深圳市海凌科电子有限公司

HLK-W800-KIT-PRO

软件开发例程说明

目　录

1. 概述

本文主要介绍基于 HLK-W800-KIT-PRO 开发板与阿里生活物联网平台的灯(IoT)方案开发例程。

基于本例程 SDK 和阿里生活物联网平台对接开发中有问题请参考阿里官方文档或提交工单到阿里生活物联网平台，原则上本公司不提供无偿技术支持。阿里生活物联网平台开发参网址如下：https://help.aliyun.com/product/123207.html

2. 创建阿里物联网产品

2.1 参照

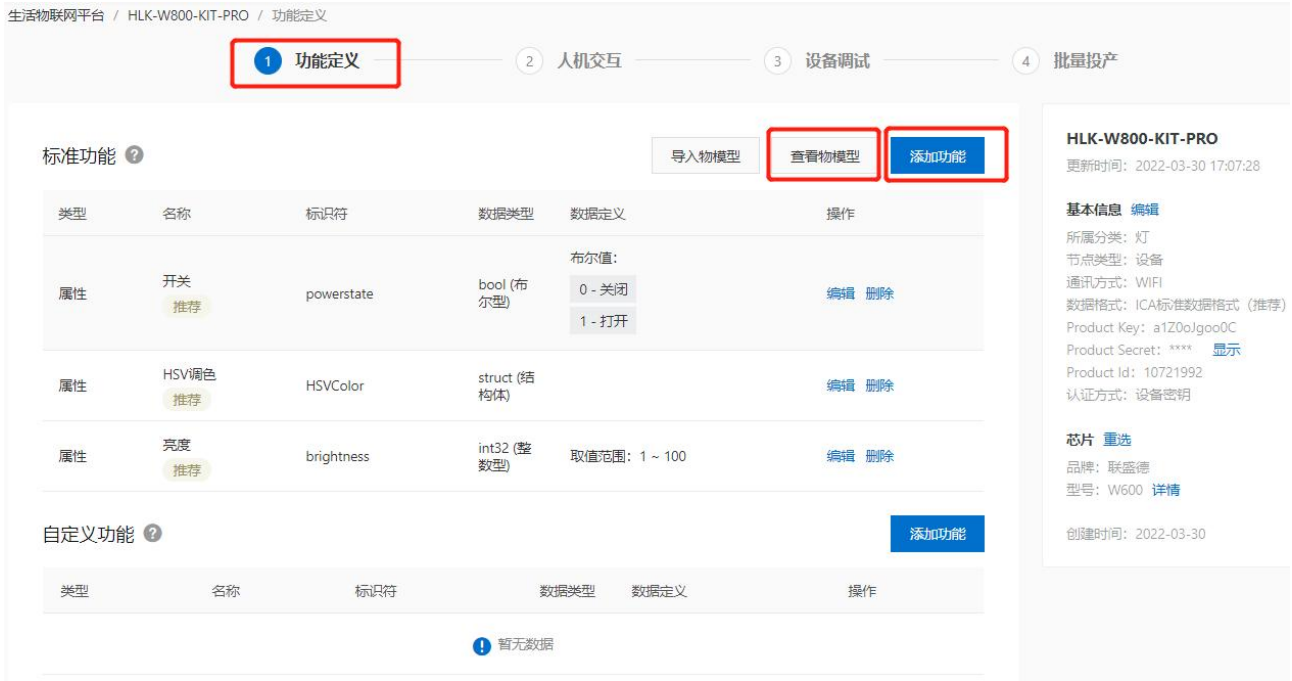
https://help.aliyun.com/document_detail/142147.html 中的指引完成阿里云账号的注册，并在生活物联网平台创建 IoT 产品。

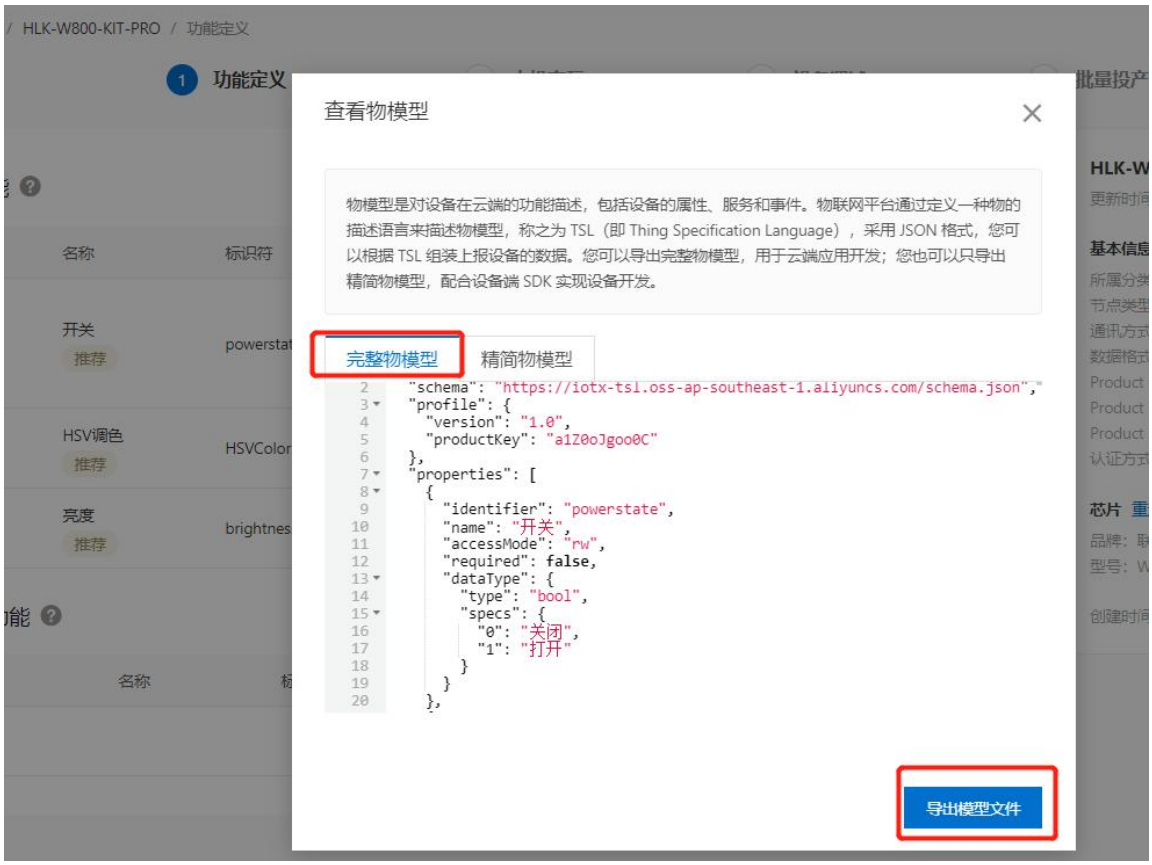
2.2 创建产品

创建产品时填写产品信息，节点类型，选择数据格式为【ICA 标准数据格式】：

2.3 功能定义

添加功能，完成功能定义后，点击右上角的【查看物模型】，选择【完整物模型】，即可导出海凌科语音定制后台所需的 IoT 物模型数据。(本方案 IoT 物模型数据详见附件:**W800-KIT-PRO-IoT 物模型.txt**)


生活物联网平台 / HLK-W800-KIT-PRO / 功能定义
1 功能定义
2 人机交互
3 设备调试
4 批量投产
标准功能
导入物模型
查看物模型
添加功能
类型 名称 标识符 数据类型 数据定义 操作
属性 开关 推荐 powerstate bool (布尔型) 布尔值: 0 - 关闭 1 - 打开 编辑 删除
属性 HSV调色 推荐 HSVColor struct (结构体) 编辑 删除
属性 亮度 推荐 brightness int32 (整数型) 取值范围：1 ~ 100 编辑 删除
自定义功能
添加功能
类型 名称 标识符 数据类型 数据定义 操作
暂无数据
HLK-W800-KIT-PRO
更新时间：2022-03-30 17:07:28
基本信息 编辑
所属分类：灯
节点类型：设备
通讯方式：WIFI
数据格式：ICA标准数据格式（推荐）
Product Key：a1Z0oJgoo0C
Product Secret：**** 显示
Product Id：10721992
认证方式：设备密钥
芯片 重选
品牌：联盛德
型号：W600 详情
创建时间：2022-03-30


查看物模型
物模型是对设备在云端的功能描述，包括设备的属性、服务和事件。物联网平台通过定义一种物的描述语言来描述物模型，称之为 TSL（即 Thing Specification Language），采用 JSON 格式，您可以根据 TSL 组装上报设备的数据。您可以导出完整物模型，用于云端应用开发；您也可以只导出精简物模型，配合设备端 SDK 实现设备开发。
完整物模型
精简物模型
"schema": "https://iotx-tsl.oss-ap-southeast-1.aliyuncs.com/schema.json",
"profile": {
"version": "1.0",
"productKey": "a1Z0oJgoo0C"
},
"properties": [
{
"identifier": "powerstate",
"name": "开关",
"accessMode": "rw",
"required": false,
"dataType": {
"type": "bool",
"specs": {
"0": "关闭",
"1": "打开"
}
}
},
导出模型文件

```
{
        "schema":"https://iotx-tsl.oss-ap-southeast-1.aliyuncs.com/schema.json",
        "profile":{
                "version":"1.0",
                "productKey":"a1Z0oJgoo0C"
        },
        "properties":[
                {
                        "identifier":"powerstate",
                        "name":"开关",
                        "accessMode":"rw",
                        "required":false,
                        "dataType":{
                                "type":"bool",
                                "specs":{
                                        "0":"关闭",
                                        "1":"打开"
                                }
                        }
                },
                {
                        "identifier":"HSVColor",
                        "name":"HSV调色",
                        "accessMode":"rw",
                        "required":false,
                        "dataType":{
                                "type":"struct",
                                "specs":[
                                        {
                                                "identifier":"Hue",
                                                "name":"色调",
                                                "dataType":{
                                                        "type":"double"
```


2.4 人机交互

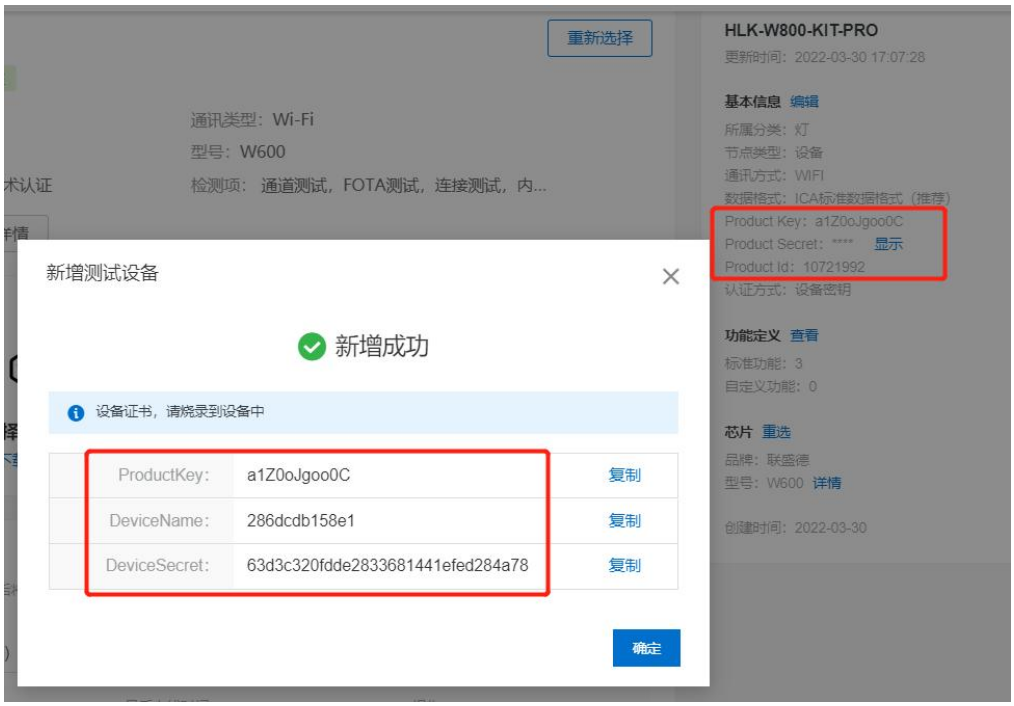
选择使用公版 App 控制产品，标准配网类型选择【灯(BLE+Wi-Fi)】，其余配置按提示完成。

2.5 设备调试

(1) 选择芯片信息

(2) 添加调试设备

(3) 添加完测试设备后，即可获得对应设备的授权码



(4) 将上述设备授权信息写入到测试设备中(文档中 6.6 烧录阿里云五元组有提供参考方法)

(5) 重启设备后生效，测试设备即可接入阿里 IoT，可在云端或手机 app 端进行联调，阿里 IoT 的调试方法可参考 https://help.aliyun.com/document_detail/142147.html 中相关章节

2.6 产品发布

注：产品发布后才能设备联网调试

3. 搭建开发环境

3.1 环境准备

(1) 下载编译工具

推荐使用 Ubuntu16.04/centos7 以上版本作为开发环境。

交叉编译工具下载地址：

链接：https://pan.baidu.com/s/1oEDxL3r875l9bist51R4jg

提取码：aruy

下载交叉编译工具并解压到开发环境“/opt”路径中，或其他路径：

```
alpha@alpha-virtual-machine:/opt/us615$ ls -l
总用量 32
drwxr-xr-x 2 root root 4096 10月 11 19:10 bin
drwxr-xr-x 6 root root 4096 10月 11 19:10 csky-elfabiv2
drwxr-xr-x 3 root root 4096 10月 11 19:10 include
drwxr-xr-x 3 root root 4096 10月 11 19:10 lib
drwxr-xr-x 2 root root 4096 10月 11 19:10 lib64
drwxr-xr-x 3 root root 4096 10月 11 19:10 libexec
drwxr-xr-x 7 root root 4096 10月 11 19:10 share
drwxr-xr-x 3 root root 4096 10月 11 19:10 x86_64-linux
alpha@alpha-virtual-machine:/opt/us615$
```

再将路径配置到系统环境变量中：

```
alpha@alpha-virtual-machine:/opt/us615$ vi ~/.bashrc
alpha@alpha-virtual-machine:/opt/us615$ source ~/.bashrc
```

添加进环境变量中

主动加载环境变量

```
117     . /etc/bash_completion
118   fi
119 fi
120
121 export PATH=/opt/us615/bin:$PATH
```

改为你的编译链路径

(2) 文件夹内包含产品方案源码，如下图所示：

```
alpha@alpha-virtual-machine:~/sdk/project/W800_KIT_PRO/unione_lite_app_hb_w$ ls -l
总用量 20
drwxrwxr-x  3 alpha alpha 4096 3月  25 12:54 boards
drwxrwxr-x 45 alpha alpha 4096 3月  25 12:54 components
-rw-rw-r--  1 alpha alpha   74 1月  19 09:17 readme.txt
drwxrwxr-x  3 alpha alpha 4096 3月  25 13:05 solutions
drwxrwxr-x  2 alpha alpha 4096 3月  25 13:03 tools
```

(3) 安装 python 环境：sudo apt-get install python

(4) 安装 lame 音频处理工具：sudo apt-get install lame

(5) 安装编译开发工具集：sudo pip install yoctools -i https://mirrors.aliyun.com/pypi/simple

(6) 安装 32 位兼容库（如果系统未安装过）：sudo apt-get install lib32stdc++6 lib32z1 lib32ncurses5 lib32bz2-1.0（仅适用于 Ubuntu 16.04 版本，其他版本请根据系统说明解决兼容性问题)

3.2 编译

(1) 执行编译

进入 solution/unisound 目录，执行 make：

```
alpha@alpha-virtual-machine:~/sdk/project/W800_KIT_PRO/unione_lite_app_hb_w$ cd solutions/unisound/
alpha@alpha-virtual-machine:~/sdk/project/W800_KIT_PRO/unione_lite_app_hb_w/solutions/unisound$ ls -l
总用量 9352
drwxrwxr-x  5 alpha alpha    4096 3月  25 12:54 app
drwxrwxr-x  3 alpha alpha    4096 3月  27 18:33 generated
drwxrwxr-x  2 alpha alpha    4096 3月  25 12:54 libs
-rw-rw-r--  1 alpha alpha    2786 2月   9 16:40 Makefile
drwxrwxr-x 37 alpha alpha    4096 3月  27 18:33 out
-rw-rw-r--  1 alpha alpha    4801 2月  25 18:59 package.yaml
-rw-rw-r--  1 alpha alpha      45 1月  19 09:17 SConscript
-rw-rw-r--  1 alpha alpha     245 2月   9 16:40 SConstruct
-rwxrwxr-x  1 alpha alpha 5447124 3月  28 09:56 yoc.elf
-rw-rw-r--  1 alpha alpha 4084547 3月  28 09:56 yoc.map
drwxrwxr-x  3 alpha alpha    4096 3月  27 18:32 yoc_sdk
alpha@alpha-virtual-machine:~/sdk/project/W800_KIT_PRO/unione_lite_app_hb_w/solutions/unisound$ make
Build Solution by
scons: Reading SConscript files ...
scons: done reading SConscript files.
scons: Building targets ...
CC out/aos/src/crc16.o
CC out/aos/src/except.o
CC out/aos/src/kv.o
CC out/aos/src/list.o
CC out/aos/src/lpm.o
CC out/aos/src/main.o
```

编译成功会看到如下信息：

```
scons: done building targets.
YoC SDK Done
[INFO] Create bin files
[2022-03-29 11:16:04] Start to sign images with key:def_otp
[2022-03-29 11:16:04] Sign [prim] with [def_otp]
[2022-03-29 11:16:04] rsa verify ok....
[2022-03-29 11:16:04] Sign prim ok.
------------------------------------------------------------
                boot, 0, 0, 0x08002400, 0x0000dc00, 0x08010000, boot
                imtb, 0, 0, 0x08010000, 0x00002000, 0x08012000, imtb
                prim, 1, 1, 0x08012000, 0x001b7000, 0x081c9000, prim
                misc, 0, 0, 0x081c9000, 0x00030000, 0x081f9000
                  kv, 0, 0, 0x081f9000, 0x00004000, 0x081fd000
    boot,    41900 bytes
    prim,  1649248 bytes
    imtb,     8192 bytes
------------------------------------------------------------
[INFO] Create bin files
In file included from /usr/include/string.h:495,
                 from ../../components/chip_us615/unisdk/tools/us615/uni_tool.c:7:
In function 'strncpy',
    inlined from 'uni_tool_get_name' at ../../components/chip_us615/unisdk/tools/us615/uni_tool.c:2907:5:
/usr/include/x86_64-linux-gnu/bits/string_fortified.h:106:10: warning: '__builtin_strncpy' specified bound 256 equals destination size [-Wstringop-truncation]
  106 |   return __builtin___strncpy_chk (__dest, __src, __len, __bos (__dest));
      |          ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
generate normal image completed.
compress binary completed.
generate compressed image completed.
[INFO] Create fls file
generate normal image completed.
generate normal image completed.
generate normal image completed.
```

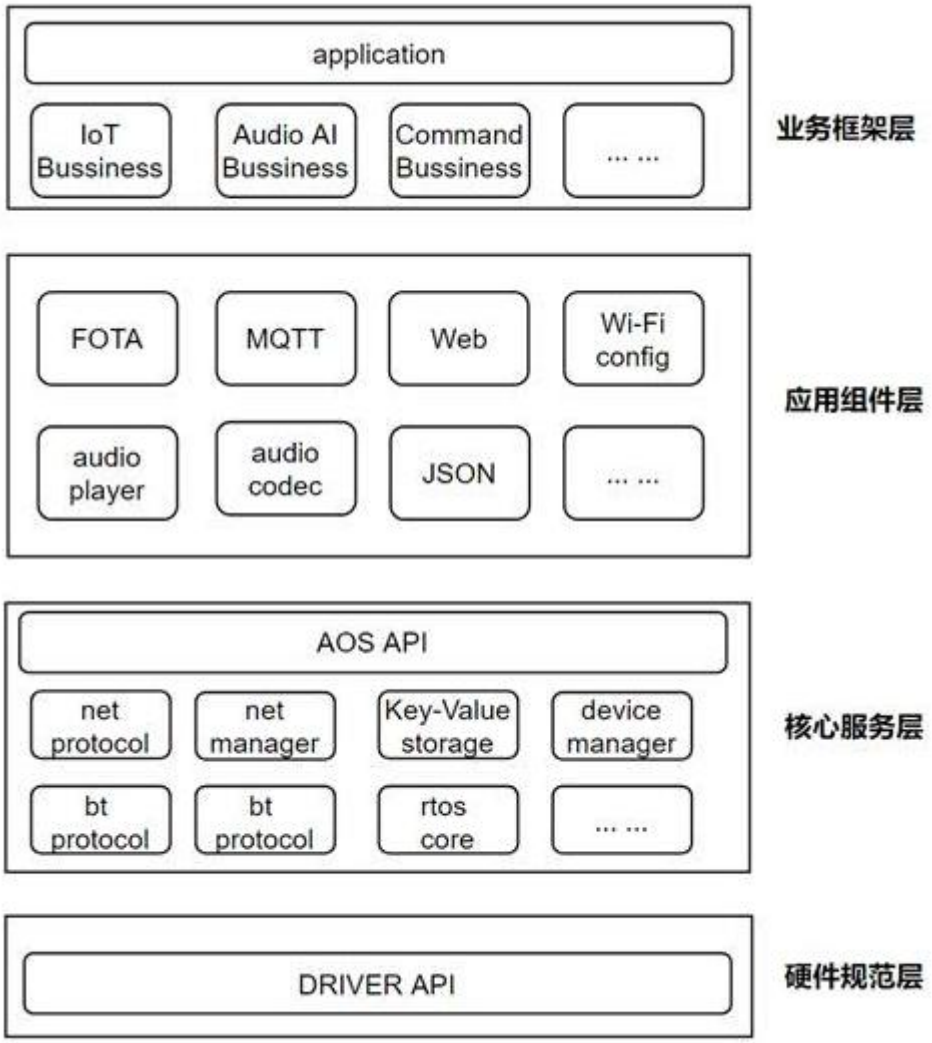
(2) out/unisound 目录下可以看到编译生成的完整固件，使用下载工具即可进行烧录使用：

```
alpha@alpha-virtual-machine:~/sdk/project/W800_KIT_PRO/unione_lite_app_hb_w/solutions/unisound$ ls -l out/unisound/
总用量 12652
drwxrwxr-x 3 alpha alpha    4096 3月  27 18:33 app
-rw-rw-r-- 1 alpha alpha 1699532 3月  29 11:16 uni_app.fls
-rwxrwxr-x 1 alpha alpha   61352 3月  29 11:16 uni_tool
-rwxrwxr-x 1 alpha alpha 1649092 3月  29 11:16 yoc.bin
-rw-rw-r-- 1 alpha alpha 1217915 3月  29 11:16 yoc.bin.gz
-rwxrwxr-x 1 alpha alpha 5447124 3月  29 11:16 yoc.elf
-rw-rw-r-- 1 alpha alpha 1649156 3月  29 11:16 yoc.img
-rw-rw-r-- 1 alpha alpha 1217980 3月  29 11:16 yoc_ota.bin
alpha@alpha-virtual-machine:~/sdk/project/W800_KIT_PRO/unione_lite_app_hb_w/solutions/unisound$
```

4. 软件架构

4.1 软件架构

源码框架采用分层设计，各个层次完成特定的功能封装，基本框架结构如下图所示：



各个层次功能描述如下：

硬件规范层：定义了 uart、spi、i2c、gpio 等芯片外设驱动的统一接口；

核心服务层：包含了操作系统内核、设备管理框架、网络协议、蓝牙协议、网络管理器等核心服务模块；

应用组件层：提供了大量的应用组件，满足不同产品需求的选择；

业务框架层：提供 IoT、语音识别、人机交互等多种应用领域软件框架，简化了应用软件的编写难度。

4.2 源码目录结构

源码各个文件功能如下所示：

一般情况下，开发者仅需改动 solution/unisound 目录即可实现大部分的需求。

根目录：

```
zhicwang@ubuntu:~/test/uni_hb_w_solution/unione_lite_app_hb_w$ tree -L 1
.
├── boards 板级配置，如默认pinmux，kernel config
├── components 组件库，包含了驱动以及各种核心、应用组件
├── solutions 业务方案，即业务逻辑的具体实现
└── tools 备用工具包
```

常用组件如下：

```
zhicwang@ubuntu:~/test/uni_hb_w_solution/unione_lite_app_hb_w$ tree -L 1 components/
components/
├── amrnb
├── amrwb
├── aos          ← aos api
├── at           ← AT 控制指令组件
├── av
├── ble_host
├── breeze
├── button
├── chip_us615
├── cjson
├── console      ← uart调试控制台
├── csi          ← 驱动及外设统一接口
├── drivers
├── drv_bt_us615
├── drv_wifi_us615
├── ers
├── flac
├── fota         ← fota在线升级组件
├── httpclient
├── key_mgr
├── kv           ← key-value持久化存储组件
├── lwip         ← 网络协议栈
├── mbedtls
├── minilibc
├── netmgr       ← 网络管理器
├── ntp
├── offline_voice ← 离线语音识别框架
├── opus
├── partition
├── pvmp3dec
├── rhino
├── rhino_arch   ← 操作系统内核
├── rhino_pwrmgmt
├── sec_crypto
├── smartliving  ← 阿里生活物联网sdk
├── sonic
├── speex
├── speexdsp
├── transport
├── ulog
├── uservice
├── vfs
└── wifi_provisioning ← 网络配置api
```

5. 接口说明

W800-KIT-PRO 的 SDK 入口函数 int main()，位于 solutions\unisound\app\src\app_main.c 文件里(如下图)。

```
app_main.c 9+
unione_lite_app_hb_w > solutions > unisound > app > src > app_main.c > main()
int main() {
  board_yoc_init();
  LOGD(TAG, "%s\n", aos_get_app_version());
  printf("****************HLK_800_PRO-V1.0.0******************\r\n");
```

int main():

a) board_yoc_init() : 初始化整个 sdk，初始化的内容包括：flash 管理模块、设备驱动模块、日志打印模块、gpio 功能模块、离线语音 AI 模块、key-value 存储模块、cli 命令模块。

b) aos_get_app_version() : 获取 APP 当前固件版本信息接口。

c) event_service_init () : 事件服务初始化接口。

d) app_sys_init() : 系统初始化标志位设置。

e) board_base_init() : uart，iic 注册驱动程序。

f) sys_event_init() : 注册系统异常处理，若错误，会调用到回调函数，可以做异常处理。

g) cli_reg_cmds() : cli 注册命令，可参考示例添加自己的命令。

h) csi_codec_init() : 音频解码驱动初始化。

i) local_audio_init() : 本地音频文件初始化。

j) app_iot_init() : IOT 事件初始化。

k) app_button_init() : 添加 gpio 按键事件初始化。

l) app_fota_init() : OTA 服务初始化。

m) app_network_init() : wifi 使能，注册 smartliving wifi 发放方式，包括 smartconfig 和设备 ap。

n) kws_start() : 使能语音识别。

6. 例程开发说明

6.1 GPIO 初始化

(1) 用户可以根据自己的使用情况，修改硬件引脚（uni_auto_control.c）

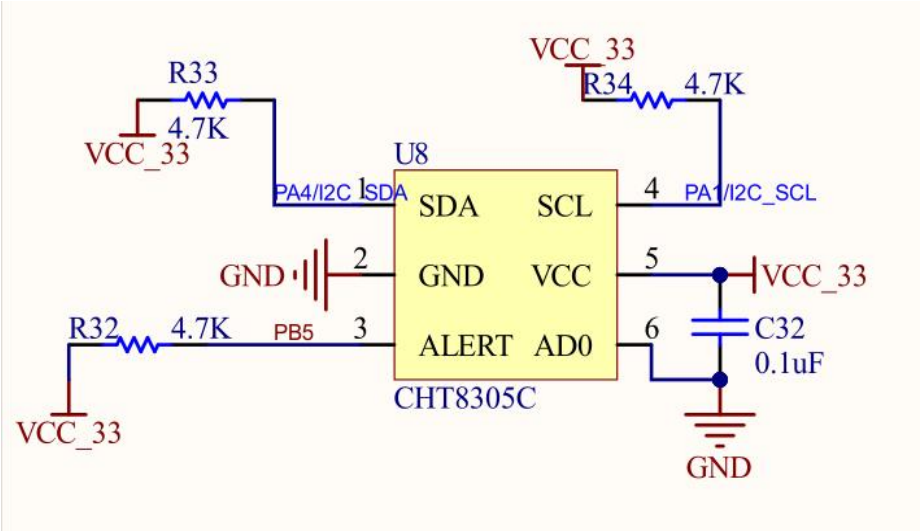
unione_lite_app_hb_w > solutions > unisound > app > src > unisound > user > C uni_auto_control.c > user_gpio_init(void)

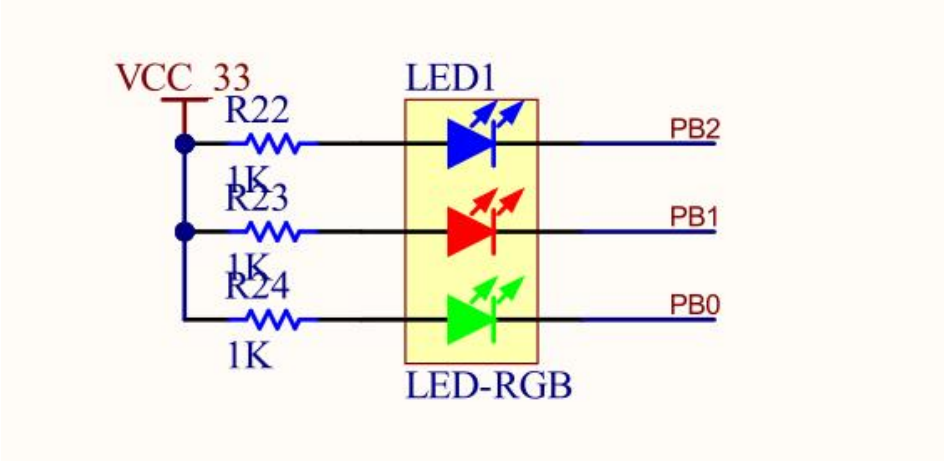
```c
static void _user_gpio_set_pinmux(void) {

  user_pin_set_func(PA1, PA1_IIC_SCL);
  user_pin_set_func(PA4, PA4_IIC_SDA);

  user_pin_set_func(PA0, PIN_FUNC_GPIO);
  user_pin_set_func(PB0, PB0_PWM);
  user_pin_set_func(PB1, PB1_PWM);
  user_pin_set_func(PB2, PB2_PWM);
  user_pin_set_func(PB4, PB4_UART4_TX);
  user_pin_set_func(PB5, PB5_UART4_RX);
  user_pin_set_func(PB6, PB6_UART1_TX);
  user_pin_set_func(PB7, PB7_UART1_RX);
}

int user_gpio_init(void) {
  _user_gpio_set_pinmux();

  tls_i2c_init(5000);

  user_gpio_set_direction(PA0, GPIO_DIRECTION_OUTPUT);
  user_gpio_set_level(PA0, UNI_GPIO_LEVEL_LOW);
  user_pwm_init(PB0, 1000, 0, 100);
  user_pwm_init(PB1, 1000, 0, 100);
  user_pwm_init(PB2, 1000, 0, 100);

  user_uart_init(PB6, 115200, 8, 1, 0);

  LOGI(LOG_TAG, " %s success", __func__);
  return 0;
}
```

温湿度传感器配置引脚

RGB灯配置引脚

(2) 温湿度传感器、LED-RGB 原理图





(3) W800 芯片数据手册

18	PB_0	I/O	GPIO, 输入，高阻	PWM0/LSPI_MISO/UART3_TX	80MHz	UP/DOWN	12mA
19	PB_1	I/O	GPIO, 输入，高阻	PWM1/LSPI_CK/UART3_RX	80MHz	UP/DOWN	12mA
20	PB_2	I/O	GPIO, 输入，高阻	PWM2/LSPI_CK/UART2_TX/SIM_DATA_2	80MHz	UP/DOWN	12mA

14	PA_1	I/O	JTAG_CK	JTAG_CK/I²C_SCL/PWM3/I²S_LRCK/ADC0	20MHz	UP/DOWN	12mA

15	PA_4	I/O	JTAG_SWO	JTAG_SWO/I²C_SDA/PWM4/I²S_BCK/ADC1	20MHz	UP/DOWN	12mA

6.2 功能修改

(1) 创建一个消息队列，利于后面处理颜色调节事件的触发；

unione_lite_app_hb_w > solutions > unisound > app > src > C msg_process_center.c > ...

```c
void send_msg_to_queue(led_strip_msg_t * cmd_msg)
{
    int ret = aos_queue_send(g_cmd_msg_queue_id, cmd_msg, sizeof(led_strip_msg_t));
    if (0 != ret)
        printf("################ERROR: CMD MSG: aos_queue_send failed! #################\r\n");
}

void init_msg_queue()
{
    if (g_cmd_msg_queue_buff == NULL) {
        g_cmd_msg_queue_id = (aos_queue_t *) aos_malloc(sizeof(aos_queue_t));
        g_cmd_msg_queue_buff = aos_malloc(MB_RGBSTATUS_COUNT * sizeof(led_strip_msg_t));

        aos_queue_new(g_cmd_msg_queue_id, g_cmd_msg_queue_buff, MB_RGBSTATUS_COUNT * sizeof(led_strip_msg_t),
                sizeof(led_strip_msg_t));
    }
}
```

(2) 在注册回调函数中，处理阿里云下发属性数据。

app_main.c 9+ app_smartliving_demo.c 8

unione_lite_app_hb_w > solutions > unisound > app > src > app_smartliving_demo.c > user_property_set_event_handler(const int, const char *, const int)

```c
static int user_property_set_event_handler(const int devid, const char *request, const int request_len)
{
    printf("Property Set Received, Devid: %d, Request: %s\n", devid, request);

    led_strip_msg_t msg;

    property_setting_handle(request, request_len, &msg);

    iotdispatcher_smartliving_set((char *)request);

#if defined(CONFIG_SMARTLIVING_AT_MODULE) && CONFIG_SMARTLIVING_AT_MODULE
        app_at_cmd_property_report_set(request, request_len);
#endif

    return 0;
}
```

(3) 通过 comonents\cjson\include\cJSON.h 中提供的 JSON 解析函数，得到相应的 HSV 颜色数值，并通过发送消息到队列中实现颜色调节。

app_main.c 9+ app_smartliving_demo.c 8

unione_lite_app_hb_w > solutions > unisound > app > src > app_smartliving_demo.c > property_setting_handle(const char *, const int, led_strip_msg_t *)

```c
static int property_setting_handle(const char *request, const int request_len, led_strip_msg_t * msg)
{
    cJSON *root = NULL, *item = NULL;
    int ret = 0;

    if ((root = cJSON_Parse(request)) == NULL) {
        printf("property set payload is not JSON format\n");
        return -1;
    }
    msg->validindex = INVALID;

    if ((item = cJSON_GetObjectItem(root, "HSVColor")) != NULL && cJSON_IsObject(item)) {
        cJSON *hsvJson = cJSON_GetObjectItem(item, "Hue");

        msg->hsvcolor[0].h = hsvJson->valueint;

        hsvJson = cJSON_GetObjectItem(item, "Saturation");
        msg->hsvcolor[0].s = hsvJson->valueint;

        hsvJson = cJSON_GetObjectItem(item, "Value");
        msg->hsvcolor[0].v = hsvJson->valueint;

        msg->arraysize = 1;
        msg->validindex = HSV_VALID;
    }
```

JSON 解析数据类型可通过阿里生活物联网平台查看物模型

标准功能

查看物模型

类型	名称	标识符	数据类型	数据定义	操作
属性	开关 推荐	powerstate	bool (布尔型)	布尔值: 0 - 关闭 1 - 打开	查看
属性	HSV调色 推荐	HSVColor	struct (结构体)		查看
属性	亮度 推荐	brightness	int32 (整数型)	取值范围: 0 ~ 100	查看

(4) 创建一个线程任务，不断读取队列中颜色调节消息并执行相应的逻辑操作，用户可根据自己实际应用去调试开发。

```
aos_task_new_ext(&task_msg_process, "cmd msg process", msg_process_task, NULL, 2048, AOS_DEFAULT_APP_PRI);
```

unione_lite_app_hb_w > solutions > unisound > app > src > msg_process_center.c > h_t_timer_callback(void *, void *)

```
void msg_process_task(void *argv)
{
    unsigned int rcvLen;

    aos_timer_new_ext(&h_t_timer, h_t_timer_callback, NULL, 2000, 1, 1);

    while (true) {
        if (aos_queue_recv(g_cmd_msg_queue_id, AOS_WAIT_FOREVER, &msg, &rcvLen) == 0) {

            switch (msg.validindex) {
                case LS_VALID:
                    if(msg.lightswitch)
                    {
                        HSV_Color.light_switch=1;
                        led_static_color_show_other(HSV_Color.h, HSV_Color.s, HSV_Color.v);
                    }
                    else{
                        HSV_Color.light_switch=0;
                        user_pwm_enable(PB0, 100);
                        user_pwm_enable(PB1, 100);
                        user_pwm_enable(PB2, 100);
                    }

                    post_property_task(HSV_Color.light_switch,HSV_Color.h, HSV_Color.s, HSV_Color.v);
                    break;
```

HSV转RGB

(5) 上报属性数据到云端对应接口函数

unione_lite_app_hb_w > solutions > unisound > app > src > C app_smartliving_demo.c > user_post_given_property(char *, int)

```c
int user_post_given_property(char *message, int len)
{
    if (!user_master_dev_available()) {
        return -1;
    }
    int res = 0;
    res = IOT_Linkkit_Report(user_example_get_ctx()->master_devid, ITM_MSG_POST_PROPERTY,
                             (unsigned char *)message, len);
    printf("Post Given Property Message ID: %d MSG: %s\n", res, message);
    return res;
}
```

6.3 语音控制

unione_lite_app_hb_w > solutions > unisound > app > src > unisound > user > C uni_auto_control.c > user_gpio_handle_action_event(const char *, uint8_t *, uint8_t *)

```c
int user_gpio_handle_action_event(const char *action,
                                  uint8_t *need_reply, uint8_t *need_post) {
  led_strip_msg_t msg;
  if (action != NULL) {
    LOGD(LOG_TAG, "handle kws result action: %s", action);
    if (0 == strcmp(action, "Network#null#0")) {
      uni_iot_set_Network(1);
    } else if (0 == strcmp(action, "powerstate#val#0")) {
      uni_iot_set_powerstate(0);
    } else if (0 == strcmp(action, "powerstate#val#1")) {
      uni_iot_set_powerstate(1);
    } else if (0 == strcmp(action, "brightness#val#0")) {
      uni_iot_set_brightness(0);
    } else if (0 == strcmp(action, "brightness#val#10")) {
      uni_iot_set_brightness(10);
    } else if (0 == strcmp(action, "brightness#val#20")) {
      uni_iot_set_brightness(20);
```

执行语音命令配网操作

执行语音命令控制开关

执行语音命令控制亮度

语音配网

unione_lite_app_hb_w > solutions > unisound > app > src > unisound > user > C uni_auto_control.c > uni_iot_set_colorTemperatureInKelvin(int)

```c
void uni_iot_set_Network(int val) {
  switch (val) {
    case 1: {
      // UniPlayStop();
      // mvoice_alg_deinit();

      awss_report_reset(0);
      aos_msleep(500);

      aos_kv_setint("wprov_method", wifi_prov_method);
      if (wifi_prov_method == WIFI_PROVISION_SL_BLE) {
        printf("wifi_prov_method=WIFI_PROVISION_SL_BLE\n");
        aos_kv_del("AUTH_AC_AS");
        aos_kv_del("AUTH_KEY_PAIRS");
        aos_kv_del("wifi_ssid");
        aos_kv_del("wifi_psk");
      }
      app_sys_set_boot_reason(BOOT_REASON_WIFI_CONFIG);//zz
      aos_reboot();
    }

    default:
      LOGW(LOG_TAG, "%s val %d unknown", __func__, val);
      break;
  }
}
```

上报阿里云解绑设备接口

6.4 温湿度检测

创建一个定时器，定时 2s 检测一次温湿度数据

```
aos_timer_new_ext(&h_t_timer, h_t_timer_callback, NULL, 2000, 1, 1);
```

unione_lite_app_hb_w > solutions > unisound > app > src > msg_process_center.c > h_t_timer_callback(void *, void *)

```
static void h_t_timer_callback(void *arg1, void *arg2)
{
    float  ct8305_temp ;
    float  ct8305_humi;
    u32 HUM,TEMPER;
    cht8305_GetTempHumi(&ct8305_temp,&ct8305_humi);
    HUM  = (ct8305_humi +0.5)*1;
    TEMPER = (ct8305_temp +0.5)*1;
    // printf("HUM %d,TEMPER %d\r\n",HUM ,TEMPER );
}
```

6.5 AT 指令

unione_lite_app_hb_w > solutions > unisound > app > src > combo_net.c > cmd_list

```
static cmd_tbl_t cmd_list[] = {
    {
        "AT",               1,   do_at,
        "AT                 - AT"
    },
    {
        "AT+VER",           2,   do_hlkver,
        "AT+VER             - get ver"
    },
    {
        "at+Get_MAC",           2,   do_Get_MAC,
        "at+Get_MAC             - get mac"
    },
    {
        "at+devset",        8,   do_alinkdev_set,
        "at+devset          - set deveic info"
    },
    {
        "at+otaset",        8,   do_fota_set,
        "at+otaset          - set fota info"
    },
    {
        "at+macset",        8,   do_mac_set,
        "at+macset          - set fota info"
    },
    {
        "at+wscan",        8,   do_wscan_handler,
        "at+wscan          - set fota info"
    },
};
```

6.6 烧录阿里云五元组

可通过以下函数实现烧写覆盖五元组

```
HAL_SetProductKey(buffer[0]);
HAL_SetDeviceName(buffer[1]);
HAL_SetDeviceSecret(buffer[2]);
HAL_SetProductSecret(buffer[3]);

pid = atoi(buffer[4]);
aos_kv_setint("hal_devinfo_pid", pid);
```

6.7 修改 APP 固件版本号

通过修改此处宏，可在 APP 联网设备上查看当前的固件版本号

unione_lite_app_hb_w > components > aos > src > sysinfo.c > CONFIG_SDK_VERSION

```
#ifndef CONFIG_SDK_VERSION
// #define CONFIG_SDK_VERSION "V7.2.2 20200319"
#define CONFIG_SDK_VERSION "V1.0.0"
#endif
```

6.8 看门狗

初始化看门狗 10s 触发，定时每隔 5s 喂一次狗

unione_lite_app_hb_w > solutions > unisound > app > src > app_main.c > main()

```
static void w_dog_timer_callback(void *arg1, void *arg2)
{
    tls_watchdog_clr();
}

static void watchdog_init_and_clr(void) {
    aos_timer_t _watchdog_flag;
    aos_timer_new_ext(&_watchdog_flag, w_dog_timer_callback, NULL, 5*1000, 1, 0);
    tls_watchdog_init(10000000);
    aos_timer_start(&_watchdog_flag);
}
```

7. 海凌科语音定制后台功能开放计划表

目前 SDK 暂未开发语音定制功能，我司会根据产品研发进度，持续推进语音功能的开发工作，研发计划如下表，所有解释权归深圳市海凌科电子有限公司所有。

功能项	功能描述	预计开放时间
唤醒词自定义	唤醒词词条，回复语，灵敏度	2022 年 4 月上旬左右
发音人配置	发音人，音高，音量，语速，亮度	2022 年 4 月上旬左右
其他配置	开机播报，超时退出时间和回复语，主动退出命令词和回复语	2022 年 4 月上旬左右
离线命令词与应答语自定义	串口协议定制	2022 年 5 月中旬左右

8. 附录 A 文档修订记录

版本号	修订范围	日 期
V1.0	初始版本。	2022 年 3 月 31 日