《基于HarmonyOS的智能物联系统快速开发实践》

实验指导手册

**课程简介**

本课程合并自原《OneNET物联网设备云平台应用开发实践》和《物联网原型系统开发》课程，基于华为OpenHarmony OS以及Hi3861开发板，总学时为32学时，包含理论授课8学时，实验指导16学时以及创新实践8学时。

本课程主要介绍华为OpenHarmony OS和Hi3861物联网芯片，并基于前述软硬件平台指导学生学习掌握嵌入式开发基础，了解并掌握OpenHarmony OS编程和Hi3861开发板使用。指导学生利用HarmonyOS和Hi3861开发板自主设计和实现创新课程项目，鼓励和带领学生持续完善和扩充课程项目，参与学科竞赛和双创活动。

本教学完成的目标功能是依托本教学平台和华为智能基座社团，推广华为OpenHarmony OS生态，让学生掌握理论知识的同时提高动手实践和创新能力；鼓励和引导学生持续孵化课程项目，参加学科竞赛和创新创业活动。具体为指导学生完成相应的基础知识学习以后，使学生具有自主进行搭载有OpenHarmony OS的嵌入式设备开发的能力。其包括但不限于：远程小车控制平台——通过华为云与OpenHarmony OS的互通，实现云端下发指令控制小车，适用于智慧城市的智慧交通管控；自动园区巡逻小车——通过小车携带的超声波，红外线等传感设备，实现室内小车巡逻功能，适用于智慧园区的智慧管理等。

本教学实验安排为两个部分：基础实验学习以及自主项目设计开发。

第一部分-基础实验学习旨在使学生掌握基本的嵌入式设备开发概念，了解OpenHarmony OS的基础特性，正确使用设备的软硬件接口，为第二部分自主项目设计开发做好充分知识、技能准备。该部分实验包含：环境配置、认识设备、OpenHarmony OS系统基础实验、OpenHarmony OS中WIFI通信。

第二部分-自主项目设计旨在充分调动学生能动性，训练学生自主学习，团队开发的能力。学生基于组队开发的模式，通过小组讨论，老师指导等方式，确定项目主题，随后自主进行项目开发，最终进行项目的验收与答辩。

目录

[1 器材需求 4](#_Toc144457272)

[1.1 润和Pegasus智能小车开发套件 4](#_Toc144457273)

[1.2 小车安装 5](#_Toc144457274)

[1.3 其他设备 6](#_Toc144457275)

[2 HarmonyOS与Hi3861开发环境配置 7](#_Toc144457276)

[2.1 开发准备 7](#_Toc144457277)

[2.3 CH340串口驱动安装 10](#_Toc144457278)

[2.2 Hi3861开发环境配置 11](#_Toc144457279)

[3.2 超声波模块 14](#_Toc144457280)

[3 关键硬件模块分析介绍 18](#_Toc144457281)

[3.1 寻迹模块 18](#_Toc144457282)

[3.3 电机模块 19](#_Toc144457283)

[3.4 舵机模块 20](#_Toc144457284)

[4 OpenHarmony基础实验-I 21](#_Toc144457285)

[4.1 Hello World 21](#_Toc144457286)

[4.1.1 实验目的 21](#_Toc144457287)

[4.1.2 实验步骤 21](#_Toc144457288)

[4.1.3 实验结果 24](#_Toc144457289)

[4.1.4 示例代码 24](#_Toc144457290)

[4.1.5 开发流程总结 24](#_Toc144457291)

[4.2 点亮LED 25](#_Toc144457292)

[4.2.1 实验目的 25](#_Toc144457293)

[4.2.2 实验步骤 26](#_Toc144457294)

[4.2.2.1 Task 1 26](#_Toc144457295)

[4.2.2.2 Task 2 26](#_Toc144457296)

[4.2.2.3 Task3 26](#_Toc144457297)

[4.2.2.4 扩展 26](#_Toc144457298)

[4.2.3 实验结果 27](#_Toc144457299)

[4.2.4 推荐阅读 27](#_Toc144457300)

[4.3 超声波模块与电机结合 27](#_Toc144457301)

[4.3.1 实验目的 27](#_Toc144457302)

[4.3.2 实验步骤 27](#_Toc144457303)

[4.3.2.1 Task 1-配置超声波模块 27](#_Toc144457304)

[4.3.2.2 Task 2-实现测距 27](#_Toc144457305)

[4.3.2.3 Task 3-配置电机模块 27](#_Toc144457306)

[4.3.2.4 Task 4-实现前进和刹车 27](#_Toc144457307)

[4.3.2.5 Task 5-组合控制 28](#_Toc144457308)

[4.3.2.6 扩展 28](#_Toc144457309)

[4.3.3 实验结果 28](#_Toc144457310)

[4.3.4 推荐阅读 28](#_Toc144457311)

[5 OpenHarmony基础实验-II 28](#_Toc144457312)

[5.1 PWM实现呼吸灯 28](#_Toc144457313)

[5.1.1 实验目的 28](#_Toc144457314)

[5.1.2 实验步骤 29](#_Toc144457315)

[5.1.2.1 Task 1: PWM初始化 29](#_Toc144457316)

[5.1.2.2 Task 2: 实现呼吸灯 29](#_Toc144457317)

[5.1.3 实验结果 29](#_Toc144457318)

[5.1.4 推荐阅读 29](#_Toc144457319)

[5.2 光通信模块手势识别 29](#_Toc144457320)

[5.2.1 实验目的 29](#_Toc144457321)

[5.2.2 实验步骤 30](#_Toc144457322)

[5.2.2.1 Task 1-配置ADC 30](#_Toc144457323)

[5.2.2.2 Task 2-获取电压值 30](#_Toc144457324)

[5.2.3 实验结果 30](#_Toc144457325)

[5.2.4 推荐阅读 30](#_Toc144457326)

[5.3 多线程 30](#_Toc144457327)

[5.3.1 实验目的 30](#_Toc144457328)

[5.3.2 实验步骤 31](#_Toc144457329)

[5.3.2.1 扩展 31](#_Toc144457330)

[5.3.3 实验结果 31](#_Toc144457331)

[5.3.4 示例代码 31](#_Toc144457332)

[5.3.5 推荐阅读 32](#_Toc144457333)

[6 OpenHarmony 组网 32](#_Toc144457334)

[6.1 实验目的 32](#_Toc144457335)

[6.2 实验步骤 32](#_Toc144457336)

[6.3 实验结果 32](#_Toc144457337)

[6.4 推荐阅读 32](#_Toc144457338)

[7 OpenHarmony 上云 33](#_Toc144457339)

[7.1 实验目的 33](#_Toc144457340)

[7.2 实验步骤 33](#_Toc144457341)

[7.3 实验结果 33](#_Toc144457342)

[7.4 推荐阅读 33](#_Toc144457343)

[7.5 小结补充 33](#_Toc144457344)

[7.6 华为云MQTT相关 36](#_Toc144457345)

# 1 器材需求

## 1.1 润和Pegasus智能小车开发套件

本课程所需基础设备由润和提供，为润和Pegasus智能小车开发套件，其套件明细如表1-1所示：

表1-1 润和Pegasus智能小车开发套件明细

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **类别** | **描述（型号/品名）** | **数量** |
| 1 | WiFi 主板 | 型号：HiSpark\_WiFi-IoT\_Hi3861\_CH340G\_VER.A | 1 |
| 2 | 通用底板 | 型号：HiSpark\_WiFi\_IoT\_EXB\_VER.A | 1 |
| 3 | 显示板 | 型号：HiSpark\_WiFi\_IoT\_OLED\_VER.A | 1 |
| 4 | NFC 板 | 型号：HiSpark\_WiFi\_IoT\_NFC\_VER.A | 1 |
| 5 | 机器人板 | 型号：HiSpark\_WiFi\_IoT\_Robot\_VER.A | 1 |
| 6 | 智能小车底盘 | 型号：2WD；尺寸：21.4 x 15 cm | 1 |
| 7 | TT 马达 | 黄色；3V~6V；单轴；1:48 | 2 |
| 8 | TT 马达固定支架 | 亚克力马达支架 | 4 |
| 螺丝：M3\*30 | 4 |
| 螺帽：M3 | 4 |
| 9 | 橡胶轮胎 | 黄色；65\*27mm | 2 |
| 10 | 万向轮 | 1 寸；白色 PP 材质 | 1 |
| 11 | 寻迹模块 | 红外寻迹传感器 | 2 |
| 12 | 寻迹连接线 | 2.54mm 间距；3PIN；母对母双头并排带外壳连接线 | 2 |
| 13 | 舵机+舵机配件 | 微型 SG90 舵机 | 1 |
| 舵机支架：单端**\***x1；双端 x1；四端**\***x1 | 3 |
| 支架螺丝：M2\*4 | 1 |
| 自攻螺丝：M2\*8 | 2 |
|  |  | 自攻螺丝：M2\*6 | 2 |
| 14 | 舵机支架 | SG90 舵机支架 | 1 |
| 螺丝：M3\*10 | 2 |
| 螺帽：M3 | 2 |
| 螺丝：M2\*10 | 2 |
| 螺帽：M2 | 2 |
| 15 | 超声波模块 | 型号：HC-SR04；电压：3.3V~5V | 1 |
| 16 | 超声波支架 | 超声波固定支架 | 1 |
| 螺丝：M2\*6 | 2 |
| 螺帽：M2 | 2 |
| 17 | 超声波连接线 | 4PIN 对 4PIN；PH2.0mm 转杜邦 2.54mm | 1 |
| 18 | NFC 板连接线 | 6PIN 对 6PIN；1.27mm；反向异面；20cm | 1 |
| 19 | 电池盒 | 锂电池（18650）盒；两节；串联；带 2.54mm 插头 | 1 |
| 干电池（5 号）盒；四节；串联；带 2.54mm 插头 | 1 |
| 20 | 配件袋 | 螺帽：M3 | 6 |
| 螺丝：M3\*6 | 24 |
| 螺柱：M3\*8+6 | 4 |
| 螺柱：M3\*30 | 10 |

套件中包含Hi3861开发板(WiFi主板)、扩展板、小车马达等，能够充分满足课程教学需求，如图1-1所示。



图1-1 套件设备图

## 1.2 小车安装

小车套件安装可参考润和官方指导视频：[【鸿蒙学院】HiSpark Wi-Fi IoT智能小车（鸿蒙【HarmonyOS】系统）套件安装指南\_](https://www.bilibili.com/video/BV1AD4y1Z7mU/?spm_id_from=333.337.search-card.all.click&vd_source=812be8baf0074e4d7fd5491cf54f958f) ，或在教学课堂中询问指导老师以及课程助教。安装完成后的小车如图1-2所示：

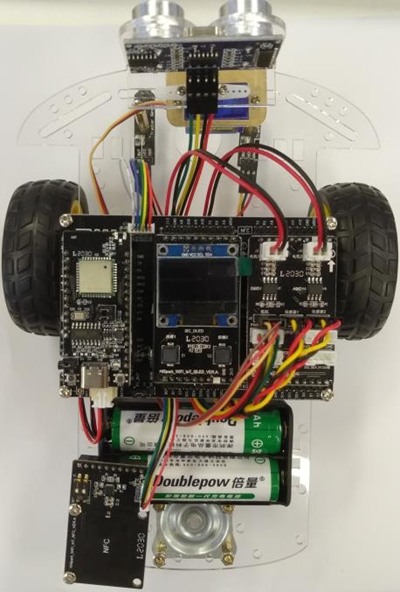


图1-2 小车外观图（组装完毕）

## 1.3 其他设备

本教学中，还需要如图1-3所示光电感应模块，由教学团队提供。该模块可感应根据光强变化输出不同的电压，用于完成OpenHarmony基础实验-II。使用方法为给模块供电后，使用杜邦线将ADC0\_DM0或ADC0\_DP0端口与Hi3861带有ADC功能的引脚连接。

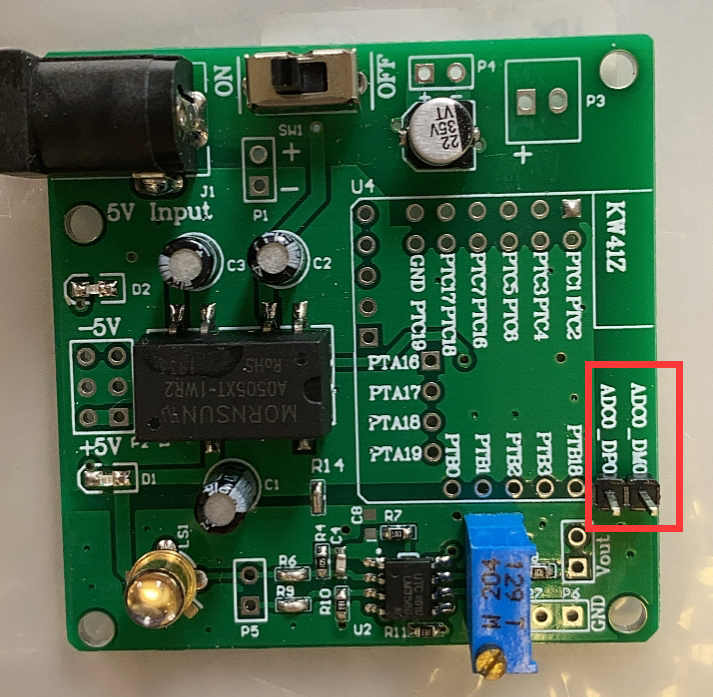


图1-2 光电感应模块

同时，学生可根据自主项目需要，自行设计或者购买额外的元器件进行开发，以满足项目开发的需求。

# 2 HarmonyOS与Hi3861开发环境配置

## 2.1 开发准备

本教学使用开发工具为华为官方提供的Huawei DevEco Device Tool。其配套支持Harmony OS的应用及服务开发，提供了代码智能编辑，低代码开发，双向预览等功能，支持一键编译烧录（仅Hi3861 V100），能够提高用户开发效率与开发体验。其对系统要求如图2-1所示：



图2-1 系统要求

该工具可在官网：[华为集成开发环境IDE DevEco Device Tool下载 | HarmonyOS设备开发](https://device.harmonyos.com/cn/develop/ide#download)进行下载，注意版本号一定要在v3.1以上，否则无法在Windows平台下进行编译，需要搭建Ubuntu+Windows混合开发环境（较复杂，不推荐）。本教学后续内容统一使用Windows平台下的v3.1 Release版本。

首先，进入官网，向下滑动，找到“更多版本”按钮并点击，进入历史版本下载页面，如图2-2所示。

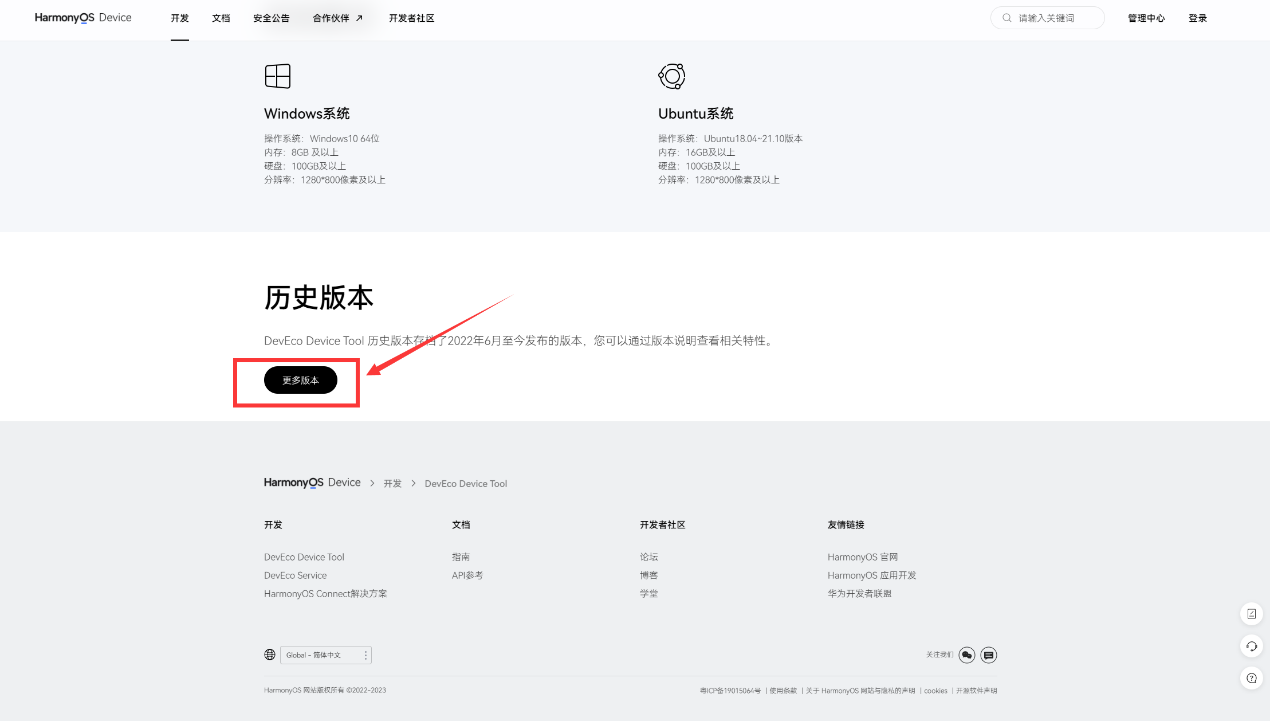


图2-2 历史版本

然后，找到DevEco Studio 3.1 Release，选择Windows(64-bit)版本，点击下载，如图2-3所示。

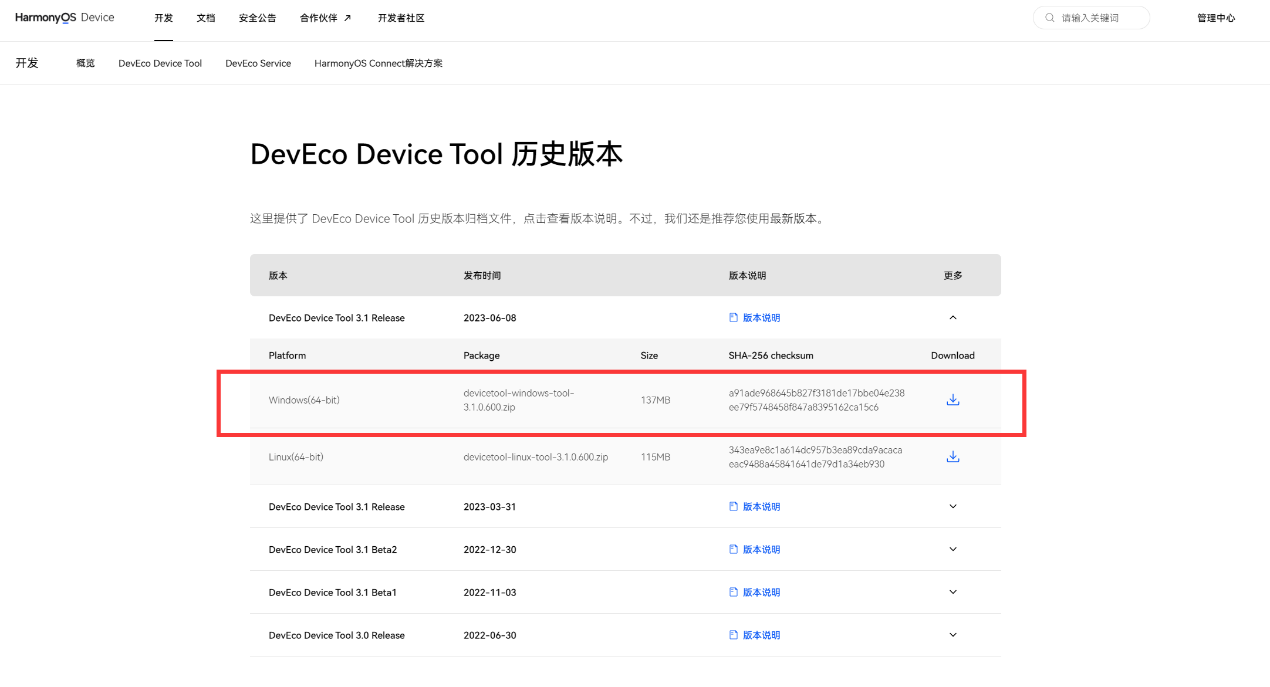


图2-3 下载页面

下载完成之后，执行以下步骤：

1. 解压压缩包，双击安装包程序，单击下一步进行安装。

2. 勾选“我接受许可证协议中的条款”。

3. 设置DevEco Device Tool的安装路径，单击下一步，如图2-4所示。请注意安装路径不能包含中文字符，同时不要安装到系统盘中(大部分情况下为C盘)。



图2-4 安装示意-I

4. 根据安装向导，安装或选择已安装（自定义安装）的依赖工具Python以及VSCode。确保各软件状态显示为OK后，如图2-5所示，单击安装。

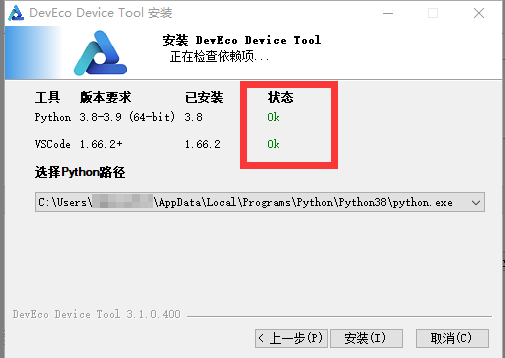


图2-5 安装示意-II

5. 等待安装DevEco Device Tool，当显示图2-6界面时，单击完成，关闭安装向导。



图2-6 安装示意-III

6. 打开VSCode，点击DevEco Device Tool图标，若显示如图2-7界面，则安装成功。

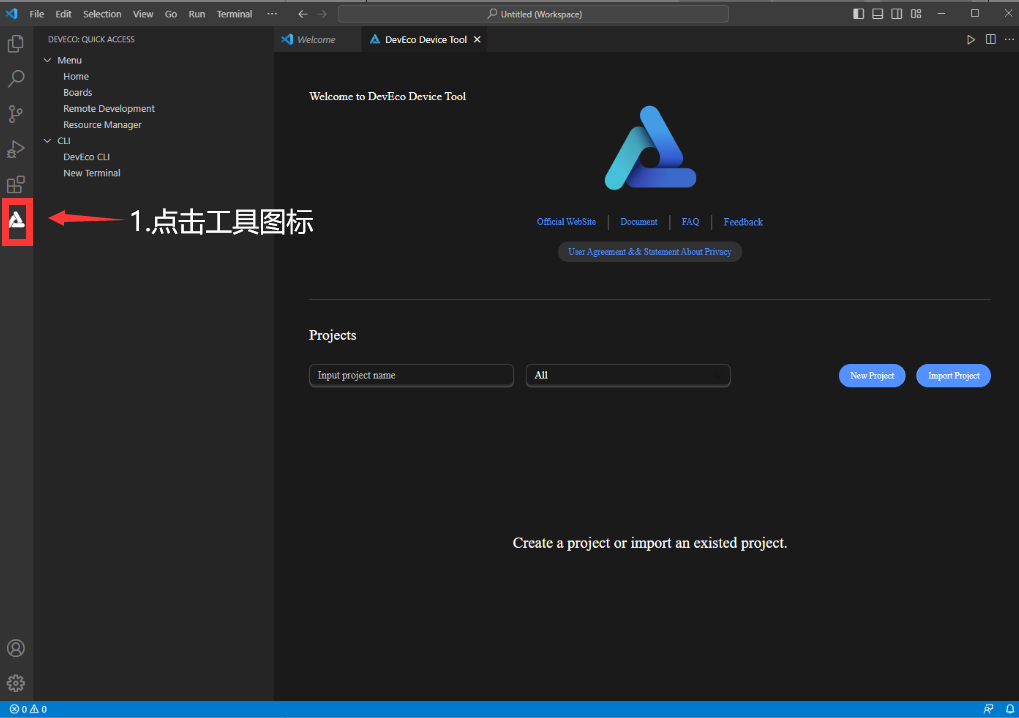


图2-7 安装成功界面

## 2.3 CH340串口驱动安装

芯片与电脑之间的通信，需要有专门的串口实现，该串口无法直接被Windows识别，因此需要安装对应的串口驱动。可在此处下载：[HiHope官网](http://www.hihope.org/download/download.aspx?mtt=8)，如图2-8所示。下载完成后，解压安装即可。



图2-8 CH340驱动下载页面

## 2.2 Hi3861开发环境配置

请按照以下步骤，配置Hi3861开发环境：

1. 打开VSCode，进入DevEco Device Tool界面，单击Home页面后，点击New Project按钮新建工程，如图2-9所示。

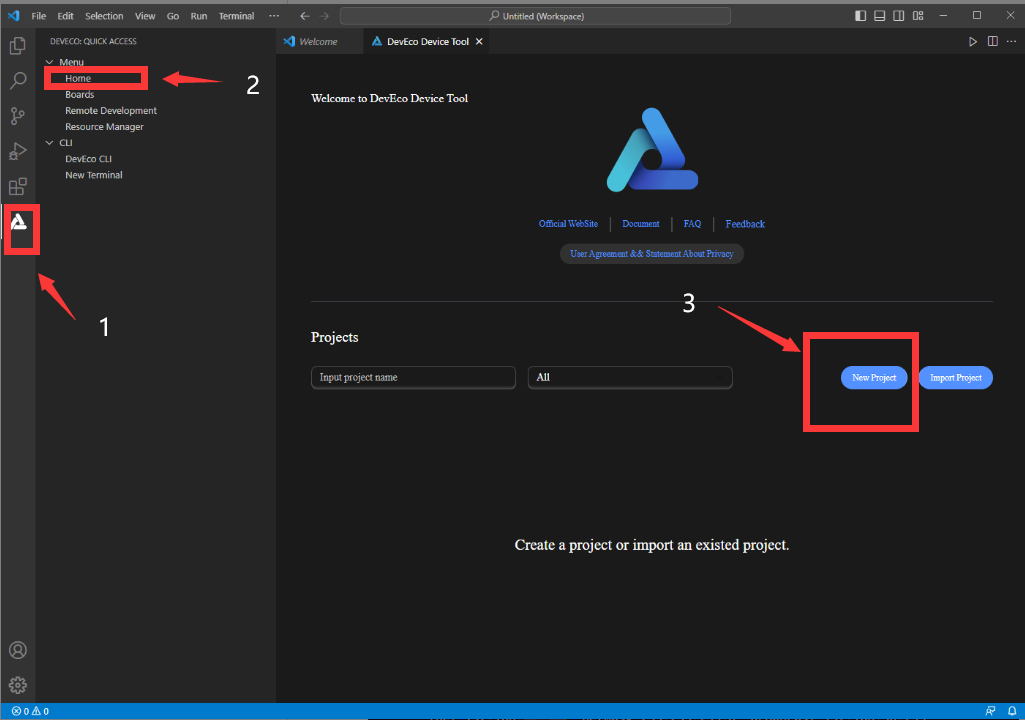
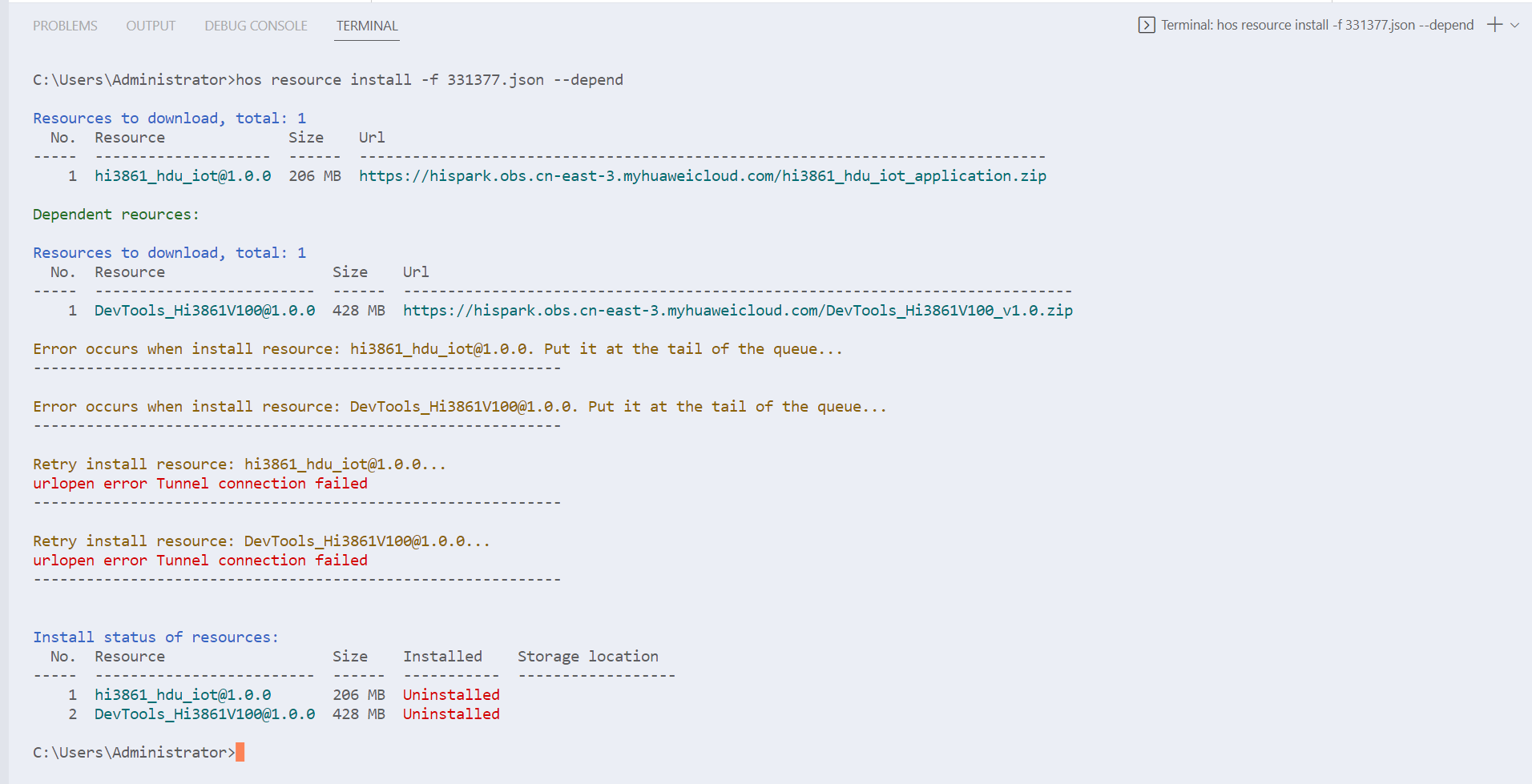


图2-9 新建工程页面-I

**Warning (2023/12/16): 由于SDK和工具包的下载链接都挂载在华为云Obs上，有可能失效。因此，2，3步的Download可能会失败（Issue请见：**[工具包链接失效](https://gitee.com/HiSpark/hi3861_hdu_iot_application/issues/I8KWEK?from=project-issue) **）。如图X-1所示：**



**图X-1 Error页面-I**

**请执行以下步骤，手动导入。SDK和工具链都在Tools文件夹下注意：**

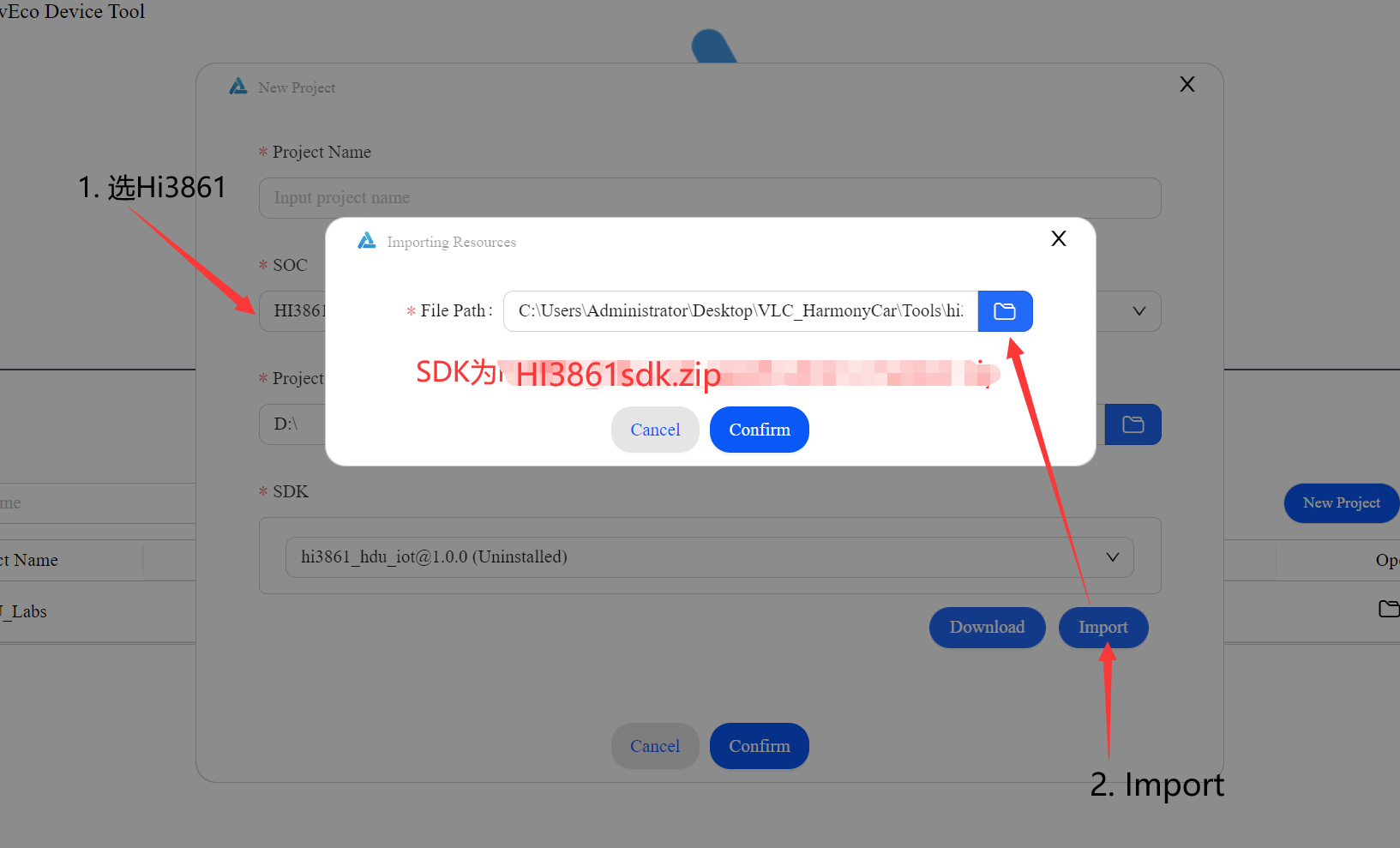
**zip文件的存放路径不要太长!**

**zip文件的存放路径不要太长!**

**zip文件的存放路径不要太长!**

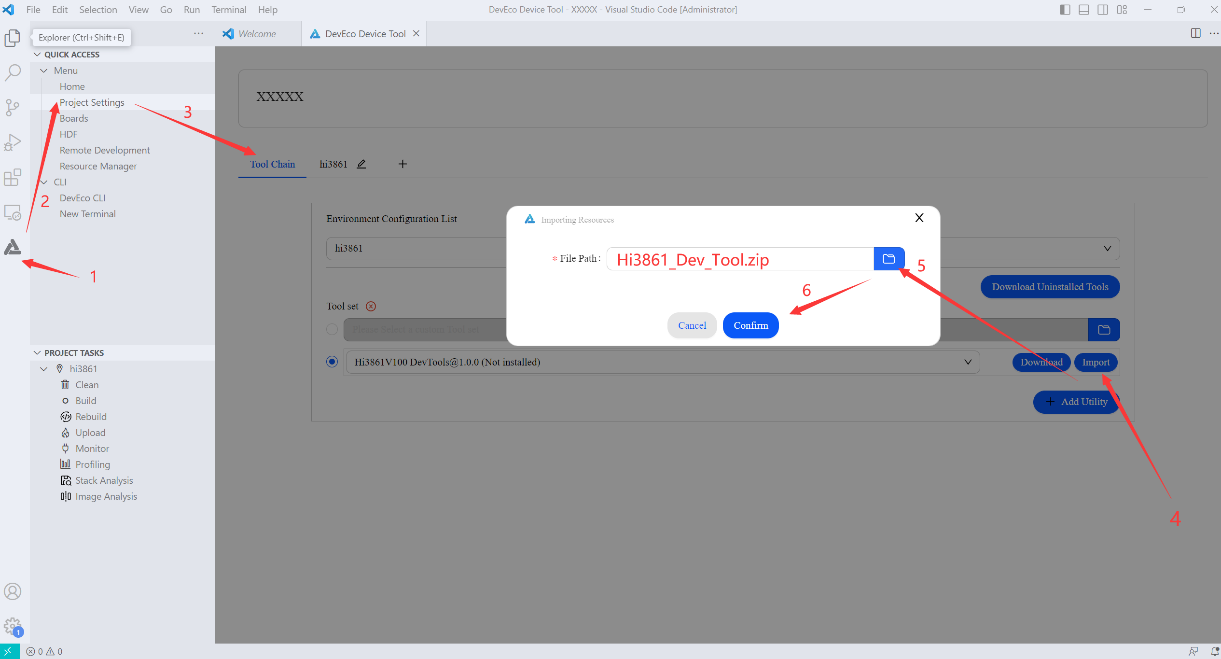
**建议把工程直接放在“盘符根目录”，即D:/XX下。**

1. **导入SDK，见图X-2**



**图X-2 导入SDK (图上路径太长导入失败了)**

1. **导入工具链，见图X-3**



**图X-3 导入Tool**

2. 配置工程参数：“Project Name”中输入项目名称，在“SOC”下拉列表中选择“Hi3861”，“Board”类型会在选择SOC后自动联想为“Hi3861”，在“Project Path”中配置工程存放路径（注意，该路径不能包含中文且长度不能超过260个字符），“SDK”中自动联想为Hi3861工程所需SDK及状态，如显示为“Unistalled”，可单击“Download”直接下载安装。

3. 配置工具链：单击左侧菜单栏的ProjectSettings，进入Hi3861工程配置界面。在Tool Chain页签中，点击“Download Unistalled Tools”，自动安装所需工具，如图2-10所示。

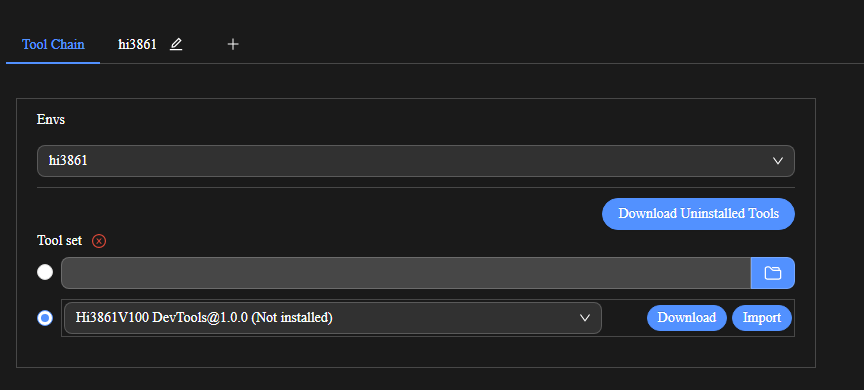


图2-10 新建工程页面-II

4. 编译：工具链配置完成后，点击Build进行编译，如图2-11所示。

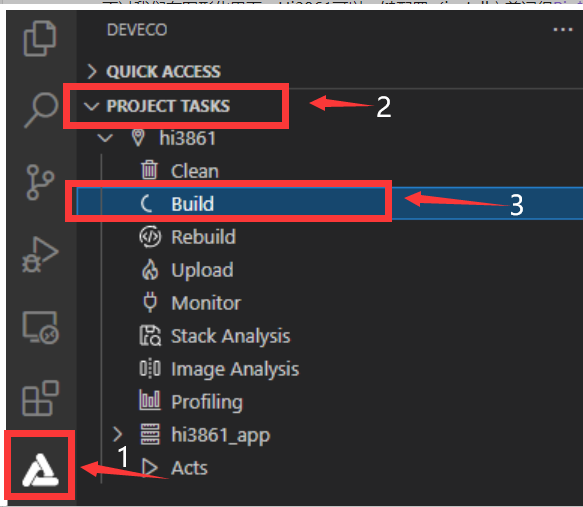


图2-11 编译

等待编译完成后，若出现图2-12所示画面，则代表编译成功，可进行烧录操作。



图2-12 编译成功

5. 烧录：首先，使用Type-C数据线连接，开发板与电脑的USB口。然后，在“Project Settings”菜单中，在右侧upload下方的upload\_port中选择端口，如图2-13所示（如COM5[USB-SERIAL CH340(COM5)]，以实际端口号为准，可重复拔插type-c数据线确定具体端口）。

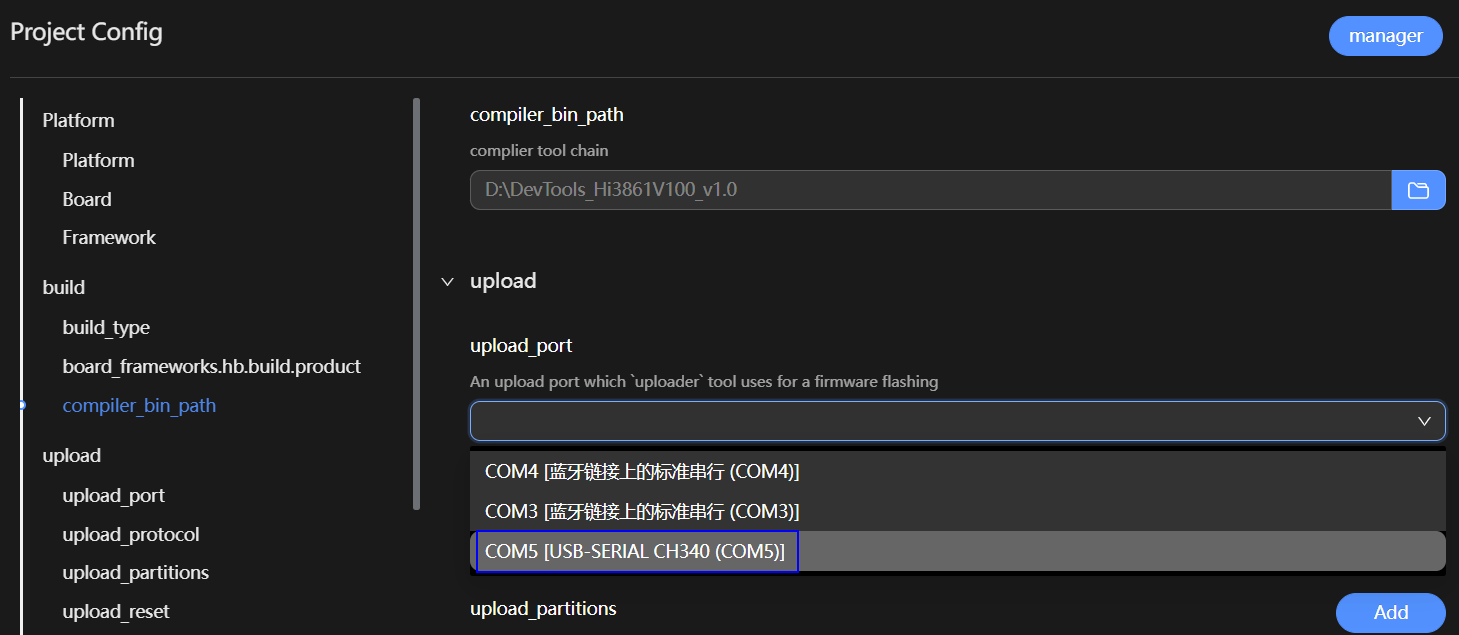


图2-13 选择串口

端口号配置完毕后，在DevEco Device Tool界面的PEOJECT TASK中，点击Upload，执行烧录。当中断出现“Connecting, please reset device…”时，在开发板上按下RST键进行开发板复位，以进入烧录模式，如图2-14，2-15所示。

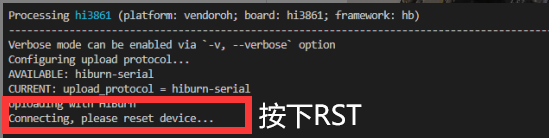


图2-14 烧录过程提示

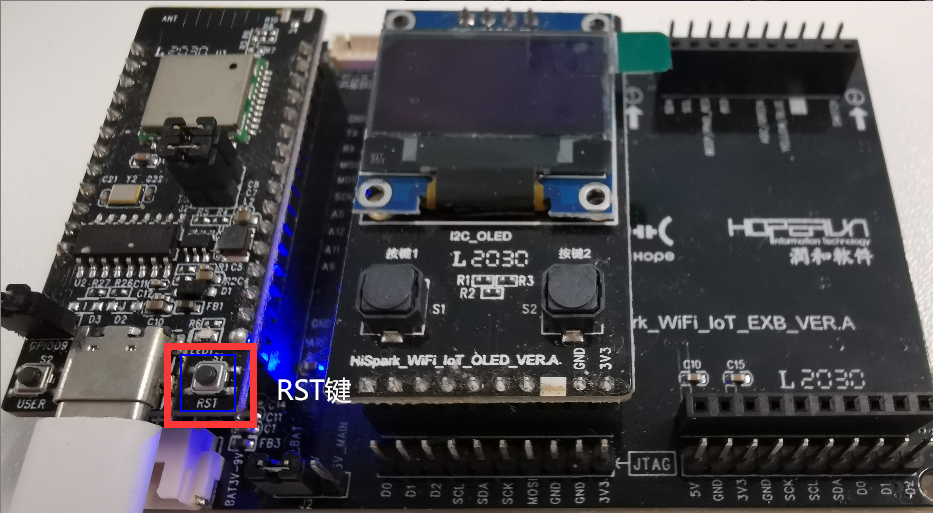


图2-15 RST按键示意

当出现图2-16所示画面时，即烧录成功，此时重新按下RST键复位芯片，即可执行烧录程序。

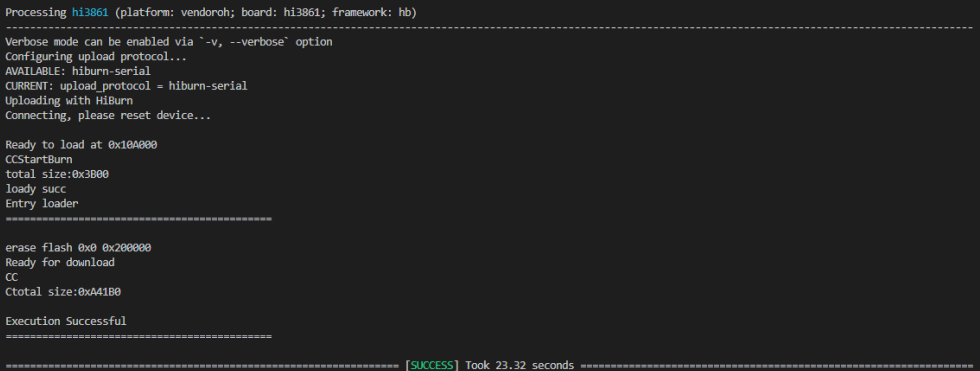


图2-16 烧录成功

## 3.2 超声波模块

参考资料：

1. [超声波HC-SR04模块原理图\_hc-sr04原理图\_被硬件攻城的狮子的博客](https://blog.csdn.net/weixin_44962352/article/details/120265699)

2. 一次弄懂低通、高通、带通、带阻、状态可调滤波器!

超声波模块电路原理图如图3-3所示：

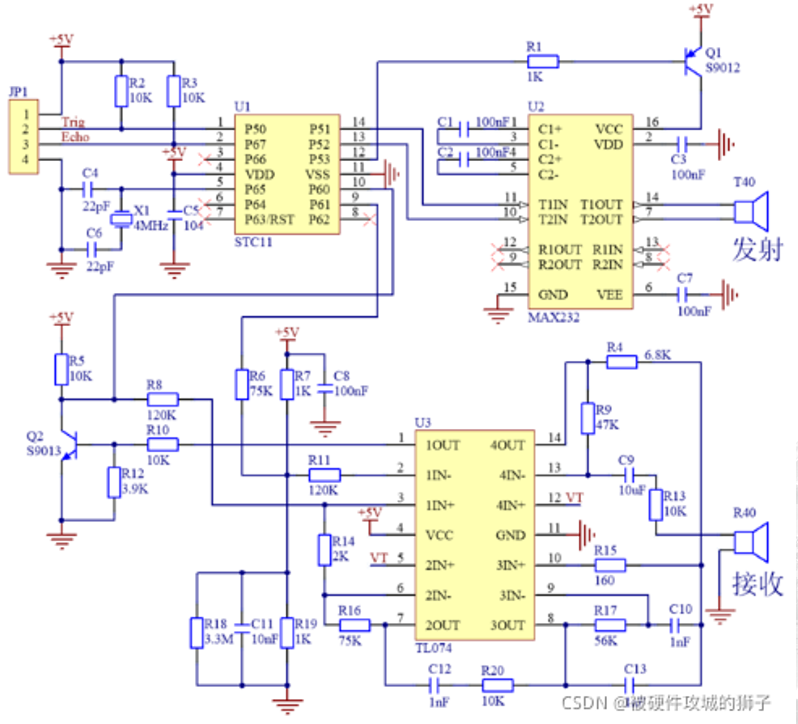


图3-3 超声波模块电路原理图

超声波电路模块包括STC11单片机，MAX232电平转换芯片（作为超声波发射器T40驱动器），TL074运算放大器作为接收驱动电路，放大接收回波信号。其测距原理如图3-4所示为：

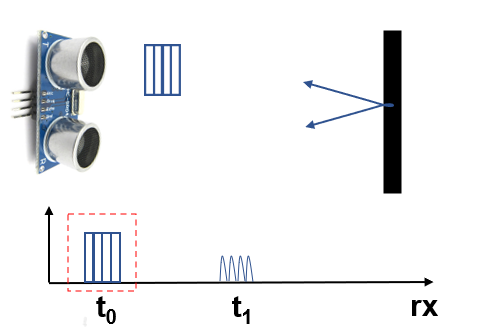


图3-4测距原理

通过障碍物反射超声波序列，接受探头R40将接收的回波转换为40kHz正弦波，经过信号放大和滤波，通过STC11单片机处理后，得到回波脉冲信号。根据两者的时间间隔，由

计算出距离。

其中，一级放大输出为：

VT为参考电压，一般为1/2 VCC，在本电路中为2.5V，电容“通交隔直”可1认为阻抗为零，因此一级放大将输入波形放大4.7倍，并将信号的直流分量移到2.5V附近，如图3-5所示。

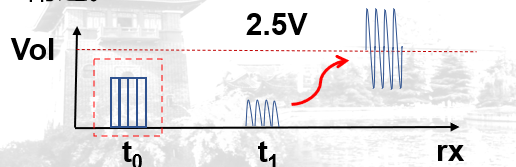
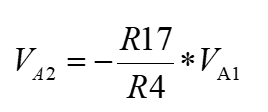


图3-5 一级放大

二级二阶滤波幅度输出有：



该滤波器为有源滤波器， 其电路增益可等效为反向输入放大器的增益，根据电路参数约等于8.2倍。信号将进一步经过比较输出，得到回波脉冲信号，如图3-6所示。

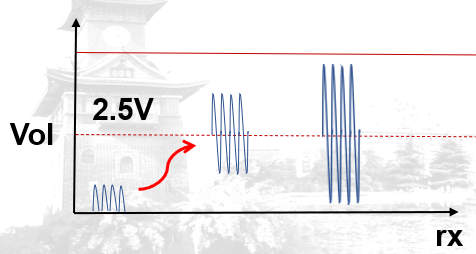


图3-6 二级滤波

流程总结如下：主控通过Trig引脚，控制超声波模块发送一串40kHz方波脉冲。在接收到回波后，通过模块处理，在Echo引脚输出回波脉冲。通过计算Trig和Echo引脚上信号的时差，就可以得到目标距离，如图3-7所示。

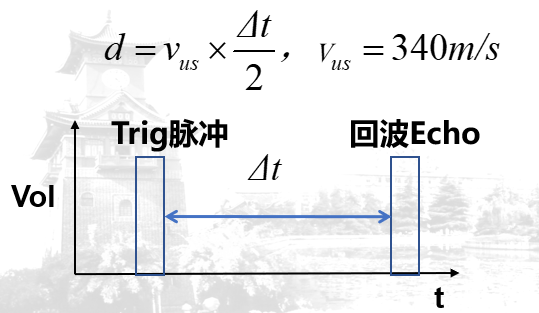


图3-7 过程总结

# 3 关键硬件模块分析介绍

## 3.1 寻迹模块

寻迹模块电路原理图如图3-1所示：



图3-1 寻迹模块原理图

其应用示意图如图3-2所示：



图3-2 寻迹模块应用示意图

假设VCC为3.3V，VR1可变电阻抽头在1/2处。

则红外LED的电流：

VR1的抽头电压：

假设白纸反射能力是黑线反射能力的2倍，且在黑线反射面时光电接收管的输出电流为0.1mA,即白纸处光电接管的电流为0.2mA。

则有，在黑线处的电压：

在白纸处的电压：

## 3.3 电机模块

电机模块电路原理图如图3-8所示：

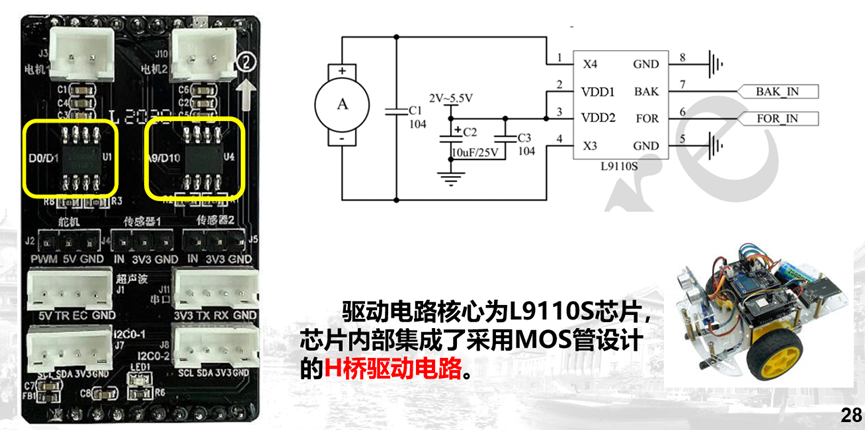


图3-8 电机模块电路原理图

H桥电路是一种常见的电机驱动电路。主要由4个开关管（三极管、MOSFET或晶闸管）组成，其原理如图3-9所示。

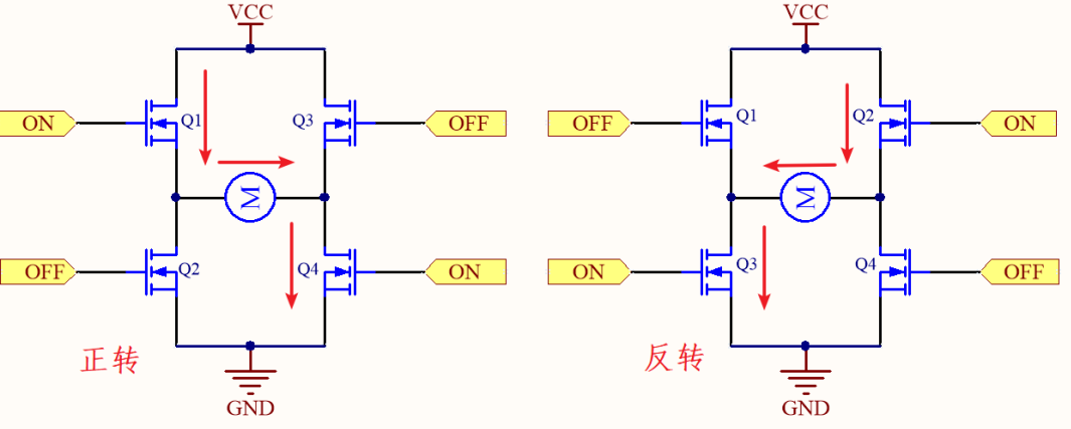


图3-9 H桥原理图

其中，Q1和Q2不能同时导通，Q3和Q4不能同时导通。电机换向控制切换时，需要先关断后开启。比如，要从正转切换到反转，需要先关断Q1和Q4，再开启Q2和Q3。这段先关断后开启所需的时间称为死区时间。示意图如图3-10所示。

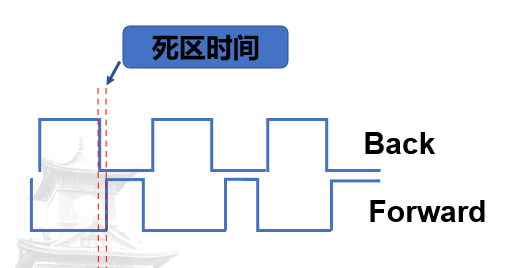


图3-10 死区时间

我们可以通过Hi3861输出PWM波来控制电机的转速，如图3-11所示，电机的转速与供电电压正相关，因此可以通过电压控制转速。

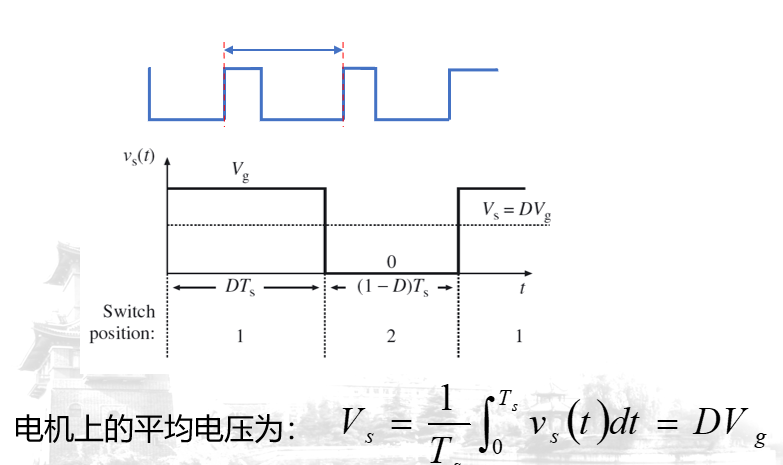


图3-11 PWM调速原理

## 3.4 舵机模块

舵机是一种位置（角度）伺服的驱动器，适用于需要角度不断变化并可以保持的控制系统。其通过电动机、传动部件和离合器组成，其原理如图3-12所示。

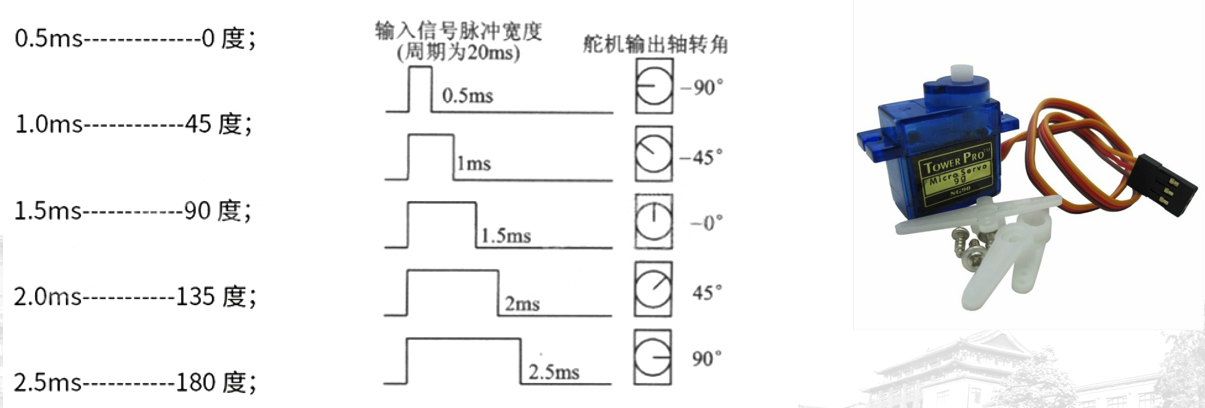


图3-12 舵机原理

# 4 OpenHarmony基础实验-I

注1：在工程目录下的doc目录可找到部分教学资料

注2：在工程目录下的vendor目录下可找到不同厂商为hi3861\_v100编写的例程程序，可供参考。

## 4.1 Hello World

本实验为后续所有实验基础，请认真阅读并完成。后续实验将不再介绍OpenHarmony OS开发的基础操作。

### 4.1.1 实验目的

1.学习OpenHarmony OS设备开发基础概念。

2.掌握OpenHarmony OS基础开发流程。

3.掌握OpenHarmonyOS基本代码结构。

4.学习使用串口输出。

### 4.1.2 实验步骤

1.在DevEco Device Tool中打开项目，如图4-1所示

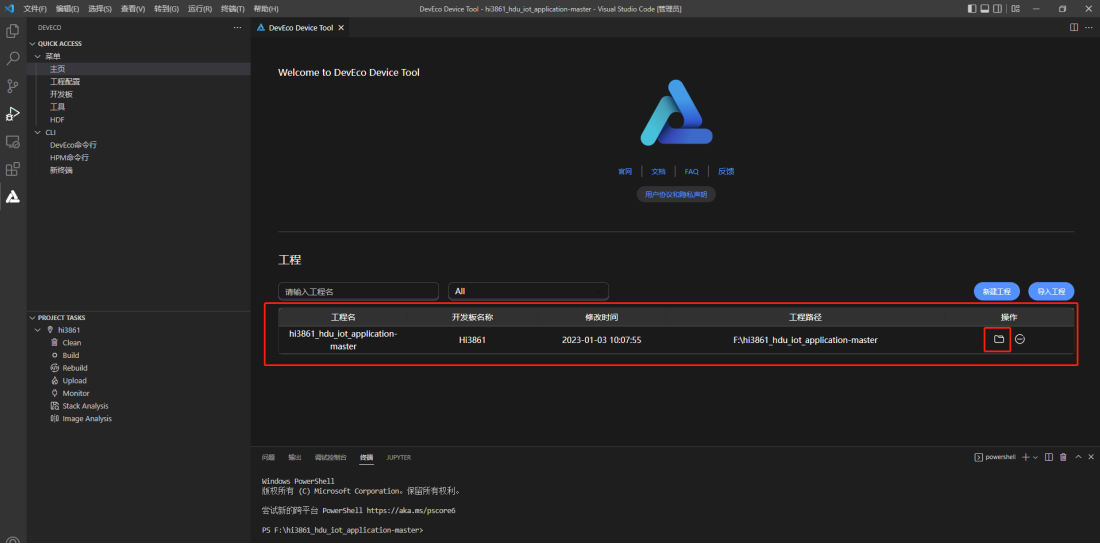


图4-1 打开工程

2. 在“src\applications\sample\wifi-iot\app”下新建“L0\_helloworld”文件夹，如图4-2所示。

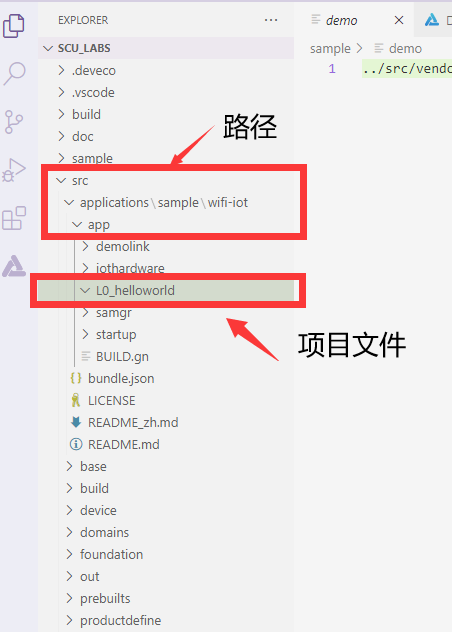


图4-2 新建目录

3. 在2中的“helloworld”文件夹下新建“helloworld\_demo.c”源文件，参照示例代码4.1.4，更改为你想要的输出内容。

4. 在2中的“helloworld”文件夹下创建“BUILD.gn”构建文件，在该文件中添加以下内容：

static\_library("helloworld\_demo") {

sources = [

"helloworld\_demo.c",

]

include\_dirs = [

"//base/iot\_hardware/peripheral/interfaces/kits",

]

}

*# static\_library：将sources中的源文件编译后生成``helloworld\_demo``库文件*

*# sources：实现本工程的功能需要编译的所有.c源文件。*

*# include\_dirs：sources中的所有.c源文件所包含的头文件的路径。*

5. 修改构建工程的“BUILD.gn”：修改“applications\sample\wifi-iot\app”下的“BUILD.gn”，在“feature”中添加“helloworld:helloworld\_demo”，如图4-3所示。



图4-3 BUILD.gn页面

第一个L0\_helloworld指的是需 要编译的工程目录，也就是我们刚才创建的项目文件夹。第二个helloworld\_demo指的是applications/sample/wifi-iot/app/L0\_helloworld/BUILD.gn文件中的静态库，我们刚才把它命名为helloworld\_demo.

至此，代码编写完毕。接着，依次点击build，upload执行编译和烧录操作。

烧录完成后，点击工具栏“Monitor”按钮，启动串口监视功能。然后，按下开发板上的RST键，复位开发板，观察控制台（串口）输出。

### 4.1.3 实验结果

实验预期结果为：可在“Monitor”窗口看见开发日志输出与和“helloworld!”，如图4-4所示。

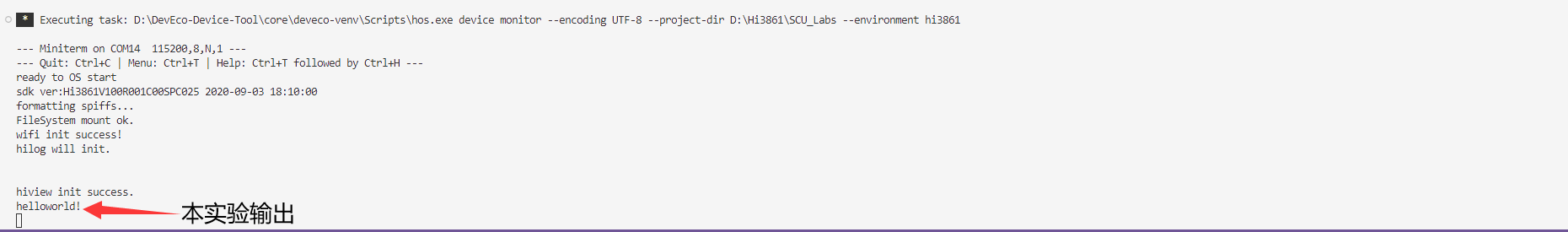


图4-4 实验结果

### 4.1.4 示例代码

#**include** <stdio.h> // *c语言的标准库文件*

#**include** "ohos\_init.h" // *提供用于openharmony初始化和启动服务*

void **HelloWorld**(void)

{

// *打印一下 helloworld!的字符串*

**printf**("helloworld! \r\n");

//*可随意输出*

}

// *APP\_FEATURE\_INIT()是openharmony封装好的函数入口，实现main函数的功能。*

APP\_FEATURE\_INIT(**HelloWorld**);

### 4.1.5 开发流程总结

OpenHarmony OS的设备开发可总结为三步：

1. 新建项目文件夹，便于项目文件管理。

2. 编写业务代码。

3. 编写构建文件“BUILD.gn”

前两步很好理解，就是根据项目需要，编写对应的代码逻辑，以实现预期功能。这两步，在大家的C语言课程上已经讲过了，不再赘述。而第三步-编写构建文件，对部分同学来说可能比较陌生。因此，接下来将对此进行简单介绍。

首先，我们需要引入一个概念-构建系统(build system)。构建系统是用于从源代码(大家编写的.c,.h等文件)生成用户可以使用的目标(targets)的自动化工具。目标可以包括库、可执行文件或脚本等。简单来说，就是把源文件变成程序的一个工具（但不是编译器）。

同学们在编写C语言程序的时候一般都会使用IDE。在编写完成之后，一般只需要点击运行按钮，IDE就会自动帮我们编译运行。那么，IDE是怎么知道我们的程序需要哪些文件来进行编译呢？如果同学们使用过gcc（一个把C文件变成程序的工具，也就是编译器），就会知道想要把一个C文件变成程序，需要我们执行类似以下的命令：

gcc main.c ..\components\helloworld.c --args（一些编译器所需的参数或指令）

也就是说，gcc需要知道待编译文件有哪些。那么，假如我们的项目有几千个源文件，每个源文件分散在各种路径里且不同的源文件还需要编译器执行不同的编译指令，同时每个文件之间还存在引用联系。在这种情况下，人工地、一遍一遍地执行gcc指令显然是不现实的。因此，构建系统就顺理成章地被设计出来解决这种问题。构建系统就是一个自动执行编译器命令的工具，我们只需要写一个很简单的构建系统脚本（比如GNU Make的makefile），就能够实现复杂的编译操作。

但是，当程序规模进一步扩大，尤其是程序需要支持多平台时，编写makefile也变得冗长且复杂。这时，人们又想出一个方法：再造一个能够自动生成makefile的工具。在这些工具里面最出名的就是CMake。程序员在编写完一个超大规模软件之后，只需要再写一个CMake脚本，就能够实现不同平台全自动编译。

那么，“BUILD.gn”又是什么呢？首先，我们要知道的是在大型软件的开发过程中，如果构建（编译）时间过长，将会在一定程度上影响开发效率。Google的程序员觉得这种原因引起的效率降低无法忍受，因此他们开发了一个名叫Ninja的构建系统来代替GNU Make。而BUILD.gn就是Ninja构建系统的脚本（相当于GNU Make里面的makefile）。Open Harmony OS正是采用Ninja的构建系统，因此我们需要编写对应的“BUILD.gn”文件，告诉这个构建系统，我们的程序组织形式，如图4-5所示。

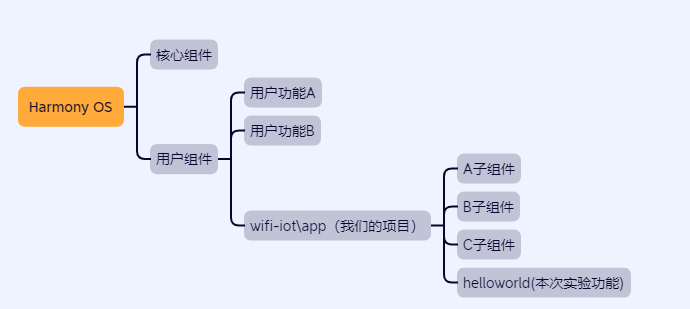


图4-5 程序组织

## 4.2 点亮LED

### 4.2.1 实验目的

1.学习硬件手册的查阅

2.学习使用Hi3861的GPIO，控制开发板上的LED灯，如图4-6所示。

3.学习掌握OpenHarmony OS任务的创建

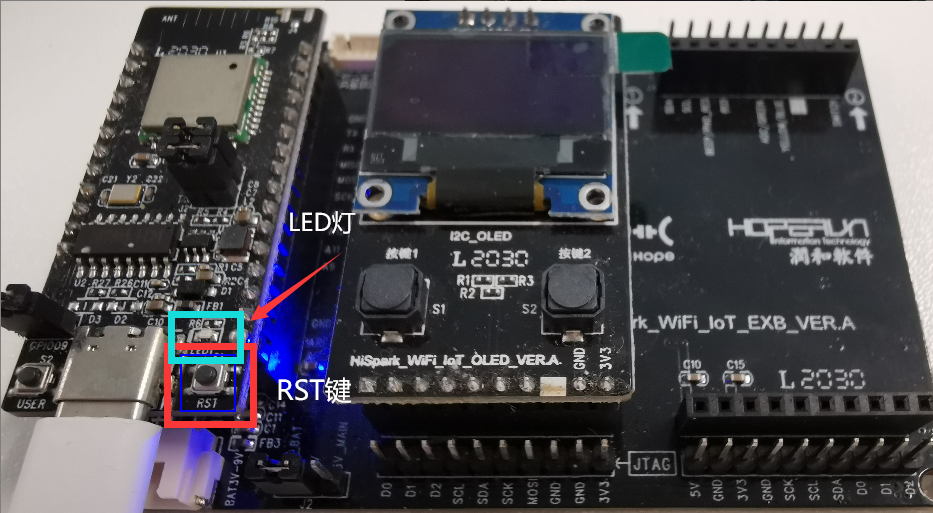


图4-6 LED灯

### 4.2.2 实验步骤

#### 4.2.2.1 Task 1

查阅润和智慧小车开发套件硬件手册，找到与LED灯相连的GPIO口后，更改LED\_GPIO的定义

#### 4.2.2.2 Task 2

根据实验需要，查阅iot\_gpio\_ex.h，获取GPIO功能定义，并为GPIO设置正确的功能。最后，正确设置GPIO的方向（输入，输出，高阻？）

#### 4.2.2.3 Task3

编写blink()函数，控制LED交替闪烁。

#### 4.2.2.4 扩展

1.可以注意到，如果电机此时也与开发板保持连接，电机将会转动。请找出原因。

2.为什么输出高电平LED灯熄灭，低电平LED灯打开？

### 4.2.3 实验结果

实验预期结果为：开发板上的LED灯交替闪烁，周期为1s

### 4.2.4 推荐阅读

1.单片机怎么输出高电平！推挽输出和开漏输出最本质的区别？

2. [中字] RTOS入门 Part 1 -什么是RTOS\_ , P1

3.[第24讲 CMSIS和驱动程序开发（1）\_](https://www.bilibili.com/video/BV1Xv41187Cd?p=24&vd_source=812be8baf0074e4d7fd5491cf54f958f) , P24

4. [Introduction to CMSIS standard in Arm Cortex-M Processor - Inspired Hobbyist](https://inspiredhobbyist.org/introduction-to-cmsis-standard-in-arm-cortex-m-processor/)

## 4.3 超声波模块与电机结合

### 4.3.1 实验目的

1.学习使用Hi3861控制超声波模块

2.学习使用Hi3861控制电机模块

2.学习掌握编写简单的业务逻辑

### 4.3.2 实验步骤

#### 4.3.2.1 Task 1-配置超声波模块

根据硬件手册，实现hcsr04.c中的Hcsr04Init()函数，正确配置超声波模块

#### 4.3.2.2 Task 2-实现测距

学习超声波模块测距原理，实现hcsr04.c中的GetDistance()函数。流程为：

1. 向trig引脚发送一段时间的高电平，启动超声波模块(可用hi\_udelay()实现)

2. 检测Echo引脚高电平持续时间。（可用hi\_get\_us()获取系统时间实现）

3. 根据持续时间，计算距离。（假设声速为340m/s）

#### 4.3.2.3 Task 3-配置电机模块

根据硬件手册，实现motor\_control.c中的MotorInit()函数，正确配置电机模块

#### 4.3.2.4 Task 4-实现前进和刹车

实现motor\_control.c中的CarForward()和CarStop()函数，实现小车前进和刹车

#### 4.3.2.5 Task 5-组合控制

实现main.c中的Hcsr04SSampleTask()函数，实现小车根据测距结果前进或刹车。比如：距离小于某个值就刹车，大于某个值就前进。

#### 4.3.2.6 扩展

1. 超声波测距模块的精度如何？如果距离太近会发生什么？距离太远呢？为什么？

2. 如何实现小车的左转和右转？

3. 这种单个任务轮询的方式会出现什么问题？

### 4.3.3 实验结果

实验预期结果为：

1. 超声波模块在精度范围内正确测距。

2. 小车正确执行前进、刹车功能。

3. 小车电机收到超声波测距模块的正确控制。

### 4.3.4 推荐阅读

1.[超声波测距项目详解！干货全拿走](https://www.bilibili.com/video/BV1Ae4y1F7wj/?spm_id_from=333.337.search-card.all.click&vd_source=812be8baf0074e4d7fd5491cf54f958f)，前10分钟

2. [Mos管的工作原理](https://www.bilibili.com/video/BV1344y167qm/?spm_id_from=333.337.search-card.all.click&vd_source=812be8baf0074e4d7fd5491cf54f958f)

3. [5种方法控制电机正反转](https://www.bilibili.com/video/BV1JR4y1n7AZ/?spm_id_from=333.999.0.0&vd_source=812be8baf0074e4d7fd5491cf54f958f)

4. [H桥的基本原理-刹车-正反转-调速](https://www.bilibili.com/video/BV1ZG4y1v7LS/?spm_id_from=333.337.search-card.all.click&vd_source=812be8baf0074e4d7fd5491cf54f958f)

# 5 OpenHarmony基础实验-II

## 5.1 PWM实现呼吸灯

### 5.1.1 实验目的

1.学习掌握Hi3861的PWM功能

2.实现LED呼吸灯效果。

### 5.1.2 实验步骤

#### 5.1.2.1 Task 1: PWM初始化

查阅硬件手册，找到与实验4.2中LED相连的GPIO口与对应的PWM端口。完成PWMInit()函数，正确配置PWM输出。

在iot\_gpio\_ex.h中可找到GPIO口支持的功能。

使用到的接口在iot\_pwm.h中定义，如下：

unsigned int **IoTPwmInit**(unsigned int port);

unsigned int **IoTPwmDeinit**(unsigned int port);

unsigned int **IoTPwmStart**(unsigned int port, unsigned short duty, unsigned int freq);

unsigned int **IoTPwmStop**(unsigned int port);

具体描述请参照源文件接口描述。

#### 5.1.2.2 Task 2: 实现呼吸灯

实现PWMTask（）函数，实现呼吸灯效果。

在本实验中，应该改变

unsigned int **IoTPwmStart**(unsigned int port, unsigned short duty, unsigned int freq)

中的duty参数，也就是占空比。

### 5.1.3 实验结果

实验预期结果为：开发板上的LED灯实现逐渐由暗到亮，再变成由亮到暗的效果。

### 5.1.4 推荐阅读

1.[搞懂什么是PWM控制](https://www.bilibili.com/video/BV1HD4y1k74L/?spm_id_from=333.337.search-card.all.click&vd_source=812be8baf0074e4d7fd5491cf54f958f)

2. [PWM原理及其应用](https://zhuanlan.zhihu.com/p/379585884)

## 5.2 光通信模块手势识别

### 5.2.1 实验目的

1.学习使用光通信模块

2.学习了解光电二极管的工作原理

2.学习使用Hi3861的ADC功能。

### 5.2.2 实验步骤

#### 5.2.2.1 Task 1-配置ADC

查阅硬件手册，找到支持ADC功能的GPIO接口，与其对应的ADC通道，实现PDInit()，完成配置。

#### 5.2.2.2 Task 2-获取电压值

完成GetVlt()函数，实现读取对应ADC通道的值并转化为对应电压值。

相关接口在hi\_adc.h中定义，可能使用到的有：

hi\_u32 **hi\_adc\_read**(hi\_adc\_channel\_index channel, hi\_u16 \*data, hi\_adc\_equ\_model\_sel equ\_model,

    hi\_adc\_cur\_bais cur\_bais, hi\_u16 delay\_cnt);

hi\_float **hi\_adc\_convert\_to\_voltage**(hi\_u16 data);

完成后，连接光电模块与Hi3861，查看串口输出。

### 5.2.3 实验结果

实验预期结果为：当手与光电二极管之间间隔不同时，可以观察到不同的输出。

### 5.2.4 推荐阅读

1. [第十二期光电二极管](https://www.bilibili.com/video/BV1fs411h7NQ/?spm_id_from=333.337.search-card.all.click&vd_source=812be8baf0074e4d7fd5491cf54f958f)

2. [极其的巧妙设计！模拟转数字信号！5分钟让你看明白！ADC的工作原理](https://www.bilibili.com/video/BV1BV4y1V7nE/?spm_id_from=333.337.search-card.all.click&vd_source=812be8baf0074e4d7fd5491cf54f958f)

3. [逐次逼近型ADC原理](https://www.bilibili.com/video/BV1LN4y1g7yt/?spm_id_from=333.337.search-card.all.click&vd_source=812be8baf0074e4d7fd5491cf54f958f)

4. [(PDF) Ambient Light Based Hand Gesture Recognition Enabled by Recurrent Neural Network (researchgate.net)](https://www.researchgate.net/publication/338332231_Ambient_Light_Based_Hand_Gesture_Recognition_Enabled_by_Recurrent_Neural_Network)

## 5.3 多线程

### 5.3.1 实验目的

1.学习了解多线程

2.学习了解OpenHarmony OS多线程开发

3.学习掌握简单的多线程业务逻辑开发。

### 5.3.2 实验步骤

首先确保5.1和5.2实验的完成。随后，对5.2中的输出ADCTask()做出修改，要求每隔1s输出一次电压。

最后，参照示例代码5.3.4，启动双线程，将PWMTask()和ADCTask()同时启动。

#### 5.3.2.1 扩展

1.线程开的越多越好吗？为什么？

2.线程的优先级的作用是什么？

3.如果多个线程要同时使用同一个资源会发生什么（比如同时控制LED灯的亮灭）？如何解决？

### 5.3.3 实验结果

实验预期结果为：光电模块输出的同时，PWM呼吸灯效果不被打断。

### 5.3.4 示例代码

void **demo**(void)

{

    osThreadAttr\_t attr;

    attr**.**name = "Task1";

    attr**.**attr\_bits = 0U;

    attr**.**cb\_mem = NULL;

    attr**.**cb\_size = 0U;

    attr**.**stack\_mem = NULL;

    attr**.**stack\_size = 2048;

    attr**.**priority = osPriorityNormal;

**if** (**osThreadNew**((osThreadFunc\_t)**Task1**, NULL, &attr) **==** NULL)

    {

**printf**("Falied to create Task!\n");

    }

    attr**.**name = "Tack2";

    attr**.**priority = osPriorityNormal;

**if** (**osThreadNew**((osThreadFunc\_t)**Task2**, NULL, &attr) **==** NULL)

    {

**printf**("Falied to create Task!\n");

    }

}

### 5.3.5 推荐阅读

1. [Processes and Threads (cornell.edu)](https://www.cs.cornell.edu/courses/cs4410/2019su/lectures/03-process-thread.pdf)

2. [线程和进程的区别是什么？](https://www.zhihu.com/question/25532384)

3. [What are CMSIS software components? - Tools, Software and IDEs blog - Arm Community blogs - Arm Community](https://community.arm.com/arm-community-blogs/b/tools-software-ides-blog/posts/what-are-cmsis-software-components)

4. [Thread Management (keil.com)](https://www.keil.com/pack/doc/cmsis/RTOS2/html/group__CMSIS__RTOS__ThreadMgmt.html)

# 6 OpenHarmony 组网

## 6.1 实验目的

1.学习掌握基础的计算机网络知识

2.学习了解不同的WIFI模式

3.学习使用Hi3861的WIFI AP模式

## 6.2 实验步骤

1. 参照鸿蒙系统 Hi3861 实现手机APP配网功能\_连志安的技术博客，理清样例代码。

2. 自定义WiFi相关配置，正确连接Hi3861。

3. 样例代码可通过发送UDP包，控制LED灯的亮灭，同时可回显收到的UDP包的内容。尝试向Hi3861发送UDP包，控制LED灯。

## 6.3 实验结果

实验预期结果为：电脑成功连接Hi3861的WIFI，且能够通过发送UDP包控制Hi3861开发板上的LED灯。

## 6.4 推荐阅读

1. [[野火]LwIP应用开发实战指南—基于野火STM32 文档 (embedfire.com)](https://doc.embedfire.com/net/lwip/zh/latest/README.html),1,2,14节

2.《Computer Networking: A Top-Down Approach》，计算机网络课程教材

3.[JSON 教程 | 菜鸟教程 (runoob.com)](https://www.runoob.com/json/json-tutorial.html)

4. [WiFi学习专栏](https://www.zhihu.com/column/c_1663498272323641344)

# 7 OpenHarmony 上云

## 7.1 实验目的

1.学习了解MQTT

2.学习使用云平台

3.学习使用OpenHarmony OS组件库

4.学习使用OpenHarmony STA模式

5.学习掌握复杂业务逻辑开发

## 7.2 实验步骤

1. 请认真阅读并理清MQTT例程逻辑，例程位于vendor\hisilicon\hispark\_pegasus\demo\oc\_demo下。（样例代码的SLAM可供参考）

2. 修改例程代码，使其适配实验所用的实验套件，实现向云端上报信息（比如LED亮灭状态，ADC采样到的电压灯）

3. 修改例程代码，使其适配实验所用的实验套件，实现云端下发信息控制小车（比如云端开关灯，云端遥控小车等）。

## 7.3 实验结果

实验预期结果为：小车与云端互通，可正常收发信息。

## 7.4 推荐阅读

1. [MQTT Essentials - All Core Concepts explained (hivemq.com)](https://www.hivemq.com/mqtt-essentials/)

## 7.5 小结补充

程序结构图如图7.1所示

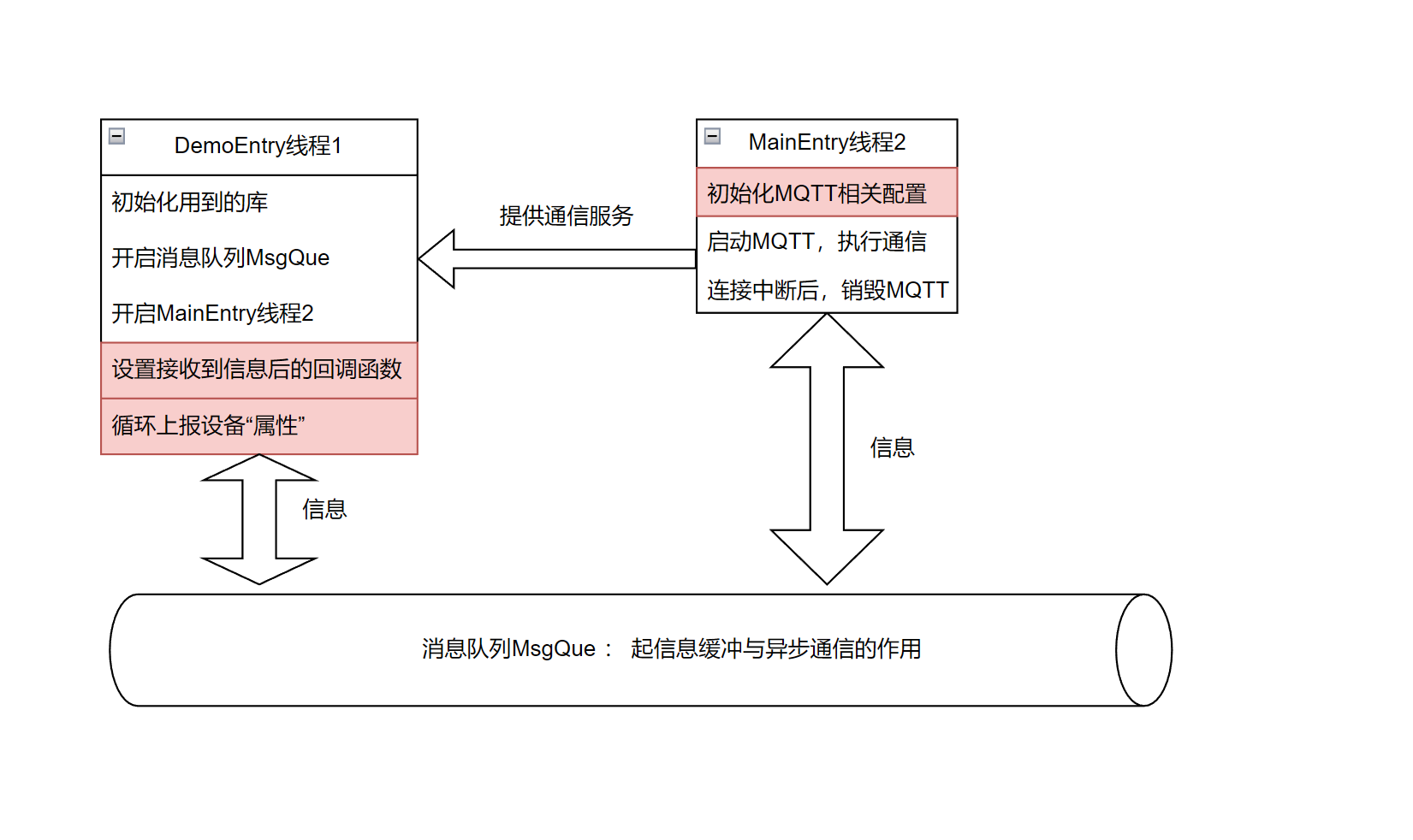


图7-1 程序结构图

现对程序分析如下：

1. SYS\_RUN(AppDemoIot):

程序入口，位于app\_demo\_iot.c​​中。

​ SYS\_RUN()​​在系统启动完毕后开始执行。比APP\_FEATURE\_INIT()​​早，后者是应用层进程启动后执行。

其包含：

1.1 osThreadNew(DemoEntry) (线程1)

位于app\_demo\_iot.c​​

1.2 ConnectToHotspot():

连接WIFI，相关文件为wifi\_connecter​​

1.3 RedLedInit()

初始化GPIO

1.4 CJsonInit()

初始化CJson库

1.5 IoTMain()

1. 开启消息队列，队列长16

2. 开启新线程MainEntry​​​

1.6 IoTSetMsgCallback()

配置接收到信息后的回调函数。云平台传送过来的信息将通过回调函数处理。

回调函数为DemoMsgRcvCallBack()​​，其中用于处理信息的函数为TrafficLightMsgRcvCallBack()​（即我们根据实际需求更改的部分）

1.7 Sleep()

等待3s，连接华为云，之后进入while循环，每隔1s，通过IotPublishSample()​​反复发布LED信息。

2. osThreadNew(MainEntry)(线程2）

位于iot\_main.c​​

其包含：

2.1 MainEntryProcess()

初始化MQTT相关配置，应根据实际情况更改MQTT鉴权的宏定义。

2.2 MqttProcess()

启动MQTT，并执行业务逻辑，如下：

1. **MQTTClient\_create**(),创建MQTT Client

注意MQTTCLIENT\_PERSISTENCE\_NONE，说明Client不使用MQTT的持久连接的特性，也就是说就算工作在QoS2情况下，仍可能会发生信息丢失。

2. **MQTTClient\_setCallbacks**()，创建MQTT回调，一共可设置三个回调函数。

    /\*\*

*\** **@param**handle*: MQTT客户端*

*\** **@param**context*：传递给回调函数的参数，不用可以置为NULL*

*\** **@param**cl*：连接断开之后执行的回调函数，可以置为NULL。本例程中，若断开连接后，会在串口打印对应提示。*

*\** **@param**ma*：接收到信息后执行的回调函数，必须设置。本例程中，接受到信息后，会打印信息。如果消息队列未满，则将信息放入消息队列，等待后续处理；否则，等待空余位置，若等待时间超时，则丢弃这次信息，串口输出错误提示。*

*\** **@param**dc*：信息成功发送后的回调函数，可以置为NULL*

\*\*/

3. **MQTTClient\_connect**()，连接MQTT

4. **MQTTClient\_subscribeMany**()：向broker订阅多个topic

5. 进入**while**()循环

2.2.1 ProcessQueueMsg()

从消息队列中取信息。

2.2.1.1 IoTMsgProcess()

如果能够取出信息，则根据信息执行对应逻辑。根据信息类型的不同，选择进行信息发送或者信息接收。

对于信息接收，执行信息接收回调，即在线程DemoEntry​​中通过IoTSetMsgCallback​​设置的回调函数。

2.2.1.2 MQTTClient\_yield()

维持MQTT心跳

2.3 MQTTClient\_disconnect()

如果MQTT不在线，则断开MQTT连接。

上述内容为整个MQTT例程总览，内容较多，但是如果我们只是使用MQTT的功能的话，只需要关注三点：

1. 订阅哪个topic

2. 收到的信息怎么处理（设置回调）

3. 要发送什么信息

对于第一点，我们需要在iot\_main.c​​的g\_defaultSubscribeTopic[]​​数组中添加自己所需的topic即可，如

**static** **const** char \*g\_defaultSubscribeTopic**[]** = {

    "$oc/devices/"CONFIG\_DEVICE\_ID"/sys/messages/down",//*平台下发消息*

    "$oc/devices/"CONFIG\_DEVICE\_ID"/sys/properties/set/#",//*订阅所有跟设置属性相关的topic*

    "$oc/devices/"CONFIG\_DEVICE\_ID"/sys/properties/get/#",//*订阅所有跟获取属性相关的topic*

    "$oc/devices/"CONFIG\_DEVICE\_ID"/sys/shadow/get/response/#",//*订阅所有用于设备上线后数据同步的topic*

    "$oc/devices/"CONFIG\_DEVICE\_ID"/sys/events/down",//*平台事件下发*

    "$oc/devices/"CONFIG\_DEVICE\_ID"/sys/commands/#"  ,//*订阅所有跟命令相关的topic*

     //*上述topic为华为云MQTT平台默认订阅topic*

     "mytopic/IloveHUAWEI/#",//*自定义订阅，#为通配符,即订阅所有相同前缀的topic*

                        //*如"mytopic/IloveHUAWEI/MATE50"*

                    // *"mytopic/IloveHUAWEI/more/than/APPLE"*

};

后两点，在iot\_main.h​​和iot\_profile.h​​中，已经提供对应接口，前者用于直接发送信息，后者在发送信息的接口上封装了华为云平台所需的响应信息格式，都可以直接使用。可浏览该文件注释，不难看出使用方法。

//*这是我们自定义的线程任务*

void **demo**()

{

**IoTMain**();//*该接口必须首先调用，用于启动MQTT服务*

**IoTSetMsgCallback**(myFunc);//*接口2，用于设置接收信息后的回调函数,具体用法*

                    //*可参考例程的回调函数写法，依葫芦画瓢*

**while**(1)

    {

**IoTSendMsg**(qos,topic,payload);//*向对应topic，发送qos级别的信息payload*

    }

}

## 7.6 华为云MQTT相关

请参阅：

1. [概述\_设备接入 IoTDA\_用户指南\_消息通信\_华为云 (huaweicloud.com)](https://support.huaweicloud.com/usermanual-iothub/iot_01_0045_2.html)

2. [创建产品\_设备接入 IoTDA\_用户指南\_设备管理\_华为云 (huaweicloud.com)](https://support.huaweicloud.com/usermanual-iothub/iot_01_0054.html)

3. [使用前必读\_设备接入 IoTDA\_API参考\_设备侧MQTT/MQTTS接口参考\_华为云 (huaweicloud.com)](https://support.huaweicloud.com/api-iothub/iot_06_v5_3002.html)

4. 鉴权：<https://iot-tool.obs-website.cn-north-4.myhuaweicloud.com/>