

广东新岸线计算机系统芯片有限公司

Guangdong Nufront CSC Co., Ltd

NL6621-Gizwits SDK

用户手册

林辉

2015 年 06 月 3 日

[文中提到的所有商标名称、商标和注册商标均属其各自所有者的财产，特此声明。版权归
©2015 广东新岸线计算机系统芯片有限公司所有。保留所有权利。]

Change Log

Date	Version	Types (New/Delete/ Modify)	Editor	Description
2015-02-25	0.01.01	New	林辉	完成文档基本框架
2015-02-28	0.01.00	Modify	林辉	1、升级 NL6621 的固件版本为 V1.08.00, DirectConfig 采用组播地址发送数据。
2015-03-10	0.02.00	Modify	林辉	1、增加 Gizwits 的配置说明
2015-03-10	0.02.00	Modify	赵伟翔	1、增加智能灯范例
2015-03-27	0.03.00	Modify	林辉	1、更新 NL6621 的固件版本为 V1.08.00
2015-05-21	0.04.00	Modify	赵伟翔	1、更新 NL6621 的固件版本为 V1.10.00
2015-06-03	0.04.00	Modify	赵伟翔	1、修改文档说明，SDK 默认为智能插座

目录

1. 引言.....	4
1.1 概述.....	4
2. SDK 基础信息.....	5
2.1 产品版本号.....	5
2.2 LED 指示灯与配置按键.....	5
2.3 配置模式.....	5
2.3.1 SoftAP.....	5
2.3.2 DirectConfig.....	6
2.4 系统资源.....	6
3. 软件描述汇整.....	7
3.1 系统框架.....	7
3.2 SDK 目录结构.....	8
3.3 日志系统.....	8
4. 配置模式说明.....	10
4.1 配置过程.....	10
4.1.1 配置 SoftAP 模式连接 AP.....	10
4.1.2 配置 DirectConfig 模式连接 AP.....	12
4.2 配置异常说明.....	12
4.2.1 设备连接 AP 失败.....	12
5. 测试范例.....	13
5.1 前期准备.....	13
5.2 智能灯与智能插座切换.....	15
5.3 智能插座.....	15
5.3.1 智能插座硬件说明.....	15
5.3.2 智能插座需求分析.....	15
5.3.3 智能插座数据点.....	16
5.3.4 协议栈解析.....	16

5.3.5	相关 APP 测试.....	18
5.4	智能灯	18
5.4.1	智能灯硬件说明.....	18
5.4.2	智能灯需求分析.....	18
5.4.3	智能灯数据点.....	19
5.4.4	协议栈解析.....	19
5.4.5	相关 APP 测试.....	21
5.4.6	智能灯使用说明.....	21
6.	注意事项.....	26

1. 引言

1.1 概述

本文描述“NL6621+机智云”软件开发包（SDK）的功能与使用方法。该 SDK 主要支持集成了 NL6621 的 SDK 软件包，机智云本地以及云端通信的协议栈，智能插排与智能灯的 Demo。

由于 Gizwits 目前提供方案为“MCU+WIFI 模组”的方式定义相关协议栈，而 NL6621 提供单芯片集成了 MCU 和 WIFI 两者功能。因此 Gizwits 中提供的与 MCU 相关的协议在 NL6621 中已不再适用。

2. SDK 基础信息

2.1 产品版本号

NL6621 SDK 版本号：V1.09.00 （NL6621 基础 SDK 版本号）

硬件版本号：06NL6621

软件版本号：04010001

Firmware 版本号：15032717

Firmware 版本号用于描述 SDK 软件发布的时间，共 8 字节的用日期和时间来描述；
格式为：年（2 位）月（2 位）日（2 位）时（2 位 24 小时格式），比如：14112521，
表示 2014 年 11 月 25 日 21 点发布的版本。

固件发布名称：GAgent_NL6621_04010001_15032717_RAMDBG.bin

格式为：GAgent_NL6621+软件版本号+Firmware 版本号+Ram/Rom 版本。

2.2 LED 指示灯与配置按键

NL6621 需要两个 GPIO 作为系统基本外设：一个 LED 指示灯以及一个触发按键：

1、触发按键（GPIO 10）：系统复位以及 DirectConfig 模式切换，其中系统上电启动阶段按下该按键系统进入 DirectConfig 模式。系统完成初始化后，经过 7s 才能正常的使能系统复位功能。长按 4s 系统复位。

2、LED 指示灯（GPIO 9）：作为 LED 指示灯所用

- 1) 模组处于 SoftAP： 指示灯灭 3 秒，亮 1 秒
- 2) 模组处于 DirectConfig： 指示灯以 100 毫秒快闪
- 3) 模组连接 AP 成功，但没有连接云端： 指示灯常亮 1 秒，灭 1 秒
- 4) 模组成功接入云端： 指示灯常灭
- 5) WIFI 停止工作： 指示灯常亮

2.3 配置模式

NL6621 支持 2 种配置方式，分别是 SoftAP、Directconfig；

2.3.1 SoftAP

1、SoftAP 配置信息

热点名称（SSID）：XPG-GAgent-XXXX，XXXX 是 mac 地址后四位；

密码：123456789；

IP 地址: 10.10.100.254

UDP 配置端口: 12414

2、SoftAP 配置过程:

- 1) 设备第一次启动, 默认进入 softap 模式;
- 2) 等待 APP 端的连接, 接收 SSID 以及 Password, 并且换到 STA 模式并连上 AP;
- 3) 设备连接成功, 则保存 SSID 以及 Password 到 norflash 并重启系统。
- 4) 系统重新启动之后, 从 norflash 中读取保存的 SSID 以及 Password 并连接到 AP;

注: 如果之前保存的 AP 无法连接, 系统会尝试连接 10 次, 10 次连接失败后, 切换回 SoftAP 模式。

2.3.2 DirectConfig

1、DirectConfig 配置信息

接收方式: 广播

2、DirectConfig 配置过程

- 1) 系统上电启动时, 按下 DirectConfig 按键 2s
- 2) LED 指示灯以 100ms 间隔进行闪烁为进入 DirectConfig 模式。
- 3) 获取到 SSID 以及 Password 之后开始连接 AP, 以 1s 间隔进行闪烁。
- 4) 连接成功后 LED 指示灯灭。并保存获取到的 SSID 以及 Password。

注:

1、与 SoftAP 配置模式一样, 系统尝试连接 10 次, 10 次连接失败后, 切换回 SoftAP 模式。

2、由于 NL6621 的基础 SDK 从 V1.08.00 开始, 由原有的广播方式改为组播的方式, 因此低于 V1.08.00 以下版本的 DirectConfig APK 将无法兼容。

2.4 系统资源

支持机智云基本协议栈以及系统代码的 bin 文件大小:

ROM 工程: 140KBytes

RAM 工程: 170KBytes

留给客户的 Code SRam 空间为:

ROM 工程: 52KBytes

RAM 工程: 22KBytes

3. 软件描述汇总

3.1 系统框架

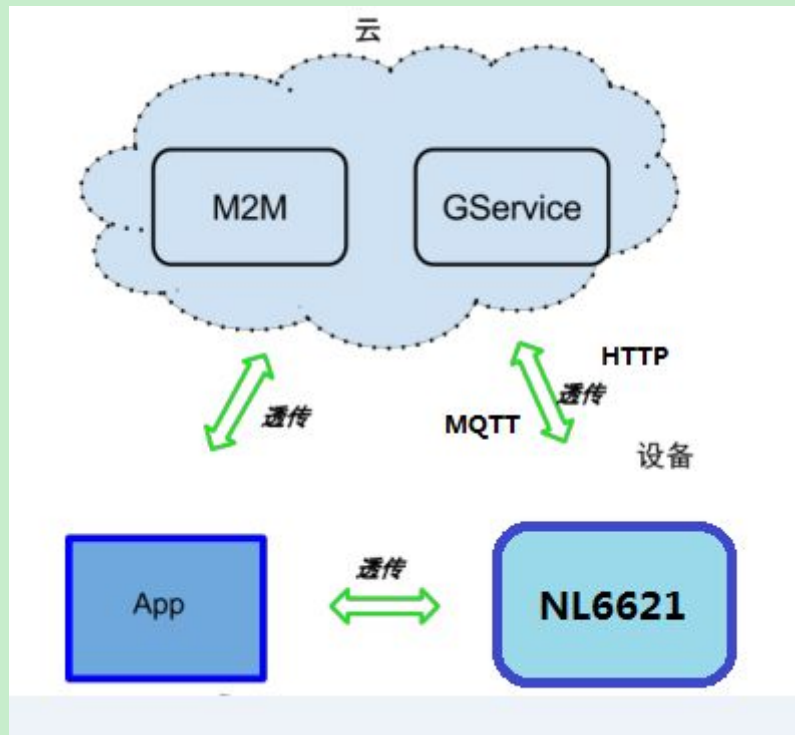


Figure 1 NL6621+Gizwits 系统框架图

设备与云端通讯采用 MQTT 和 HTTP 两种方式。HTTP 方式主要是提供设备注册，设备 Provision，OTA 升级等接口。而 MQTT 提供设备上报状态及设备接收远程控制指令的通道。MQTT 是建立在 TCP 基础上的长连接通讯协议。APP、云以及 NL6621 之间的相互通信目前采用的都是透传的模式进行通信。

3.2 SDK 目录结构

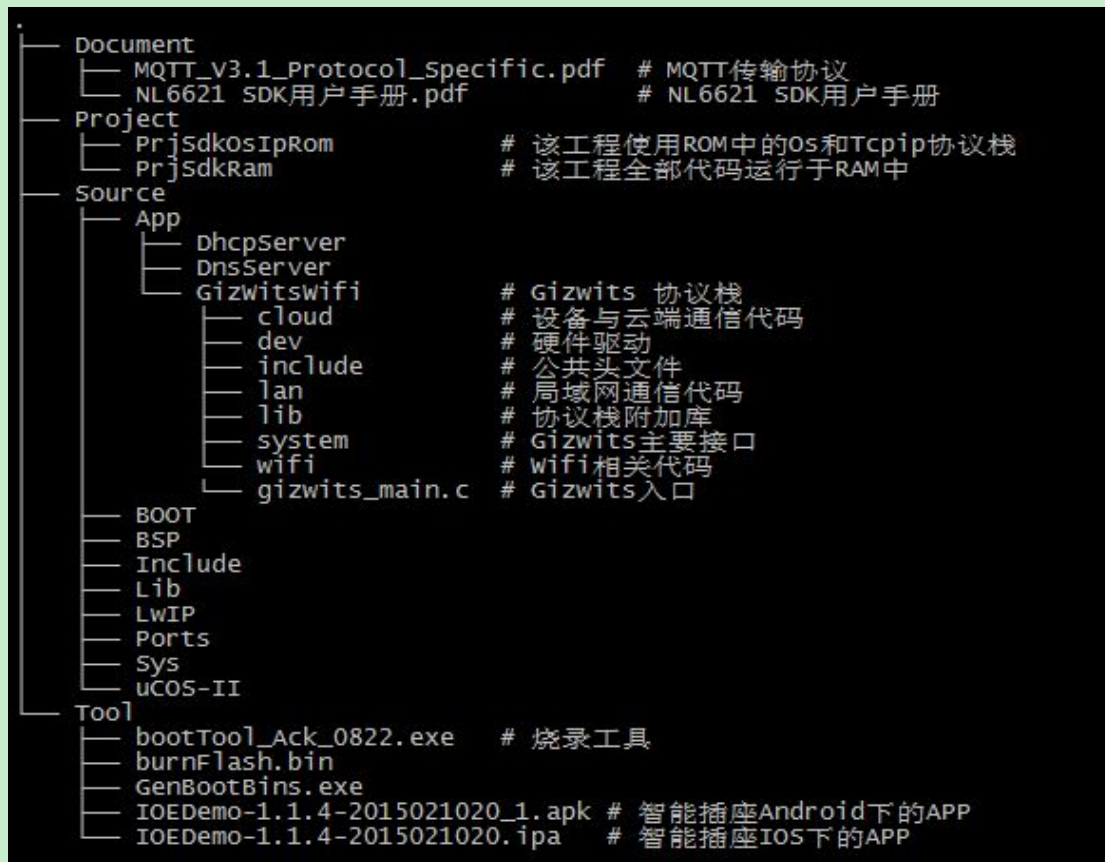


Figure 2 SDK 工程目录

“NL6621+Gizwits”SDK 工程目录如上图所示：

Document 目录存放工程文档；

Project 目录存放 ROM/RAM 工程、

Source 目录存放 SDK 源码

Tool 存放编译、固件、APP 以及烧录工具

其中 SDK 源码中 Source/App/GizwitsWifi 目录存放所有的 Gizwits 工程代码。

3.3 日志系统

为了减少固件占用 code Sram 中的空间，需要优化 Gizwits 在 NL6621 中的代码。在优化的过程中发现系统的 debug 信息占用了大量的代码空间，因此对原有的 Gizwits 协议栈的 debug 系统进行了优化，定义了一套新的日志系统。

相关代码位于 Source/App/GizwitsWifi/include/log.h 文件。自定义 5 个级别的日志信息输出，分别为 ERROR,WARNING,NOTICE,INFO,DEBUG。使用宏 DEBUG_LEVEL_SWITCH 作为开关，开启相关级别的打印信息。该级别信息在编译的时候决定相关级别的 debug 信息是否编进固件。

日志系统的各个级别信息使用规范：

log_err：系统错误打印

log_warn: 系统警告打印

log_notice: 系统运行必要信息打印

log_info: 系统运行状态信息打印

log_debug: 系统调试信息打印

以上 5 个级别的打印中,log_err、log_warn 和 log_notice 级别信息默认打开,info 和 debug 级别信息默认关闭 (DEBUG_LEVEL_SWITCH 值为 0x07)。

注: 在开发正式的产品时,最终的产品应该屏蔽无用的打印信息,只保留 log_err 级别信息即可。

4. 配置模式说明

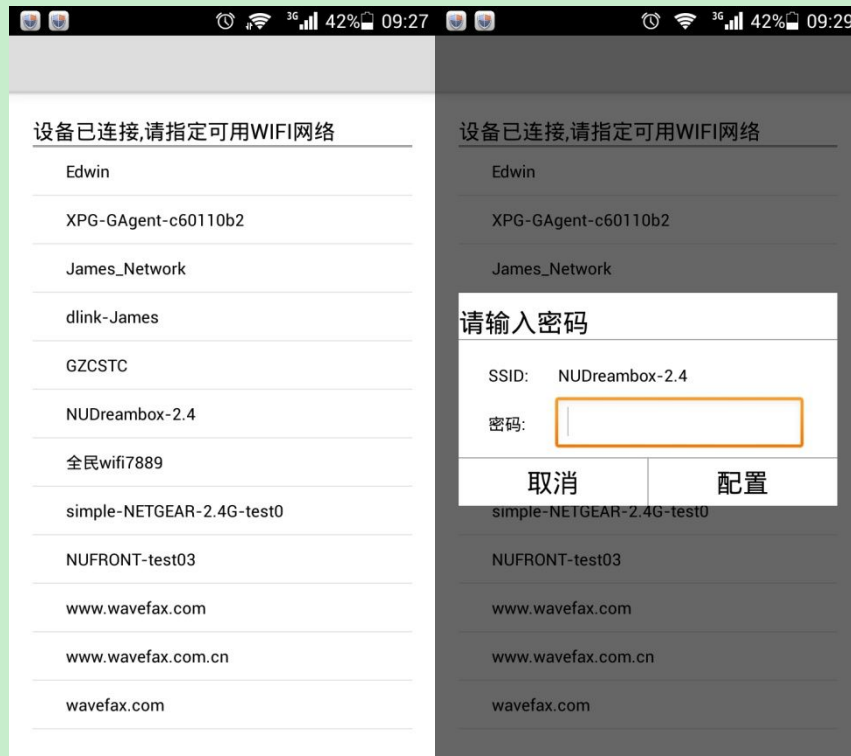
4.1 配置过程

4.1.1 配置 SoftAP 模式连接 AP

1) 启动烧录 SDK 后的设备，如果是第一次烧录，nor flash 中没有保存任何 AP 的信息，并且可能出现连接 AP 行为，此时需要多重启几次，等到设备进入 SoftAP 模式。

```
SDK Version:<1.08.00>; GAgent Version:<1.02.01> Release:<dbg>
=====
Entry gizwits initialize(time:0(s); 70(ms)).
Timer value:100.
GAgent Version: B4R012D0.
Product Version: 06NL6621.
MCU WiFi MAC:000ec600c899
MCU Protocol Version:00000001.
MCU P0 Version:00000004.
MCU Hard Version:00000001.
MCU Soft Version:00000001.
MCU Product Version:43de75d17b4a4d8eae41738df92138b6.
Create system LED indicator task success.
Reset GAgent config data.
MQTT_STATUS_START
passcode:KIOHAWLSEP(1en:10)
Connect2LateWiFi, working at AP mode
(237)USE STATIC IP ADDRESS:10.10.100.254 !!!
SYS_EVT_LINK_UP
Create UDP Broadcast socket success.
Create TCP server socket success.
Create NTP request thread success.
Create UDP server socket success.
Create GAgent reset task success.
Create cloud task success.
    Send NTP package success.
    Send NTP package success.
```

2) 设备进入 SoftAP 模式后，手机端打开对应 APK（这里以智能灯为例），进入配置 SoftAP 配置界面



3) 设备端收到配置信息，自动根据收到的 SSID 和 Password 保存并重启设备，重启后连接到 AP。

```

SDK Version:<1.08.00>; GAgent version:<1.02.01> Release:<dbg>
=====
Entry gizwits initialize(time:0(s); 70(ms)).
Timer value:100.
GAgent Version: B4R012D0.
Product Version: 06NL6621.
MCU WiFi MAC:000ec600c899
MCU Protocol Version:00000001.
MCU P0 Version:00000004.
MCU Hard Version:00000001.
MCU Soft Version:00000001.
MCU Product Version:43de75d17b4a4d8eae41738df92138b6.
with DID:T9jCq54NaNhdXYwhrpg4 len:22
MQTT_STATUS_PROVISION
passcode:KIOHAWLSEP(len:10)
Try to connect wifi...[00000001]
SSID: NUDreambox-2.4
KEY: google.cn
Create system LED indicator task success.
Connecting to AP(NUDreambox-2.4,google.cn)...
(574)[DHCP] MY IP ADDRESS:192.168.1.30
SYS_EVT_LINK_UP
Create UDP Broadcast socket success.
Create TCP server socket success.
Create NTP request thread success.
Create UDP server socket success.
Create GAgent reset task success.
Create cloud task success.
New Http socketID [4]

```

4) 连接设备成功后，登录机智云云端服务

TBD

5) 设备端连接到 AP，刷新设备列表



6) 绑定设备



4.1.2 配置 DirectConfig 模式连接 AP

TBD

4.2 配置异常说明

4.2.1 设备连接 AP 失败

1) 输入 AP 密码错误，导致设备连不上 AP，设备会在尝试 5 次联网失败后自动进入 SoftAP 模式。

```
=====
Entry gizwits_initialize(time:0(s); 70(ms)).
Timer value:100.
GAgent Version: B4R012D0.
Product Version: 06NL6621.
MCU WiFi MAC:000ec600c899
MCU Protocol Version:00000001.
MCU P0 Version:00000004.
MCU Hard Version:00000001.
MCU Soft Version:00000001.
MCU Product Version:43de75d17b4a4d8eae41738df92138b6.
Create system LED inReset GAgent config data.
MQTT_STATUS_START
passcode:KIOHAWLSEP(len:10)
Try to connect wifi...[00000001]
SSID: NUDreambox-2.4
KEY: google.cn
indicator task success.
Connecting to AP(NUDreambox-2.4,google.cn)...
SYS_EVT_JOIN_FAIL
Connect2LateWifi, working at AP mode
(6455)USE STATIC IP ADDRESS:10.10.100.254 !!!
SYS_EVT_LINK_UP
Create UDP Broadcast socket success.
Create TCP server socket success.
Create NTP request thread success.
Create UDP server socket success.
Create GAgent reset task success.
Create cloud task success.
Send NTP package success.
```

5. 测试范例

本 SDK 提供了智能插座以及智能灯的测试范例（位于 Source/App/GizwitsWifi/dev 目录中，如下图所示），可以从局域网以及广域网对设备进行控制测试。

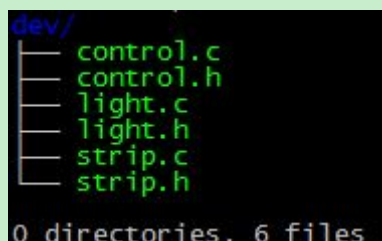


Figure 3 Gizwits 测试代码

上图中 control.c 为设备控制的数据接入源代码，light.c 为智能灯控制源代码，strip.c 为智能插座的控制源代码。

5.1 前期准备

相关示例代码，需要按照 Gizwits 的协议文档进行配置才能正常的运行。用户需要完成以下几步：

- 1、注册 Gizwits 的账号并登陆；
- 2、进入“个人项目”，并点击“新建设备接入”，采用相关模板（例如机智云插座模板）或者自定义数据点。
- 3、生成相关设备之后，在“产品信息”中可以获得 APP ID 以及产品标识码(Product key)

APP ID : 1efb69ea6ae349edb6224d915750731e

产品标识码 (字符串) : 1d9a35123eaf4fe0bb6e1a1d8cd7cb52

- 4、将获取到的 Product Key 填入（ Source/App/GizWitsWifi/lan/mcu_v4.c ） GAgent_GetMCUInfo 接口中的 product_key。

```
011
012 int GAgent_GetMCUInfo( u32 Time )
013 {
014     memcpy( g_Xpg_GlobalVar.Xpg_Mcu.protocol_ver, "00000001", 8 );
015     g_Xpg_GlobalVar.Xpg_Mcu.protocol_ver[8] = '\0';
016
017     memcpy( g_Xpg_GlobalVar.Xpg_Mcu.p0_ver, "00000004", 8 );
018     g_Xpg_GlobalVar.Xpg_Mcu.p0_ver[8] = '\0';
019
020     memcpy( g_Xpg_GlobalVar.Xpg_Mcu.hard_ver, "00000001", 8 );
021     g_Xpg_GlobalVar.Xpg_Mcu.hard_ver[8] = '\0';
022
023     memcpy( g_Xpg_GlobalVar.Xpg_Mcu.soft_ver, "00000001", 8 );
024     g_Xpg_GlobalVar.Xpg_Mcu.soft_ver[8] = '\0';
025
026     memcpy( g_Xpg_GlobalVar.Xpg_Mcu.product_key, "1d9a35123eaf4fe0bb6e1a1d8cd7cb52", 32 );
027     g_Xpg_GlobalVar.Xpg_Mcu.product_key[32] = '\0';
028     g_productKeyLen = 32;
029
030     log_notice("MCU Protocol Version:%s.\n", g_Xpg_GlobalVar.Xpg_Mcu.protocol_ver);
031     log_notice("MCU P0 Version:%s.\n", g_Xpg_GlobalVar.Xpg_Mcu.p0_ver);
032     log_notice("MCU Hard Version:%s.\n", g_Xpg_GlobalVar.Xpg_Mcu.hard_ver);
033     log_notice("MCU Soft Version:%s.\n", g_Xpg_GlobalVar.Xpg_Mcu.soft_ver);
034     log_notice("MCU Product Version:%s.\n", g_Xpg_GlobalVar.Xpg_Mcu.product_key);
035
036     return 0;
037 }
```


5、下载 APP 源码，下载地址：“<https://github.com/gizwits/gokit-android>”

6、将获取到的 APP ID 填写到 app 源码文件 WApplication.java 中 startWithAppID 接口，如下图所示：

```
/**
 * The Class WApplication.
 */
public class WApplication extends Application {

    /** (non-Javadoc)
     * @see android.app.Application#onCreate()
     */
    public void onCreate() {
        super.onCreate();

        try {
            //复制assert文件夹中的json文件到设备安装目录，json文件是解析数据点必备的文件，sdk根据该文件，把二进制或
            AssertsUtils.copyAllAssertToCacheFolder(this.getApplicationContext());
        } catch (IOException e) {
            e.printStackTrace();
        }

        // 启动SDK
        XPGWifiSDK.sharedInstance().startWithAppID(getApplicationContext(), "1efb69ea6ae349edb6224d915750731e");
        // "6f3074fe43894547a4f1314bd7e3ae0b");
        // 设定日志打印级别
        XPGWifiSDK.sharedInstance().setLogLevel(XPGWifiSDK.XPGWifiLogLevel.XPGWifiLogLevelAll, "GoKitDemo.log", true);
    }
}
```

7、把 Product Key 填写到 app 源码文件 MessageCenter.java 中接口 xpgWifiGCC.getBoundDevices，如下图所示：

```
public void cGetBoundDevices(String uid, String token) {
    xpgWifiGCC.getBoundDevices(uid, token, "1d9a35123eaf4fe0bb6e1a1d8cd7cb52");
}
```

8、点击“产品开发资源”，获取并安装相关 APP，分别支持 IOS 以及 Android 系统。如果用户使用 SDK 库调用的方式，则需要按照 <http://site.gizwits.com/document/m2m/> 目录下的开发指南进行开发。

至此，用户可以简单的完成相关的设备控制，用户也可以到 github 下载相关公版 APK 进行相关代码的研发。Github 的相关下载路径为：

<https://github.com/gizwits>

用户可以通过该路径下载最新的 Gizwits 相关公版代码。

5.2 智能灯与智能插座切换

- 1、本 SDK 默认为智能插座，如需编译智能灯，需要在（Gizwits_SDK/Source/App/GizWitsWifi/dev/contro.h）将智能灯宏打开，把智能插座宏注释。如下图所示：

```
*/
#ifdef __CONTROL_H__
#define __CONTROL_H__

// #define DEV_LIGHT_MASK
#define DEV_STRIP_MASK

#include "../include/common.h"
#include "light.h"
#include "strip.h"

extern void device_init(void);
extern void device_control(unsigned char action, unsigned char attr_flags, unsigned char *ctrl_data);

#endif /* ----- #ifndef __CONTROL_H__ ----- */
```

- 2、将智能插座的 Product Key 填入到（Gizwits_SDK/Source/App/GizWitsWifi/lan/mcu_v4.c）GAgent_GetMCUInfo 接口中的 product_key。
智能灯 product_key: c853fcbddfb64fc796ac22cf975708ae
智能插座 product_key: 9877965c9cf04dff9a585e5f8d77dedb
如下图所示：

```
extern int g_SocketLogin[8];

int GAgent_GetMCUInfo( u32 Time )
{
    memcpy( g_Xpg_GlobalVar.Xpg_Mcu.protocol_ver, "00000001", 8 );
    g_Xpg_GlobalVar.Xpg_Mcu.protocol_ver[8] = '\0';

    memcpy( g_Xpg_GlobalVar.Xpg_Mcu.p0_ver, "00000004", 8 );
    g_Xpg_GlobalVar.Xpg_Mcu.p0_ver[8] = '\0';

    memcpy( g_Xpg_GlobalVar.Xpg_Mcu.hard_ver, "00000001", 8 );
    g_Xpg_GlobalVar.Xpg_Mcu.hard_ver[8] = '\0';

    memcpy( g_Xpg_GlobalVar.Xpg_Mcu.soft_ver, "00000001", 8 );
    g_Xpg_GlobalVar.Xpg_Mcu.soft_ver[8] = '\0';

    memcpy( g_Xpg_GlobalVar.Xpg_Mcu.product_key, "9877965c9cf04dff9a585e5f8d77dedb", 32 );
    g_Xpg_GlobalVar.Xpg_Mcu.product_key[32] = '\0';
    g_productKeyLen = 32;

    log_notice("MCU Protocol Version:%s.\n", g_Xpg_GlobalVar.Xpg_Mcu.protocol_ver);
    log_notice("MCU P0 Version:%s.\n", g_Xpg_GlobalVar.Xpg_Mcu.p0_ver);
    log_notice("MCU Hard Version:%s.\n", g_Xpg_GlobalVar.Xpg_Mcu.hard_ver);
    log_notice("MCU Soft Version:%s.\n", g_Xpg_GlobalVar.Xpg_Mcu.soft_ver);
    log_notice("MCU Product Version:%s.\n", g_Xpg_GlobalVar.Xpg_Mcu.product_key);
}
```

5.3 智能插座

5.3.1 智能插座硬件说明

本智能插座使用 SPI_DO（GPIO30）控制插座开关。

5.3.2 智能插座需求分析

Table 1 智能插座功能需求

功能	需求点	需求描述
控制功能	开关机	开机、关机设置；
	定时开机	设置具体开机时间，设备端时钟与手机端时钟同步，由设备端控制定时响应；时间 step=1 min；（时钟指令）
	定时关机	设置具体关机时间，设备端时钟与手机端时钟同步，由设备端控制定时响应；step=1 min；
	延时预约	倒计时预约，设备端时钟与手机端时钟同步，由设备端控制定时响应；预约时间到后，改变设备开关机状态；step=1 min；
	用电量统计	返回设备用电量统计；（该功能暂时不支持）

5.3.3 智能插座数据点

接口名称	显示名称	类型	权限
OnOff	开关	bool	W
Week_Repeat	每周重复	uint8	W
Time_On_Minute	定时开机	uint16	W
Time_Off_Minute	定时关机	uint16	W
CountDown_Minute	倒计时	uint16	W
Power_Consumption	能耗	uint16	R
Time_OnOff	是否启用定时	bool	W
CountDown_OnOff	是否启用倒计时	bool	W

5.3.4 协议栈解析

1、控制设备

APP 发给 NL6621 的控制设备数据格式：

header(2B)	len(2B)	cmd(1B)	sn(1B)	flags(2B)	action(1B)	attr_flags(1B)	attr_vals(8B)	checksum(1B)
0xFFFF	0x000F	0x03	0x##	0x0000	0x01	是否设置标志位	设置数据值	0x##

注：

1. 是否设置标志位(attr_flags)表示相关的数据值是否为有效值，相关的标志位为 1 表示值有效，为 0 表示值无效，

从右到左的标志位依次为：

bit0: 设置 OnOff

bit1: 设置 Time_OnOff

bit2: 设置 Countdown_OnOff

bit3: 设置 Week_Repeat

bit4: 设置 Time_On_Minute

bit5: 设置 Time_Off_Minute

bit6: 设置 Countdown_Minute

2. 设置数据值(attr_vals)存放数据值，只有相关的设置标志位为 1 时，数据值才有效。

例如数据包为

0x07 FF 05 A0 05 A0 05 A0 时，其格式为：

字节序	bit序	数据内容	说明
byte0	bit7 bit6 . . . bit1 bit0	0b00000111	OnOff, 类型为bool, 值为true; 字段bit0, 字段值为0b1; Time_OnOff, 类型为bool, 值为true; 字段bit1, 字段值为0b1; CountDown_OnOff, 类型为bool, 值为true; 字段bit2, 字段值为0b1;
byte1		0xFF	Week_Repeat, 类型为uint8, 字段值为255; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为255
byte2 byte3		0x05 A0	Time_On_Minute, 类型为uint16, 字段值为1440; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为1440
byte4 byte5		0x05 A0	Time_Off_Minute, 类型为uint16, 字段值为1440; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为1440
byte6 byte7		0x05 A0	CountDown_Minute, 类型为uint16, 字段值为1440; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为1440

2、上报设备信息

NL6621 设备向 APP 上报控制设备数据格式

header (2B)	len (2B)	cmd (1B)	sn (1B)	flags (2B)	action (1B)	dev_status (10B)	checksum (1B)
0xFFFF	0x0010	0x05	0x##	0x0000	0x04	设备状态	0x##

注：

1. 设备状态(dev_status)使用一个或多个字节表示。例如数据包为

0x07 FF 05 A0 05 A0 05 A0 FF FF 时，其格式为：

字节序	位序	数据内容	说明
byte0	bit7 bit6 . . . bit1 bit0	0b00000111	OnOff, 类型为bool, 值为true: 字段bit0, 字段值为0b1; Time_OnOff, 类型为bool, 值为true: 字段bit1, 字段值为0b1; CountDown_OnOff, 类型为bool, 值为true: 字段bit2, 字段值为0b1;
byte1		0xFF	Week_Repeat, 类型为uint8, 字段值为255; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为255
byte2 byte3		0x05 A0	Time_On_Minute, 类型为uint16, 字段值为1440; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为1440
byte4 byte5		0x05 A0	Time_Off_Minute, 类型为uint16, 字段值为1440; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为1440
byte6 byte7		0x05 A0	CountDown_Minute, 类型为uint16, 字段值为1440; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为1440
byte8 byte9		0xFF FF	Power_Consumption, 类型为uint16, 字段值为65535; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为65535

5.3.5 相关 APP 测试

用户可以直接下载 Gizwits 自动生成的 APP 进行测试，也可以参考提供到 Github 上的
公版代码进行二次开发。详细资料请参考 Gizwits 文档中心
（<http://site.gizwits.com/document/m2m/>）。

5.4 智能灯

5.4.1 智能灯硬件说明

本智能灯使用管脚 SMID_DATA1（GPIO6）连接红灯，SMID_DATA2（GPIO7）连接
绿灯，SMID_DATA0（GPIO5）连接蓝灯，SMID_DATA3（GPIO8）连接白灯。如用户需要
模拟 PWM，建议使用速度较快的 I2C 或 I2S 这两组引脚。

5.4.2 智能灯需求分析

Table 2 智能灯功能需求

功能	需求点	需求描述
控制功能	红光亮度	红光亮度调节, 范围为 0-255, 0 为灭灯, 255 为最大亮度。

	蓝光亮度	蓝光亮度调节, 范围为 0-255, 0 为灭灯, 255 为最大亮度。
	绿光亮度	绿光亮度调节, 范围为 0-255, 0 为灭灯, 255 为最大亮度。
	白光亮度	白光亮度调节, 范围为 0-255, 0 为灭灯, 255 为最大亮度。

5.4.3 智能灯数据点

接口名称	显示名称	类型	权限
socket	socket	bool	W
led_red	led_red	uint8	W
led_blue	led_blue	uint8	W
led_green	led_green	uint8	W
led_white	led_white	uint8	W

5.4.4 协议栈解析

1、控制设备

APP 发给 NL6621 的控制设备数据格式:

header(2B)	len(2B)	cmd(1B)	sn(1B)	flags(2B)	action(1B)	attr_flags(1B)	attr_vals(8B)	checksum(1B)
0xFFFF	0x000C	0x03	0x##	0x0000	0x01	是否设置标志位	设置数据值	0x##

注:

1. 是否设置标志位(attr_flags)表示相关的数据值是否为有效值, 相关的标志位为 1 表示值有效, 为 0 表示值无效,

从右到左的标志位依次为:

bit0: 设置 socket

bit1: 设置 led_red

bit2: 设置 led_blue

bit3: 设置 led_green

bit4: 设置 led_white

2. 设置数据值(attr_vals)存放数据值, 只有相关的设置标志位为 1 时, 数据值才有效。

例如数据包为

0x01 FF FF FF FF 时, 其格式为:

字节序	位序	数据内容	说明
byte0	bit7 bit6 . . . bit1 bit0	0b00000001	socket, 值为true: 字段bit0, 字段值为0b1;
byte1		0xFF	led_red字段值为255; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为255
byte2		0xFF	led_green字段值为255; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为255
byte3		0xFF	led_blue字段值为255; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为255
byte4		0xFF	led_white字段值为255; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为255

2、上报设备信息

NL6621 设备向 APP 上报控制设备数据格式

header (2B)	len (2B)	cmd (1B)	sn (1B)	flags (2B)	action (1B)	dev_status (5B)	checksum (1B)
0xFFFF	0x000B	0x05	0x##	0x0000	0x04	设备状态	0x##

注:

- 1. 设备状态(dev_status)使用一个或多个字节表示。例如数据包为0x01 FF FF FF FF 时，其格式为:

字节序	位序	数据内容	说明
byte0	bit7 bit6 . . . bit1 bit0	0b00000001	socket, 值为true: 字段bit0, 字段值为0b1;
byte1		0xFF	led_red字段值为255; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为255
byte2		0xFF	led_green字段值为255; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为255
byte3		0xFF	led_blue字段值为255; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为255
byte4		0xFF	led_white字段值为255; 实际值计算公式 $y=1.000000*x+(0.000000)$ x最小值为0, 最大值为255

5.4.5 相关 APP 测试

用户可以直接下载 Gizwits 自动生成的 APP 进行测试, 也可以参考提供到 Github 上的
 公版代码进行二次开发。详细资料请参考 Gizwits 文档中心
 (<http://site.gizwits.com/document/m2m/>)

5.4.6 智能灯使用说明

1、安装相关软件

手机端安装控制智能灯软件 (IOE Demo_com.xpg.wifidemo_1.apk)。



2、配置 SoftAP 模式连接 AP

- 1) 启动烧录 SDK 后的设备, 如果是第一次烧录, nor flash 中没有保存任何 AP 的信息, 并且可能出现连接 AP 行为, 此时需要多重启几次, 等到设备进入 SoftAP 模式, 设备成功进入 SoftAP 模式后, 串口会打印出: Connect2LateWiFi, Working at AP mode。如图 5-1:

```
SDK Version:<1.08.00>; GAgent Version:<1.02.01> Release:<dbg>
=====
Entry gizwits initialize(time:0(s); 70(ms)).
Timer value:100.
GAgent Version: B4R012D0.
Product Version: 06NL6621.
MCU WiFi MAC:000ec600c899
MCU Protocol Version:00000001.
MCU P0 Version:00000004.
MCU Hard Version:00000001.
MCU Soft Version:00000001.
MCU Product Version:43de75d17b4a4d8eae41738df92138b6.
Create system LED indicator task success.
Reset GAgent config data.
MQTT_STATUS_START
passcode:KIOHAWLSEP(len:10)
Connect2LateWiFi, working at AP mode
(237)USE STATIC IP ADDRESS:10.10.100.254 !!!
SYS_EVT_LINK_UP
Create UDP Broadcast socket success.
Create TCP server socket success.
Create NTP request thread success.
Create UDP server socket success.
Create GAgent reset task success.
Create cloud task success.
    Send NTP package success.
    Send NTP package success.
```

图 5-1 设备进入 SoftAP 模式打印信息

- 2) 设备进入 SoftAP 模式后,手机端连接热点名称为: XPG-GAgent-XXXX, XXXX 是 mac 地址后四位, 密码为: 123456789, 如图 5-2:

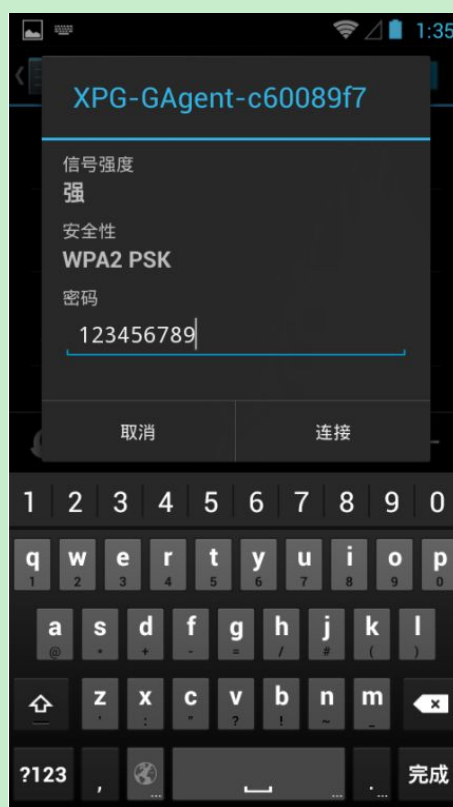



图 5-2 智能灯 softap 模式

- 3) 成功连接上 XPG-GAgent-XXXX 热点后,手机端打开  IOE Demo 应用, 进入

配置 SoftAP 配置界面，配置你所要连接的 Wi-Fi。配置如图 5-3：



图 5-3 SoftAP 配置界面

- 4) 设备端收到配置信息，自动根据收到的 SSID 和 Password 保存并重启设备，重启后连接到 AP。如图 5-4：

```
SDK Version:<1.08.00>; GAgent Version:<1.02.01> Release:<dbg>
=====
Entry gizwits initialize(time:0(s); 70(ms)).
Timer value:100.
GAgent Version: B4R012D0.
Product Version: 06NL6621.
MCU WiFi MAC:000ec600c899
MCU Protocol Version:00000001.
MCU P0 Version:00000004.
MCU Hard Version:00000001.
MCU Soft Version:00000001.
MCU Product Version:43de75d17b4a4d8eae41738df92138b6.
with DID:T9jCq54NaANhDXywhrpg4 len:22
MQTT_STATUS_PROVISION
passcode:KIOHAWLSEP(len:10)
Try to connect wifi...[00000001]
SSID: NUDreambox-2.4
KEY: google.cn
Create system LED indicator task success.
Connecting to AP(NUDreambox-2.4,google.cn)...
(574)[DHCP] MY IP ADDRESS:192.168.1.30
SYS_EVT_LINK_UP
Create UDP Broadcast socket success.
Create TCP server socket success.
Create NTP request thread success.
Create UDP server socket success.
Create GAgent reset task success.
Create cloud task success.
New Http socket ID [4]
```

图 5-4 配置成功打印信息

- 5) 设备连接成功后，手机端连接到 AP，点击登录，



进入注册界面，注册机智云账号，注册成功后登陆机智云云端服务。



6) 刷新设备列表



7) 绑定设备：



3、设备连接 AP 失败说明

- 1) 输入 AP 密码错误，导致设备连不上 AP，设备会在尝试 5 次联网失败后自动进入 SoftAP 模式。如图 5-5：

```

SDK Version:<1.08.00>; GAgent Version:<1.02.01> Release:<dbg>
=====

Entry gizwits initialize(time:0(s); 70(ms)).
Timer value:100.
GAgent Version: B4R012D0.
Product Version: 06NL6621.
MCU WiFi MAC:000ec60089f7
MCU Protocol Version:00000001.
MCU PO Version:00000004.
MCU Hard Version:00000001.
MCU Soft Version:00000001.
MCU Product Version:c853fcbddfb64fc796ac22cf975708ae.
with DID:RTVLgu8f2UrKdNbsPrAy6x len:22
MQTT_STATUS_PROVISION
passcode:LOBIWPOWAO (len:10)
Try to connect wifi... [00000001]
SSID: dlink-James
KEY: 123qweASD
Create system LED indicator task success.
Connecting to AP (dlink-James,123qweASD)...
SYS_EVT_JOIN_FAIL
Connect2LateWiFi, Working at AP mode
(1050)USE STATIC IP ADDRESS:10.10.100.254 !!!
SYS_EVT_LINK_UP
Create UDP Broadcast socket success.
Create TCP server socket success.
Create NTP request thread success.
Create UDP server socket success.
Create GAgent reset task success.
Create cloud task success.
=====

```

图 5-5 联网失败

4、智能灯控制说明

绑定设备成功，点击设备进入控制界面。智能灯能控制红、蓝、绿、白四种颜色灯的亮度，亮度范围为 0 ~ 255，拖动进度条可调节红、蓝、绿、白的亮度搭配出各种颜色。如图 5-7:

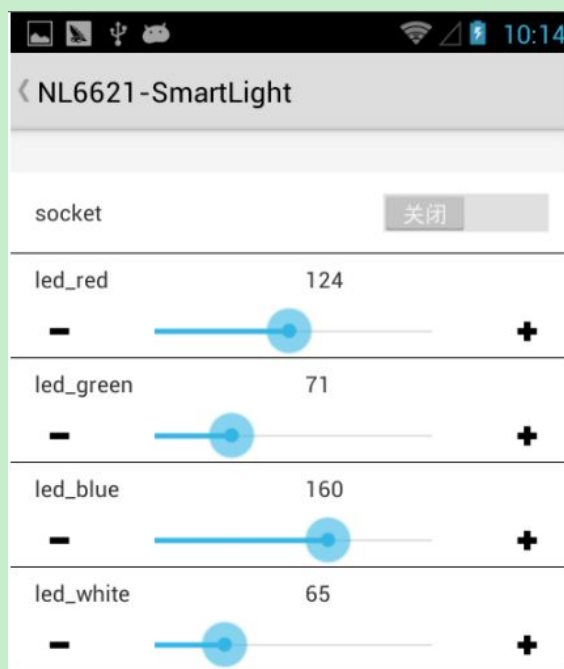


图 5-7 智能灯控制界面

6. 注意事项
