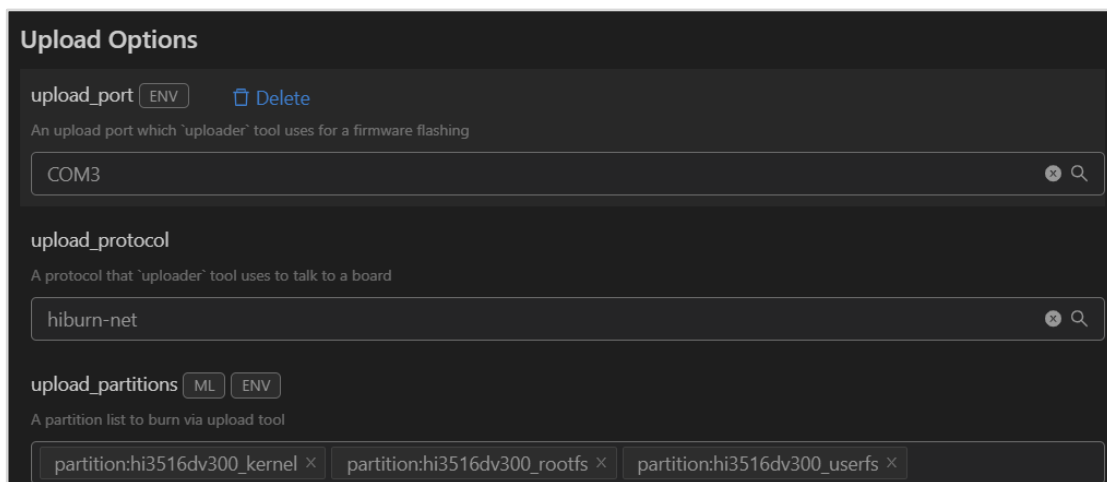
#### 步骤 4 设置烧录选项

在“hi3516dv300”页签，设置烧录选项，包括 upload\_port、upload\_partitions 和 upload\_protocol。

upload\_port：选择步骤 2 中查询的串口号。

upload\_protocol：选择烧录协议，固定选择“hiburn-net”。

upload\_partitions：选择待烧录的文件，默认情况下会同时烧录 fastboot、kernel、rootfs 和 usersfs。（fastboot 一般单板自带的，不需要烧写，点“x”去掉）



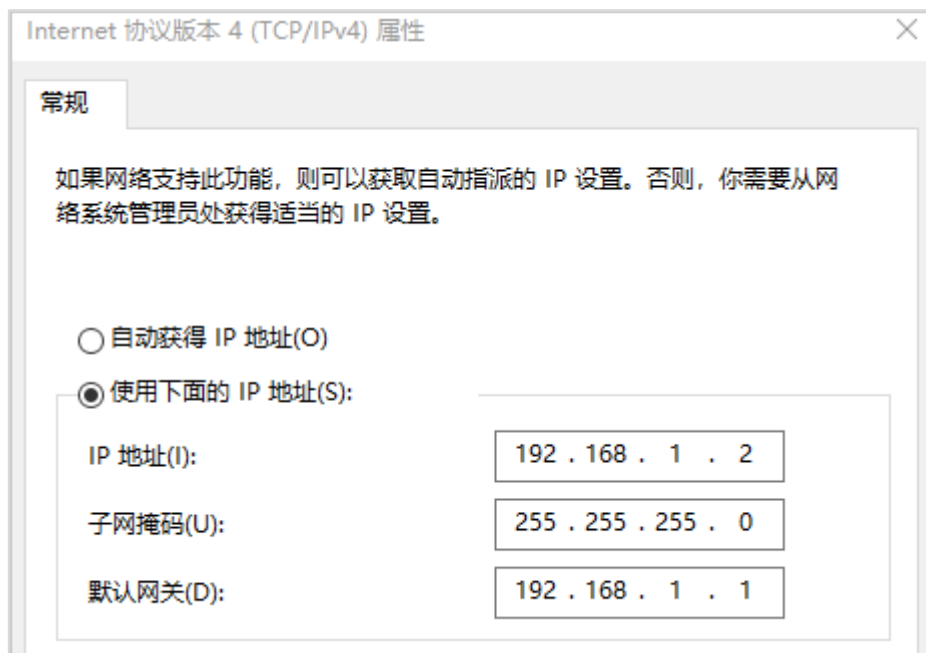
#### 步骤 5 检查和设置连接开发板后的网络适配器的 IP 地址信息

使用网口连接开发板。

打开 PC 上的网络适配器，找到连接开发板后出现的网卡。

点击右键 > 属性 > 网络 > Internet 协议版本 4（TCP/IPv4），打开 IP 地址设置页面。

勾选“使用下面的 IP 地址”，然后手动输入 IP 地址、子网掩码和默认网关，如下所示。



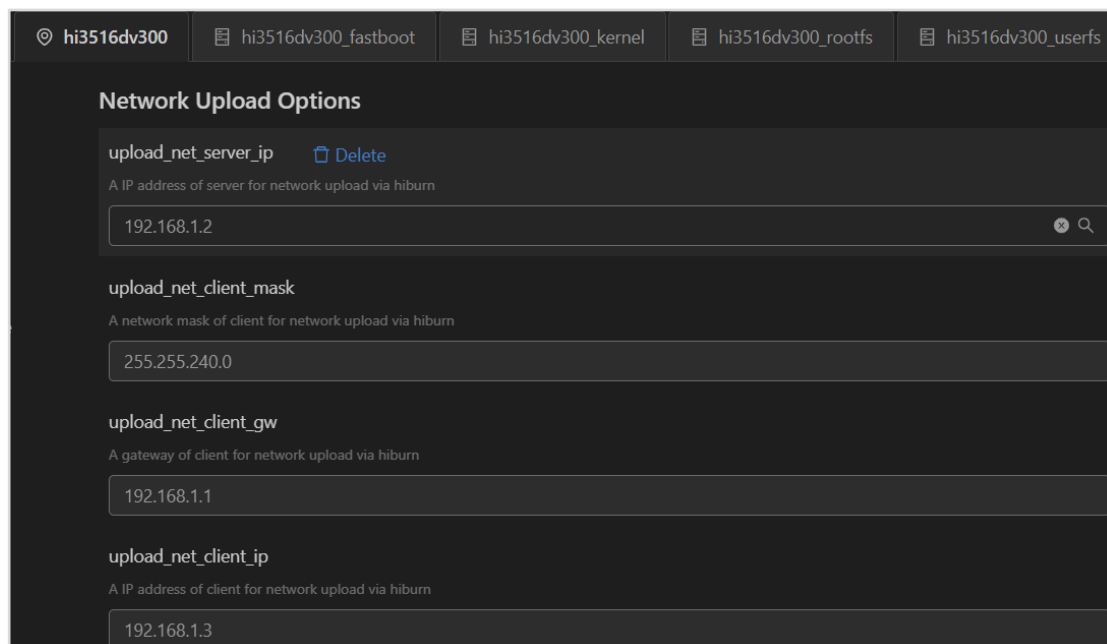
#### 步骤 6 设置网口烧录的 IP 地址信息

upload\_net\_server\_ip: 选择 6 中设置的 IP 地址信息。例如 192.168.1.2

upload\_net\_client\_mask: 设置开发板的子网掩码，工具会自动根据选择的 upload\_net\_server\_ip 进行设置。例如 255.255.255.0

upload\_net\_client\_gw: 设置开发板的网关，工具会自动根据选择的 upload\_net\_server\_ip 进行设置。例如 192.168.1.1

upload\_net\_client\_ip: 设置开发板的 IP 地址，工具会自动根据选择的 upload\_net\_server\_ip 进行设置。例如 192.168.1.3

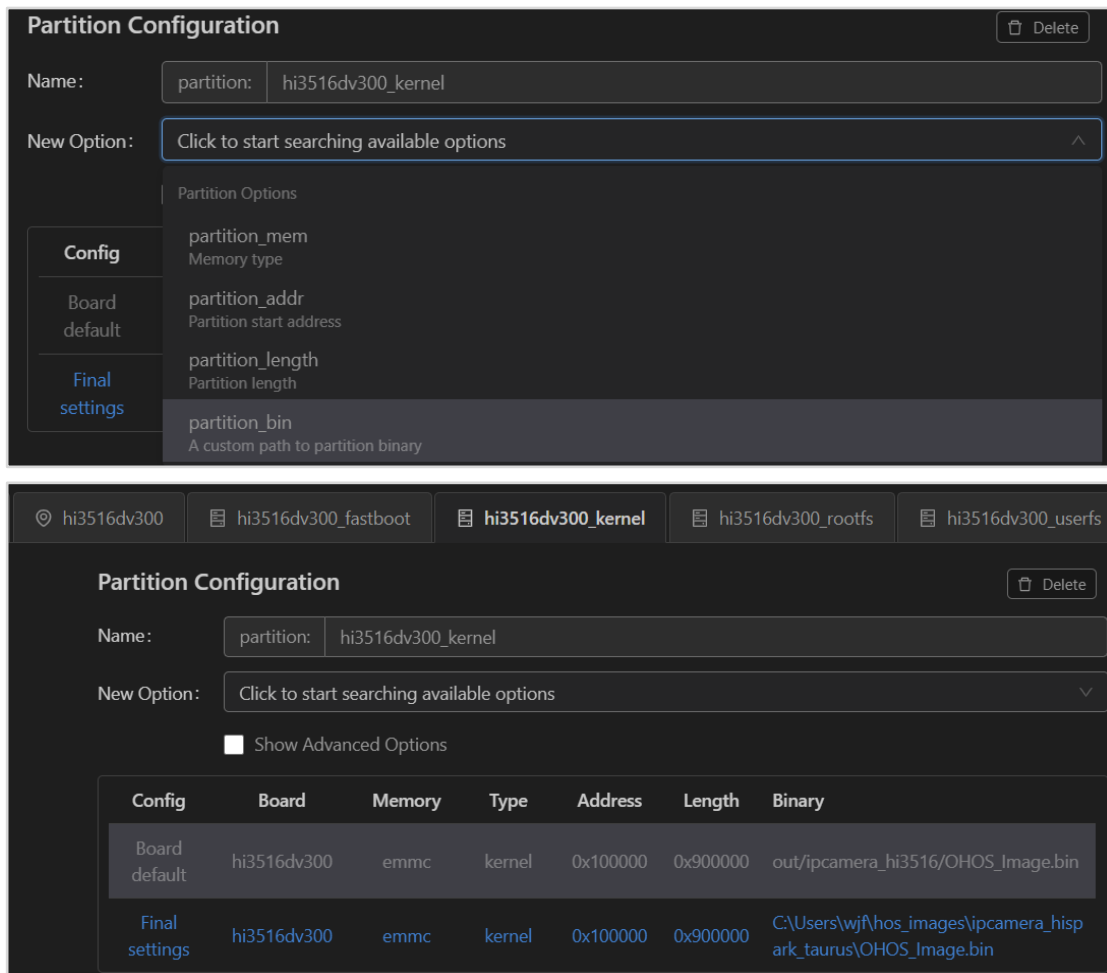




## 步骤 7 修改待烧录文件

对应 hi3516dv300\_fastboot、hi3516dv300\_kernel、hi3516dv300\_rootfs 和 hi3516dv300\_userfs 页签的设置，因为前面在 docker 容器中已经把源码编译完成了，镜像也拷贝到 windows 下的目录下，所以需要设置下镜像的路径。

在 New Option 中，选择 partition\_bin 进行路径更改：



**Partition Configuration** [Delete]

Name: partition: hi3516dv300\_kernel

New Option: Click to start searching available options ^

**Config**

- partition\_mem  
Memory type
- partition\_addr  
Partition start address
- partition\_length  
Partition length
- partition\_bin**  
A custom path to partition binary

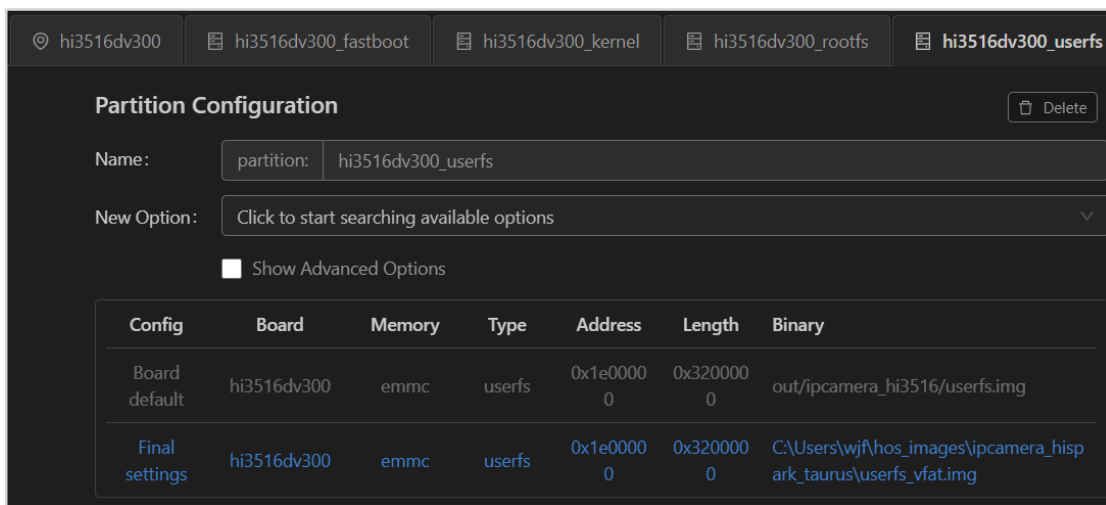
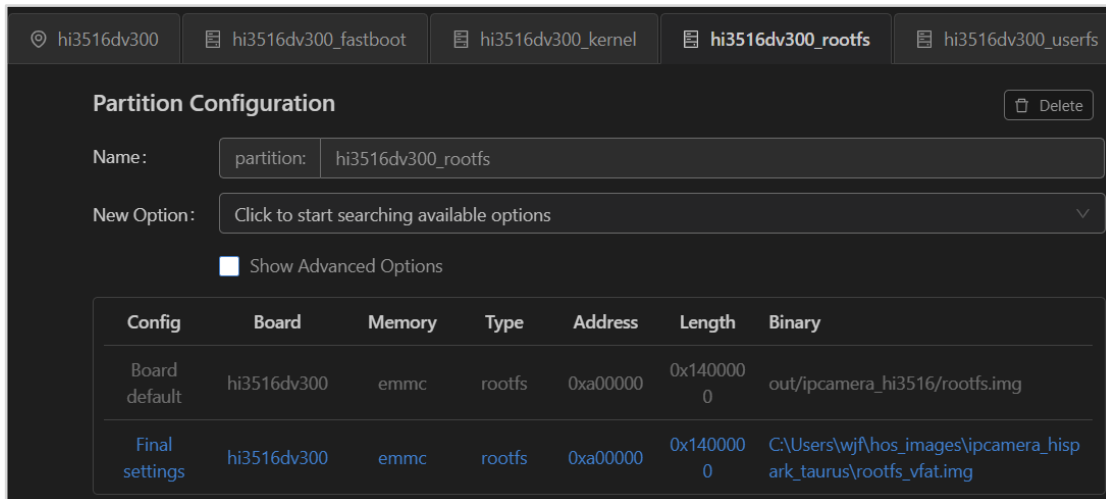
**Board default**

**Final settings**

☒ Show Advanced Options

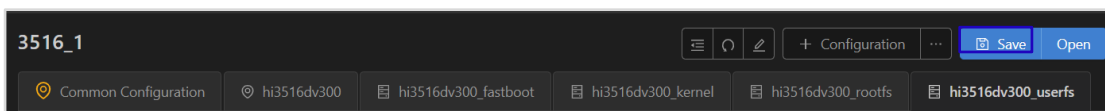
Config	Board	Memory	Type	Address	Length	Binary
Board default	hi3516dv300	emmc	kernel	0x100000	0x900000	out/ipcamera_hi3516/OHOS_Image.bin
Final settings	hi3516dv300	emmc	kernel	0x100000	0x900000	C:\Users\wjf\hos_images\ipcamera_hisp_ark_taurus\OHOS_Image.bin

同理，需要设置 hi3516dv300\_rootfs 和 hi3516dv300\_userfs 页签中的镜像路径，如下：



## 步骤 8 配置保存

所有的配置都修改完成后，在工程配置页签的顶部，点击 Save 进行保存。



## 步骤 9 烧录镜像

点击 图标，打开 DevEco Device Tool 界面，在“PROJECT TASKS”中，点击 env:hi3516dv300 下的 Upload 按钮，启动烧录。

启动烧录后，显示如下提示信息时，请重启开发板（点击单板电源开关下电再上电，即按两下）：

```
问题 5 输出 调试控制台 终端 2: devedco: upload - hi3! + - □ ☰ ^ ×

Verbose mode can be enabled via '-v, --verbose' option
Configuring upload protocol...
AVAILABLE: hiburn-net, hiburn-serial, hiburn-usb
CURRENT: upload_protocol = hiburn-net
Uploading with HiBurn
The burn.config file was loaded successfully!
The current select chip: Hi3516DV300
chip file path: Resources/common/chipProperties/Hi3516CV500.chip

The preference.config file was loaded successfully!

SerialPort has been connented, Please power off, then power on the device.
If it doesn't work, please try to repower on.
```

界面出现如下信息，表示烧录成功：

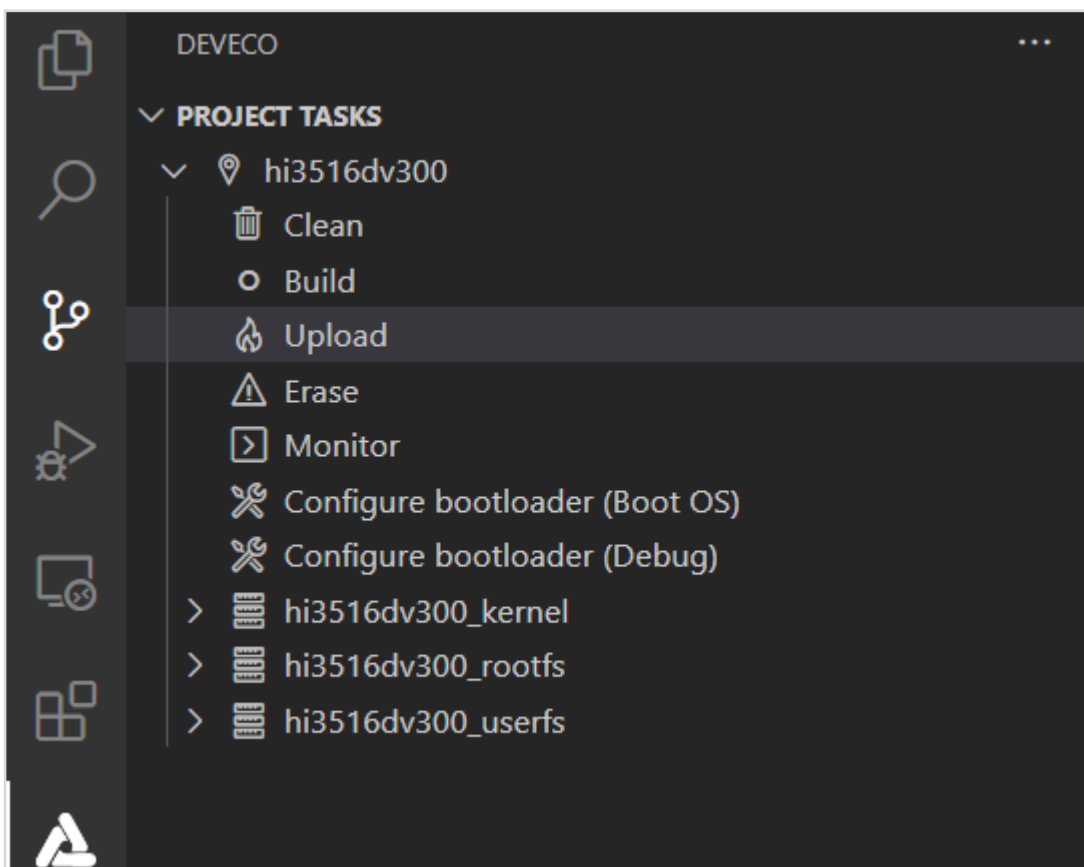
```
问题 5 输出 调试控制台 终端 1: devedco: upload - hi3! + - □ ☰ ^ ×

954.1 KiB/s
done
Bytes transferred = 52428800 (3200000 hex)
[EOT](OK)

Send command: crc32 81000000 32000000
crc32 for 81000000 ... 841fffff ==> 7fbfc06b
[EOT](OK)

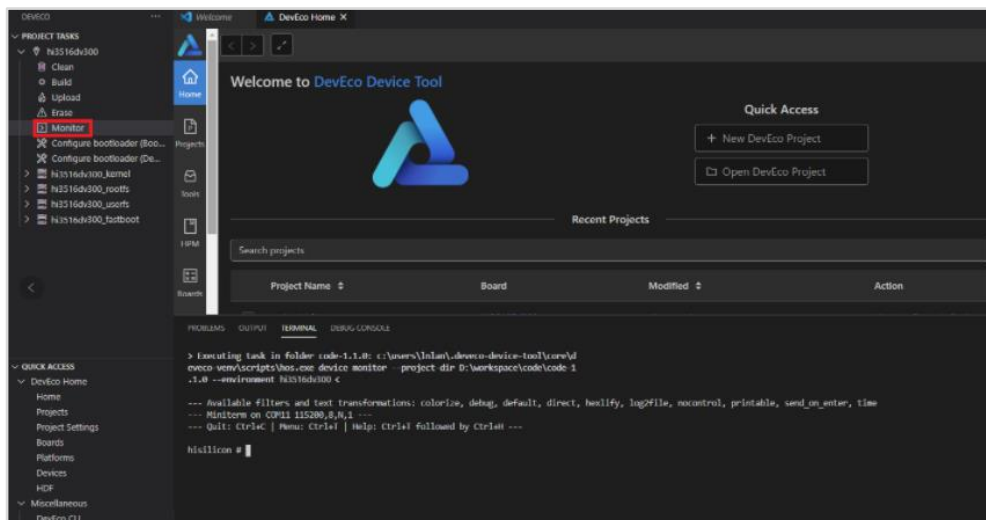
Send command: mmc write 0x0 0x81000000 0xf000 0x19000

MMC write: dev # 0, block # 61440, count 102400 ... 102400 blocks written: OK
37.05 MB/Partition userfs burned successfully!
s
[EOT](
Send command: reset
OK)
Partition burnt completed!
===== [SUCCESS] Took 124.02 seconds =====
```



## 步骤 10 单板启动

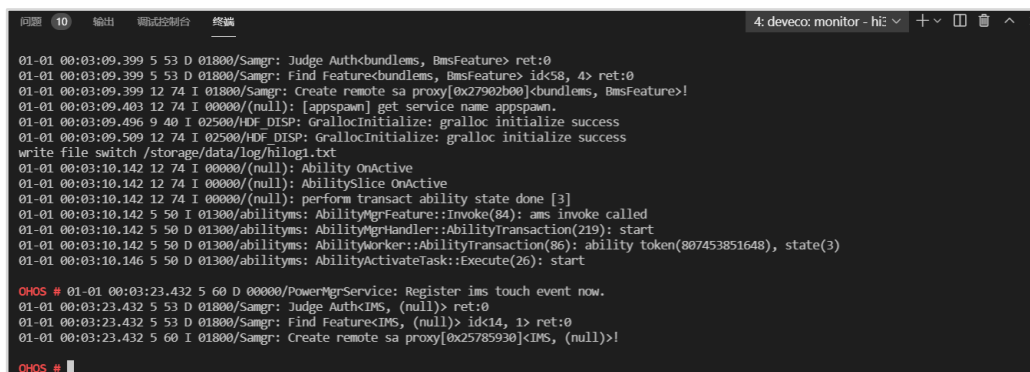
单击 Monitor 打开串口，如下图：



连续输入回车直到串口显示"hisilicon"，在串口窗口中输入如下命令：

```
setenv bootargs "console=ttyAMA0,115200n8 root=emmc fstype=vfat rootaddr=10M rootsize=20M rw"
```

输入完成后回车，就开始 loader 单板中的 harmonyOS 系统，启动完成后串口窗口如下：



单板的界面如下：

