

驱动开发示例与WIFI连接

内容

- 驱动开发代码示例
- 如何使用驱动示例
- 实验5-1 PWM实现led呼吸灯
- WiFi 基本概念
- WiFi STA模式编程
- 实验5-2 连接WiFi热点

一、驱动代码示例

OpenHarmony 代码中，Hi3861 提供了绝大部分的驱动示例代码，包括I2C、ADC、PWM、SPI 等，为我们快速开发应用层代码提供了参考。

一、驱动代码示例

3.0-lts > src > device > hisilicon > hispark_pegasus > sdk_liteos > app > demo > src

排序 查看

名称	修改日期	类型	大小
app_demo_adc.c	2022-01-10 13:53	C 文件	5 KB
app_demo_efuse.c	2022-01-10 13:53	C 文件	7 KB
app_demo_flash.c	2022-01-10 13:53	C 文件	4 KB
app_demo_i2c.c	2022-01-10 13:53	C 文件	4 KB
app_demo_i2s.c	2022-01-10 13:53	C 文件	8 KB
app_demo_io_gpio.c	2022-01-10 13:53	C 文件	3 KB
app_demo_nv.c	2022-01-10 13:53	C 文件	4 KB
app_demo_pwm.c	2022-01-10 13:53	C 文件	2 KB
app_demo_sdio_device.c	2022-01-10 13:53	C 文件	6 KB
app_demo_sdio_device.h	2022-01-10 13:53	H 文件	1 KB
app_demo_spi.c	2022-01-10 13:53	C 文件	32 KB
app_demo_timer_systick.c	2022-01-10 13:53	C 文件	5 KB
app_demo_tsensor.c	2022-01-10 13:53	C 文件	5 KB
app_demo_uart.c	2022-01-10 13:53	C 文件	3 KB
app_demo_upg_verify.c	2022-01-10 13:53	C 文件	2 KB
app_demo_upg_verify.h	2022-01-10 13:53	H 文件	1 KB
app_http_client.c	2022-01-10 13:53	C 文件	3 KB
app_main.c	2022-01-10 13:53	C 文件	18 KB
app_promis.c	2022-01-10 13:53	C 文件	3 KB
app_promis.h	2022-01-10 13:53	H 文件	1 KB
es8311_codec.c	2022-01-10 13:53	C 文件	7 KB

```
3.0-LTS
├── src
│   ├── device\hisilicon\hispark_pegasus
│   │   ├── sdk_liteos
│   │   │   ├── license
│   │   │   ├── app
│   │   │   │   ├── wifiot_app
│   │   │   │   ├── demo
│   │   │   │   │   ├── src
│   │   │   │   │   │   ├── app_demo_adc.c
│   │   │   │   │   │   ├── app_demo_efuse.c
│   │   │   │   │   │   ├── app_demo_flash.c
│   │   │   │   │   │   ├── app_demo_i2c.c
│   │   │   │   │   │   ├── app_demo_i2s.c
│   │   │   │   │   │   ├── app_demo_io_gpio.c
│   │   │   │   │   │   ├── app_demo_nv.c
│   │   │   │   │   │   ├── app_demo_pwm.c
│   │   │   │   │   │   ├── app_demo_sdio_device.c
│   │   │   │   │   │   ├── app_demo_sdio_device.h
│   │   │   │   │   │   ├── app_demo_spi.c
│   │   │   │   │   │   ├── app_demo_timer_systick.c
│   │   │   │   │   │   ├── app_demo_tsensor.c
│   │   │   │   │   │   ├── app_demo_uart.c
│   │   │   │   │   │   ├── app_demo_upg_verify.c
│   │   │   │   │   │   ├── app_demo_upg_verify.h
│   │   │   │   │   │   ├── app_http_client.c
│   │   │   │   │   │   ├── app_main.c
│   │   │   │   │   │   ├── app_promis.c
│   │   │   │   │   │   └── app_promis.h
```

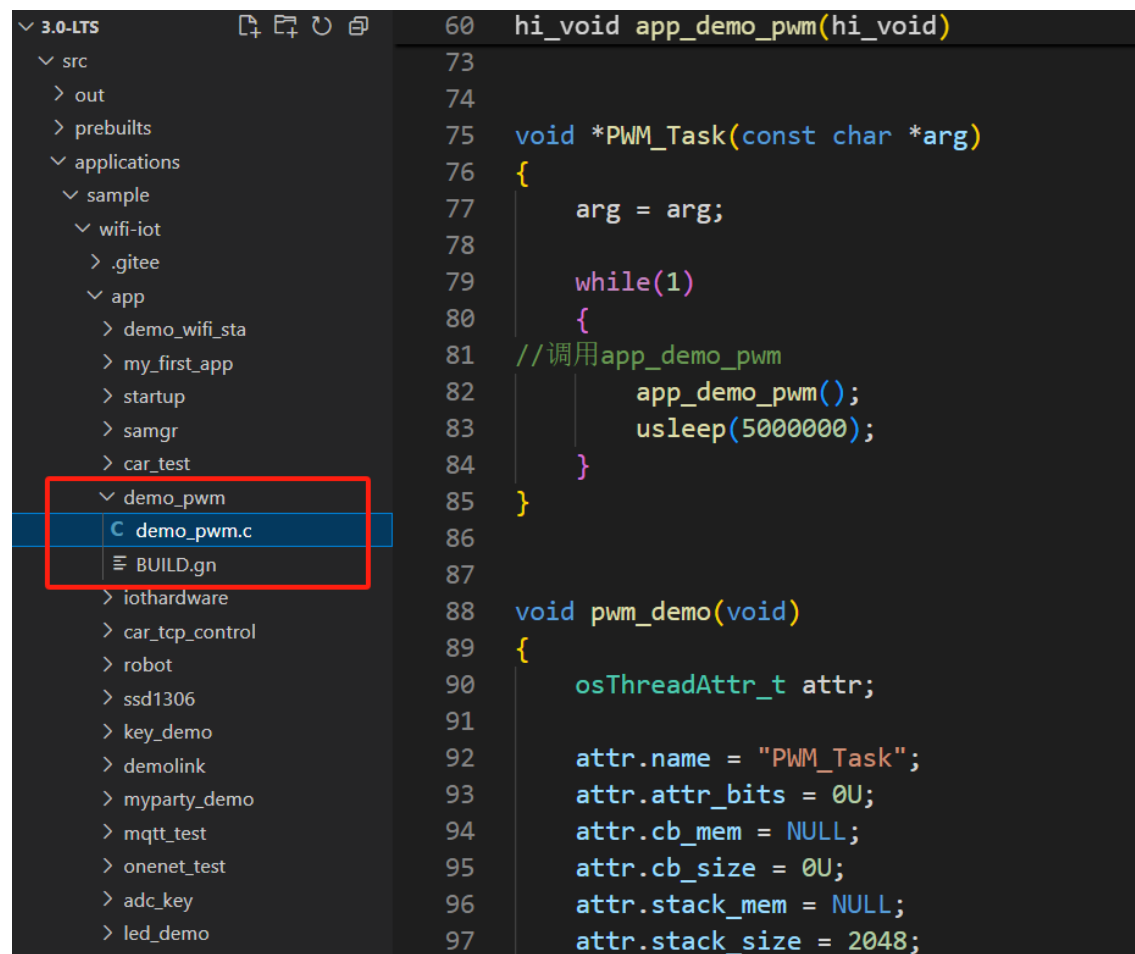
二、如何使用示例

驱动示例使用步骤：

- 1、创建文件夹，复制示例
- 2、编写入口函数
- 3、修改头文件
- 4、打开对应的宏

二、如何使用示例

1、创建文件夹，复制示例。以app_demo_pwm.c 为例。在app 中新建文件夹demo_pwm， 里面存放代码demo_pwm.c， 然后app_demo_pwm.c 所有的代码都复制到demo_pwm.c 中， 整个文件夹如下：



```
3.0-LTS
src
  out
  prebuilts
  applications
    sample
      wifi-iot
        .gitee
        app
          demo_wifi_sta
          my_first_app
          startup
          samgr
          car_test
          demo_pwm
            demo_pwm.c
            BUILD.gn
          iothardware
          car_tcp_control
          robot
          ssd1306
          key_demo
          demolink
          myparty_demo
          mqtt_test
          onenet_test
          adc_key
          led_demo
60 hi_void app_demo_pwm(hi_void)
73
74
75 void *PWM_Task(const char *arg)
76 {
77     arg = arg;
78
79     while(1)
80     {
81         //调用app_demo_pwm
82         app_demo_pwm();
83         usleep(5000000);
84     }
85 }
86
87
88 void pwm_demo(void)
89 {
90     osThreadAttr_t attr;
91
92     attr.name = "PWM_Task";
93     attr.attr_bits = 0U;
94     attr.cb_mem = NULL;
95     attr.cb_size = 0U;
96     attr.stack_mem = NULL;
97     attr.stack_size = 2048;
```

二、如何使用示例

2、为demo_pwm.c 编写一个入口函数，通常情况下，是创建一个线程去执行，通用的代码示例如下：

```
74
75 void *PWM_Task(const char *arg)
76 {
77     arg = arg;
78
79     while(1)
80     {
81         //调用app_demo_pwm
82         app_demo_pwm();
83         usleep(5000000);
84     }
85 }
86
87 void pwm_demo(void)
88 {
89     osThreadAttr_t attr;
90
91     attr.name = "PWM_Task";
92     attr.attr_bits = 0U;
93     attr.cb_mem = NULL;
94     attr.cb_size = 0U;
95     attr.stack_mem = NULL;
96     attr.stack_size = 2048;
97     attr.priority = 26;
98
99     if (osThreadNew((osThreadFunc_t)PWM_Task, NULL, &attr) == NULL) {
100         printf("[PWM_Task] Failed to create PWM_Task!\n");
101     }
102 }
103
104
105 SYS_RUN(pwm_demo);
```

二、如何使用示例

3、修改头文件，删除掉原先的include 的头文件，然后添加如下通用头文件，同时修改BUILD.gn 文件：

```
src > applications > sample > wifi-iot > app > demo_pwm > C demo_pwm.c >
1  #include <stdio.h>
2  #include <unistd.h>
3  #include "ohos_init.h"
4  #include "cmsis_os2.h"
5
6  #include <hi_types_base.h>
7  #include <hi_early_debug.h>
8
9  #include <hi_pwm.h>
10
```

```
src > applications > sample > wifi-iot > app > demo_pwm > BUILD.gn
1  static_library("demo_pwm") {
2      sources = [
3          "demo_pwm.c"
4      ]
5
6      include_dirs = [
7          "//utils/native/lite/include",
8          "//kernel/liteos_m/components/cmsis/2.0",
9          "//base/iot_hardware/peripheral/interfaces/kits",
10         "//device/soc/hisilicon/hi3861v100/hi3861_adapter/hals/communication/wifi_lite/wifiservice",
11         "//device/soc/hisilicon/hi3861v100/hi3861_adapter/kal",
12     ]
13 }
```


二、如何使用示例

4、打开对应的宏：如果某个驱动对应的宏我们如果没有打开，那么我们可能还得修改usr_config.mk

文件，该文件通常路径为：device\hisilicon\hispark_pegasus\sdk_liteos\build\config\usr_config.mk

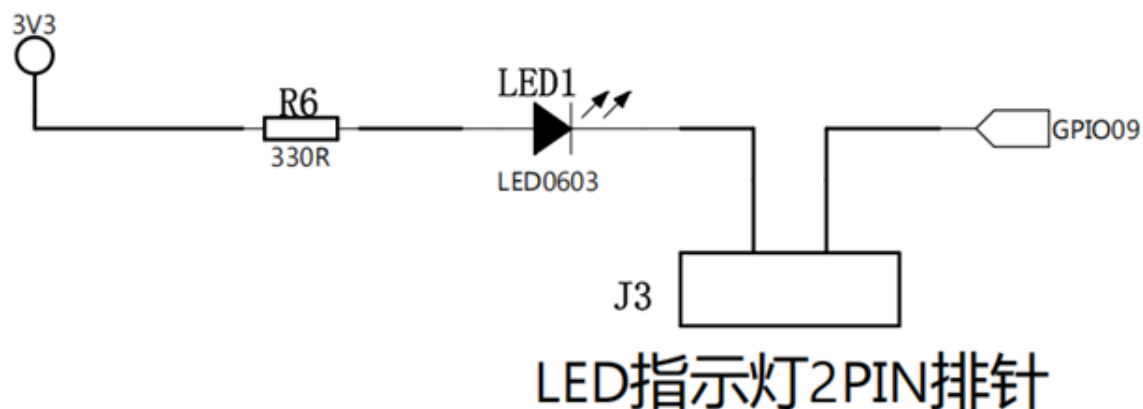
```
33  CONFIG_I2C_SUPPORT=y
34  # CONFIG_I2S_SUPPORT is not set
35  # CONFIG_SPI_SUPPORT is not set
36  # CONFIG_DMA_SUPPORT is not set
37  # CONFIG_SDIO_SUPPORT is not set
38  # CONFIG_SPI_DMA_SUPPORT is not set
39  # CONFIG_UART_DMA_SUPPORT is not set
40  # CONFIG_PWM_SUPPORT is not set
41  # CONFIG_PWM_HOLD_AFTER_REBOOT is not set
42  CONFIG_AT_SUPPORT=y
43  CONFIG_FILE_SYSTEM_SUPPORT=y
44  CONFIG_UART0_SUPPORT=y
45  CONFIG_UART1_SUPPORT=y
46  # CONFIG_UART2_SUPPORT is not set
47  # end of BSP Settings
```



```
src > device > hisilicon > hispark_pegasus > sdk_liteos > build > config > M usr_config.mk
37  # CONFIG_DMA_SUPPORT is not set
38  # CONFIG_SDIO_SUPPORT is not set
39  # CONFIG_SPI_DMA_SUPPORT is not set
40  # CONFIG_UART_DMA_SUPPORT is not set
41  CONFIG_PWM_SUPPORT=y
42  CONFIG_PWM_HOLD_AFTER_REBOOT=y
43  CONFIG_AT_SUPPORT=y
44  CONFIG_FILE_SYSTEM_SUPPORT=y
45  CONFIG_UART0_SUPPORT=y
46  CONFIG_UART1_SUPPORT=y
47  # CONFIG_UART2_SUPPORT is not set
48  # end of BSP Settings
```

5-1、实验，用PWM实现LED呼吸灯

实验内容：自由编程，通过PWM实现LED呼吸灯，实现led灯渐亮和渐灭。



四、WIFI基本概念

WiFi相关术语

IEEE： Institute of Electrical and Electronics Engineers，即电气和电子工程师协会。这是一个国际性的电子技术与信息科学工程师的协会。制定了超过900 个现行的工业标准。

IEEE 802： 是局域网/城域网标准委员会，是IEEE 下设的众多标准委员会之一。IEEE802 致力于研究局域网与城域网的介质访问控制和物理层规范（包括有线和无线），是全球范围内在该领域的倡导者。

IEEE 802 系列标准： IEEE 802 制定了一系列的标准，统称为IEEE 802 系列标准，是面向局域网和城域网的技术标准集。其中，最广泛使用的有以太网、令牌环网、无线局域网等。

IEEE 802.11： 定义了无线局域网（WLAN）的介质访问控制协议，以及物理层的技术规范。它涉及的具体标准也有很多： IEEE 802.11、 IEEE 802.11a、 IEEE 802.11b、 IEEE 802.11g、 IEEE 802.11n、 IEEE 802.11ac、 IEEE 802.11ax，主要区别在于技术的不断补充完善，速度越来越快

四、WIFI基本概念

SSID : Service Set Identifier, 即服务集标识。SSID 用于标识不同的网络, 长度为2 ~ 32 字节。平常我们说的 “你的Wi-Fi 名称是什么?” 指的其实就是SSID

AP : Wireless Access Point, 即无线接入点。AP 用于其他无线设备的连接, 相当于有线网络的交换机。AP 需要连接到路由器上才能接入上级网络, 很多时候AP 会和路由器整合到一个设备中, 比如家庭中使用的无线路由器。

STA : Station, 即工作站。IEEE 802.11 标准将其定义为 “支持IEEE 802.11 标准的设备”。因此, 从理论上来说, 所有的Wi-Fi 设备都可以被称为Station, 比如手机、电脑等, 也包括AP。

四、WIFI基本概念

BSS : Basic Service Set, 即基本服务集。BSS 由一个AP 和所有连接到这个AP 上的Wi-Fi 设备组成。连接到AP 上的Wi-Fi 设备也叫AP 客户端或Wi-Fi客户端。

BSSID : Basic Service Set Identifier, 即基本服务集标识。BSSID用于标识一个BSS, 通常是AP 的MAC 地址。

WEP: Wired Equivalent Privacy, 即有线等效保密。WEP 用于对两台无线设备间传输的数据进行加密。

WPA : Wi-Fi Protected Access, 即Wi-Fi 保护访问。这是一种保护无线网络访问安全的技术, 目前有WPA、WPA2 和WPA3 三个标准

四、WIFI基本概念

PSK：Pre-Shared Key，即预共享密钥。PSK 一般在家庭或小型无线网络中使用，用户输入事先约定好的密钥接入网络。密钥的长度为8 ~ 63 个ASCII字符，或64 个16 进制数。

Band：频段，也就是频率范围。无线网络使用无线电波进行通信，IEEE802.11 标准定义了2.4GHz、3.6GHz、4.9GHz ~ 5.8GHz 等不同的频段。频率高容易导致反射，穿透能力就弱；频率低，穿透能力会强一些。OpenHarmony 定义了2.4GHz 和5GHz两种频段。

Channel：信道，指的是无线网络数据传输的频道。每个频段都被划分为若干个信道。

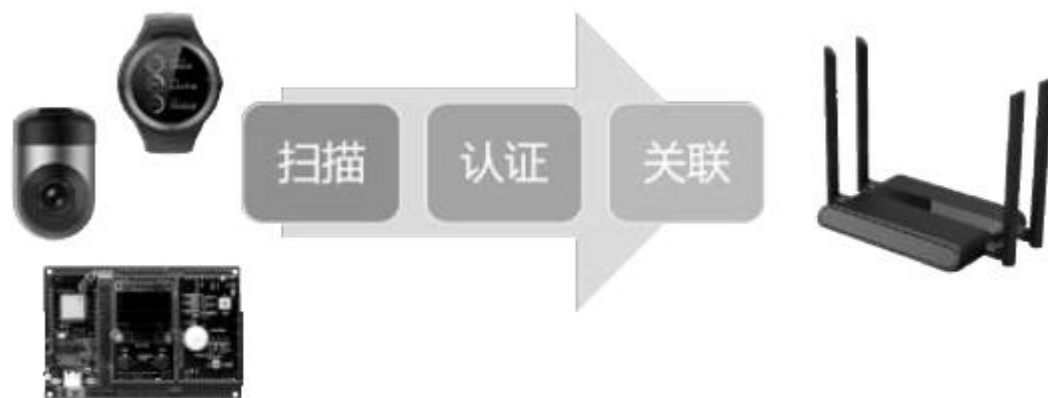
四、WIFI基本概念

Hi3861V100芯片的Wi-Fi 特性， 特性如下：

- (1) 支持IEEE 802.11b/g/n， 最大速率为72.2Mb/s。
- (2) 工作在2.4GHz 频段， 支持全部14 个信道ch1 ~ ch14。
- (3) 支持标准20MHz 带宽和5MHz / 10MHz 窄带宽。
- (4) 不支持40MHz 带宽。
- (5) 单收单发， 不支持MIMO。
- (6) 支持WPA 个人版/WPA2 个人版和WPS（Wi-Fi Protected Setup， Wi-Fi 保护设置） 2.0。
- (7) 支持STA 和AP 模式。
- (8) 作为AP 时， 最大支持6 个STA 接入。

五、Wi-Fi 的连接过程

Wi-Fi 的连接过程，就是将Wi-Fi 客户端和AP（或者路由器）组成一个无线局域网的过程。这个过程分为3 个阶段，分别是扫描阶段、认证阶段和关联阶段。



五、Wi-Fi 的连接过程

Wi-Fi 扫描： Wi-Fi 客户端发现AP 或路由器的过程。它有两种不同的方式：

- 1、主动扫描 (Active Scan): Wi-Fi 客户端在每个信道上都发送探测请求帧，AP或路由器在收到探测请求之后，返回探测响应。
- 2、被动扫描 (Passive Scan): Wi-Fi 客户端在每个信道上都监听AP 或路由器发出的信标帧。

认证： 扫描完成后，需要调用相应的API 获取扫描结果，再根据SSID 选择一个AP 或路由器进行连接。目前，主要使用的认证标准是WPA 或WPA2，最新的标准是WPA3。

关联： 将Wi-Fi 客户端注册到AP 或路由器的过程。Step1： Wi-Fi 客户端发送关联请求帧； step2： AP 或路由器处理关联请求。如果允许关联，就返回0（表示成功）， 否则返回状态码。

五、WiFi 工作模式

Wi-Fi 设备可以工作在不同的模式中，在每个模式中扮演的角色都不同。常用的Wi-Fi 工作模式有两种：一种是STA 模式，另一种是AP 模式。一个Wi-Fi 设备可以同时支持多种模式，需要通过程序代码让Wi-Fi 设备处于指定的模式中。

STA模式： 具有Wi-Fi 客户端行为的设备，可以连接到AP 或路由器上，如手机、电脑、边缘设备、汽车等，STA 会扫描附近的AP 或路由器，选择其中一个想要连接的，经过认证、关联等步骤之后，就与它建立了连接。

AP 模式： 无线接入点模式，允许其他Wi-Fi 客户端与之建立连接，并且提供无线网络服务。

五、WIFI相关API介绍

Hi3861 提供了非常多的wifi 相关API，主要文件是hi_wifi_api.h。这里只列举最重要的几个API：

(1) 开启STA

```
int hi_wifi_sta_start(char *ifname, int *len);
```

(2) 停止STA

```
int hi_wifi_sta_stop(void);
```

(3) 扫描附件的热点

```
int hi_wifi_sta_scan(void);
```

(4) 连接热点

```
int hi_wifi_sta_connect(hi_wifi_assoc_request *req);
```

通常加密方式是：HI_WIFI_SECURITY_WPA2PSK

5.2、实验，连接自己手机热点

任务：开启手机热点，开发板连接上自己的热点，通过串口打印出连接状态和连接上的WIFI名称。

思路：参照上个实验的开发示例流程，编写demo_wifi_sta.c 业务代码和BUILD.gn 编译脚本，通过wifi api实现wifi连接和串口打印。