

# HarmonyOS设备开发实践



# 目标

---

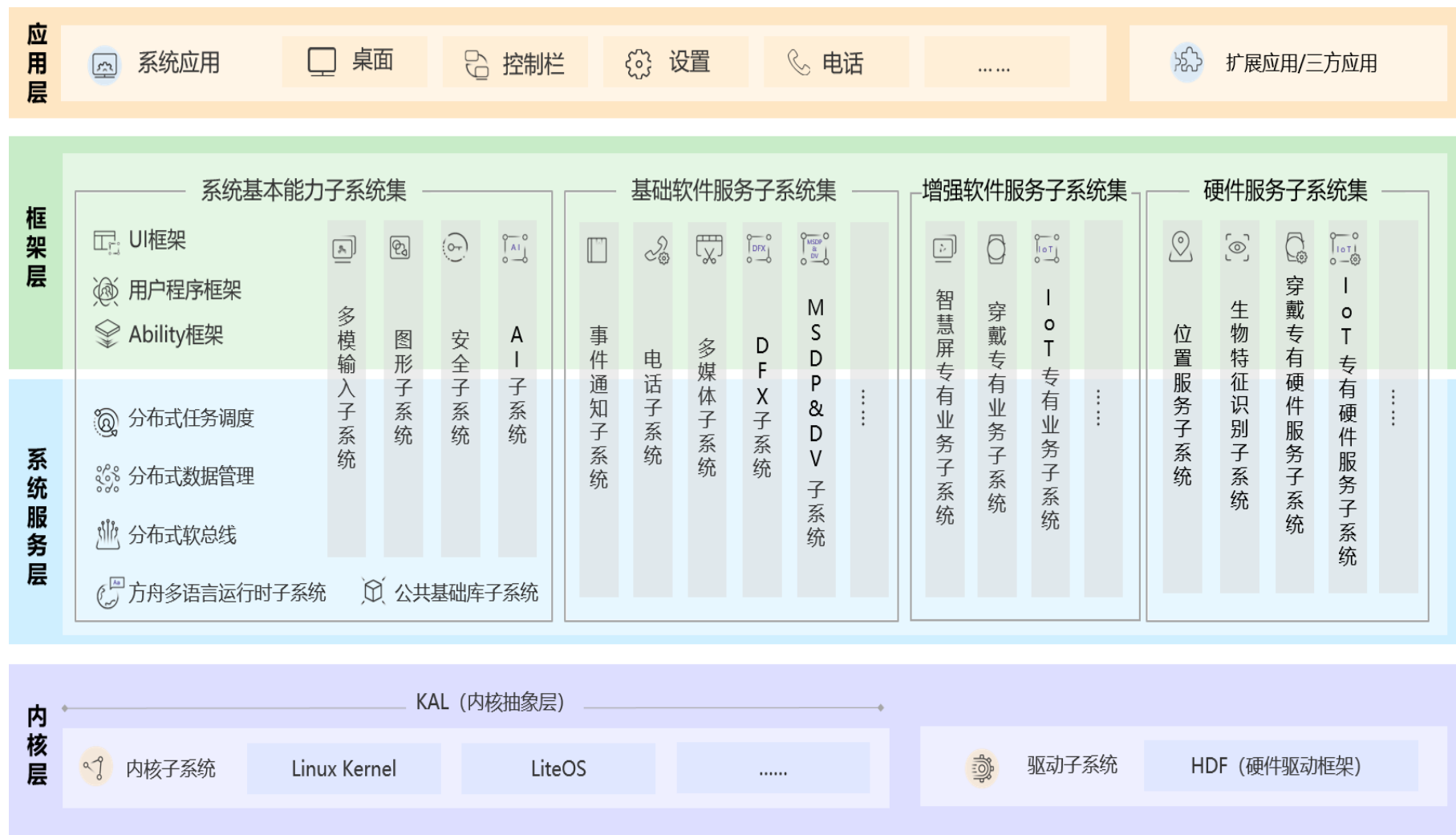
- 学完本课程后，您将能够：
  - 了解HarmonyOS系统的技术架构
  - 典型设备HarmonyOS系统开发实战

# 目录

---

1. 设备开发基础知识
2. WIFI模组-Hi3861开发板介绍
3. 软总线概述
4. 开发实战

# HarmonyOS系统技术架构



面向全场景的开源分布式操作系统

分布式

- HarmonyOS应用/服务在不同设备的流转
- 硬件虚拟化和池化

一次开发,多端部署

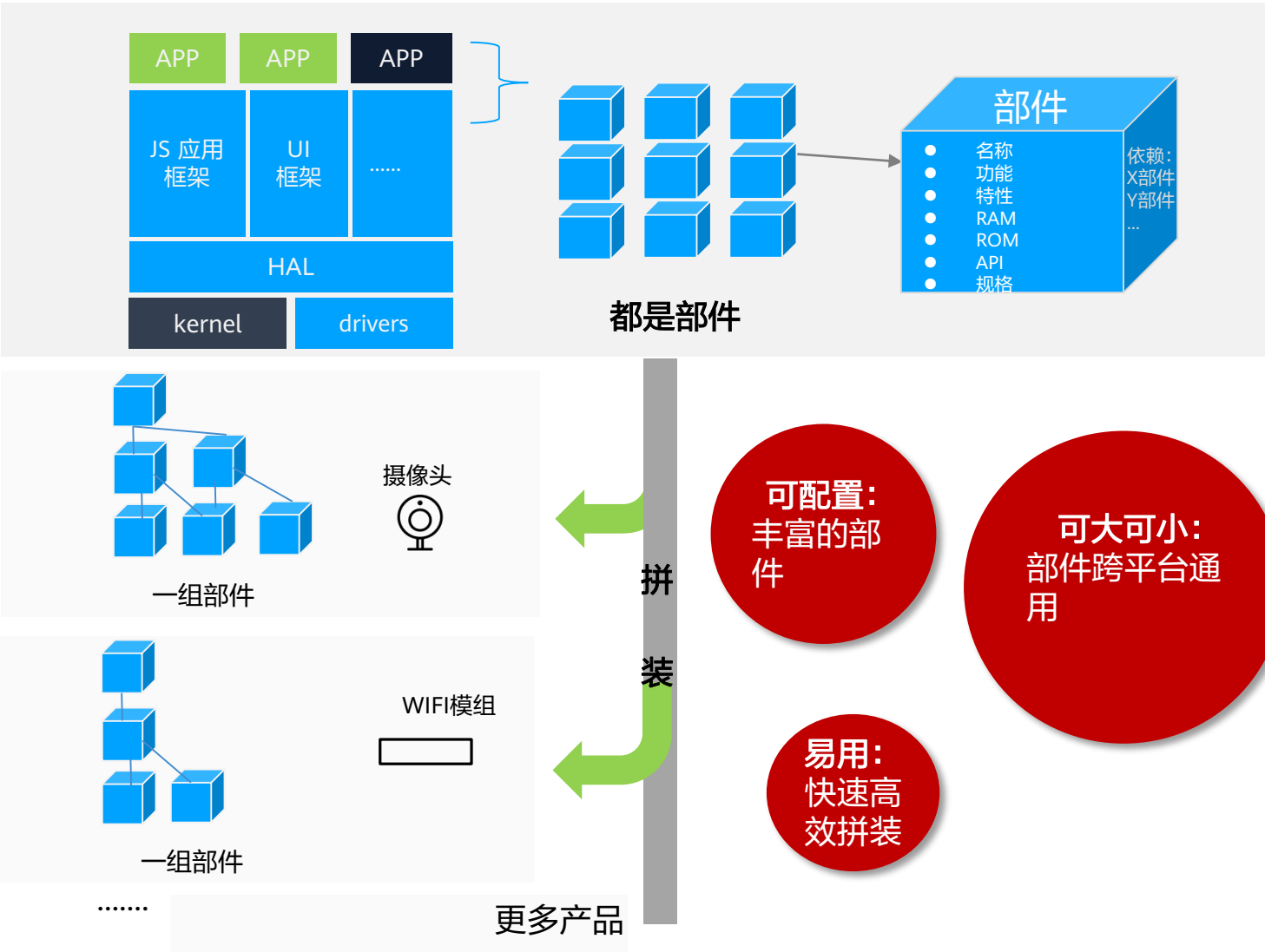
- 多终端软件平台API具备一致性,确保用户程序的运行兼容性

统一OS,弹性部署

- 系统组件化,积木化的弹性部署



# HarmonyOS弹性部署 - 部件化



## OpenHarmony 1.1.0

The screenshot shows the OpenHarmony 1.1.0 GitHub repository page. The repository is owned by OpenHarmony and is a GIP repository. The description states that OpenHarmony is an open-source project under the OpenAtom Foundation, designed as a fully open-source distributed operating system. The first version supports 128K-128M devices. The page shows 119 repositories, 391 tasks, 231 pull requests, and 94 members. A yellow banner at the top right indicates that the community plans to adjust code and CI/CD infrastructure for the 1.1.0 version, with a release date of May 31st. Below the banner, there is a list of featured components:

- kernel\_liteos\_a**: LiteOS kernel for embedded devices with rich resources | 适用于资源丰富的嵌入式设备的LiteOS内核
- communication\_softbus\_lite**: Implementation code for virtual bus discovery, networking, and transmission | 软总线发现、组网、传输功能实现
- docs**: OpenHarmony documentation | OpenHarmony开发文档
- ace\_engine\_lite**: ACE JS lite framework | 轻量级JS核心开发框架
- distributedschedule\_samgr\_lite**: System manager service framework | 系统能力管理框架
- community**: OpenHarmony community governance, developer contribution guide, contribution agreement, and community ...

总仓: 119个  
部件: 87个

即将开源的部件: 180+

# HarmonyOS系统开发的通用开发流程&实验手册回顾

- 开发环境搭建
  - 代码获取
  - 系统开发/编译/调试/镜像生成
- 
- 通过HarmonyOS设备开发环境搭建指导手册我们已经具备以上的能力，现场通过开发板我们继续学习：
    - 镜像烧录
    - 验证系统

# 开发环境搭建



搭建过程详见 [https://device.harmonyos.com/cn/docs/start/introduce/oem\\_build\\_os\\_env-0000001121152509](https://device.harmonyos.com/cn/docs/start/introduce/oem_build_os_env-0000001121152509)

Linux编译服务器	
开发工具	用途
Python3.7+	编译构建工具
gn	产生ninja编译脚本
ninja	执行ninja编译脚本
LLVM	编译工具链
hb	HarmonyOS编译构建命令行工具

Windows工作台	
开发工具	用途
Visual Studio Code	代码编辑工具。
HUAWEI DevEco Device Tool	IDE开发工具，支持WLAN模组的代码编写、远程编译、版本烧录、串口调试等功能。HarmonyOS面向智能设备开发者提供的一站式集成开发环境，支持HarmonyOS的组件按需定制，支持C/C++语言，以插件的形式部署在Visual Studio Code上。

实验手册中对应的方式：Linux编译服务器与Windows工作平台合并，通过在Windows上加载已包含所有Linux开发工具的docker镜像方式 在虚拟机中运行Linux编译环境

# HarmonyOS系统OS源码下载

- OS 源码下载、编译详见：<https://gitee.com/openharmony/docs/blob/master/zh-cn/device-dev/get-code/%E6%BA%90%E7%A0%81%E8%8E%B7%E5%8F%96.md>
- 如果没有远程终端工具(SecureCRT,putty等)，安装putty，安装包路径：`//192.168.0.2/tools`
- 选择ssh,连接现场的Linux开发服务器 192.168.0.3，各组账号均为test{组号}，例如test1,密码均为123456
- 进入各个小组对应账号的工作目录 `cd ~`
- 在工作目录新建一个目录 `mkdir HarmonyOS`
- 复制一份OS源码到工作目录 `cp /public/OpenHarmony_release_v1.1.0.tgz ~`
- 解压到代码目录 `tar -xzvf OpenHarmony_release_v1.1.0.tgz ./HarmonyOS`



# HarmonyOS系统代码结构 (1)

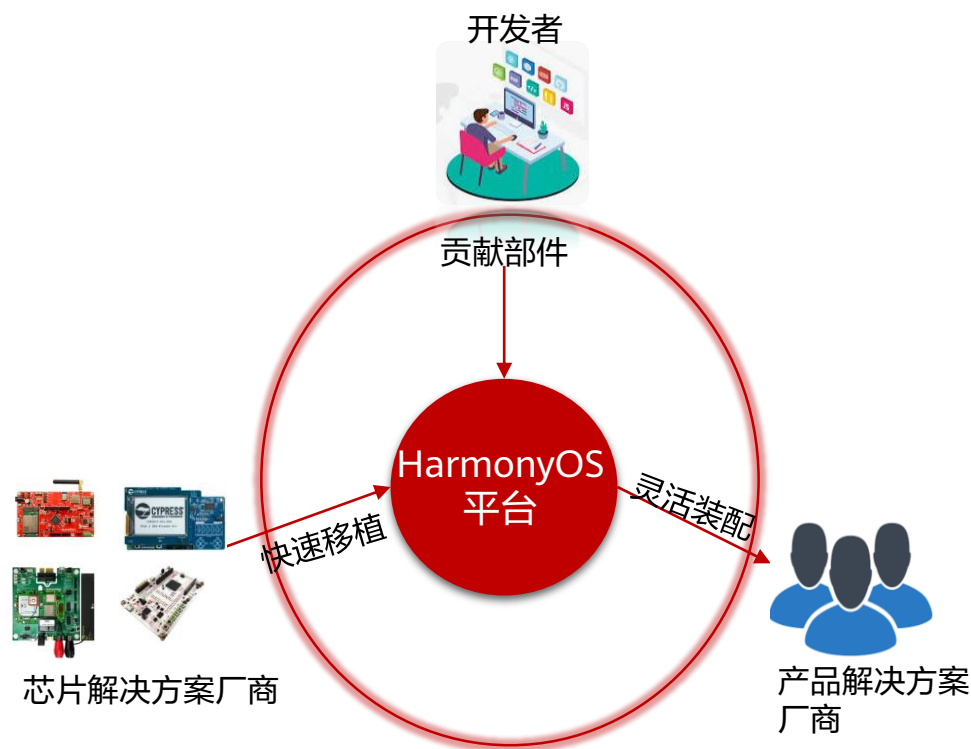
- OS系统按照“系统 > 子系统 > 功能/模块”逐级展开，在多设备部署场景下，支持根据实际需求裁剪某些非必要的子系统或功能/模块。

- kernel目录-内核子系统：采用多内核（Linux内核或者LiteOS）设计，支持针对不同资源受限设备选用适合的OS内核。
- drivers目录-驱动子系统：驱动框架（HDF）是系统硬件生态开放的基础，提供统一外设访问能力和驱动开发、管理框架。
- base目录-基础软件服务子系统集
- foundation目录-系统基本能力子系统集
- domains:增强软件服务子系统集
- build目录：编译框架子系统

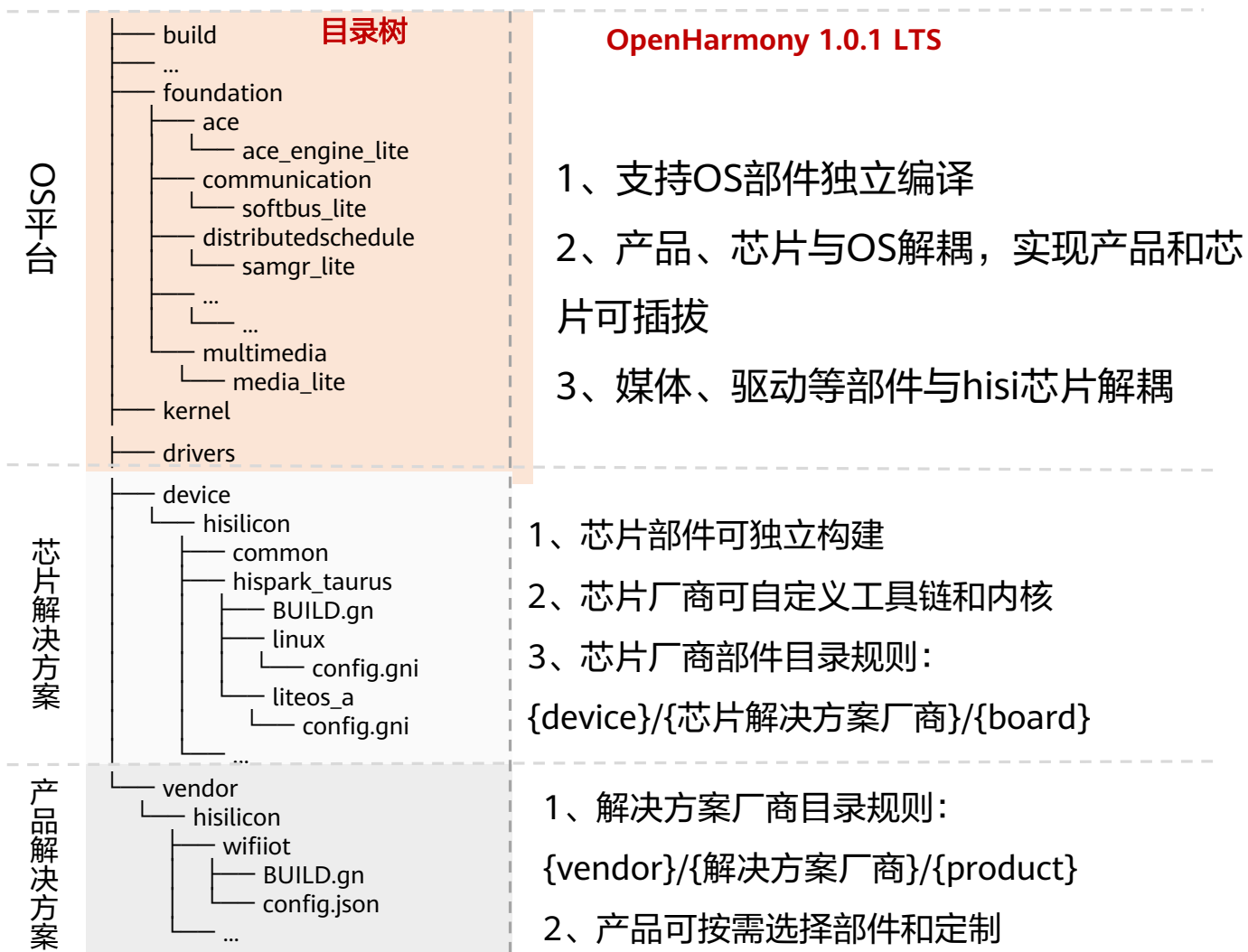
```
$ ll
total 88
drwxrwxr-x 21 4096 May  8 19:17 .
drwxrwxr-x  4 4096 May  8 19:02 ..
drwxrwxr-x  4 4096 May  8 19:08 applications
drwxrwxr-x 11 4096 May  8 19:09 base
drwxrwxr-x 28 4096 May  8 19:09 build
lrwxrwxrwx  1      19 May  8 19:09 build.py -> build/lite/build.py
drwxrwxr-x  3 4096 May  8 19:09 deceptools
drwxrwxr-x 14 4096 May  8 19:09 device
drwxrwxr-x  3 4096 May  8 19:09 docs
drwxrwxr-x  3 4096 May  8 19:09 domains
drwxrwxr-x  6 4096 May  8 19:09 drivers
drwxrwxr-x 10 4096 May  8 19:09 foundation
drwxrwxr-x  3 4096 May  8 19:09 ide
drwxrwxr-x  6 4096 May  8 19:09 kernel
-rw-rw-r--  1    370 May 13 20:42 ohos_config.json
drwxrwxr-x  4 4096 May 10 10:03 out
drwxrwxr-x  8 4096 May  8 19:10 prebuilts
drwxrwxr-x  7 4096 May  8 19:08 .repo
drwxrwxr-x  5 4096 May  8 19:10 test
drwxrwxr-x 44 4096 May  8 19:11 third_party
drwxrwxr-x  4 4096 May  8 19:11 tools
drwxrwxr-x  3 4096 May  8 19:11 utils
drwxrwxr-x  5 4096 May  8 19:11 vendor
```

注：该项目源码基于OpenHarmony 1.1.0 LTS版本

# HarmonyOS系统代码结构 (2)



OS平台、芯片解决方案、产品解决方案解耦，支持三方芯片快速移植和产品化。



# HarmonyOS系统开发实战

- WIFI模组
  - 内存百K级别物联网，智能家居的必不可少的联接模块。
  - 体验嵌入式开发
  - 介绍软总线，体验系统级的分布式能力
- 智能网络摄像头
  - 内存百M级别的物联网设备，应用在家用摄像头，电子门铃，行车记录仪等等
  - 介绍HarmonyOS应用框架，体验如何在IOT设备上开发HarmonyOS应用
  - 体验硬件虚拟化

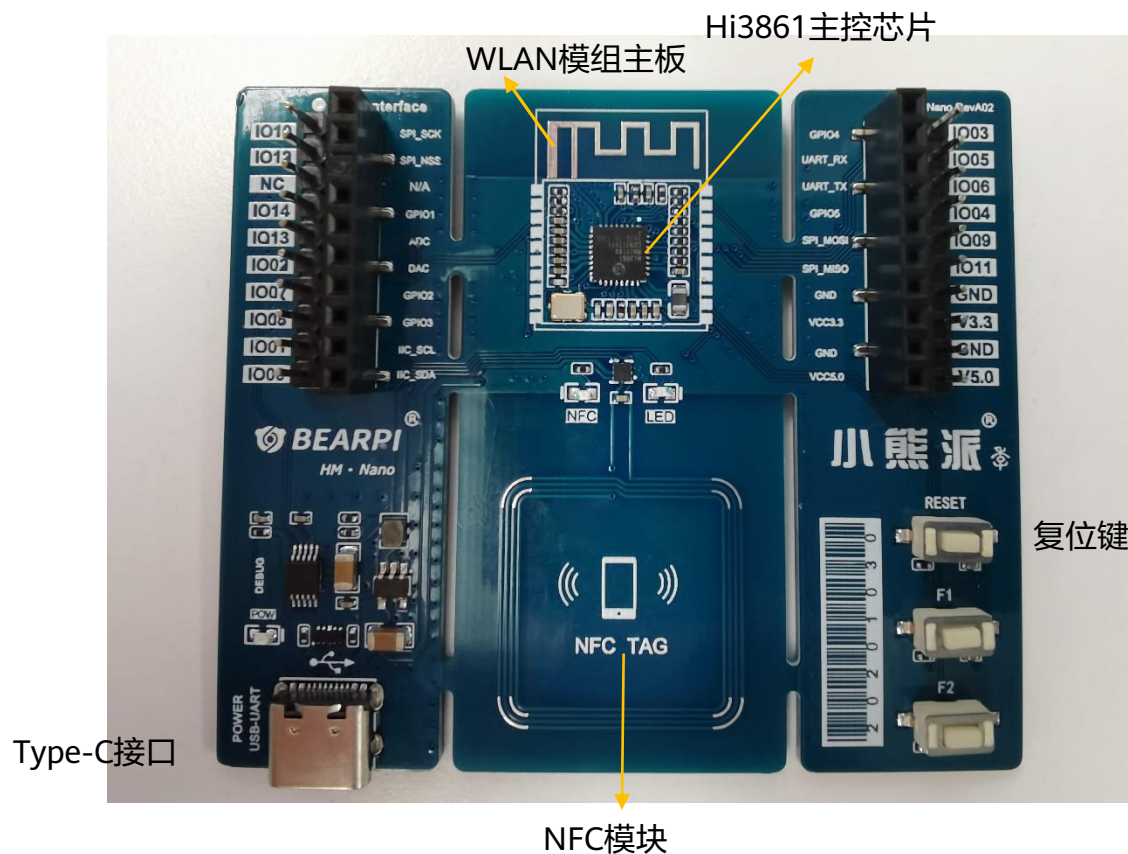
# 目录

---

1. 设备开发基础知识
- 2. WIFI模组-Hi3861开发板介绍**
3. 软总线概述
4. 开发实战

# Hi3861开发版本介绍 (1)

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



## Hi3861开发版本介绍 (2)

- 使用AT指令，在deveco工具中，配置Hi3861，进行WiFi联网
- 复位WLAN模组，终端界面显示“ready to OS start”，则启动成功
- 在DevEco的串口终端中，依次执行如下AT命令，启动STA模式，连接指定AP热点，并开启DHCP功能

AT+STARTSTA	# 启动STA模式
AT+SCAN	# 扫描周边AP
AT+SCANRESULT	# 显示扫描结果
AT+CONN="SSID",,2,"PASSWORD"	# 连接指定AP，其中SSID/PASSWORD为待连接的热点名称和密码
AT+STASTAT	# 查看连接结果
AT+DHCP=wlan0,1	# 通过DHCP向AP请求wlan0的IP地址



# Hi3861开发版本介绍 (3)

- 查看WLAN模组与网关联通是否正常，如下图所示

AT+IFCFG

# 查看模组接口IP

AT+PING=X.X.X.X

# 检查模组与网关的联通性，其中X.X.X.X需替换为实际的网关地址

```
PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL 1: Task - deveco: uploac + [ ] [ ] ^ x

AT+IFCFG
+IFCFG:wlan0,ip=192.168.43.233,netmask=255.255.255.0,gateway=192.168.43.1,ip6=FE80::B6C9:B9FF:FE61:9961,ip6=2409:8954:3450:B6C9:B9FF:FE61:9961,Hwaddr=,MTU=1500,Link
Status=1,RunStatus=1
+IFCFG:lo,ip=127.0.0.1,netmask=255.0.0.0,gateway=127.0.0.1,ip6::1,Hwaddr=00,MTU=16436,LinkStatus=1,RunStatus=1
OK

AT+PING=192.168.43.1
+PING:

[0]Reply from 192.168.43.1:time=10ms TTL=64
[1]Reply from 192.168.43.1:time=13ms TTL=64
[2]Reply from 192.168.43.1:time=12ms TTL=64
[3]Reply from 192.168.43.1:time=17ms TTL=64
4 packets transmitted, 4 received, 0 loss, rtt min/avg/max = 10/13/17 ms

OK
```

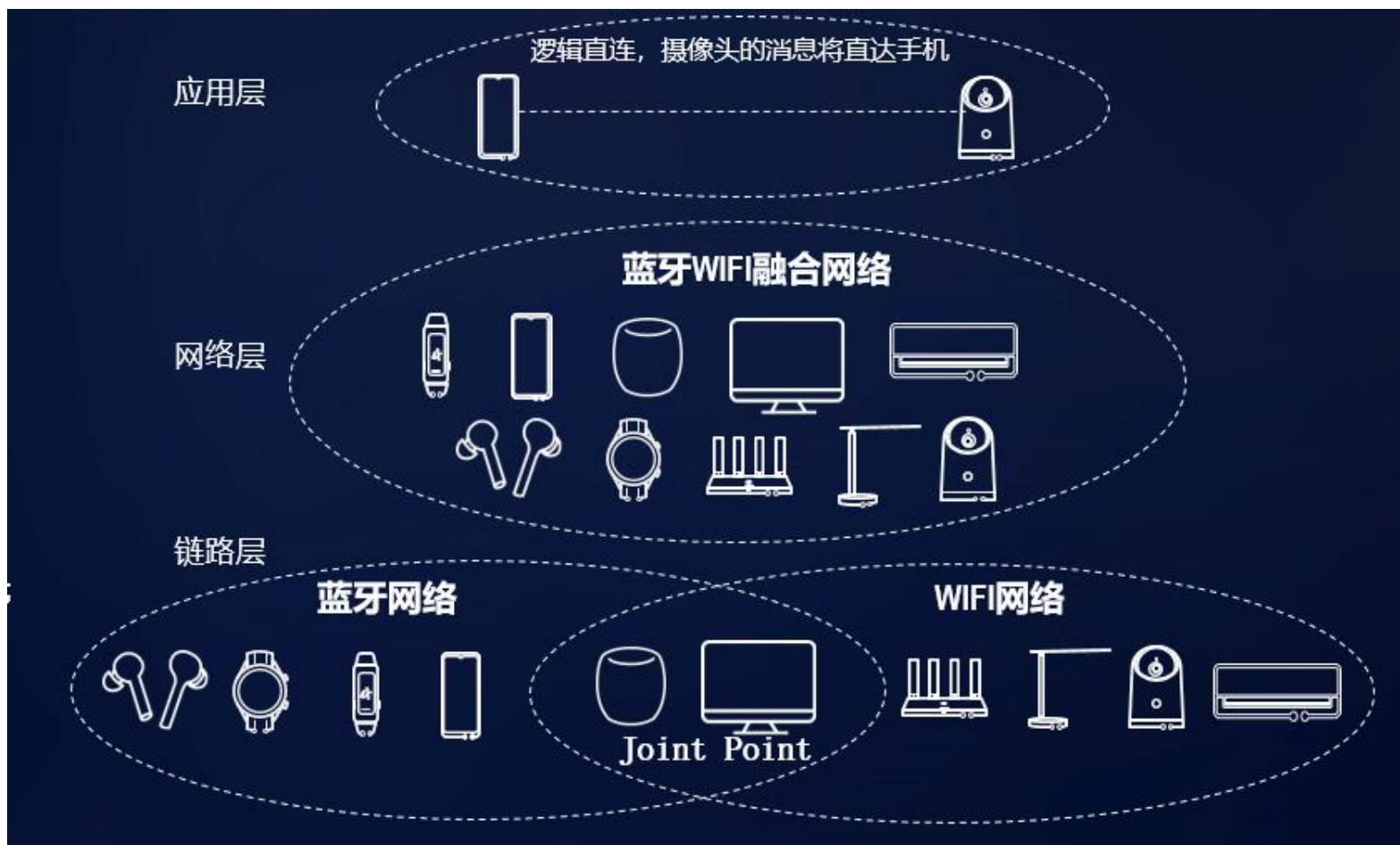
# 目录

---

1. 设备开发基础知识
2. WIFI模组-Hi3861开发板介绍
- 3. 软总线概述**
4. 开发实战

# 软总线 - 异构网络组网

- 目标：自动构建一个逻辑全连接网络，业务开发者无需关心组网方式与物理协议



# 软总线 - 逻辑架构

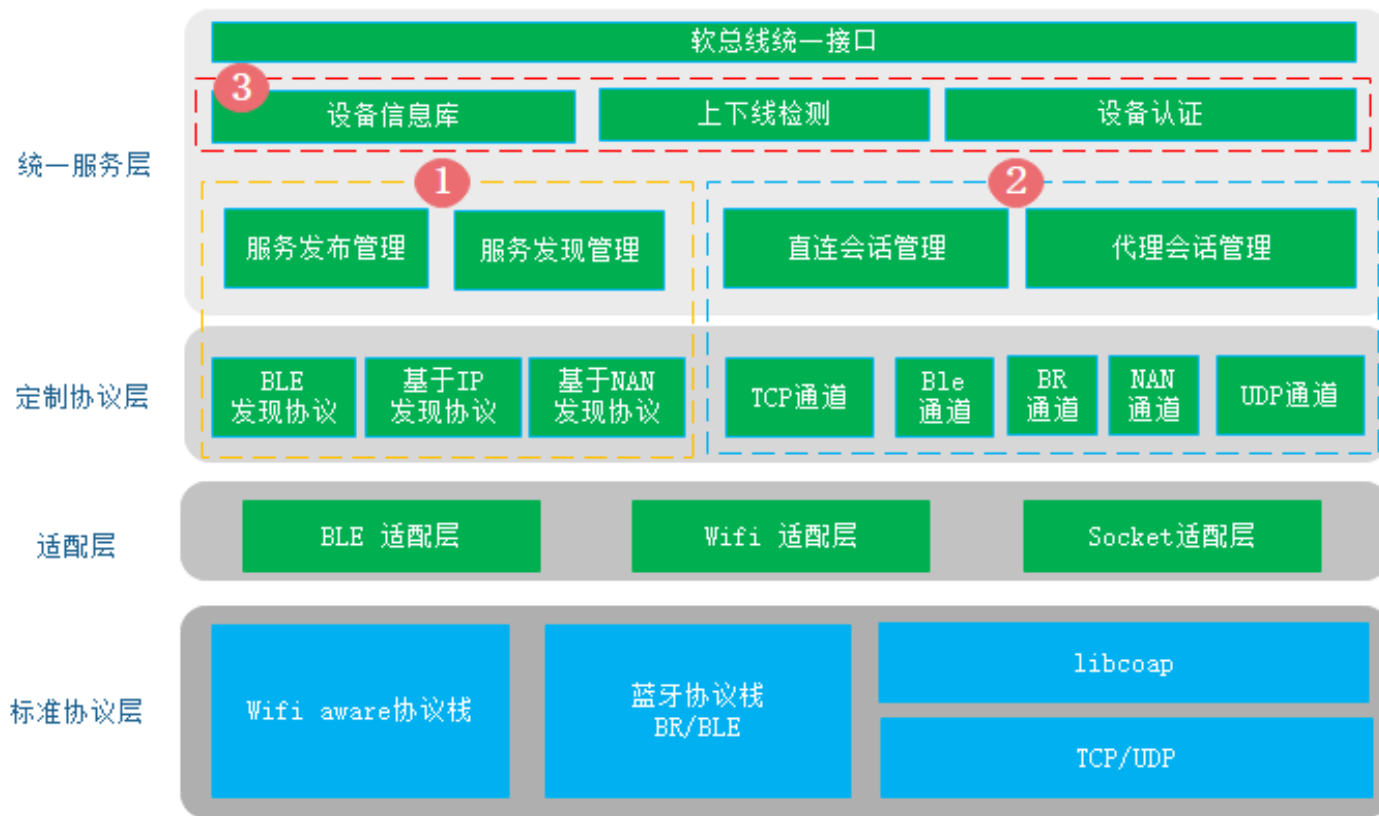
- 从逻辑层次分为三个层次：

- 服务层
- 定制协议层
- 协议适配层
- 标准协议层

- 从功能划分为：

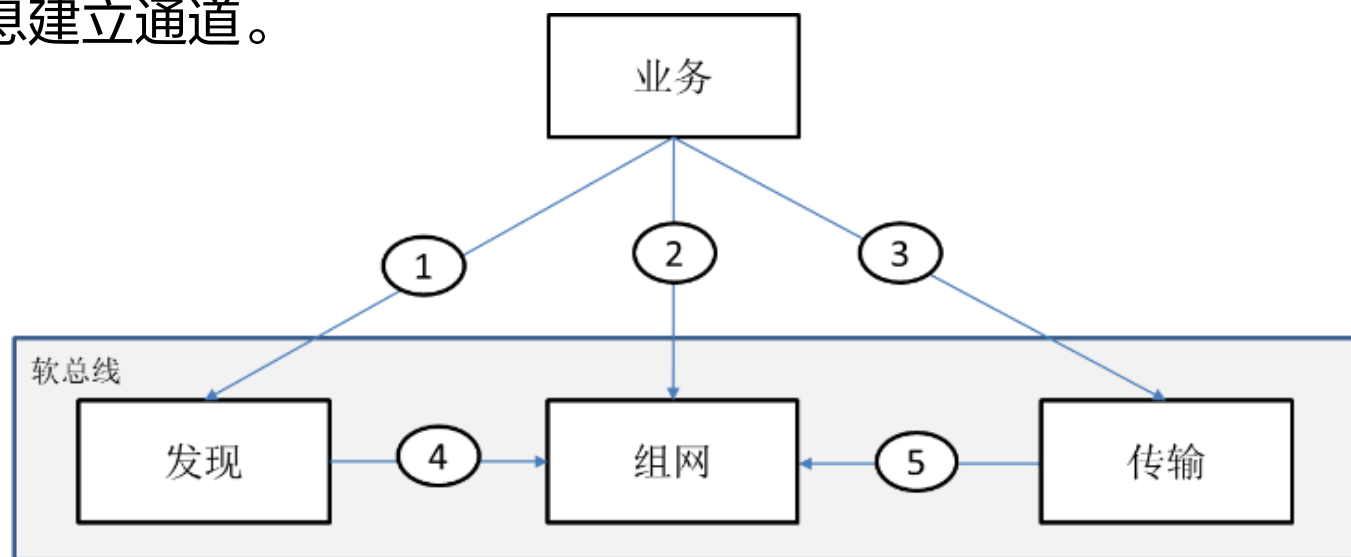
- 发现
- 传输
- 组网

- 统一接口，软总线对外提供不区分链路的一致接口。



# 软总线 - 交互模型

- 业务通过发现接口获取未认证设备列表。（通信地址）
- 业务通过组网接口获取认证设备列表。（组网ID）
- 业务使用1，2获取的组网ID（openSession加密），调用传输接口进行数据的收发。
- 发现的设备，认证过后，加入组网节点列表。
- 传输从组网查询路由，通信地址等信息建立通道。



# 目录

---

1. 设备开发基础知识
2. WIFI模组-Hi3861开发板介绍
3. 软总线概述
- 4. 开发实战**

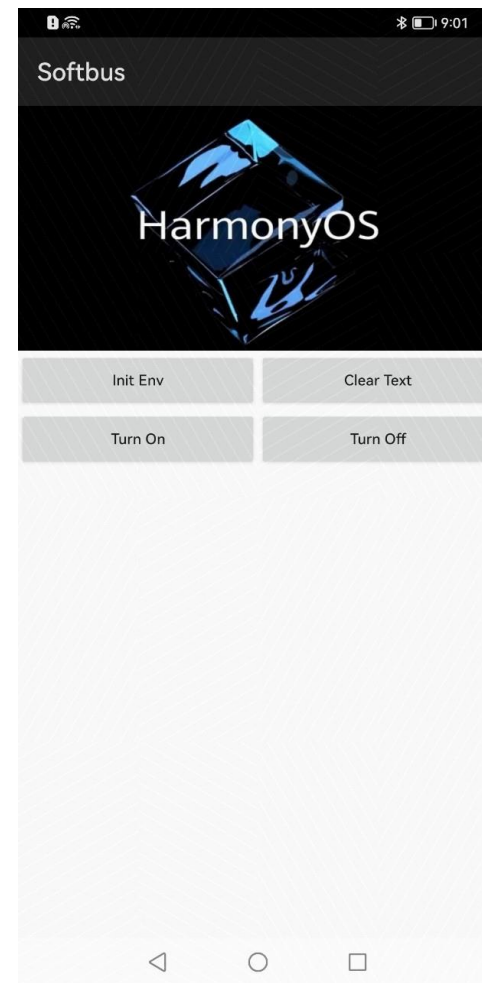
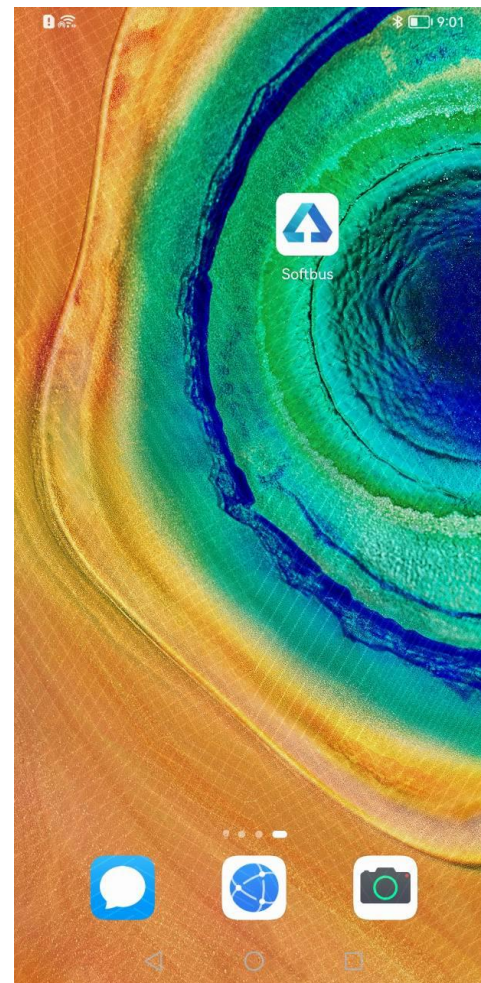


# 开发实战 - 基于软总线的3861设备点灯控制

- 应用场景：
  - 3861设备与手机在同一个局域网内，手机端APP通过软总线发现设备，与设备建立建立信任关系，创建会话，基于会话实现手机与设备之间的安全数据传输；APP通过软总线提供的数据传输通道，控制3861指示灯亮、灭行为。
- 手机端：安装APP，APP基于软总线实现与3861通信，控制设备。
- 设备端：实现业务代码，同样基于软总线与手机端APP通信，并根据手机APP指令控制3861上LED亮灭行为。

# 开发实战 - 手机端

- 安装APP，APP支持以下功能：
  - 软总线初始化，完成与3861设备绑定；
  - 创建会话，实现与3861设备通信，打开3861设备指示灯；
  - 创建会话，实现与3861设备通信，关闭3861设备指示灯。
- 注：APP需要有系统权限才能使用软总线功，故手机需要有root权限，修改系统应用。

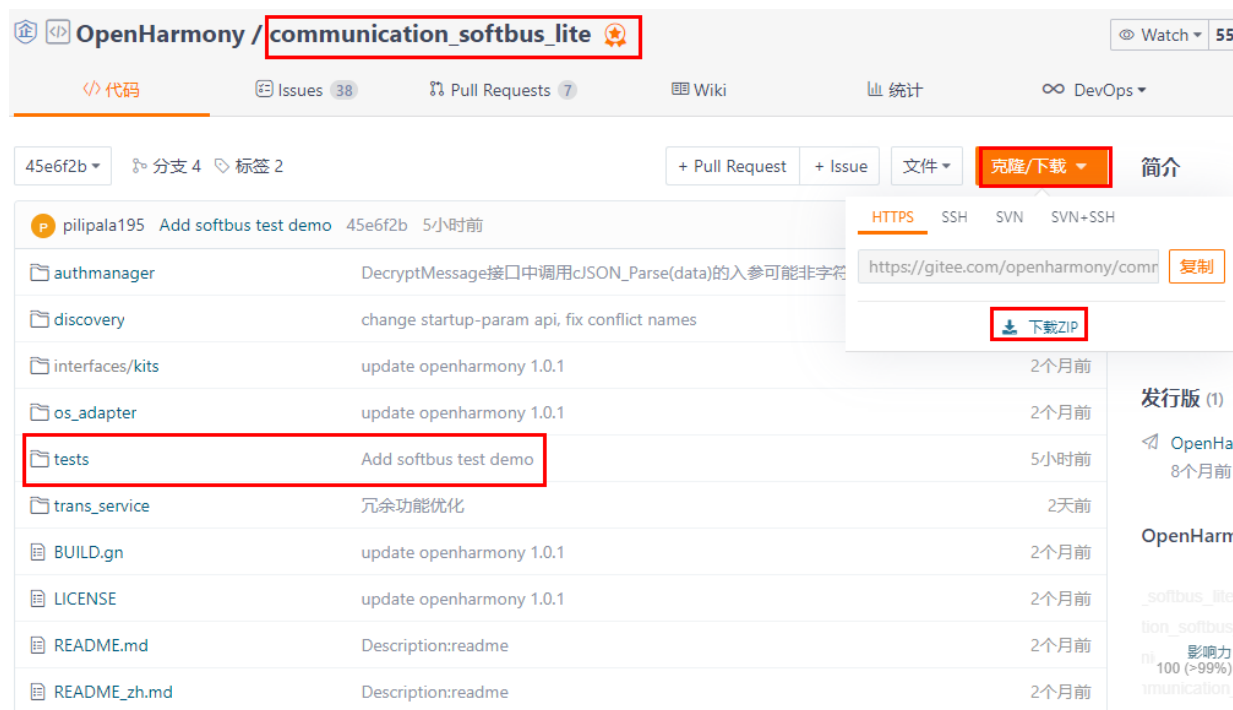
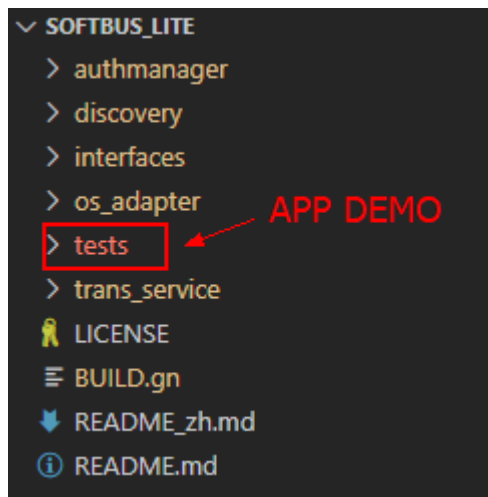


# 开发实战 - 3861设备端

- 软总线中增加业务代码，基于软总线实现与手机端APP交付。
- 业务代码主要支持以下功能：
  - 3861设备网络配置，加入指定网络
  - 服务发布，支持3861设备被手机发现
  - 创建会话服务，并指定会话相关回调
  - 根据手机端APP的控制命令，完成对3861设备指示灯行为控制

# 样例代码下载

- 业务代码已提交到码云，可以直接下载放到指定目录下编译即可，如下左图中tests目录代码。
- 下载链接：
  - [https://gitee.com/openharmony/communication\\_softbus\\_lite/tree/45e6f2b3e14e059cb567eb7c8e9f61a78c7fee6e](https://gitee.com/openharmony/communication_softbus_lite/tree/45e6f2b3e14e059cb567eb7c8e9f61a78c7fee6e)



# WIFI模组设备系统开发 - 软总线网路配置

1、使能wifi服务，并注册wifi事件回调。

2、连接到指定热点，代码中指定ssid和pwd

```
static int WadStaStart(void)
{
    WifiErrorCode error;
    error = EnableWifi();
    if (error != WIFI_SUCCESS) {
        printf("enablewifi failed, error = %d\n", error);
        return -1;
    }
    g_eventHandler.OnWifiConnectionChanged = WadWifiConnectionChangedHandler;
    error = RegisterWifiEvent(&g_eventHandler);
    if (error != WIFI_SUCCESS) {
        printf("RegisterWifiEvent failed, error = %d\n", error);
        return -1;
    }
    if (IsWifiActive() == 0) {
        printf("Wifi station is not active\n");
        return -1;
    }
    g_lwipStaNetif = netif_find("wlan0");
    if (g_lwipStaNetif == NULL) {
        printf("get netif failed\n");
        return -1;
    }
    return 0;
}
```

```
static int WadWapStaConnect(const char *ssid, const char *pwd)
{
    int netId = 0;
    WifiErrorCode error;
    WifiDeviceConfig config = {0};

    config.securityType = WIFI_SEC_TYPE_PSK;
    if (strcpy_s(config.ssid, sizeof(config.ssid), ssid) != 0) {
        return -1;
    }
    if (strcpy_s(config.preSharedKey, sizeof(config.preSharedKey), pwd) != 0) {
        return -1;
    }

    error = AddDeviceConfig(&config, &netId);
    if (error != WIFI_SUCCESS) {
        return -1;
    }

    error = ConnectTo(netId);
    if (error != WIFI_SUCCESS) {
        return -1;
    }

    return 0;
}
```

注：详细实现参见码云中提交。

# WIFI模组设备系统开发 - 软总线服务发布

## 1、定义服务发布结果通知回调

```
void OnSuccess(int publishId)
{
    ... (void)publishId;
}

void OnFail(int publishId, PublishFailReason reason)
{
    ... (void)publishId;
    ... (void)reason;
}

static IPublishCallback g_publishCallback = {
    ... .onPublishSuccess = OnSuccess,
    ... .onPublishFail = OnFail,
};
```

## 2、定义服务信息，发布服务，支持设备被手机发现

```
static PublishInfo g_publishInfo = {
    ... .capabilityData = (unsigned char *)"1",
    ... .capability = "dumpCapability",
    ... .dataLen = 1,
    ... .publishId = 1,
    ... .mode = DISCOVER_MODE_ACTIVE,
    ... .medium = COAP,
    ... .freq = MID,
};

int ret = PublishService(g_demoModuleName, &g_publishInfo, &g_publishCallback);
if (ret != 0) {
    ... printf("PublishService err");
    ... return;
}
```



# WIFI模组设备系统开发 – 软总线创建会话服务

## 1、定义会话管理相关回调

```
void OnSessionClosed(int32_t sessionId)
{
    printf("OnSessionClosed: sessionId = %d\n", sessionId);
}

int32_t OnSessionOpened(int32_t sessionId)
{
    printf("OnSessionOpened: sessionId = %d\n", sessionId);
    return 0;
}

void OnBytesReceived(int32_t sessionId, const void *data, uint32_t dataLen)
{
    if (data == NULL || dataLen == 0) {
        return;
    }
    if (strcmp(data, "TurnOn") == 0) {
        IoTgpioSetOutputVal(LED_TEST_GPIO, 0);
    } else if (strcmp(data, "TurnOff") == 0) {
        IoTgpioSetOutputVal(LED_TEST_GPIO, 1);
    } else {
        printf("invalid cmd\n");
    }
}
```

## 2、创建会话服务

```
static struct ISessionListener g_sessionCallback = {
    .onSessionOpened = OnSessionOpened,
    .onSessionClosed = OnSessionClosed,
    .onBytesReceived = OnBytesReceived,
};

int ret = CreateSessionServer(g_demoModuleName, g_demoSessionName, &g_sessionCallback);
if (ret != 0) {
    printf("CreateSessionServer\n");
}
```

# WIFI模组设备系统开发 - 指示灯亮、灭控制实现

根据手机APP命令类型执行，执行响应的操作，完成指示灯亮、灭控制。

- “TurnOn”：3861设备指示灯打开
- “TurnOff”：3861设备指示灯关闭

```
void OnBytesReceived(int32_t sessionId, const void *data, uint32_t dataLen)
{
    if (data == NULL || dataLen == 0) {
        return;
    }
    if (strcmp(data, "TurnOn") == 0) {
        IoTgpioSetOutputVal(LED_TEST_GPIO, 0);
    } else if (strcmp(data, "TurnOff") == 0) {
        IoTgpioSetOutputVal(LED_TEST_GPIO, 1);
    } else {
        printf("invalid cmd\n");
    }
}
```

# WIFI模组设备系统开发 - 体验一下

- 准备wifi热点，手机和3861设备均接入同一局域网；
- 3861编译，生成二进制文件，烧录到3861设备中；
- 3861设备上电，设备进入**启动流程**，包括配网、软总线初始化、服务发布、会话服务创建等（**3861指示灯状态：闪烁**）；
- 3861**启动完成**，设备支持被发现（**3861指示灯状态：常亮**）；
- 启动手机端APP，待3861启动完成后，点击“Init Env”按钮，APP启动，并实现与3861设备绑定（通过提示框显示绑定结果）
- 绑定成功后，手机亮灭屏，发现设备，认证通过后，APP文本框显示设备设备上线；
- 点击手机APP上“Turn On”按钮打开指示灯，“Turn Off”按钮关闭指示灯。

3861编译产品选择：

```
z 2023-12-15 10:00:00 [root@hisilicon:~]# hb set
[OHOS INFO] Input code path: .
OHOS Which product do you need? (Use arrow keys)

hisilicon
  □ wifiiot
    ipcamera_hi3516dv300_liteos
    wifiiot_hispark_pegasus
    ipcamera_hispark_aries
    ipcamera_hi3518ev300_liteos
    softbus_wifiiot
    ipcamera_hispark_taurus
    ipcamera_hispark_taurus_linux
```

注：点击“Clear Text”按钮，可清除文本框显示信息。

# 更多信息

---

- HarmonyOS官网：
  - <https://www.harmonyos.com/cn/home/>
- HarmonyOS设备开发官网：
  - <https://device.harmonyos.com/cn/home/>
- HarmonyOS开发者社区：
  - <https://developer.huawei.com/consumer/cn/forum/communityHome>

# Thank you.

把数字世界带入每个人、每个家庭、  
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and  
organization for a fully connected,  
intelligent world.

**Copyright©2021 Huawei Technologies Co., Ltd.  
All Rights Reserved.**

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.

