

华为认证 IoT 系列教程

HCIP-IoT Developer

华为认证物联网开发高级工程师

实验指导手册

版本:2.5



华为技术有限公司

版权所有 © 华为技术有限公司 2020。保留一切权利。

未经本公司书面许可，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>

华为认证体系介绍

华为认证是华为公司基于“平台+生态”战略，围绕“云-管-端”协同的新ICT技术架构，打造的ICT技术架构认证、平台与服务认证、行业ICT认证三类认证，是业界唯一覆盖ICT（Information and Communications Technology 信息通信技术）全技术领域的认证体系。

根据ICT从业者的学和进阶需求，华为认证分为工程师级别、高级工程师级别和专家级别三个认证等级。华为认证覆盖ICT全领域，符合ICT融合的技术趋势，致力于提供领先的人才培养体系和认证标准，培养数字化时代新型ICT人才，构建良性ICT人才生态。

HCIP-IoT Developer (Huawei Certified ICT Professional-Internet of Things Developer，华为认证高级物联网工程师) 主要面向希望学习华为IoT产品技术人士（物联网相关专业的高校学生或物联网从业者），以及华为公司办事处、代表处一线工程师。HCIP-IoT Developer认证在内容上涵盖华为云物联网平台、华为物联网操作系统Huawei LiteOS、物联网通信技术（无线通信技术与物联网关技术）等内容。

华为认证协助您打开行业之窗，开启改变之门，屹立在IoT网络世界的潮头浪尖！

华为认证架构

Huawei Certification





前言

简介

本书为 HCIP-IoT Developer 认证培训实验教程，适用于准备参加 HCIP-IoT Developer 考试的学员或者希望了解华为云物联网全栈解决方案的读者。

内容描述

本实验指导书共包含 12 个实验和 2 个综合训练，从物联网平台的开发开始，到 Huawei LiteOS 的基础实验，再基于物联网平台进行二次开发，然后进行实践案例，最后进行综合训练。

- 实验一为基于华为云物联网平台进行开发，帮助读者掌握平台的工作机制与原理。
- 实验二为基于 NB-IoT 模组的 AT 指令实验，通过 NB-IoT 网络将终端接入到平台，帮助读者了解 AT 指令，掌握端云互通实践。
- 实验三为 Huawei LiteOS 操作系统内核实验，帮助读者掌握 Huawei LiteOS 的任务使用。
- 实验四为单片机实验，通过控制 LCD 屏幕和 LED 灯闪烁，帮助读者了解开发板工作原理。
- 实验五为基于 NB-IoT 和 WIFI 实现智慧农业案例的端云互通，掌握智慧农业案例开发。
- 实验六为基于 NB-IoT 和 WIFI 实现智慧烟感案例的端云互通，掌握智慧烟感案例开发。
- 实验七为基于 NB-IoT 和 WIFI 实现智慧井盖案例的端云互通，掌握智慧井盖案例开发。
- 实验八为基于 NB-IoT 和 WIFI 实现人体红外的端云互通，掌握人体红外案例开发。
- 实验九为基于华为云 API Explorer 进行物联网平台的接口调用，熟悉平台的二次开发。
- 实验十为物联网平台设备联动，根据光照强度自动实现 LED 灯的开关。
- 实验十一为消息通知服务 SMN，实现邮件通知功能。
- 实验十二 MQTT 开发自动售货机，基于 MQTT 协议，实现自动售货机案例。
- 综合训练一为基于 NB-IoT 和 WIFI 实现智慧物流和智慧路灯案例。
- 综合训练二为基于 MQTT 协议完成智慧农业开发实验。

读者知识背景

本课程为华为物联网高级工程师认证 HCIP-IoT Developer 课程，为了更好地掌握本书内容，阅读本书的读者应首先具备以下基本条件：

- 具备基础的 ICT 知识。

实验环境说明

设备介绍

为了满足 HCIP-IoT Developer 实验需要，建议每套实验环境采用以下配置：

设备名称、型号与版本的对应关系如下：

| 设备名称 | 设备型号 |
|----------|--|
| 物联网开发板套件 | BearPi开发板套件： 1、BearPi-IoT主板 2、通信扩展板：NB-IoT、WIFI 3、案例扩展板：智慧农业、智慧烟感、智慧物流、智慧路灯、智慧井盖、人体红外 |
| 华为物联网平台 | IoTDA服务 |

准备实验环境

检查设备

实验开始之前请每组学员检查自己的实验设备是否齐全，实验清单如下。

| 设备名称 | 数量 | 备注 |
|----------|--------|-------------|
| 笔记本或台式机 | 每组 1 台 | 能够访问公网 |
| 物联网开发板套件 | 每组 1 套 | BearPi 开发套件 |

目录

| | |
|-------------------------------------|-----------|
| 前 言..... | 3 |
| 简介 | 3 |
| 内容描述 | 3 |
| 读者知识背景 | 3 |
| 实验环境说明 | 4 |
| 准备实验环境 | 4 |
| 1 物联网平台开发实验 | 10 |
| 1.1 实验介绍 | 10 |
| 1.1.1 关于本实验 | 10 |
| 1.1.2 实验目的..... | 10 |
| 1.2 实验任务配置步骤 | 10 |
| 1.2.1 登录华为云设备接入服务 IoTDA | 10 |
| 1.2.2 功能定义..... | 12 |
| 1.2.3 插件开发..... | 23 |
| 1.2.4 验证功能定义及插件..... | 48 |
| 1.3 练习题（在同一个产品中完成以下练习） | 55 |
| 1.3.1 完成智慧烟感的平台侧开发 | 55 |
| 1.3.2 完成智慧物流的平台侧开发 | 56 |
| 1.3.3 完成智慧井盖的平台侧开发 | 58 |
| 1.3.4 完成人体感应的平台侧开发 | 59 |
| 1.3.5 基于模拟器完成智慧烟感、物流数据上报及命令响应 | 60 |
| 2 基于 NB-IoT 模组的 AT 指令实验..... | 61 |
| 2.1 实验介绍 | 61 |
| 2.1.1 关于本实验 | 61 |
| 2.1.2 实验目的..... | 61 |
| 2.2 实验任务配置步骤 | 61 |
| 2.2.1 在物联网平台中注册设备 | 61 |
| 2.2.2 使用 AT 指令完成智慧农业的调测..... | 65 |
| 2.2.3 其他常用 AT 指令 | 69 |
| 2.3 练习题 | 69 |



| | |
|--|-----------|
| 2.3.1 使用 AT 指令完成智慧烟感、物流的调测 | 69 |
| 3 Huawei LiteOS 操作系统内核实验..... | 70 |
| 3.1 实验介绍 | 70 |
| 3.1.1 关于本实验 | 70 |
| 3.1.2 实验目的..... | 70 |
| 3.2 实验任务配置步骤 | 70 |
| 3.2.1 打开 LiteOS 工程 | 70 |
| 3.2.2 运行 HelloWorld 任务..... | 71 |
| 3.2.3 任务管理实验..... | 76 |
| 3.2.4 互斥锁实验 | 79 |
| 3.2.5 内存管理实验..... | 83 |
| 3.2.6 信号量实验 | 85 |
| 3.3 练习题 | 88 |
| 3.3.1 改变任务优先级，使 task2 任务优先于 Helloworld 打印..... | 88 |
| 4 单片机基础实验 | 89 |
| 4.1 实验介绍 | 89 |
| 4.1.1 关于本实验 | 89 |
| 4.1.2 实验目的..... | 89 |
| 4.2 实验任务配置步骤 | 89 |
| 4.2.1 板载 LCD 屏幕显示..... | 89 |
| 4.2.2 板载 LED 灯闪烁 | 91 |
| 4.2.3 GPIO 扫描检测板载按键控制 LED..... | 93 |
| 4.2.4 EXIT 检测板载按键控制 LED | 94 |
| 4.3 练习题 | 96 |
| 4.3.1 创建多行 LCD 屏幕显示..... | 96 |
| 4.3.2 创建多种颜色的 LCD 屏幕显示..... | 96 |
| 4.3.3 在控制台打印 LED 灯的开关状态 | 96 |
| 4.3.4 在 LCD 屏幕显示 LED 灯的开关状态..... | 96 |
| 5 基于 NB-IoT 和 WIFI 的智慧农业实验 | 97 |
| 5.1 实验介绍 | 97 |
| 5.1.1 关于本实验 | 97 |
| 5.1.2 实验目的..... | 97 |
| 5.2 实验任务配置步骤 | 97 |
| 5.2.1 配置智慧农业案例..... | 97 |

| | |
|---|------------|
| 5.2.2 创建智慧农业所需的数据结构 | 99 |
| 5.2.3 创建数据收集任务 | 104 |
| 5.2.4 创建数据上报任务 | 107 |
| 5.2.5 创建命令响应任务 | 110 |
| 5.2.6 使用 WIFI 通信方式 | 115 |
| 5.3 练习题 | 120 |
| 5.3.1 下发 Light 和 Motor 所有开关命令 | 120 |
| 6 基于 NB-IoT 和 WIFI 的智慧烟感实验 | 121 |
| 6.1 实验介绍 | 121 |
| 6.1.1 关于本实验 | 121 |
| 6.1.2 实验目的 | 121 |
| 6.2 实验任务配置步骤 | 121 |
| 6.2.1 配置智慧烟感案例 | 121 |
| 6.2.2 创建智慧烟感所需的数据结构 | 124 |
| 6.2.3 创建数据收集任务 | 126 |
| 6.2.4 创建数据上报任务 | 129 |
| 6.2.5 创建命令响应任务 | 132 |
| 6.2.6 使用 WIFI 通信方式 | 135 |
| 6.3 练习题 | 139 |
| 6.3.1 使用按键控制蜂鸣器 | 139 |
| 7 基于 NB-IoT 和 WIFI 的智慧井盖实验 | 140 |
| 7.1 实验介绍 | 140 |
| 7.1.1 关于本实验 | 140 |
| 7.1.2 实验目的 | 140 |
| 7.2 实验任务配置步骤 | 140 |
| 7.2.1 配置智慧井盖案例 | 140 |
| 7.2.2 创建智慧井盖所需的数据结构 | 143 |
| 7.2.3 创建数据收集任务 | 145 |
| 7.2.4 创建数据上报任务 | 149 |
| 7.3 练习题 | 152 |
| 7.3.1 使用 WIFI 通信方式实现智慧井盖案例 | 152 |
| 8 基于 NB-IoT 和 WIFI 的人体感应实验 | 153 |
| 8.1 实验介绍 | 153 |
| 8.1.1 关于本实验 | 153 |



| | |
|-------------------------------------|------------|
| 8.1.2 实验目的..... | 153 |
| 8.2 实验任务配置步骤 | 153 |
| 8.2.1 配置人体感应案例..... | 153 |
| 8.2.2 创建人体感应所需的数据结构 | 155 |
| 8.2.3 创建数据收集任务 | 158 |
| 8.2.4 创建数据上报任务 | 161 |
| 8.3 练习题 | 163 |
| 8.3.1 使用 WIFI 通信方式实现人体感应案例 | 163 |
| 9 综合训练 1..... | 164 |
| 9.1 实验介绍 | 164 |
| 9.1.1 关于本实验 | 164 |
| 9.1.2 实验目的..... | 164 |
| 9.2 实验任务配置步骤 | 164 |
| 9.2.1 基于 NB-IoT 和 WIFI 的智慧物流实验..... | 164 |
| 9.2.2 基于 NB-IoT 实现智慧路灯案例 | 164 |
| 9.2.3 使用按键选择案例运行 | 165 |
| 9.3 结果验证 | 165 |
| 10 物联网平台接口调用实验..... | 166 |
| 10.1 实验介绍..... | 166 |
| 10.1.1 关于本实验..... | 166 |
| 10.1.2 实验目的 | 166 |
| 10.2 实验任务配置步骤 | 166 |
| 10.2.1 登录华为云 API Explorer 服务 | 167 |
| 10.2.2 调用删除设备接口..... | 168 |
| 10.2.3 调用创建设备接口..... | 169 |
| 10.2.4 调用查询设备影子接口..... | 172 |
| 10.3 练习题 | 174 |
| 10.3.1 调用接口创建带有设备名称的设备 | 174 |
| 10.3.2 上报智慧烟感、物流数据，查看响应结果..... | 174 |
| 11 物联网平台设备联动实验..... | 175 |
| 11.1 实验介绍..... | 175 |
| 11.1.1 关于本实验..... | 175 |
| 11.1.2 实验目的 | 175 |
| 11.2 实验任务配置步骤 | 175 |



| | |
|--|------------|
| 11.2.1 实现根据光照强度自动控制 LED 灯开 | 175 |
| 11.3 练习题 | 179 |
| 11.3.1 创建规则，当光照强度大于 500 时，关闭 LED 灯 | 179 |
| 11.3.2 设置触发机制，进行无效触发抑制，时效 20s | 179 |
| 12 消息通知服务 SMN 实验 | 180 |
| 12.1 实验介绍 | 180 |
| 12.1.1 关于本实验 | 180 |
| 12.1.2 实验目的 | 180 |
| 12.2 实验任务配置步骤 | 180 |
| 12.2.1 配置华为云 SMN 服务 | 180 |
| 12.3 练习题 | 184 |
| 12.3.1 添加邮件通知功能 | 184 |
| 13 MQTT 开发自动售货机实验 | 185 |
| 13.1 实验介绍 | 185 |
| 13.1.1 关于本实验 | 185 |
| 13.1.2 实验目的 | 185 |
| 13.2 实验任务配置步骤 | 185 |
| 13.2.1 功能定义 | 185 |
| 13.2.2 自动售货机商品显示实验 | 186 |
| 13.2.3 商品选择实验 | 197 |
| 13.2.4 上报数据到平台实验 | 199 |
| 13.2.5 下发命令实验 | 211 |
| 13.3 练习题 | 213 |
| 13.3.1 添加售货机颜色字段进行上报 | 213 |
| 14 综合训练 2 | 214 |
| 14.1 实验介绍 | 214 |
| 14.1.1 关于本实验 | 214 |
| 14.1.2 实验目的 | 214 |
| 14.2 实验任务配置步骤 | 214 |
| 14.2.1 基于 MQTT 协议的智慧农业案例实验 | 214 |
| 14.3 结果验证 | 214 |

1

物联网平台开发实验

1.1 实验介绍

1.1.1 关于本实验

本实验通过在华为云物联网平台上创建产品，进行案例的功能定义和编解码插件开发，掌握物联网平台的操作流程，以及如何验证编解码插件是否正确。

1.1.2 实验目的

- 掌握物联网平台的功能定义
- 掌握物联网平台编解码插件的开发
- 掌握物联网平台的调试

1.2 实验任务配置步骤

1.2.1 登录华为云设备接入服务 IoTDA

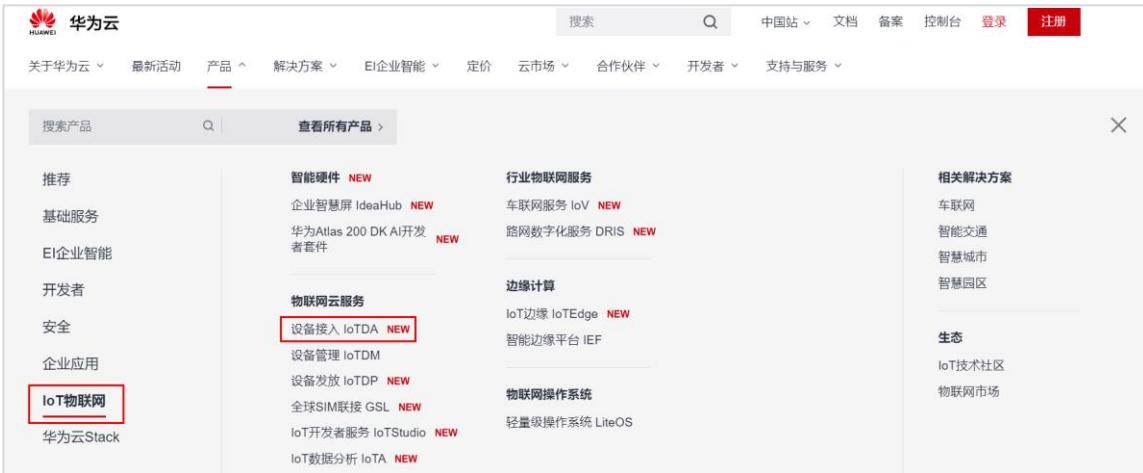
步骤 1 登录华为云

登录华为云官网：<https://www.huaweicloud.com/>



步骤 2 进入设备接入服务

选择“产品”->“IoT 物联网”->“设备接入 IoTDA”；



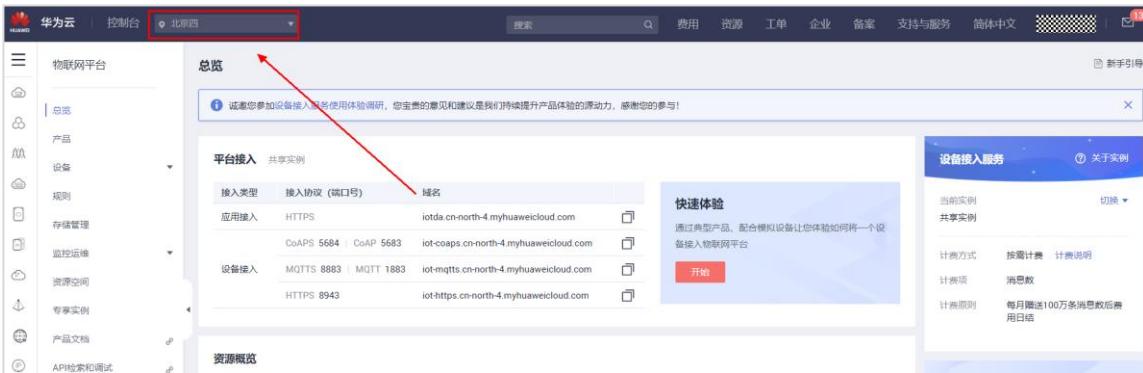
The screenshot shows the Huawei Cloud IoT homepage. The left sidebar has a red box around the 'IoT物联网' (IoT) category. The main content area has a red box around the '设备接入 IoTDA NEW' (Device Access IoTDA NEW) link under the '物联网云服务' (IoT Cloud Services) section. To the right, there's a sidebar with '相关解决方案' (Related Solutions) like '车联网' (V2X), '智能交通' (Smart Transportation), etc.

点击“立即使用”；



The screenshot shows the '设备接入 IoTDA' (Device Access IoTDA) product page. It features a large blue hexagonal graphic with three blue cylinders. Below it, there's a red box around the '立即使用' (Use Now) button. The page also includes a brief description and a '购买专享版' (Purchase Exclusive Edition) button.

确认控制台为“北京四”；



The screenshot shows the IoT Platform Control Console. A red arrow points to the top-left corner where the text '北京四' is displayed. The interface includes a sidebar with various service icons and a main dashboard with tabs like '总览' (Overview), '平台接入' (Platform Access), and '快速体验' (Quick Experience). A red box highlights the '开始' (Start) button in the quick experience section.

1.2.2 功能定义

步骤 1 智慧农业案例功能设计思路

表1-1 产品信息

| 属性 | 属性值 |
|------|------------|
| 产品名称 | 自定义 |
| 协议类型 | LwM2M/CoAP |
| 数据格式 | 二进制码流 |
| 厂商名称 | 自定义 |
| 所属行业 | 智慧农业 |
| 设备类型 | 自定义 |

表1-2 智慧农业服务

| 服务描述 | 服务名称(service_id) | 扩展板 |
|--------------|------------------|------|
| 检测实时温湿度和光照强度 | Agriculture | 智慧农业 |

表1-3 智慧农业属性

| 能力描述 | 属性名称 | 数据类型 | 访问权限 | 数据范围 |
|------|-------------|------|-----------|---------|
| 属性列表 | Temperature | int | 可读、可写、可执行 | 0~65535 |
| | Humidity | int | 可读、可写、可执行 | 0~65535 |
| | Luminance | int | 可读、可写、可执行 | 0~65535 |

表1-4 智慧农业命令

| 命令名称 | 参数 | 参数名称 | 数据类型 | 数据范围/长度 | 枚举值 |
|---------------------------|------|-------------|--------|---------|--------|
| Agriculture_Control_Light | 下发参数 | Light | string | 3 | ON,OFF |
| | 响应参数 | Light_State | int | 0~1 | -- |
| Agriculture_Control_Motor | 下发参数 | Motor | string | 3 | ON,OFF |
| | 响应参数 | Motor_State | int | 0~1 | -- |

步骤 2 创建产品

点击“产品”->“创建产品”；

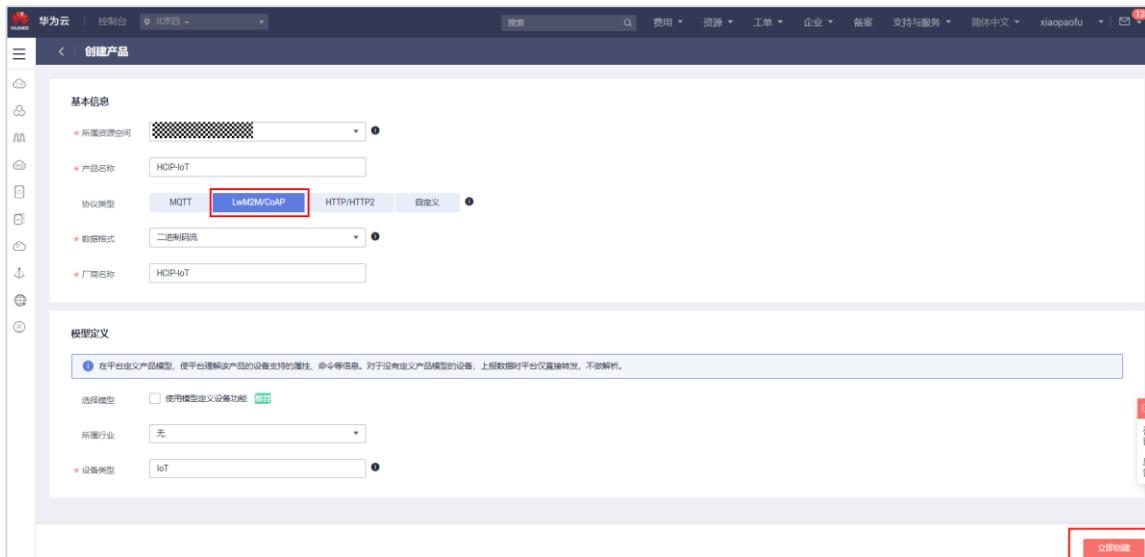


The screenshot shows the IoT Platform Control Console interface. On the left sidebar, '产品' (Product) is selected. At the top right, there is a red box around the '创建产品' (Create Product) button. Below the sidebar, there is a diagram illustrating the product creation process:

1. 定义Profile: A product profile is defined.
2. 注册设备: The device is registered.
3. 设备侧开发: Device-side development is performed.
4. 在线调试: Online debugging is conducted.

The main area shows a table with columns: 产品名称 (Product Name), 产品ID (Product ID), 设备类型 (Device Type), 协议类型 (Protocol Type), and 操作 (Operations). A dropdown menu shows 'DefaultApp_HCNAlot_iot'.

根据设计思路填写信息，点击右下角“立即创建”；注意协议类型选择“LwM2M/CoAP”；



The screenshot shows the 'Create Product' form. In the '基本信息' (Basic Information) section, the '协议类型' (Protocol Type) field has 'LwM2M/CoAP' selected, which is highlighted with a red box. Other fields include '所属资源空间' (Resource Space), '产品名称' (Product Name), '数据格式' (Data Format), and '厂商名称' (Manufacturer Name).

In the '模型定义' (Model Definition) section, there are fields for '选择模型' (Select Model), '所属行业' (Industry), and '设备类型' (Device Type). A note at the top says: '在平台定义产品模型，使平台理解该产品的设备支持的属性、命令等信息。对于没有定义产品模型的设备，上报数据时平台仅直接转发，不做解析。' (Define the product model in the platform to make the platform understand the device's supported properties, commands, etc. For devices without defined product models, data reporting will be forwarded directly by the platform without parsing.)

At the bottom right of the form, there is a red box around the '立即创建' (Create Now) button.

提示创建产品成功，点击“确认”。

步骤 3 添加 Agriculture 服务

点击产品名称“HCIP-IoT”；



The screenshot shows the IoT Platform Control Console interface with the '产品' (Product) selected in the sidebar. The main area displays the 'HCIP-IoT' product details. The '产品名称' (Product Name) field is highlighted with a red box and contains 'HCIP-IoT'. The table below shows the product configuration:

| 产品名称 | 产品ID | 设备类型 | 协议类型 | 操作 |
|----------|------------|------|------------|---------|
| HCIP-IoT | XXXXXXXXXX | IoT | LwM2M/CoAP | 详情 编辑 |

点击“功能定义”->“自定义功能”；



根据设计思路，输入“服务名称”和“服务描述”，点击“确认”；



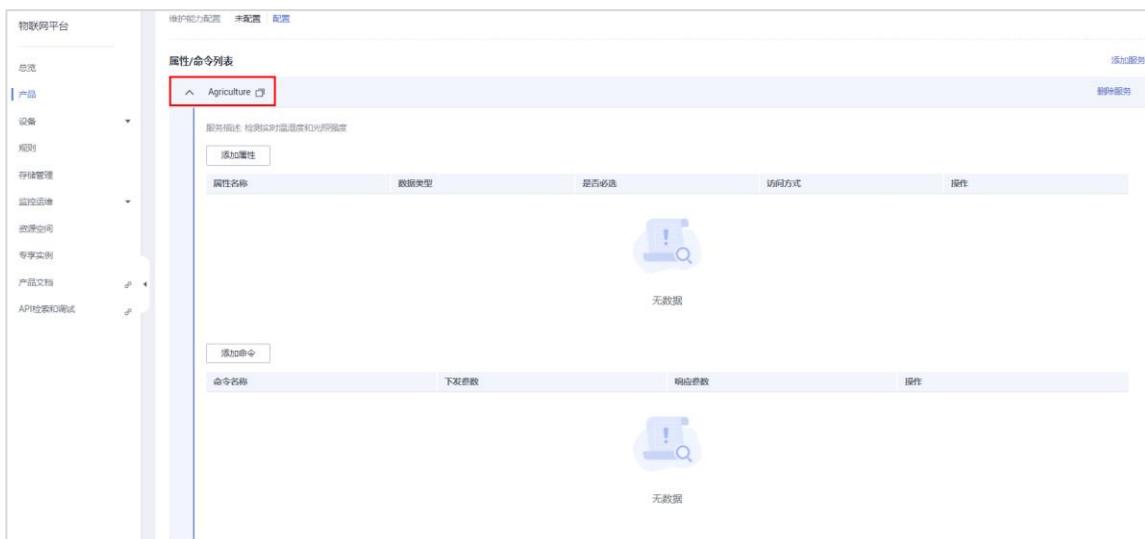
新增服务

服务名称: Agriculture

服务描述: 检测实时温湿度和光照强度
12/1024

确认 取消

点击服务名称“Agriculture”，展开属性和命令；



物联网平台

维护能力配置 未配置 配置

属性/命令列表

Agriculture

| 属性名称 | 数据类型 | 是否必选 | 访问方式 | 操作 |
|------|------|------|------|----|
| 无数据 | | | | |

| 命令名称 | 下发参数 | 响应参数 | 操作 |
|------|------|------|----|
| 无数据 | | | |

步骤 4 添加属性

点击“添加属性”；



The screenshot shows a table for managing attributes. The columns are: 属性名称 (Attribute Name), 数据类型 (Data Type), 是否必选 (Is Required), 访问方式 (Access Method), and 操作 (Operations). A search icon is visible above the table, and a message '无数据' (No Data) is displayed below it.

根据设计思路，添加“Temperature”属性，数据类型“int”，访问权限“可读、可写、可执行”，点击“确认”；

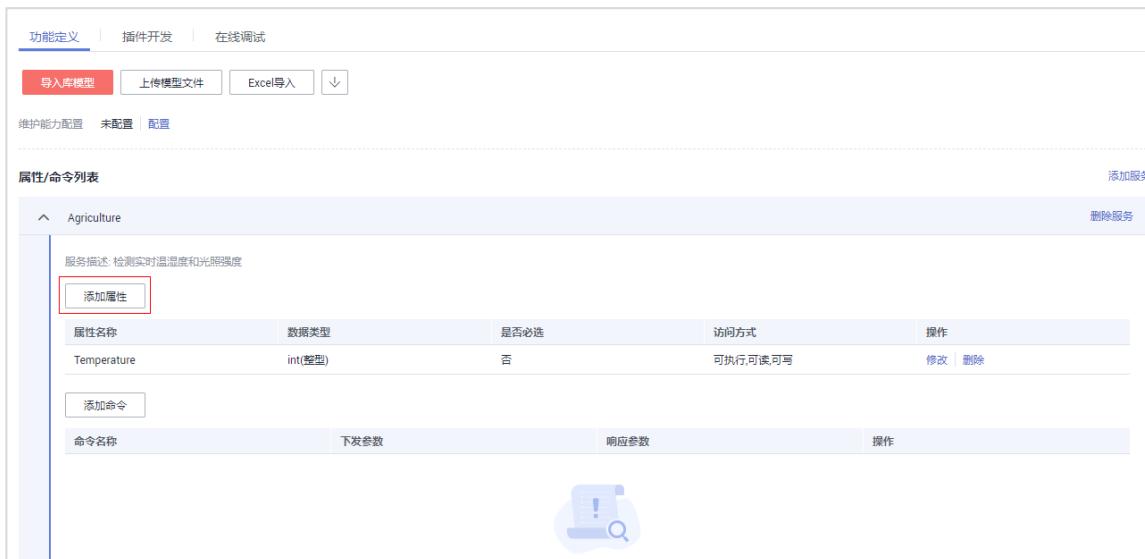


The dialog box contains the following fields:

- 属性名称: Temperature
- 数据类型: int(整型)
- 访问权限: 可读, 可写, 可执行
- 取值范围: 0 - 65535
- 步长: (empty)
- 单位: (empty)

Buttons at the bottom: 确认 (Confirm) and 取消 (Cancel).

继续添加属性；



The screenshot shows the same table as before, now with one row for the 'Temperature' attribute. The table columns are: 属性名称 (Attribute Name), 数据类型 (Data Type), 是否必选 (Is Required), 访问方式 (Access Method), and 操作 (Operations). The 'Temperature' row has the value 'int(整型)' in the Data Type column, '否' (No) in the Is Required column, and '可执行,可读,可写' (Execute, Read, Write) in the Access Method column. The '操作' (Operations) column shows '修改' (Modify) and '删除' (Delete).

根据设计思路，添加“Humidity”属性，点击“确认”；

新增属性

| | | |
|--------|-----------|-----------------------------|
| ★ 属性名称 | Humidity | <input type="checkbox"/> 必选 |
| ★ 数据类型 | int(整型) | ▼ |
| ★ 访问权限 | 可读 可写 可执行 | |
| ★ 取值范围 | 0 | - 65535 |
| 步长 | | |
| 单位 | | |

确认 **取消**

根据设计思路，添加“Luminance”属性，点击“确认”；

新增属性

| | | |
|--------|-----------|-----------------------------|
| ★ 属性名称 | Luminance | <input type="checkbox"/> 必选 |
| ★ 数据类型 | int(整型) | ▼ |
| ★ 访问权限 | 可读 可写 可执行 | |
| ★ 取值范围 | 0 | - 65535 |
| 步长 | | |
| 单位 | | |

确认 **取消**

步骤 5 添加命令

点击“添加命令”；



The screenshot shows a table with three rows: Temperature (int), Humidity (int), and Luminance (int). Below the table is a section labeled 'Add Command' with a red box around it. At the bottom of this section is a 'Command Name' input field.

根据设计思路，输入命令名称“Agriculture_Control_Light”，点击“新增输入参数”；



The dialog box has a 'Command Name' field containing 'Agriculture_Control_Light'. Below it is a 'Add Input Parameter' button with a red box around it. At the bottom is a table for defining parameters.

根据设计思路，新增“Light”参数，点击“确认”；



The dialog box shows parameter details: 'Parameter Name' is 'Light', 'Data Type' is 'string(字符串)', 'Length' is '3', and 'Enum Value' is 'ON,OFF'. At the bottom are 'Confirm' and 'Cancel' buttons.

点击“新增输出参数”；



The screenshot shows the 'Add Command' dialog box. In the '下发参数' (Downstream Parameters) section, there is a table with one row: Light (string type, optional). In the '响应参数' (Response Parameters) section, there is a button labeled '新增输出参数' (Add Output Parameter) which is highlighted with a red box. Below this button is another table where the first column is '参数名称' (Parameter Name) and the second column is '数据类型' (Data Type).

根据设计思路，新增“Light_State”参数，点击“确认”；



The screenshot shows the 'Add Parameter' dialog box. It has fields for '参数名称' (Parameter Name) set to 'Light_State', '数据类型' (Data Type) set to 'int(整型)', '取值范围' (Value Range) set to '0 - 65535', and '步长' (Step) and '单位' (Unit) both empty. At the bottom are '确认' (Confirm) and '取消' (Cancel) buttons.

“Agriculture_Control_Light”命令新增完成，点击“确认”；

新增命令

★ 命令名称

下发参数 新增输入参数

| 参数名称 | 数据类型 | 是否必选 | 操作 |
|-------|-------------|------|--|
| Light | string(字符串) | 否 | 修改 删除 |

响应参数 新增输出参数

| 参数名称 | 数据类型 | 是否必选 | 操作 |
|-------------|---------|------|--|
| Light_State | int(整型) | 否 | 修改 删除 |

确认 取消

点击“添加命令”；

功能定义 | 插件开发 | 在线调试

导入库模型 上传模型文件 Excel导入 ↓

维护能力配置 未配置 | 配置

属性/命令列表 添加服务 删除服务

▲ Agriculture 属性 操作

服务描述: 检测实时温湿度和光照强度

添加属性 属性名称 数据类型 是否必选 访问方式 操作

| | | | | |
|-------------|---------|---|-----------|--|
| Temperature | int(整型) | 否 | 可执行,可读,可写 | 修改 删除 |
| Humidity | int(整型) | 否 | 可执行,可读,可写 | 修改 删除 |
| Luminance | int(整型) | 否 | 可执行,可读,可写 | 修改 删除 |

添加命令 命令名称 下发参数 响应参数 操作

| | | | |
|---------------------------|-------|-------------|--|
| Agriculture_Control_Light | Light | Light_State | 修改 删除 |
|---------------------------|-------|-------------|--|

根据设计思路，输入命令名称“Agriculture_Control_Motor”，点击“新增输入参数”；



新增命令

* 命令名称: Agriculture_Control_Motor

下发参数: 新增输入参数

| 参数名称 | 数据类型 | 是否必选 | 操作 |
|------|------|------|----|
| 无数据 | | | |

响应参数: 新增输出参数

| 参数名称 | 数据类型 | 是否必选 | 操作 |
|------|------|------|----|
| 无数据 | | | |

根据设计思路，新增“Motor”参数，点击“确认”；



新增参数

* 参数名称: Motor 必选

* 数据类型: string(字符串)

* 长度: 3

枚举值: ON,OFF
6/1024

确认 取消

点击“新增输出参数”；



根据设计思路，新增“Motor_State”参数，点击“确认”；



“Agriculture_Motor_Light”命令新增完成，点击“确认”；

新增命令

* 命令名称: Agriculture_Control_Motor

下发参数:

| 参数名称 | 数据类型 | 是否必选 | 操作 |
|-------|-------------|------|---------|
| Motor | string(字符串) | 否 | 修改 删除 |

响应参数:

| 参数名称 | 数据类型 | 是否必选 | 操作 |
|-------------|---------|------|---------|
| Motor_State | int(整型) | 否 | 修改 删除 |

确认 **取消**

智慧农业服务 Agriculture 功能定义完成；

功能定义 | 插件开发 | 在线调试

导入模型 | 上传模型文件 | Excel导入 | 下载

维护能力配置 | 未配置 | 配置

属性/命令列表

Agriculture

添加服务 | 删服务

服务描述: 检测实时温湿度和光照强度

添加属性

| 属性名称 | 数据类型 | 是否必选 | 访问方式 | 操作 |
|-------------|---------|------|-----------|---------|
| Temperature | int(整型) | 否 | 可执行,可读,可写 | 修改 删除 |
| Humidity | int(整型) | 否 | 可执行,可读,可写 | 修改 删除 |
| Luminance | int(整型) | 否 | 可执行,可读,可写 | 修改 删除 |

添加命令

| 命令名称 | 下发参数 | 响应参数 | 操作 |
|---------------------------|-------|-------------|---------|
| Agriculture_Control_Light | Light | Light_State | 修改 删除 |
| Agriculture_Control_Motor | Motor | Motor_State | 修改 删除 |

1.2.3 插件开发

步骤 1 智慧农业案例插件设计思路

表1-5 消息列表

| | 消息名 | 消息类型 | messageld |
|---|---------------------------|------|-----------|
| 1 | Agriculture | 数据上报 | 00 |
| 2 | Agriculture_Control_Light | 命令下发 | 01 |
| | | 响应 | 02 |
| 3 | Agriculture_Control_Motor | 命令下发 | 03 |
| | | 响应 | 04 |

表1-6 消息 Agriculture

| 码流偏移值 | 0 | 1 | 2 | 3 | 4 |
|--------|-----------|-------------|----------|-----------|----|
| 字段名 | messageld | Temperature | Humidity | Luminance | |
| 数据类型 | int8u | int8u | int8u | int16u | |
| 长度 | 1 | 1 | 1 | 2 | |
| 16进制码流 | 00 | 19 | 3C | 00 | 64 |

表1-7 消息 Agriculture_Control_Light

| 码流偏移值 | 0 | 1 | 2 | 3 | 4 | 5 | |
|---------|-----------|--------|----|---------|-------------|----|--|
| 命令下发字段名 | messageld | mid | | Light | | | |
| 数据类型 | int8u | int16u | | string | | | |
| 长度 | 1 | 2 | | 3 | | | |
| 16进制码流 | 01 | 00 | 01 | 4F | 4E | -- | |
| | | | | 4F | 46 | 46 | |
| 命令响应字段名 | messageld | mid | | errcode | Light_State | -- | |
| 数据类型 | int8u | int16u | | int8u | int8u | -- | |
| 长度 | 1 | 2 | | 1 | 1 | -- | |
| 16进制码流 | 02 | 00 | 01 | 00/01 | 00/01 | -- | |

表1-8 消息 Agriculture_Control_Motor

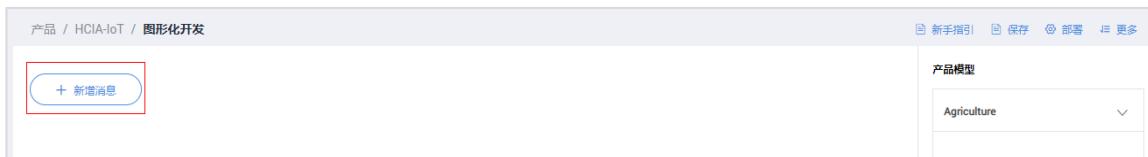
| 码流偏移值 | 0 | 1 | 2 | 3 | 4 | 5 | |
|---------|-----------|--------|----|---------|--------------|----|--|
| 命令下发字段名 | messageld | mid | | Motor | | | |
| 数据类型 | int8u | int16u | | string | | | |
| 长度 | 1 | 2 | | 3 | | | |
| 16进制码流 | 03 | 00 | 01 | 4F | 4E | -- | |
| | | | | 4F | 46 | 46 | |
| 命令响应字段名 | messageld | mid | | errcode | Motor_Status | -- | |
| 数据类型 | int8u | int16u | | int8u | int8u | -- | |
| 长度 | 1 | 2 | | 1 | 1 | -- | |
| 16进制码流 | 04 | 00 | 01 | 00/01 | 00/01 | -- | |

步骤 2 新增数据上报消息

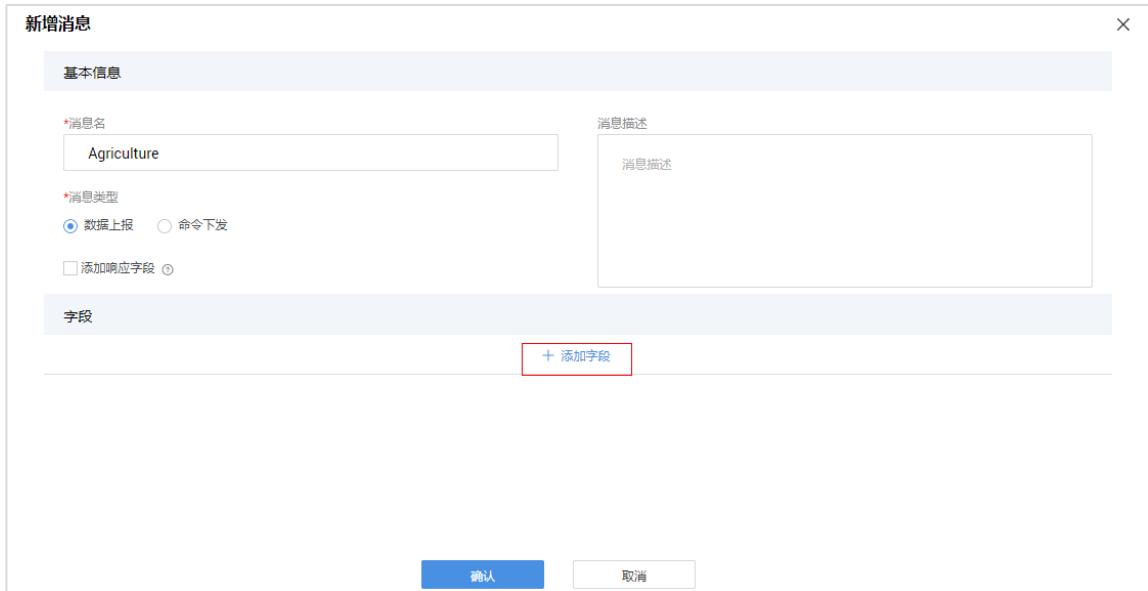
点击“插件开发”->“图形化开发”->“图形化开发”；



点击“新增消息”；



根据设计思路，输入消息名“Agriculture”，消息类型选择“数据上报”，点击“添加字段”；



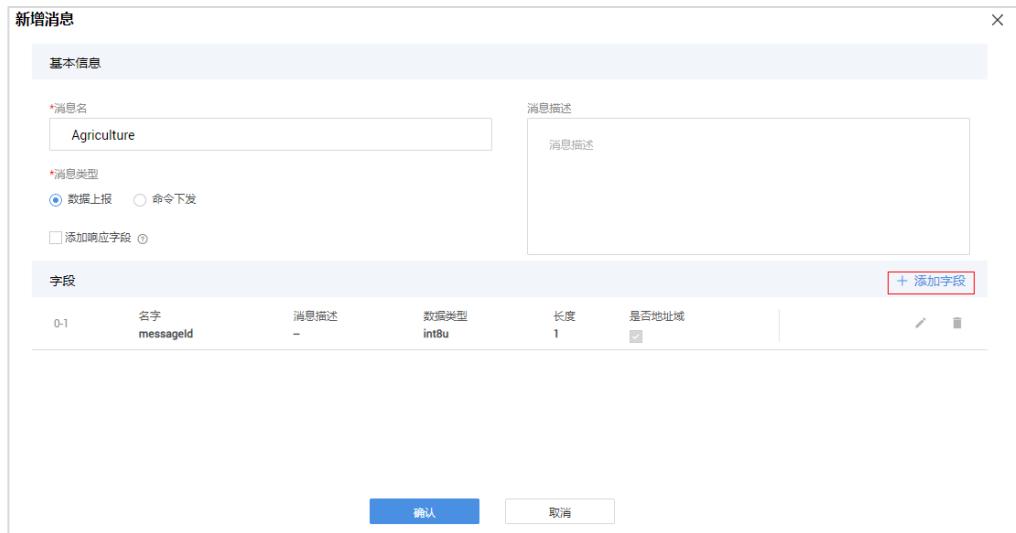
The screenshot shows the '新增消息' (Add Message) dialog box. In the '基本信息' (Basic Information) tab, the '消息名' (Message Name) is set to 'Agriculture', '消息类型' (Message Type) is selected as '数据上报' (Data Report), and the '添加响应字段' (Add Response Field) checkbox is unchecked. The '消息描述' (Message Description) field is empty. In the '字段' (Fields) tab, there is a red-bordered '添加字段' (Add Field) button. At the bottom, there are '确认' (Confirm) and '取消' (Cancel) buttons.

勾选“标记为地址域”，其他默认，点击“确认”；



The screenshot shows the '添加字段' (Add Field) dialog box. Under the '描述' (Description) tab, the '名字' (Name) is set to 'messageld' and the '输入字段描述' (Input Field Description) is empty. Under the '数据类型 (大端模式)' (Data Type (Big-endian Mode)) tab, the type is 'int8u'. The '长度' (Length) is set to '1', '默认值' (Default Value) is '0x0', and '偏移值' (Offset) is '0-1'. At the bottom, there are '确认' (Confirm) and '取消' (Cancel) buttons.

点击“添加字段”；



The screenshot shows the 'Add Message' dialog box. In the 'Basic Information' tab, the message name is set to 'Agriculture', the message type is 'Data Report' (selected), and there is a table below with one row:

| 字段 | 名字 | 消息描述 | 数据类型 | 长度 | 是否地址域 |
|-----|-----------|------|-------|----|-------------------------------------|
| 0-1 | messageId | - | int8u | 1 | <input checked="" type="checkbox"/> |

A red box highlights the '+ 添加字段' button at the top right of the table.

根据设计思路，输入字段名字“Temperature”，点击“确认”；



The screenshot shows the 'Add Field' dialog box. The 'Name' field contains 'Temperture'. Other fields include:

- Checkboxes for 'Address Domain' and 'Default Value' (both are unchecked).
- 'Description' field: '输入字段描述'.
- 'Data Type (Big-endian mode)': 'int8u'.
- 'Length': '1'.
- 'Default Value' field: empty.
- 'Offset': '1-2'.

A red box highlights the '确认' (Confirm) button at the bottom left.

点击“添加字段”；



The screenshot shows the 'Add Field' dialog box for a message named 'Agriculture'. The 'Basic Information' tab is selected, showing the message name 'Agriculture' and message type 'Data Report'. The 'Fields' tab lists two fields: 'messageId' (0-1, int8u, length 1) and 'Temperature' (1-2, int8u, length 1). A red box highlights the '+ Add Field' button at the top right of the field list.

根据设计思路，输入字段名字“Humidity”，点击“确认”；



The screenshot shows the 'Add Field' dialog box with the field name set to 'Humidity'. Other settings include: Description: '输入字段描述', Data Type: 'int8u' (Big-endian mode), Length: '1', Default Value: empty, and Offset: '2-3'. At the bottom are 'Confirm' and 'Cancel' buttons.

点击“添加字段”；



The screenshot shows the 'Add Message' dialog with the 'Fields' tab selected. It displays three fields:

| 索引 | 名字 | 消息描述 | 数据类型 | 长度 | 是否地址域 |
|-----|-------------|------|-------|----|-------------------------------------|
| 0-1 | messageid | - | int8u | 1 | <input checked="" type="checkbox"/> |
| 1-2 | Temperature | - | int8u | 1 | <input type="checkbox"/> |
| 2-3 | Humidity | - | int8u | 1 | <input type="checkbox"/> |

A red box highlights the '+ 添加字段' button at the top right of the table.

根据设计思路，输入字段名字“Luminance”，数据类型“int16u”，长度“2”，点击“确认”；

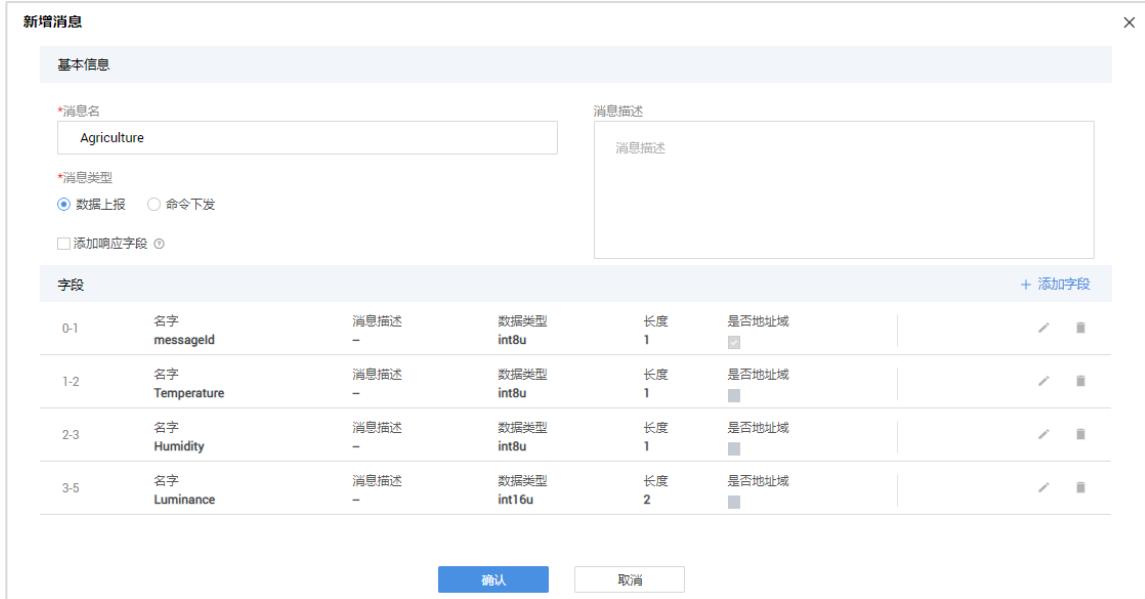


The screenshot shows the 'Add Field' dialog with the following settings:

- 标记为地址域:
- 名字: **Luminance**
- 描述: 输入字段描述
- 数据类型 (大端模式): **int16u**
- 长度: **2**
- 默认值: (empty)
- 偏移值: **3-5**

At the bottom, there are '确认' (Confirm) and '取消' (Cancel) buttons.

点击“确认”；



基本信息

*消息名: Agriculture
*消息类型: 数据上报 (radio button selected)
□添加响应字段 (checkbox)

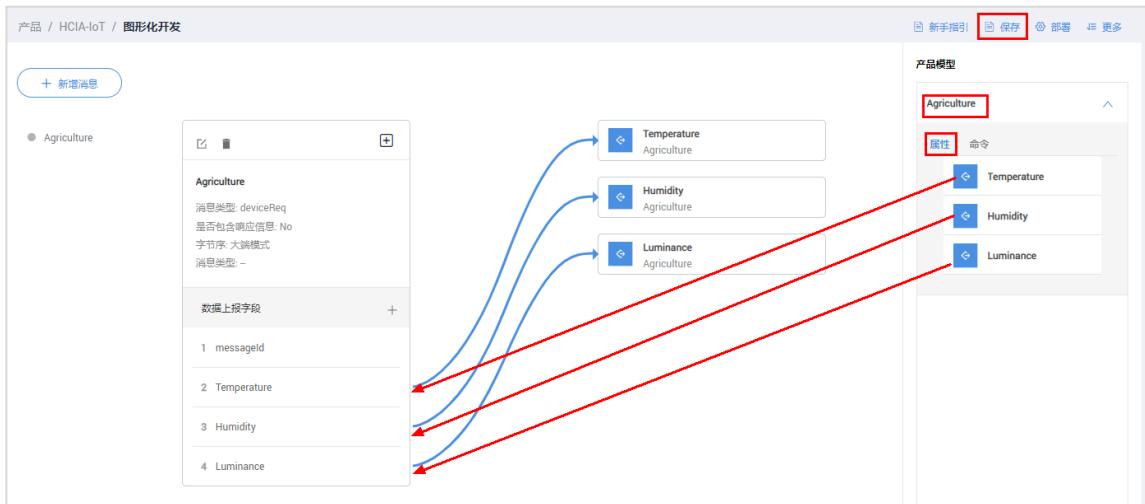
消息描述

字段

| 序号 | 名字 | 消息描述 | 数据类型 | 长度 | 是否地址域 | 操作 | |
|-----|----------------|------|-------------|----|-------|----|--|
| 0-1 | messageId | - | int8u | 1 | 是 | | |
| 1-2 | 名字 Temperature | 消息描述 | 数据类型 int8u | 1 | 是 | | |
| 2-3 | 名字 Humidity | 消息描述 | 数据类型 int8u | 1 | 是 | | |
| 3-5 | 名字 Luminance | 消息描述 | 数据类型 int16u | 2 | 是 | | |

确认 取消

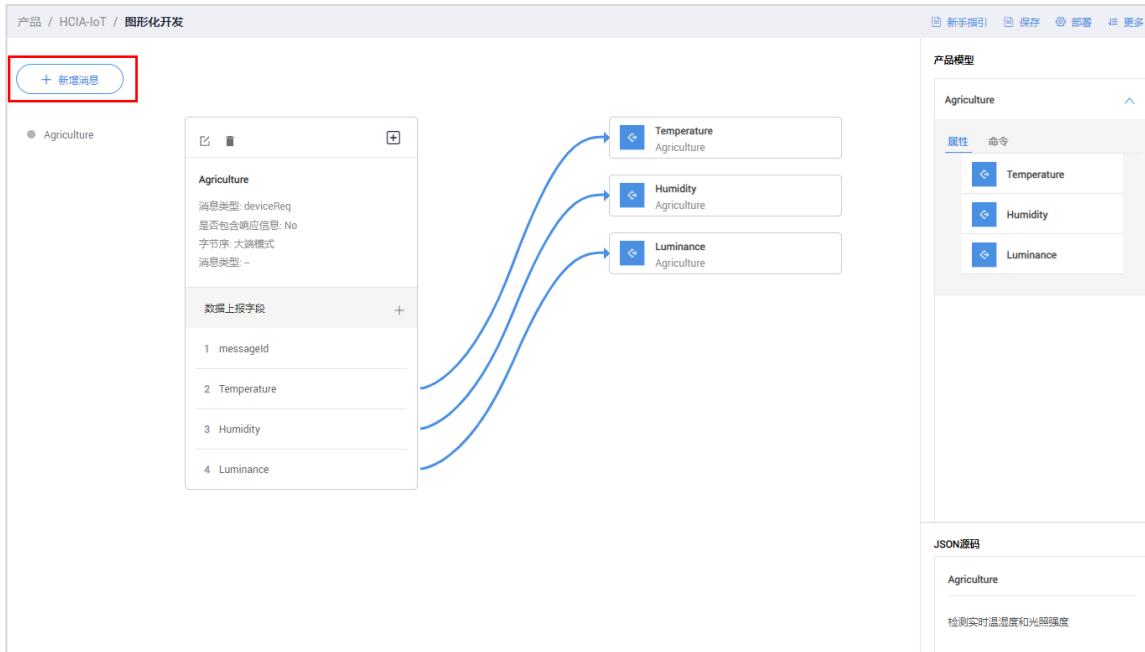
点击右侧产品模型“Agriculture”->“属性”，将三个属性逐个拖动到左侧，与消息中的字段一一对应。



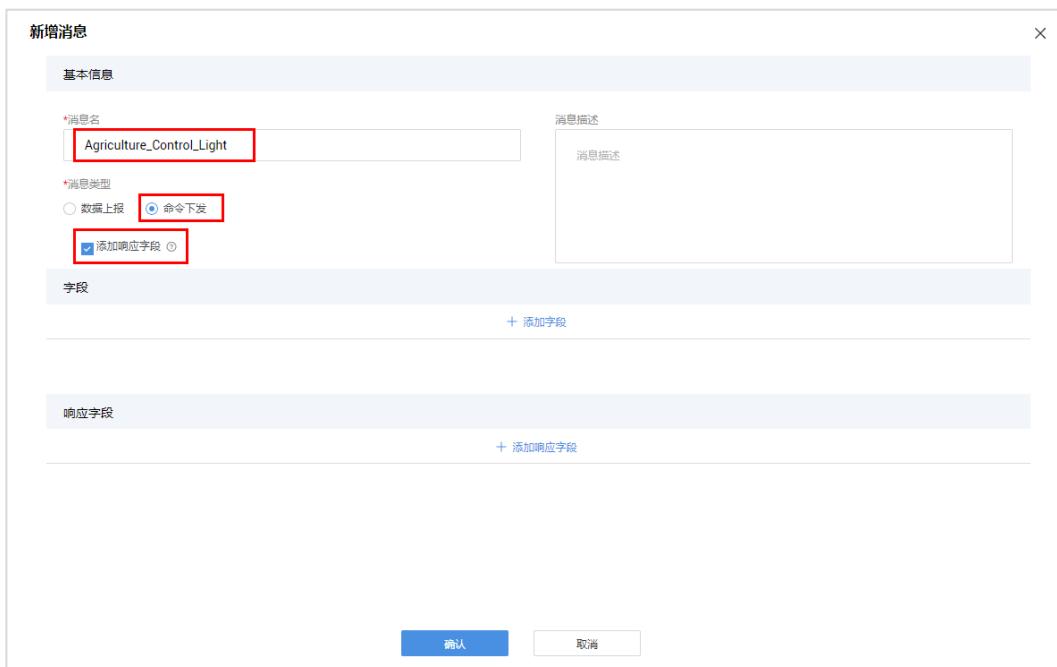
点击右上角“保存”，智慧农业数据上报消息新增成功。

步骤 3 新增 Light 命令消息

点击“新增消息”；

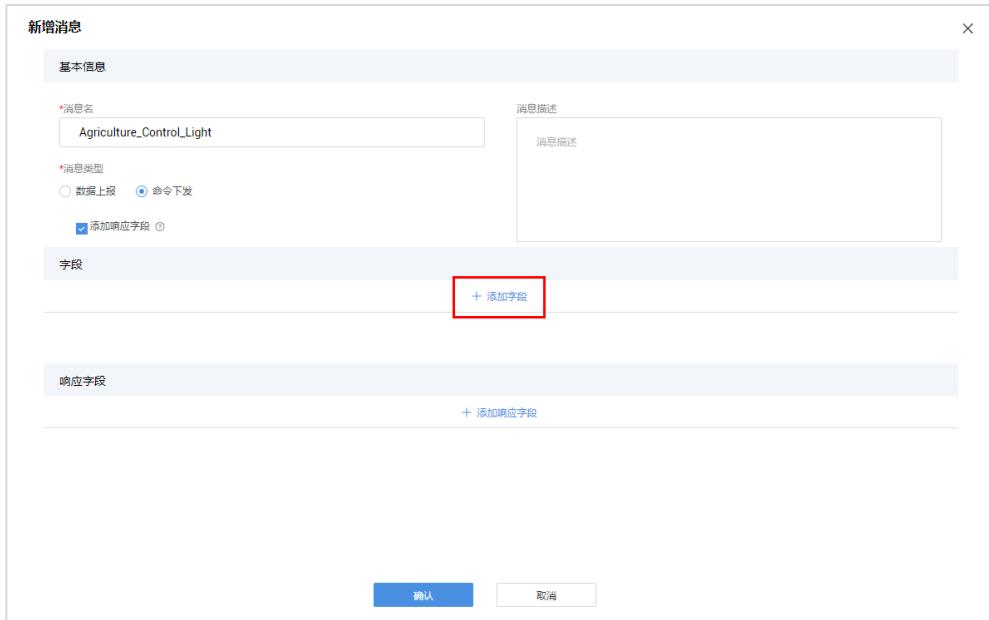


根据设计思路，输入消息名“`Agriculture_Control_Light`”，消息类型“命令下发”，勾选“添加响应字段”；



The screenshot shows the '新增消息' (Add Message) dialog box. In the '基本信息' (Basic Information) tab, the '消息名' (Message Name) field is set to 'Agriculture_Control_Light' (highlighted by a red box). The '消息类型' (Message Type) section has '命令下发' (Command Downlink) selected (highlighted by a red box). The '添加响应字段' (Add Response Field) checkbox is checked (highlighted by a red box). The '字段' (Fields) tab shows a single field entry with a '+' button to add more. The '响应字段' (Response Fields) tab is empty. At the bottom, there are '确认' (Confirm) and '取消' (Cancel) buttons.

点击“添加字段”；



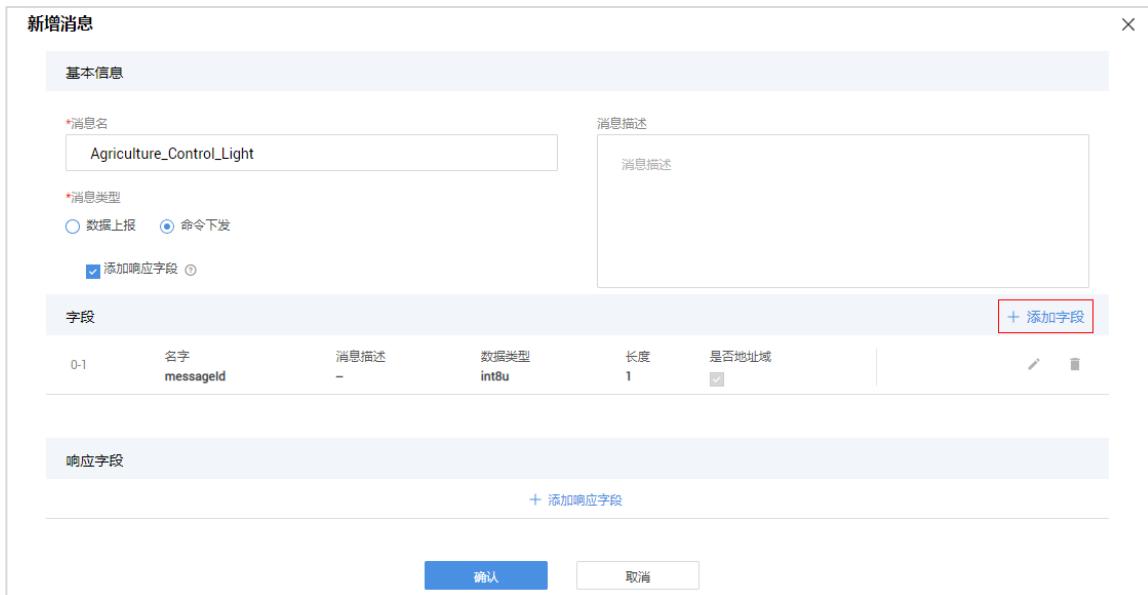
The screenshot shows the 'Add Message' dialog. In the 'Basic Information' tab, the message name is set to 'Agriculture_Control_Light' and the message type is selected as 'Command Downlink'. A checkbox for 'Add Response Identifier' is checked. In the 'Fields' tab, there is a red box around the '+ Add Field' button. Below it, the 'Response Fields' tab is visible with a '+ Add Response Field' button.

勾选“标记为地址域”，点击“确认”；



The screenshot shows the 'Add Field' dialog. Under the 'Address Domain' section, the 'Check' box is checked. The field name is set to 'messageId'. In the 'Description' section, the placeholder text is 'Input field description'. Under 'Data Type (Big-endian mode)', the selection is 'int8u'. The 'Length' is set to '1'. The 'Default Value' is '0x1'. The 'Offset' is '0-1'. At the bottom, there are 'Confirm' and 'Cancel' buttons.

点击“添加字段”；



The screenshot shows the 'Add Field' dialog box. In the 'Basic Information' tab, the message name is set to 'Agriculture_Control_Light', the message type is 'Command Downlink', and the response field is checked. In the 'Fields' tab, there is one field entry: 'messageId' (name), 'int8u' (data type), and length '1'. The 'Is Address Domain' checkbox is unchecked. A red-bordered button '+ Add Field' is visible. In the 'Response Fields' tab, there is a '+ Add Response Field' button. At the bottom are 'Confirm' and 'Cancel' buttons.

勾选“标记为响应标识字段”，其他默认，点击“确认”；



The screenshot shows the 'Add Field' dialog box with the 'Mark as Response Identifier Field' checkbox checked. The field name is 'mid', and the data type is 'int16u (16-bit unsigned integer)'. Other fields like length (2), default value, and offset (1-3) are also filled. The 'Confirm' and 'Cancel' buttons are at the bottom.

点击“添加字段”；



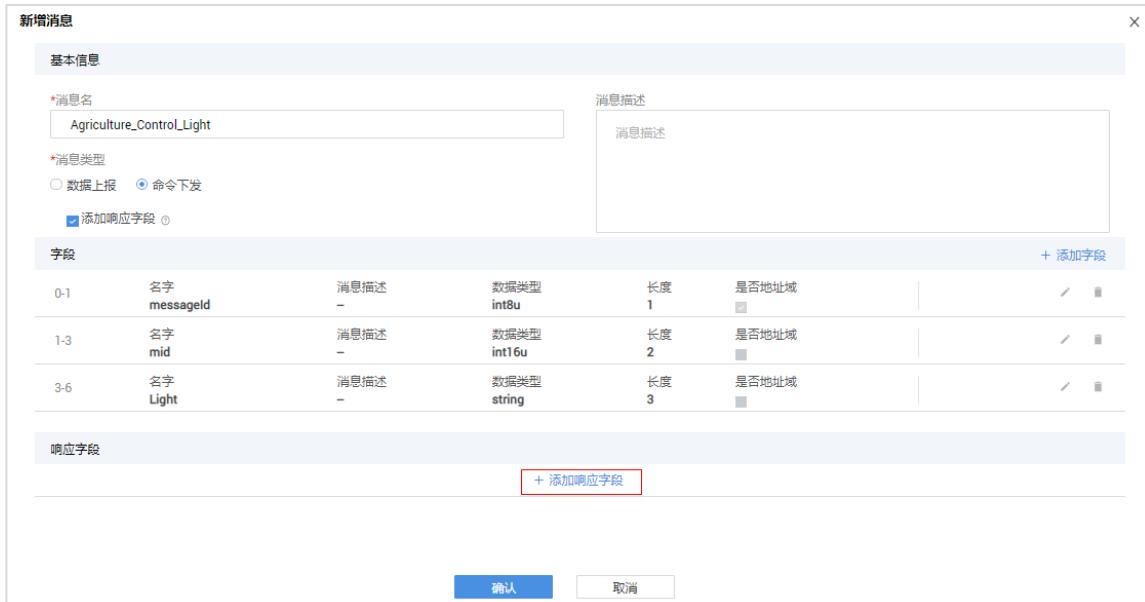
The screenshot shows the 'Add Message' dialog box. In the 'Basic Information' tab, the message name is set to 'Agriculture_Control_Light', the message type is selected as 'Command', and the 'Add Response Field' checkbox is checked. In the 'Fields' tab, there are two fields defined: 'messageId' (length 1) and 'mid' (length 2). A red box highlights the '+ Add Field' button. At the bottom, there are 'Confirm' and 'Cancel' buttons.

根据设计思路，输入字段名字“Light”，数据类型“string”，长度“3”，点击“确认”；



The screenshot shows the 'Add Field' dialog box. The 'Name' field is filled with 'Light', and the 'Type' dropdown is set to 'string'. The 'Length' field is set to '3'. The 'Default Value' and 'Offset' fields are empty. At the bottom, there are 'Confirm' and 'Cancel' buttons.

点击“添加响应字段”；



新增消息

基本信息

*消息名: Agriculture_Control_Light

消息类型: 命令下发

添加响应字段

字段

| Index | Name | Description | Data Type | Length | 是否地址域 |
|-------|-----------|-------------|-----------|--------|-------------------------------------|
| 0-1 | messageId | - | int8u | 1 | <input checked="" type="checkbox"/> |
| 1-3 | mid | - | int16u | 2 | <input checked="" type="checkbox"/> |
| 3-6 | Light | - | string | 3 | <input checked="" type="checkbox"/> |

响应字段

+ 添加响应字段

确认 取消

勾选“标记为地址域”，点击“确认”；



添加字段

标记为地址域

标记为响应标识字段

标记为命令执行状态字段

*名字 只有标记为地址域时，名字固定为messageId；其他字段名字不能设置为messageId。

messageId

描述

输入字段描述

数据类型 (大端模式)

int8u

*长度

1

*默认值

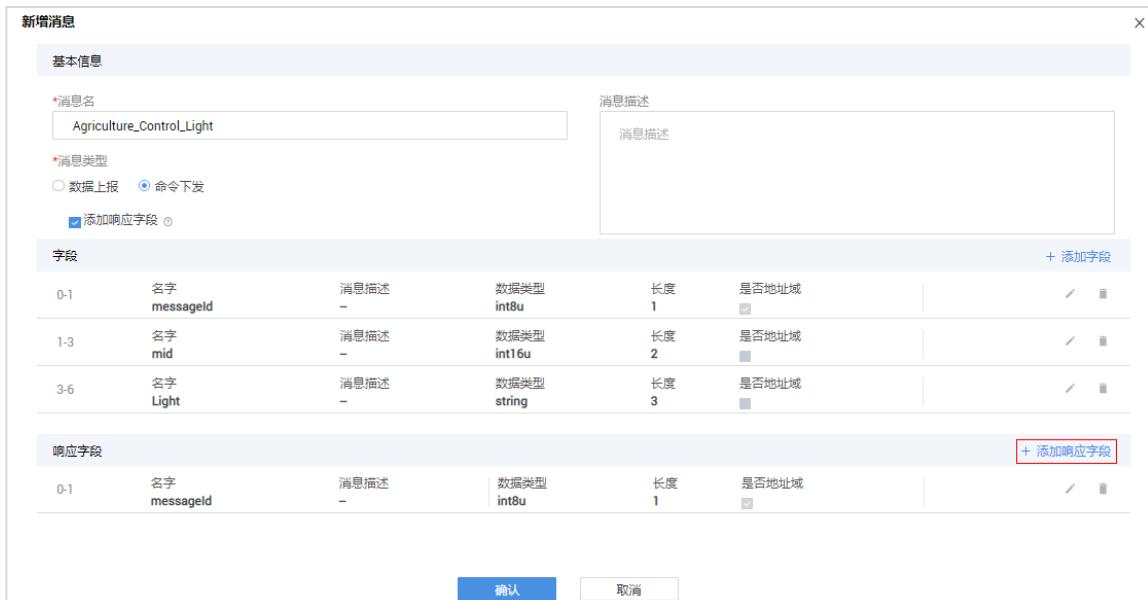
0x2

偏移值

0-1

确认 取消

点击“添加响应字段”；



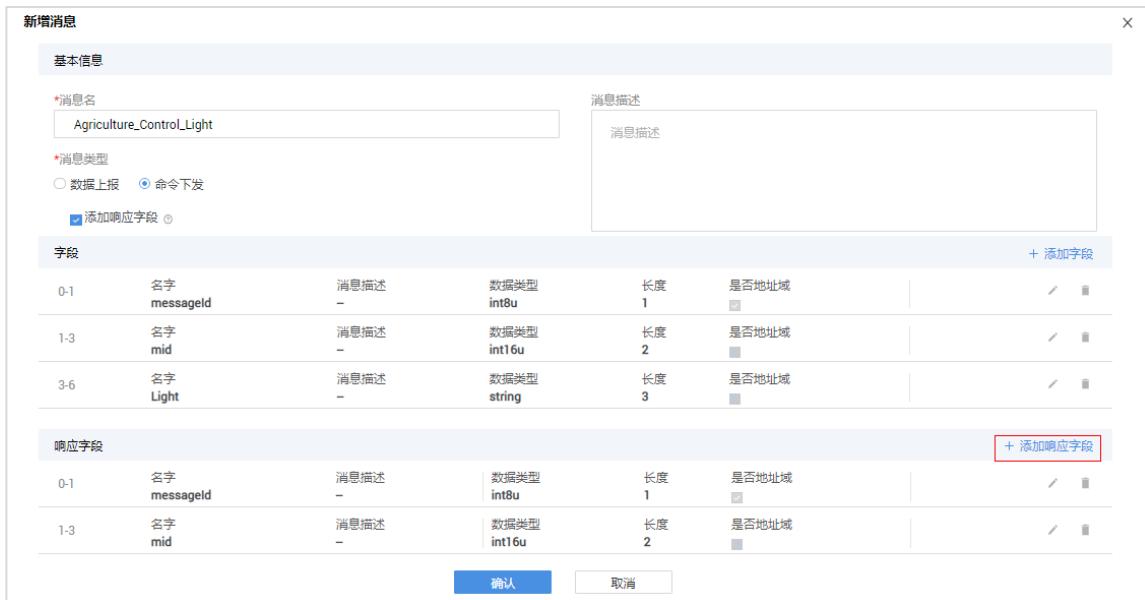
The screenshot shows the 'Add Response Field' dialog. At the top left is the message name 'Agriculture_Control_Light'. Under 'Message Type', 'Command Downlink' is selected. A checked checkbox labeled 'Add Response Field' is visible. The main area contains two tables: one for fields (0-1, 1-3, 3-6) and one for response fields (0-1). The 'Response Field' table has a red border around its header. Buttons for 'Confirm' and 'Cancel' are at the bottom.

勾选“标记为响应标识字段”，点击“确认”；



The screenshot shows the 'Add Field' dialog. Under 'Field Marking', 'Marked as Response Identifier Field' is checked. The 'Name' field contains 'mid'. The 'Description' field is empty. Under 'Data Type (Big-endian mode)', 'int16u (16-bit unsigned integer)' is selected. The 'Length' is set to '2'. The 'Default Value' and 'Offset' fields are empty. Buttons for 'Confirm' and 'Cancel' are at the bottom.

点击“添加响应字段”；



新增消息

基本信息

*消息名: Agriculture_Control_Light

*消息类型: 命令下发

添加响应字段

字段

| Address | Name | Description | Type | Length | 是否地址域 |
|---------|-----------|-------------|--------|--------|-------------------------------------|
| 0-1 | messageld | - | int8u | 1 | <input checked="" type="checkbox"/> |
| 1-3 | mid | - | int16u | 2 | <input checked="" type="checkbox"/> |
| 3-6 | Light | - | string | 3 | <input checked="" type="checkbox"/> |

响应字段

| Address | Name | Description | Type | Length | 是否地址域 |
|---------|-----------|-------------|--------|--------|-------------------------------------|
| 0-1 | messageId | - | int8u | 1 | <input checked="" type="checkbox"/> |
| 1-3 | mid | - | int16u | 2 | <input checked="" type="checkbox"/> |

+ 添加响应字段

确认 取消

点击“标记为命令执行状态字段”，点击“确认”；



添加字段

标记为地址域

标记为响应标识字段

标记为命令执行状态字段

*名字 只有标记为命令执行状态字段时，名字固定为errcode; 其他字段名字不能设置为errcode。

errcode

描述

输入字段描述

数据类型 (大端模式)

int8u

*长度

1

默认值

偏移值

3-4

确认 取消

点击“添加响应字段”；



新增消息

Agriculture_Control_Light

消息描述

消息类型

数据上报 命令下发

添加响应字段

字段

| 序号 | 名字 | 消息描述 | 数据类型 | 长度 | 是否地址域 | 操作 |
|-----|-----------|------|--------|----|-------------------------------------|----|
| 0-1 | messageid | - | int8u | 1 | <input checked="" type="checkbox"/> | |
| 1-3 | mid | - | int16u | 2 | <input checked="" type="checkbox"/> | |
| 3-6 | Light | - | string | 3 | <input checked="" type="checkbox"/> | |

响应字段

| 序号 | 名字 | 消息描述 | 数据类型 | 长度 | 是否地址域 | 操作 |
|-----|-----------|------|--------|----|-------------------------------------|----|
| 0-1 | messageid | - | int8u | 1 | <input checked="" type="checkbox"/> | |
| 1-3 | mid | - | int16u | 2 | <input checked="" type="checkbox"/> | |
| 3-4 | encode | - | int8u | 1 | <input checked="" type="checkbox"/> | |

确认 取消

根据设计思路，输入字段名字“Light_State”，点击“确认”；



添加字段

标记为地址域

标记为响应标识字段

标记为命令执行状态字段

*名字

Light_State

描述

输入字段描述

数据类型 (大端模式)

int8u

*长度

1

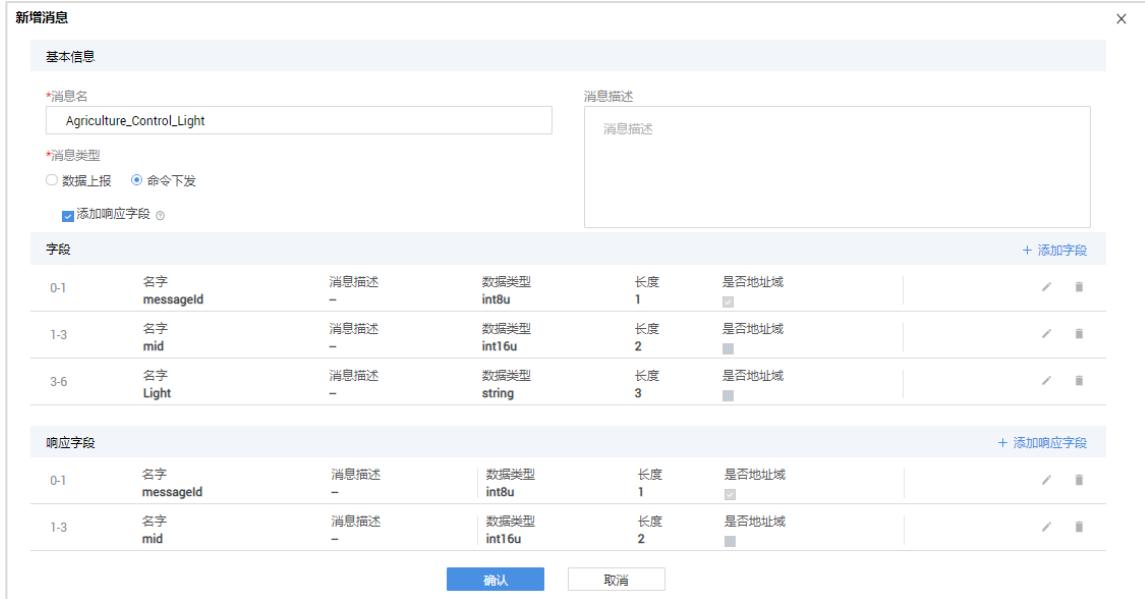
默认值

偏移值

4-5

确认 取消

点击“确认”；



新增消息

基本信息

*消息名: Agriculture_Control_Light

*消息类型: 命令下发

○ 数据上报 ○ 命令下发

添加响应字段

字段

| 序号 | 名字 | 消息描述 | 数据类型 | 长度 | 是否地址域 |
|-----|-----------|------|--------|----|--------------------------|
| 0-1 | messageld | - | int8u | 1 | <input type="checkbox"/> |
| 1-3 | mid | - | int16u | 2 | <input type="checkbox"/> |
| 3-6 | Light | - | string | 3 | <input type="checkbox"/> |

响应字段

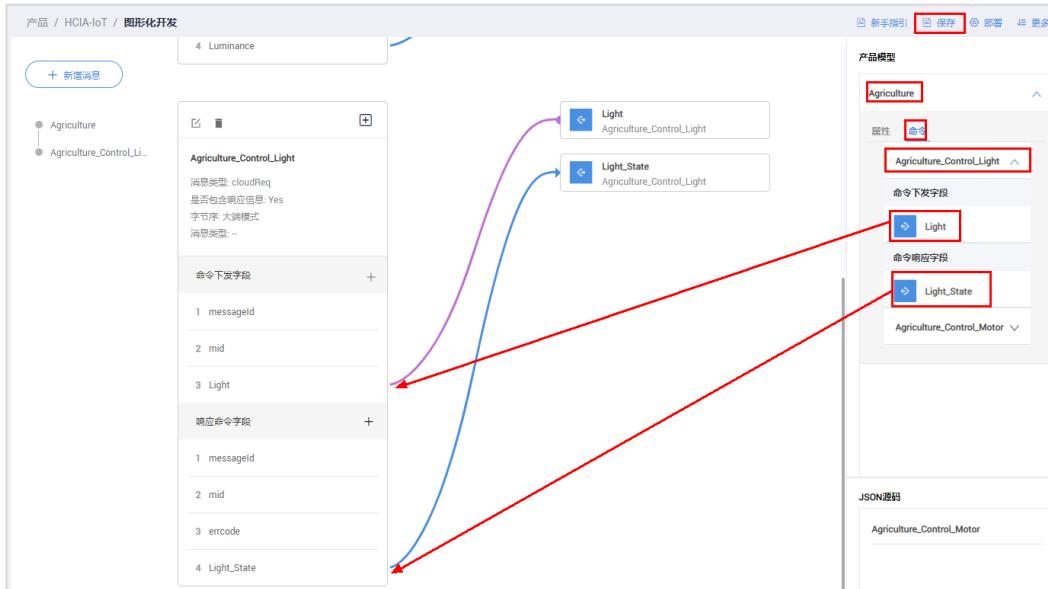
| 序号 | 名字 | 消息描述 | 数据类型 | 长度 | 是否地址域 |
|-----|-----------|------|--------|----|--------------------------|
| 0-1 | messageld | - | int8u | 1 | <input type="checkbox"/> |
| 1-3 | mid | - | int16u | 2 | <input type="checkbox"/> |

+ 添加字段

+ 添加响应字段

确认 取消

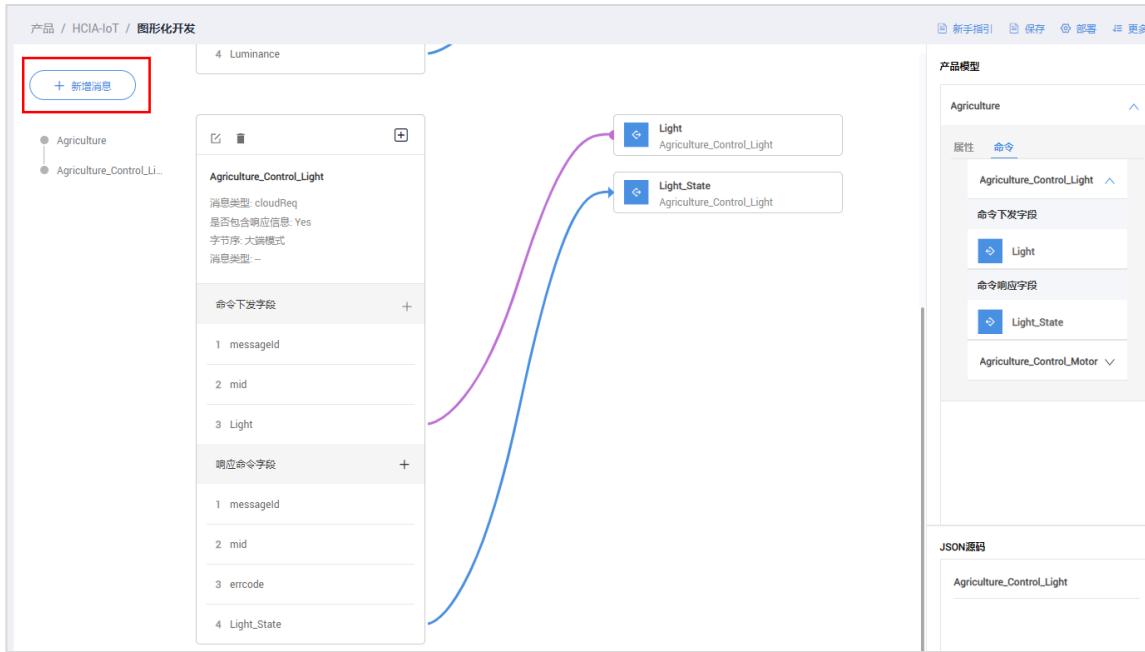
点击“Agriculture”->“命令”->“Agriculture_Control_Light”，将 Light 和 Light_State 两个字段逐个拖动到左侧，消息中的字段一一对应。



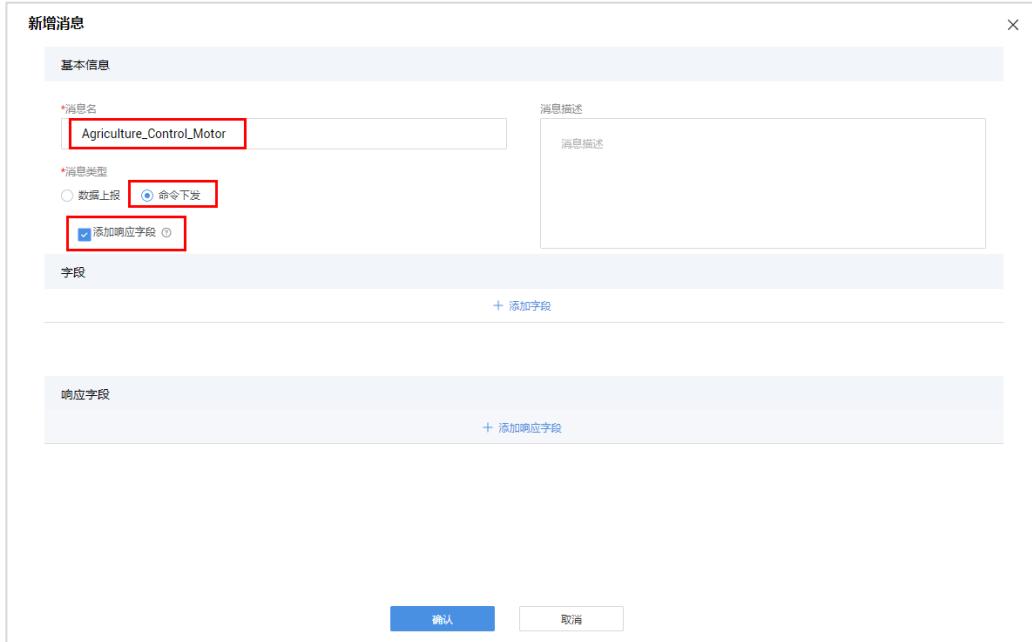
点击右上角“保存”，智慧农业 Light 命令消息新增成功。

步骤 4 新增 Motor 命令消息

点击“新增消息”；

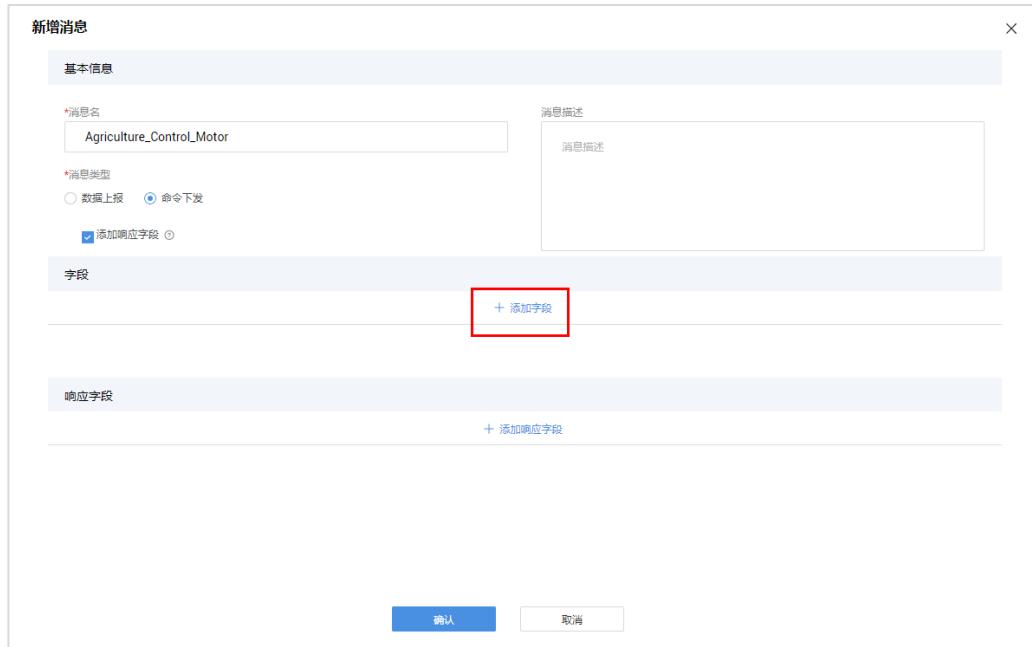


根据设计思路，输入消息名“`Agriculture_Control_Motor`”，消息类型“命令下发”，勾选“添加响应字段”；



The screenshot shows the '新增消息' (Add New Message) dialog box. In the '基本信息' tab, the '消息名' field is filled with 'Agriculture_Control_Motor', the '消息类型' field has '命令下发' (Command Downlink) selected, and the '添加响应字段' (Add Response Fields) checkbox is checked. The '字段' (Fields) and '响应字段' (Response Fields) tabs are also visible at the bottom of the dialog.

点击“添加字段”；



The screenshot shows the 'Add Message' dialog with the 'Basic Information' tab selected. Under 'Message Name', 'Agriculture_Control_Motor' is entered. Under 'Message Type', 'Command Downlink' is selected. A checkbox for 'Add Response Field' is checked. The 'Fields' tab is active, showing a single row with a '+' button labeled '+ Add Field' highlighted by a red box. The 'Response Fields' tab is also visible.

勾选“标记为地址域”，点击“确认”；



The screenshot shows the 'Add Field' dialog. The 'Address Domain' checkbox is checked. The 'Name' field contains 'messageId'. The 'Description' field is empty. Under 'Data Type (Big-endian mode)', 'int8u' is selected. The 'Length' field contains '1'. The 'Default Value' field contains '0x1'. The 'Offset' field contains '0-1'. At the bottom, the 'Confirm' button is highlighted by a blue box.

点击“添加字段”；



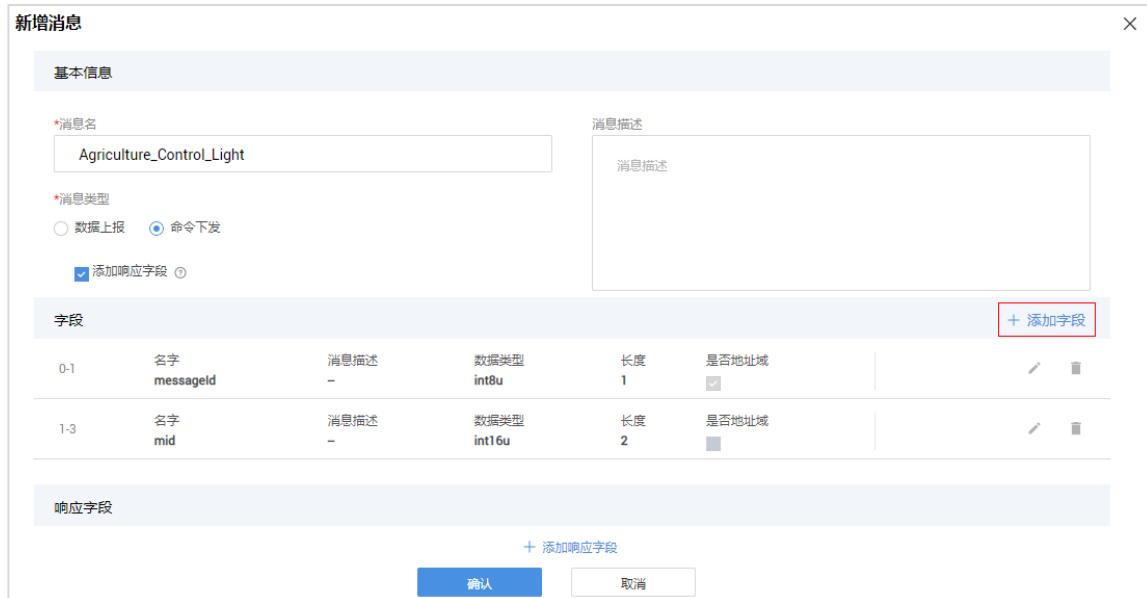
The screenshot shows the 'Add Message' dialog. In the 'Basic Information' tab, the message name is set to 'Agriculture_Control_Motor' and the message type is selected as 'Command Dispatch'. A checkbox for 'Add Response Identifier Field' is checked. The 'Fields' section contains one field entry: 'messageld' (name), '-' (description), 'int8u' (data type), '1' (length), and a checked 'Is Address Domain' checkbox. A red box highlights the '+ Add Field' button. Below this is the 'Response Field' section with a '+ Add Response Field' button.

勾选“标记为响应标识字段”，其他默认，点击“确认”；



The screenshot shows the 'Add Field' dialog. Under 'Field Properties', the 'Mark as Response Identifier Field' checkbox is checked. The field name is 'mid'. In the 'Description' section, there is a placeholder 'Input field description'. Under 'Data Type (Big-endian mode)', the type is 'int16u (16-bit unsigned integer)'. The 'Length' is set to '2'. There are empty fields for 'Default Value' and 'Offset'. At the bottom are 'Confirm' and 'Cancel' buttons.

点击“添加字段”；



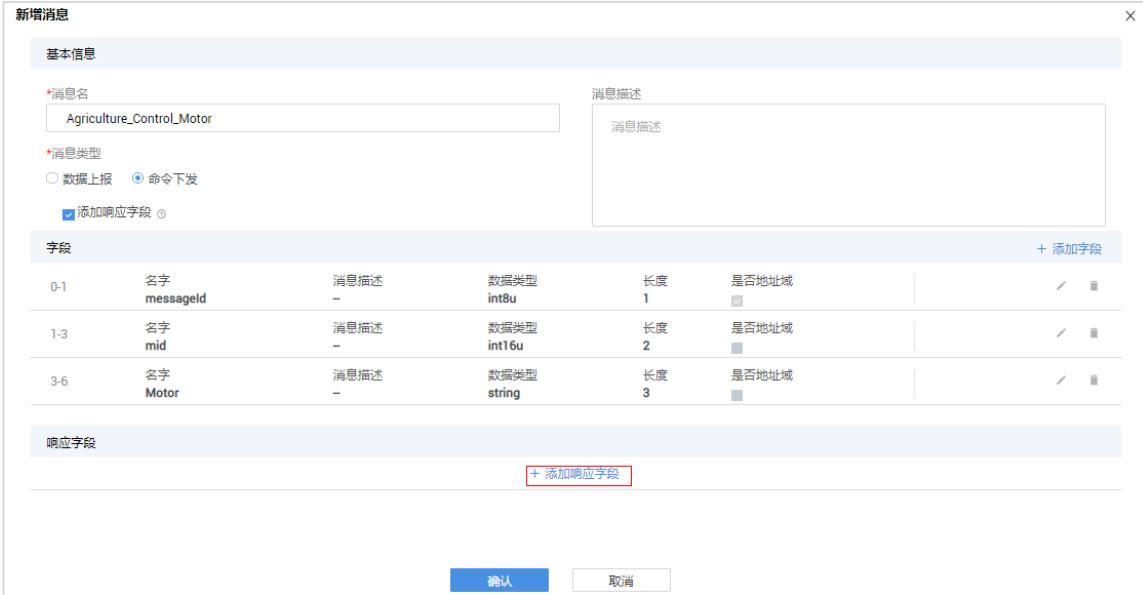
The screenshot shows the 'Add Message' dialog. In the 'Basic Information' tab, the message name is set to 'Agriculture_Control_Light', the message type is selected as 'Command Dispatch', and the 'Add Response Field' checkbox is checked. In the 'Fields' tab, there are two fields defined: 'messageId' (length 1, type int8u) and 'mid' (length 2, type int16u). A red box highlights the '+ Add Field' button. At the bottom, there are 'Confirm' and 'Cancel' buttons.

根据设计思路，输入字段名字“Motor”，数据类型“string”，长度“3”，点击“确认”；



The screenshot shows the 'Add Field' dialog. It includes fields for marking it as an address domain or a response identifier, a name input field containing 'Motor', a description input field, a data type dropdown set to 'string', a length input field set to '3', a default value input field, and a offset input field set to '3-6'. At the bottom, there are 'Confirm' and 'Cancel' buttons.

点击“添加响应字段”；



The screenshot shows the 'Add Response Field' dialog. At the top left is the title '新增消息'. Below it is a '基本信息' tab with fields for '消息名' (Agriculture_Control_Motor) and '消息类型' (Command Downlink). A checked checkbox '添加响应字段' is present. On the right is a '消息描述' text area. Below this is a '字段' table with three rows:

| 序号 | 名字 | 消息描述 | 数据类型 | 长度 | 是否地址域 |
|-----|-----------|------|--------|----|-------------------------------------|
| 0-1 | messageld | - | int8u | 1 | <input checked="" type="checkbox"/> |
| 1-3 | mid | - | int16u | 2 | <input checked="" type="checkbox"/> |
| 3-6 | Motor | - | string | 3 | <input checked="" type="checkbox"/> |

Below the table is a '响应字段' section with a red-bordered '+ 添加响应字段' button. At the bottom are '确认' and '取消' buttons.

勾选“标记为地址域”，点击“确认”；

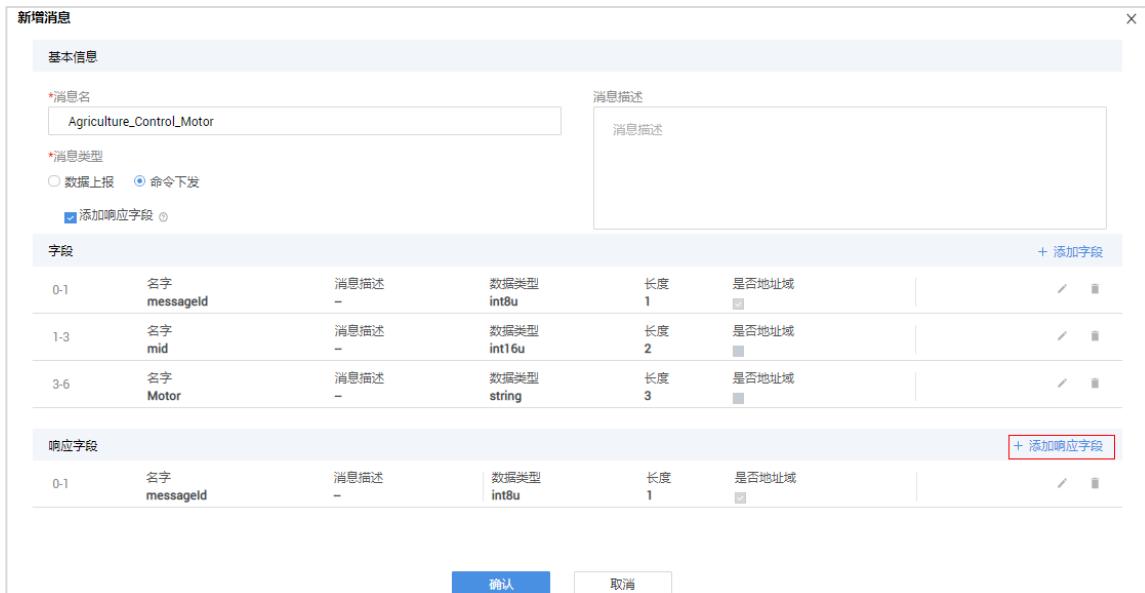


The screenshot shows the 'Mark as Address Domain' dialog. It has several sections:

- 标记为地址域**: A checked checkbox.
- 名字**: A note in red: '只有标记为地址域时，名字固定为messageld; 其他字段名字不能设置为messageld.' Below is a text input field containing 'messageld'.
- 描述**: A text input field with placeholder '输入字段描述'.
- 数据类型 (大端模式)**: A dropdown menu showing 'int8u'.
- *长度**: An input field containing '1'.
- *默认值**: An input field containing '0x2'.
- 偏移值**: An input field containing '0-1'.

At the bottom are '确认' and '取消' buttons.

点击“添加响应字段”；



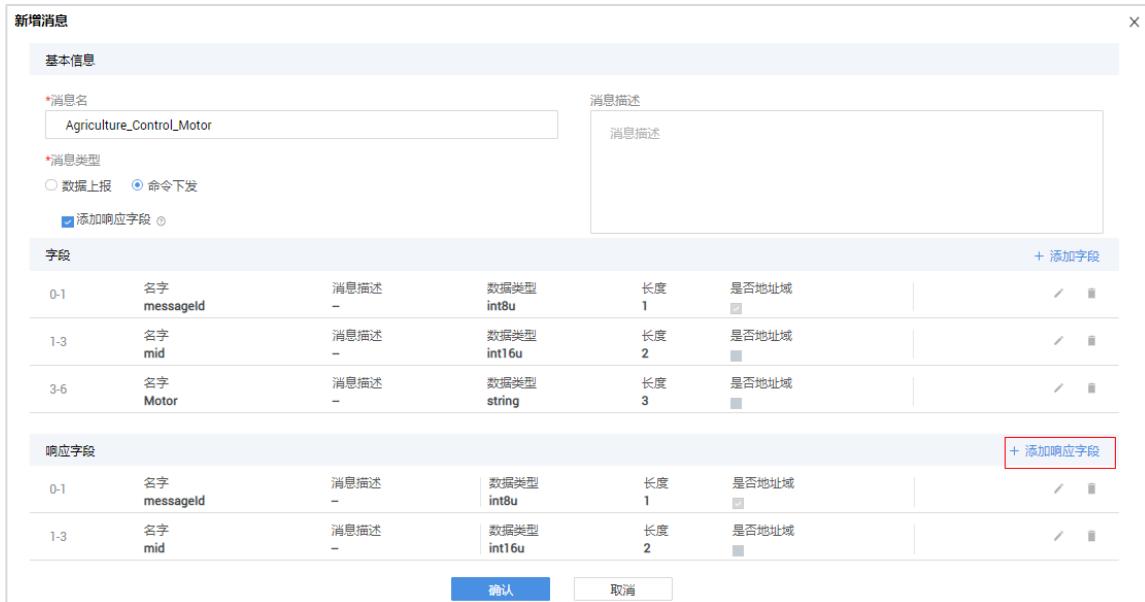
The screenshot shows the 'Add Response Field' dialog. In the 'Basic Information' tab, the message name is set to 'Agriculture_Control_Motor' and the message type is selected as 'Command Dispatch'. The 'Add Response Field' checkbox is checked. In the 'Fields' tab, three fields are listed: 'messageld' (0-1), 'mid' (1-3), and 'Motor' (3-6). Each field has its name, description, data type (int8u or int16u), length, and whether it is an address domain. A red box highlights the '+ Add Response Field' button. In the 'Response Fields' tab, there is one entry for 'messageld' with the same properties. A red box highlights the '+ Add Response Field' button again. At the bottom are 'Confirm' and 'Cancel' buttons.

勾选“标记为响应标识字段”，点击“确认”；



The screenshot shows the 'Mark as Response Identifier Field' dialog. It contains four checkboxes: 'Address Domain' (unchecked), 'Response Identifier Field' (checked), 'Command Execution Status Field' (unchecked), and 'Command Identifier Field' (unchecked). Below these is a note: '*Name 只有标记为响应标识字段时，名字固定为mid; 其他字段名字不能设置为mid.' A text input field contains 'mid'. The 'Description' section has a placeholder 'Input field description'. The 'Data Type (Big-endian mode)' dropdown is set to 'int16u (16-bit unsigned integer)'. The 'Length' input field contains '2'. The 'Default Value' input field is empty. The 'Offset' input field contains '1-3'. At the bottom are 'Confirm' and 'Cancel' buttons.

点击“添加响应字段”；



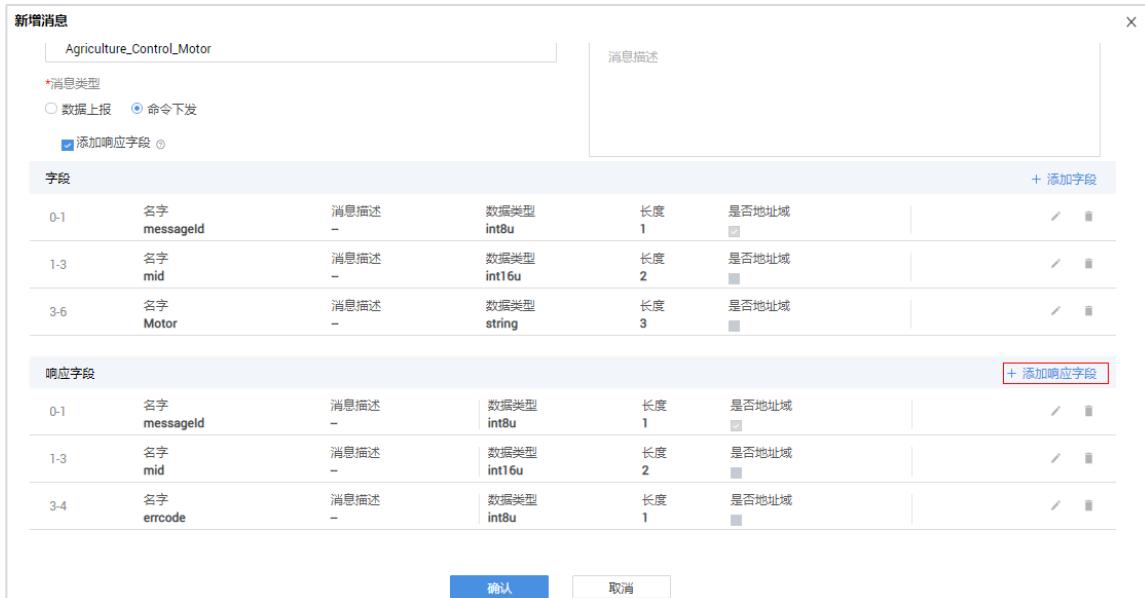
The screenshot shows the 'Add Response Field' dialog. At the top left is the title '新增消息' (New Message). Below it is a 'Basic Information' section with fields for 'Message Name' (Agriculture_Control_Motor) and 'Message Type' (Command Downlink). A checked checkbox 'Add Response Field' is present. The main area is divided into two sections: 'Fields' and 'Response Fields'. The 'Fields' section contains three entries: 0-1 (name: messageid, type: int8u, length: 1), 1-3 (name: mid, type: int16u, length: 2), and 3-6 (name: Motor, type: string, length: 3). The 'Response Fields' section also contains three entries: 0-1 (name: messageid, type: int8u, length: 1) and 1-3 (name: mid, type: int16u, length: 2). A red-bordered button '+ Add Response Field' is located at the top right of the response field section. At the bottom are 'Confirm' and 'Cancel' buttons.

点击“标记为命令执行状态字段”，点击“确认”；



The screenshot shows the 'Mark as Command Execution Status Field' dialog. It has several sections: 'Mark as Address Domain' (unchecked), 'Mark as Response Identifier Field' (unchecked), and 'Mark as Command Execution Status Field' (checked). A note below states: '*Name 只有标记为命令执行状态字段时，名字固定为errcode; 其他字段名字不能设置为errcode.' (Only when marked as a command execution status field, the name is fixed as errcode; other field names cannot be set to errcode.). An input field contains 'errcode'. The 'Description' section has a placeholder 'Input field description'. The 'Data Type (Big-endian Mode)' dropdown is set to 'int8u'. The 'Length' field is set to '1'. The 'Default Value' field is empty. The 'Offset' field is set to '3-4'. At the bottom are 'Confirm' and 'Cancel' buttons.

点击“添加响应字段”；



| 字段 | | | | | | |
|-----|-----------------|-----------|----------------|---------|--|---|
| 0-1 | 名字 messageld | 消息描述 - | 数据类型 int8u | 长度 1 | 是否地址域 <input checked="" type="checkbox"/> | <input type="checkbox"/> <input type="checkbox"/> |
| 1-3 | 名字 mid | 消息描述 - | 数据类型 int16u | 长度 2 | 是否地址域 <input checked="" type="checkbox"/> | <input type="checkbox"/> <input type="checkbox"/> |
| 3-6 | 名字 Motor | 消息描述 - | 数据类型 string | 长度 3 | 是否地址域 <input checked="" type="checkbox"/> | <input type="checkbox"/> <input type="checkbox"/> |

| 响应字段 | | | | | | |
|------|-----------------|-----------|----------------|---------|--|---|
| 0-1 | 名字 messageld | 消息描述 - | 数据类型 int8u | 长度 1 | 是否地址域 <input checked="" type="checkbox"/> | <input type="checkbox"/> <input type="checkbox"/> |
| 1-3 | 名字 mid | 消息描述 - | 数据类型 int16u | 长度 2 | 是否地址域 <input checked="" type="checkbox"/> | <input type="checkbox"/> <input type="checkbox"/> |
| 3-4 | 名字 errcode | 消息描述 - | 数据类型 int8u | 长度 1 | 是否地址域 <input checked="" type="checkbox"/> | <input type="checkbox"/> <input type="checkbox"/> |

+ 添加响应字段

确认 取消

根据设计思路，输入字段名字“Motor_State”，点击“确认”；



添加字段

标记为地址域 ①

标记为响应标识字段 ②

标记为命令执行状态字段 ③

*名字
Motor_State

描述
输入字段描述

数据类型 (大端模式)
int8u

*长度 ④
1

默认值 ⑤

偏移值 ⑥
4-5

确认 取消

点击“确认”；

基本信息

*消息名: Agriculture_Control_Motor

*消息类型: 命令下发

+ 添加响应字段

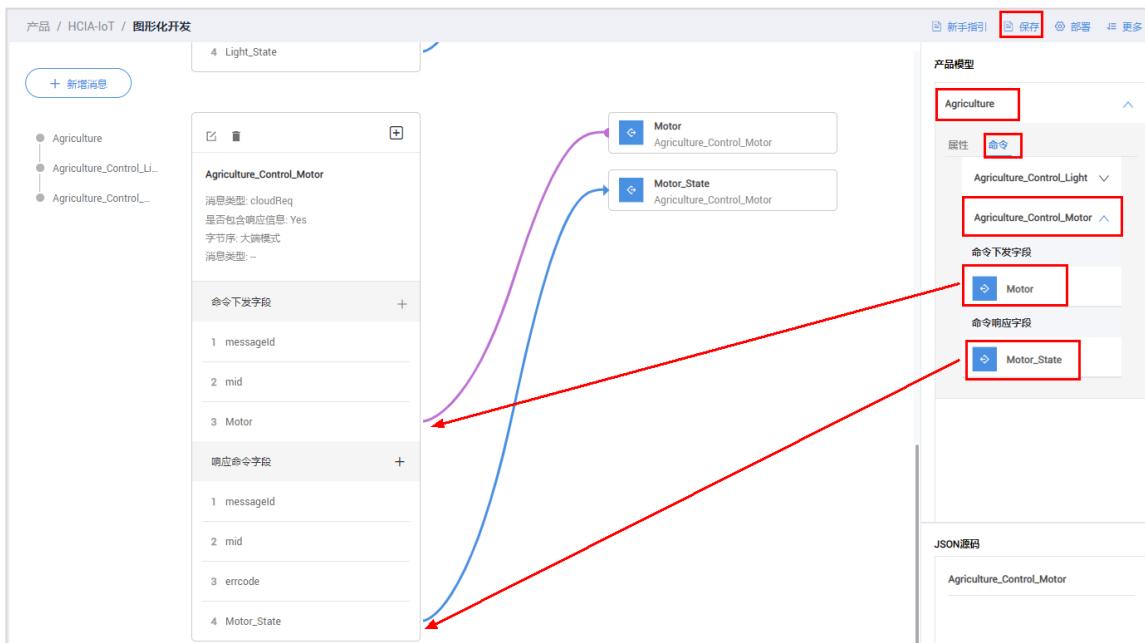
| 字段 | 名字 | 消息描述 | 数据类型 | 长度 | 是否地址域 |
|-----|-----------|------|--------|----|-------|
| 0-1 | messageId | - | int8u | 1 | 否 |
| 1-3 | mid | - | int16u | 2 | 否 |
| 3-6 | Motor | - | string | 3 | 否 |

响应字段

| 字段 | 名字 | 消息描述 | 数据类型 | 长度 | 是否地址域 |
|-----|-----------|------|--------|----|-------|
| 0-1 | messageId | - | int8u | 1 | 否 |
| 1-3 | mid | - | int16u | 2 | 否 |

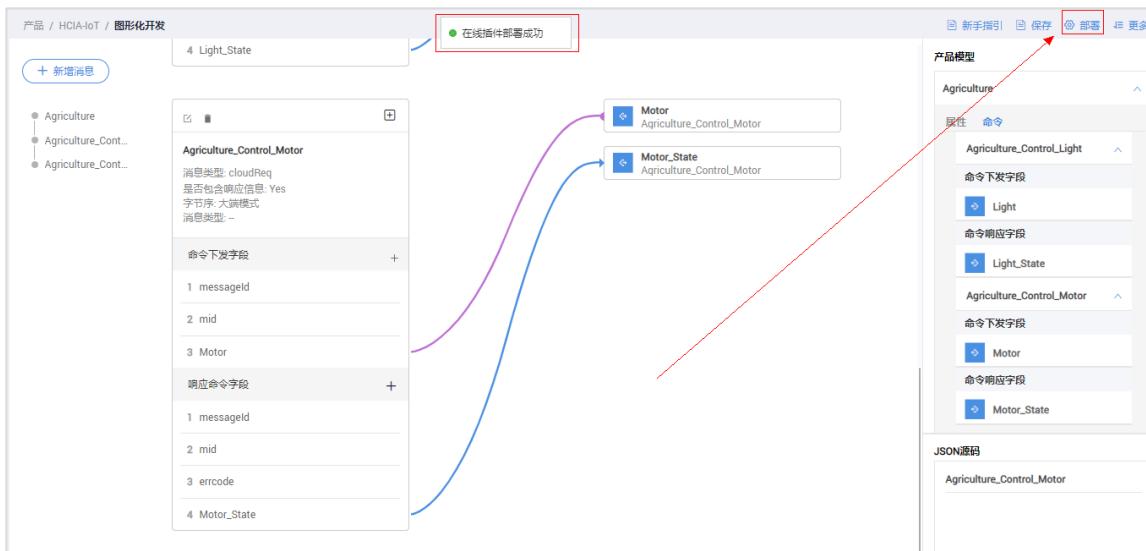
确认 **取消**

点击“Agriculture”->“命令”->“Agriculture_Control_Motor”，将 Motor 和 Motor_State 两个字段逐个拖动到左侧，消息中的字段一一对应。



点击右上角“保存”，智慧农业 Motor 命令消息新增成功。

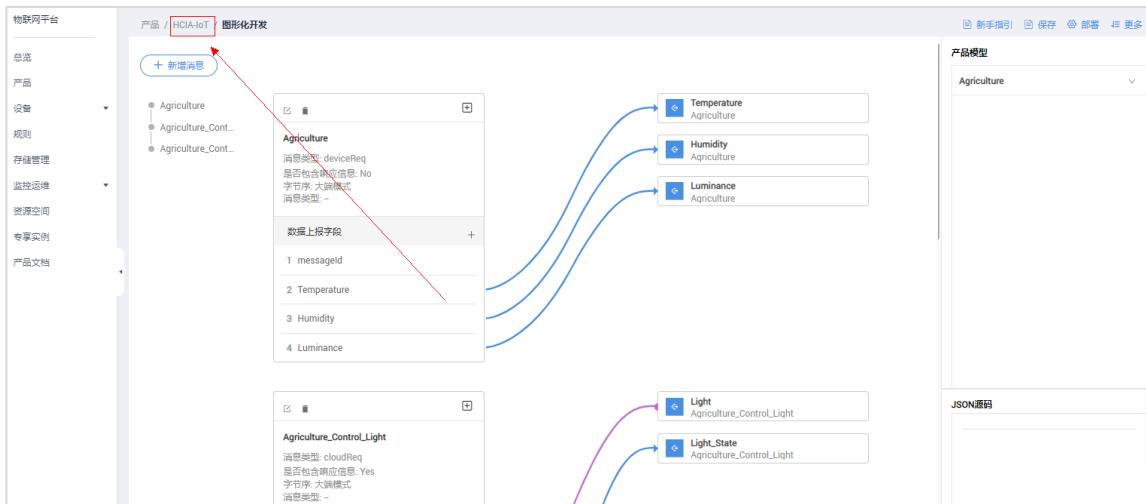
点击右上角“部署”->“确认”，等待提示在线插件部署成功。



1.2.4 验证功能定义及插件

步骤 1 新增模拟设备

点击“HCIP-IoT”，回到产品界面；



点击“在线调试”->“新增测试设备”；



The screenshot shows the HCIA-IoT device management interface. At the top, it displays the product name 'HCIA-IoT' and its ID, along with the number of registered devices (0). Below this, there are several device details: Product Name (HCIA-IoT), Device Type (IoT), Data Format (Binary Stream), and Manufacturer (HCIA-IoT). On the right, it shows the Resource Space (归属资源空间), Protocol Type (协议类型), Last Modified Time (最后修改时间), and Manufacturer ID (厂商ID), all of which are redacted with black bars. Below these details, there are three tabs: 'Function Definition' (功能定义), 'Plugin Development' (插件开发), and 'Online Debugging' (在线调试), with 'Online Debugging' being the active tab and highlighted with a blue border. Underneath the tabs, there is a red rectangular button labeled 'Add Test Device' (新增测试设备). A table below lists columns for Device Name (设备名称), Device Identifier (设备标识码), Device ID (设备ID), and Type (类型). To the right of the table, there is a blue circular icon with a white exclamation mark and a magnifying glass, accompanied by the text 'You can connect to the test device or create a simulated device for debugging data reporting' (您可以接入测试设备，或者创建模拟器调试数据上报).

设备类型选择“模拟设备”，点击“确定”；



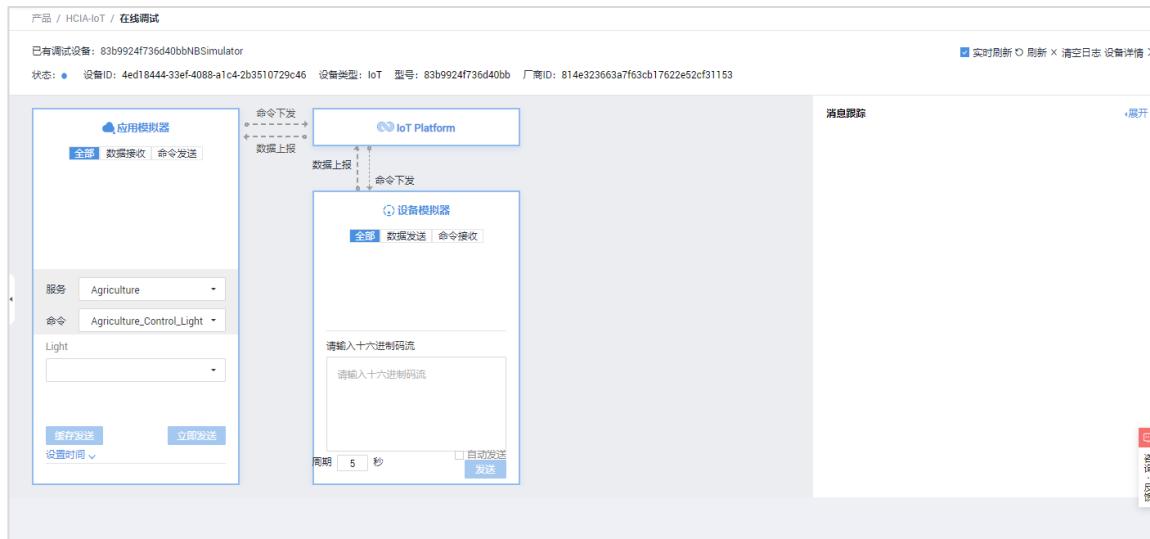
The screenshot shows the 'Add Test Device' dialog box. At the top, it says 'Add Test Device'. Below that, there are two tabs: 'Real Device' (真实设备) and 'Simulated Device' (模拟设备), with 'Simulated Device' being the active tab and highlighted with a blue background. In the center, a message says 'You are registering a virtual device' (您正在注册一个虚拟的设备). At the bottom, there are two buttons: a red 'Confirm' (确定) button and a white 'Cancel' (取消) button.

点击设备右侧的“调试”；



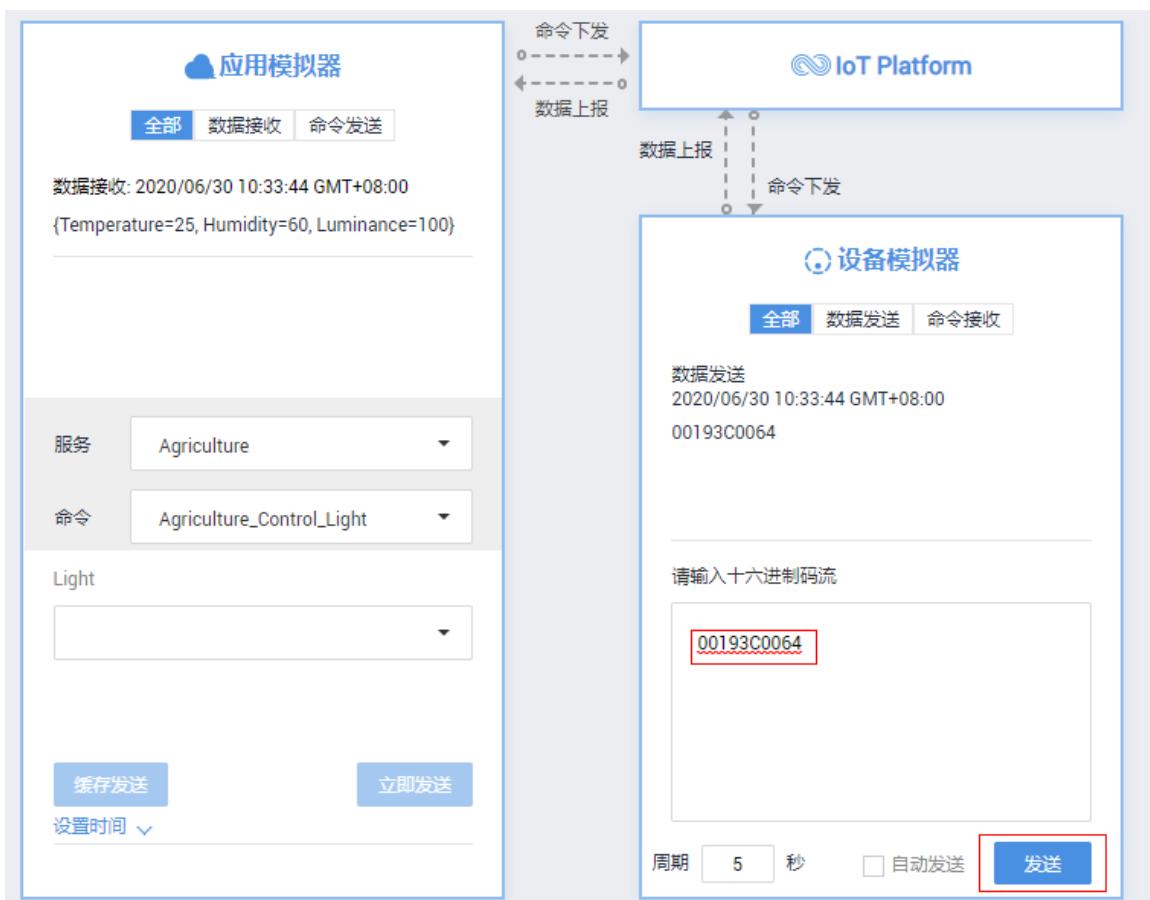
The screenshot shows the HCIA-IoT device management interface again. It displays the same device details as before. Below the tabs, the table now includes a column for 'Last Debug Time' (上次调试时间) and an 'Operations' (操作) column. A red arrow points from the text 'Click the "Debug" button on the right side of the device' (点击设备右侧的“调试”按钮) to the 'Debug' (调试) button in the 'Operations' column for the first device listed in the table.

进入在线调试界面；



步骤 2 数据上报调试

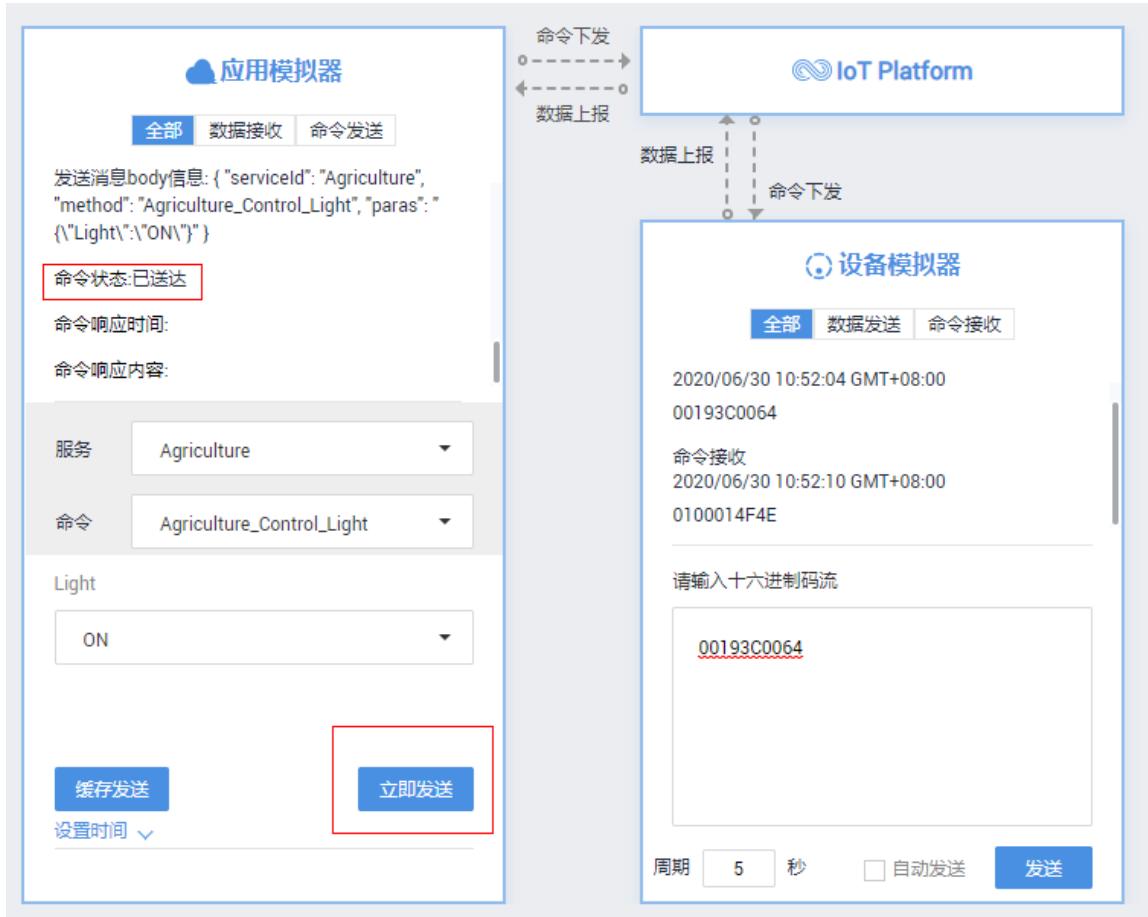
调测 Agriculture 消息，在文本框中输入“00193C0064”，点击“发送”；



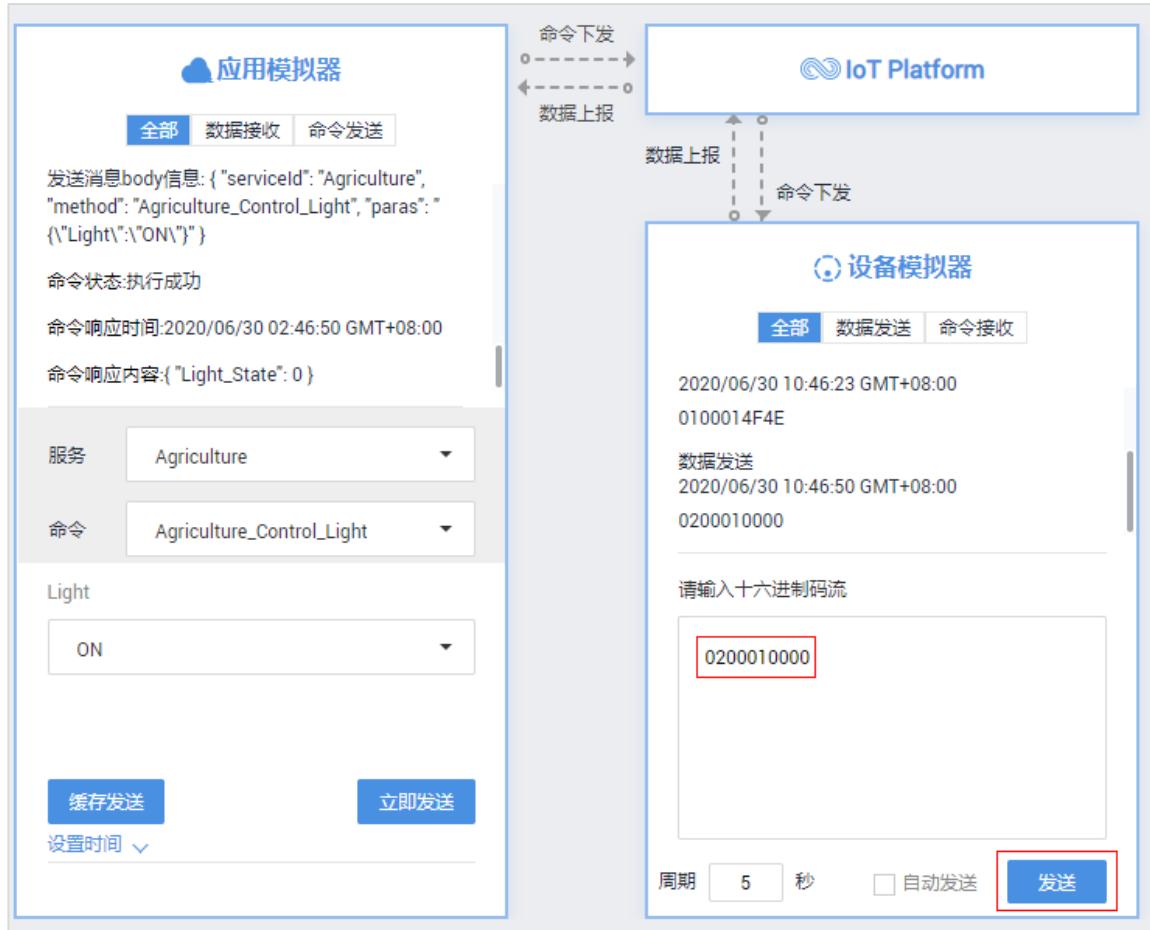
可以看到在应用模拟器中显示模拟数据接收“{ "Temperature": 25, "Humidity": 60, "Luminance": 100 }”；

步骤 3 Light 命令调试

服务选择“Agriculture”，命令选择“Agriculture_Control_Light”，Light 选择“ON”，点击“立即发送”，命令状态显示已送达；

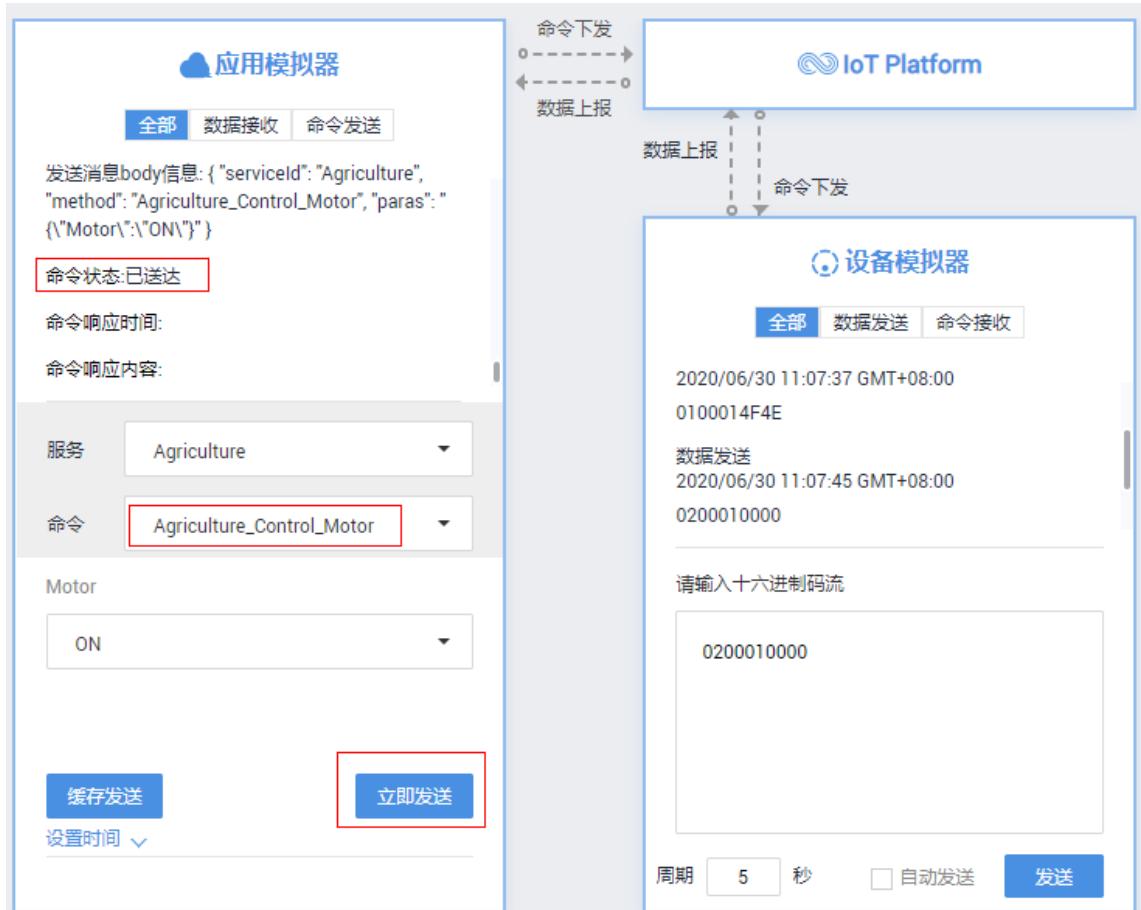


设备模拟器文本框中输入“0200010000”，点击“发送”，应用模拟器命令执行状态变为执行成功；

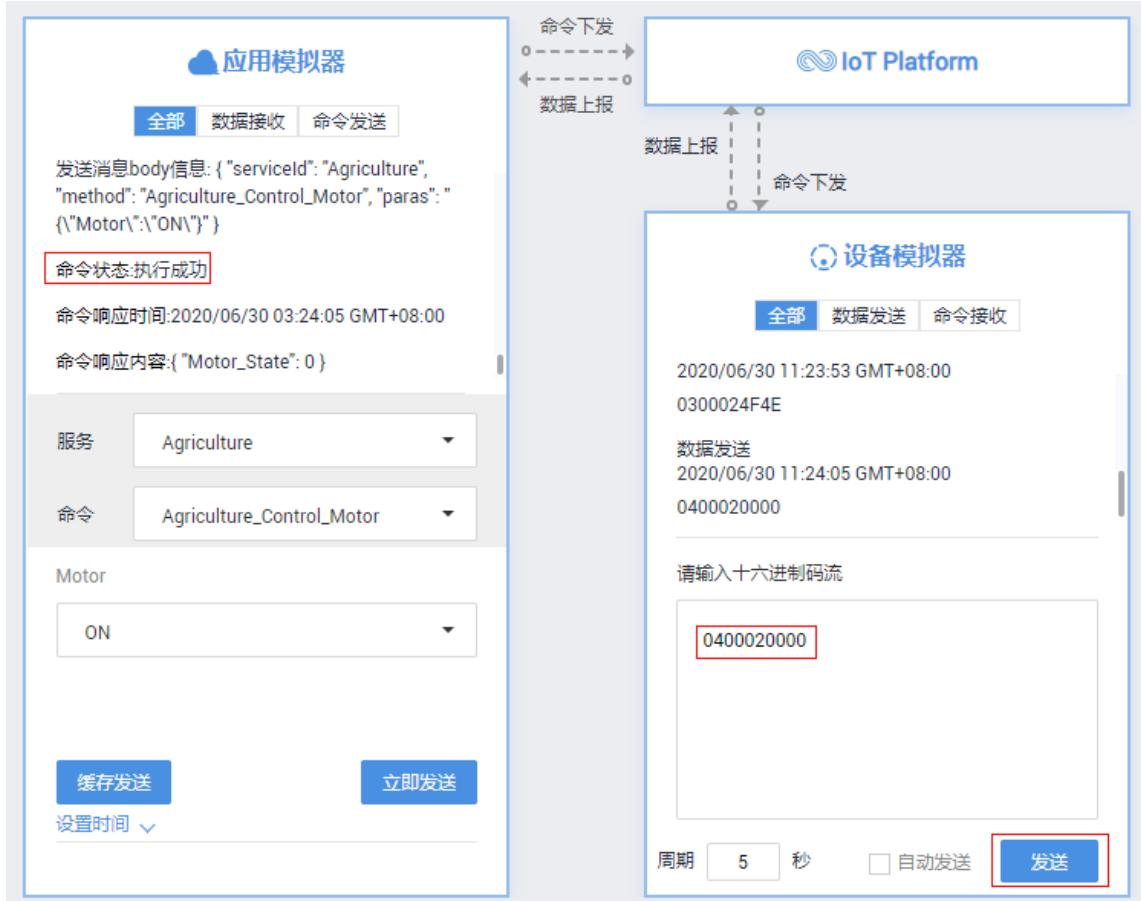


步骤 4 Motor 命令调试

服务选择“Agriculture”，命令选择“Agriculture_Control_Motor”，Motor 选择“ON”，点击“立即发送”，命令状态显示已送达；



设备模拟器文本框中输入“0400020000”，点击“发送”，应用模拟器命令执行状态变为执行成功；



1.3 练习题（在同一个产品中完成以下练习）

1.3.1 完成智慧烟感的平台侧开发

步骤 1 智慧烟感功能定义

表1-9 设计思路

| 服务描述 | 服务标识(serviceId) | 扩展板 |
|--------|-----------------|-----|
| 检测烟雾浓度 | Smoke | 烟感 |

表1-10 Smoke

| 能力描述 | 属性名称 | 数据类型 | 访问权限 | 数据范围 |
|------|-------------|------|-----------|---------|
| 属性列表 | Smoke_Value | int | 可读、可写、可执行 | 0~65535 |

表1-11 Smoke 命令列表

| 命令名称 | 命令字段 | 字段名称 | 类型 | 长度 | 枚举值 |
|--------------------|------|------------|--------|-----|--------|
| Smoke_Control_Beep | 下发命令 | Beep | string | 3 | ON,OFF |
| | 响应命令 | Beep_State | int | 0~1 | -- |

步骤 2 智慧烟感插件开发

表1-12 设计思路

| | 消息名 | 消息类型 | messageId |
|---|--------------------|------|-----------|
| 4 | Somke | 数据上报 | 05 |
| 5 | Somke_Control_Beep | 命令下发 | 06 |
| | | 响应 | 07 |

表1-13 Smoke

| 码流偏移值 | 0 | 1 | 2 |
|--------|-----------|-------------|----|
| 参数名 | messageId | Smoke_Value | |
| 数据类型 | int8u | int16u | |
| 长度 | 1 | 2 | |
| 16进制码流 | 05 | 00 | 3C |

表1-14 Smoke_Control_Beep

| 码流偏移值 | 0 | 1 | 2 | 3 | 4 | 5 | |
|---------|-----------|--------|----|---------|------------|----|--|
| 命令下发参数名 | messageld | mid | | Beep | | | |
| 数据类型 | int8u | int16u | | string | | | |
| 长度 | 1 | 2 | | 3 | | | |
| 16进制码流 | 06 | 00 | 01 | 4F | 4E | -- | |
| | | | | 4F | 46 | 46 | |
| 命令响应参数名 | messageld | mid | | errcode | Beep_State | -- | |
| 数据类型 | int8u | int16u | | int8u | int8u | -- | |
| 长度 | 1 | 2 | | 1 | 1 | -- | |
| 16进制码流 | 07 | 00 | 01 | 00/01 | 00/01 | -- | |

1.3.2 完成智慧物流的平台侧开发

步骤 1 智慧物流功能定义

表1-15 设计思路

| 服务描述 | 服务标识(serviceId) | 扩展板 |
|-------|-----------------|------|
| 检测经纬度 | Track | 智慧物流 |

表1-16 Track

| 能力描述 | 属性名称 | 数据类型 | 访问权限 | 数据范围 |
|------|-----------|---------|-----------|-------|
| 属性列表 | Longitude | decimal | 可读、可写、可执行 | 0~180 |
| | Latitude | decimal | 可读、可写、可执行 | 0~180 |

表1-17 Track 命令列表

| 命令名称 | 命令字段 | 字段名称 | 类型 | 长度 | 枚举值 |
|--------------------|------|------------|--------|-----|--------|
| Track_Control_Beep | 下发命令 | Beep | string | 3 | ON,OFF |
| | 响应命令 | Beep_State | int | 0~1 | -- |

步骤 2 智慧物流插件开发

表1-18 设计思路

| | 消息名 | 消息类型 | messageId |
|---|--------------------|------|-----------|
| 6 | Track | 数据上报 | 08 |
| 7 | Track_Control_Beep | 命令下发 | 09 |
| | | 响应 | 0a |

表1-19 Track

| 码流偏移值 | 参数名 | 数据类型 | 长度 | 16进制码流 |
|-------|-----------|--------|----|--------|
| 0 | messageId | int8u | 1 | 08 |
| 1 | Longitude | string | 9 | 31 |
| 2 | | | | 32 |
| 3 | | | | 30 |
| 4 | | | | 2E |
| 5 | | | | 31 |
| 6 | | | | 34 |
| 7 | | | | 32 |
| 8 | | | | 33 |
| 9 | | | | 30 |
| 10 | Latitude | string | 8 | 33 |
| 11 | | | | 30 |
| 12 | | | | 2E |
| 13 | | | | 31 |
| 14 | | | | 38 |
| 15 | | | | 33 |
| 16 | | | | 33 |
| 17 | | | | 33 |

表1-20 Track_Control_Beep

| 码流偏移值 | 0 | 1 | 2 | 3 | 4 | 5 | |
|---------|-----------|--------|----|---------|------------|----|--|
| 命令下发参数名 | messageld | mid | | Beep | | | |
| 数据类型 | int8u | int16u | | string | | | |
| 长度 | 1 | 2 | | 3 | | | |
| 16进制码流 | 09 | 00 | 01 | 4F | 4E | -- | |
| | | | | 4F | 46 | 46 | |
| 命令响应参数名 | messageld | mid | | errcode | Beep_State | -- | |
| 数据类型 | int8u | int16u | | int8u | int8u | -- | |
| 长度 | 1 | 2 | | 1 | 1 | -- | |
| 16进制码流 | 0a | 00 | 01 | 00/01 | 00/01 | -- | |

1.3.3 完成智慧井盖的平台侧开发

步骤 1 智慧井盖功能定义

表1-21 设计思路

| 服务描述 | 服务标识(serviceId) | 扩展板 |
|--------|-----------------|-----|
| 检测井盖状态 | Cover | 烟感 |

表1-22 Cover

| 能力描述 | 属性名称 | 数据类型 | 访问权限 | 数据范围 |
|------|-------------|--------|-----------|--------------|
| 属性列表 | Temperature | int | 可读、可写、可执行 | 0~65535 |
| | Accel_x | int | 可读、可写、可执行 | -65535~65535 |
| | Accel_y | int | 可读、可写、可执行 | -65535~65535 |
| | Accel_z | int | 可读、可写、可执行 | -65535~65535 |
| | status | String | 可读、可写、可执行 | 5 |

步骤 2 智慧井盖插件开发

表1-23 设计思路

| | 消息名 | 消息类型 | messageId |
|---|-------|------|-----------|
| 8 | Cover | 数据上报 | 0b |

表1-24 Cover

| 码流偏移值 | 0 | 1 | 2 | 3 | 4 | 5 |
|-------|-----------|-------------|---------|---------|---------|--------|
| 参数名 | messageId | Temperature | Accel_x | Accel_y | Accel_z | status |
| 数据类型 | int8u | int8u | int16s | int16s | int16s | string |
| 长度 | 1 | 1 | 2 | 2 | 2 | 5 |

1.3.4 完成人体感应的平台侧开发

步骤 1 人体感应功能定义

表1-25 设计思路

| 服务描述 | 服务标识(serviceId) | 扩展板 |
|--------|-----------------|------|
| 人体红外感测 | Infrared | 人体感应 |

表1-26 Infrared

| 能力描述 | 属性名称 | 数据类型 | 访问权限 | 数据范围 |
|------|--------|--------|-----------|------|
| 属性列表 | Status | String | 可读、可写、可执行 | 4 |

步骤 2 人体感应插件开发

表1-27 设计思路

| | 消息名 | 消息类型 | messageId |
|---|----------|------|-----------|
| 9 | Infrared | 数据上报 | 0c |

表1-28 Infrared

| | | |
|-------|-----------|--------|
| 码流偏移值 | 0 | 1 |
| 参数名 | messageId | Status |
| 数据类型 | int8u | String |
| 长度 | 1 | 4 |

1.3.5 基于模拟器完成智慧烟感、物流数据上报及命令响应

2 基于 NB-IoT 模组的 AT 指令实验

2.1 实验介绍

2.1.1 关于本实验

本实验使用 AT 指令，查询 NB-IoT 模组的 IMEI 号，配置模组入网，使用 AT 指令上报数据到平台以及进行命令的响应。

2.1.2 实验目的

- 掌握如何使用 AT 指令调试
- 掌握如何使用 AT 指令调试 NB-IoT 模组
- 掌握如何使用 AT 指令上报数据
- 掌握如何使用 AT 指令进行命令响应

2.2 实验任务配置步骤

2.2.1 在物联网平台中注册设备

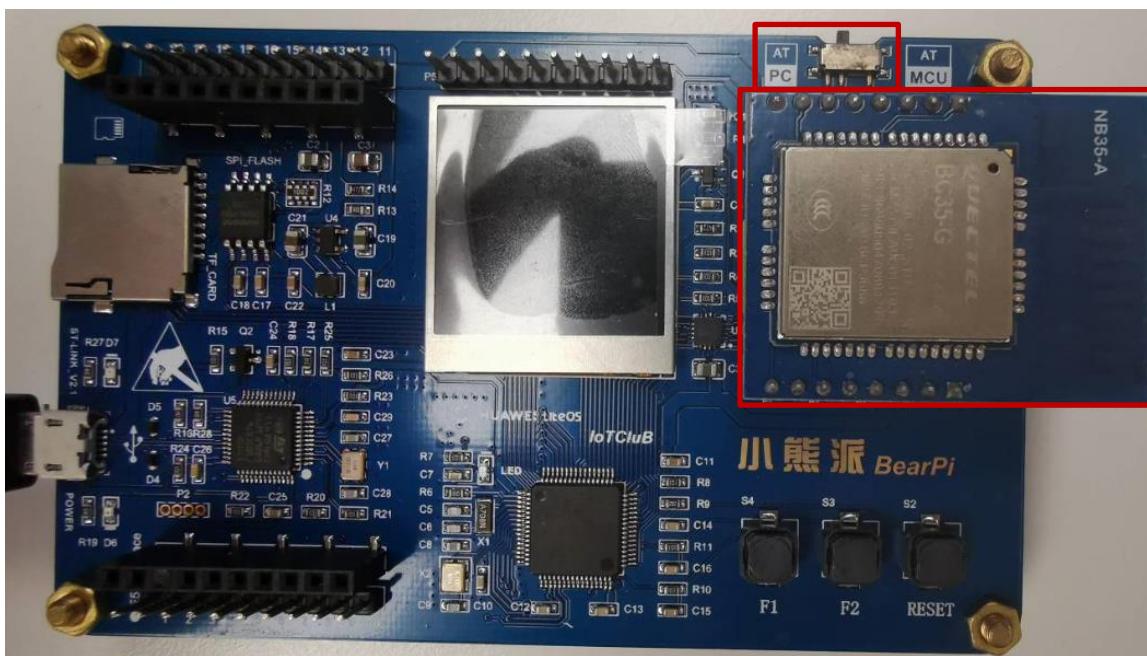
步骤 1 连接开发板

将 SIM 卡按照 NB35-A 通信扩展板上的提示插入扩展板背面的卡槽；



将 NB35-A 通信扩展板插入小熊派扩展板，天线朝外；

将串口模式的切换开关拨到 AT<->PC 模式（表示 NB-IoT 模组连接在 PC 机上）；

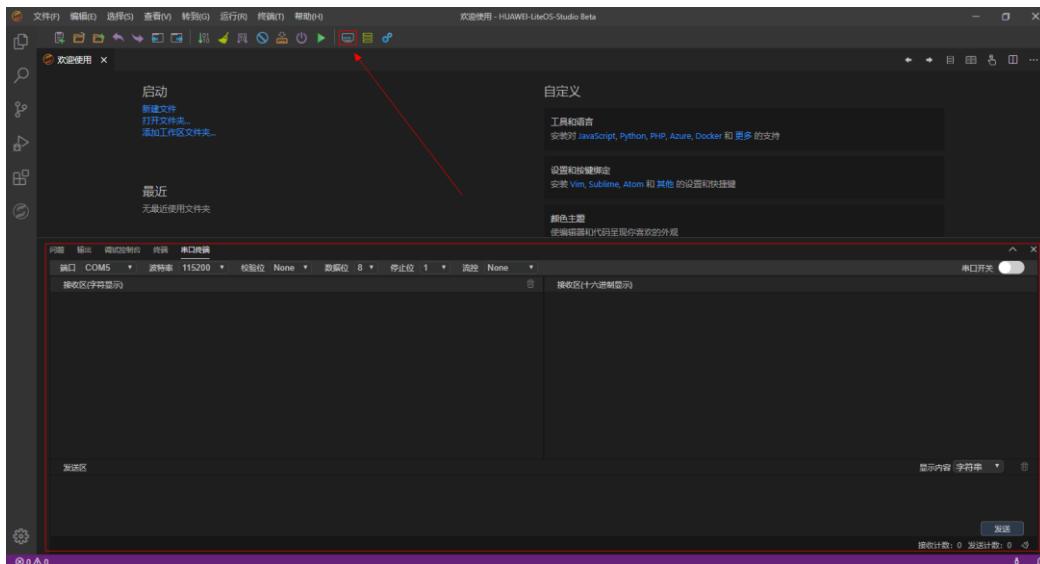


将开发板用 USB 线与电脑连接，右击“计算机”，点击“管理”，打开“计算机管理”中的“设备管理器”，点击“端口”，找到 STLink 的端口号；

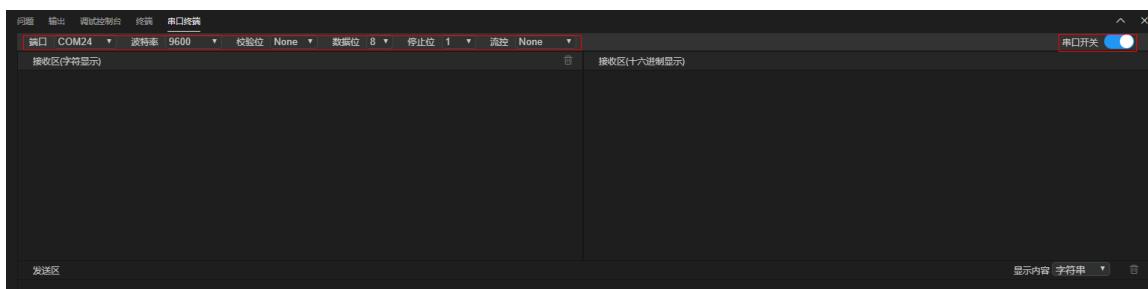


步骤 2 打开 LiteOS Studio 的串口终端工具

打开已安装好的 LiteOS Studio，点击工具栏的串口终端 ；



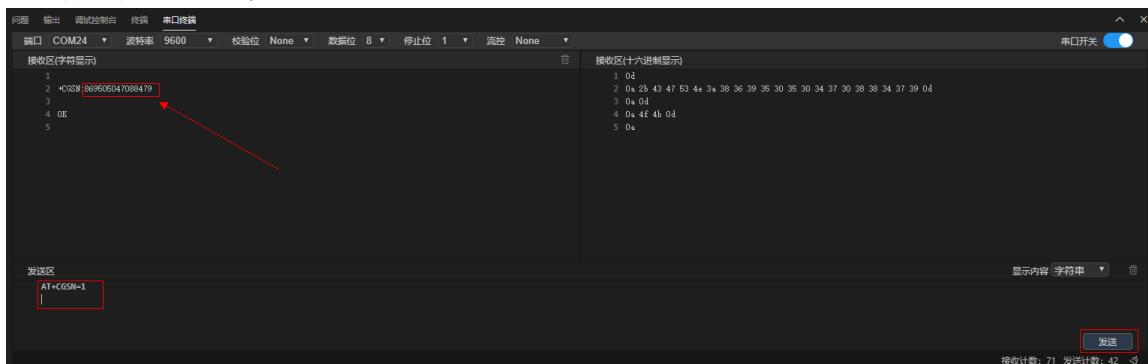
在串口终端中，选择对应的端口号，波特率 9600，校验位 None，数据位 8，停止位 1，流控 None，点击右侧串口开关按钮；



步骤 3 查询 NB-IoT 模组的 IMEI 号

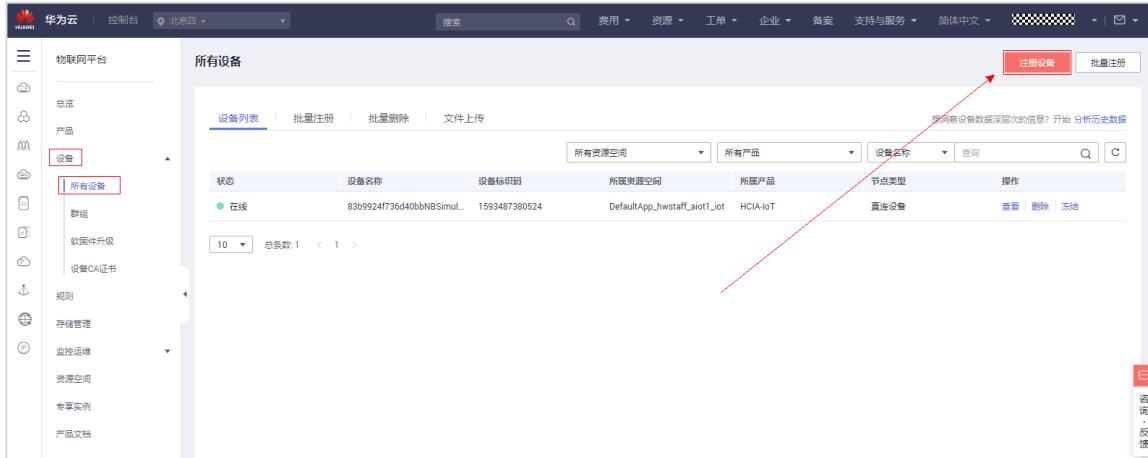
添加真实设备时，设备标识码必须使用 IMEI 号，可以通过“AT+CGSN=1”指令查询返回“+CGSN: 86XXXXXXXXXXXXXX”。（使用 NB 模组对接时，需填写模组的 IMEI 号。NB 模组的 IMEI 号通常为 15 位的数字，一般以 86 开头，刻于 NB 模组上）。

在发送区输入“AT+CGSN=1”，加回车换行，点击“发送”，接收区返回的值“+CGSN:”后面的 86 开头数字为 IMEI 号；



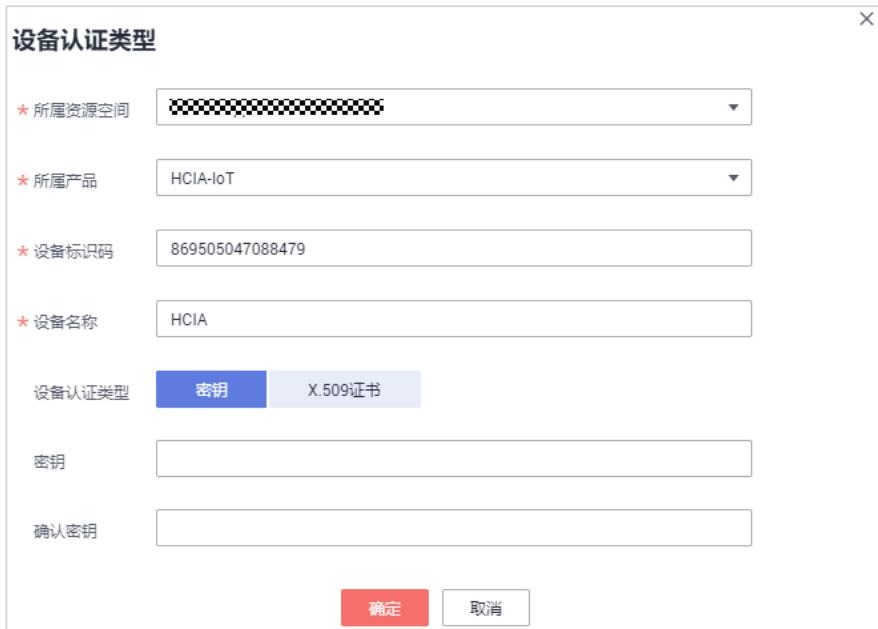
步骤 4 在平台中注册设备

打开华为云物联网平台，点击“设备”->“所有设备”->“注册设备”；



The screenshot shows the 'All Devices' page of the Huawei Cloud IoT Platform. On the left, there's a sidebar with various options like 'Overview', 'Products', 'Devices', and 'Groups'. Under 'Devices', 'All Devices' is selected. The main area displays a table of devices with columns: Status, Device Name, Device Serial Number, Associated Resource Space, Associated Product, Node Type, and Operations (View, Delete, Lock). One device is listed as 'Online'. At the top right of the page, there are buttons for 'Register Device' (highlighted with a red box and arrow) and 'Batch Register'.

使用默认资源空间，所属产品“HCIP-IoT”，设备标识码填写上一步查询到的 IMEI 号，设备名称自定义，点击“确定”；

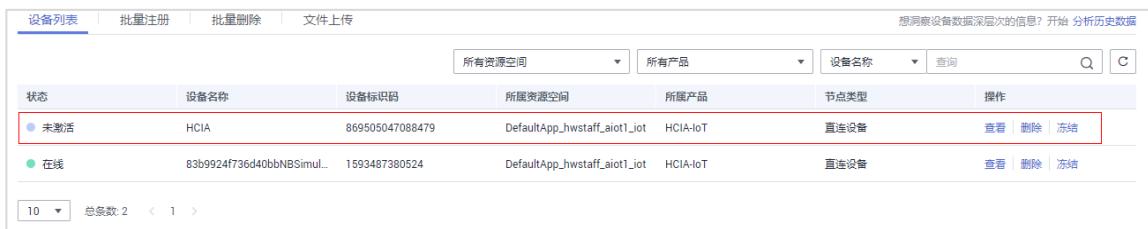


This is a modal dialog titled 'Device Authentication Type'. It contains the following fields:

- 所属资源空间: DefaultApp_hwstaff_iot1_iot
- 所属产品: HCIA-IoT
- 设备标识码: 869505047088479
- 设备名称: HCIA
- 设备认证类型: 密钥 (selected)
- 密钥: (empty input field)
- 确认密钥: (empty input field)

At the bottom are '确定' (Confirm) and '取消' (Cancel) buttons.

设备创建成功，在未上报数据之前，处于未激活状态；



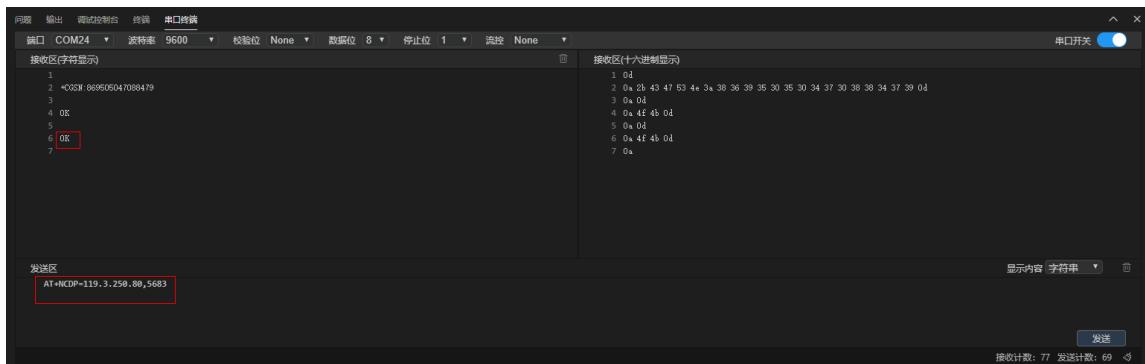
The screenshot shows the 'All Devices' page again. The newly created device 'HCIA' is listed in the table. The 'Status' column shows it as '未激活' (Not Activated). Other columns include 'Device Name', 'Device Serial Number', 'Associated Resource Space', 'Associated Product', 'Node Type', and 'Operations'. The 'Operations' column for this device shows '查看 | 删除 | 冻结' (View | Delete | Lock).

2.2.2 使用 AT 指令完成智慧农业的调测

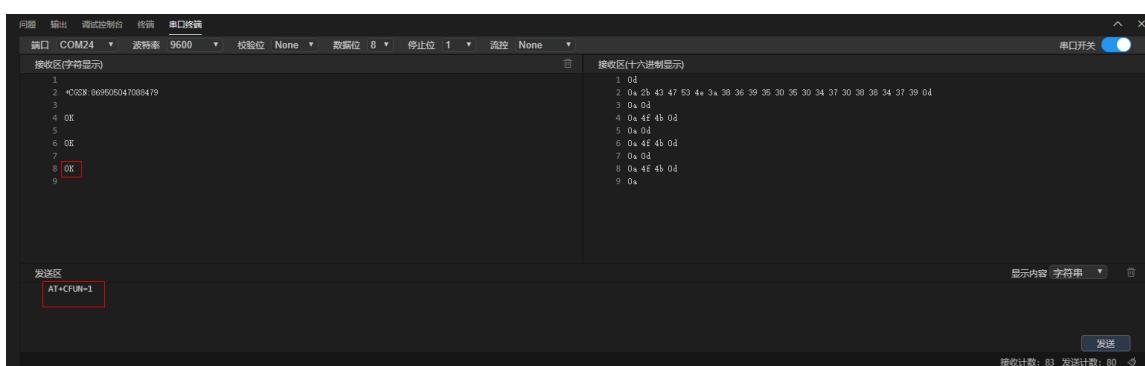
步骤 1 配置 NB-IoT 模组入网

获取物联网平台的 IP 和端口号 (IP 和端口号, 可以从平台上的“对接信息”获取, 此处华为云上的物联网平台 IP 地址为: 119.3.250.80, 端口号为: 5683);

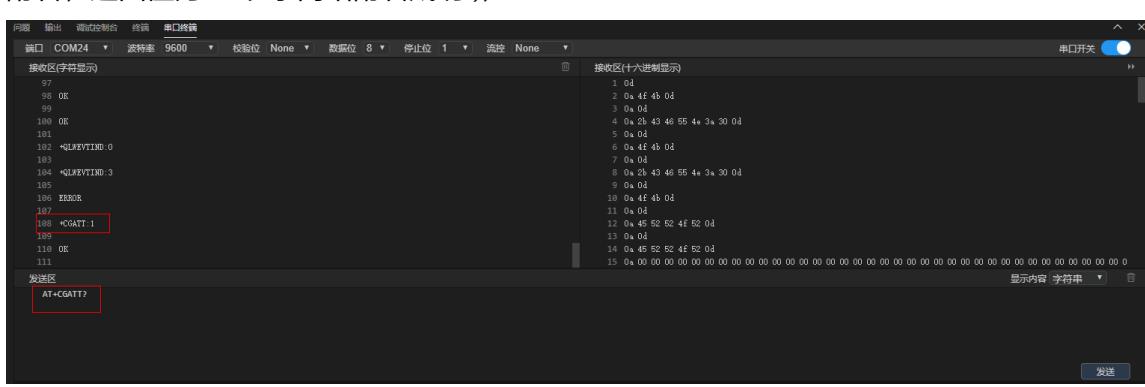
输入“AT+NCDP=119.3.250.80,5683”进行设置, 点击“发送”(返回“OK”表示设置成功);



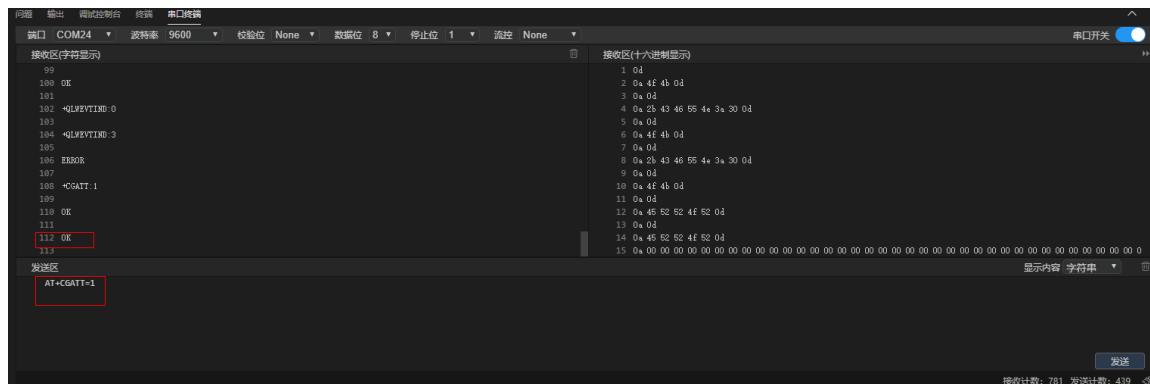
输入“AT+CFUN=1”, 点击“发送”, 打开协议栈功能 (返回“ok”表示打开成功);



输入“AT+CGATT?”, 点击“发送”, 查看 NB-IoT 模组网络附着状态 (返回值为“0”表示网络未附着, 返回值为“1”表示网络附着成功);



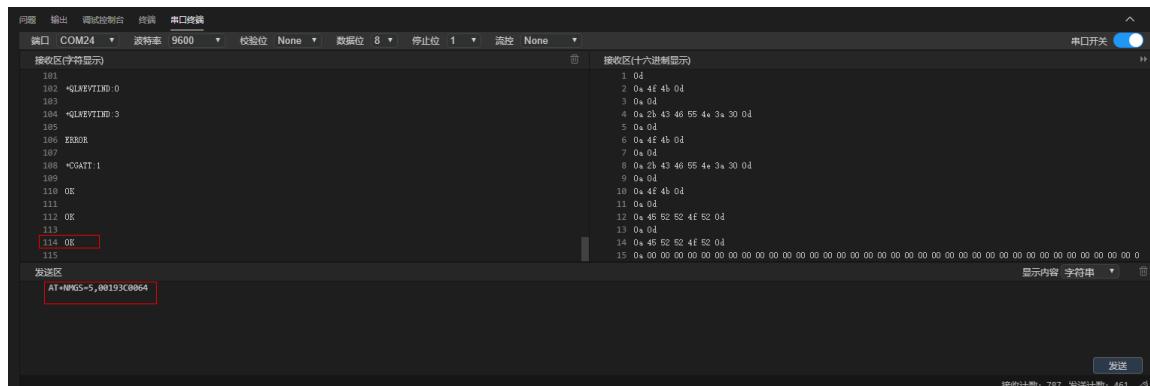
如果返回值为“0”，输入“AT+CGATT=1”，点击“发送”，进行网络附着（返回“OK”表示网络附着成功）。



步骤 2 上报数据

通过 NB-IoT 模组发送数据到华为物联网平台，

输入“AT+NMGS=5,00193C0064”，点击“发送”（返回“OK”表示发送成功）；



打开物联网平台，查看设备，可以看到之前添加的设备已在线；

| 状态 | 设备名称 | 设备标识码 | 所属资源空间 | 所属产品 | 节点类型 | 操作 |
|----|--------------------------|-----------------|------------------------------|----------|------|--|
| 在线 | HCIA | 869505047088479 | DefaultApp_hwstaff_aiot1_iot | HCIA-IoT | 直连设备 | 查看 删除 冻结 |
| 在线 | 83b9924f736d40bbNBsimul_ | 1593487380524 | DefaultApp_hwstaff_aiot1_iot | HCIA-IoT | 直连设备 | 查看 删除 冻结 |

点击设备后面的“查看”，可以查看到 NB-IoT 模组上报的数据。

设备管理 / 设备详情

概述 | 命令 | 设备影子 | 消息跟踪 | 子设备 | 标签

HCIA | 在线

| | | | |
|-------|--|--------|------------------------------|
| 设备标识码 | 869505047088479 | 设备名称 | HCIA |
| 设备ID | 5ef9b5a069c46102cb120886_869505047088479 | 认证类型 | 密钥 重置密钥 |
| 注册时间 | 2020/06/30 14:56:10 GMT+08:00 | 所属产品 | 5ef9b5a069c46102cb120886 |
| 节点类型 | 直连设备 | 固件版本 | - |
| 软件版本 | - | 所属资源空间 | DefaultApp_hwstaff_aiot1_iot |

最新上报数据

| Temperature | Humidity | Luminance |
|---------------------|---------------------|----------------------|
| 25 <Agriculture> | 60 <Agriculture> | 100 <Agriculture> |

步骤 3 命令响应

输入“AT+NNMI=1”，开启下行数据回显功能；

串口 COM24 波特率 9600 校验位 None 数据位 8 停止位 1 流控 None

接收区(字符显示) | 发送区

显示内容：字符串

发送区：AT+NNMI=1

接收区显示：

```

183
184 "+NNMI:TIND:3"
185
186 ERROR
187
188 "+CGATT:1"
189
190 OK
111
112 OK
113
114 OK
115
116 OK
117

```

点击“产品”->“HCIP-IoT”->“在线调试”->“调试”；

物联网平台 | **产品 / HCIA-IoT**

产品

设备

功能定义 | **插件开发** | **在线调试**

新增测试设备

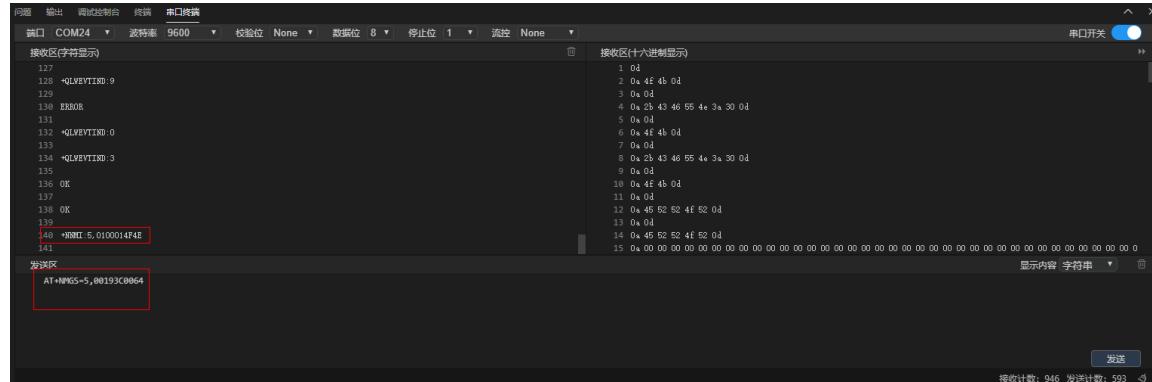
| 设备名称 | 设备标识码 | 设备ID | 类型 | 上次调试时间 | 操作 |
|-------------------------------|-----------------|---------------------------------------|------|-------------------------------|---------|
| HCIA | 869505047088479 | 5ef9b5a069c46102cb120886_869505047... | 真实设备 | 2020/07/01 09:41:46 GMT+08:00 | 测试 删除 |
| 83b9924f736d40bNbNBSimulat... | 1593487380524 | 1fb5d66b-43b3-4ada-b413-8f76d747558d | 虚拟设备 | 2020/06/30 11:23:01 GMT+08:00 | 测试 删除 |

在应用模拟器中，服务选择“**Agriculture**”，命令“**Agriculture_Control_Light**”，Light“ON”，点击“缓存发送”，可以查看到处于“等待”状态的命令；

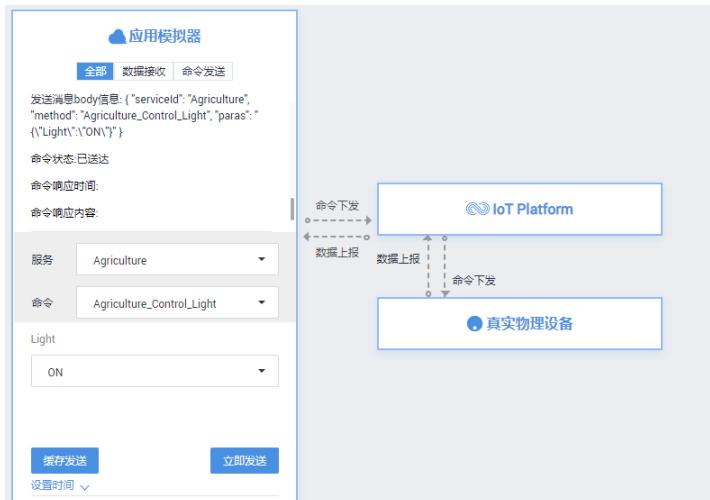


打开 LiteOS Studio，输入“AT+NMGS=5,00193C0064”，点击“发送”（ 返回“OK”表示发送成功）；

接收区回显下发命令的码流“+NNMI: 0100014F4E”，获取到 mid 值为“0001”；



点击在线设备的“历史命令”，可以查看到处于“已送达”状态的命令；



根据上报数据时，获取到 mid 值为 0001，输入命令响应“AT+NMGS=5,0200010000”；

```
139
140 +NMGI:5,0100014F4E
141
142 OK
143

发送区
AT+NMGS=5,0200010000
```

点击在线设备的“历史命令”，可以查看到处于“执行成功”状态的命令。



2.2.3 其他常用 AT 指令

获取信号强度：AT+CSQ

查询模块 IP 地址：AT+CGPADDR

2.3 练习题

2.3.1 使用 AT 指令完成智慧烟感、物流的调测

3

Huawei LiteOS 操作系统内核实验

3.1 实验介绍

3.1.1 关于本实验

本实验基于 LiteOS Studio 工具进行物联网终端的开发，使用 LiteOS 操作系统进行物联网开发板的控制。

3.1.2 实验目的

- 掌握 LiteOS Studio 的使用
- 掌握 LiteOS 操作系统任务的使用
- 熟悉 LCD 屏幕的使用
- 熟悉开发板的 LED 和按键使用

3.2 实验任务配置步骤

3.2.1 打开 LiteOS 工程

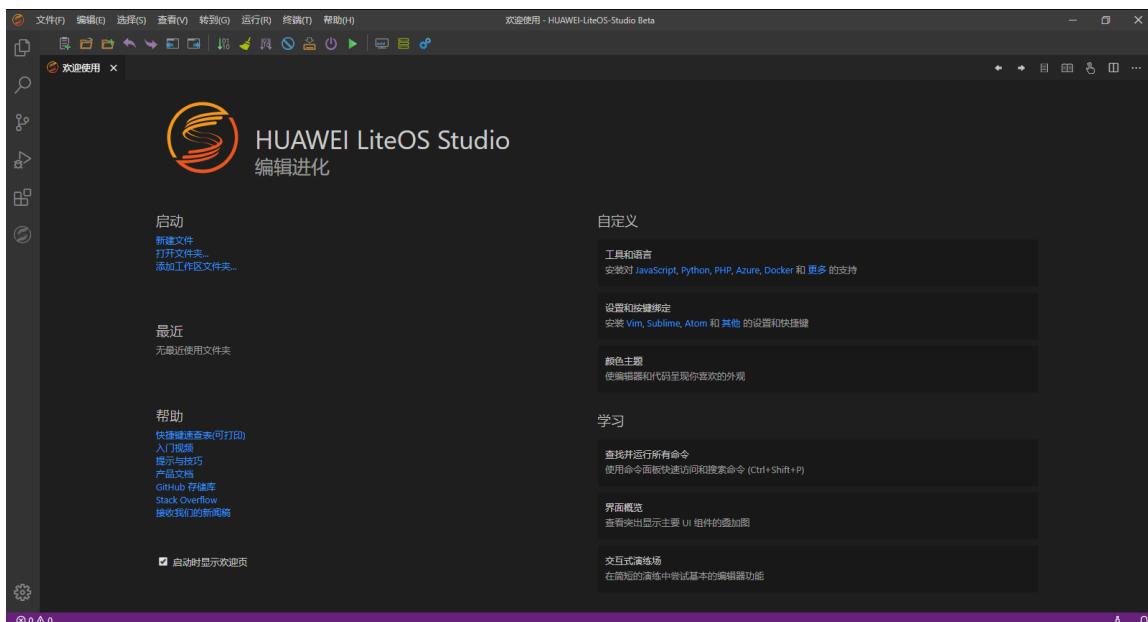
步骤 1 解压下载好的工程

将下载的 LiteOS_Lab_HCIP.rar，解压到任意磁盘根目录，路径中尽量不要出现中文或者空格。

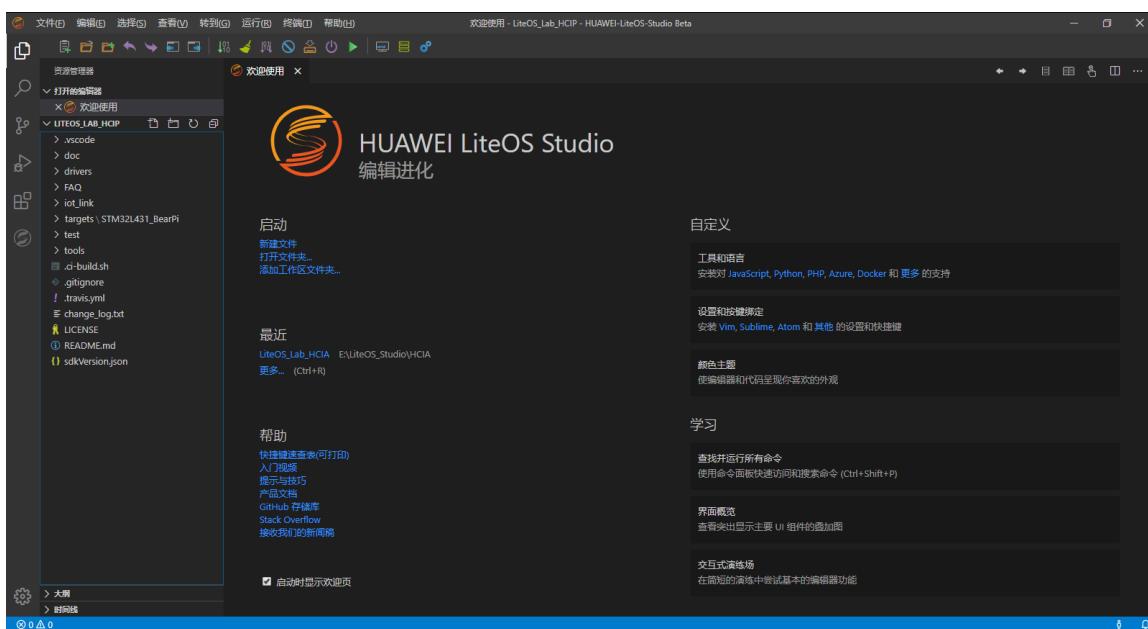


步骤 2 打开工程

打开安装好的 LiteOS Studio；



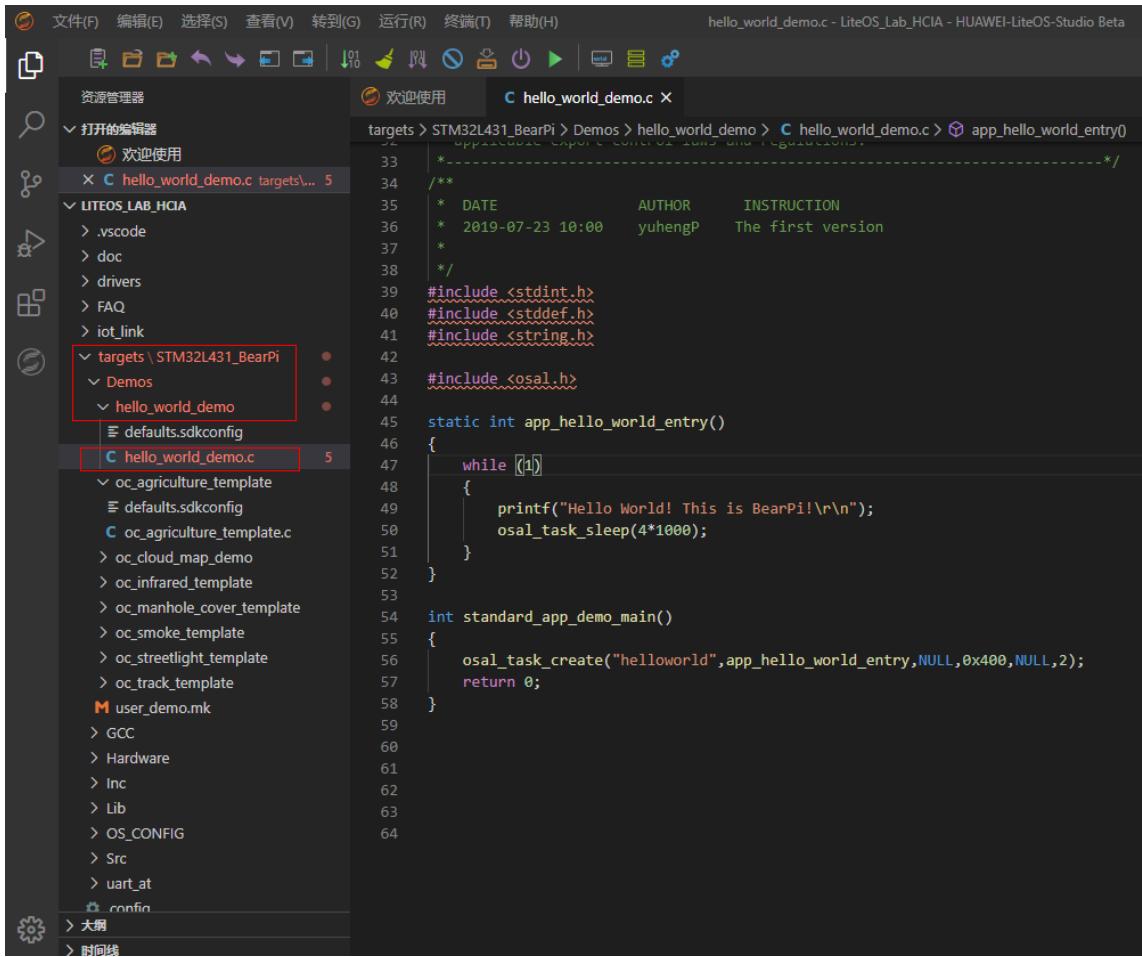
点击左上角“文件”->“打开文件夹”，选择解压好的文件夹“LiteOS_Lab_HCIP”；



3.2.2 运行 HelloWorld 任务

步骤 1 打开 hello_world_demo.c

打开 LiteOS_Lab_HCIP->targets->STM32L431_BearPi->
Demos->hello_world_demo->hello_world_demo.c 文件



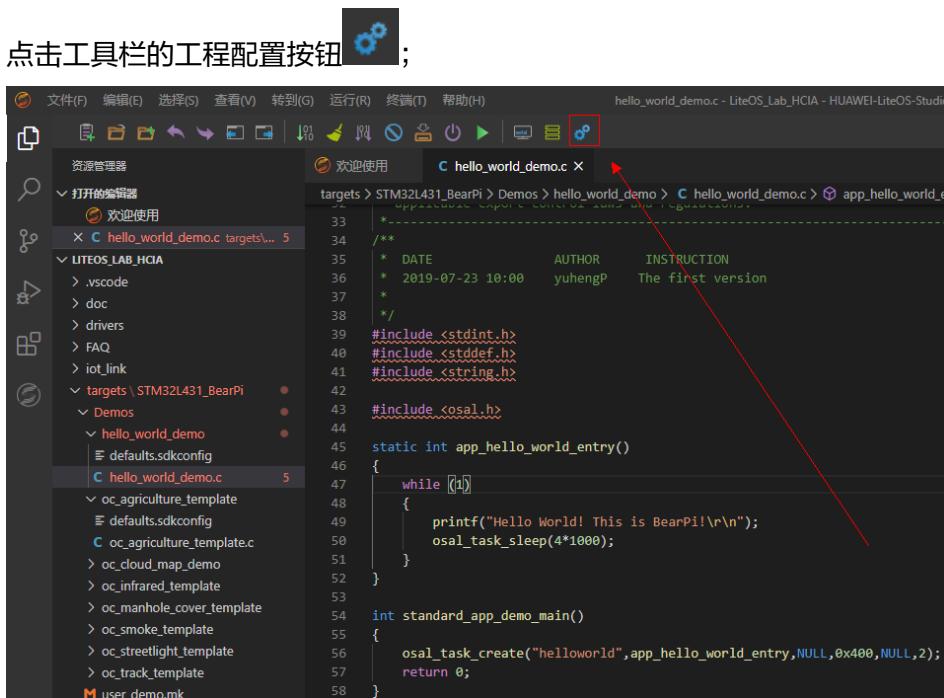
The screenshot shows the LiteOS Studio Beta interface. The left sidebar displays the project structure under 'LITEOS_LAB_HCIA' and 'targets \ STM32L431_BearPi'. The 'Demos' folder contains 'hello_world_demo'. The code editor on the right shows the 'hello_world_demo.c' file with the following content:

```

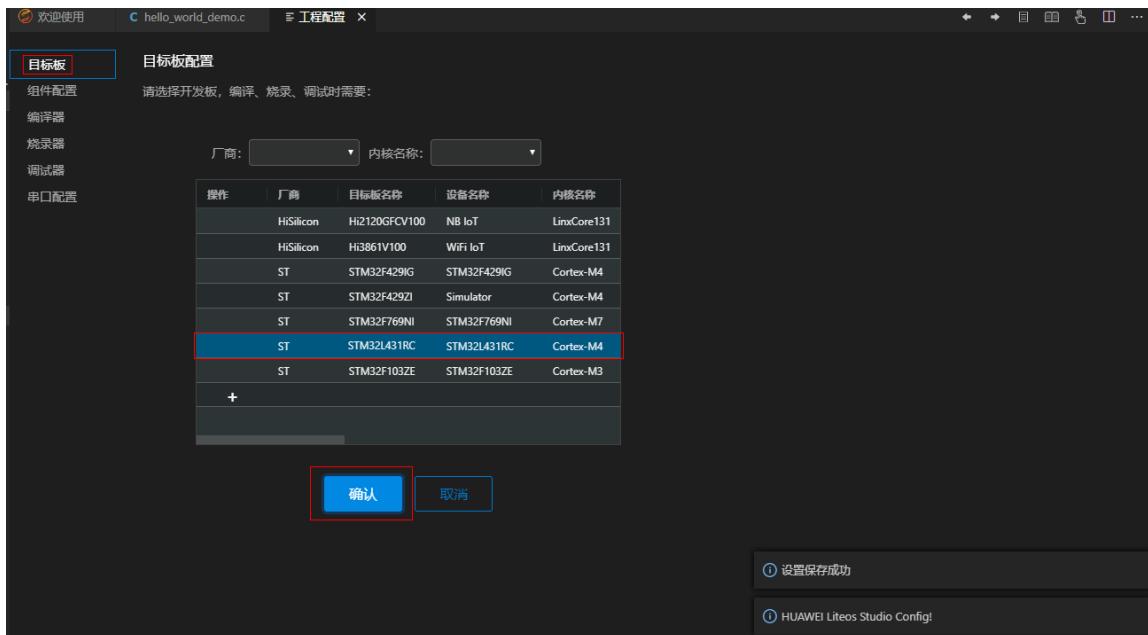
33  /*
34  */
35  * DATE           AUTHOR      INSTRUCTION
36  * 2019-07-23 10:00  yuhengP   The first version
37  *
38  */
39 #include <stdint.h>
40 #include <stddef.h>
41 #include <string.h>
42
43 #include <osal.h>
44
45 static int app_hello_world_entry()
46 {
47     while ([1])
48     {
49         printf("Hello World! This is BearPi!\r\n");
50         osal_task_sleep(4*1000);
51     }
52 }
53
54 int standard_app_demo_main()
55 {
56     osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,2);
57     return 0;
58 }

```

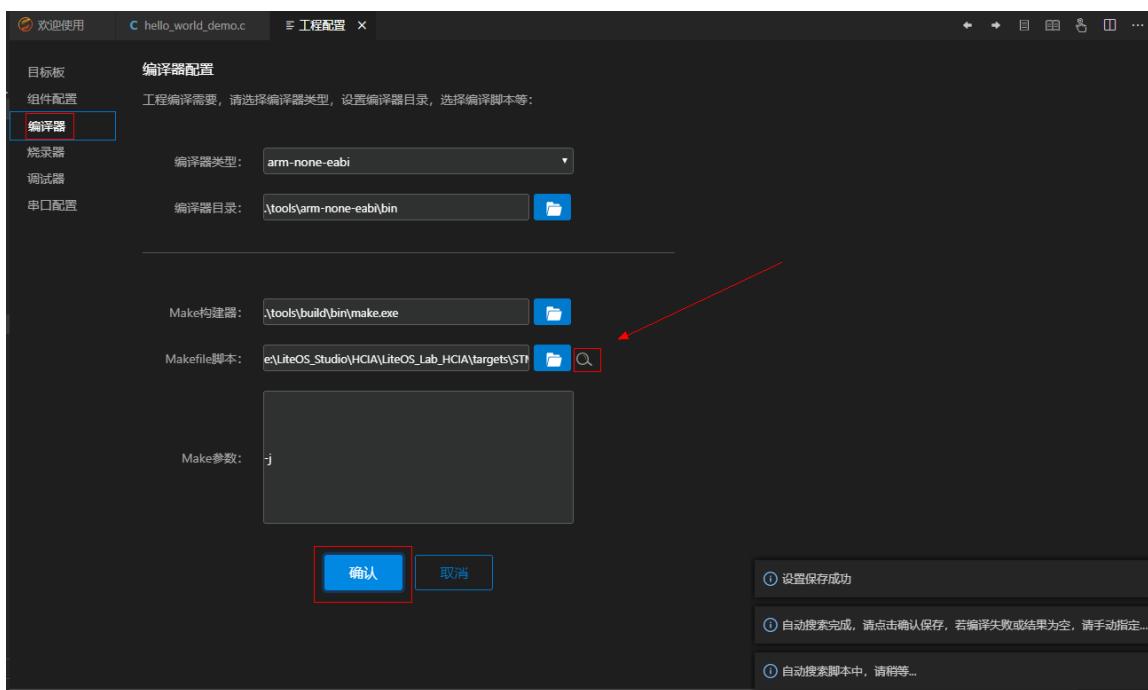
步骤 2 工程配置



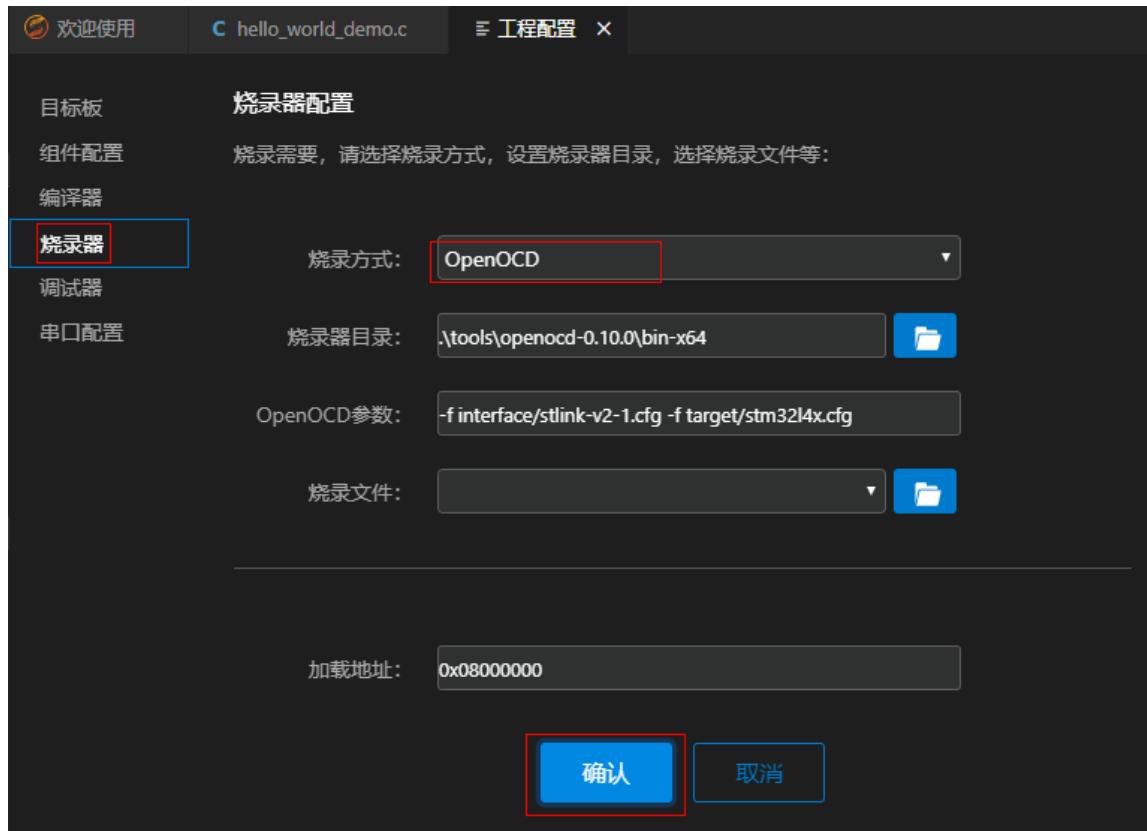
目标板选择“STM32L431RC”，点击“确认”；



点击“编译器”，Makefile 脚本点击  自动搜索，点击“确认”；Make 构建器需要根据您的安装目录来修改；



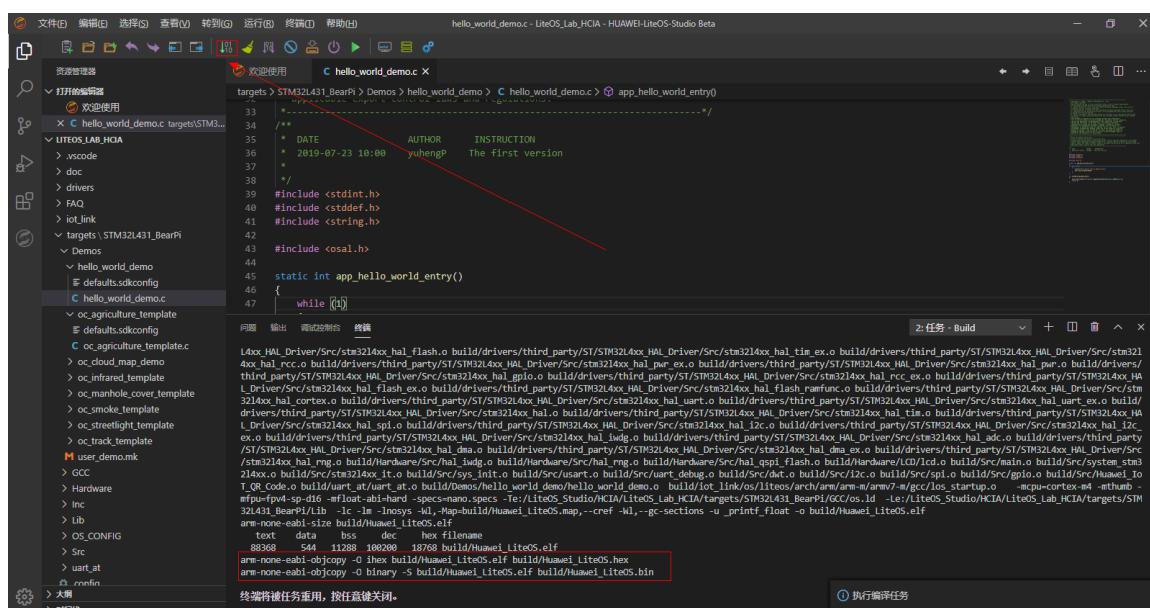
点击“烧录器”，烧录方式选择“OpenOCD”，点击“确认”；烧录器目录根据您安装目录来修改；



关闭工程配置 ；

步骤 3 编译程序

点击工具栏编译按钮 ，等待编译完成，会提示生成 Huawei_LiteOS.bin 文件；

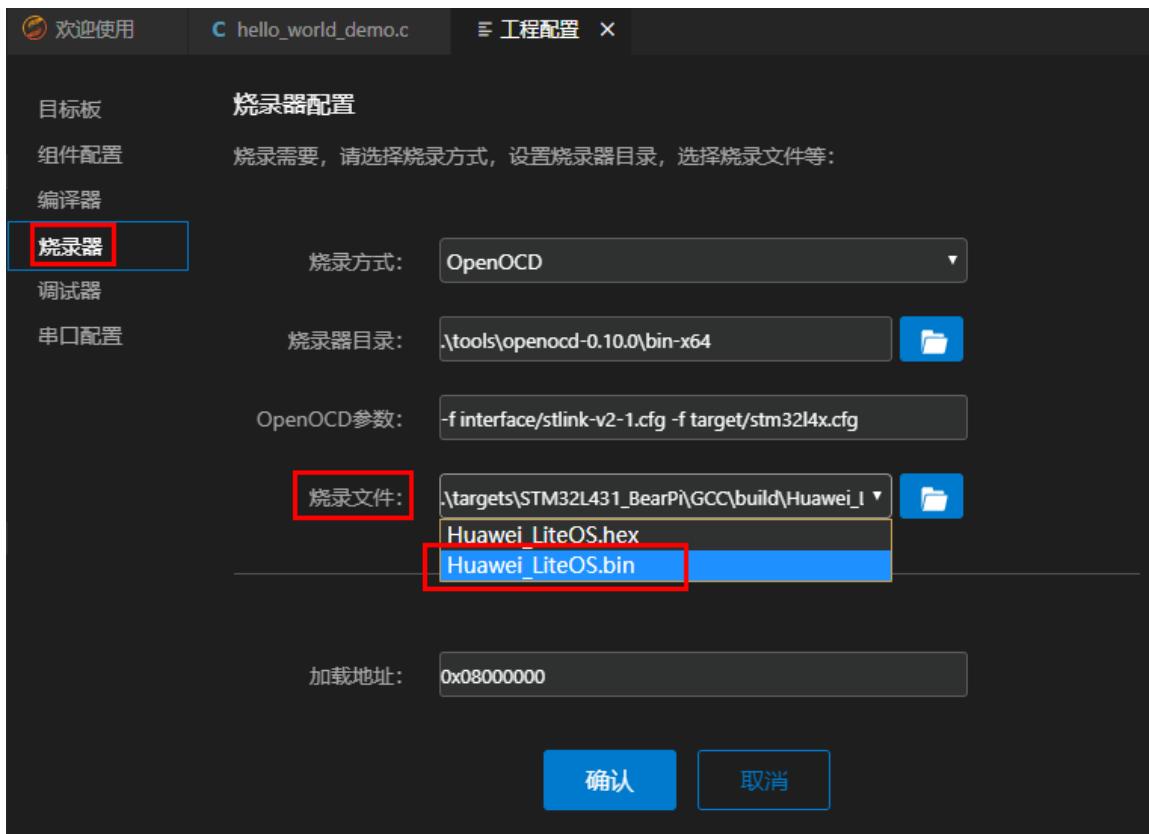


步骤 4 配置开发板

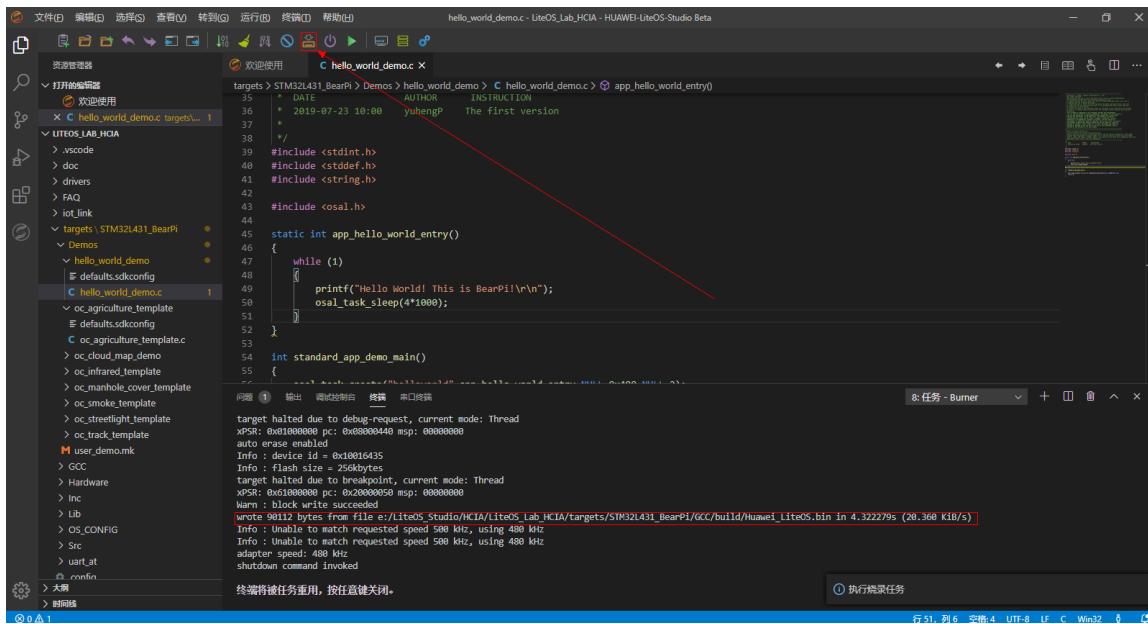
将开发板串口模式的切换开关拨到 AT<->MCU 模式（表示 NB-IoT 模组连接在 MCU 上）；
将开发板用 USB 线连接到电脑上；

步骤 5 烧录程序

打开工程配置，点击“烧录器”，烧录文件选择“Huawei_LiteOS.bin”，点击“确认”；

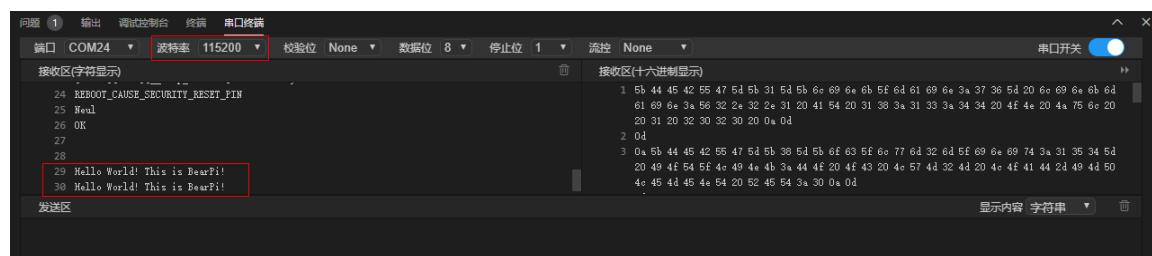


点击工具栏烧录按钮 ，等待烧录完成；



步骤 6 查看结果

打开串口终端，波特率选择“115200”，打开串口开关，可以看到接收区有“Hello World! This is BearPi!”显示；



3.2.3 任务管理实验

步骤 1 添加 task2 任务代码

在 hello_world_demo.c 中，添加 task2 任务执行的代码；

```

static int task2()
{
    while (1)
    {
        printf("This is Task2!\r\n");
        osal_task_sleep(4*1000);
    }
}
    
```



```
targets > STM32L431_BearPi > Demos > hello_world_demo > C hello_world_demo.c > ...
44
45     static int app_hello_world_entry()
46     {
47         while (1)
48         {
49             printf("Hello World! This is BearPi!\r\n");
50             osal_task_sleep(4*1000);
51         }
52     }
53
54     static int task2()
55     {
56         while (1)
57         {
58             printf("This is Task2!\r\n");
59             osal_task_sleep(4*1000);
60         }
61     }
62
```

步骤 2 创建 task2 任务

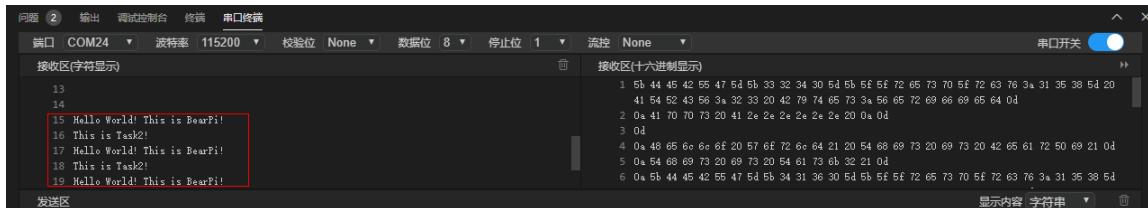
```
osal_task_create("task2",task2,NULL,0x400,NULL,2);

63     int standard_app_demo_main()
64     {
65         osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,2);
66         osal_task_create("task2",task2,NULL,0x400,NULL,2);
67         return 0;
68     }
```

步骤 3 编译烧录查看结果

编译烧录程序，打开串口终端，查看打印，

有“Hello World! This is BearPi!”和“This is Task2!”交替打印；



步骤 4 在 task2 中删除 Helloworld 任务

添加任务 ID；

```
void* task_id;

39  #include <stdint.h>
40  #include <stddef.h>
41  #include <string.h>
42
43  #include <osal.h>
44  void* task_id;
45  static int app_hello_world_entry()
46  {
47      while (1)
48      {
49          printf("Hello World! This is BearPi!\r\n");
50          osal_task_sleep(4*1000);
51      }
52 }
```

获取任务 ID；

```
task_id = osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,2);
```

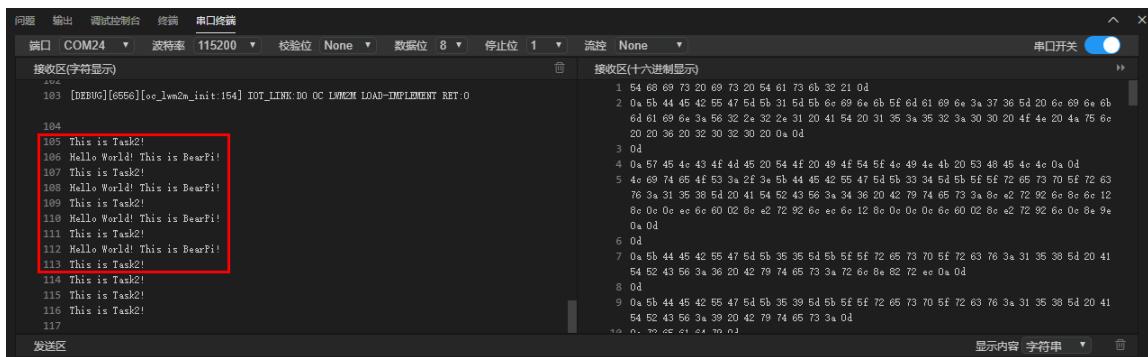
添加删除任务代码；

```
int num = 0;
while (1)
{
    printf("This is Task2!\r\n");
    num++;
    if(num == 5){
        osal_task_kill(task_id);
    }
    osal_task_sleep(4*1000);
}
```

```
54 static int task2()
55 {
56     int num = 0;
57     while (1)
58     {
59         printf("This is Task2!\r\n");
60         num++;
61         if(num == 5){
62             osal_task_kill(task_id);
63         }
64         osal_task_sleep(4*1000);
65     }
66 }
67 
```

步骤 5 编译烧录查看结果

编译烧录程序，打开串口终端，查看打印，task2 在打印 5 次后，Helloworld 停止打印；



3.2.4 互斥锁实验

步骤 1 添加 task1 代码

```
uint32_t public_value = 0;
osal_mutex_t public_value_mutex;
static int mutex_task1_entry()
{
    while(1)
    {
        if(true == osal_mutex_lock(public_value_mutex))
        {
            printf("\r\ntask1: lock a mutex.\r\n");
            public_value += 10;
            printf("task1: public_value = %ld.\r\n", public_value);
        }
    }
}
```

```
printf("task1: sleep...\r\n");
osal_task_sleep(10);
printf("task1: continue...\r\n");
printf("task1: unlock a mutex.\r\n\r\n");
osal_mutex_unlock(public_value_mutex);
if(public_value > 100)
break;
}
}

return 0;
}
```

```
69 uint32_t public_value = 0;
70 osal_mutex_t public_value_mutex;
71 static int mutex_task1_entry()
72 {
73     while(1)
74     {
75         if(true == osal_mutex_lock(public_value_mutex))
76         {
77             printf("\r\ntask1: lock a mutex.\r\n");
78             public_value += 10;
79             printf("task1: public_value = %ld.\r\n", public_value);
80             printf("task1: sleep...\r\n");
81             osal_task_sleep(10);
82             printf("task1: continue...\r\n");
83             printf("task1: unlock a mutex.\r\n\r\n");
84             osal_mutex_unlock(public_value_mutex);
85             if(public_value > 100)
86                 break;
87         }
88     }
89     return 0;
90 }
```

步骤 2 添加 task2 代码

```
static int mutex_task2_entry()
{
    while (1)
    {
        if(true == osal_mutex_lock(public_value_mutex))
        {
            printf("\r\ntask2: lock a mutex.\r\n");
            public_value += 5;
            printf("task2: public_value = %ld.\r\n", public_value);
            printf("task2: unlock a mutex.\r\n\r\n");
            osal_mutex_unlock(public_value_mutex);
            if(public_value > 90)
                break;
            osal_task_sleep(10);
        }
    }
    return 0;
}
```

```
91
92 static int mutex_task2_entry()
93 [
94     while (1)
95     {
96         if(true == osal_mutex_lock(public_value_mutex))
97         {
98             printf("\r\ntask2: lock a mutex.\r\n");
99             public_value += 5;
100            printf("task2: public_value = %ld.\r\n", public_value);
101            printf("task2: unlock a mutex.\r\n\r\n");
102            osal_mutex_unlock(public_value_mutex);
103            if(public_value > 90)
104                break;
105            osal_task_sleep(10);
106        }
107    }
108    return 0;
109 ]
110
111 int standard_app_demo_main()
112 {
```

步骤 3 注释掉任务管理实验任务创建代码

```
111 int standard_app_demo_main()
112 {
113     // task_id = osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,3);
114     // osal_task_create("task2",task2,NULL,0x400,NULL,2);
115     return 0;
116 }
```

步骤 4 创建互斥锁

```
osal_mutex_create(&public_value_mutex);
```

```
111 int standard_app_demo_main()
112 {
113     // task_id = osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,3);
114     // osal_task_create("task2",task2,NULL,0x400,NULL,2);
115
116     osal_mutex_create(&public_value_mutex);
117     return 0;
118 }
```

步骤 5 创建 task1 和 task2

```
osal_task_create("mutex_task1", mutex_task1_entry, NULL, 0x400, NULL, 12);
osal_task_create("mutex_task2", mutex_task2_entry, NULL, 0x400, NULL, 11);
```

```
111 int standard_app_demo_main()
112 {
113     // task_id = osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,3);
114     // osal_task_create("task2",task2,NULL,0x400,NULL,2);
115
116     osal_mutex_create(&public_value_mutex);
117     osal_task_create("mutex_task1", mutex_task1_entry, NULL, 0x400, NULL, 12);
118     osal_task_create("mutex_task2", mutex_task2_entry, NULL, 0x400, NULL, 11);
119
120     return 0;
121 }
```

步骤 6 编译烧录查看结果

编译烧录程序，打开串口终端，查看打印，task1 先创建，但是优先级较低，所以后创建的 task2 抢占执行，task2 获取到互斥锁，对共享资源进行操作，操作完毕解锁，然后主动挂起，task1 获取到互斥锁，对共享资源进行另一个操作，操作完毕主动挂起，task2 运行但是获取不到互斥锁，所以阻塞等待，在 task1 运行解锁后，阻塞的 task2 被唤醒开始执行。

```
173 task2: lock a mutex.  
174 task2: public_value = 5.  
175 task2: unlock a mutex.  
176  
177  
178 task1: lock a mutex.  
179 task1: public_value = 15.  
180 task1: sleep...  
181 task1: continue...  
182 task1: unlock a mutex.  
183  
184  
185 task2: lock a mutex.  
186 task2: public_value = 20.  
187 task2: unlock a mutex.  
188  
189  
190 task1: lock a mutex.  
191 task1: public_value = 30.  
192 task1: sleep...  
193 task1: continue...  
194 task1: unlock a mutex.
```

3.2.5 内存管理实验

步骤 1 添加 task 代码

```
static int mem_access_task_entry()  
{  
    uint32_t i = 0;      //循环变量  
    size_t mem_size;    //申请的内存块大小  
    uint8_t* mem_ptr = NULL; //内存块指针  
    while (1)  
    {  
        mem_size = 1 << i++; //每次循环将申请内存的大小扩大一倍  
        mem_ptr = osal_malloc(mem_size); //尝试申请分配内存  
        if(mem_ptr != NULL)  
        {  
            printf("access %d bytes memory success!\r\n", mem_size); // 申请成功  
            osal_free(mem_ptr); //释放申请的内存，便于下次申请  
            mem_ptr = NULL; //将内存块指针置为 NULL，避免称为野指针  
            printf("free memory success!\r\n");  
        }  
    }  
}
```

```
        else
        {
            /* 申请失败，打印信息，任务结束 */
            printf("access %d bytes memory failed!\r\n", mem_size);
            return 0;
        }
    }
}
```

```
111 static int mem_access_task_entry()
112 {
113     uint32_t i = 0;      //循环变量
114     size_t mem_size;    //申请的内存块大小
115     uint8_t* mem_ptr = NULL; //内存块指针
116     while (1)
117     {
118         mem_size = 1 << i++; //每次循环将申请内存的大小扩大一倍
119         mem_ptr = osal_malloc(mem_size); //尝试申请分配内存
120         if(mem_ptr != NULL)
121         {
122             printf("access %d bytes memory success!\r\n", mem_size); // 申请成功
123             osal_free(mem_ptr); //释放申请的内存，便于下次申请
124             mem_ptr = NULL; //将内存块指针置为NULL，避免称为野指针
125             printf("free memory success!\r\n");
126         }
127         else
128         {
129             /* 申请失败，打印信息，任务结束 */
130             printf("access %d bytes memory failed!\r\n", mem_size);
131             return 0;
132         }
133     }
134 }
135
136 int standard_app_demo_main()
137 {
```

步骤 2 注释掉互斥锁实验任务创建代码

```
136 int standard_app_demo_main()
137 {
138     // task_id = osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,3);
139     // osal_task_create("task2",task2,NULL,0x400,NULL,2);
140
141     // osal_mutex_create(&public_value_mutex);
142     // osal_task_create("mutex_task1", mutex_task1_entry, NULL, 0x400, NULL, 12);
143     // osal_task_create("mutex_task2", mutex_task2_entry, NULL, 0x400, NULL, 11);
144
145     return 0;
146 }
```

步骤 3 创建内存任务

```
osal_task_create("mem_access_task",mem_access_task_entry,NULL,0x400,NULL,11);
```

```
136 int standard_app_demo_main()
137 {
138     // task_id = osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,3);
139     // osal_task_create("task2",task2,NULL,0x400,NULL,2);
140
141     // osal_mutex_create(&public_value_mutex);
142     // osal_task_create("mutex_task1", mutex_task1_entry, NULL, 0x400, NULL, 12);
143     // osal_task_create("mutex_task2", mutex_task2_entry, NULL, 0x400, NULL, 11);
144     osal_task_create("mem_access_task",mem_access_task_entry,NULL,0x400,NULL,11);
145
146     return 0;
147 }
```

步骤 4 编译烧录查看结果

编译烧录程序，打开串口终端，查看打印如下。

```
241 access 1 bytes memory success!
242 free memory success!
243 access 2 bytes memory success!
244 free memory success!
245 access 4 bytes memory success!
246 free memory success!
247 access 8 bytes memory success!
248 free memory success!
249 access 16 bytes memory success!
250 free memory success!
251 access 32 bytes memory success!
252 free memory success!
253 access 64 bytes memory success!
254 free memory success!
```

3.2.6 信号量实验

步骤 1 添加 task1 代码

```
osal_semp_t sync_semp;
static int semp_task1_entry()
{
    printf("task 1 post a semp!\r\n");
    osal_semp_post(sync_semp);
    printf("task 1 end!\r\n");
}
```

```
136     osal_semp_t sync_semp;
137     static int semp_task1_entry()
138     {
139         printf("task 1 post a semp!\r\n");
140         osal_semp_post(sync_semp);
141         printf("task 1 end!\r\n");
142     }
```

步骤 2 添加 task2 代码

```
static int semp_task2_entry()
{
    printf("task2 is waiting for a semp...\r\n");
    osal_semp_pend(sync_semp, cn_osal_timeout_forever);
    printf("task 2 access a semp!\r\n");
}
```

```
144     static int semp_task2_entry()
145     {
146         printf("task2 is waiting for a semp...\r\n");
147         osal_semp_pend(sync_semp, cn_osal_timeout_forever);
148         printf("task 2 access a semp!\r\n");
149     }
150 }
```

步骤 3 注释掉内存管理实验任务创建代码

```
151 int standard_app_demo_main()
152 {
153     // task_id = osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,3);
154     // osal_task_create("task2",task2,NULL,0x400,NULL,2);
155
156     // osal_mutex_create(&public_value_mutex);
157     // osal_task_create("mutex_task1", mutex_task1_entry, NULL, 0x400, NULL, 12);
158     // osal_task_create("mutex_task2", mutex_task2_entry, NULL, 0x400, NULL, 11);
159     // osal_task_create("mem_access_task",mem_access_task_entry,NULL,0x400,NULL,11);
160
161     return 0;
162 }
```

步骤 4 创建信号量

```
osal_semp_create(&sync_semp, 1, 0);
printf("sync_semp semp create success.\r\n");
```

```
151 int standard_app_demo_main()
152 {
153     // task_id = osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,3);
154     // osal_task_create("task2",task2,NULL,0x400,NULL,2);
155
156     // osal_mutex_create(&public_value_mutex);
157     // osal_task_create("mutex_task1", mutex_task1_entry, NULL, 0x400, NULL, 12);
158     // osal_task_create("mutex_task2", mutex_task2_entry, NULL, 0x400, NULL, 11);
159     // osal_task_create("mem_access_task",mem_access_task_entry,NULL,0x400,NULL,11);
160     osal_semp_create(&sync_semp, 1, 0);
161     printf("sync_semp semp create success.\r\n");
162
163     return 0;
164 }
```

步骤 5 创建 task1 和 task2

```
osal_task_create("semp_task1",semp_task1_entry,NULL,0x400,NULL,12);
osal_task_create("semp_task2",semp_task2_entry,NULL,0x400,NULL,11);
```

```
151 int standard_app_demo_main()
152 {
153     // task_id = osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,3);
154     // osal_task_create("task2",task2,NULL,0x400,NULL,2);
155
156     // osal_mutex_create(&public_value_mutex);
157     // osal_task_create("mutex_task1", mutex_task1_entry, NULL, 0x400, NULL, 12);
158     // osal_task_create("mutex_task2", mutex_task2_entry, NULL, 0x400, NULL, 11);
159     // osal_task_create("mem_access_task",mem_access_task_entry,NULL,0x400,NULL,11);
160     osal_semp_create(&sync_semp, 1, 0);
161     printf("sync_semp semp create success.\r\n");
162     osal_task_create("semp_task1",semp_task1_entry,NULL,0x400,NULL,12);
163     osal_task_create("semp_task2",semp_task2_entry,NULL,0x400,NULL,11);
164
165     return 0;
166 }
```

步骤 6 编译烧录查看结果

编译烧录程序，打开串口终端，查看打印如下。

```
128 task2 is waiting for a semp...
129 task 1 post a semp!
130 task 2 access a semp!
131 task 1 end!
```

步骤 7 注释掉信号量实验任务创建代码

```
152 int standard_app_demo_main()
153 {
154     // task_id = osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,3);
155     // osal_task_create("task2",task2,NULL,0x400,NULL,2);
156
157     // osal_mutex_create(&public_value_mutex);
158     // osal_task_create("mutex_task1", mutex_task1_entry, NULL, 0x400, NULL, 12);
159     // osal_task_create("mutex_task2", mutex_task2_entry, NULL, 0x400, NULL, 11);
160     // osal_task_create("mem_access_task",mem_access_task_entry,NULL,0x400,NULL,11);
161     // osal_semp_create(&sync_semp, 1, 0);
162     // printf("sync_semp semp create success.\r\n");
163     // osal_task_create("semp_task1",semp_task1_entry,NULL,0x400,NULL,12);
164     // osal_task_create("semp_task2",semp_task2_entry,NULL,0x400,NULL,11);
165
166     return 0;
167 }
```

3.3 练习题

3.3.1 改变任务优先级，使 task2 任务优先于 Helloworld 打印

4 单片机基础实验

4.1 实验介绍

4.1.1 关于本实验

通过控制 LCD 屏幕和 LED 灯闪烁，了解开发板工作原理。

4.1.2 实验目的

- 实现板载 LCD 屏幕显示
- 学会板载 LED 灯闪烁
- GPIO 扫描检测板载按键控制 LED
- EXIT 检测板载按键控制 LED

4.2 实验任务配置步骤

4.2.1 板载 LCD 屏幕显示

步骤 1 导入 LCD 驱动头文件

```
#include "lcd.h"
```

```
39  #include <stdint.h>
40  #include <stddef.h>
41  #include <string.h>
42
43  #include <osal.h>
44  #include "lcd.h"
45  void *task_id;
46  static int app_hello_world_entry()
47  {
48      while (1)
49      {
50          printf("Hello World! This is BearPi!\r\n");
51          osal_task_sleep(4 * 1000);
52      }
53 }
```

步骤 2 添加清除 LCD 屏显示代码

```
LCD_Clear(BLACK);
```

```
152 int standard_app_demo_main()
153 {
154     LCD_Clear(BLACK);
155     // task_id = osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,3);
156     // osal_task_create("task2",task2,NULL,0x400,NULL,2);
157
158     // osal_mutex_create(&public_value_mutex);
159     // osal_task_create("mutex_task1", mutex_task1_entry, NULL, 0x400, NULL, 12);
160     // osal_task_create("mutex_task2", mutex_task2_entry, NULL, 0x400, NULL, 11);
161     // osal_task_create("mem_access_task",mem_access_task_entry,NULL,0x400,NULL,11);
162     // osal_semp_create(&sync_semp, 1, 0);
163     // printf("sync_semp semp create success.\r\n");
164     // osal_task_create("semp_task1",semp_task1_entry,NULL,0x400,NULL,12);
165     // osal_task_create("semp_task2",semp_task2_entry,NULL,0x400,NULL,11);
166
167     return 0;
168 }
```

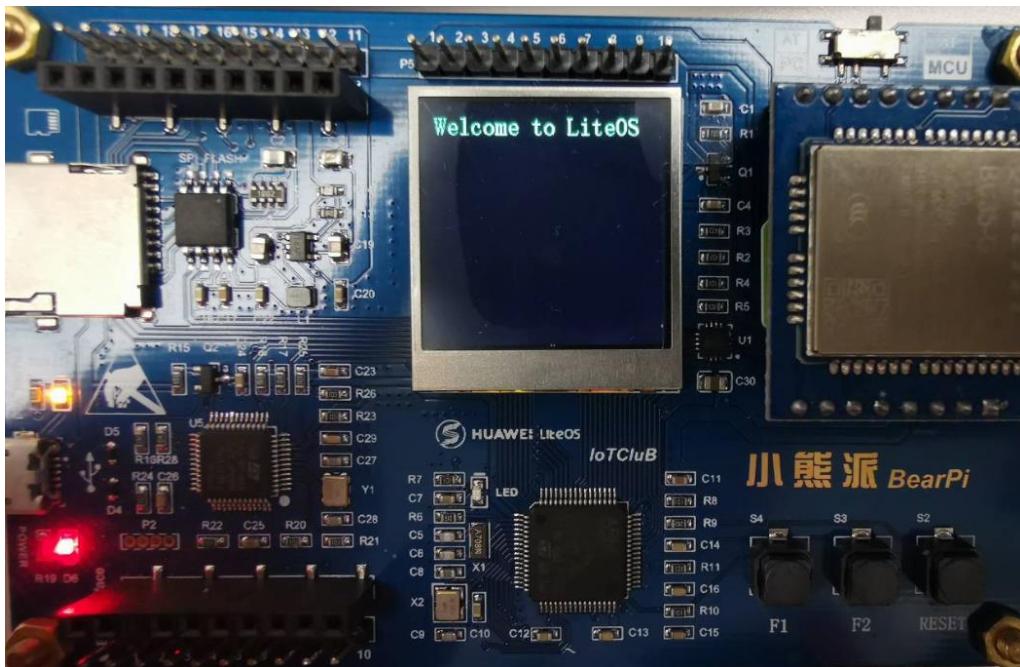
步骤 3 添加 LCD 屏显示代码

```
POINT_COLOR = GREEN;
LCD_ShowString(10, 10, 200, 16, 24, "Welcome to LiteOS");
```

```
152 int standard_app_demo_main()
153 {
154     LCD Clear(BLACK);
155     POINT_COLOR = GREEN;
156     LCD_ShowString(10, 10, 200, 16, 24, "Welcome to LiteOS");
157
158     // task_id = osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,3);
159     // osal_task_create("task2",task2,NULL,0x400,NULL,2);
160 }
```

步骤 4 编译烧录查看结果

编译烧录程序，查看小熊派 LCD 屏幕显示“Welcome to LiteOS”字样；



4.2.2 板载 LED 灯闪烁

步骤 1 添加 LED 闪烁任务代码

```
static int led_task()
{
    GPIO_InitTypeDef GPIO_InitStruct;
    GPIO_InitStruct.Pin = GPIO_PIN_13;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
    while (1)
    {
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
        osal_task_sleep(1*1000);
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
        osal_task_sleep(1*1000);
    }
}
```

```
151
152 static int led_task()
153 {
154     GPIO_InitTypeDef GPIO_InitStruct;
155     GPIO_InitStruct.Pin = GPIO_PIN_13;
156     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
157     GPIO_InitStruct.Pull = GPIO_NOPULL;
158     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
159     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
160     while (1)
161     {
162         HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
163         osal_task_sleep(1*1000);
164         HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
165         osal_task_sleep(1*1000);
166     }
167 }
168
169 int standard_app_demo_main()
```

步骤 2 创建 LED 闪烁任务

```
osal_task_create("led_task",led_task,NULL,0x400,NULL,2);
```

```
169 int standard_app_demo_main()
170 {
171     LCD_Clear(BLACK);
172     POINT_COLOR = GREEN;
173     LCD_ShowString(10, 10, 200, 16, 24, "Welcome to LiteOS");
174
175     // task_id = osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,3);
176     // osal_task_create("task2",task2,NULL,0x400,NULL,2);
177
178     // osal_mutex_create(&public_value_mutex);
179     // osal_task_create("mutex_task1", mutex_task1_entry, NULL, 0x400, NULL, 12);
180     // osal_task_create("mutex_task2", mutex_task2_entry, NULL, 0x400, NULL, 11);
181     // osal_task_create("mem_access_task",mem_access_task_entry,NULL,0x400,NULL,11);
182     // osal_semp_create(&sync_semp, 1, 0);
183     // printf("sync_semp semp create success.\r\n");
184     // osal_task_create("semp_task1",semp_task1_entry,NULL,0x400,NULL,12);
185     // osal_task_create("semp_task2",semp_task2_entry,NULL,0x400,NULL,11);
186     osal_task_create("led_task",led_task,NULL,0x400,NULL,2);
187
188     return 0;
189 }
```

步骤 3 编译烧录查看结果

编译烧录程序，可以看到小熊派板载 LED 灯闪烁。

4.2.3 GPIO 扫描检测板载按键控制 LED

步骤 1 注释掉 LED 闪烁循环代码

```
152 static int led_task()
153 {
154     GPIO_InitTypeDef GPIO_InitStruct;
155     GPIO_InitStruct.Pin = GPIO_PIN_13;
156     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
157     GPIO_InitStruct.Pull = GPIO_NOPULL;
158     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
159     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
160     while (1)
161     {
162         // HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
163         // osal_task_sleep(1*1000);
164         // HAL_GPIO_WritePin(GPIOC,GPIO_PIN_13,GPIO_PIN_RESET);
165         // osal_task_sleep(1*1000);
166     }
167 }
```

步骤 2 添加按键判断代码

```
if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_2)==GPIO_PIN_RESET)//查询按键
KEY1 低电平
{
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
}
else if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_3)==GPIO_PIN_RESET)//查询按键
KEY2 低电平
{
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
}
```

```
152 static int led_task()
153 {
154     GPIO_InitTypeDef GPIO_InitStruct;
155     GPIO_InitStruct.Pin = GPIO_PIN_13;
156     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
157     GPIO_InitStruct.Pull = GPIO_NOPULL;
158     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
159     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
160     while (1)
161     {
162         // HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
163         // osal_task_sleep(1*1000);
164         // HAL_GPIO_WritePin(GPIOC,GPIO_PIN_13,GPIO_PIN_RESET);
165         // osal_task_sleep(1*1000);
166         if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_2)==GPIO_PIN_RESET)//查询按键KEY1低电平
167         {
168             HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
169         }else if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_3)==GPIO_PIN_RESET)//查询按键KEY2低电平
170         {
171             HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
172         }
173     }
174 }
```

步骤 3 编译烧录，查看结果

编译烧录程序，按下开发板右下角 F1 按键 LED 灯开，按下 F2 按键 LED 灯关；

4.2.4 EXIT 检测板载按键控制 LED

步骤 1 添加 key1 中断处理函数

```
static Key1_interrupt_entry()
{
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
    __HAL_GPIO_EXTI_CLEAR_FLAG(GPIO_PIN_2);
}

176 static Key1_interrupt_entry()
177 {
178     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
179     __HAL_GPIO_EXTI_CLEAR_FLAG(GPIO_PIN_2);
180 }
181
```

步骤 2 添加 key2 中断处理函数

```
static Key2_interrupt_entry()
{
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
    __HAL_GPIO_EXTI_CLEAR_FLAG(GPIO_PIN_3);
}
```

```
182 static Key2_interrupt_entry()
183 {
184     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
185     __HAL_GPIO_EXTI_CLEAR_FLAG(GPIO_PIN_3);
186 }
187
```

步骤 3 注释掉 LED 灯闪烁任务创建代码

```
188 int standard_app_demo_main()
189 {
190     LCD_Clear(BLACK);
191     POINT_COLOR = GREEN;
192     LCD_ShowString(10, 10, 200, 16, 24, "Welcome to LiteOS");
193
194 // task_id = osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,3);
195 // osal_task_create("task2",task2,NULL,0x400,NULL,2);
196
197 // osal_mutex_create(&public_value_mutex);
198 // osal_task_create("mutex_task1", mutex_task1_entry, NULL, 0x400, NULL, 12);
199 // osal_task_create("mutex_task2", mutex_task2_entry, NULL, 0x400, NULL, 11);
200 // osal_task_create("mem_access_task",mem_access_task_entry,NULL,0x400,NULL,11);
201 // osal_semp_create(&sync_semp, 1, 0);
202 // printf("sync_semp semp create success.\r\n");
203 // osal_task_create("semp_task1",semp_task1_entry,NULL,0x400,NULL,12);
204 // osal_task_create("semp_task2",semp_task2_entry,NULL,0x400,NULL,11);
205 // osal_task_create("led_task",led_task,NULL,0x400,NULL,2);
206
207     return 0;
208 }
```

步骤 4 添加中断创建代码

```
GPIO_InitTypeDef GPIO_InitStruct;
GPIO_InitStruct.Pin = GPIO_PIN_13;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
osal_int_connect(EXTI2_IRQn, 3, NULL, Key1_interrupt_entry, NULL);
osal_int_connect(EXTI3_IRQn, 4, NULL, Key2_interrupt_entry, NULL);
```

```
188     int standard_app_demo_main()
189     {
190         LCD_Clear(BLACK);
191         POINT_COLOR = GREEN;
192         LCD_ShowString(10, 10, 200, 16, 24, "Welcome to LiteOS");
193
194         // task_id = osal_task_create("helloworld",app_hello_world_entry,NULL,0x400,NULL,3);
195         // osal_task_create("task2",task2,NULL,0x400,NULL,2);
196
197         // osal_mutex_create(&public_value_mutex);
198         // osal_task_create("mutex_task1", mutex_task1_entry, NULL, 0x400, NULL, 12);
199         // osal_task_create("mutex_task2", mutex_task2_entry, NULL, 0x400, NULL, 11);
200         // osal_task_create("mem_access_task",mem_access_task_entry,NULL,0x400,NULL,11);
201         // osal_semp_create(&sync_semp, 1, 0);
202         // printf("sync_semp semp create success.\r\n");
203         // osal_task_create("semp_task1",semp_task1_entry,NULL,0x400,NULL,12);
204         // osal_task_create("semp_task2",semp_task2_entry,NULL,0x400,NULL,11);
205         // osal_task_create("led_task",led_task,NULL,0x400,NULL,2);
206         GPIO_InitTypeDef GPIO_InitStruct;
207         GPIO_InitStruct.Pin = GPIO_PIN_13;
208         GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
209         GPIO_InitStruct.Pull = GPIO_NOPULL;
210         GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
211         HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
212         osal_int_connect(EXTI2_IRQn, 3, NULL, Key1_interrupt_entry, NULL);
213         osal_int_connect(EXTI3_IRQn, 4, NULL, Key2_interrupt_entry, NULL);
214
215     }
216 }
```

步骤 5 编译烧录查看结果

编译烧录程序，按下开发板右下角 F1 按键 LED 灯开，按下 F2 按键 LED 灯关；

4.3 练习题

4.3.1 创建多行 LCD 屏幕显示

4.3.2 创建多种颜色的 LCD 屏幕显示

4.3.3 在控制台打印 LED 灯的开关状态

4.3.4 在 LCD 屏幕显示 LED 灯的开关状态

5

基于 NB-IoT 和 WIFI 的智慧农业实验

5.1 实验介绍

5.1.1 关于本实验

本实验基于 NB-IoT 和 WIFI 实现智慧农业案例，实现实时数据的采集，实现命令下发的响应，实现端云互通。

5.1.2 实验目的

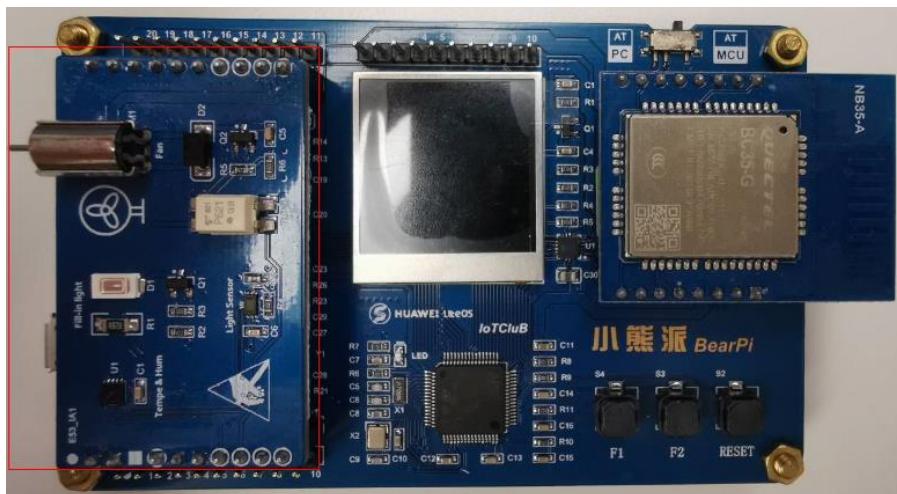
- 掌握 NB-IoT 通信方式的配置
- 掌握 WIFI 通信方式的配置
- 掌握智慧农业案例的开发过程

5.2 实验任务配置步骤

5.2.1 配置智慧农业案例

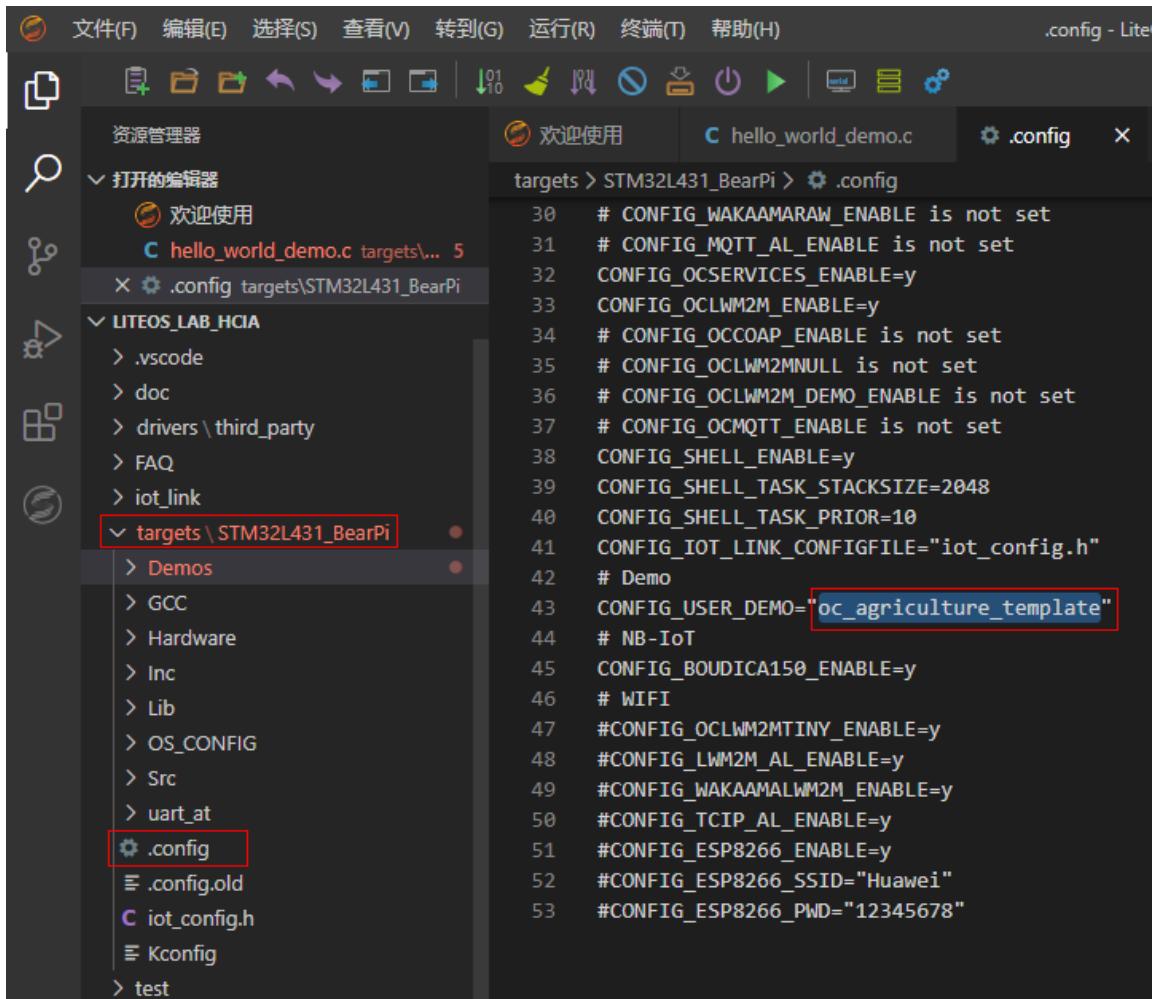
步骤 1 安装智慧农业扩展板 E53_IA1

将智慧农业扩展板 E53_IA1 按照正确的方向插入小熊派开发板；



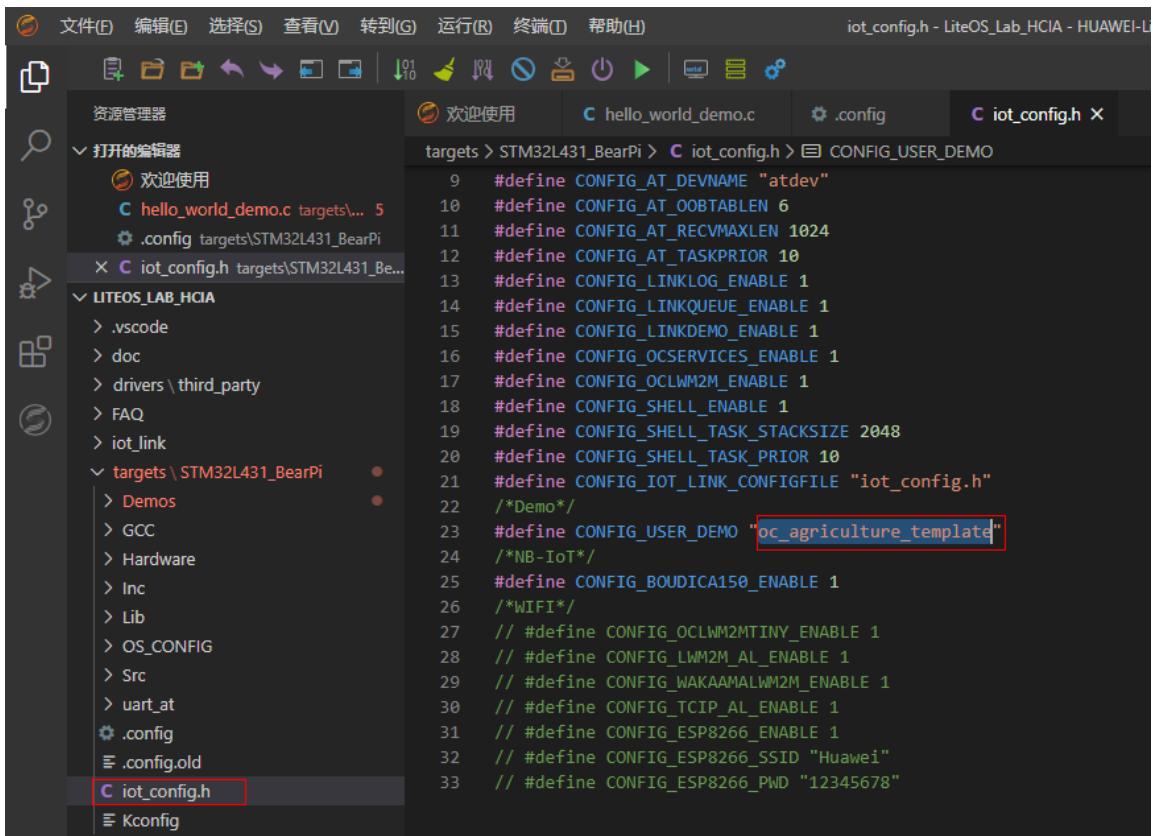
步骤 2 修改.config 文件

打开 targets->STM32L431_BearPi->.config 文件；
修改“CONFIG_USER_DEMO”为“oc_agriculture_template”；
“Ctrl+S”保存.config 文件；



步骤 3 修改 iot_config.h 文件

打开 targets->STM32L431_BearPi->iot_config.h 文件；
修改“CONFIG_USER_DEMO”为“oc_agriculture_template”；
“Ctrl+S”保存 iot_config.h 文件；



```
#define CONFIG_AT_DEVNAME "atdev"
#define CONFIG_AT_OOBTABLEN 6
#define CONFIG_AT_RECVMAXLEN 1024
#define CONFIG_AT_TASKPRIOR 10
#define CONFIG_LINKLOG_ENABLE 1
#define CONFIG_LINKQUEUE_ENABLE 1
#define CONFIG_LINKDEMO_ENABLE 1
#define CONFIG_OCSERVICES_ENABLE 1
#define CONFIG_OCLWM2M_ENABLE 1
#define CONFIG_SHELL_ENABLE 1
#define CONFIG_SHELL_TASK_STACKSIZE 2048
#define CONFIG_SHELL_TASK_PRIOR 10
#define CONFIG_IOT_LINK_CONFIGFILE "iot_config.h"
/*Demo*/
#define CONFIG_USER_DEMO "oc_agriculture_template"
/*NB-IoT*/
#define CONFIG_BOUDICA150_ENABLE 1
/*WIFI*/
// #define CONFIG_OCLWM2MTINY_ENABLE 1
// #define CONFIG_LWM2M_AL_ENABLE 1
// #define CONFIG_WAKAAMALWM2M_ENABLE 1
// #define CONFIG_TCIP_AL_ENABLE 1
// #define CONFIG_ESP8266_ENABLE 1
// #define CONFIG_ESP8266_SSID "Huawei"
// #define CONFIG_ESP8266_PWD "12345678"
```

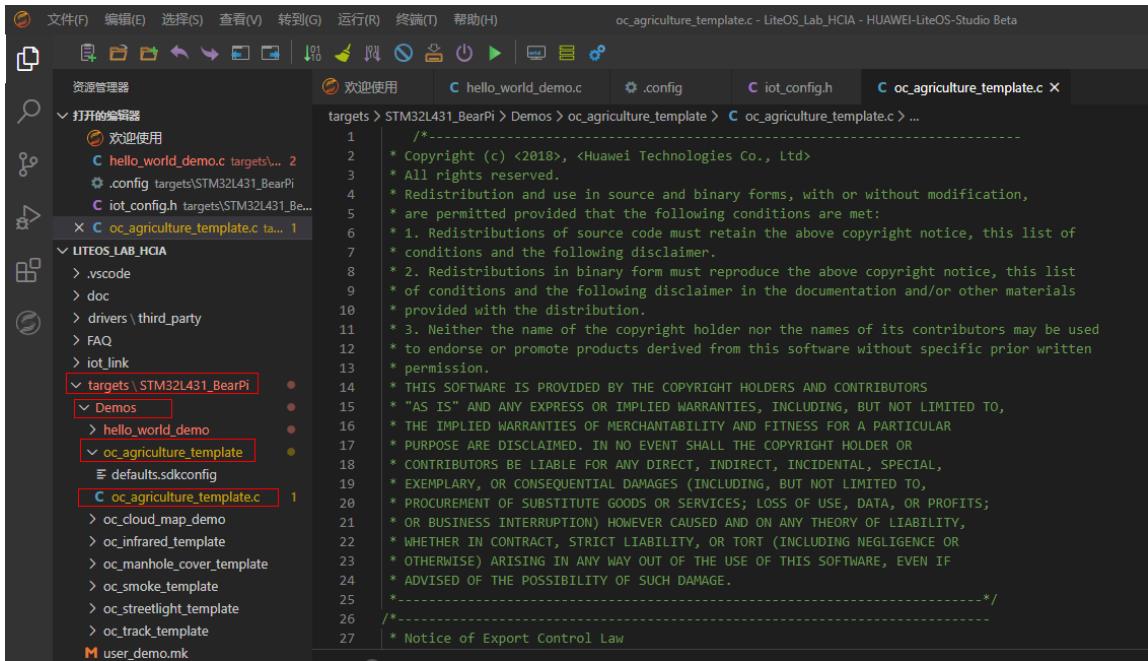
5.2.2 创建智慧农业所需的数据结构

步骤 1 打开 oc_agriculture_template.c 文件

打开案例程序文件 targets->STM32L431_BearPi->Demos->

oc_agriculture_template->oc_agriculture_template.c;

智慧农业案例的代码都在 oc_agriculture_template.c 中修改；



The screenshot shows the LiteOS Studio Beta interface with the file `oc_agriculture_template.c` open. The code contains a standard copyright notice and redistribution terms, typical for open-source software.

步骤 2 添加智慧农业案例扩展板驱动头文件

```
#include "E53_IA1.h"
```

```
39  #include <stdint.h>
40  #include <stddef.h>
41  #include <string.h>
42  #include <osal.h>
43  #include <oc_lwm2m_al.h>
44  #include <link_endian.h>
45  #include <gpio.h>
46  #include <stm32l4xx_it.h>
47  #include "lcd.h"
48  //驱动头文件
49  #include "E53_IA1.h"
```

步骤 3 添加平台所需的基本变量

```
#define cn_endpoint_id          "BearPi_0001"  
#define cn_app_server           "119.3.250.80"  
#define cn_app_port             "5683"  
  
50  typedef unsigned char int8u;  
51  typedef char int8s;  
52  typedef unsigned short int16u;  
53  typedef short int16s;  
54  typedef unsigned char int24u;  
55  typedef char int24s;  
56  typedef int int32s;  
57  typedef char string;  
58  typedef char array;  
59  typedef char varstring;  
60  typedef char variant;  
61  
62  //连接平台基本变量  
63  #define cn_endpoint_id          "BearPi_0001"  
64  #define cn_app_server           "119.3.250.80"  
65  #define cn_app_port             "5683"
```

步骤 4 添加 MessageID

```
#define cn_app_Agriculture 0x0  
#define cn_app_Agriculture_Control_Light 0x1  
#define cn_app_response_Agriculture_Control_Light 0x2  
#define cn_app_Agriculture_Control_Motor 0x3  
#define cn_app_response_Agriculture_Control_Motor 0x4
```

```
62  //连接平台基本变量  
63  #define cn_endpoint_id          "BearPi_0001"  
64  #define cn_app_server           "119.3.250.80"  
65  #define cn_app_port             "5683"  
66  //MessageID  
67  #define cn_app_Agriculture 0x0  
68  #define cn_app_Agriculture_Control_Light 0x1  
69  #define cn_app_response_Agriculture_Control_Light 0x2  
70  #define cn_app_Agriculture_Control_Motor 0x3  
71  #define cn_app_response_Agriculture_Control_Motor 0x4  
72  //消息结构体
```

步骤 5 添加消息结构体

```
#pragma pack(1)
typedef struct
{
    int8u messageId;
    int8u Temperature;
    int8u Humidity;
    int16u Luminance;
} tag_app_Agriculture;
typedef struct
{
    int8u messageId;
    int16u mid;
    int8u errcode;
    int8u Light_State;
} tag_app_Response_Agriculture_Control_Light;
typedef struct
{
    int8u messageId;
    int16u mid;
    int8u errcode;
    int8u Motor_State;
} tag_app_Response_Agriculture_Control_Motor;
typedef struct
{
    int8u messageId;
    int16u mid;
    string Light[3];
} tag_app_Agriculture_Control_Light;
typedef struct
{
    int8u messageId;
    int16u mid;
    string Motor[3];
} tag_app_Agriculture_Control_Motor;
#pragma pack()
```

```
72 //消息结构体
73 #pragma pack(1)
74 typedef struct
75 {
76     int8u messageId;
77     int8u Temperature;
78     int8u Humidity;
79     int16u Luminance;
80 } tag_app_Agriculture;
81 typedef struct
82 {
83     int8u messageId;
84     int16u mid;
85     int8u errcode;
86     int8u Light_State;
87 } tag_app_Response_Agriculture_Control_Light;
88 typedef struct
89 {
90     int8u messageId;
91     int16u mid;
92     int8u errcode;
93     int8u Motor_State;
94 } tag_app_Response_Agriculture_Control_Motor;
95 typedef struct
96 {
97     int8u messageId;
98     int16u mid;
99     string Light[3];
100 } tag_app_Agriculture_Control_Light;
101 typedef struct
102 {
103     int8u messageId;
104     int16u mid;
105     string Motor[3];
106 } tag_app_Agriculture_Control_Motor;
107 #pragma pack()
```

步骤 6 添加存储数据的变量

```
E53_IA1_Data_TypeDef E53_IA1_Data;
```

```
108 //存储数据的变量
109 E53_IA1_Data_TypeDef E53_IA1_Data;
110
```

5.2.3 创建数据收集任务

步骤 1 添加数据收集任务

```
static int app_collect_task_entry()
{
    Init_E53_IA1();
    while (1)
    {
        E53_IA1_Read_Data();
        printf("\r\n*****Lux Value is %d\r\n",
(int)E53_IA1_Data.Lux);
        printf("\r\n*****Humidity is %d\r\n",
(int)E53_IA1_Data.Humidity);
        printf("\r\n*****Temperature is %d\r\n",
(int)E53_IA1_Data.Temperature);

        LCD_ShowString(10, 140, 200, 16, 16, "Temperature:");
        LCD_ShowNum(140, 140, (int)E53_IA1_Data.Temperature, 5, 16);
        LCD_ShowString(10, 170, 200, 16, 16, "Humidity:");
        LCD_ShowNum(140, 170, (int)E53_IA1_Data.Humidity, 5, 16);
        LCD_ShowString(10, 200, 200, 16, 16, "Luminance:");
        LCD_ShowNum(140, 200, (int)E53_IA1_Data.Lux, 5, 16);

        osal_task_sleep(2*1000);
    }

    return 0;
}
```

```
147 //数据收集任务
148 static int app_collect_task_entry()
149 {
150     Init_E53_IA1();
151     while (1)
152     {
153         E53_IA1_Read_Data();
154         printf("\r\n*****Lux Value is %d\r\n", (int)E53_IA1_Data.Lux);
155         printf("\r\n*****Humidity is %d\r\n", (int)E53_IA1_Data.Humidity);
156         printf("\r\n*****Temperature is %d\r\n", (int)E53_IA1_Data.Temperature);
157
158         LCD_ShowString(10, 140, 200, 16, 16, "Temperature:");
159         LCD_ShowNum(140, 140, (int)E53_IA1_Data.Temperature, 5, 16);
160         LCD_ShowString(10, 170, 200, 16, 16, "Humidity:");
161         LCD_ShowNum(140, 170, (int)E53_IA1_Data.Humidity, 5, 16);
162         LCD_ShowString(10, 200, 200, 16, 16, "Luminance:");
163         LCD_ShowNum(140, 200, (int)E53_IA1_Data.Lux, 5, 16);
164
165         osal_task_sleep(2*1000);
166     }
167
168     return 0;
169 }
170 int standard_app_demo_main()
171 {
```

步骤 2 创建数据收集任务

```
osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
```

```
170 int standard_app_demo_main()
171 {
172     osal_semp_create(&s_rcv_sync,1,0);
173     //LCD屏幕显示
174
175     //创建任务
176     osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
177     return 0;
178 }
```

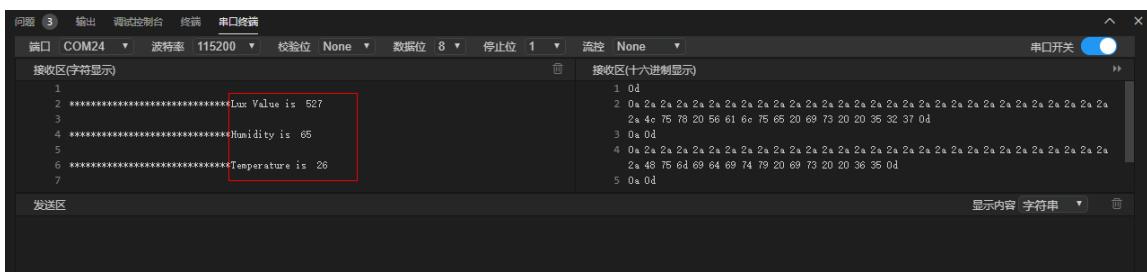
步骤 3 添加 LCD 屏幕显示代码

```
LCD_Clear(BLACK);
POINT_COLOR = GREEN;
LCD_ShowString(10, 10, 200, 16, 24, "Welcome to BearPi");
LCD_ShowString(15, 40, 200, 16, 24, "Agriculture Demo");
LCD_ShowString(10, 80, 200, 16, 16, "NCDP_IP:");
LCD_ShowString(80, 80, 200, 16, 16, cn_app_server);
LCD_ShowString(10, 110, 200, 16, 16, "NCDP_PORT:");
LCD_ShowString(100, 110, 200, 16, 16, cn_app_port);
```

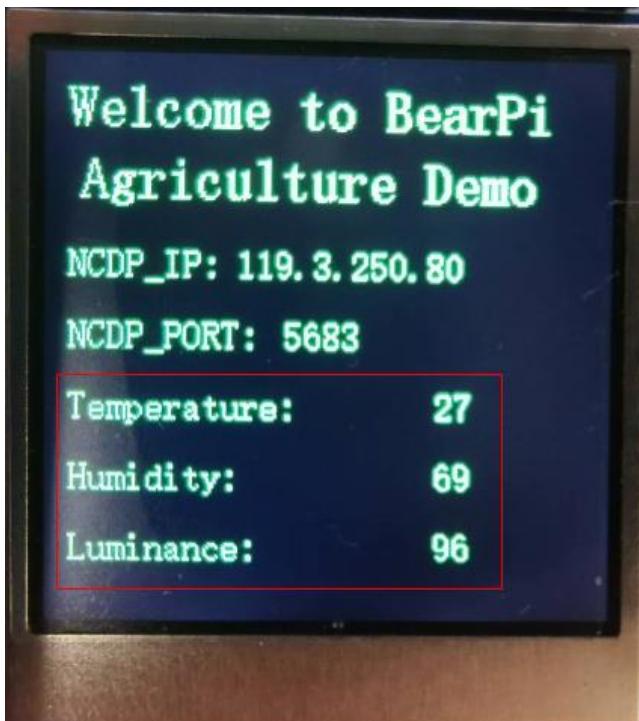
```
170     int standard_app_demo_main()
171     {
172         osal_semp_create(&s_rcv_sync,1,0);
173         //LCD屏幕显示
174         LCD_Clear(BLACK);
175         POINT_COLOR = GREEN;
176         LCD_ShowString(10, 10, 200, 16, 24, "Welcome to BearPi");
177         LCD_ShowString(15, 40, 200, 16, 24, "Agriculture Demo");
178         LCD_ShowString(10, 80, 200, 16, 16, "NCDP_IP:");
179         LCD_ShowString(80, 80, 200, 16, 16, cn_app_server);
180         LCD_ShowString(10, 110, 200, 16, 16, "NCDP_PORT:");
181         LCD_ShowString(100, 110, 200, 16, 16, cn_app_port);
182         //创建任务
183         osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
184         return 0;
185     }
```

步骤 4 编译烧录，查看结果

编译烧录程序，在串口终端中可以看到有实时的数据打印；



在 LCD 屏幕上，也可以看到实时的数据显示；



5.2.4 创建数据上报任务

步骤 1 添加数据上报任务

```
static int app_report_task_entry()
{
    int ret = -1;

    oc_config_param_t      oc_param;
    tag_app_Agriculture Agriculture;

    (void) memset(&oc_param,0,sizeof(oc_param));

    oc_param.app_server.address = cn_app_server;
    oc_param.app_server.port = cn_app_port;
    oc_param.app_server.ep_id = cn_endpoint_id;
    oc_param.boot_mode = en_oc_boot_strap_mode_factory;
    oc_param.rcv_func = app_msg_deal;

    ret = oc_lwm2m_config( &oc_param);
    if (0 != ret)
    {
        return ret;
    }

    //install a dealer for the led message received
    while(1) //--TODO ,you could add your own code here
    {
        Agriculture.messageId = cn_app_Agriculture;
        Agriculture.Temperature = (int8_t)E53_IA1_Data.Temperature;
        Agriculture.Humidity = (int8_t)E53_IA1_Data.Humidity;
        Agriculture.Luminance = htons((uint16_t)E53_IA1_Data.Lux);
        oc_lwm2m_report( (char *)&Agriculture, sizeof(Agriculture), 1000);
        osal_task_sleep(2*1000);
    }
    return ret;
}
```

```
targets > STM32L431_BearPi > Demos > oc_agriculture_template > C oc_agriculture_template.c > app_repo
145 //数据上报任务
146 static int app_report_task_entry()
147 {
148     int ret = -1;
149
150     oc_config_param_t      oc_param;
151     tag_app_Agriculture Agriculture;
152
153     (void) memset(&oc_param,0,sizeof(oc_param));
154
155     oc_param.app_server.address = cn_app_server;
156     oc_param.app_server.port = cn_app_port;
157     oc_param.app_server.ep_id = cn_endpoint_id;
158     oc_param.boot_mode = en_oc_boot_strap_mode_factory;
159     oc_param.recv_func = app_msg_deal;
160
161     ret = oc_lwm2m_config( &oc_param);
162     if (0 != ret)
163     {
164         return ret;
165     }
166
167     //install a dealer for the led message received
168     while(1) //--TODO ,you could add your own code here
169     {
170         Agriculture.messageId = cn_app_Agriculture;
171         Agriculture.Temperature = (int8_t)E53_IA1_Data.Temperature;
172         Agriculture.Humidity = (int8_t)E53_IA1_Data.Humidity;
173         Agriculture.Luminance = htons((uint16_t)E53_IA1_Data.Lux);
174         oc_lwm2m_report( (char *)&Agriculture, sizeof(Agriculture), 1000);
175         osal_task_sleep(2*1000);
176     }
177     return ret;
178 }
179 //数据收集任务
```

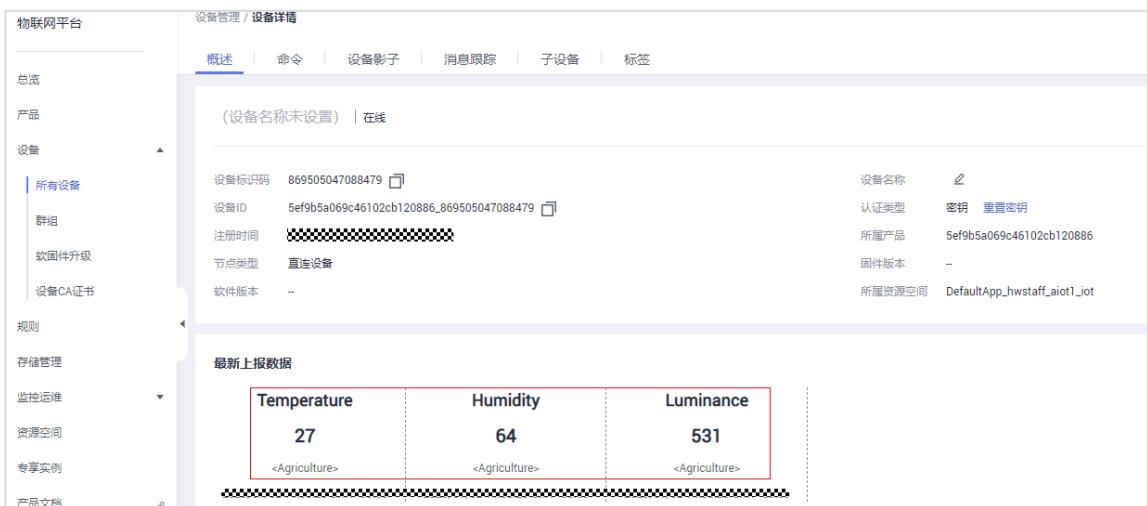
步骤 2 创建数据上报任务

```
osal_task_create("app_report",app_report_task_entry,NULL,0x1000,NULL,2);

● 202 int standard_app_demo_main()
203 {
204     osal_semp_create(&s_rcv_sync,1,0);
205     //LCD屏幕显示
206     LCD_Clear(BLACK);
207     POINT_COLOR = GREEN;
208     LCD_ShowString(10, 10, 200, 16, 24, "Welcome to BearPi");
209     LCD_ShowString(15, 40, 200, 16, 24, "Agriculture Demo");
210     LCD_ShowString(10, 80, 200, 16, 16, "NCDP_IP:");
211     LCD_ShowString(80, 80, 200, 16, 16, cn_app_server);
212     LCD_ShowString(10, 110, 200, 16, 16, "NCDP_PORT:");
213     LCD_ShowString(100, 110, 200, 16, 16, cn_app_port);
214     //创建任务
215     osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
216     osal_task_create("app_report",app_report_task_entry,NULL,0x1000,NULL,2);
217     return 0;
218 }
```

步骤 3 编译烧录，查看结果

编译烧录程序，在物联网平台中，可以看到实时的数据；



The screenshot shows the Huawei IoT Platform's Device Management interface. On the left sidebar, under the '设备' (Device) section, '所有设备' (All Devices) is selected. In the main content area, the '概述' (Overview) tab is active. It displays device details such as device ID, registration time, and node type. Below this, the '最新上报数据' (Latest Reported Data) section shows three data points: Temperature (27), Humidity (64), and Luminance (531), all associated with the 'Agriculture' category.

5.2.5 创建命令响应任务

步骤 1 添加命令处理任务

```
static int app_cmd_task_entry()
{
    int ret = -1;
    tag_app_Response_Agriculture_Control_Light Response_Agriculture_Control_Light;
    tag_app_Response_Agriculture_Control_Motor
Response_Agriculture_Control_Motor;
    tag_app_Agriculture_Control_Light *Agriculture_Control_Light;
    tag_app_Agriculture_Control_Motor *Agriculture_Control_Motor;
    int8_t msgid;

    while(1)
    {
        if(osal_semp_pend(s_rcv_sync,cn_osal_timeout_forever))
        {
            msgid = s_rcv_buffer[0] & 0x000000FF;
            switch (msgid)
            {
                case cn_app_Agriculture_Control_Light:
                    Agriculture_Control_Light = (tag_app_Agriculture_Control_Light
*)s_rcv_buffer;
                    printf("Agriculture_Control_Light:msgid:%d mid:%d",
Agriculture_Control_Light->messagId, ntohs(Agriculture_Control_Light->mid));
                    /***** code area for cmd from IoT cloud *****/
                    if (Agriculture_Control_Light->Light[0] == 'O' &&
Agriculture_Control_Light->Light[1] == 'N')
                    {
                        HAL_GPIO_WritePin(IA1_Light_GPIO_Port, IA1_Light_Pin,
GPIO_PIN_SET);
                        Response_Agriculture_Control_Light.messagId =
cn_app_response_Agriculture_Control_Light;
                        Response_Agriculture_Control_Light.mid =
Agriculture_Control_Light->mid;
                        Response_Agriculture_Control_Light.errcode = 0;
                        Response_Agriculture_Control_Light.Light_State = 1;
                    }
            }
        }
    }
}
```

```
        oc_lwm2m_report((char
*)&Response_Agriculture_Control_Light,sizeof(Response_Agriculture_Control_Light),100
0);    ///< report cmd reply message
    }
    if (Agriculture_Control_Light->Light[0] == 'O' &&
Agriculture_Control_Light->Light[1] == 'F' && Agriculture_Control_Light->Light[2] ==
'F')
    {
        HAL_GPIO_WritePin(IA1_Light_GPIO_Port, IA1_Light_Pin,
GPIO_PIN_RESET);
        Response_Agriculture_Control_Light.messageId =
cn_app_response_Agriculture_Control_Light;
        Response_Agriculture_Control_Light.mid =
Agriculture_Control_Light->mid;
        Response_Agriculture_Control_Light.errcode = 0;
        Response_Agriculture_Control_Light.Light_State = 0;
        oc_lwm2m_report((char
*)&Response_Agriculture_Control_Light,sizeof(Response_Agriculture_Control_Light),100
0);    ///< report cmd reply message
    }
    /***** code area end *****/
    break;
case cn_app_Agriculture_Control_Motor:
    Agriculture_Control_Motor =
(tag_app_Agriculture_Control_Motor *)s_rcv_buffer;
    printf("Agriculture_Control_Motor:msgid:%d mid:%d",
Agriculture_Control_Motor->messageId, ntohs(Agriculture_Control_Motor->mid));
    /***** code area for cmd from IoT cloud *****/
    if (Agriculture_Control_Motor->Motor[0] == 'O' &&
Agriculture_Control_Motor->Motor[1] == 'N')
    {
        HAL_GPIO_WritePin(IA1_Motor_GPIO_Port, IA1_Motor_Pin,
GPIO_PIN_SET);
        Response_Agriculture_Control_Motor.messageId =
cn_app_response_Agriculture_Control_Motor;
        Response_Agriculture_Control_Motor.mid =
Agriculture_Control_Motor->mid;
        Response_Agriculture_Control_Motor.errcode = 0;
        Response_Agriculture_Control_Motor.Motor_State = 1;
```

```
        oc_lwm2m_report((char
*)&Response_Agriculture_Control_Motor,sizeof(Response_Agriculture_Control_Motor),1
000);    ///< report cmd reply message
    }
    if (Agriculture_Control_Motor->Motor[0] == 'O' &&
Agriculture_Control_Motor->Motor[1] == 'F' && Agriculture_Control_Motor->Motor[2]
== 'F')
{
    HAL_GPIO_WritePin(IA1_Motor_GPIO_Port, IA1_Motor_Pin,
GPIO_PIN_RESET);
    Response_Agriculture_Control_Motor.messageId =
cn_app_response_Agriculture_Control_Motor;
    Response_Agriculture_Control_Motor.mid =
Agriculture_Control_Motor->mid;
    Response_Agriculture_Control_Motor.errcode = 0;
    Response_Agriculture_Control_Motor.Motor_State = 0;
    oc_lwm2m_report((char
*)&Response_Agriculture_Control_Motor,sizeof(Response_Agriculture_Control_Motor),1
000);    ///< report cmd reply message
}
***** code area end *****/
break;
default:
    break;
}
}

return ret;
}
```

```
143 //下发命令处理任务
144 static int app_cmd_task_entry()
145 {
146     int ret = -1;
147     tag_app_Response_Agriculture_Control_Light Response_Agriculture_Control_Light;
148     tag_app_Response_Agriculture_Control_Motor Response_Agriculture_Control_Motor;
149     tag_app_Agriculture_Control_Light *Agriculture_Control_Light;
150     tag_app_Agriculture_Control_Motor *Agriculture_Control_Motor;
151     int8_t msgid;
152
153     while(1)
154     {
155         if(osal_semp_pend(s_rcv_sync,cn_osal_timeout_forever))
156         {
157             msgid = s_rcv_buffer[0] & 0x000000FF;
158             switch (msgid)
159             {
160                 case cn_app_Agriculture_Control_Light:
161                     Agriculture_Control_Light = (tag_app_Agriculture_Control_Light *)s_rcv_buffer;
162                     printf("Agriculture_Control_Light:msgid:%d mid:%d", Agriculture_Control_Light->messageId, ntohs(Agriculture_Control_Light-
163                     /***** code area for cmd from IoT cloud *****/
164                     if (Agriculture_Control_Light->Light[0] == '0' && Agriculture_Control_Light->Light[1] == 'N')
165                     {
166                         HAL_GPIO_WritePin(IA1_Light_GPIO_Port, IA1_Light_Pin, GPIO_PIN_SET);
167                         Response_Agriculture_Control_Light.messageId = cn_app_response_Agriculture_Control_Light;
168                         Response_Agriculture_Control_Light.mid = Agriculture_Control_Light->mid;
169                         Response_Agriculture_Control_Light.errcode = 0;
170                         Response_Agriculture_Control_Light.Light_State = 1;
171                         oc_lwm2m_report((char *)&Response_Agriculture_Control_Light,sizeof(Response_Agriculture_Control_Light),1000);    ///<
172
173                     if (Agriculture_Control_Light->Light[0] == '0' && Agriculture_Control_Light->Light[1] == 'F' && Agriculture_Control_Light-
174                     {
175                         HAL_GPIO_WritePin(IA1_Light_GPIO_Port, IA1_Light_Pin, GPIO_PIN_RESET);
176                         Response_Agriculture_Control_Light.messageId = cn_app_response_Agriculture_Control_Light;
177                         Response_Agriculture_Control_Light.mid = Agriculture_Control_Light->mid;
178                         Response_Agriculture_Control_Light.errcode = 0;
179                         Response_Agriculture_Control_Light.Light_State = 0;
180                         oc_lwm2m_report((char *)&Response_Agriculture_Control_Light,sizeof(Response_Agriculture_Control_Light),1000);    ///<
181
182                     }***** code area end *****/
183                     break;
184                 case cn_app_Agriculture_Control_Motor:
185                     Agriculture_Control_Motor = (tag_app_Agriculture_Control_Motor *)s_rcv_buffer;
186                     printf("Agriculture_Control_Motor:msgid:%d mid:%d", Agriculture_Control_Motor->messageId, ntohs(Agriculture_Control_Motor-
187                     /***** code area for cmd from IoT cloud *****/
188                     if (Agriculture_Control_Motor->Motor[0] == '0' && Agriculture_Control_Motor->Motor[1] == 'N')
189                     {
190                         HAL_GPIO_WritePin(IA1_Motor_GPIO_Port, IA1_Motor_Pin, GPIO_PIN_SET);
191                         Response_Agriculture_Control_Motor.messageId = cn_app_response_Agriculture_Control_Motor;
192                         Response_Agriculture_Control_Motor.mid = Agriculture_Control_Motor->mid;
193                         Response_Agriculture_Control_Motor.errcode = 0;
194                         Response_Agriculture_Control_Motor.Motor_State = 1;
195                         oc_lwm2m_report((char *)&Response_Agriculture_Control_Motor,sizeof(Response_Agriculture_Control_Motor),1000);    ///<
196
197                     if (Agriculture_Control_Motor->Motor[0] == '0' && Agriculture_Control_Motor->Motor[1] == 'F' && Agriculture_Control_Motor-
198                     {
199                         HAL_GPIO_WritePin(IA1_Motor_GPIO_Port, IA1_Motor_Pin, GPIO_PIN_RESET);
200                         Response_Agriculture_Control_Motor.messageId = cn_app_response_Agriculture_Control_Motor;
201                         Response_Agriculture_Control_Motor.mid = Agriculture_Control_Motor->mid;
202                         Response_Agriculture_Control_Motor.errcode = 0;
203                         Response_Agriculture_Control_Motor.Motor_State = 0;
204                         oc_lwm2m_report((char *)&Response_Agriculture_Control_Motor,sizeof(Response_Agriculture_Control_Motor),1000);    ///<
205
206                     }***** code area end *****/
207                     break;
208                 default:
209                     break;
210             }
211         }
212     }
213
214     return ret;
215 }
216 //数据上报任务
```

步骤 2 创建命令处理任务

```
osal_task_create("app_command",app_cmd_task_entry,NULL,0x1000,NULL,3);

273 int standard_app_demo_main()
274 {
275     osal_semp_create(&s_rcv_sync,1,0);
276     //LCD屏幕显示
277     LCD_Clear(BLACK);
278     POINT_COLOR = GREEN;
279     LCD_ShowString(10, 10, 200, 16, 24, "Welcome to BearPi");
280     LCD_ShowString(15, 40, 200, 16, 24, "Agriculture Demo");
281     LCD_ShowString(10, 80, 200, 16, 16, "NCDP_IP:");
282     LCD_ShowString(80, 80, 200, 16, 16, cn_app_server);
283     LCD_ShowString(10, 110, 200, 16, 16, "NCDP_PORT:");
284     LCD_ShowString(100, 110, 200, 16, 16, cn_app_port);
285     //创建任务
286     osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
287     osal_task_create("app_report",app_report_task_entry,NULL,0x1000,NULL,2);
288     osal_task_create("app_command",app_cmd_task_entry,NULL,0x1000,NULL,3);
289
290 }
```

步骤 3 编译烧录，查看结果

编译烧录程序，在物联网平台中下发开灯命令，查看命令下发结果及开发板的响应；

应用模拟器

全部 数据接收 命令发送

发送消息body信息: { "serviceld": "Agriculture", "method": "Agriculture_Control_Light", "paras": " {"Light"\:"ON\}" }

命令状态:执行成功

命令响应时间:2020/07/02 15:11:45 GMT+08:00

命令响应内容:{ "Light_State": 1 }

服务: Agriculture

命令: Agriculture_Control_Light

Light: ON

缓存发送 立即发送

设置时间 ▾

IoT Platform

命令下发 →
数据上报 ←

真实物理设备

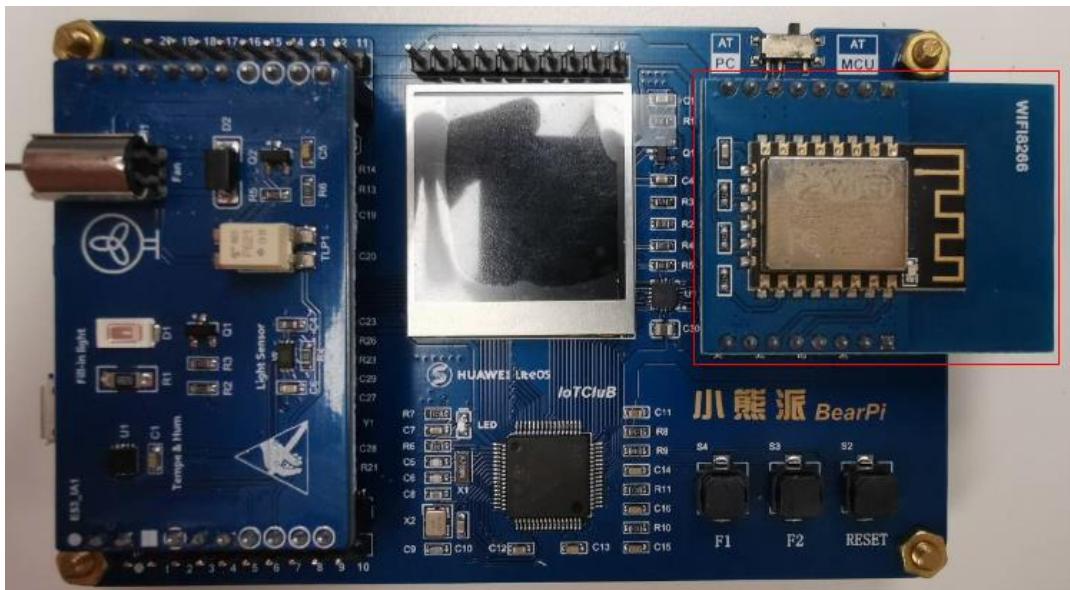
数据上报 ↑
命令下发 ↓



5.2.6 使用 WIFI 通信方式

步骤 1 安装 WIFI8266 通信扩展板

将 NB35-A 通信扩展板替换为 WIFI8266 通信扩展板，天线朝外；

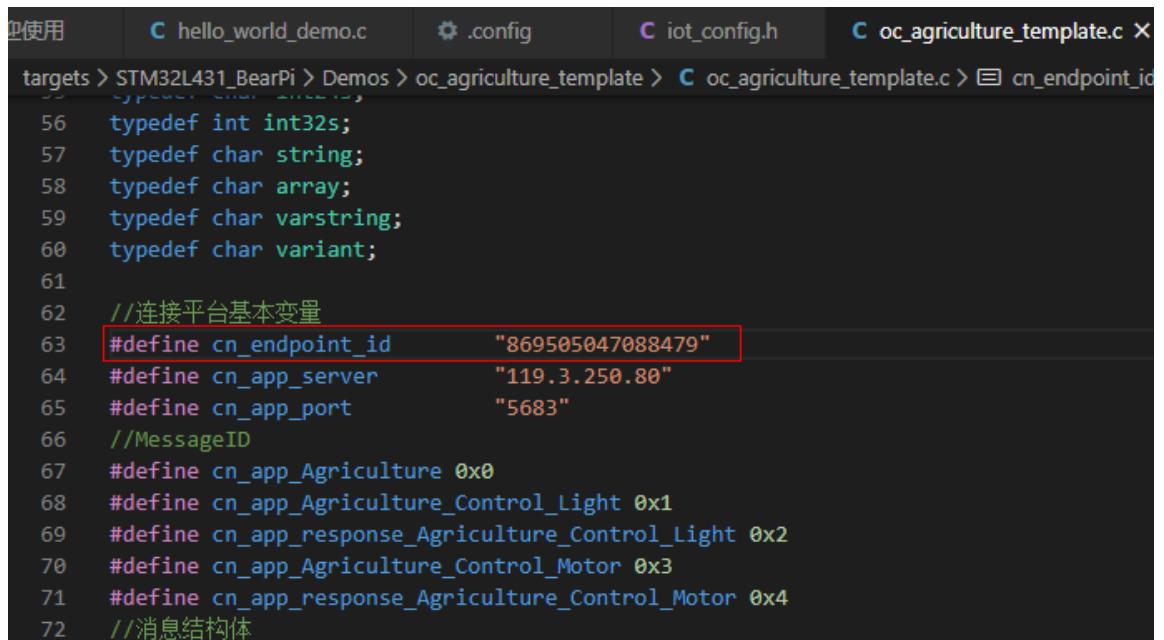


步骤 2 修改设备标识码

在 `oc_agriculture_template.c` 中，

修改“`cn_endpoint_id`”为前面实验使用 AT 指令查询到的 IMEI 号；

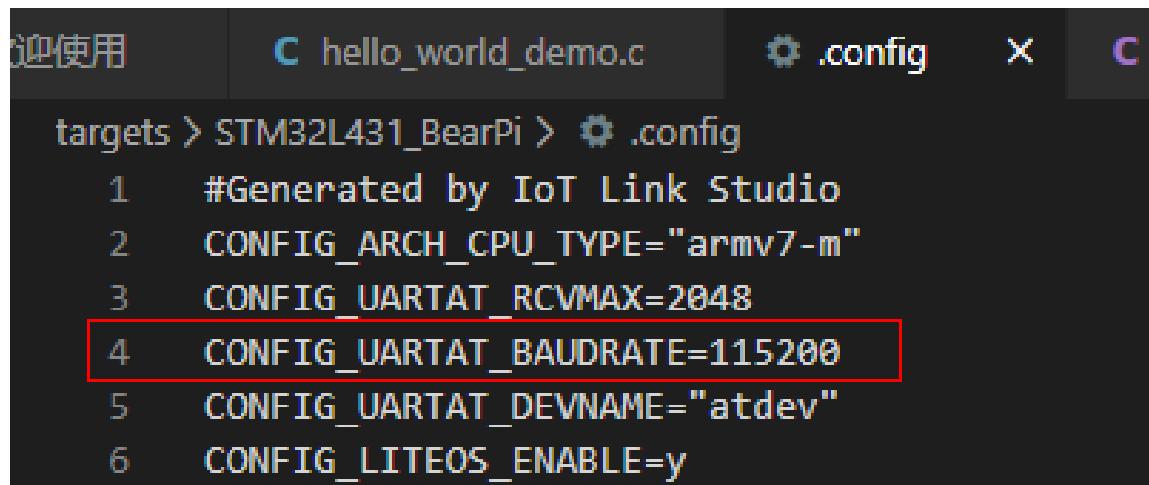
```
#define cn_endpoint_id      "869505047088479"
```



```
targets > STM32L431_BearPi > Demos > oc_agriculture_template > C oc_agriculture_template.c > cn_endpoint_id
56     typedef int int32s;
57     typedef char string;
58     typedef char array;
59     typedef char varstring;
60     typedef char variant;
61
62     //连接平台基本变量
63     #define cn_endpoint_id      "869505047088479"
64     #define cn_app_server        "119.3.250.80"
65     #define cn_app_port          "5683"
66     //MessageID
67     #define cn_app_Agriculture   0x0
68     #define cn_app_Agriculture_Control_Light 0x1
69     #define cn_app_response_Agriculture_Control_Light 0x2
70     #define cn_app_Agriculture_Control_Motor 0x3
71     #define cn_app_response_Agriculture_Control_Motor 0x4
72     //消息结构体
```

步骤 3 修改.config 文件并保存

打开.config 文件，将“CONFIG_UARTAT_BAUDRATE”波特率设置为“115200”；



```
targets > STM32L431_BearPi > .config
1 #Generated by IoT Link Studio
2 CONFIG_ARCH_CPU_TYPE="armv7-m"
3 CONFIG_UARTAT_RCVMAX=2048
4 CONFIG_UARTAT_BAUDRATE=115200
5 CONFIG_UARTAT_DEVNAME="atdev"
6 CONFIG_LITEOS_ENABLE=y
```

将“CONFIG_BOUDICA150_ENABLE=y”前面加上#号，将 LwM2M 和 WIFI 配置前的#号去除；

并将 SSID 和 PWD 修改为自己手机热点或者路由器的用户名密码；

开发板将通过 WIFI8266 扩展板连接到配置的热点中；

```
# NB-IoT
#CONFIG_BOUDICA150_ENABLE=y
# LWM2M
CONFIG_OCLWM2MTINY_ENABLE=y
CONFIG_LWM2M_AL_ENABLE=y
CONFIG_WAKAAMALWM2M_ENABLE=y
# WIFI
CONFIG_TCIP_AL_ENABLE=y
CONFIG_ESP8266_ENABLE=y
CONFIG_ESP8266_SSID="xxxxxx"
CONFIG_ESP8266_PWD="xxxxxxxx"
```

```
LiteOS_Lab_HCIP > targets > STM32L431_BearPi > .config
40 CONFIG_SHELL_TASK_PRIOR=10
41 CONFIG_IOT_LINK_CONFIGFILE="iot_config.h"
42 # Demo
43 CONFIG_USER_DEMO="oc_agriculture_template"
44 # NB-IoT
45 #CONFIG_BOUDICA150_ENABLE=y
46 # LWM2M
47 CONFIG_OCLWM2MTINY_ENABLE=y
48 CONFIG_LWM2M_AL_ENABLE=y
49 CONFIG_WAKAAMALWM2M_ENABLE=y
50 # WIFI
51 CONFIG_TCIP_AL_ENABLE=y
52 CONFIG_ESP8266_ENABLE=y
53 CONFIG_ESP8266_SSID="Huawei"
54 CONFIG_ESP8266_PWD="12345678"
55 # MQTT
56 #CONFIG_CJSON_ENABLE=y
```

步骤 4 修改 iot_config.h 文件并保存

打开 iot_config.h 文件，将“CONFIG_UARTAT_BAUDRATE”波特率设置为“115200”；

```
迎使用 C hello_world_demo.c .config C iot_config.h X
targets > STM32L431_BearPi > C iot_config.h > CONFIG_UARTAT_BAUDRATE
1 /* Generated by IoT Link Studio*/
2 #define CONFIG_ARCH_CPU_TYPE "armv7-m"
3 #define CONFIG_UARTAT_RCVMAX 2048
4 #define CONFIG_UARTAT_BAUDRATE 115200
5 #define CONFIG_UARTAT_DEVNAME "atdev"
```

将“CONFIG_BOUDICA150_ENABLE 1”注释掉；

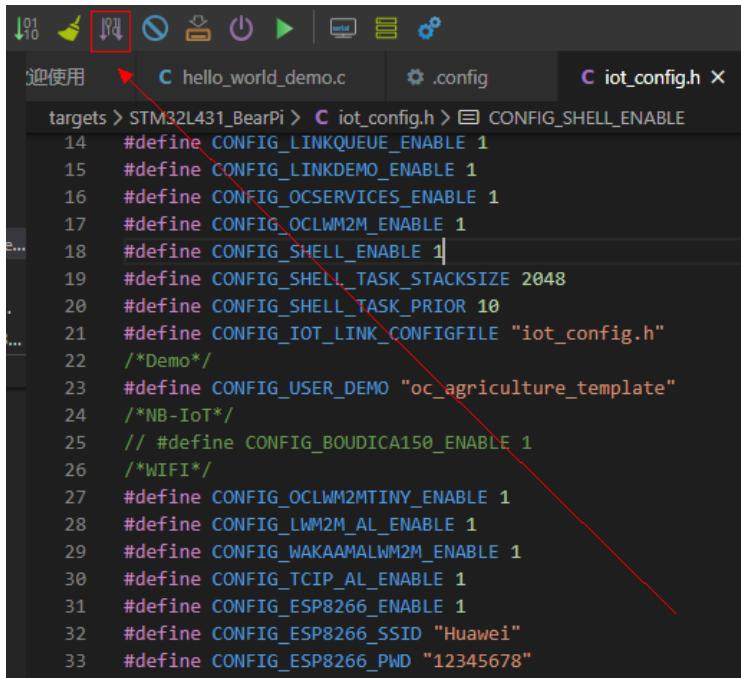
将 WIFI 配置的注释去除，并将 CONFIG_ESP8266_SSID 和 CONFIG_ESP8266_PWD 修改为和.config 中一致的用户名密码；

```
/*NB-IoT*/
// #define CONFIG_BOUDICA150_ENABLE 1
/*LWM2M*/
#define CONFIG_OCLWM2MTINY_ENABLE 1
#define CONFIG_LWM2M_AL_ENABLE 1
#define CONFIG_WAKAAMALWM2M_ENABLE 1
/*WIFI*/
#define CONFIG_TCIP_AL_ENABLE 1
#define CONFIG_ESP8266_ENABLE 1
#define CONFIG_ESP8266_SSID "Huawei"
#define CONFIG_ESP8266_PWD "12345678"
```

```
22  /*Demo*/
23  #define CONFIG_USER_DEMO "oc_agriculture_template"
24  /*NB-IoT*/
25  // #define CONFIG_BOUDICA150_ENABLE 1
26  /*LWM2M*/
27  // #define CONFIG_OCLWM2MTINY_ENABLE 1
28  // #define CONFIG_LWM2M_AL_ENABLE 1
29  // #define CONFIG_WAKAAMALWM2M_ENABLE 1
30  /*WIFI*/
31  #define CONFIG_TCIP_AL_ENABLE 1
32  #define CONFIG_ESP8266_ENABLE 1
33  #define CONFIG_ESP8266_SSID "Huawei"
34  #define CONFIG_ESP8266_PWD "12345678"
35  /*MQTT*/
36  // #define CONFIG_CJSON_ENABLE 1
```

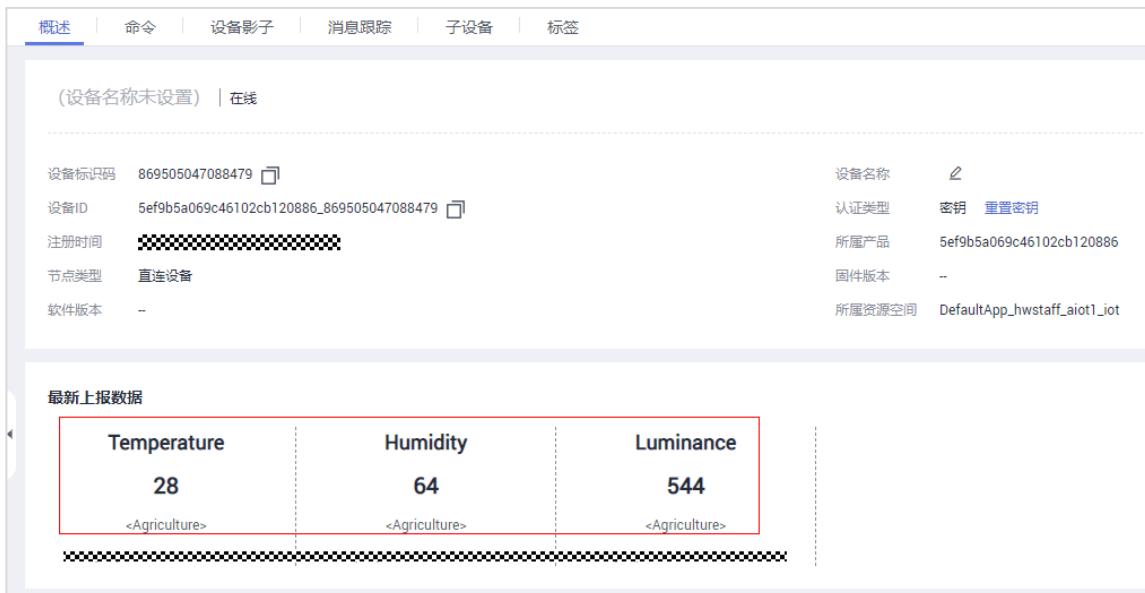
步骤 5 重新编译烧录，查看结果

打开手机热点，点击  进行重新编译；



```
targets > STM32L431_BearPi > iot_config.h > CONFIG_SHELL_ENABLE
14 #define CONFIG_LINKQUEUE_ENABLE 1
15 #define CONFIG_LINKDEMO_ENABLE 1
16 #define CONFIG_OCSERVICES_ENABLE 1
17 #define CONFIG_OCLWM2M_ENABLE 1
18 #define CONFIG_SHELL_ENABLE 1
19 #define CONFIG_SHELL_TASK_STACKSIZE 2048
20 #define CONFIG_SHELL_TASK_PRIOR 10
21 #define CONFIG_IOT_LINK_CONFIGFILE "iot_config.h"
22 /*Demo*/
23 #define CONFIG_USER_DEMO "oc_agriculture_template"
24 /*NB-IoT*/
25 // #define CONFIG_BOUDICA150_ENABLE 1
26 /*WIFI*/
27 #define CONFIG_OCLWM2MTINY_ENABLE 1
28 #define CONFIG_LWM2M_AL_ENABLE 1
29 #define CONFIG_WAKAAMALWM2M_ENABLE 1
30 #define CONFIG_TCIP_AL_ENABLE 1
31 #define CONFIG_ESP8266_ENABLE 1
32 #define CONFIG_ESP8266_SSID "Huawei"
33 #define CONFIG_ESP8266_PWD "12345678"
```

烧录程序，在物联网平台中查看数据上报和命令下发功能；



| Temperature | Humidity | Luminance |
|---------------|---------------|---------------|
| 28 | 64 | 544 |
| <Agriculture> | <Agriculture> | <Agriculture> |



The screenshot shows the Application Simulator interface with the following details:

- 发送消息body信息:** { "serviceId": "Agriculture", "method": "Agriculture_Control_Motor", "paras": "{"Motor": "OFF"}" }
- 命令状态:** 执行成功
- 命令响应时间:** 2020/07/02 15:12:49 GMT+08:00
- 命令响应内容:** { "Motor_State": 0 }
- 服务:** Agriculture
- 命令:** Agriculture_Control_Motor
- Motor:** ON

Below the interface is a flow diagram illustrating the communication between the IoT Platform and the Real Physical Device:

```
graph TD; IoT[IoT Platform] <-->|命令下发| RP[真实物理设备]; IoT <-->|数据上报| RP; RP <-->|命令下发| IoT
```

The diagram shows bidirectional dashed arrows labeled "命令下发" (Command Downlink) between the IoT Platform and the Real Physical Device, and a double-headed dashed arrow labeled "数据上报" (Data Report) between them.

5.3 练习题

5.3.1 下发 Light 和 Motor 所有开关命令

6

基于 NB-IoT 和 WIFI 的智慧烟感实验

6.1 实验介绍

6.1.1 关于本实验

本实验基于 NB-IoT 和 WIFI 实现智慧烟感案例，实现实时数据的采集，实现命令下发的响应，实现端云互通。

6.1.2 实验目的

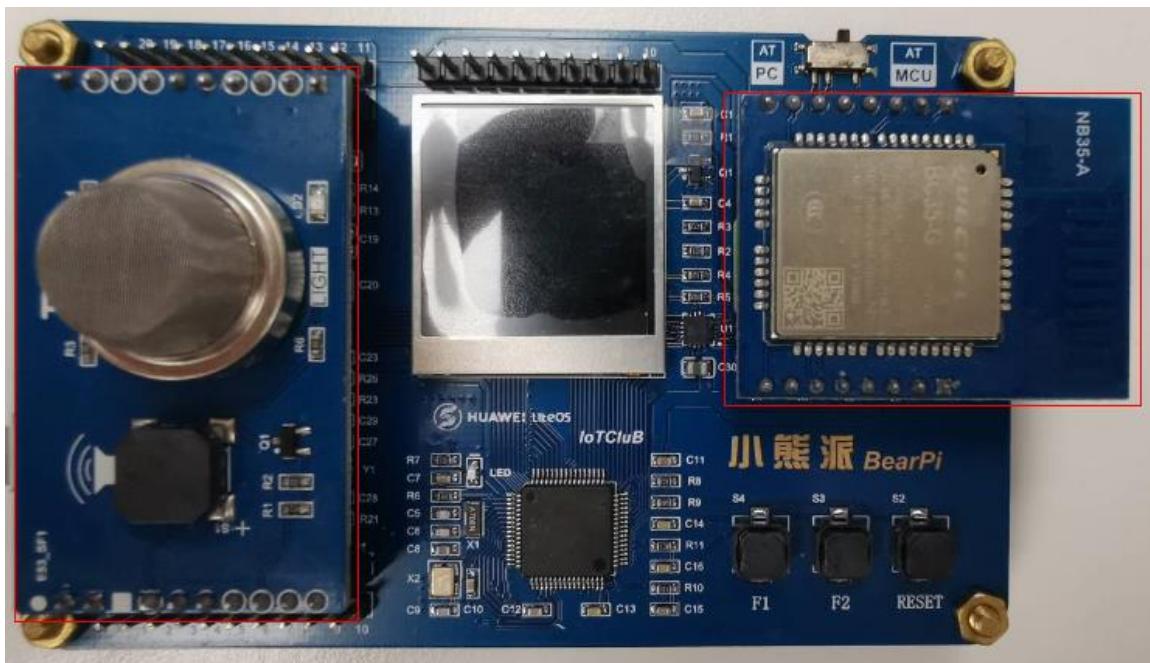
- 掌握智慧烟感案例的开发流程
- 掌握案例更换和通信方式更换的配置方法

6.2 实验任务配置步骤

6.2.1 配置智慧烟感案例

步骤 1 安装智慧烟感扩展板 E53_SF1

将智慧烟感扩展板 E53_SF1、NB 通信扩展板 NB35-A 按照正确的方向插入小熊派开发板；



步骤 2 修改.config 文件

打开 targets->STM32L431_BearPi->.config 文件；

修改“CONFIG_USER_DEMO”为“oc_smoke_template”；

“Ctrl+S”保存.config 文件；

```
使用   C hello_world_demo.c   .config  X
targets > STM32L431_BearPi > .config
40 CONFIG_SHELL_TASK_PRIORITY=10
41 CONFIG_IOT_LINK_CONFIGFILE="iot_config.h"
42 # Demo
43 CONFIG_USER_DEMO="oc_smoke_template" [highlight]
44 # NB-IoT
```

修改“CONFIG_UARTAT_BAUDRATE”波特率为“9600”；

```
使用   C hello_world_demo.c   .config  X
targets > STM32L431_BearPi > .config
2 CONFIG_ARCH_CPU_TYPE="armv7-m"
3 CONFIG_UARTAT_RCVMAX=2048
4 CONFIG_UARTAT_BAUDRATE=9600 [highlight]
5 CONFIG_UARTAT_DEVNAME="atdev"
```

将“CONFIG_BOUDICA150_ENABLE=y”前面#号去除，将 WIFI 配置前加上#号；

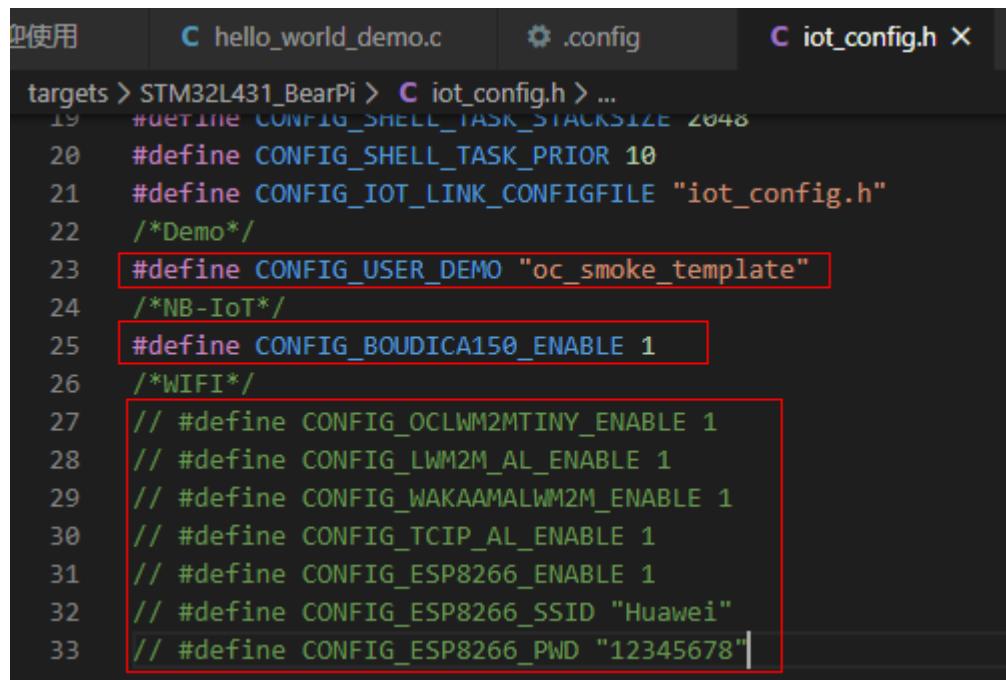
```
44 # NB-IoT
45 CONFIG_BOUDICA150_ENABLE=y
46 # WIFI
47 // #define CONFIG_OCLWM2MTINY_ENABLE 1
48 // #define CONFIG_LWM2M_AL_ENABLE 1
49 // #define CONFIG_WAKAAMALWM2M_ENABLE 1
50 // #define CONFIG_TCIP_AL_ENABLE 1
51 // #define CONFIG_ESP8266_ENABLE 1
52 // #define CONFIG_ESP8266_SSID "Huawei"
53 // #define CONFIG_ESP8266_PWD "12345678"
```

步骤 3 修改 iot_config.h 文件

打开 targets->STM32L431_BearPi->iot_config.h 文件；

修改“CONFIG_USER_DEMO”为“oc_agriculture_template”；

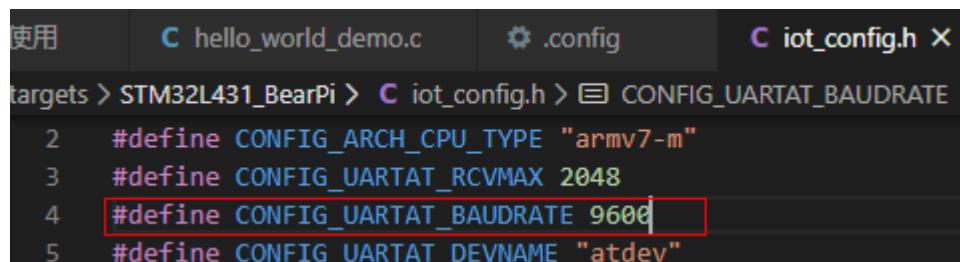
将“CONFIG_BOUDICA150_ENABLE=y”注释去除，将 WIFI 配置的注释掉；



```
targets > STM32L431_BearPi > C iot_config.h > ...
19 // #define CONFIG_SHELL_TASK_STACKSIZE 2048
20 #define CONFIG_SHELL_TASK_PRIOR 10
21 #define CONFIG_IOT_LINK_CONFIGFILE "iot_config.h"
22 /*Demo*/
23 #define CONFIG_USER_DEMO "oc_smoke_template"
24 /*NB-IoT*/
25 #define CONFIG_BOUDICA150_ENABLE 1
26 /*WIFI*/
27 // #define CONFIG_OCLWM2MTINY_ENABLE 1
28 // #define CONFIG_LWM2M_AL_ENABLE 1
29 // #define CONFIG_WAKAAMALWM2M_ENABLE 1
30 // #define CONFIG_TCIP_AL_ENABLE 1
31 // #define CONFIG_ESP8266_ENABLE 1
32 // #define CONFIG_ESP8266_SSID "Huawei"
33 // #define CONFIG_ESP8266_PWD "12345678"
```

修改“CONFIG_UARTAT_BAUDRATE”波特率为“9600”；

“Ctrl+S”保存 iot_config.h 文件；



```
targets > STM32L431_BearPi > C iot_config.h > CONFIG_UARTAT_BAUDRATE
2 #define CONFIG_ARCH_CPU_TYPE "armv7-m"
3 #define CONFIG_UARTAT_RCVMAX 2048
4 #define CONFIG_UARTAT_BAUDRATE 9600
5 #define CONFIG_UARTAT_DEVNAME "atdev"
```

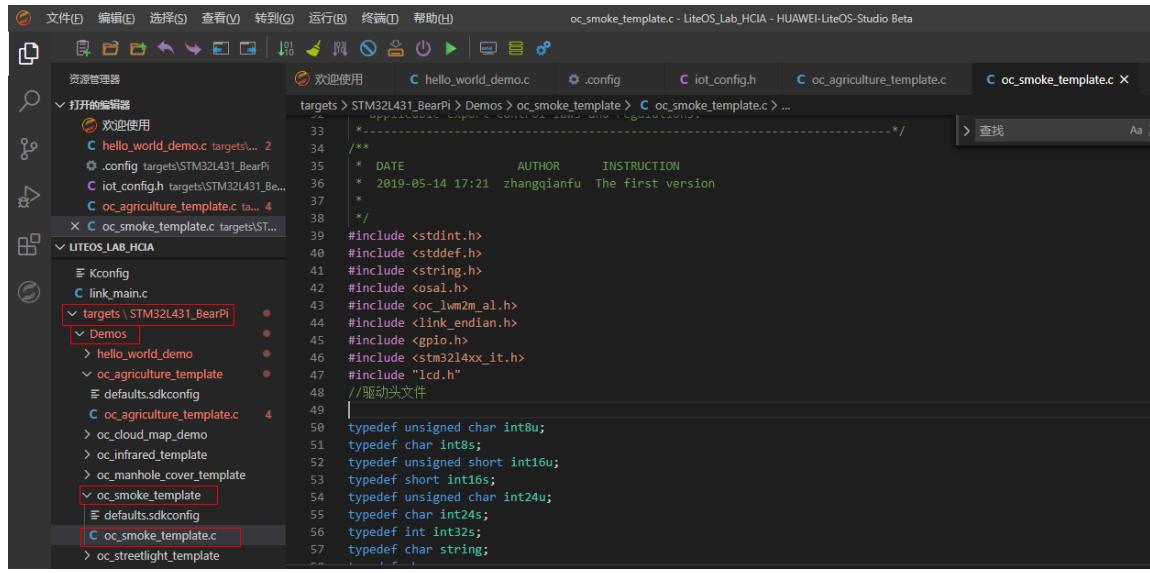
6.2.2 创建智慧烟感所需的数据结构

步骤 1 打开 oc_smoke_template.c 文件

打开案例程序文件 targets->STM32L431_BearPi->Demos->

oc_smoke_template->oc_smoke_template.c;

智慧烟感案例的代码都在 oc_smoke_template.c 中修改；



The screenshot shows the LiteOS Studio interface. The left sidebar displays a tree view of project files under '资源管理器' (Resource Manager). The 'Demos' folder contains several subfolders and files, with 'targets : STM32L431_BearPi' and 'oc_smoke_template' both highlighted with red boxes. The main code editor window shows the content of 'oc_smoke_template.c'. The code includes standard headers like stdint.h, stddef.h, string.h, osal.h, lwm2m.h, link_endian.h, gpio.h, and lcd.h. It also defines several typedefs for data types such as int8u, int8s, int16u, int16s, int24u, int24s, int32s, and string.

```
33 //-----  
34 /* DATE AUTHOR INSTRUCTION  
35 * 2019-05-14 17:21 zhangqianfu The first version  
36 *  
37 */  
38  
39 #include <stdint.h>  
40 #include <stddef.h>  
41 #include <string.h>  
42 #include <osal.h>  
43 #include <oc_lwm2m.h>  
44 #include <link_endian.h>  
45 #include <gpio.h>  
46 #include <stm32l4xx_it.h>  
47 #include "lcd.h"  
48 //驱动头文件  
49  
50 typedef unsigned char int8u;  
51 typedef char int8s;  
52 typedef unsigned short int16u;  
53 typedef short int16s;  
54 typedef unsigned char int24u;  
55 typedef char int24s;  
56 typedef int int32s;  
57 typedef char string;
```

步骤 2 添加智慧烟感案例扩展板驱动头文件

```
#include "E53_SF1.h"

47 #include "lcd.h"
48 //驱动头文件
49 #include "E53_SF1.h"
```

步骤 3 添加平台所需的基本变量

```
#define cn_endpoint_id      "BearPi_0001"
#define cn_app_server        "119.3.250.80"
#define cn_app_port          "5683"
```

```
62 //连接平台基本变量
63 #define cn_endpoint_id      "BearPi_0001"
64 #define cn_app_server        "119.3.250.80"
65 #define cn_app_port          "5683"
66 //MessageID
```

步骤 4 添加 MessageID

```
#define cn_app_Smoke 0x5
#define cn_app_response_Smoke_Control_Beep 0x7
#define cn_app_Smoke_Control_Beep 0x6

66 //MessageID
67 #define cn_app_Smoke 0x5
68 #define cn_app_response_Smoke_Control_Beep 0x7
69 #define cn_app_Smoke_Control_Beep 0x6
70 //消息结构体
```

步骤 5 添加消息结构体

```
#pragma pack(1)
typedef struct
{
    int8u messageId;
    int16u Smoke_Value;
} tag_app_Smoke;

typedef struct
{
    int8u messageId;
    int16u mid;
    int8u errcode;
    int8u Beep_State;
} tag_app_Response_Smoke_Control_Beep;

typedef struct
{
    int8u messageId;
    int16u mid;
    string Beep[3];
} tag_app_Smoke_Control_Beep;
#pragma pack()
```

```
70 //消息结构体
71 #pragma pack(1)
72 typedef struct
73 {
74     int8u messageId;
75     int16u Smoke_Value;
76 } tag_app_Smoke;
77
78 typedef struct
79 {
80     int8u messageId;
81     int16u mid;
82     int8u errcode;
83     int8u Beep_State;
84 } tag_app_Response_Smoke_Control_Beep;
85
86 typedef struct
87 {
88     int8u messageId;
89     int16u mid;
90     string Beep[3];
91 } tag_app_Smoke_Control_Beep;
92 #pragma pack()
93 //存储数据的变量
```

步骤 6 添加存储数据的变量

```
E53_SF1_Data_TypeDef E53_SF1_Data;
```

```
92 #pragma pack()
93 //存储数据的变量
94 E53_SF1_Data_TypeDef E53_SF1_Data;
95
```

6.2.3 创建数据收集任务

步骤 1 添加数据收集任务

```
static int app_collect_task_entry()
{
    Init_E53_SF1();
    while (1)
    {
        E53_SF1_Read_Data();
```

```
    printf("\r\n*****Smoke Value is %d\r\n",
(int)E53_SF1_Data.Smoke_Value);
LCD_ShowString(10, 170, 200, 16, 16, "Smoke Value is:");
LCD_ShowNum(140, 170, (int)E53_SF1_Data.Smoke_Value, 5, 16);

osal_task_sleep(2*1000);
}

return 0;
}
```

```
132 //数据收集任务
133 static int app_collect_task_entry()
134 {
135     Init_E53_SF1();
136     while (1)
137     {
138         E53_SF1_Read_Data();
139         printf("\r\n*****Smoke Value is %d\r\n", (int)E53_SF1_Data.Smoke_Value);
140         LCD_ShowString(10, 170, 200, 16, 16, "Smoke Value is:");
141         LCD_ShowNum(140, 170, (int)E53_SF1_Data.Smoke_Value, 5, 16);
142
143         osal_task_sleep(2*1000);
144     }
145
146     return 0;
147 }
148 int standard_app_demo_main()
```

步骤 2 创建数据收集任务

```
osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
```

```
148 int standard_app_demo_main()
149 {
150     osal_semp_create(&s_rcv_sync,1,0);
//LCD屏幕显示
152
153     //创建任务
154     osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
155
156 }
```

步骤 3 添加 LCD 屏幕显示代码

```
LCD_Clear(BLACK);
POINT_COLOR = GREEN;
LCD_ShowString(10, 10, 200, 16, 24, "Welcome to BearPi");
LCD_ShowString(40, 50, 200, 16, 24, "Smoke Demo");
LCD_ShowString(10, 90, 200, 16, 16, "NCDP_IP:");
LCD_ShowString(80, 90, 200, 16, 16, cn_app_server);
```

```
LCD_ShowString(10, 130, 200, 16, 16, "NCDP_PORT:");
LCD_ShowString(100, 130, 200, 16, 16, cn_app_port);
```

```
148 int standard_app_demo_main()
149 {
150     osal_semp_create(&s_rcv_sync, 1, 0);
151     //LCD屏幕显示
152     LCD_Clear(BLACK);
153     POINT_COLOR = GREEN;
154     LCD_ShowString(10, 10, 200, 16, 24, "Welcome to BearPi");
155     LCD_ShowString(40, 50, 200, 16, 24, "Smoke Demo");
156     LCD_ShowString(10, 90, 200, 16, 16, "NCDP_IP:");
157     LCD_ShowString(80, 90, 200, 16, 16, cn_app_server);
158     LCD_ShowString(10, 130, 200, 16, 16, "NCDP_PORT:");
159     LCD_ShowString(100, 130, 200, 16, 16, cn_app_port);
160     //创建任务
161     osal_task_create("app_collect", app_collect_task_entry, NULL, 0x400, NULL, 3);
162     return 0;
163 }
164
```

步骤 4 重新编译烧录，查看结果

点击  进行重新编译，烧录程序，在串口终端中可以看到有实时的数据打印；



在 LCD 屏幕上，也可以看到实时的数据显示；



6.2.4 创建数据上报任务

步骤 1 添加数据上报任务

```
static int app_report_task_entry()
{
    int ret = -1;

    oc_config_param_t      oc_param;
    tag_app_Smoke Smoke;

    (void) memset(&oc_param,0,sizeof(oc_param));

    oc_param.app_server.address = cn_app_server;
    oc_param.app_server.port = cn_app_port;
    oc_param.app_server.ep_id = cn_endpoint_id;
    oc_param.boot_mode = en_oc_boot_strap_mode_factory;
    oc_param.rcv_func = app_msg_deal;

    ret = oc_lwm2m_config(&oc_param);
    if (0 != ret)
    {
        return ret;
    }

    while(1) //--TODO ,you could add your own code here
    {
        Smoke.messageId = cn_app_Smoke;
        Smoke.Smoke_Value = htons((int)E53_SF1_Data.Smoke_Value);
        oc_lwm2m_report( (char *)&Smoke, sizeof(Smoke), 1000);
        osal_task_sleep(2*1000);
    }
    return ret;
}
```

```
130 //数据上报任务
131 static int app_report_task_entry()
132 {
133     int ret = -1;
134
135     oc_config_param_t      oc_param;
136     tag_app_Smoke Smoke;
137
138     (void) memset(&oc_param, 0, sizeof(oc_param));
139
140     oc_param.app_server.address = cn_app_server;
141     oc_param.app_server.port = cn_app_port;
142     oc_param.app_server.ep_id = cn_endpoint_id;
143     oc_param.boot_mode = en_oc_boot_strap_mode_factory;
144     oc_param.recv_func = app_msg_deal;
145
146     ret = oc_lwm2m_config(&oc_param);
147     if (0 != ret)
148     {
149         return ret;
150     }
151
152     while(1) //--TODO ,you could add your own code here
153     {
154         Smoke.messageId = cn_app_Smoke;
155         Smoke.Smoke_Value = htons((int)E53_SF1_Data.Smoke_Value);
156         oc_lwm2m_report( (char *)&Smoke, sizeof(Smoke), 1000);
157         osal_task_sleep(2*1000);
158     }
159     return ret;
160 }
161 //数据收集任务
162 static int app_collect_task_entry()
```

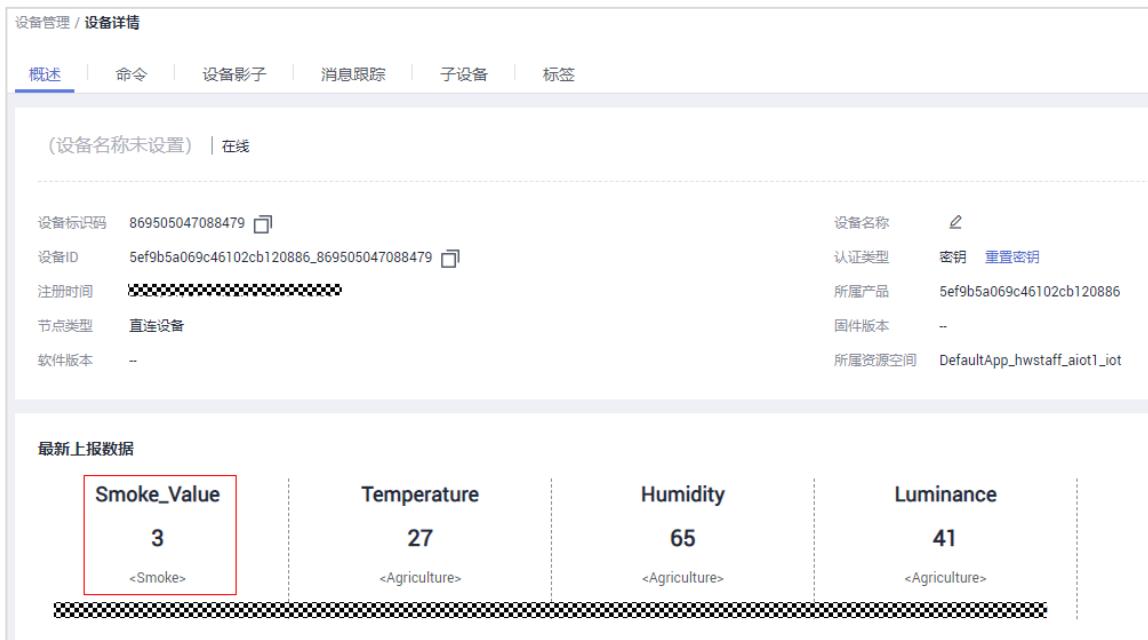
步骤 2 创建数据上报任务

```
osal_task_create("app_report",app_report_task_entry,NULL,0x1000,NULL,2);

177 int standard_app_demo_main()
178 {
179     osal_semp_create(&s_rcv_sync,1,0);
180     //LCD屏幕显示
181     LCD_Clear(BLACK);
182     POINT_COLOR = GREEN;
183     LCD_ShowString(10, 10, 200, 16, 24, "Welcome to BearPi");
184     LCD_ShowString(40, 50, 200, 16, 24, "Smoke Demo");
185     LCD_ShowString(10, 90, 200, 16, 16, "NCDP_IP:");
186     LCD_ShowString(80, 90, 200, 16, 16, cn_app_server);
187     LCD_ShowString(10, 130, 200, 16, 16, "NCDP_PORT:");
188     LCD_ShowString(100, 130, 200, 16, 16, cn_app_port);
189     //创建任务
190     osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
191     osal_task_create("app_report",app_report_task_entry,NULL,0x1000,NULL,2);
192     return 0;
193 }
```

步骤 3 编译烧录，查看结果

编译烧录程序，在物联网平台中，可以看到实时的数据；



The screenshot shows the 'Device Management / Device Details' page. At the top, there are tabs: 概述 (Overview), 命令 (Commands), 设备影子 (Device Shadow), 消息跟踪 (Message Tracing), 子设备 (Sub-device), and 标签 (Tags). The '概述' tab is selected.

设备名称未设置 | 在线

| | | | |
|-------|--|--------|------------------------------|
| 设备标识码 | 869505047088479 | 设备名称 | |
| 设备ID | 5ef9b5a069c46102cb120886_869505047088479 | 认证类型 | 密钥 重置密钥 |
| 注册时间 | xxxxxxxxxxxxxx | 所属产品 | 5ef9b5a069c46102cb120886 |
| 节点类型 | 直连设备 | 固件版本 | -- |
| 软件版本 | - | 所属资源空间 | DefaultApp_hwstaff_aiot1_iot |

最新上报数据

| Smoke_Value | Temperature | Humidity | Luminance |
|--------------|---------------------|---------------------|---------------------|
| 3 <Smoke> | 27 <Agriculture> | 65 <Agriculture> | 41 <Agriculture> |

6.2.5 创建命令响应任务

步骤 1 添加命令处理任务

```
static int app_cmd_task_entry()
{
    int ret = -1;
    tag_app_Response_Smoke_Control_Beep Response_Smoke_Control_Beep;
    tag_app_Smoke_Control_Beep *Smoke_Control_Beep;
    int8_t msgid;

    while(1)
    {
        if(osal_semp_pend(s_rcv_sync,cn_osal_timeout_forever))
        {
            msgid = s_rcv_buffer[0] & 0x000000FF;
            switch (msgid)
            {
                case cn_app_Smoke_Control_Beep:
                    Smoke_Control_Beep = (tag_app_Smoke_Control_Beep
*)s_rcv_buffer;
                    printf("Smoke_Control_Beep:msgid:%d mid:%d",
Smoke_Control_Beep->messageId, ntohs(Smoke_Control_Beep->mid));
                    /****** code area for cmd from IoT cloud *****/
                    if (Smoke_Control_Beep->Beep[0] == 'O' &&
Smoke_Control_Beep->Beep[1] == 'N')
                    {
                        E53_SF1_Beep_StatusSet(ON);
                        Response_Smoke_Control_Beep.messageId =
cn_app_response_Smoke_Control_Beep;
                        Response_Smoke_Control_Beep.mid =
Smoke_Control_Beep->mid;
                        Response_Smoke_Control_Beep.errcode = 0;
                        Response_Smoke_Control_Beep.Beep_State = 1;
                        oc_lwm2m_report((char
*)&Response_Smoke_Control_Beep,sizeof(Response_Smoke_Control_Beep),1000);
//< report cmd reply message
                    }
            }
        }
    }
}
```



```

        if (Smoke_Control_Beep->Beep[0] == 'O' &&
Smoke_Control_Beep->Beep[1] == 'F' && Smoke_Control_Beep->Beep[2] == 'F')
    {
        E53_SF1_Beep_StatusSet(OFF);
        Response_Smoke_Control_Beep.messageId =
cn_app_response_Smoke_Control_Beep;
        Response_Smoke_Control_Beep.mid =
Smoke_Control_Beep->mid;
        Response_Smoke_Control_Beep.errcode = 0;
        Response_Smoke_Control_Beep.Beep_State = 0;
        oc_lwm2m_report((char
*)&Response_Smoke_Control_Beep,sizeof(Response_Smoke_Control_Beep),1000);
///< report cmd reply message
    }
    /***** code area end *****/
    break;
default:
    break;
}
}

return ret;
}

```

```
128 //下发命令处理任务
129 static int app_cmd_task_entry()
130 {
131     int ret = -1;
132     tag_app_Response_Smoke_Control_Beep Response_Smoke_Control_Beep;
133     tag_app_Smoke_Control_Beep *Smoke_Control_Beep;
134     int8_t msgid;
135
136     while(1)
137     {
138         if(osal_semp_pend(s_rcv_sync,cn_osal_timeout_forever))
139         {
140             msgid = s_rcv_buffer[0] & 0x000000FF;
141             switch (msgid)
142             {
143                 case cn_app_Smoke_Control_Beep:
144                     Smoke_Control_Beep = (tag_app_Smoke_Control_Beep *)s_rcv_buffer;
145                     printf("Smoke_Control_Beep:msgid=%d mid=%d", Smoke_Control_Beep->messageId, ntohs(Smoke_Control_Beep->mid));
146                     //***** code area for cmd from IoT cloud *****/
147                     if (Smoke_Control_Beep->Beep[0] == 'O' && Smoke_Control_Beep->Beep[1] == 'N')
148                     {
149                         E53_SF1_Beep_StatusSet(ON);
150                         Response_Smoke_Control_Beep.messageId = cn_app_response_Smoke_Control_Beep;
151                         Response_Smoke_Control_Beep.mid = Smoke_Control_Beep->mid;
152                         Response_Smoke_Control_Beep.errcode = 0;
153                         Response_Smoke_Control_Beep.Beep_State = 1;
154                         oc_lwm2m_report((char *)&Response_Smoke_Control_Beep,sizeof(Response_Smoke_Control_Beep),1000);    // < report cmd
155                     }
156                     if (Smoke_Control_Beep->Beep[0] == 'O' && Smoke_Control_Beep->Beep[1] == 'F' && Smoke_Control_Beep->Beep[2] == 'F')
157                     {
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1290
1291
1292
1293
1294
1295
1295
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1495
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1595
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1695
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1989
1989
1990
1991
1992
1993
1994
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2195
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2295
2296
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2387
2388
2389
2389
2390
2391
2392
2393
2394
2395
2395
2396
2397
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2
```

```
158     E53_SF1_Beep_StatusSet(OFF);
159     Response_Smoke_Control_Beep.messageId = cn_app_response_Smoke_Control_Beep;
160     Response_Smoke_Control_Beep.mid = Smoke_Control_Beep->mid;
161     Response_Smoke_Control_Beep.errcode = 0;
162     Response_Smoke_Control_Beep.Beep_State = 0;
163     oc_lwm2m_report((char *)&Response_Smoke_Control_Beep,sizeof(Response_Smoke_Control_Beep),1000); ///< report cmd rep
164 }
165 /**
166 * code area end ****
167 */
168 default:
169     break;
170 }
171 }
172
173 return ret;
174 //数据上报任务
175 }
```

步骤 2 创建命令处理任务

```
osal_task_create("app_command",app_cmd_task_entry,NULL,0x1000,NULL,3);
```

```
222 int standard_app_demo_main()
223 {
224     osal_semp_create(&s_rcv_sync,1,0);
225     //LCD屏幕显示
226     LCD_Clear(BLACK);
227     POINT_COLOR = GREEN;
228     LCD_ShowString(10, 10, 200, 16, 24, "Welcome to BearPi");
229     LCD_ShowString(40, 50, 200, 16, 24, "Smoke Demo");
230     LCD_ShowString(10, 90, 200, 16, 16, "NCDP_IP:");
231     LCD_ShowString(80, 90, 200, 16, 16, cn_app_server);
232     LCD_ShowString(10, 130, 200, 16, 16, "NCDP_PORT:");
233     LCD_ShowString(100, 130, 200, 16, 16, cn_app_port);
234     //创建任务
235     osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
236     osal_task_create("app_report",app_report_task_entry,NULL,0x1000,NULL,2);
237     osal_task_create("app_command",app_cmd_task_entry,NULL,0x1000,NULL,3);
238
239 }
240
```

步骤 3 编译烧录，查看结果

编译烧录程序，在物联网平台中下发开灯命令，查看命令下发结果及开发板的响应；



6.2.6 使用 WIFI 通信方式

步骤 1 安装 WIFI8266 通信扩展板

将 NB35-A 通信扩展板替换为 WIFI8266 通信扩展板，天线朝外；

步骤 2 修改设备标识码

在 oc_smoke_template.c 中，

修改“cn_endpoint_id”为前面实验使用 AT 指令查询到的 IMEI 号；

```
#define cn_endpoint_id "869505047088479"
```

```
62 //连接平台基本变量  
63 #define cn_endpoint_id "869505047088479"  
64 #define cn_app_server "119.3.250.80"  
65 #define cn_app_port "5683"
```

步骤 3 修改.config 文件并保存

打开.config 文件，将“CONFIG_UARTAT_BAUDRATE”波特率设置为“115200”；

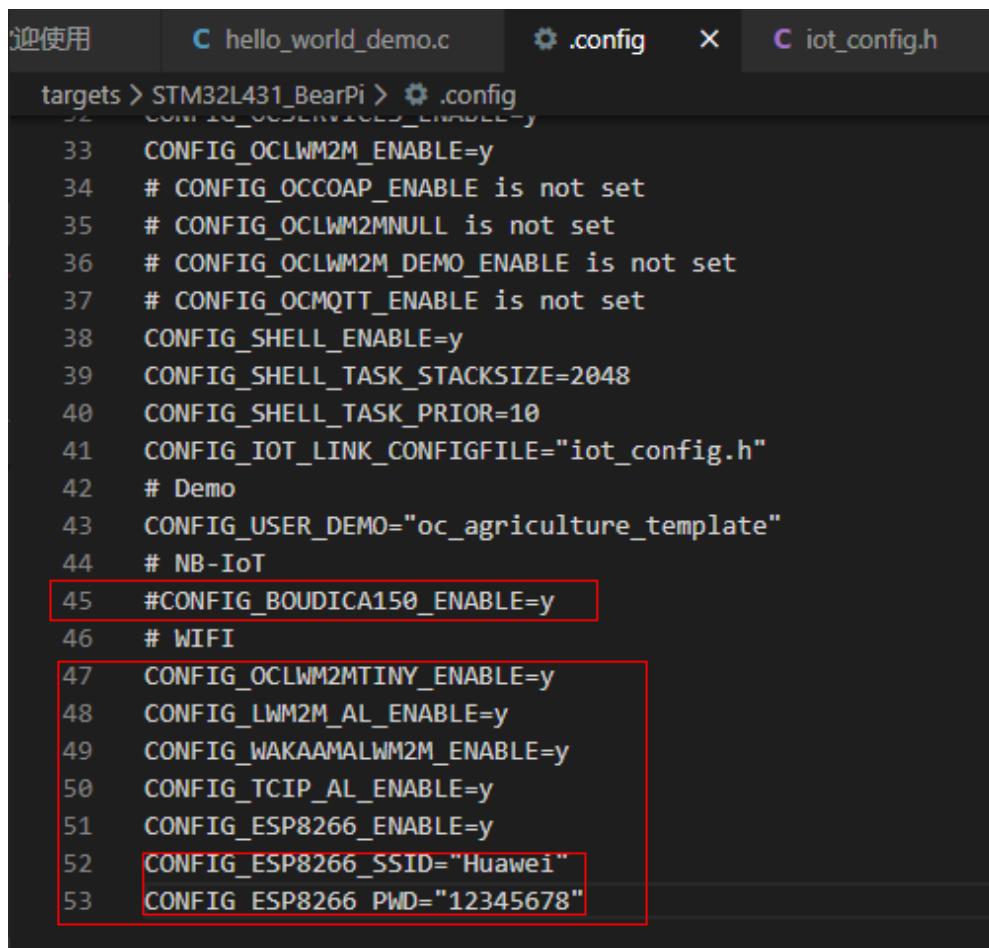
```
targets > STM32L431_BearPi > .config  
2 CONFIG_ARCH_CPU_TYPE="armv7-m"  
3 CONFIG_UARTAT_RCVMAX=2048  
4 CONFIG_UARTAT_BAUDRATE=115200  
5 CONFIG_UARTAT_DEVNAME="atdev"
```

将“CONFIG_BOUDICA150_ENABLE=y”前面加上#号，将 WIFI 配置前的#号去除；

并将 SSID 和 PWD 修改为自己手机热点或者路由器的用户名密码；

开发板将通过 WIFI8266 扩展板连接到配置的热点中；

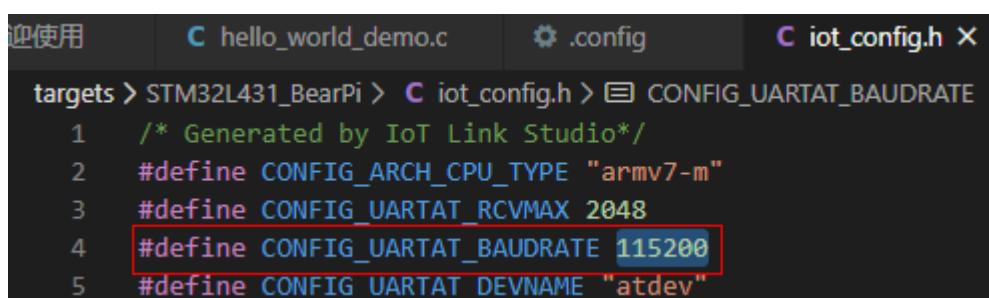
```
# NB-IoT  
#CONFIG_BOUDICA150_ENABLE=y  
# WIFI  
CONFIG_OCLWM2MTINY_ENABLE=y  
CONFIG_LWM2M_AL_ENABLE=y  
CONFIG_WAKAAMALWM2M_ENABLE=y  
CONFIG_TCIP_AL_ENABLE=y  
CONFIG_ESP8266_ENABLE=y  
CONFIG_ESP8266_SSID="XXXXXX"  
CONFIG_ESP8266_PWD="XXXXXX"
```



```
迎使用 C hello_world_demo.c .config C iot_config.h
targets > STM32L431_BearPi > .config
32 CONFIG_OCLWM2M_ENABLE=y
33 CONFIG_OCLWM2M_ENABLE=y
34 # CONFIG_OCCOAP_ENABLE is not set
35 # CONFIG_OCLWM2MNULL is not set
36 # CONFIG_OCLWM2M_DEMO_ENABLE is not set
37 # CONFIG_OCMQTT_ENABLE is not set
38 CONFIG_SHELL_ENABLE=y
39 CONFIG_SHELL_TASK_STACKSIZE=2048
40 CONFIG_SHELL_TASK_PRIOR=10
41 CONFIG_IOT_LINK_CONFIGFILE="iot_config.h"
42 # Demo
43 CONFIG_USER_DEMO="oc_agriculture_template"
44 # NB-IoT
45 #CONFIG_BOUDICA150_ENABLE=y
46 # WIFI
47 CONFIG_OCLWM2MTINY_ENABLE=y
48 CONFIG_LWM2M_AL_ENABLE=y
49 CONFIG_WAKAAMALWM2M_ENABLE=y
50 CONFIG_TCIP_AL_ENABLE=y
51 CONFIG_ESP8266_ENABLE=y
52 CONFIG_ESP8266_SSID="Huawei"
53 CONFIG_ESP8266_PWD="12345678"
```

步骤 4 修改 iot_config.h 文件并保存

打开 iot_config.h 文件，将“CONFIG_UARTAT_BAUDRATE”波特率设置为“115200”；

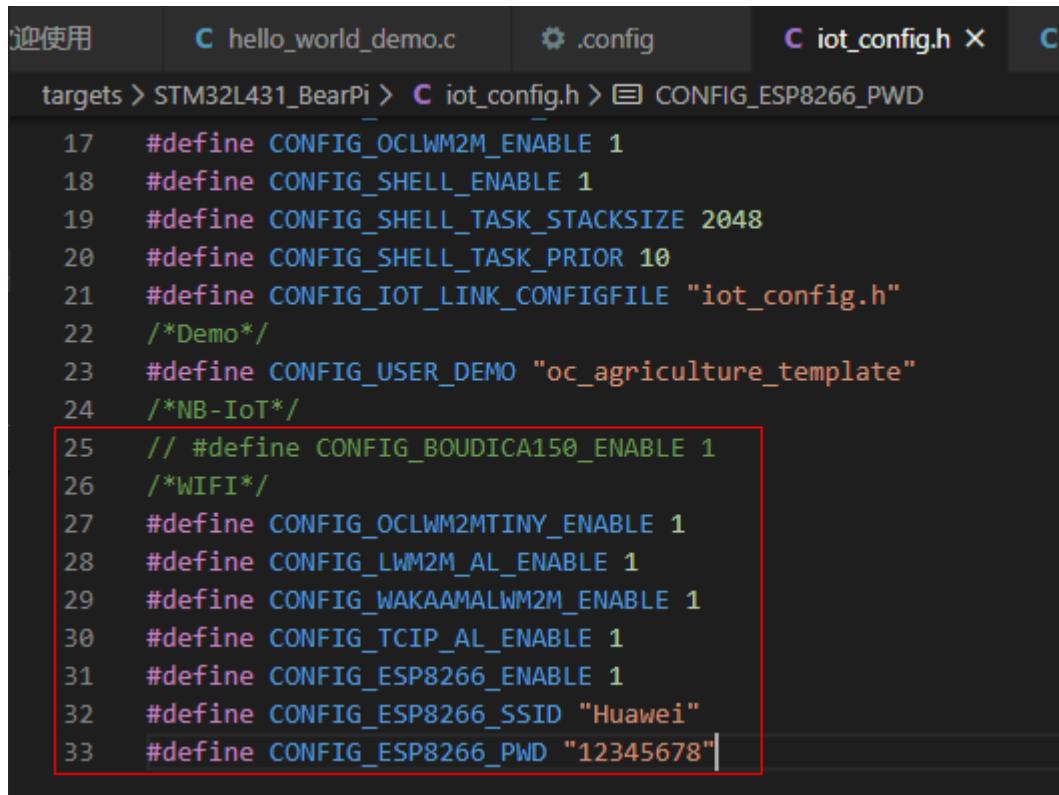


```
迎使用 C hello_world_demo.c .config C iot_config.h X
targets > STM32L431_BearPi > C iot_config.h > CONFIG_UARTAT_BAUDRATE
1 /* Generated by IoT Link Studio*/
2 #define CONFIG_ARCH_CPU_TYPE "armv7-m"
3 #define CONFIG_UARTAT_RCVMAX 2048
4 #define CONFIG_UARTAT_BAUDRATE 115200
5 #define CONFIG_UARTAT_DEVNAME "atdev"
```

将“CONFIG_BOUDICA150_ENABLE=y”注释掉；

将 WIFI 配置的注释去除，并将 CONFIG_ESP8266_SSID 和 CONFIG_ESP8266_PWD 修改为和.config 中一致的用户名密码；

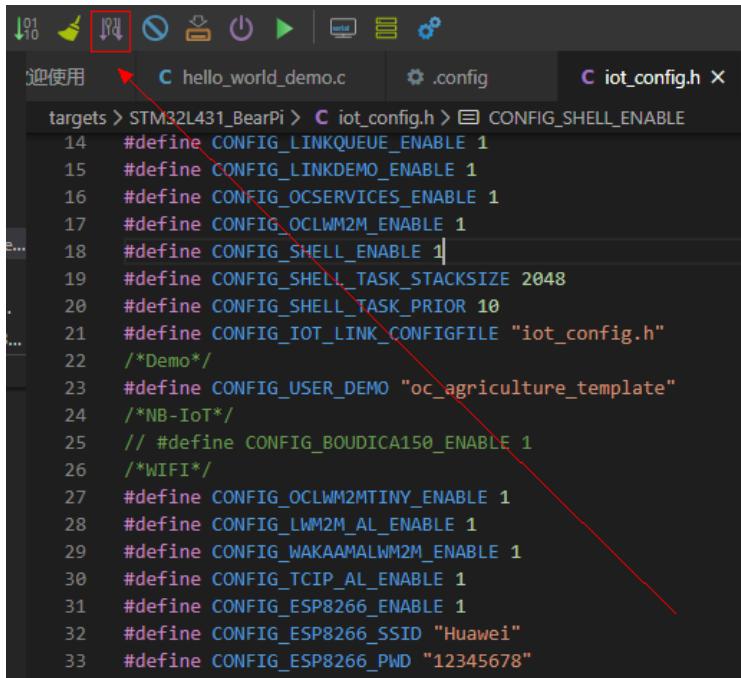
```
/*NB-IoT*/
// #define CONFIG_BOUDICA150_ENABLE 1
/*WIFI*/
#define CONFIG_OCLWM2MTINY_ENABLE 1
#define CONFIG_LWM2M_AL_ENABLE 1
#define CONFIG_WAKAAMALWM2M_ENABLE 1
#define CONFIG_TCIP_AL_ENABLE 1
#define CONFIG_ESP8266_ENABLE 1
#define CONFIG_ESP8266_SSID "XXXXXX"
#define CONFIG_ESP8266_PWD "XXXXXX"
```



```
targets > STM32L431_BearPi > iot_config.h > CONFIG_ESP8266_PWD
17 #define CONFIG_OCLWM2M_ENABLE 1
18 #define CONFIG_SHELL_ENABLE 1
19 #define CONFIG_SHELL_TASK_STACKSIZE 2048
20 #define CONFIG_SHELL_TASK_PRIOR 10
21 #define CONFIG_IOT_LINK_CONFIGFILE "iot_config.h"
22 /*Demo*/
23 #define CONFIG_USER_DEMO "oc_agriculture_template"
24 /*NB-IoT*/
25 // #define CONFIG_BOUDICA150_ENABLE 1
26 /*WIFI*/
27 #define CONFIG_OCLWM2MTINY_ENABLE 1
28 #define CONFIG_LWM2M_AL_ENABLE 1
29 #define CONFIG_WAKAAMALWM2M_ENABLE 1
30 #define CONFIG_TCIP_AL_ENABLE 1
31 #define CONFIG_ESP8266_ENABLE 1
32 #define CONFIG_ESP8266_SSID "Huawei"
33 #define CONFIG_ESP8266_PWD "12345678"
```

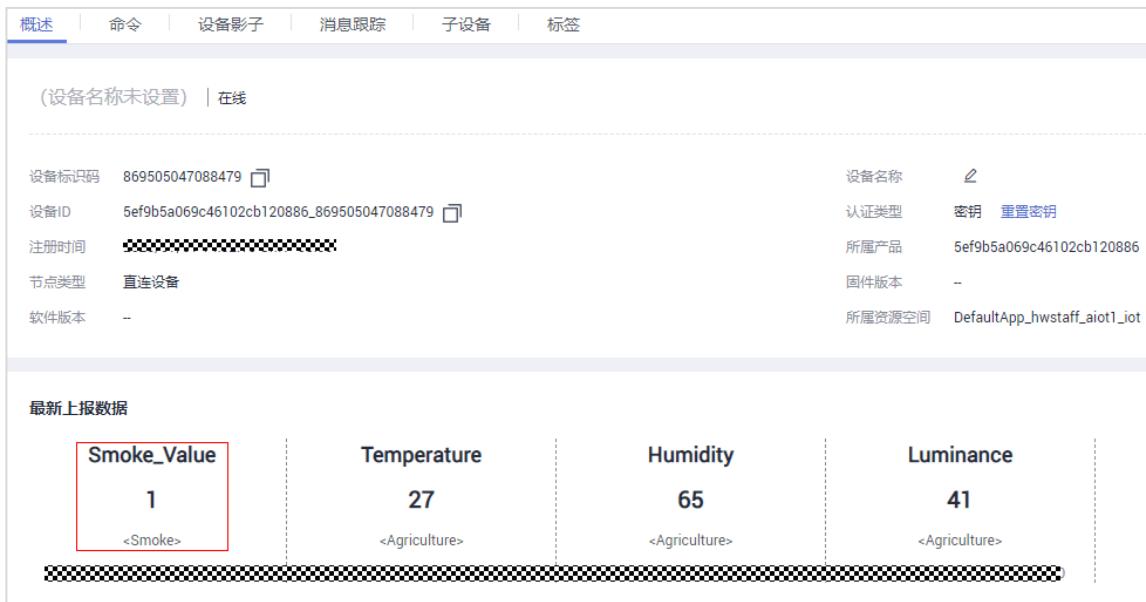
步骤 5 重新编译烧录，查看结果

打开手机热点，点击  进行重新编译；



```
targets > STM32L431_BearPi > C iot_config.h > CONFIG_SHELL_ENABLE
14 #define CONFIG_LINKQUEUE_ENABLE 1
15 #define CONFIG_LINKDEMO_ENABLE 1
16 #define CONFIG_OCSERVICES_ENABLE 1
17 #define CONFIG_OCLWM2M_ENABLE 1
18 #define CONFIG_SHELL_ENABLE 1
19 #define CONFIG_SHELL_TASK_STACKSIZE 2048
20 #define CONFIG_SHELL_TASK_PRIOR 10
21 #define CONFIG_IOT_LINK_CONFIGFILE "iot_config.h"
22 /*Demo*/
23 #define CONFIG_USER_DEMO "oc_agriculture_template"
24 /*NB-IoT*/
25 // #define CONFIG_BOUDICA150_ENABLE 1
26 /*WIFI*/
27 #define CONFIG_OCLWM2MTINY_ENABLE 1
28 #define CONFIG_LWM2M_AL_ENABLE 1
29 #define CONFIG_WAKAAMALWM2M_ENABLE 1
30 #define CONFIG_TCIP_AL_ENABLE 1
31 #define CONFIG_ESP8266_ENABLE 1
32 #define CONFIG_ESP8266_SSID "Huawei"
33 #define CONFIG_ESP8266_PWD "12345678"
```

烧录程序，在物联网平台中查看数据上报和命令下发功能；



概述 | 命令 | 设备影子 | 消息跟踪 | 子设备 | 标签

(设备名称未设置) | 在线

| | | | |
|-------|--|--------|---|
| 设备标识码 | 869505047088479 | 设备名称 |  |
| 设备ID | 5ef9b5a069c46102cb120886_869505047088479 | 认证类型 | 密钥 重置密钥 |
| 注册时间 | 2024-01-15 10:23:15 | 所属产品 | 5ef9b5a069c46102cb120886 |
| 节点类型 | 直连设备 | 固件版本 | -- |
| 软件版本 | -- | 所属资源空间 | DefaultApp_hwstaff_aiot1_iot |

最新上报数据

| Smoke_Value | Temperature | Humidity | Luminance |
|--------------|---------------------|---------------------|---------------------|
| 1 <Smoke> | 27 <Agriculture> | 65 <Agriculture> | 41 <Agriculture> |

应用模拟器

全部 数据接收 命令发送

发送消息body信息: { "serviceId": "Smoke", "method": "Smoke_Control_Beep", "paras": "({\"Beep\":\"OFF\"})" }

命令状态:执行成功

命令响应时间: 00:00:00.000

命令响应内容: { "Beep_State": 0 }

服务: Smoke

命令: Smoke_Control_Beep

Beep: ON

缓存发送 立即发送

设置时间 ▾



6.3 练习题

6.3.1 使用按键控制蜂鸣器

7

基于 NB-IoT 和 WIFI 的智慧井盖实验

7.1 实验介绍

7.1.1 关于本实验

本实验基于 NB-IoT 和 WIFI 实现智慧井盖案例，实现实时数据的采集，实现端云互通。

7.1.2 实验目的

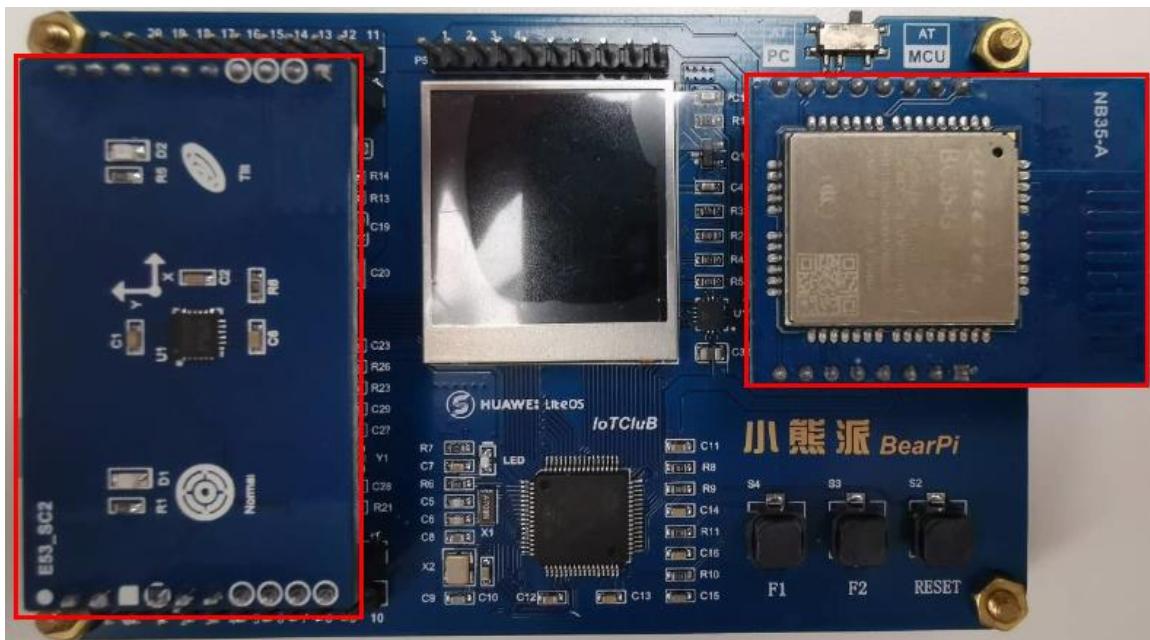
- 掌握 NB-IoT 通信方式的配置
- 掌握 WIFI 通信方式的配置
- 掌握智慧井盖案例的开发过程

7.2 实验任务配置步骤

7.2.1 配置智慧井盖案例

步骤 1 安装智慧井盖扩展板 E53_SC2

将智慧井盖扩展板 E53_SC2、NB 通信扩展板 NB35-A 按照正确的方向插入小熊派开发板；



步骤 2 修改.config 文件

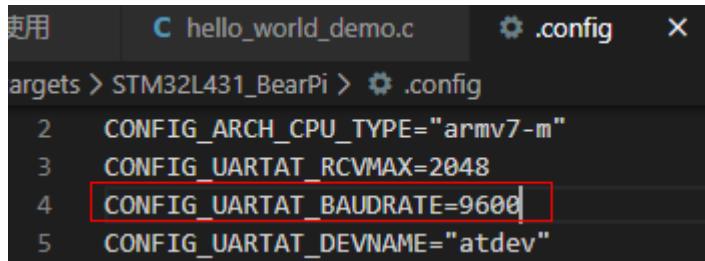
打开 targets->STM32L431_BearPi->.config 文件；

修改“CONFIG_USER_DEMO”为“oc_manhole_cover_template”；

“Ctrl+S”保存.config 文件；

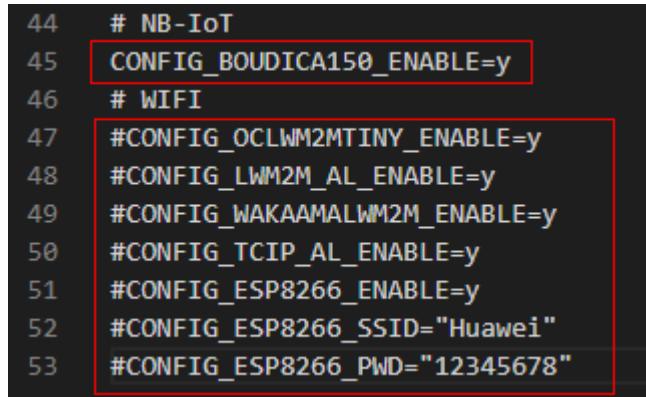
```
37      # CONFIG_OCMQTT_ENABLE is not set
38  CONFIG_SHELL_ENABLE=y
39  CONFIG_SHELL_TASK_STACKSIZE=2048
40  CONFIG_SHELL_TASK_PRIOR=10
41  CONFIG_IOT_LINK_CONFIGFILE="iot_config.h"
42  # Demo
43  CONFIG_USER_DEMO='oc_manhole_cover_template' -----
44  # NB-IoT
45  #CONFIG_BOUDICA150_ENABLE=y
46  # WIFI
47  CONFIG_OCLWM2MTINY_ENABLE=y
48  CONFIG_LWM2M_AL_ENABLE=y
49  CONFIG_WAKAAMALWM2M_ENABLE=y
50  CONFIG_TCIP_AL_ENABLE=y
51  CONFIG_ESP8266_ENABLE=y
52  CONFIG_ESP8266_SSID="Huawei"
53  CONFIG_ESP8266_PWD="12345678"
```

修改“CONFIG_UARTAT_BAUDRATE”波特率为“9600”；



```
使用 C hello_world_demo.c .config X
targets > STM32L431_BearPi > .config
2 CONFIG_ARCH_CPU_TYPE="armv7-m"
3 CONFIG_UARTAT_RCVMAX=2048
4 CONFIG_UARTAT_BAUDRATE=9600
5 CONFIG_UARTAT_DEVNAME="atdev"
```

将“CONFIG_BOUDICA150_ENABLE=y”前面#号去除，将 WIFI 配置前加上#号；



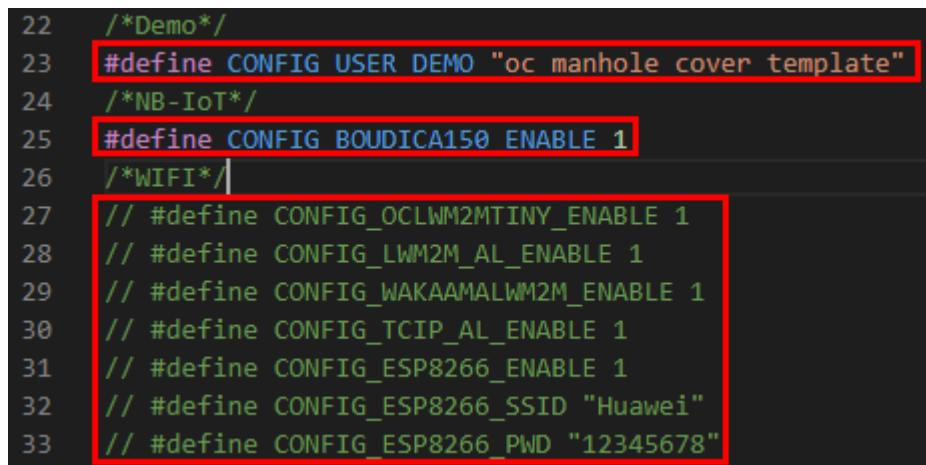
```
44 # NB-IoT
45 CONFIG_BOUDICA150_ENABLE=y
46 # WIFI
47 #CONFIG_OCLWM2MTINY_ENABLE=y
48 #CONFIG_LWM2M_AL_ENABLE=y
49 #CONFIG_WAKAAMALWM2M_ENABLE=y
50 #CONFIG_TCIP_AL_ENABLE=y
51 #CONFIG_ESP8266_ENABLE=y
52 #CONFIG_ESP8266_SSID="Huawei"
53 #CONFIG_ESP8266_PWD="12345678"
```

步骤 3 修改 iot_config.h 文件

打开 targets->STM32L431_BearPi->iot_config.h 文件；

修改“CONFIG_USER_DEMO”为“oc_mahole_cover_template”；

将“CONFIG_BOUDICA150_ENABLE=y”注释去除，将 WIFI 配置的注释掉；



```
22 /*Demo*/
23 #define CONFIG_USER_DEMO "oc_mahole_cover_template"
24 /*NB-IoT*/
25 #define CONFIG_BOUDICA150_ENABLE 1
26 /*WIFI*/
27 // #define CONFIG_OCLWM2MTINY_ENABLE 1
28 // #define CONFIG_LWM2M_AL_ENABLE 1
29 // #define CONFIG_WAKAAMALWM2M_ENABLE 1
30 // #define CONFIG_TCIP_AL_ENABLE 1
31 // #define CONFIG_ESP8266_ENABLE 1
32 // #define CONFIG_ESP8266_SSID "Huawei"
33 // #define CONFIG_ESP8266_PWD "12345678"
```

修改“CONFIG_UARTAT_BAUDRATE”波特率为“9600”；

“Ctrl+S”保存 iot_config.h 文件；

```
1  /* Generated by IoT Link Studio*/
2  #define CONFIG_ARCH_CPU_TYPE "armv7-m"
3  #define CONFIG_UARTAT_RCVMAX 2048
4  #define CONFIG_UARTAT_BAUDRATE 9600
5  #define CONFIG_UARTAT_DEVNAME "atdev"
6  #define CONFIG_LITEOS_ENABLE 1
7  #define CONFIG_AT_ENABLE 1
```

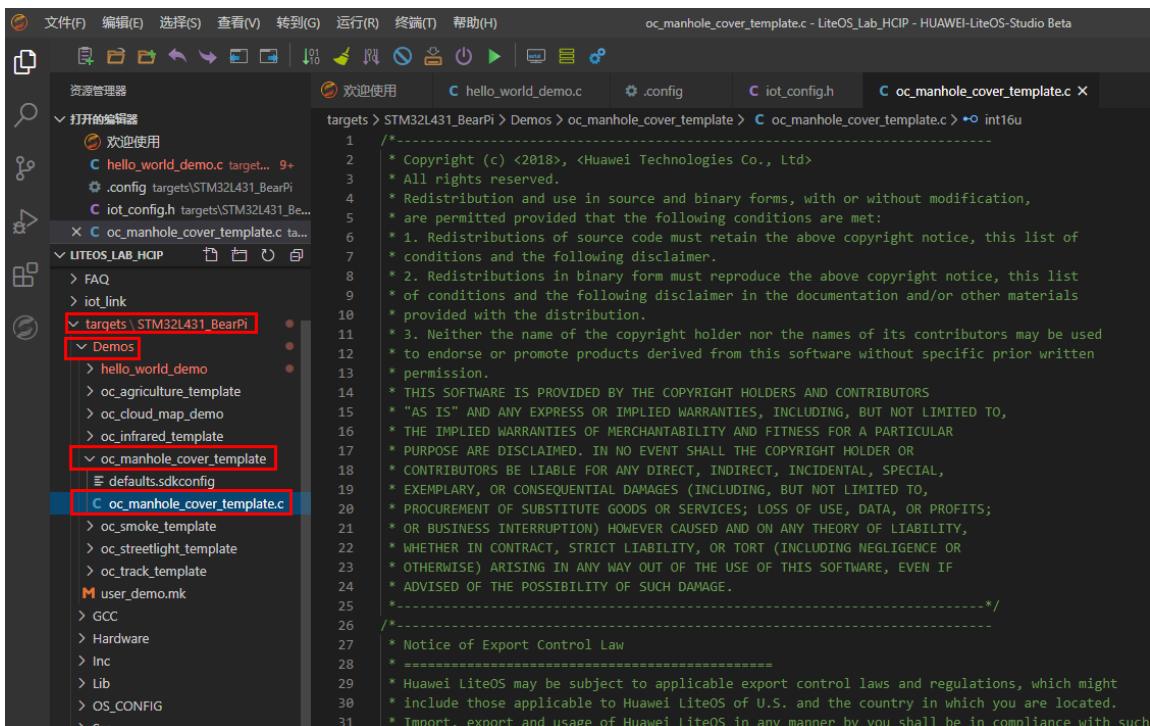
7.2.2 创建智慧井盖所需的数据结构

步骤 1 打开 oc_mahole_cover_template.c 文件

打开案例程序文件 targets->STM32L431_BearPi->Demos->

oc_mahole_cover_template->oc_mahole_cover_template.c;

智慧井盖案例的代码都在 oc_mahole_cover_template.c 中修改；



步骤 2 添加智慧井盖案例扩展板驱动头文件

```
#include "E53_SC2.h"

48 //驱动头文件
49 #include "E53_SC2.h"
```

步骤 3 添加平台所需的基本变量

```
#define cn_endpoint_id      "BearPi_0001"  
#define cn_app_server        "119.3.250.80"  
#define cn_app_port          "5683"  
  
50   typedef unsigned char int8u;  
51   typedef char int8s;  
52   typedef unsigned short int16u;  
53   typedef short int16s;  
54   typedef unsigned char int24u;  
55   typedef char int24s;  
56   typedef int int32s;  
57   typedef char string;  
58   typedef char array;  
59   typedef char varstring;  
60   typedef char variant;  
61  
62   //连接平台基本变量  
63   #define cn_endpoint_id      "BearPi_0001"  
64   #define cn_app_server        "119.3.250.80"  
65   #define cn_app_port          "5683"
```

步骤 4 添加 MessageID

```
#define cn_app_Manhole_Cover 0xb  
  
66   //MessageID  
67   #define cn_app_Manhole_Cover 0xb
```

步骤 5 添加消息结构体

```
#pragma pack(1)  
  
typedef struct  
{  
    int8u messageId;  
    int8_t Temperature;  
    int16_t Accel_x;  
    int16_t Accel_y;  
    int16_t Accel_z;  
    string Status[5];  
} tag_app_Manhole_Cover;  
#pragma pack()
```

```
68 //消息结构体
69 #pragma pack(1)
70 typedef struct
71 {
72     int8u messageId;
73     int8_t Temperature;
74     int16_t Accel_x;
75     int16_t Accel_y;
76     int16_t Accel_z;
77     string Status[5];
78 } tag_app_Manhole_Cover;
79 #pragma pack()
80 //存储数据的变量
```

步骤 6 添加存储数据的变量

```
E53_SC2_Data_TypeDef E53_SC2_Data;
tag_app_Manhole_Cover Manhole_Cover;
```

```
80 //存储数据的变量
81 E53_SC2_Data_TypeDef E53_SC2_Data;
82 tag app Manhole Cover Manhole Cover;
```

7.2.3 创建数据收集任务

步骤 1 添加数据收集任务

```
static int app_collect_task_entry()
{
    int X = 0,Y = 0,Z = 0;
    Init_E53_SC2();
    while (1)
    {
        E53_SC2_Read_Data();
        printf("\r\n*****Temperature      is %d\r\n",
(int)E53_SC2_Data.Temperature);
        printf("\r\n*****Accel[0]          is %d\r\n",
(int)E53_SC2_Data.Accel[0]);
        printf("\r\n*****Accel[1]          is %d\r\n",
(int)E53_SC2_Data.Accel[1]);
        printf("\r\n*****Accel[2]          is %d\r\n",
(int)E53_SC2_Data.Accel[2]);
        if( X == 0 && Y == 0 && Z == 0)
```

```
{  
    X = (int)E53_SC2_Data.Accel[0];  
    Y = (int)E53_SC2_Data.Accel[1];  
    Z = (int)E53_SC2_Data.Accel[2];  
}  
else  
{  
    if(X+100<E53_SC2_Data.Accel[0]||X-  
100>E53_SC2_Data.Accel[0]||Y+100<E53_SC2_Data.Accel[1]||Y-  
100>E53_SC2_Data.Accel[1]||Z+100<E53_SC2_Data.Accel[2]||Z-  
100>E53_SC2_Data.Accel[2])  
    {  
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_8,GPIO_PIN_SET);  
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_9,GPIO_PIN_RESET);  
        Manhole_Cover.Status[0] = ' ';  
        Manhole_Cover.Status[1] = 'T';  
        Manhole_Cover.Status[2] = 'i';  
        Manhole_Cover.Status[3] = 'l';  
        Manhole_Cover.Status[4] = 't';  
    }  
    else  
{  
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_8,GPIO_PIN_RESET);  
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_9,GPIO_PIN_SET);  
        Manhole_Cover.Status[0] = 'L';  
        Manhole_Cover.Status[1] = 'e';  
        Manhole_Cover.Status[2] = 'v';  
        Manhole_Cover.Status[3] = 'e';  
        Manhole_Cover.Status[4] = 'l';  
    }  
}  
LCD_ShowString(10, 135, 200, 16, 16, "Temperature:");  
LCD_ShowNum(140, 135, (int)E53_SC2_Data.Temperature, 5, 16);  
LCD_ShowString(10, 160, 200, 16, 16, "Acce_X:");  
LCD_ShowNum(140, 160, (int)E53_SC2_Data.Accel[0], 5, 16);  
LCD_ShowString(10, 185, 200, 16, 16, "Acce_Y:");  
LCD_ShowNum(140, 185, (int)E53_SC2_Data.Accel[1], 5, 16);
```

```
LCD_ShowString(10, 210, 200, 16, 16, "Acce_Z:");
LCD_ShowNum(140, 210, (int)E53_SC2_Data.Accel[2], 5, 16);
osal_task_sleep(2*1000);
}

return 0;
}
```

```
115 //数据收集任务
116 static int app_collect_task_entry()
117 {
118     int X = 0,Y = 0,Z = 0;
119     Init_E53_SC2();
120     while ([1])
121     {
122         E53_SC2_Read_Data();
123         printf("\r\n*****Temperature*****\r\n");
124         printf("\r\n*****Accel[0]*****\r\n");
125         printf("\r\n*****Accel[1]*****\r\n");
126         printf("\r\n*****Accel[2]*****\r\n");
127         if(X == 0 && Y == 0 && Z == 0)
128         {
129             X = (int)E53_SC2_Data.Accel[0];
130             Y = (int)E53_SC2_Data.Accel[1];
131             Z = (int)E53_SC2_Data.Accel[2];
132         }
133         else
134         {
135             if(X+100>E53_SC2_Data.Accel[0]||X-100>E53_SC2_Data.Accel[0]||Y+100>E53_SC2_Data.Accel[1]||Y-100>E53_SC2_Data.Accel[1]||Z+100>E53_SC2_Data.Accel[2]||Z-100>E53_SC2_Data.Accel[2])
136             {
137                 HAL_GPIO_WritePin(GPIOB,GPIO_PIN_8,GPIO_PIN_SET);
138                 HAL_GPIO_WritePin(GPIOB,GPIO_PIN_9,GPIO_PIN_RESET);
139                 Manhole_Cover_Status[0] = 'L';
140                 Manhole_Cover_Status[1] = 'T';
141                 Manhole_Cover_Status[2] = 'I';
142                 Manhole_Cover_Status[3] = 'L';
143                 Manhole_Cover_Status[4] = 't';
144             }
145         else
146         {
147             HAL_GPIO_WritePin(GPIOB,GPIO_PIN_8,GPIO_PIN_RESET);
148             HAL_GPIO_WritePin(GPIOB,GPIO_PIN_9,GPIO_PIN_SET);
149             Manhole_Cover_Status[0] = 'L';
150             Manhole_Cover_Status[1] = 'e';
151             Manhole_Cover_Status[2] = 'v';
152             Manhole_Cover_Status[3] = 'e';
153             Manhole_Cover_Status[4] = 'l';
154         }
155     }
156     LCD_ShowString(10, 135, 200, 16, 16, "Temperature:");
157     LCD_ShowNum(140, 135, (int)E53_SC2_Data.Temperature, 5, 16);
158     LCD_ShowString(10, 160, 200, 16, 16, "Acce_X:");
159     LCD_ShowNum(140, 160, (int)E53_SC2_Data.Accel[0], 5, 16);
160     LCD_ShowString(10, 185, 200, 16, 16, "Acce_Y:");
161     LCD_ShowNum(140, 185, (int)E53_SC2_Data.Accel[1], 5, 16);
162     LCD_ShowString(10, 210, 200, 16, 16, "Acce_Z:");
163     LCD_ShowNum(140, 210, (int)E53_SC2_Data.Accel[2], 5, 16);
164     osal_task_sleep(2*1000);
165 }
166 return 0;
167 }
```

步骤 2 创建数据收集任务

```
osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
```

```
168
169 int standard_app_demo_main()
170 {
171     osal_semp_create(&s_rcv_sync,1,0);
172     //LCD屏幕显示
173
174     //创建任务
175     osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
176
177
178     return 0;
179 }
```

步骤 3 添加 LCD 屏幕显示代码

```
LCD_Clear(BLACK);
POINT_COLOR = GREEN;
LCD_ShowString(10, 10, 200, 16, 24, "Welcome to BearPi");
LCD_ShowString(30, 40, 200, 16, 24, "Manhole Demo");
LCD_ShowString(10, 80, 200, 16, 16, "NCDP_IP:");
LCD_ShowString(80, 80, 200, 16, 16, cn_app_server);
LCD_ShowString(10, 110, 200, 16, 16, "NCDP_PORT:");
LCD_ShowString(100, 110, 200, 16, 16, cn_app_port);
```

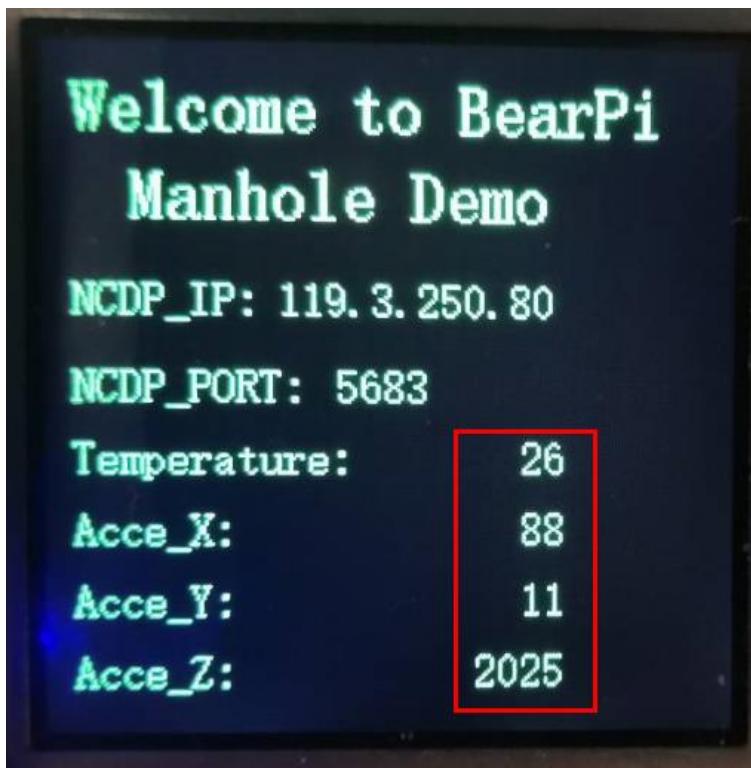
```
169 int standard_app_demo_main()
170 {
171     osal_semp_create(&s_rcv_sync,1,0);
172     //LCD屏幕显示
173
174     LCD_Clear(BLACK);
175     POINT_COLOR = GREEN;
176     LCD_ShowString(10, 10, 200, 16, 24, "Welcome to BearPi");
177     LCD_ShowString(30, 40, 200, 16, 24, "Manhole Demo");
178     LCD_ShowString(10, 80, 200, 16, 16, "NCDP_IP:");
179     LCD_ShowString(80, 80, 200, 16, 16, cn_app_server);
180     LCD_ShowString(10, 110, 200, 16, 16, "NCDP_PORT:");
181     LCD_ShowString(100, 110, 200, 16, 16, cn_app_port);
182     //创建任务
183     osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
184
185     return 0;
186 }
```

步骤 4 重新编译烧录，查看结果

点击  进行重新编译，烧录程序，在串口终端中可以看到有实时的数据打印；

```
735  
736 *****Accel[0] is 85  
737  
738 *****Accel[1] is 10  
739  
740 *****Accel[2] is 2026  
741
```

在 LCD 屏幕上，也可以看到实时的数据显示；



7.2.4 创建数据上报任务

步骤 1 添加数据上报任务

```
static int app_report_task_entry()  
{  
    int ret = -1;  
    oc_config_param_t      oc_param;  
    memset(&oc_param,0,sizeof(oc_param));  
    oc_param.app_server.address = cn_app_server;  
    oc_param.app_server.port = cn_app_port;  
    oc_param.app_server.ep_id = cn_endpoint_id;
```

```
oc_param.boot_mode = en_oc_boot_strap_mode_factory;
oc_param.rcv_func = app_msg_deal;

ret = oc_lwm2m_config( &oc_param);
if (0 != ret)
{
    return ret;
}

//install a dealer for the led message received
while(1) //--TODO ,you could add your own code here
{
    Manhole_Cover.messageId = cn_app_Manhole_Cover;
    Manhole_Cover.Temperature = (int)E53_SC2_Data.Temperature;
    Manhole_Cover.Accel_x = htons(E53_SC2_Data.Accel[0] & 0x0000FFFF);
    Manhole_Cover.Accel_y = htons(E53_SC2_Data.Accel[1] & 0x0000FFFF);
    Manhole_Cover.Accel_z = htons(E53_SC2_Data.Accel[2] & 0x0000FFFF);
    oc_lwm2m_report((char *)&Manhole_Cover, sizeof(Manhole_Cover), 1000);
    osal_task_sleep(2*1000);
}
return ret;
}
```

```
113 //数据上报任务
114 static int app_report_task_entry()
115 {
116     int ret = -1;
117     oc_config_param_t      oc_param;
118     memset(&oc_param, 0, sizeof(oc_param));
119     oc_param.app_server.address = cn_app_server;
120     oc_param.app_server.port = cn_app_port;
121     oc_param.app_server.ep_id = cn_endpoint_id;
122     oc_param.boot_mode = en_oc_boot_strap_mode_factory;
123     oc_param.recv_func = app_msg_deal;
124
125     ret = oc_lwm2m_config( &oc_param);
126     if [0 != ret]
127     {
128         return ret;
129     }
130     //install a dealer for the led message received
131     while(1) //--TODO ,you could add your own code here
132     {
133         Manhole_Cover.messageId = cn_app_Manhole_Cover;
134         Manhole_Cover.Temperature = (int)E53_SC2_Data.Temperature;
135         Manhole_Cover.Accel_x = htons(E53_SC2_Data.Accel[0] & 0x0000FFFF);
136         Manhole_Cover.Accel_y = htons(E53_SC2_Data.Accel[1] & 0x0000FFFF);
137         Manhole_Cover.Accel_z = htons(E53_SC2_Data.Accel[2] & 0x0000FFFF);
138         oc_lwm2m_report((char *)&Manhole_Cover, sizeof(Manhole_Cover), 1000);
139         osal_task_sleep(2*1000);
140     }
141     return ret;
142 }
143 //数据收集任务
```

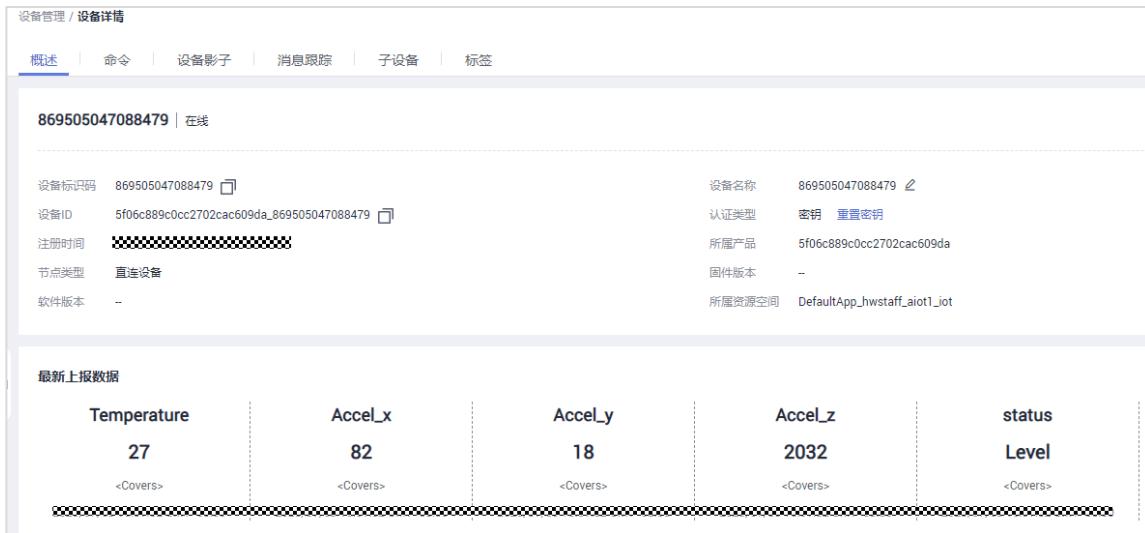
步骤 2 创建数据上报任务

```
osal_task_create("app_report",app_report_task_entry,NULL,0x1000,NULL,2);
```

```
197 int standard_app_demo_main()
198 {
199
200     osal_semp_create(&s_rcv_sync,1,0);
201     //LCD屏幕显示
202     LCD_Clear(BLACK);
203     POINT_COLOR = GREEN;
204     LCD_ShowString(10, 10, 200, 16, 24, "Welcome to BearPi");
205     LCD_ShowString(30, 40, 200, 16, 24, "Manhole Demo");
206     LCD_ShowString(10, 80, 200, 16, 16, "NCDP_IP:");
207     LCD_ShowString(80, 80, 200, 16, 16, cn_app_server);
208     LCD_ShowString(10, 110, 200, 16, 16, "NCDP_PORT:");
209     LCD_ShowString(100, 110, 200, 16, 16, cn_app_port);
210     //创建任务
211     osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
212     osal_task_create("app_report",app_report_task_entry,NULL,0x1000,NULL,2);
213
214 }
```

步骤 3 编译烧录，查看结果

编译烧录程序，在物联网平台中，可以看到实时的数据；



The screenshot shows the 'Device Management / Device Details' page for a device with ID 869505047088479. The device is online. Key details include:

| | | | |
|-------|--|--------|------------------------------|
| 设备标识码 | 869505047088479 | 设备名称 | 869505047088479 |
| 设备ID | 5f06c889c0cc2702cac609da_869505047088479 | 认证类型 | 密钥 重置密钥 |
| 注册时间 | [REDACTED] | 所属产品 | 5f06c889c0cc2702cac609da |
| 节点类型 | 直连设备 | 固件版本 | - |
| 软件版本 | -- | 所属资源空间 | DefaultApp_hwstaff_aiot1_iot |

Below this, the 'Latest Reported Data' section displays real-time sensor values:

| Temperature | Accel_x | Accel_y | Accel_z | status |
|-------------|----------|----------|----------|----------|
| 27 | 82 | 18 | 2032 | Level |
| <Covers> | <Covers> | <Covers> | <Covers> | <Covers> |

7.3 练习题

7.3.1 使用 WIFI 通信方式实现智慧井盖案例

8

基于 NB-IoT 和 WIFI 的人体感应实验

8.1 实验介绍

8.1.1 关于本实验

本实验基于 NB-IoT 和 WIFI 实现人体感应案例，实现实时数据的采集，实现端云互通。

8.1.2 实验目的

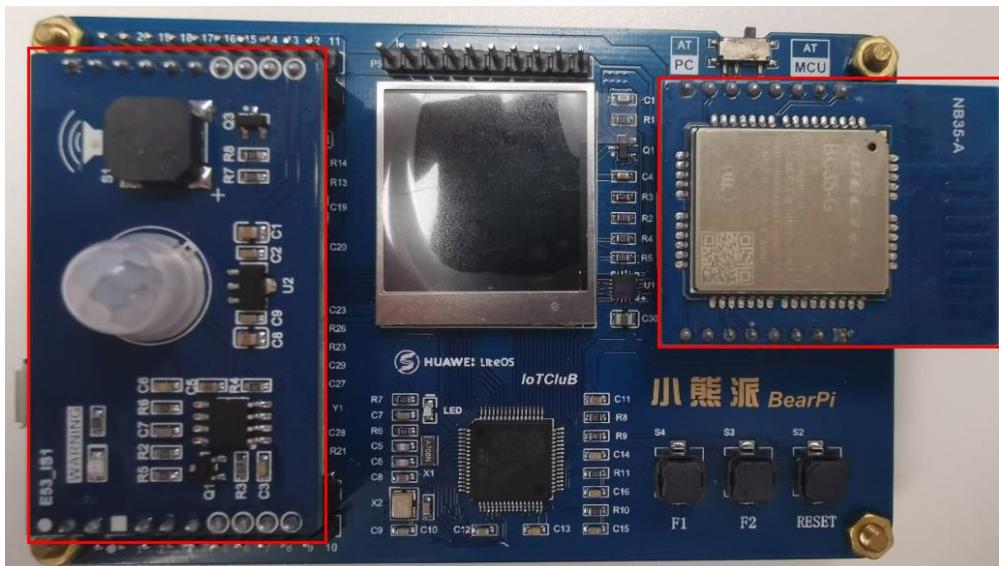
- 掌握 NB-IoT 通信方式的配置
- 掌握 WIFI 通信方式的配置
- 掌握人体感应案例的开发过程

8.2 实验任务配置步骤

8.2.1 配置人体感应案例

步骤 1 安装人体感应扩展板 E53_IS1

将人体感应扩展板 E53_IS1、NB 通信扩展板 NB35-A 按照正确的方向插入小熊派开发板；

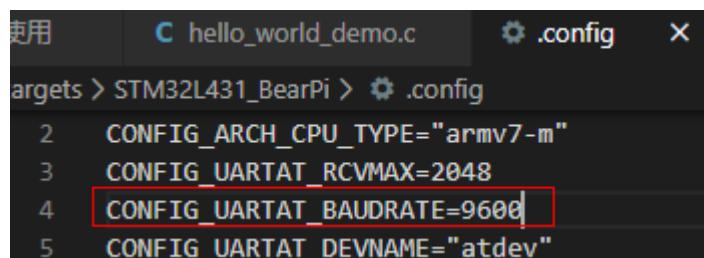


步骤 2 修改.config 文件

打开 targets->STM32L431_BearPi->.config 文件；
修改“CONFIG_USER_DEMO”为“oc_infrared_template”；
“Ctrl+S”保存.config 文件；

```
42 # Demo
43 CONFIG_USER_DEMO="oc_infrared_template"
44 # NB-IoT
```

修改“CONFIG_UARTAT_BAUDRATE”波特率为“9600”；



```
使用 C hello_world_demo.c .config ×
targets > STM32L431_BearPi > .config
2 CONFIG_ARCH_CPU_TYPE="armv7-m"
3 CONFIG_UARTAT_RCVMAX=2048
4 CONFIG_UARTAT_BAUDRATE=9600
5 CONFIG_UARTAT_DEVNAME="atdev"
```

将“CONFIG_BOUDICA150_ENABLE=y”前面#号去除，将 WIFI 配置前加上#号；

```
44 # NB-IoT
45 CONFIG_BOUDICA150_ENABLE=y
46 # WIFI
47 #CONFIG_OCLWM2MTINY_ENABLE=y
48 #CONFIG_LWM2M_AL_ENABLE=y
49 #CONFIG_WAKAAMALWM2M_ENABLE=y
50 #CONFIG_TCIP_AL_ENABLE=y
51 #CONFIG_ESP8266_ENABLE=y
52 #CONFIG_ESP8266_SSID="Huawei"
53 #CONFIG_ESP8266_PWD="12345678"
```

步骤 3 修改 iot_config.h 文件

打开 targets->STM32L431_BearPi->iot_config.h 文件；
修改“CONFIG_USER_DEMO”为“oc_infrared_template”；
将“CONFIG_BOUDICA150_ENABLE=y”注释去除，将 WIFI 配置的注释掉；

```
22  /*Demo*/
23  #define CONFIG_USER DEMO "oc_manhole_cover_template"
24  /*NB-IoT*/
25  #define CONFIG_BOUDICA150_ENABLE 1
26  /*WIFI*/
27  // #define CONFIG_OCLWM2MTINY_ENABLE 1
28  // #define CONFIG_LWM2M_AL_ENABLE 1
29  // #define CONFIG_WAKAAMALWM2M_ENABLE 1
30  // #define CONFIG_TCIP_AL_ENABLE 1
31  // #define CONFIG_ESP8266_ENABLE 1
32  // #define CONFIG_ESP8266_SSID "Huawei"
33  // #define CONFIG_ESP8266_PWD "12345678"
```

修改“CONFIG_UARTAT_BAUDRATE”波特率为“9600”；

“Ctrl+S”保存 iot_config.h 文件；

```
1  /* Generated by IoT Link Studio*/
2  #define CONFIG_ARCH_CPU_TYPE "armv7-m"
3  #define CONFIG_UARTAT_RCVMAX 2048
4  #define CONFIG_UARTAT_BAUDRATE 9600
5  #define CONFIG_UARTAT_DEVNAME "atdev"
6  #define CONFIG_LITEOS_ENABLE 1
7  #define CONFIG_AT_ENABLE 1
```

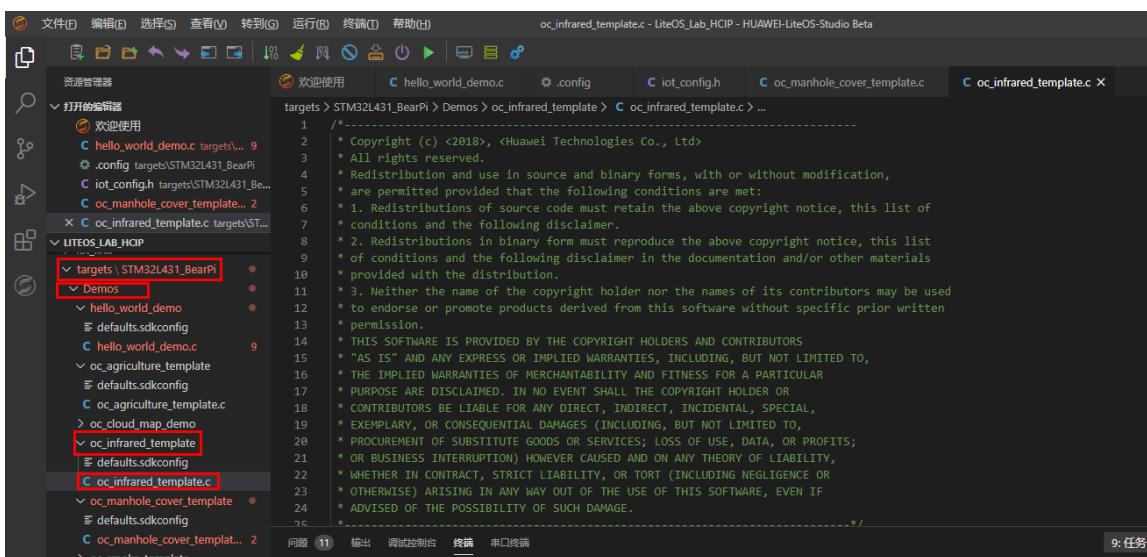
8.2.2 创建人体感应所需的数据结构

步骤 1 打开 oc_infrared_template.c 文件

打开案例程序文件 targets->STM32L431_BearPi->Demos->

oc_infrared_template -> oc_infrared_template.c；

人体感应案例的代码都在 oc_infrared_template.c 中修改；



步骤 2 添加人体感应案例扩展板驱动头文件

```
#include "E53_IS1.h"

48 //驱动头文件
49 #include "E53_IS1.h"
```

步骤 3 添加平台所需的基本变量

```
#define cn_endpoint_id      "BearPi_0001"
#define cn_app_server        "119.3.250.80"
#define cn_app_port          "5683"

50  typedef unsigned char int8u;
51  typedef char int8s;
52  typedef unsigned short int16u;
53  typedef short int16s;
54  typedef unsigned char int24u;
55  typedef char int24s;
56  typedef int int32s;
57  typedef char string;
58  typedef char array;
59  typedef char varstring;
60  typedef char variant;
61
62  //连接平台基本变量
63  #define cn_endpoint_id      "BearPi_0001"
64  #define cn_app_server        "119.3.250.80"
65  #define cn_app_port          "5683"
```

步骤 4 添加 MessageID

```
#define cn_app_infrared 0xc  
66 //MessageID  
67 #define cn_app_infrared 0xc
```

步骤 5 添加消息结构体

```
#pragma pack(1)  
typedef struct  
{  
    int8u messageId;  
    string Status[4];  
} tag_app_infrared;  
#pragma pack()
```

```
68 //消息结构体  
69 #pragma pack(1)  
70 typedef struct  
71 {  
    int8u messageId;  
    string Status[4];  
} tag_app_infrared;  
75 #pragma pack()  
76 //存储数据的变量
```

步骤 6 添加存储数据的变量

```
tag_app_infrared infrared;  
76 //存储数据的变量  
77 tag_app_infrared infrared;
```

8.2.3 创建数据收集任务

步骤 1 添加数据收集任务

```
static int app_collect_task_entry()
{
    Init_E53_IS1();
    while (1)
    {
        if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_9)==SET)
        {
            // E53_IS1_Beep_StatusSet(ON);
            infrared.Status[0] = 'H';
            infrared.Status[1] = 'a';
            infrared.Status[2] = 'v';
            infrared.Status[3] = 'e';
        }
        else
        {
            // E53_IS1_Beep_StatusSet(OFF);
            infrared.Status[0] = ' ';
            infrared.Status[1] = ' ';
            infrared.Status[2] = 'N';
            infrared.Status[3] = 'O';
        }
        LCD_ShowString(10, 160, 200, 16, 16, infrared.Status);
        osal_task_sleep(2*1000);
    }
    return 0;
}
```

```
110 //数据收集任务
111 static int app_collect_task_entry()
112 {
113     Init_E53_IS1();
114     while (1)
115     {
116         if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_9)==SET)
117         {
118             // E53_IS1_Beep_StatusSet(ON);
119             infrared.Status[0] = 'H';
120             infrared.Status[1] = 'a';
121             infrared.Status[2] = 'v';
122             infrared.Status[3] = 'e';
123         }
124         else
125         {
126             // E53_IS1_Beep_StatusSet(OFF);
127             infrared.Status[0] = ' ';
128             infrared.Status[1] = ' ';
129             infrared.Status[2] = 'N';
130             infrared.Status[3] = 'O';
131         }
132         LCD_ShowString(10, 160, 200, 16, 16, infrared.Status);
133         osal_task_sleep(2*1000);
134     }
135     return 0;
136 }
```

步骤 2 创建数据收集任务

```
osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
```

```
137 int standard_app_demo_main()
138 {
139     osal_semp_create(&s_rcv_sync,1,0);
140     //LCD屏幕显示
141
142     //创建任务
143     osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
144
145 }
```

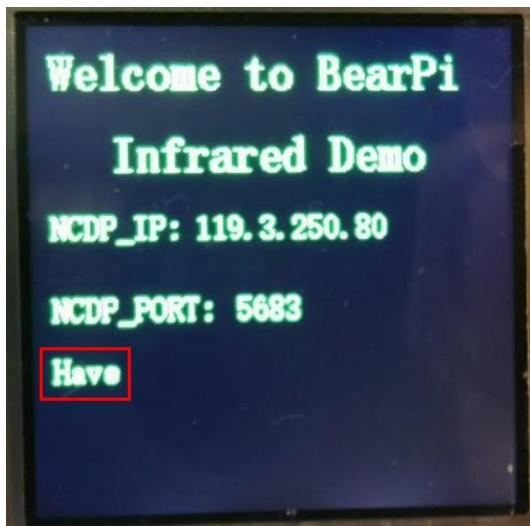
步骤 3 添加 LCD 屏幕显示代码

```
LCD_Clear(BLACK);
POINT_COLOR = GREEN;
LCD_ShowString(10, 10, 200, 16, 24, "Welcome to BearPi");
LCD_ShowString(40, 50, 200, 16, 24, "Infrared Demo");
LCD_ShowString(10, 90, 200, 16, 16, "NCDP_IP:");
LCD_ShowString(80, 90, 200, 16, 16, cn_app_server);
LCD_ShowString(10, 130, 200, 16, 16, "NCDP_PORT:");
LCD_ShowString(100, 130, 200, 16, 16, cn_app_port);
```

```
137 int standard_app_demo_main()
138 {
139     osal_semp_create(&s_rcv_sync,1,0);
//LCD屏幕显示
140
141     LCD_Clear(BLACK);
142     POINT_COLOR = GREEN;
143     LCD_ShowString(10, 10, 200, 16, 24, "Welcome to BearPi");
144     LCD_ShowString(40, 50, 200, 16, 24, "Infrared Demo");
145     LCD_ShowString(10, 90, 200, 16, 16, "NCDP_IP:");
146     LCD_ShowString(80, 90, 200, 16, 16, cn_app_server);
147     LCD_ShowString(10, 130, 200, 16, 16, "NCDP_PORT:");
148     LCD_ShowString(100, 130, 200, 16, 16, cn_app_port);
//创建任务
149     osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
150
151     return 0;
152 }
```

步骤 4 重新编译烧录，查看结果

重新编译烧录程序，将人体在感应器前进出，在 LCD 屏幕上，也可以看到实时的人体感应结果显示；



8.2.4 创建数据上报任务

步骤 1 添加数据上报任务

```
static int app_report_task_entry()
{
    int ret = -1;
    oc_config_param_t      oc_param;
    memset(&oc_param,0,sizeof(oc_param));
    oc_param.app_server.address = cn_app_server;
    oc_param.app_server.port = cn_app_port;
    oc_param.app_server.ep_id = cn_endpoint_id;
    oc_param.boot_mode = en_oc_boot_strap_mode_factory;
    oc_param.rcv_func = app_msg_deal;

    ret = oc_lwm2m_config( &oc_param);
    if (0 != ret)
    {
        return ret;
    }
    //install a dealer for the led message received
    while(1) //--TODO ,you could add your own code here
    {
        infrared.messageId = cn_app_infrared;
        oc_lwm2m_report((char *)&infrared, sizeof(infrared), 1000);
        osal_task_sleep(2*1000);
    }
    return 0;
}
```

```
108 //数据上报任务
109 static int app_report_task_entry()
110 {
111     int ret = -1;
112     oc_config_param_t      oc_param;
113     memset(&oc_param,0,sizeof(oc_param));
114     oc_param.app_server.address = cn_app_server;
115     oc_param.app_server.port = cn_app_port;
116     oc_param.app_server.ep_id = cn_endpoint_id;
117     oc_param.boot_mode = en_oc_boot_strap_mode_factory;
118     oc_param.recv_func = app_msg_deal;
119
120     ret = oc_lwm2m_config( &oc_param);
121     if (0 != ret)
122     {
123         return ret;
124     }
125     //install a dealer for the led message received
126     while(1) //--TODO ,you could add your own code here
127     {
128         infrared.messageId = cn_app_infrared;
129         oc_lwm2m_report((char *)&infrared, sizeof(infrared), 1000);
130         osal_task_sleep(2*1000);
131     }
132     return 0;
133 }
134 //数据收集任务
```

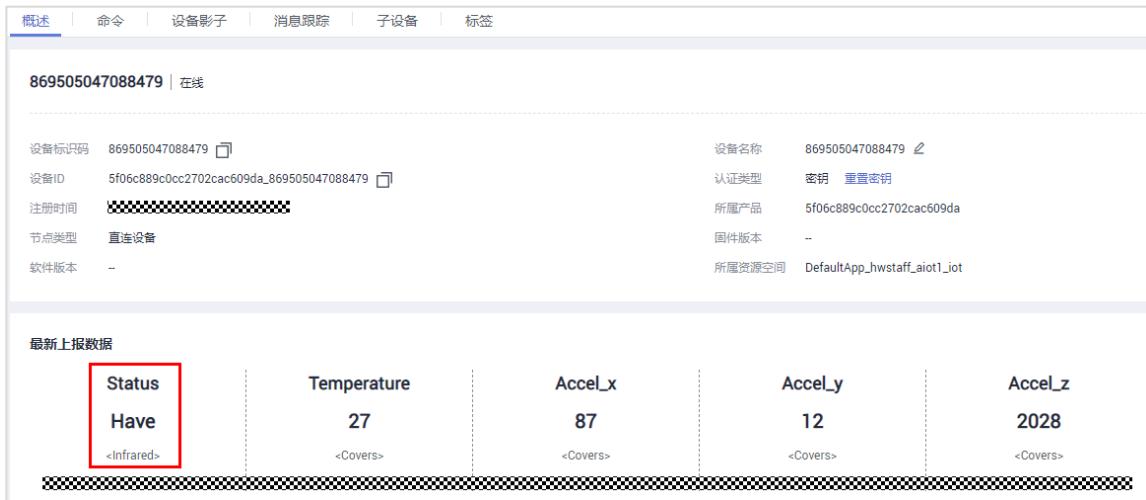
步骤 2 创建数据上报任务

```
osal_task_create("app_report",app_report_task_entry,NULL,0x1000,NULL,2);
```

```
162 int standard_app_demo_main()
163 {
164     osal_semp_create(&s_rcv_sync,1,0);
165     //LCD屏幕显示
166     LCD_Clear(BLACK);
167     POINT_COLOR = GREEN;
168     LCD_ShowString(10, 10, 200, 16, 24, "Welcome to BearPi");
169     LCD_ShowString(40, 50, 200, 16, 24, "Infrared Demo");
170     LCD_ShowString(10, 90, 200, 16, 16, "NCDP_IP:");
171     LCD_ShowString(80, 90, 200, 16, 16, cn_app_server);
172     LCD_ShowString(10, 130, 200, 16, 16, "NCDP_PORT:");
173     LCD_ShowString(100, 130, 200, 16, 16, cn_app_port);
174     //创建任务
175     osal_task_create("app_collect",app_collect_task_entry,NULL,0x400,NULL,3);
176     osal_task_create("app_report",app_report_task_entry,NULL,0x1000,NULL,2);
177     return 0;
178 }
```

步骤 3 编译烧录，查看结果

编译烧录程序，在物联网平台中，可以看到实时的结果；



The screenshot shows the device details page for a device with ID 869505047088479. The page includes tabs for Overview, Commands, Device Shadow, Message Tracking, Child Devices, and Tags. The Overview tab is selected. The device information section shows the device ID, name, authentication type (Key), product ID, and resource space. The latest reported data section displays five metrics: Status (Have <Infrared>), Temperature (27), Accel_x (87), Accel_y (12), and Accel_z (2028). The 'Status' row is highlighted with a red box.

| 设备标识码 | 869505047088479 | 设备名称 | 869505047088479 |
|-------|--|--------|------------------------------|
| 设备ID | 5f06c889c0cc2702cac609da_869505047088479 | 认证类型 | 密钥 重置密钥 |
| 注册时间 | ████████████████ | 所属产品 | 5f06c889c0cc2702cac609da |
| 节点类型 | 直连设备 | 固件版本 | -- |
| 软件版本 | -- | 所属资源空间 | DefaultApp_hwstaff_aiot1_iot |

| 最新上报数据 | | | | | |
|------------------------------|-------------------------------|---------------------------|---------------------------|-----------------------------|--|
| Status Have <Infrared> | Temperature 27 <Covers> | Accel_x 87 <Covers> | Accel_y 12 <Covers> | Accel_z 2028 <Covers> | |

8.3 练习题

8.3.1 使用 WIFI 通信方式实现人体感应案例

9

综合训练 1

9.1 实验介绍

9.1.1 关于本实验

本实验分为自主训练，根据前面六个实验所用到的端云互通方法，实现基于智慧物流完整案例的运行，实现基于智慧路灯空白案例的 NB-IoT 端云互通。

9.1.2 实验目的

- 掌握如何利用物联网平台和 Huawei LiteOS 物联网操作系统实现不同案例的端云互通

9.2 实验任务配置步骤

9.2.1 基于 NB-IoT 和 WIFI 的智慧物流实验

步骤 1 配置智慧物流案例

智慧物流案例：targets->STM32L431_BearPi->Demos->oc_track_template->
oc_track_template.c

步骤 2 使用 NB-IoT 通信方式编译烧录运行智慧物流案例

步骤 3 使用 WIFI 通信方式编译烧录运行智慧物流案例

9.2.2 基于 NB-IoT 实现智慧路灯案例

步骤 1 在物联网平台创建智慧路灯的功能定义和编解码插件

步骤 2 配置智慧路灯案例

智慧路灯案例：targets->STM32L431_BearPi->Demos->oc_streetlight_template->
oc_streetlight_template.c

步骤 3 创建智慧路灯所需的数据结构

步骤 4 创建数据收集任务

步骤 5 创建数据上报任务

步骤 6 创建命令响应任务

9.2.3 使用按键选择案例运行

步骤 1 在任意案例.c 文件中添加其他案例的数据采集和上报任务

步骤 2 创建 GPIO 或 EXIT 按键检测任务

步骤 3 在 LCD 屏幕显示 6 个案例任务名称

步骤 4 在按键检测任务中创建删除案例的数据采集和上报任务

9.3 结果验证

- 基于 NB-IoT 采集智慧物流扩展板的经纬度信息
- 基于 NB-IoT 采集智慧路灯扩展板的光敏传感器数据和控制开关灯
- 使用按键选择案例运行



The screenshot shows the 'Device Management / Device Details' page. At the top, there are tabs: 概述 (Overview), 命令 (Commands), 设备影子 (Device Shadow), 消息跟踪 (Message Tracking), 子设备 (Sub-device), and 标签 (Tags). The '概述' tab is selected.

Device ID: 20200629T093424ZNBSimulator (Online)

Device Details:

- 设备标识码: 1598439177178
- 设备ID: 5ef9b5a069c46102cb120886_1598439177178
- 注册时间: 2020/08/26 18:52:57 GMT+08:00
- 节点类型: 直连设备
- 软件版本: -

Latest Reported Data:

| Longitude | Latitude |
|-----------|----------|
| 132.12345 | 32.12345 |

Timestamps: 2020/09/03 10:12:10 GMT+08:00 (for both coordinates)

10 物联网平台接口调用实验

10.1 实验介绍

10.1.1 关于本实验

本实验使用华为云 API Explorer 调用物联网平台的接口，熟悉物联网平台二次开发。

10.1.2 实验目的

- 掌握 API Explorer 的使用
- 掌握物联网平台的创建、删除设备接口
- 掌握物联网平台的查询设备影子接口

10.2 实验任务配置步骤

10.2.1 登录华为云 API Explorer 服务

步骤 1 登录华为云 API Explorer 服务

登录华为云官网，选择“开发者”->“API Explorer”；



The screenshot shows the Huawei Cloud developer portal. The top navigation bar includes links for About, Latest Activities, Products, Solutions, EI Intelligence, Pricing, Cloud Market, Partners, Developers (which is highlighted), and Support & Services. Below the navigation is a search bar and a language dropdown set to Chinese. The main content area is titled "面向开发者的一站式导航" (One-stop navigation for developers) and "帮助开发者快速到达资源获取、工具获取与实践参与平台, 形成开发者学习交流、分享互助的主阵地" (Help developers quickly reach the platform for resource acquisition, tool acquisition, and practical participation, forming the main阵地 for developer learning exchange and sharing). A red box highlights the "API Explorer" link under the "Resource Tools" section. Other sections include Cloud Community, Cloud Academy, Soil Improvement Plan, and various developer resources like forums, video, and MVP programs.

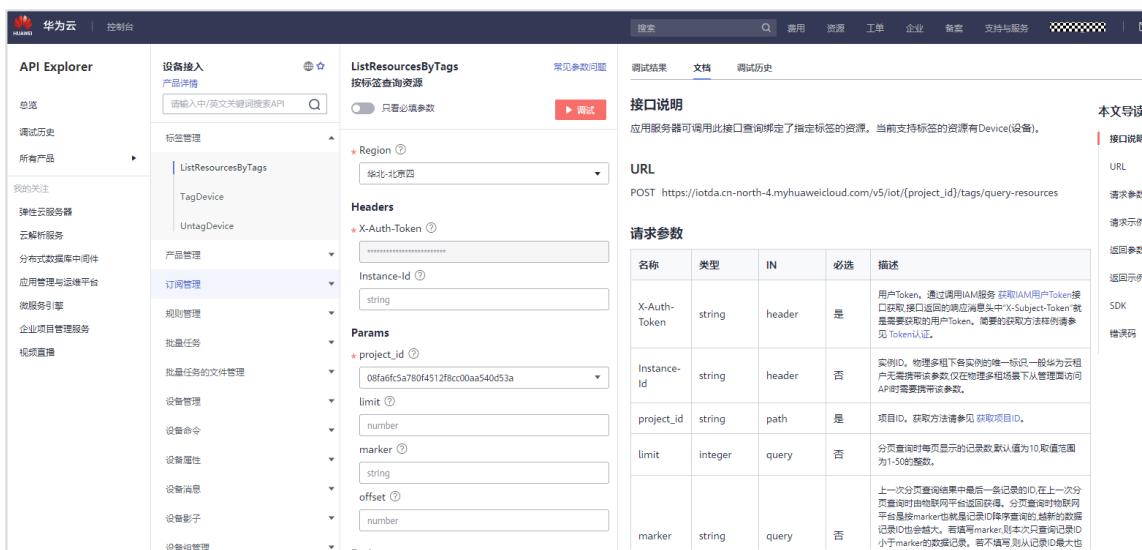
步骤 2 进入设备接入服务接口

在搜索框中输入“设备接入”，点击“设备接入”；



The screenshot shows the API Explorer interface. On the left sidebar, there are tabs for Overview, Test History, All Products, and My Favorites. The main search bar has "设备接入" (Device Access) typed into it. Below the search bar, there are sections for "最近调用的API" (Recently called APIs) and "物联网" (Internet of Things). A red box highlights the "设备接入" button in the search results area.

进入到设备接入接口调用界面；



The screenshot shows the API Explorer interface for the Device Access service. The left sidebar lists categories like Overview, Test History, All Products, and My Favorites. The main area shows the "ListResourcesByTags" endpoint for the "设备接入" product. It includes fields for Region (Region 4 - Beijing 4), Headers (X-Auth-Token), Params (project_id, limit, marker, offset), and a description of the API. To the right, there are tabs for Test Result, Documentation, and Test History. A detailed description of the API parameters is provided, including examples and notes about markers and offsets. A red box highlights the "ListResourcesByTags" endpoint in the main list.

10.2.2 调用删除设备接口

步骤 1 查看接口调用必填的参数

点击“设备管理”->“DeleteDevice”，从接口中可以看出，需要 device_id；

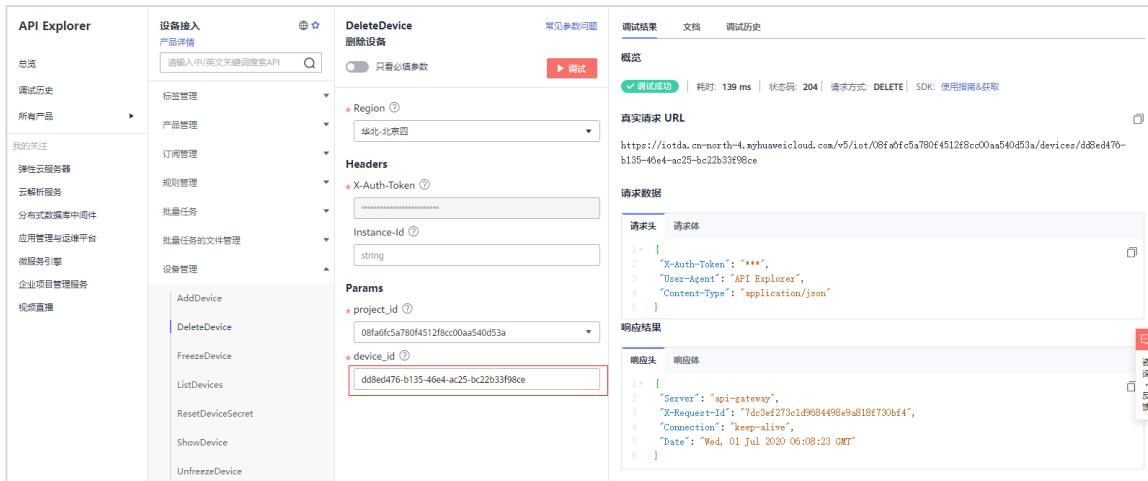
The screenshot shows the API Explorer interface for the DeleteDevice endpoint. In the 'Headers' section, 'X-Auth-Token' and 'Instance-Id' are listed. In the 'Params' section, 'project_id' and 'device_id' are required parameters. The 'device_id' field is highlighted with a red border.

步骤 2 从物联网平台中找到对应的参数

device_id 查看位置：“物联网平台”->“设备”->“所有设备”->“设备列表”->“查看”；

The screenshot shows the IoT Platform's device management interface. It displays a device named 'HCIA' with the status '在线'. Key information shown includes the device identifier 'dd8ed476-b135-46e4-ac25-bc22b33f98ce', which is highlighted with a red box. Other details like registration time, node type, and software version are also visible.

步骤 3 将参数复制到接口中，点击“调试”



The screenshot shows the API Explorer interface for the DeleteDevice endpoint. The 'Headers' section includes 'X-Auth-Token' and 'Instance-Id'. The 'Params' section includes 'project_id' and 'device_id'. The 'Request Body' shows a JSON object with 'X-Auth-Token', 'User-Agent', and 'Content-Type'. The 'Response' section shows a successful response with headers like 'Server', 'X-Request-ID', 'Connection', and 'Date'.

步骤 4 查看结果

在物联网平台设备列表中可以看到设备已经删除；

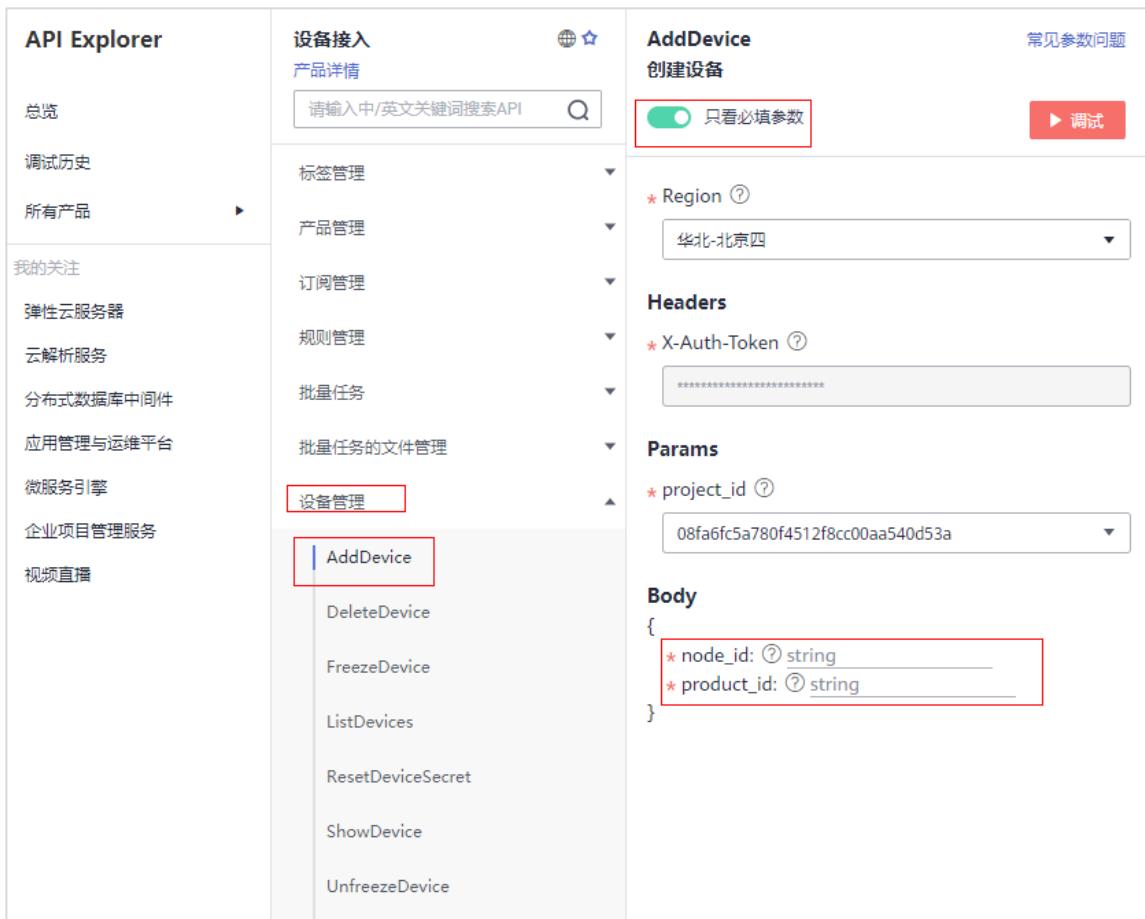


The screenshot shows the IoT Platform Device List page. It lists a single device with the following details: 离线 (Status), 83b9924ff736d40bbNBsimul... (Device Name), 1593487380524 (Device Identifier), DefaultApp_hwstaff_aiot1_iot (Resource Space), HCIA-IoT (Product), 直连设备 (Node Type). There are '查看' (View), '删除' (Delete), and '冻结' (Freeze) buttons in the '操作' (Operation) column.

10.2.3 调用创建设备接口

步骤 1 查看接口调用必填的参数

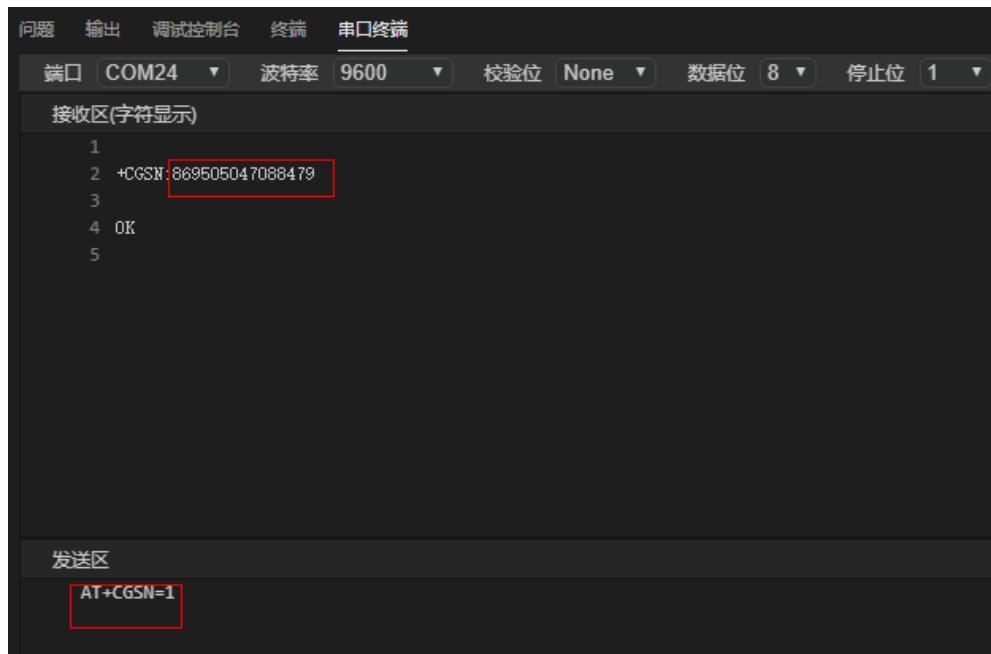
点击“设备管理”->“AddDevice”，点击“只看必填参数”，从接口中可以看出，需要 node_id，product_id；



The screenshot shows the API Explorer interface for the HCIP-IoT Developer experiment. On the left, there's a sidebar with various service links. The main area is focused on the '设备接入' (Device Access) section. Under '产品详情', there's a search bar and a toggle switch labeled '只看必填参数' (Show required parameters only), which is highlighted with a red box. Below it is a dropdown for 'Region' set to '华北-北京四'. The 'Headers' section contains an 'X-Auth-Token' field with placeholder text. The 'Params' section has a 'project_id' field set to '08fa6fc5a780f4512f8cc00aa540d53a'. The 'Body' section shows a JSON object with two required parameters: 'node_id' and 'product_id', both highlighted with red boxes. The 'AddDevice' button at the bottom of the list is also highlighted with a red box.

步骤 2 从物联网平台中找到对应的参数

node_id: 从 LiteOS Studio 串口终端中发送 AT 指令“AT+CGMN=1”获取 IMEI 号;



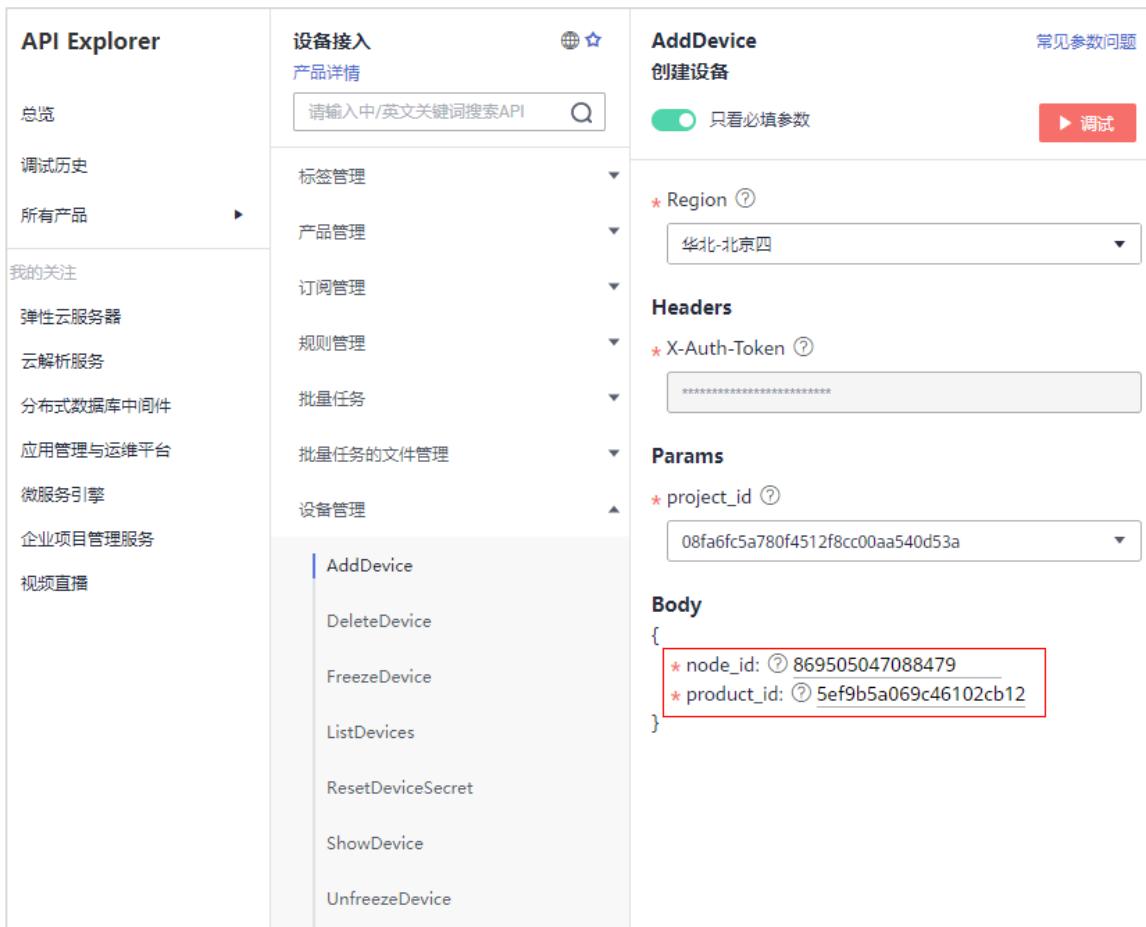
product_id 查看位置：“物联网平台”->“产品”，可以看到对应产品的 product ID；



The screenshot shows the "Products" section of the IoT Platform. On the left sidebar, under "Products", there is a table with one row. The table columns are: Product Name, Product ID, Device Type, Protocol Type, and Operation. The data is as follows:

| 产品名称 | 产品ID | 设备类型 | 协议类型 | 操作 |
|-----------|--------------------------|------|------------|---|
| HCI-A-IoT | 5ef9b5a069c46102cb120886 | IoT | LwM2M/CoAP | 详情 删除 |

步骤 3 将参数复制到接口中，点击“调试”



The screenshot shows the API Explorer interface for the "AddDevice" endpoint. The left sidebar lists various services, and the main area shows the "AddDevice" configuration.

Headers:

- * X-Auth-Token: (Redacted)

Params:

- * project_id: 08fa6fc5a780f4512f8cc00aa540d53a

Body:

```
{  
    * node_id: ⑦ 869505047088479  
    * product_id: ⑦ 5ef9b5a069c46102cb12  
}
```

步骤 4 调试接口查看结果

在物联网平台设备列表中可以看到添加的设备；

| 设备列表 | | | | | | |
|------|----------------------------|-----------------|------------------------------|----------|------|--|
| 状态 | 设备名称 | 设备标识码 | 所属资源空间 | 所属产品 | 节点类型 | 操作 |
| 未激活 | | 869505047088479 | DefaultApp_hwstaff_aiot1_iot | HCIA-IoT | 直连设备 | 查看 删除 冻结 |
| 离线 | 83b9924f736d40bbNBSimul... | 1593487380524 | DefaultApp_hwstaff_aiot1_iot | HCIA-IoT | 直连设备 | 查看 删除 冻结 |
| 10 | 总条数: 2 < 1 > | | | | | |

步骤 5 上报数据

发送 AT 指令“AT+NMGS=5,00193C0064”上报数据；



登录物联网平台，可以看到设备已在线；

| 设备列表 | | | | | | |
|------|----------------------------|-----------------|------------------------------|----------|------|--|
| 状态 | 设备名称 | 设备标识码 | 所属资源空间 | 所属产品 | 节点类型 | 操作 |
| 在线 | | 869505047088479 | DefaultApp_hwstaff_aiot1_iot | HCIA-IoT | 直连设备 | 查看 删除 冻结 |
| 离线 | 83b9924f736d40bbNBSimul... | 1593487380524 | DefaultApp_hwstaff_aiot1_iot | HCIA-IoT | 直连设备 | 查看 删除 冻结 |
| 10 | 总条数: 2 < 1 > | | | | | |

10.2.4 调用查询设备影子接口

步骤 1 查看接口调用必填的参数

点击“设备影子”->“ShowDeviceShadow”，从接口中可以看出，需要 device_id；

步骤 2 从物联网平台中找到对应参数

device_id 查看位置：“物联网平台”->“设备”->“所有设备”->“设备列表”->“查看”；

步骤 3 将参数复制到接口中，点击“调试”

在响应结果的响应体中，可以看到上报的数据；

```

1 * [
2   "X-Auth-Token": "***",
3   "User-Agent": "API Explorer",
4   "Content-Type": "application/json"
5 ]
    
```

```

1 * {
2   "service_id": "agriculture",
3   "desired": [
4     "properties": null,
5     "event_time": null
6   ],
7   "reported": [
8     "properties": [
9       "Temperature": 25,
10      "Humidity": 60,
11      "Luminance": 100
12    ],
13    "event_time": "20200701T063830Z"
14  ],
15  "version": 1
16 }
17
18 ]
19
20
21
    
```

10.3 练习题

10.3.1 调用接口创建带有设备名称的设备

10.3.2 上报智慧烟感、物流数据，查看响应结果

11

物联网平台设备联动实验

11.1 实验介绍

11.1.1 关于本实验

本实验使用物联网平台的设备联动功能，实现指定规则下设备自动下发指定命令。

11.1.2 实验目的

- 掌握物联网平台规则的使用

11.2 实验任务配置步骤

11.2.1 实现根据光照强度自动控制 LED 灯开

步骤 1 创建规则

登陆华为云物联网平台，点击“规则”》“创建规则”；



点击设备联动下方的“定义规则”；



自定义输入规则名称；



基本信息

所属资源空间: DefaultApp_hwstaff_aiot1_iot

* 规则名称: LED_ON 立即触发

生效时间: 一直生效

描述: 可选填 0/256

点击“添加条件”；



触发条件

需满足 全部 ▾ 以下条件:

④ 添加条件

执行动作

④ 添加动作

选择“匹配设备触发”，依次选择产品“HCIP-IoT”，服务“Agriculture”，属性“Luminance”，范围是小于 30；

触发条件

需满足 全部 ▾ 以下条件:

| | | | | | |
|--------|----------|-------------|-----------|---|----|
| 匹配设备触发 | HCIP-IoT | Agriculture | Luminance | < | 30 |
|--------|----------|-------------|-----------|---|----|

触发机制 触发时效 300秒

点击“添加动作”；

触发条件

需满足 全部 ▾ 以下条件:

| | | | | | |
|--------|----------|-------------|-----------|---|----|
| 匹配设备触发 | HCIP-IoT | Agriculture | Luminance | < | 30 |
|--------|----------|-------------|-----------|---|----|

④ 添加条件

执行动作

下发命令 869505047088479 重新选择 Agriculture Agriculture... 参数配置

④ 添加动作

选择“下发命令”，点击“指定下发设备”；

④ 添加条件

执行动作

下发命令 指定下发设备

④ 添加动作

选择指定设备，点击“确认”；

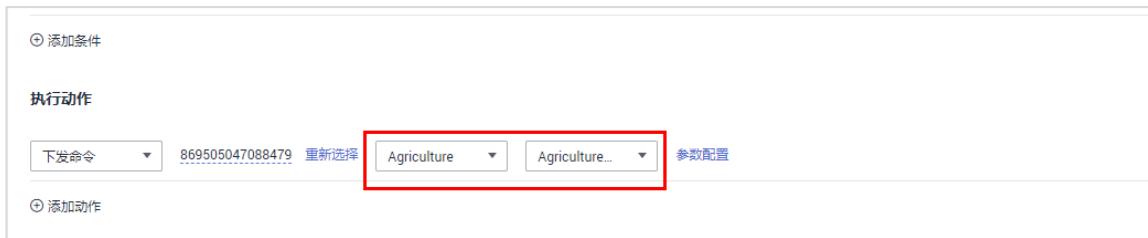
指定设备

| 所有产品 | 设备名称 | 查询 | |
|--|-----------------|--------------------------|----|
| 设备名称 | 设备标识码 | 所属产品 | 描述 |
| <input checked="" type="radio"/> 869505047088479 | 869505047088479 | 5f06c889c0cc2702cac609da | |

10 总条数: 2 < 1 >

确认 取消

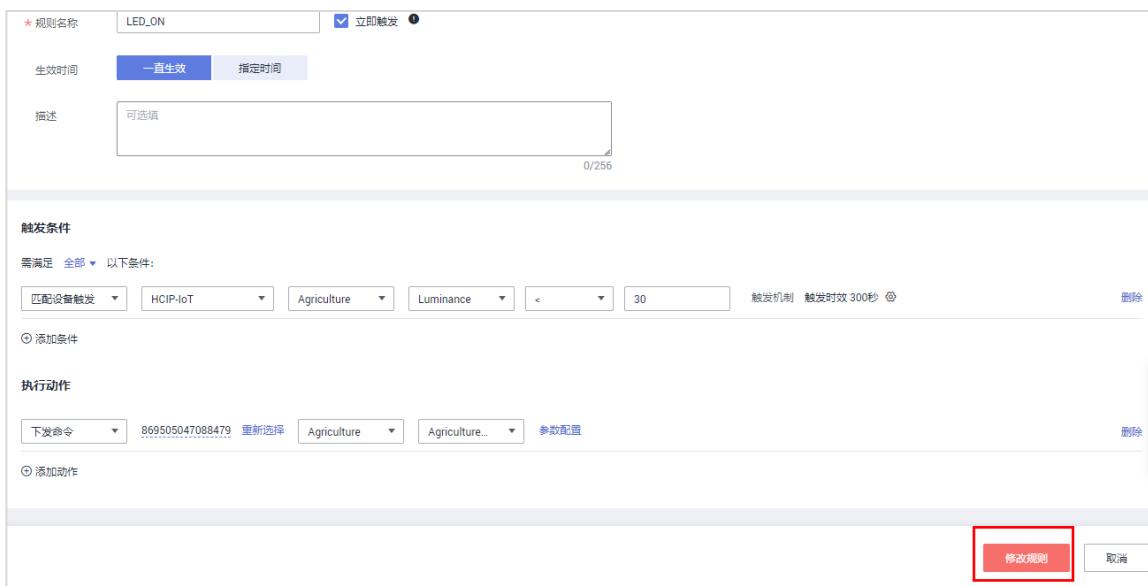
服务选择“Agriculture”，命令选择“Agriculture_Control_Light”，点击“参数配置”；



Light 选择“ON”，点击确认；



点击“创建规则”；



点击“返回规则列表”；



规则创建成功。



步骤 2 编译烧录智慧农业案例

将智慧农业扩展板插入小熊派，编译烧录智慧农业案例程序到开发板；

当智慧农业扩展板上光照强度小于 30，触发规则下发命令，LED 灯开。

11.3 练习题

11.3.1 创建规则，当光照强度大于 500 时，关闭 LED 灯

11.3.2 设置触发机制，进行无效触发抑制，时效 20s

12 消息通知服务 SMN 实验

12.1 实验介绍

12.1.1 关于本实验

本实验通过华为云物联网平台和消息通知服务 SMN 的联动，实现烟感告警的邮件和短信通知。

12.1.2 实验目的

- 掌握 SMN 服务的使用

12.2 实验任务配置步骤

12.2.1 配置华为云 SMN 服务

步骤 1 登录华为云 SMN 服务

登录华为云官网，搜索框输入 SMN，点击“消息通知服务 SMN”；



短信和邮件 100 条以内不收费，点击“立即使用”；



步骤 2 创建主题

点击“主题管理”》“主题”》“创建主题”；



自定义主题名称，自定义显示名，点击确定；



创建主题

* 主题名称: Cover

主题创建后，不允许修改主题名称。

显示名: 井盖

标签: 如果您需要使用同一标签标识多种云资源，即所有服务均可在标签输入框下拉选择同一标签，建议在 TMS 中创建预定义标签。 查看预定义标签 C

该主题还可以创建10个标签

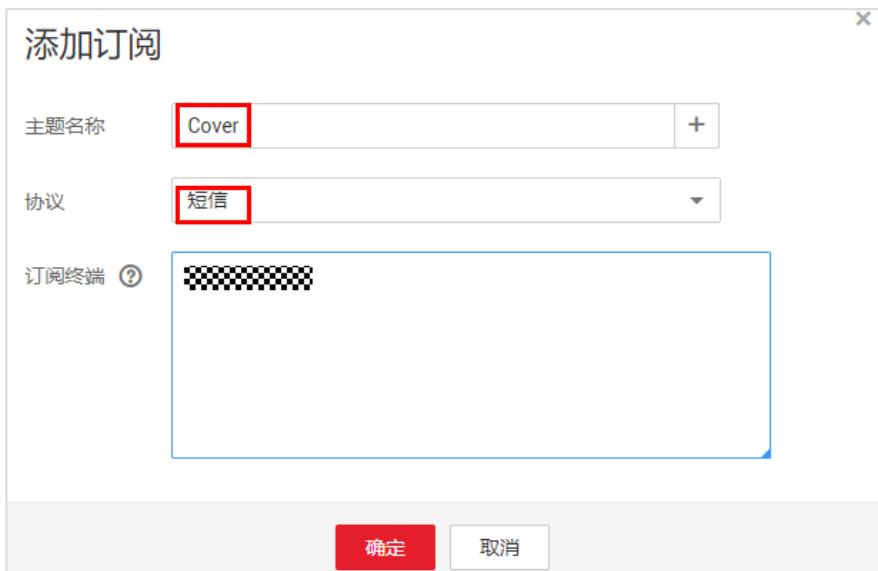
确定 取消

步骤 3 添加短信订阅

点击“订阅”》“添加订阅”；



选择前面创建的主题，协议选择“短信”，订阅终端输入自己的手机号，点击确定；



查收短信，点击短信中的确认链接，进行确认，刷新 SMN 页面，可以看到状态已经变成“已确认”。

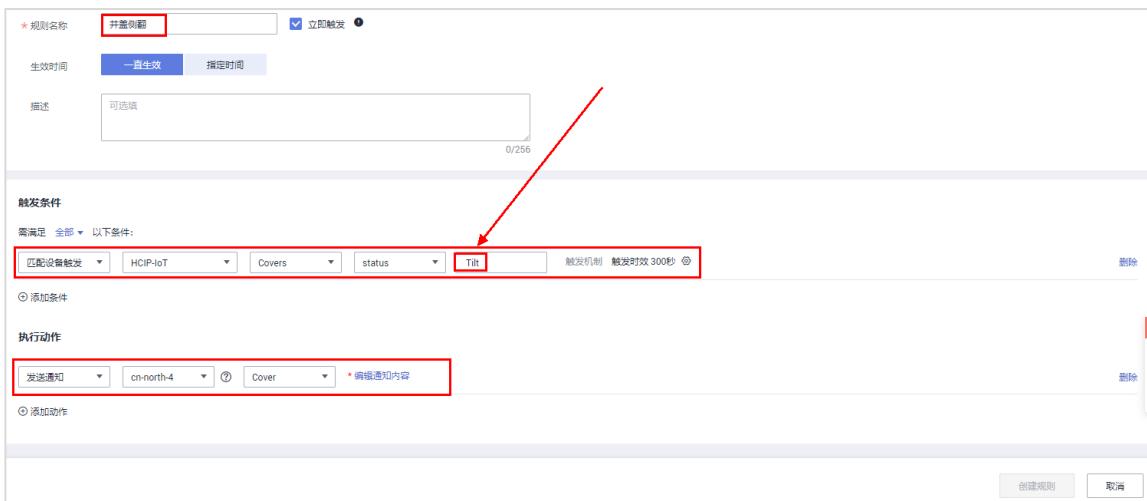


步骤 4 创建规则

登录华为云物联网平台，规则->创建规则->设备联动；

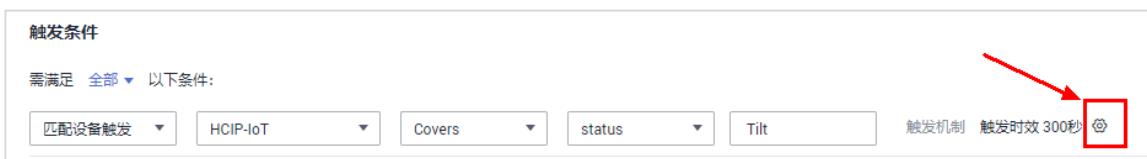
输入规则名称，触发条件选择井盖侧翻，Tilt 前要加一个空格；

执行动作选择发送通知，主题选择在 SMN 服务中创建的主题；



The screenshot shows the 'Create Rule' page. In the 'Trigger Conditions' section, under 'Match Device Trigger', the 'Covers' field is selected. In the dropdown menu, 'status' is chosen, followed by a space and 'Tilt'. A red arrow points from the text '触发机制选择“需要”，点击“确认”；' to the 'Tilt' button. The 'Action' section shows a notification action for 'cn-north-4' with a red box around the 'Edit Notification Content' link.

触发机制选择“需要”，点击“确认”；

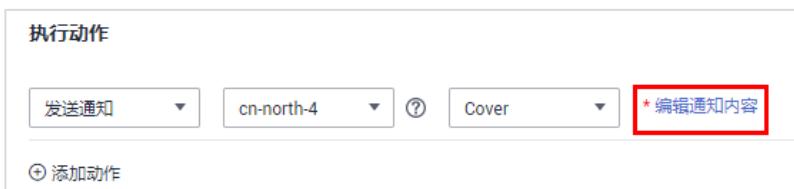


The screenshot shows the 'Device Data Trigger Mechanism' configuration dialog. Under 'Trigger Mechanism', the 'Trigger Effectiveness Duration' is set to '300 seconds'. A red arrow points from the text '自定义编辑通知内容，点击“确认”；' to the 'Confirm' button at the bottom right of the dialog.



The screenshot shows the 'Edit Notification Content' dialog. It contains two tabs: '不需要' (Not Required) and '需要' (Required). The '需要' tab is selected, highlighted with a red box. Below it is a note about setting a timer to trigger an action. The 'Data Duration' is set to '300 seconds'. At the bottom are 'Confirm' and 'Cancel' buttons.

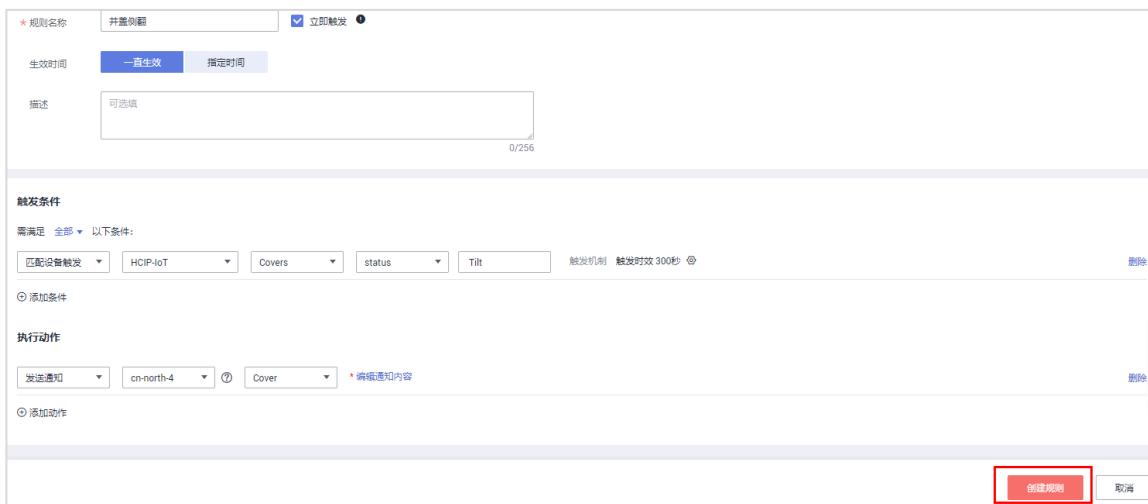
自定义编辑通知内容，点击“确认”；



The screenshot shows the 'Edit Notification Content' configuration dialog. It includes fields for 'Send Notification' (to 'cn-north-4'), 'Topic' ('Cover'), and a red box around the 'Edit Notification Content' link. Below is a '+ Add Action' button.



点击“创建规则”；



步骤 5 编译烧录智慧井盖案例

将智慧井盖扩展板插入小熊派，编译烧录智慧井盖案例程序到开发板；可以看到，当开发板发生侧翻时，订阅的手机会收到短信通知。

12.3 练习题

12.3.1 添加邮件通知功能

13

MQTT 开发自动售货机实验

13.1 实验介绍

13.1.1 关于本实验

本实验基于 MQTT 协议，实现自动售货机案例。

13.1.2 实验目的

- 掌握 MQTT 协议与物联网平台进行通信的方法

13.2 实验任务配置步骤

13.2.1 功能定义

步骤 1 自动售货机功能定义设计思路

表13-1 产品信息

| 属性 | 属性值 |
|------|------|
| 产品名称 | 自定义 |
| 协议类型 | MQTT |
| 数据格式 | JSON |
| 厂商名称 | 自定义 |
| 所属行业 | 自定义 |
| 设备类型 | 自定义 |

表13-2 服务名称: order

| 属性名称 | 数据类型 | 访问权限 | 数据范围 |
|----------------|-------------|-----------|---------|
| orderID | string(字符串) | 可读、可写、可执行 | 8 |
| userID | string(字符串) | 可读、可写、可执行 | 6 |
| userAge | int(整型) | 可读、可写、可执行 | 0~100 |
| deviceID | string(字符串) | 可读、可写、可执行 | 8 |
| area | string(字符串) | 可读、可写、可执行 | 2 |
| region | string(字符串) | 可读、可写、可执行 | 10 |
| longitude | decimal(小数) | 可读、可写、可执行 | 0~65535 |
| latitude | decimal(小数) | 可读、可写、可执行 | 0~65535 |
| orderTime | string(字符串) | 可读、可写、可执行 | 13 |
| payment | string(字符串) | 可读、可写、可执行 | 10 |
| water_Num | int(整型) | 可读、可写、可执行 | 0~65535 |
| cola_Num | int(整型) | 可读、可写、可执行 | 0~65535 |
| tea_Num | int(整型) | 可读、可写、可执行 | 0~65535 |
| coffee_Num | int(整型) | 可读、可写、可执行 | 0~65535 |
| milk_Num | int(整型) | 可读、可写、可执行 | 0~65535 |
| juice_Num | int(整型) | 可读、可写、可执行 | 0~65535 |
| yogurt_Num | int(整型) | 可读、可写、可执行 | 0~65535 |
| bread_Num | int(整型) | 可读、可写、可执行 | 0~65535 |
| sandwiches_Num | int(整型) | 可读、可写、可执行 | 0~65535 |
| sugar_Num | int(整型) | 可读、可写、可执行 | 0~65535 |
| status | int(整型) | 可读、可写、可执行 | 0~65535 |
| totalCost | int(整型) | 可读、可写、可执行 | 0~65535 |

13.2.2 自动售货机商品显示实验

步骤 1 案例配置

双击打开 targets\STM32L431_BearPi\Datas\user_demo.mk，并在底部插入如下配置，用于编译 oc_mqtt_vending_machine.c；

```
#example for oc mqtt
ifeq ($(CONFIG_USER_DEMO), "oc_mqtt_demo")
    user_demo_src = ${wildcard
$(TARGET_DIR)/Datas/oc_mqtt_demo/oc_mqtt_vending_machine.c}
endif

86 #example for cloud map
87 ifeq ($(CONFIG_USER_DEMO), "oc_cloud_map_demo")
88     user_demo_src = ${wildcard ${TARGET_DIR}/Datas/oc_cloud_map_demo/*.c}
89     user_demo_inc = -I ${TARGET_DIR}/Datas/oc_cloud_map_demo
90     user_demo_defs = -D CONFIG_OC_LWM2M_CLOUD_MAP_ENABLE=1
91 endif

92 #example for oc mqtt
93 ifeq ($(CONFIG_USER_DEMO), "oc_mqtt_demo")
94     user_demo_src = ${wildcard ${TARGET_DIR}/Datas/oc_mqtt_demo/oc_mqtt_vending_machine.c}
95 endif

96 C_SOURCES += $(user_demo_src)
97 C_INCLUDES += $(user_demo_inc)
98 C_SOURCES += ${user_hardware_src}
99 C_INCLUDES += ${user_hardware_inc}
100 C_DEFS += $(user_demo_defs)
101
102
```

步骤 2 修改.config 文件

打开 targets->STM32L431_BearPi->.config 文件；

注释掉 LWM2M 协议；

```
32 CONFIG_OCSERVICES_ENABLE=y
33 #CONFIG_OCLWM2M_ENABLE=y
34 # CONFIG_OCCOAP_ENABLE is not set
35 # CONFIG_OCLWM2MNULL is not set

46 # LWM2M
47 #CONFIG_OCLWM2MTINY_ENABLE=y
48 #CONFIG_LWM2M_AL_ENABLE=y
49 #CONFIG_WAKAAMALWM2M_ENABLE=v
```

注释掉 NB-IoT 配置；

```
44 # NB-IoT
45 #CONFIG_BOUDICA150_ENABLE=y
```

打开 WIFI 配置；

```
50 # WIFI
51 CONFIG_TCPIP_AL_ENABLE=y
52 CONFIG_ESP8266_ENABLE=y
53 CONFIG_ESP8266_SSID="Huawei"
54 CONFIG_ESP8266_PWD="12345678"
55 # MQTT
```

波特率修改为 115200;

```
3 CONFIG_UARTAT_RCVMAX=2048
4 CONFIG_UARTAT_BAUDRATE=115200
5 CONFIG_UARTAT_DEVNAME="atdev"
```

打开 MQTT 配置;

```
55 # MQTT
56 CONFIG_CJSON_ENABLE=y
57 CONFIG_MQTT_AL_ENABLE=y
58 CONFIG_PAHO_MQTT=y
59 CONFIG_PAHO_CONNECT_TIMEOUT=10000
60 CONFIG_PAHO_CMD_TIMEOUT=10000
61 CONFIG_PAHO_LOOPTIMEOUT=10
62 CONFIG_PAHO_SNDBUF_SIZE=2048
63 CONFIG_PAHO_RCVBUF_SIZE=2048
64 CONFIG_OCMQTT_ENABLE=y
65 CONFIG_OCMQTT_ATCMD=y
66 CONFIG_OC_MQTT_V5=y
67 CONFIG_OC_TINYMQTTV5_ENABLE=y
68 CONFIG_OC_MQTTV5_PROFILE=y
69 CONFIG_DTLS_AL_ENABLE=y
70 CONFIGMBEDTLS_ENABLE=y
71 CONFIGMBEDTLS_PSK=y
```

修改 CONFIG_USER_DEMO 为 oc_mqtt_demo;

```
42 # Demo
43 CONFIG_USER_DEMO="oc_mqtt_demo"
```

步骤 3 修改 iot_config.h 文件

打开 targets->STM32L431_BearPi->.config 文件;

注释掉 LWM2M 协议;

```
15 #define CONFIG_LINKDEMO_ENABLE 1
16 #define CONFIG_OCSERVICES_ENABLE 1
17 // #define CONFIG_OCLWM2M_ENABLE 1
18 #define CONFIG_SHELL_ENABLE 1
```

```
26  /*LWM2M*/
27  // #define CONFIG_OCLWM2MTINY_ENABLE 1
28  // #define CONFIG_LWM2M_AL_ENABLE 1
29  // #define CONFIG_WAKAAMALWM2M_ENABLE 1
```

注释掉 NB-IoT 配置；

```
24  /*NB-IoT*/
25  // #define CONFIG_BOUDICA150_ENABLE 1
```

打开 WIFI 配置；

```
30  /*WIFI*/
31  #define CONFIG_TCIP_AL_ENABLE 1
32  #define CONFIG_ESP8266_ENABLE 1
33  #define CONFIG_ESP8266_SSID "Huawei"
34  #define CONFIG_ESP8266_PWD "12345678"
```

波特率修改为 115200；

```
3  #define CONFIG_UARTAT_RCVMAX 2048
4  #define CONFIG_UARTAT_BAUDRATE 115200
5  #define CONFIG_UARTAT_DEVNAME "atdev"
```

打开 MQTT 配置；

```
35  /*MQTT*/
36  #define CONFIG_CJSON_ENABLE 1
37  #define CONFIG_MQTT_AL_ENABLE 1
38  #define CONFIG_PAHO_MQTT 1
39  #define CONFIG_PAHO_CONNECT_TIMEOUT 10000
40  #define CONFIG_PAHO_CMD_TIMEOUT 10000
41  #define CONFIG_PAHO_LOOPTIMEOUT 10
42  #define CONFIG_PAHO_SNDBUF_SIZE 2048
43  #define CONFIG_PAHO_RCVBUF_SIZE 2048
44  #define CONFIG_OCMQTT_ENABLE 1
45  #define CONFIG_OCMQTT_ATCMD 1
46  #define CONFIG_OC_MQTT_V5 1
47  #define CONFIG_OC_TINYMQTTCV5_ENABLE 1
48  #define CONFIG_OC_MQTTCV5_PROFILE 1
49  #define CONFIG_DTLS_AL_ENABLE 1
50  #define CONFIGMBEDTLS_ENABLE 1
51  #define CONFIGMBEDTLS_PSK 1
```

修改 CONFIG_USER_DEMO 为 oc_mqtt_demo；

```
22  /*Demo*/
23  #define CONFIG_USER_DEMO "oc mqtt demo"
```

步骤 4 在平台上注册设备

打开前面注册的物联网平台，点击左侧的“产品”，点击“创建产品”；

| 产品名称 | 产品ID | 设备类型 | 协议类型 | 操作 |
|----------|--------------------------|------|------------|---------------------------------------|
| HCIP-IoT | 5f06c889c0cc2702cac609da | IoT | LwM2M/CoAP | 详情 删除 |
| HCIA-IoT | 5ef9b5a069c46102cb120886 | IoT | LwM2M/CoAP | 详情 删除 |

填写产品信息，协议类型选择 MQTT，数据格式选择 JSON，其余参数自定义，需使用字母和数字的组合，点击右下角“立即创建”。

可以看到创建成功的产品。

| 产品名称 | 产品ID | 设备类型 | 协议类型 | 操作 |
|----------|--------------------------|------|------------|---------------------------------------|
| MQTT | 5fe88a4f958e402caded29 | MQTT | MQTT | 详情 删除 |
| HCIP-IoT | 5f06c889c0cc2702cac609da | IoT | LwM2M/CoAP | 详情 删除 |
| HCIA-IoT | 5ef9b5a069c46102cb120886 | IoT | LwM2M/CoAP | 详情 删除 |

进入“设备—>所有设备”页面，单击“注册设备”，填写设备注册参数，点击“确定”。

单设备注册

* 所属资源空间 ⑦ DefaultApp_hwstaff_aiot1_iot

* 所属产品 MQTT

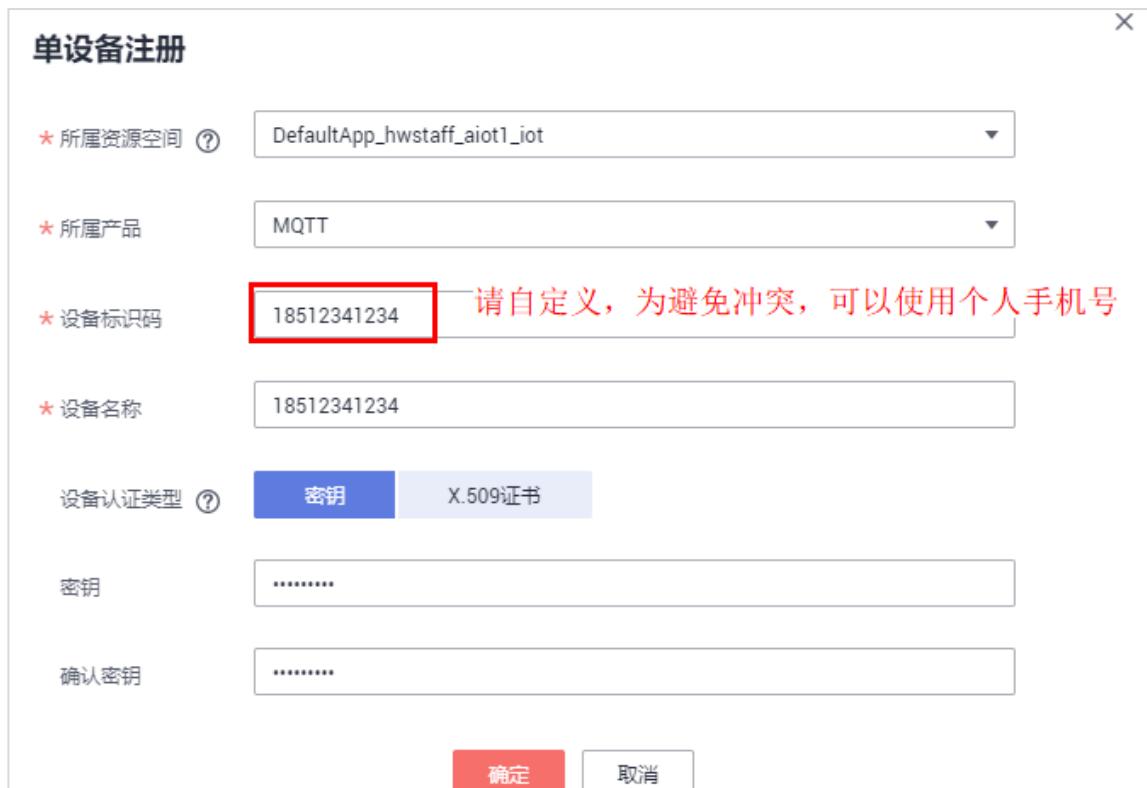
* 设备标识码 18512341234 请自定义，为避免冲突，可以使用个人手机号

* 设备名称 18512341234

设备认证类型 ⑦ 密钥 X.509证书

密钥
确认密钥

确定 **取消**



- 选择产品：选择上一步创建的产品模型。
- 设备标识码：自定义，填写任意包含数字或英文字母的字符串。
- 密钥：自定义，填写任意包含数字或英文字母的字符串。

注册设备成功，将设备 ID、密钥复制粘贴自行保存到本地，点击“确定”；

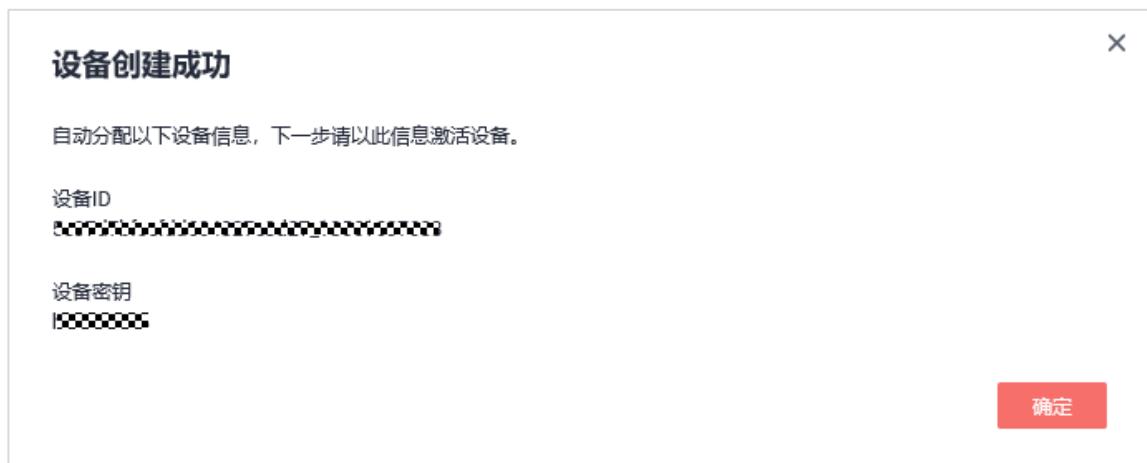
设备创建成功

自动分配以下设备信息，下一步请以此信息激活设备。

设备ID
XXXXXXXXXXXXXX

设备密钥
XXXXXXXX

确定



在“所有设备”中可以看到注册的设备。



The screenshot shows a table of registered devices. There is one entry:

| 状态 | 设备名称 | 设备标识码 | 所属资源空间 | 所属产品 | 节点类型 | 操作 |
|-----|-------------|-------------|------------------------------|------|------|--------------|
| 未激活 | 18512341234 | 18512341234 | DefaultApp_hwstaff_aiot1_iot | MQTT | 直连设备 | 查看 删除 联接 |

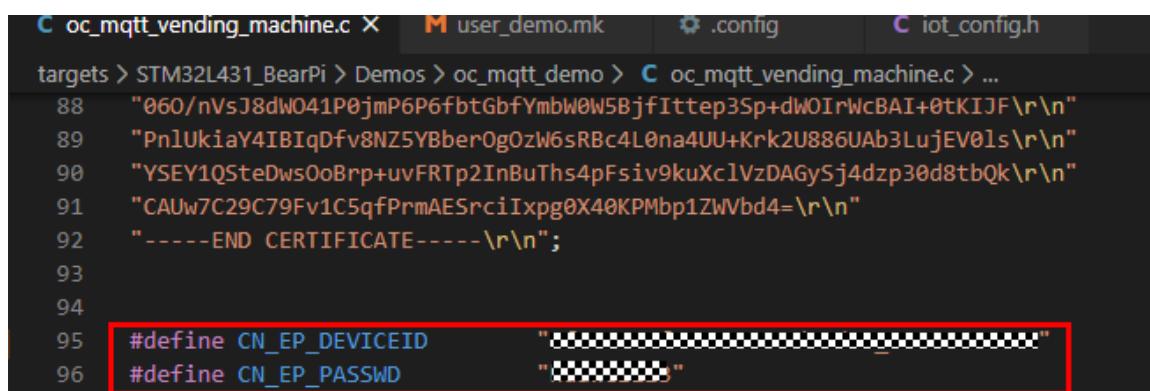
Page navigation: 10 / Total: 1

步骤 5 在代码中修改设备信息

打开 targets\STM32L431_BearPi\Datas\oc_mqtt_vending_machine.c 文件；

修改 CN_EP_DEVICEID 为在物联网平台注册设备时生成的设备 ID；

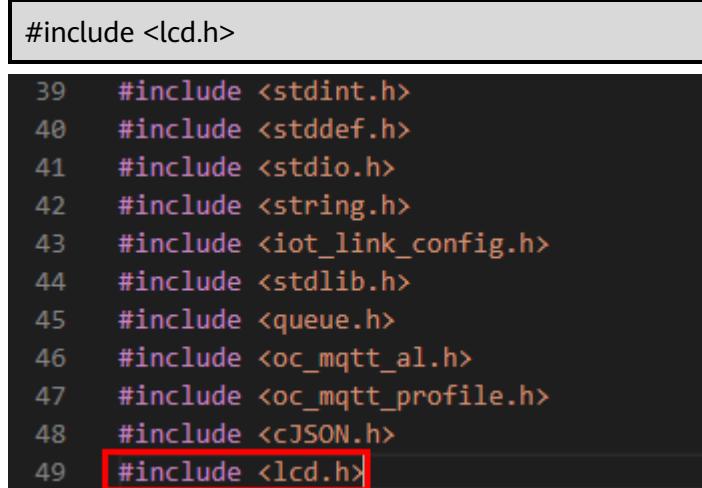
修改 CN_EP_PASWD 为在物联网平台注册设备时填写的密钥；



```
targets > STM32L431_BearPi > Datas > oc_mqtt_demo > C oc_mqtt_vending_machine.c > ...
88     "060/nVsJ8dW041P0jmP6P6fbtGbFYmbW0W5BjfIttep3Sp+dWOIrWcBAI+0tKIJF\r\n"
89     "Pn1UkiaY4IBIqDfv8Nz5YBberOg0zW6sRBc4L0na4UU+Krk2U886UAb3LujeV0ls\r\n"
90     "YSEY1QSteDwsOoBrp+uvFRTp2InBuThs4pFsiv9kuXc1VzDAGySj4dzp30d8tbQk\r\n"
91     "CAUw7C29C79Fv1C5qfPrmAESrciIxpg0X40KPMbp1ZWbd4=\r\n"
92     "-----END CERTIFICATE-----\r\n";
93
94
95 #define CN_EP_DEVICEID      "██████████"
96 #define CN_EP_PASWD         "██████████"
```

步骤 6 添加 LCD 屏幕显示所需变量并初始化

添加头文件 lcd.h 和 stdlib.h



```
#include <lcd.h>

39  #include <stdint.h>
40  #include <stddef.h>
41  #include <stdio.h>
42  #include <string.h>
43  #include <iot_link_config.h>
44  #include <stdlib.h>
45  #include <queue.h>
46  #include <oc_mqtt_al.h>
47  #include <oc_mqtt_profile.h>
48  #include <cJSON.h>
49  #include <lcd.h>
```

添加商品显示所需的变量并初始化。

```
static int X[2] = {10,120};  
static int Y[7] = {10,40,70,100,130,160,190};  
static int goodsOptionX = 10;  
static int goodsOptionY = 10;  
char* goodsView[10] = {"water 1","cola 3","tea 3","coffee 5","milk 4","juice 3","yogurt  
4","bread 7","sandwich 7","sugar 2"};  
char* Submit_View[2] = {"Submit","Cancel"};  
int Goods_Price[10] = {1,3,3,5,4,3,4,7,7,2};  
int goods_position[10] = {0,1,2,3,4,5,6,7,8,9};
```

```
155     "-----END PRIVATE KEY-----\r\n";  
156     static const char *s_client_pk_pwd = "123456";  
157  
158     static int X[2] = {10,120};  
159     static int Y[7] = {10,40,70,100,130,160,190};  
160     static int goodsOptionX = 10;  
161     static int goodsOptionY = 10;  
162     char* goodsView[10] = {"water 1","cola 3","tea 3","coffee 5","milk 4","juice 3","yogurt 4","bread 7","sandwich 7","sugar 2"};  
163     char* Submit_View[2] = {"Submit","Cancel"};  
164     int Goods_Price[10] = {1,3,3,5,4,3,4,7,7,2};  
165     int goods_position[10] = {0,1,2,3,4,5,6,7,8,9};  
166  
167     //use this function to push all the message to the buffer  
168     static int app_msg_deal(oc_mqtt_profile_msgrcv_t *msg)  
169     {
```

步骤 7 添加 LCD 初始化并显示连接平台的代码

```
LCD_Init();  
LCD_Clear(BLACK);  
POINT_COLOR = GREEN;  
LCD_ShowString(X[0], Y[3], 240, 24, 24, "Connecting to");  
LCD_ShowString(X[0], Y[4], 240, 24, 24, "IoT Platform...");
```

```
380 //案例入口函数  
381 int standard_app_demo_main()  
382 {  
383  
384     s_queue_rcvmsg = queue_create("queue_rcvmsg",2,1); //创建队列，用于命令响应  
385  
386     LCD_Init();  
387     LCD_Clear(BLACK);  
388     POINT_COLOR = GREEN;  
389     LCD_ShowString(X[0], Y[3], 240, 24, 24, "Connecting to");  
390     LCD_ShowString(X[0], Y[4], 240, 24, 24, "IoT Platform...");  
391  
392  
393     (void) osal_task_create("demo_reportmsg",task_reportmsg_entry,NULL,0x1200,NULL,8);  
394     (void) osal_task_create("demo_rcvmsg",task_rcvmsg_entry,NULL,0x500,NULL,8);  
395  
396 }
```

步骤 8 添加连接失败 LCD 显示代码

```
LCD_Clear(BLACK);
LCD_ShowString(10, Y[3], 240, 24, 24, "Connect IoT Platform");
LCD_ShowString(10, Y[4], 240, 24, 24, "failed, Please Reset!");

365  /*配置设备连接到物联网平台*/
366  ret = oc_mqtt_profile_connect(&connect_para);
367  if((ret != (int)en_oc_mqtt_err_ok))
368  {
369      (void) printf("config:err :code:%d\r\n",ret);//连接平台失败
370
371      LCD_Clear(BLACK);
372      LCD_ShowString(10, Y[3], 240, 24, 24, "Connect IoT Platform");
373      LCD_ShowString(10, Y[4], 240, 24, 24, "failed, Please Reset!");
374
375      return -1;
376  }else{
377      printf("config OK!\r\n");//连接平台成功
378 }
```

步骤 9 添加显示商品代码

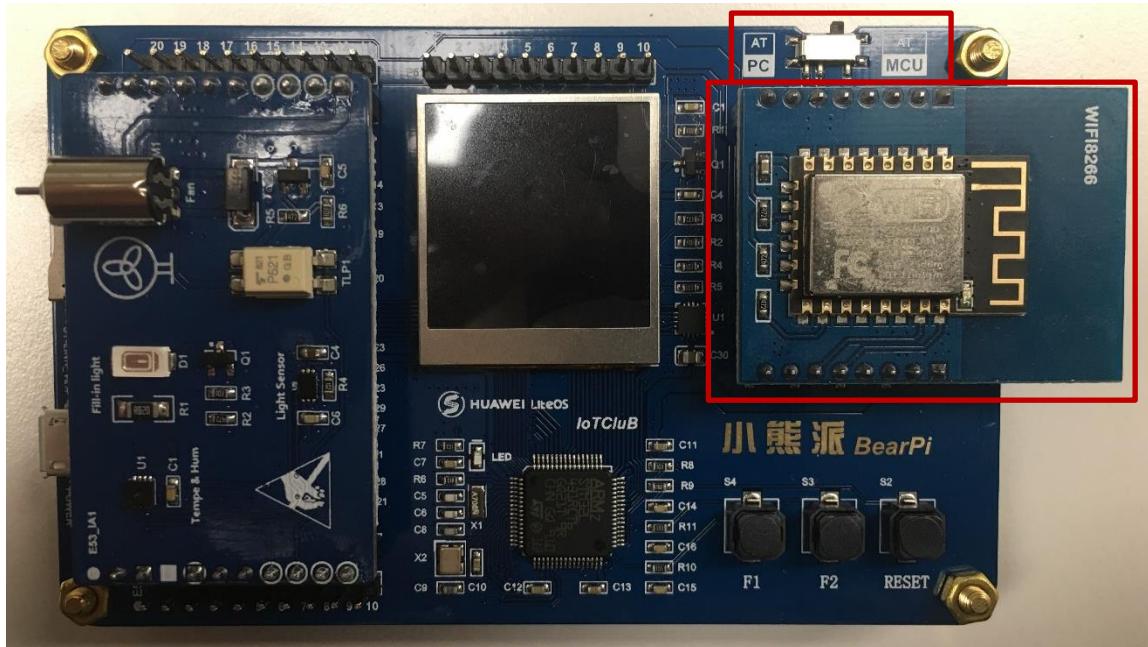
```
LCD_Clear(BLACK);
for(int i=0;i<10;i++){
    if(i<5){
        LCD_ShowString(X[0], Y[i], 240, 24, 24, goodsView[goods_position[i]]);
    }else{
        LCD_ShowString(X[1], Y[i-5], 240, 24, 24, goodsView[goods_position[i]]);
    }
}
LCD_ShowString(X[0], Y[5], 240, 24, 24, Submit_View[0]);
LCD_ShowString(X[1], Y[5], 240, 24, 24, Submit_View[1]);

LCD_ShowString(X[0]-10, goodsOptionY, 10, 24, 24, "*");
LCD_ShowxNum(X[1], Y[6], 0, 3, 24, 1);
```

```
371     LCD_Clear(BLACK);
372     LCD_ShowString(10, Y[3], 240, 24, 24, "Connect IoT Platform");
373     LCD_ShowString(10, Y[4], 240, 24, 24, "failed, Please Reset!");
374
375     return -1;
376 }else{
377     printf("config OK!\r\n"); //连接平台成功
378     LCD_Clear(BLACK);
379     for(int i=0;i<10;i++){
380         if(i<5){
381             LCD_ShowString(X[0], Y[i], 240, 24, 24, goodsView[goods_position[i]]);
382         }else{
383             LCD_ShowString(X[1], Y[i-5], 240, 24, 24, goodsView[goods_position[i]]);
384         }
385     }
386     LCD_ShowString(X[0], Y[5], 240, 24, 24, Submit_View[0]);
387     LCD_ShowString(X[1], Y[5], 240, 24, 24, Submit_View[1]);
388
389     LCD_ShowString(X[0]-10, goodsOptionY, 10, 24, 24, "*");
390     LCD_ShowxNum(X[1], Y[6], 0, 3, 24, 1);
391 }
392
393     oc_report_normal();
394
395     return 0;
396 }
```

步骤 10 编译烧录程序，查看结果

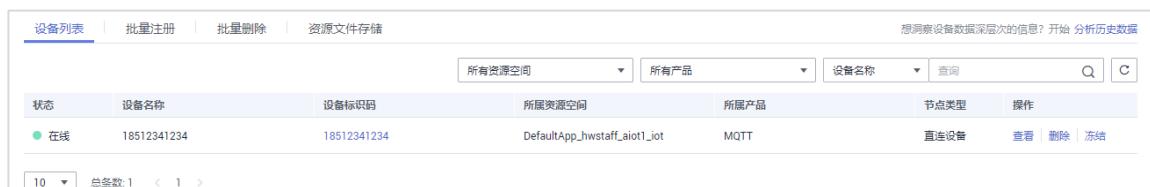
将 WIFI8266 通信扩展板按正确方向插到小熊派开发板上，并将串口模式的切换开关拨到 AT<->MCU 模式，开启 WIFI 热点或路由器；



将开发板用 USB 线，连接到电脑上，编译烧录，等待开发板上 LCD 屏幕显示商品信息。



登录华为云物联网平台，可以看到设备已经在线。



The screenshot shows the 'Device List' page of the Huawei Cloud IoT Platform. The table displays the following information for one device:

| 状态 | 设备名称 | 设备标识码 | 所属资源空间 | 所属产品 | 节点类型 | 操作 |
|------|-------------|-------------|-----------------------------|------|------|--|
| ● 在线 | 18512341234 | 18512341234 | DefaultApp_hwstaff_iot1_iot | MQTT | 直连设备 | 查看 删除 冻结 |

Below the table, there are pagination controls: '10' (selected), '总条数: 1', '< 1 >'.

13.2.3 商品选择实验

步骤 1 添加按键检测函数，实现商品选择

开发板上有按键 F1 (GPIOB, GPIO_PIN_2) 和按键 F2 (GPIOB, GPIO_PIN_3), 我们使用按键 F1 来选择商品，F2 来确定添加购物车。

在 oc_mqtt_v5_demo.c 中添加按键检测函数 key_detect;

```
static int key_detect(void *args)
{
    while(1)
    {
        if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_2)==GPIO_PIN_RESET)//查询按键
KEY1 低电平
        {
            osal_task_sleep(50);
            if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_2)==GPIO_PIN_RESET)//查询按键
KEY1 低电平
            {
                for(int i=0;i<6;i++){
                    if(goodsOptionY==Y[i]&&i<5){
                        LCD_ShowString(goodsOptionX-10, goodsOptionY, 10, 24,
24, " ");
                        goodsOptionY=Y[i+1];
                        LCD_ShowString(goodsOptionX-10, goodsOptionY, 10, 24,
24, "***");
                        break;
                    }else if(goodsOptionY==Y[i]&&i==5){
                        LCD_ShowString(goodsOptionX-10, goodsOptionY, 10, 24,
24, " ");
                        goodsOptionY=Y[0];
                        goodsOptionX=(goodsOptionX==X[0])?X[1]:X[0];
                        LCD_ShowString(goodsOptionX-10, goodsOptionY, 10, 24,
24, "***");
                    }
                }
            }
        }
        osal_task_sleep(50);
    }
}
```

```
        }
        return 0;
    }

398 static int key_detect(void *args)
399 {
400     while(1)
401     {
402         if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_2)==GPIO_PIN_RESET)//查询按键KEY1低电平
403         {
404             osal_task_sleep(50);
405             if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_2)==GPIO_PIN_RESET)//查询按键KEY1低电平
406             {
407                 for(int i=0;i<6;i++){
408                     if(goodsOptionY==Y[i]&&i<5){
409                         LCD_ShowString(goodsOptionX-10, goodsOptionY, 10, 24, 24, " ");
410                         goodsOptionY=Y[i+1];
411                         LCD_ShowString(goodsOptionX-10, goodsOptionY, 10, 24, 24, "*");
412                         break;
413                     }else if(goodsOptionY==Y[i]&&i==5){
414                         LCD_ShowString(goodsOptionX-10, goodsOptionY, 10, 24, 24, " ");
415                         goodsOptionY=Y[0];
416                         goodsOptionX=(goodsOptionX==X[0])?X[1]:X[0];
417                         LCD_ShowString(goodsOptionX-10, goodsOptionY, 10, 24, 24, "*");
418                     }
419                 }
420             }
421             osal_task_sleep(50);
422         }
423     }
424     return 0;
425 }
426
427 //案例入口函数
428 int standard_app_demo_main()
```

在 standard_app_demo_main 中添加创建按键检测任务的代码，并修改上报数据任务的初始化内存大小为 0x1200；

```
osal_task_create("key_detect",key_detect,NULL,0x500,NULL,8);
```

```
427 //案例入口函数
428 int standard_app_demo_main()
429 {
430
431     s_queue_rcvmsg = queue_create("queue_rcvmsg",2,1); //创建队列，用于命令响应
432
433     LCD_Init();
434     LCD_Clear(BLACK);
435     POINT_COLOR = GREEN;
436     LCD_ShowString(X[0], Y[3], 240, 24, 24, "Connecting to");
437     LCD_ShowString(X[0], Y[4], 240, 24, 24, "IoT Platform...");
438
439     osal_task_create("key_detect",key_detect,NULL,0x500,NULL,8);
440
441     (void) osal_task_create("demo_reportmsg",task_reportmsg_entry,NULL,0x1200,NULL,8);
442     (void) osal_task_create("demo_rcvmsg",task_rcvmsg_entry,NULL,0x500,NULL,8);
443
444 }
```

编译烧录，提示成功后，开启 WiFi 热点，在小熊派开发板上使用按键 F1 选择商品。

13.2.4 上报数据到平台实验

步骤 1 添加数据上报所需的变量代码

在函数 oc_report_normal 中为每一个订单字段定义用于拼装 JSON 数据的变量；

```
oc_mqtt_profile_kv_t orderID_List;
oc_mqtt_profile_kv_t userID_List;
oc_mqtt_profile_kv_t userAge_List;
oc_mqtt_profile_kv_t deviceID_List;
oc_mqtt_profile_kv_t area_List;
oc_mqtt_profile_kv_t Region_List;
oc_mqtt_profile_kv_t Longitude_List;
oc_mqtt_profile_kv_t Latitude_List;
oc_mqtt_profile_kv_t orderTime_List;
oc_mqtt_profile_kv_t Pay_List;
oc_mqtt_profile_kv_t Goods_Num_List[10];
oc_mqtt_profile_kv_t Goods_Price_List[10];
oc_mqtt_profile_kv_t Status_List;
oc_mqtt_profile_kv_t Total_Cost_List;
```

```
301     else if(times == EN_OC_MQTT_PROFILE_MSG_TYPE_UP_PROPERTYREPORT)
302     {
303         oc_mqtt_profile_kv_t orderID_List;
304         oc_mqtt_profile_kv_t userID_List;
305         oc_mqtt_profile_kv_t userAge_List;
306         oc_mqtt_profile_kv_t deviceID_List;
307         oc_mqtt_profile_kv_t area_List;
308         oc_mqtt_profile_kv_t Region_List;
309         oc_mqtt_profile_kv_t Longitude_List;
310         oc_mqtt_profile_kv_t Latitude_List;
311         oc_mqtt_profile_kv_t orderTime_List;
312         oc_mqtt_profile_kv_t Pay_List;
313         oc_mqtt_profile_kv_t Goods_Num_List[10];
314         oc_mqtt_profile_kv_t Goods_Price_List[10];
315         oc_mqtt_profile_kv_t Status_List;
316         oc_mqtt_profile_kv_t Total_Cost_List;
317
318     }
319     return;
320 }
```

添加初始化服务代码；

```
///< initialize the service
s_device_service.event_time = NULL;
s_device_service.service_id = "order";
s_device_service.service_property = &orderID_List;
s_device_service.nxt = NULL;
```

```
315     oc_mqtt_profile_kv_t Status_List;
316     oc_matt_profile_kv_t Total_Cost_List;
317     ///< initialize the service
318     s_device_service.event_time = NULL;
319     s_device_service.service_id = "order";
320     s_device_service.service_property = &orderID_List;
321     s_device_service.nxt = NULL;
322
323 }
324 return;
325 }
326 /*命令处理任务*/
327 static int task_rcvmsg_entry( void *args )
```

继续添加每一个订单数据所需要的变量；

```
int orderID = 10000001;
int userID = 123456;
int userAge = 23;
char* deviceID = "WZ_1-001";
char* area = "WZ";
char* Regions[] =
{"School","Mall","Hospital","Community","Industry","Park","Station"};
int Region_Random = 0;
int longitude = 120.65;
int latitude = 28.02;
int orderTime = 9335848;
int orderTime1 = 157255;
char* Pay[] = {"Cash","WeChat","Alipay","UnionPay"};
int pay_Random = 0;
int Status = 0;
int Total_Num = 0;
int Total_Cost = 0;
int Goods_Num[10] = {0};
```

```
char* Goods_Num_String[10] =  
{"water_Num","cola_Num","tea_Num","coffee_Num","milk_Num","juice_Num","yogurt  
_Num","bread_Num","sanwiches_Num","sugar_Num"};  
char* Goods_Price_String[10] =  
{"water_Price","cola_Price","tea_Price","coffee_Price","milk_Price","juice_Price","yogurt  
_Price","bread_Price","sanwiches_Price","sugar_Price"};  
char orderID_string[9];  
char userID_string[6];  
char orderTime_string[13];  
  
319     s_device_service.service_id = "order";  
320     s_device_service.service_property = &orderID_List;  
321     s_device_service.nxt = NULL;  
322  
323     int orderID = 10000001;  
324     int userID = 123456;  
325     int userAge = 23;  
326     char* deviceID = "WZ_1-001";  
327     char* area = "WZ";  
328     char* Regions[] = {"School","Mall","Hospital","Community","Industry","Park","Station"};  
329     int Region_Random = 0;  
330     int longitude = 120.65;  
331     int latitude = 28.02;  
332     int orderTime = 9335848;  
333     int orderTime1 = 157255;  
334     char* Pay[] = {"Cash","WeChat","Alipay","UnionPay"};  
335     int pay_Random = 0;  
336     int Status = 0;  
337     int Total_Num = 0;  
338     int Total_Cost = 0;  
339     int Goods_Num[10] = {0};  
340  
341     char* Goods_Num_String[10] = {"water_Num","cola_Num","tea_Num","coffee_Num","milk_Num","juice_Num","yogurt  
_Num","bread_Num","sanwiches_Num","sugar_Num"};  
342     char* Goods_Price_String[10] = {"water_Price","cola_Price","tea_Price","coffee_Price","milk_Price","juice_Price","yogurt_Price","bread  
_Price","sanwiches_Price","sugar_Price"};  
343     char orderID_string[9];  
344     char userID_string[6];  
345     char orderTime_string[13];  
346  
347 }  
348 return;  
349 }  
/*命令处理任务*/
```

步骤 2 添加购物车工程代码

在函数 oc_report_normal 中添加按键 F2 循环监测和将商品加入购物车的代码；

```
while(1)  
{  
    osal_task_sleep(50);  
    if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_3)==GPIO_PIN_RESET)//查询按键  
KEY2 低电平  
    {  
        osal_task_sleep(50);  
        if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_3)==GPIO_PIN_RESET)//查询  
按键 KEY2 低电平  
        {  
            if(goodsOptionY==Y[5]) {
```

```
    }else {
        for(int i=0;i<10;i++){
            if(goodsOptionX==X[0]){
                if(goodsOptionY==Y[i]){
                    Goods_Num[goods_position[i]]++;
                    Total_Num++;
                    Total_Cost+=Goods_Price[goods_position[i]];
                    LCD_ShowxNum(X[1], Y[6], Total_Num, 3, 24,
1);
                    break;
                }
            }else{
                if(goodsOptionY==Y[i-5]){
                    Goods_Num[goods_position[i]]++;
                    Total_Num++;
                    Total_Cost+=Goods_Price[goods_position[i]];
                    LCD_ShowxNum(X[1], Y[6], Total_Num, 3, 24,
1);
                    break;
                }
            }
        }
    }
}
```

```
341     char* Goods_Num_String[10] = {"water_Num", "cola_Num", "tea_Num", "coffee_Num", "milk_Num", "juice_Num", "yogurt_Num", "b  
342     char* Goods_Price_String[10] = {"water_Price", "cola_Price", "tea_Price", "coffee_Price", "milk_Price", "juice_Price", "yogurt_Price", "b  
343     char orderID_string[9];  
344     char userID_string[6];  
345     char orderTime_string[13];  
346     while(1)  
347     {  
348         osal_task_sleep(50);  
349         if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_3)==GPIO_PIN_RESET)//查询按键KEY2低电平  
350         {  
351             osal_task_sleep(50);  
352             if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_3)==GPIO_PIN_RESET)//查询按键KEY2低电平  
353             {  
354                 if(goodsOptionY==Y[5])  
355                 {  
356                     for(int i=0;i<10;i++)  
357                     {  
358                         if(goodsOptionX==X[0])  
359                         {  
360                             if(goodsOptionY==Y[i])  
361                             {  
362                                 Goods_Num[goods_position[i]]++;  
363                                 Total_Num++;  
364                                 Total_Cost+=Goods_Price[goods_position[i]];  
365                                 LCD_ShowxNum(X[1], Y[6], Total_Num, 3, 24, 1);  
366                                 break;  
367                             }  
368                         }  
369                         else{  
370                             if(goodsOptionY==Y[i-5])  
371                             {  
372                                 Goods_Num[goods_position[i]]++;  
373                                 Total_Num++;  
374                                 Total_Cost+=Goods_Price[goods_position[i]];  
375                                 LCD_ShowxNum(X[1], Y[6], Total_Num, 3, 24, 1);  
376                                 break;  
377                             }  
378                         }  
379                     }  
380                 }  
381             }  
382         }  
383     }  
384 }
```

编译烧录程序，开启 WIFI 热点，可以使用 F1 选择商品，使用 F2 按键将商品添加到购物车，LCD 屏幕右下角可以看到数量增加。



步骤 3 添加组装 JSON 数据的代码

```
sprintf(orderID_string,"%8d",orderID);
orderID_List.key = "orderID";
orderID_List.value = (char *)orderID_string;
orderID_List.type = EN_OC_MQTT_PROFILE_VALUE_STRING;
orderID_List.nxt = &userID_List;

pay_Random = rand()%3;
userID = rand()%1000000;
sprintf(userID_string,"%06d",userID);
userID_List.key = "userID";
userID_List.value = (char *)userID_string;
userID_List.type = EN_OC_MQTT_PROFILE_VALUE_STRING;
userID_List.nxt = &userAge_List;

userAge_List.key = "userAge";
userAge_List.value = (char *)&userAge;
userAge_List.type = EN_OC_MQTT_PROFILE_VALUE_INT;
userAge_List.nxt = &deviceID_List;

deviceID_List.key = "deviceID";
deviceID_List.value = (char *)deviceID;
deviceID_List.type = EN_OC_MQTT_PROFILE_VALUE_STRING;
deviceID_List.nxt = &area_List;

area_List.key = "area";
area_List.value = (char *)area;
area_List.type = EN_OC_MQTT_PROFILE_VALUE_STRING;
area_List.nxt = &Region_List;

Region_Random = rand()%6;
Region_List.key = "region";
Region_List.value = (char *)Regions[Region_Random];
Region_List.type = EN_OC_MQTT_PROFILE_VALUE_STRING;
Region_List.nxt = &Longitude_List;
```

```
Longitude_List.key = "longitude";
Longitude_List.value = (char *)&longitude;
Longitude_List.type = EN_OC_MQTT_PROFILE_VALUE_INT;
Longitude_List.nxt = &Latitude_List;

Latitude_List.key = "latitude";
Latitude_List.value = (char *)&latitude;
Latitude_List.type = EN_OC_MQTT_PROFILE_VALUE_INT;
Latitude_List.nxt = &orderTime_List;

sprintf(orderTime_string,"%d%d",orderTime1,orderTime);
orderTime_List.key = "orderTime";
orderTime_List.value = (char *)orderTime_string;
orderTime_List.type =
EN_OC_MQTT_PROFILE_VALUE_STRING;
orderTime_List.nxt = &Pay_List;

pay_Random = rand()%3;
Pay_List.key = "payment";
Pay_List.value = (char *)Pay[pay_Random];
Pay_List.type = EN_OC_MQTT_PROFILE_VALUE_STRING;
Pay_List.nxt = &Goods_Num_List[0];

if(goodsOptionX==X[1]){
    Status = 0;
    for(int i=0;i<10;i++){
        Goods_Num[i]=0;
    }
    Total_Cost = 0;
}else{
    Status = 1;
}

for(int i=0;i<10;i++){
    Goods_Num_List[i].key = Goods_Num_String[i];
    Goods_Num_List[i].value = (char *)&Goods_Num[i];
```

```
    Goods_Num_List[i].type =
EN_OC_MQTT_PROFILE_VALUE_INT;
    Goods_Num_List[i].nxt = &Goods_Price_List[i];
    Goods_Price_List[i].key = Goods_Price_String[i];
    Goods_Price_List[i].value = (char *)&Goods_Price[i];
    Goods_Price_List[i].type =
EN_OC_MQTT_PROFILE_VALUE_INT;
    if(i<9){
        Goods_Price_List[i].nxt = &Goods_Num_List[i+1];
    }
}
Goods_Price_List[9].nxt = &Status_List;

Status_List.key = "status";
Status_List.value = (char *)&Status;
Status_List.type = EN_OC_MQTT_PROFILE_VALUE_INT;
Status_List.nxt = &Total_Cost_List;

Total_Cost_List.key = "totalCost";
Total_Cost_List.value = (char *)&Total_Cost;
Total_Cost_List.type = EN_OC_MQTT_PROFILE_VALUE_INT;
Total_Cost_List.nxt = NULL;
```

```
351     osal_task_sleep(50);
352     if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_3)==GPIO_PIN_RESET)//查询按键KEY2低电平
353     {
354         if(goodsOptionY==Y[5])
355         {
356             sprintf(orderID_string,"%8d",orderID);
357             orderID_List.key = "orderID";
358             orderID_List.value = (char *)orderID_string;
359             orderID_List.type = EN_OC_MQTT_PROFILE_VALUE_STRING;
360             orderID_List.nxt = &userID_List;
361
362             pay_Random = rand()%3;
363             userID = rand()%1000000;
364             sprintf(userID_string,"%06d",userID);
365             userID_List.key = "userID";
366             userID_List.value = (char *)userID_string;
367             userID_List.type = EN_OC_MQTT_PROFILE_VALUE_STRING;
368             userID_List.nxt = &userAge_List;
369
370             userAge_List.key = "userAge";
371             userAge_List.value = (char *)&userAge;
372             userAge_List.type = EN_OC_MQTT_PROFILE_VALUE_INT;
373             userAge_List.nxt = &deviceID_List;
374
375             deviceID_List.key = "deviceID";
376             deviceID_List.value = (char *)deviceID;
377             deviceID_List.type = EN_OC_MQTT_PROFILE_VALUE_STRING;
378             deviceID_List.nxt = &area_List;
379
380             area_List.key = "area";
381             area_List.value = (char *)area;
382             area_List.type = EN_OC_MQTT_PROFILE_VALUE_STRING;
383             area_List.nxt = &Region_List;
384
385             Region_Random = rand()%6;
386             Region_List.key = "region";
387             Region_List.value = (char *)Regions[Region_Random];
388             Region_List.type = EN_OC_MQTT_PROFILE_VALUE_STRING;
389             Region_List.nxt = &Longitude_List;
390
391             Longitude_List.key = "longitude";
392             Longitude_List.value = (char *)&longitude;
393             Longitude_List.type = EN_OC_MQTT_PROFILE_VALUE_INT;
394             Longitude_List.nxt = &Latitude_List;
395
396             Latitude_List.key = "latitude";
397             Latitude_List.value = (char *)&latitude;
398             Latitude_List.type = EN_OC_MQTT_PROFILE_VALUE_INT;
399             Latitude_List.nxt = &orderTime_List;
400
401             sprintf(orderTime_string,"%d%d",orderTime1,orderTime);
402             orderTime_List.key = "orderTime";
403             orderTime_List.value = (char *)orderTime_string;
404             orderTime_List.type = EN_OC_MQTT_PROFILE_VALUE_STRING;
405             orderTime_List.nxt = &Pay_List;
406
407             pay_Random = rand()%3;
408             Pay_List.key = "payment";
409             Pay_List.value = (char *)Pay[pay_Random];
410             Pay_List.type = EN_OC_MQTT_PROFILE_VALUE_STRING;
411             Pay_List.nxt = &Goods_Num_List[0];
```

```
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
if(goodsOptionX==X[1]){
    Status = 0;
    for(int i=0;i<10;i++){
        Goods_Num[i]=0;
    }
    Total_Cost = 0;
}else{
    Status = 1;
}

for(int i=0;i<10;i++){
    Goods_Num_List[i].key = Goods_Num_String[i];
    Goods_Num_List[i].value = (char *)&Goods_Num[i];
    Goods_Num_List[i].type = EN_OC_MQTT_PROFILE_VALUE_INT;
    Goods_Num_List[i].nxt = &Goods_Price_List[i];
    Goods_Price_List[i].key = Goods_Price_String[i];
    Goods_Price_List[i].value = (char *)&Goods_Price[i];
    Goods_Price_List[i].type = EN_OC_MQTT_PROFILE_VALUE_INT;
    if(i<9){
        Goods_Price_List[i].nxt = &Goods_Num_List[i+1];
    }
}
Goods_Price_List[9].nxt = &Status_List;

Status_List.key = "status";
Status_List.value = (char *)&Status;
Status_List.type = EN_OC_MQTT_PROFILE_VALUE_INT;
Status_List.nxt = &Total_Cost_List;

Total_Cost_List.key = "totalCost";
Total_Cost_List.value = (char *)&Total_Cost;
Total_Cost_List.type = EN_OC_MQTT_PROFILE_VALUE_INT;
Total_Cost_List.nxt = NULL;

}else {
    for(int i=0;i<10;i++){
```

步骤 4 添加数据上报代码

```
oc_mqtt_profile_propertyreport(NULL,&s_device_service);

printf("%s\r\n","My report success");

441
442
443
444
445
446
447
448
449
450
451
    Total_Cost_List.key = "totalCost";
    Total_Cost_List.value = (char *)&Total_Cost;
    Total_Cost_List.type = EN_OC_MQTT_PROFILE_VALUE_INT;
    Total_Cost_List.nxt = NULL;

    oc_mqtt_profile_propertyreport(NULL,&s_device_service);

    printf("%s\r\n","My report success");
.

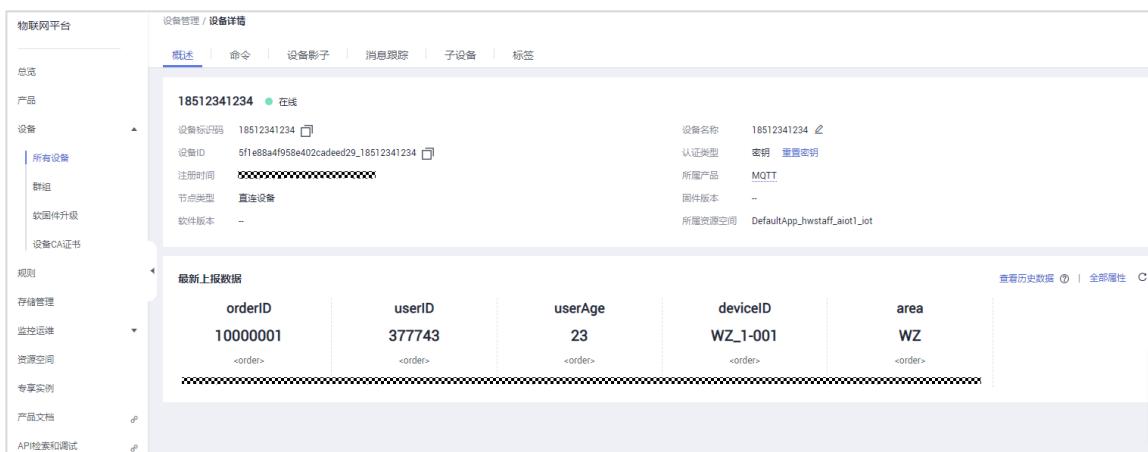
}else {
    for(int i=0;i<10;i++){
```

步骤 5 添加购物车清空功能代码

```
Total_Cost = 0;  
Total_Num = 0;  
orderID++;  
orderTime += 1000;  
for(int i=0;i<10;i++){  
    Goods_Num[i]=0;  
}  
LCD_ShowxNum(X[1], Y[6], Total_Num, 3, 24, 1);  
LCD_ShowString(goodsOptionX-10, goodsOptionY, 10, 24,  
24, " ");  
goodsOptionX = X[0];  
goodsOptionY = Y[0];  
LCD_ShowString(goodsOptionX-10, goodsOptionY, 10, 24,  
24, "**");
```

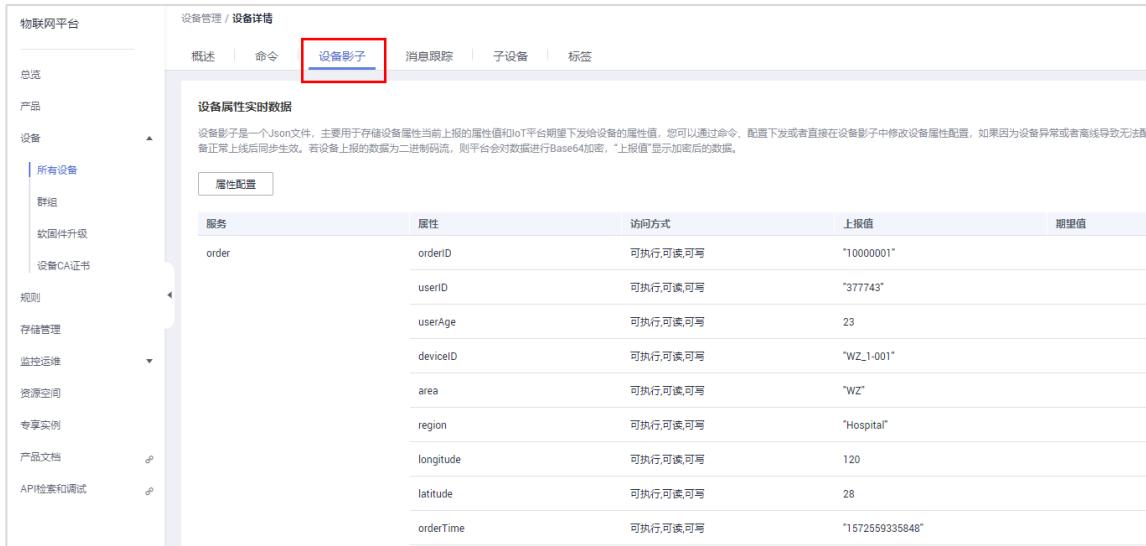
```
446     oc_mqtt_profile_propertyreport(NULL,&s_device_service);  
447  
448     printf("%s\r\n","My report success");  
449     Total_Cost = 0;  
450     Total_Num = 0;  
451     orderID++;  
452     orderTime += 1000;  
453     for(int i=0;i<10;i++){  
        Goods_Num[i]=0;  
    }  
    LCD_ShowxNum(X[1], Y[6], Total_Num, 3, 24, 1);  
    LCD_ShowString(goodsOptionX-10, goodsOptionY, 10, 24, 24, " ");  
    goodsOptionX = X[0];  
    goodsOptionY = Y[0];  
    LCD_ShowString(goodsOptionX-10, goodsOptionY, 10, 24, 24, "**");  
461 }  
462 }
```

编译烧录，打开 WIFI 热点，使用 F2 将商品添加购物车后，使用 F1 选择到“Submit”处，按 F2 提交订单，在物联网平台上刷新可以看到最新上报的订单信息。



| orderID | userID | userAge | deviceID | area |
|----------|--------|---------|----------|------|
| 10000001 | 377743 | 23 | WZ_1-001 | WZ |

在设备影子中可以看到详细数据；



设备属性实时数据

设备影子是一个JSON文件，主要用于存储设备属性当前上报的属性值和IoT平台期望下发给设备的属性值。您可以通过命令、配置下发或者直接在设备影子中修改设备属性配置，如果因为设备异常或者离线导致无法配置正常上线后同步生效。若设备上报的数据为二进制码流，则平台会对数据进行Base64加密，“上报值”显示加密后的数据。

| 服务 | 属性 | 访问方式 | 上报值 | 期望值 |
|-------|-----------|-----------|-----------------|-----|
| order | orderID | 可执行,可读,可写 | "10000001" | |
| | userID | 可执行,可读,可写 | "377743" | |
| | userAge | 可执行,可读,可写 | 23 | |
| | deviceID | 可执行,可读,可写 | "WZ_1-001" | |
| | area | 可执行,可读,可写 | "WZ" | |
| | region | 可执行,可读,可写 | "Hospital" | |
| | longitude | 可执行,可读,可写 | 120 | |
| | latitude | 可执行,可读,可写 | 28 | |
| | orderTime | 可执行,可读,可写 | "1572559335848" | |

13.2.5 下发命令实验

步骤 1 设备侧添加命令处理代码

在 oc_cmd_normal 函数中添加取出命令内容的代码；

```
char temp[20];
cJSON *cmd_value = NULL;
cJSON *paras = NULL;
paras = cJSON_GetObjectItem(cJSON_Parse((char *)demo_msg->msg),"paras");
cmd_value = cJSON_GetObjectItem(paras,"cmd_value");

224 /*下发命令处理函数*/
225 static int oc_cmd_normal(oc_mqtt_profile_msgrcv_t *demo_msg)
226 {
227     static int value = 0;
228     oc_mqtt_profile_cmdresp_t cmdresp;
229     oc_mqtt_profile_propertysetresp_t propertysetresp;
230     oc_mqtt_profile_propertygetresp_t propertygetresp;
231
232     printf("DEALMSG:type:%d requestID:%s payloadlen:%d payload:%s\n\r",
233           demo_msg->type,demo_msg->request_id==NULL?"NULL":demo_msg->request_id,
234           demo_msg->msg_len,(char *)demo_msg->msg);
235     char temp[20];
236     cJSON *cmd_value = NULL;
237     cJSON *paras = NULL;
238     paras = cJSON_GetObjectItem(cJSON_Parse((char *)demo_msg->msg),"paras");
239     cmd_value = cJSON_GetObjectItem(paras,"cmd_value");
240
241     switch(demo_msg->type)
242     {
```

添加命令处理的代码；

```
sprintf(temp,"%s",cmd_value->valuestring);
for(int i=0;i<10;i++){
    goods_position[i] = temp[i]-'0';
}
LCD_Clear(BLACK);
POINT_COLOR = GREEN;
for(int i=0;i<10;i++){
    if(i<5){
        LCD_ShowString(X[0], Y[i], 240, 24, 24,
goodsView[goods_position[i]]);
    }else{
        LCD_ShowString(X[1], Y[i-5], 240, 24, 24,
goodsView[goods_position[i]]);
```

```
        }
    }

LCD_ShowString(X[0], Y[5], 240, 24, 24, Submit_View[0]);
LCD_ShowString(X[1], Y[5], 240, 24, 24, Submit_View[1]);

LCD_ShowString(X[0]-10, goodsOptionY, 10, 24, 24, "*");
LCD_ShowxNum(X[1], Y[6], 0, 3, 24, 1);
```

```
241     switch(demo_msg->type)
242     {
243         case EN_OC_MQTT_PROFILE_MSG_TYPE_DOWN_MSGDOWN:
244             //;< add your own deal here
245             break;
246         case EN_OC_MQTT_PROFILE_MSG_TYPE_DOWN_COMMANDS:
247             //;< add your own deal here
248             sprintf(temp,"%s",cmd_value->valuestring);
249             for(int i=0;i<10;i++){
250                 goods_position[i] = temp[i]-'0';
251             }
252             LCD_Clear(BLACK);
253             POINT_COLOR = GREEN;
254             for(int i=0;i<10;i++){
255                 if(i<5){
256                     LCD_ShowString(X[0], Y[i], 240, 24, 24, goodsView[goods_position[i]]);
257                 }else{
258                     LCD_ShowString(X[1], Y[i-5], 240, 24, 24, goodsView[goods_position[i]]);
259                 }
260             }
261             LCD_ShowString(X[0], Y[5], 240, 24, 24, Submit_View[0]);
262             LCD_ShowString(X[1], Y[5], 240, 24, 24, Submit_View[1]);

263             LCD_ShowString(X[0]-10, goodsOptionY, 10, 24, 24, "*");
264             LCD_ShowxNum(X[1], Y[6], 0, 3, 24, 1);

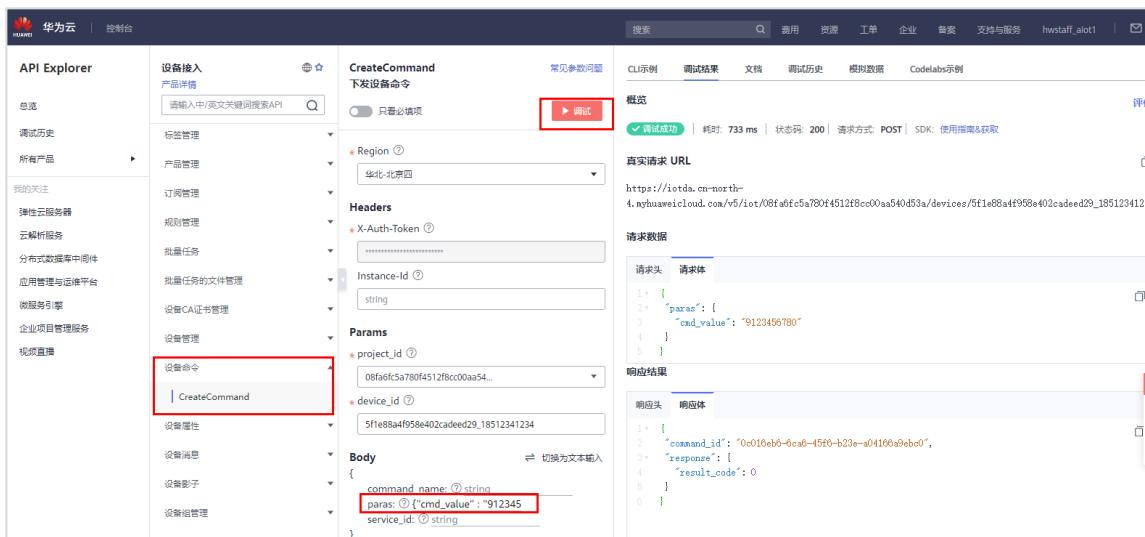
265             //;< do the response
266             cmdresp.paras = NULL;
267             cmdresp.request_id = demo_msg->request_id;
```

编译烧录，等待提示烧录成功，打开 WIFI 热点。

打开 API Explorer，打开设备命令接口；

填写参数 paras 为{"cmd_value" : "9123456780"}，点击调试；

其中数字为商品顺序，可以自定义；



The screenshot shows the Huawei Cloud API Explorer interface. On the left, there's a sidebar with various service categories like Device Access, Tag Management, Product Management, etc. Under 'Device Access', '设备命令' (Device Command) is selected, and the 'CreateCommand' sub-item is highlighted with a red box. At the top right, there's a toolbar with search, document, history, and other developer tools. Below the toolbar, it says '调试成功' (Test successful), '耗时: 733 ms | 状态码: 200 | 请求方式: POST | SDK: 使用指南&获取'. The main area is divided into '真实请求 URL' (Real Request URL) and '请求数据' (Request Data). '请求头' (Request Headers) includes 'X-Auth-Token' and 'Instance-Id'. '参数' (Parameters) include 'project_id' and 'device_id'. 'Body' contains 'command_name' and 'params' (with 'cmd_value' set to '912345'). The '响应结果' (Response Result) shows a JSON object with 'command_id', 'response' (containing 'result_code': 0), and 'service_id'.

13.3 练习题

13.3.1 添加售货机颜色字段进行上报

14 综合训练 2

14.1 实验介绍

14.1.1 关于本实验

本实验分为自主训练，根据前面实验所用到的端云互通方法，实现基于 MQTT 协议的智慧农业案例。

14.1.2 实验目的

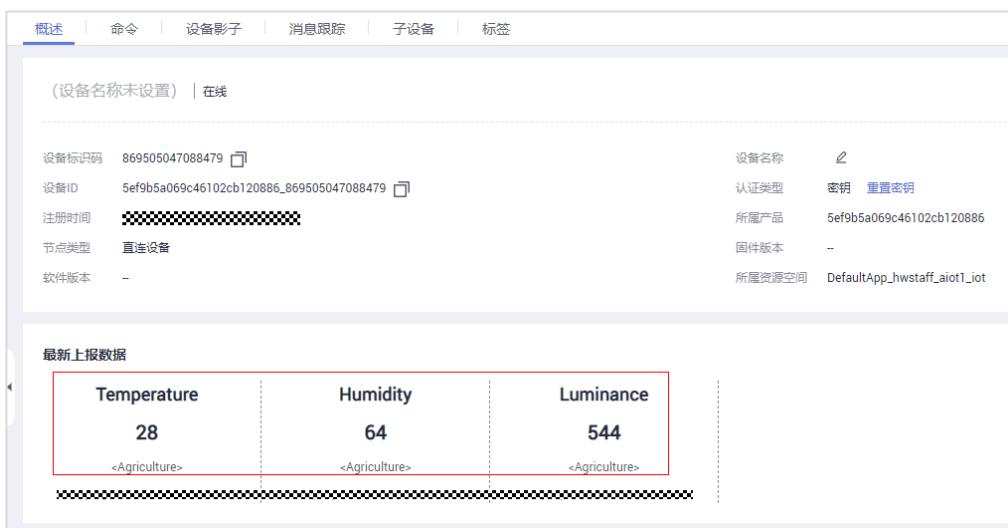
- 掌握如何利用物联网平台和 Huawei LiteOS 物联网操作系统实现不同协议的端云互通

14.2 实验任务配置步骤

14.2.1 基于 MQTT 协议的智慧农业案例实验

14.3 结果验证

基于 MQTT 协议，实现智慧农业案例的温湿度数据采集以及电机控制和开关灯



14.4 思考题

14.4.1 大数据与人工智能结合

步骤 1 将华为云物联网平台自动售货机数据转发到 DIS 服务

步骤 2 将 DIS 数据转发到 OBS

步骤 3 大数据服务 MapReduce 从 OBS 同步数据并进行分析

步骤 4 人工智能服务 ModelArts 从 OBS 同步数据并进行模型训练