

Gizwits 串口协议移植

移植准备工作

软件准备：

1. keil5(RAM)
2. 其他平台 SDK
3. 串口调试助手
4. 机智云 demo APP

硬件准备：

1. 科派科技开发板
2. usb 数据线
3. 如果是其它 stm32 开发板，还是需要 esp8266

WIFI 固件烧写

移植前对开发板上的 esp8266 模块刷入机智云 GAgent 固件



下载中心

硬件开发资源

- GoKit MCU 示例工程
- GAgent**

客户端开发资源

- 开源框架
- 设备接入 SDK
- 统计分析 SDK

开发与调试工具

- 机智云 Wi-Fi 移动通信产品测试 APP
- 产测工具
- 机智云串口调试助手

开放源码

- 开源 APP
- 开源硬件
- 开源示例

GAgent

GAgent 是运行在各种通讯模组上的一款应用程序（固件），可以提供上层应用（手机 APP 等）控制端、云端到产品设备的双向数据通讯。此外，还提供对设备的配置上网、发现绑定、程序升级等功能。

产品开发者使用 GAgent 后，只需要关心产品的业务逻辑开发，不用关心数据的通讯功能开发，大大降低了开发的难度。

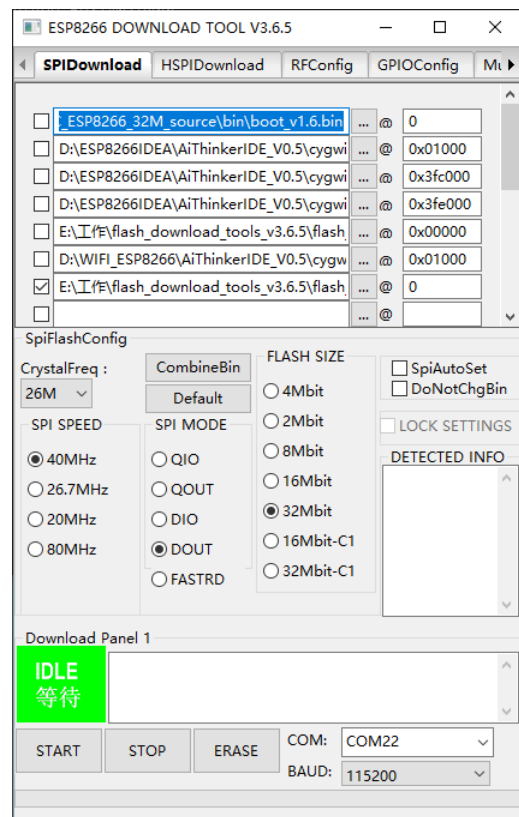
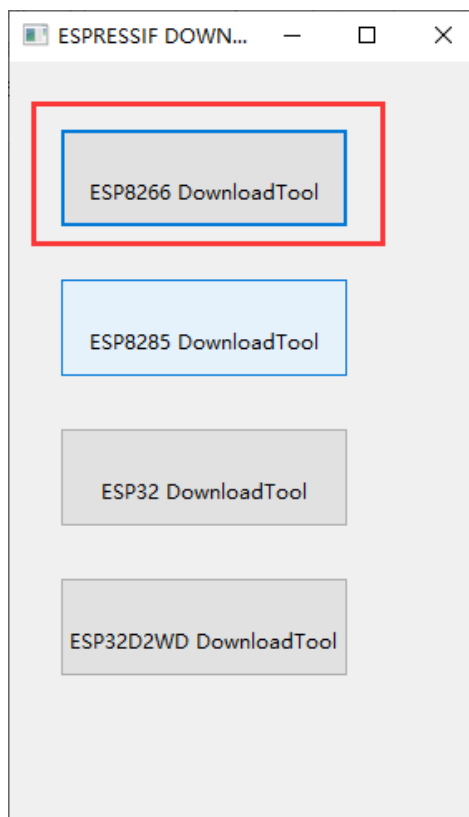
该固件遵循以下协议《机智云平台标准接入协议之 MCU 与 Wi-Fi 模组通讯》、《机智云平台标准接入协议之设备与云端通讯》、《机智云平台标准接入协议之 App 与设备通讯》。

| | |
|--|----------------------|
| ECE 云计算 esp8266 固件包 试用版 | 资源下载 |
| 发布时间：2017-07-20 03:50 更新信息 旧版本下载 | |
| GAgent for ESP8266 04020034 | 资源下载 |
| 发布时间：2018-06-16 07:14 更新信息 旧版本下载 | |
| GAgent for HFLPT120/LPB120/LPB125/LPT220 04020035 | 资源下载 |
| 发布时间：2018-09-04 06:00 更新信息 旧版本下载 | |
| GAgent for HF LPB100 04020022 | 资源下载 |
| 发布时间：2017-09-01 10:14 更新信息 旧版本下载 | |
| GAgent for MXCHIP 04020011 | 资源下载 |
| 发布时间：2016-03-04 06:11 更新信息 旧版本下载 | |

| | | | |
|--|-----------------|--------|----------|
| GAgent_00ESP826_04020034_8MbitUser1_201806091441.bin | 2018/6/9 14:37 | BIN 文件 | 426 KB |
| GAgent_00ESP826_04020034_8MbitUser1_combine_201806091441.bin | 2018/6/16 19:06 | BIN 文件 | 1,024 KB |
| GAgent_00ESP826_04020034_8MbitUser2_201806091442.bin | 2018/6/9 14:39 | BIN 文件 | 426 KB |
| GAgent_00ESP826_04020034_16Mbit_201806091444.bin | 2018/6/9 14:41 | BIN 文件 | 446 KB |
| GAgent_00ESP826_04020034_16Mbit_combine_201806091444.bin | 2018/6/16 19:07 | BIN 文件 | 2,048 KB |
| GAgent_00ESP826_04020034_32Mbit_201806091446.bin | 2018/6/9 14:43 | BIN 文件 | 446 KB |
| GAgent_00ESP826_04020034_32Mbit_combine_201806091446.bin | 2018/6/16 19:08 | BIN 文件 | 4,096 KB |
| readme | 2018/6/16 19:11 | 文件 | 1 KB |

通过乐鑫原厂烧写工具烧写固件

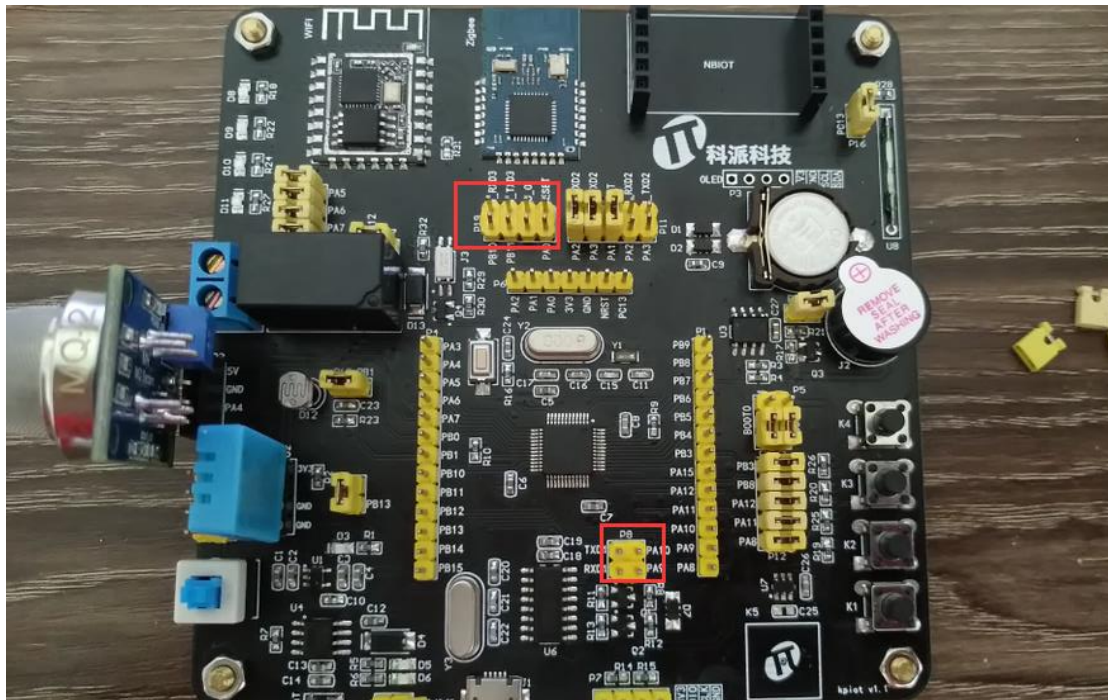
| 名称 | 修改日期 | 类型 | 大小 |
|------------------------------------|-----------------|-------------|-----------|
| combine | 2019/5/14 15:08 | 文件夹 | |
| configure | 2019/7/5 0:28 | 文件夹 | |
| dl_temp | 2019/5/14 15:08 | 文件夹 | |
| hewu | 2019/6/25 15:42 | 文件夹 | |
| init_data | 2019/5/14 15:08 | 文件夹 | |
| RESOURCE | 2019/5/14 15:08 | 文件夹 | |
| secure | 2019/5/14 15:08 | 文件夹 | |
| .DS_Store | 2017/9/22 19:41 | DS_STORE 文件 | 7 KB |
| flash_download_tools_v3.6.5 | 2018/9/18 16:40 | 应用程序 | 13,468 KB |
| GAgent_00ESP826_04020034_8MbitU... | 2018/6/16 19:06 | BIN 文件 | 1,024 KB |
| GAgent_00ESP826_04020034_32Mbit... | 2018/6/16 19:08 | BIN 文件 | 4,096 KB |
| Readme | 2017/9/22 19:41 | 看图王 PDF 文件 | 455 KB |



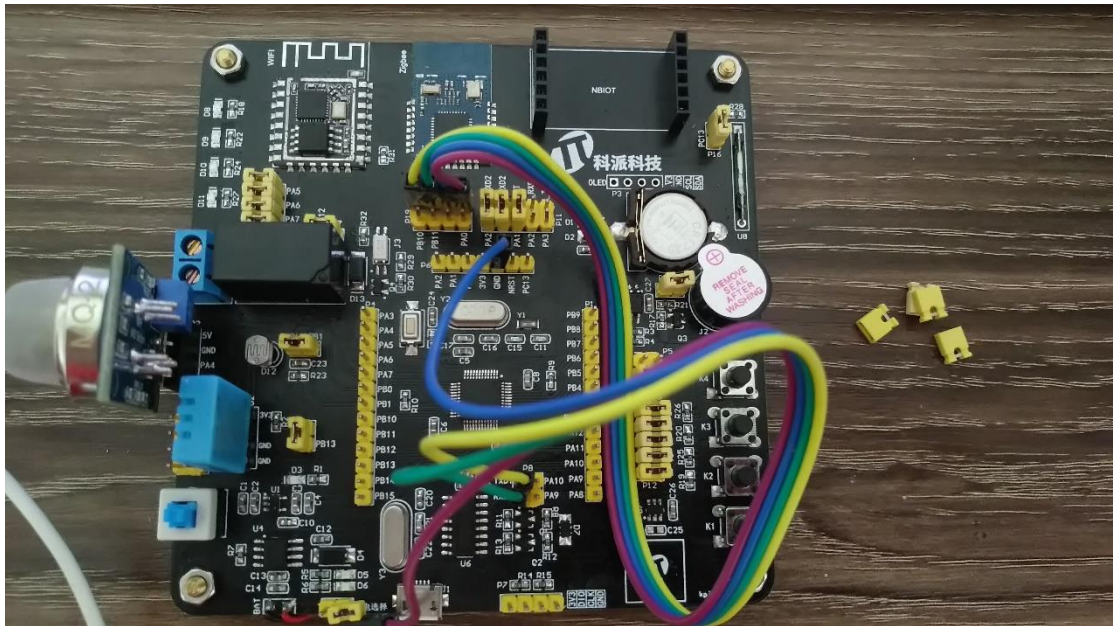
硬件设备连接

我们的开发板上自带了一颗 ch340 串口芯片，通过 ESP8266 与板子串口进行连接就可以下载 GAgent 固件了。

去掉两处的跳帽



使用杜邦线连接



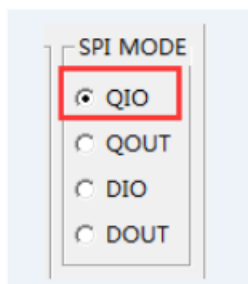
连接示意图

| 开发板 | WiFi |
|------|--------|
| TXD1 | W_RXD3 |
| RXD1 | W_TXD3 |
| GND | IO_0 |

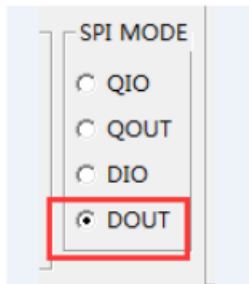
注意：下载时 IO_0 引脚必须与 GND 相连，拉低才能进入下载模式

GAgent 固件烧写

修改 SPI 模式：



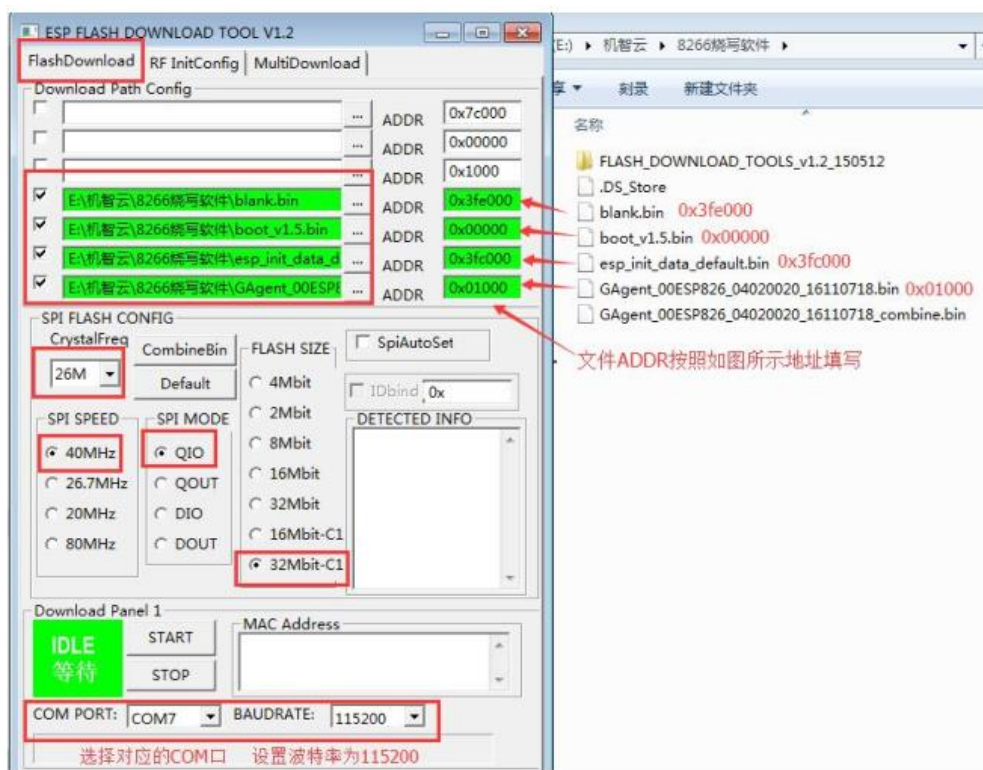
安信可12F



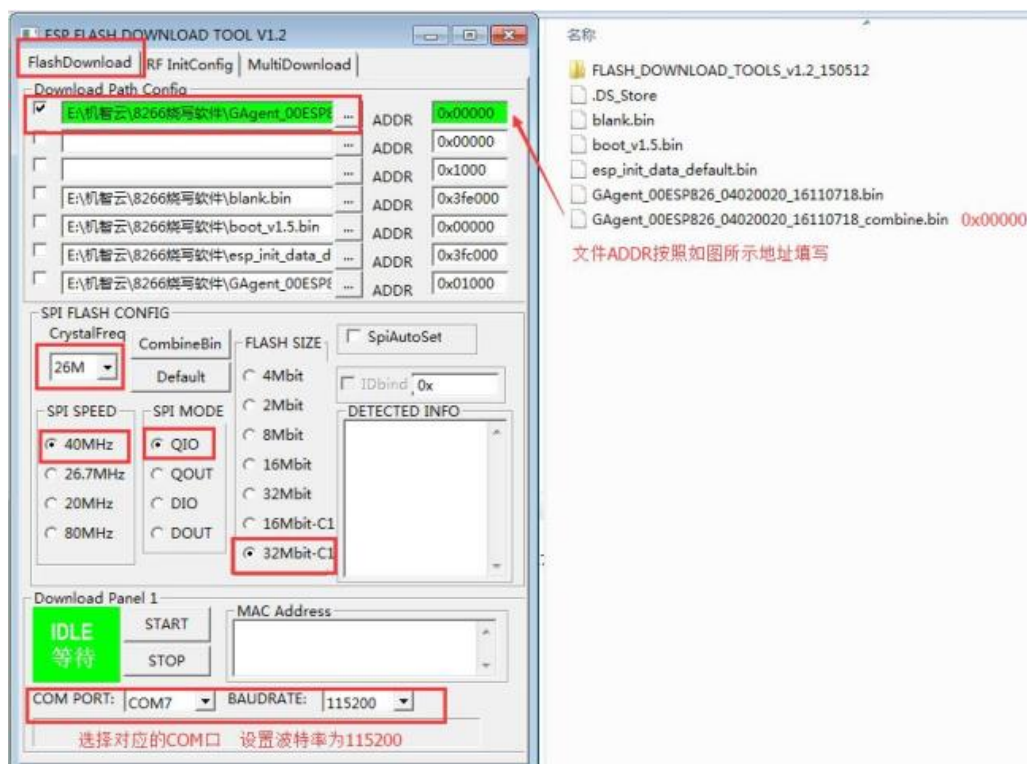
汇思锐12H

下载时请根据所用板本而进行 SPI 模式选择，接下来我们将以安信可 12F 版本为例进行讲解。

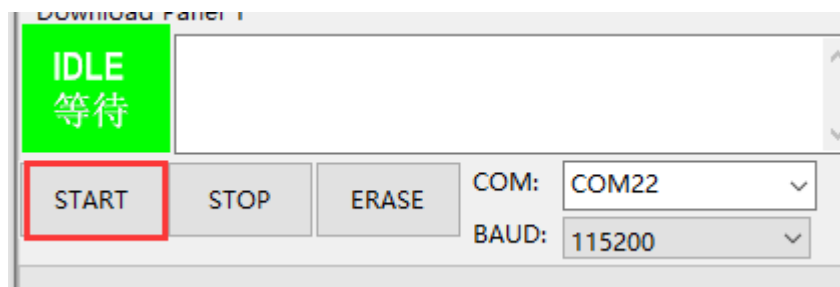
固件（GAgent_00ESP826_04020023_17032418）



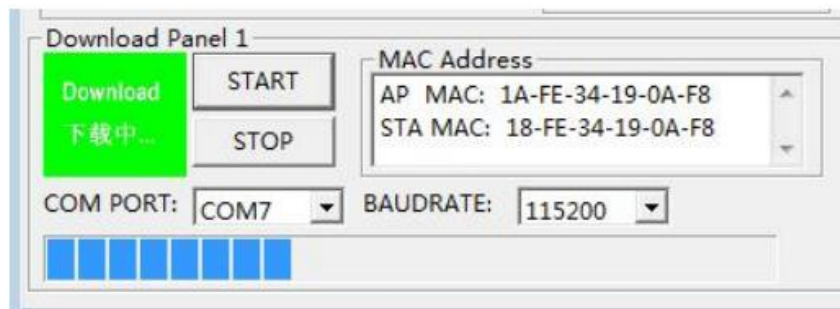
固件 (GAgent_00ESP826_04020023_17032418_combine)



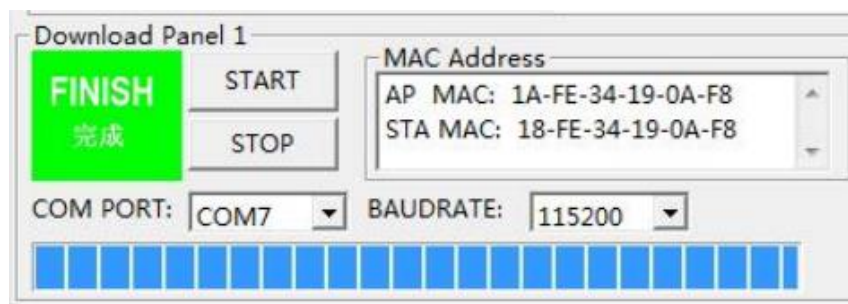
点击“START”按钮



ESP8266 内部将自动进行一次复位，复位成功后将会出现如下信息，表示模块正在进行烧写



等待一段时间后，出现“FINISH”字样表示烧写成功



烧写完成后，需将板子上排针接口重新用跳线帽连接

| | |
|--------|------|
| W_RXD3 | PB10 |
| W_TXD3 | PB11 |
| TXD1 | PA10 |
| RXD1 | PA9 |

串口协议移植

工程配置

协议的移植需要用到串口和定时器,在我们给大家提供的工程模板中已经有相应的程序,我们只需要把之前通用平台 SDK 下"Gizwits"和"Utils"复制到我们的工程下。

| 名称 | 修改日期 | 类型 | 大小 |
|----------|-----------------|-----|----|
| CMSIS | 2019/7/21 14:33 | 文件夹 | |
| Gizwits | 2019/7/21 14:33 | 文件夹 | |
| hardware | 2019/7/21 14:33 | 文件夹 | |
| Lib | 2019/7/21 14:33 | 文件夹 | |
| project | 2019/7/21 14:33 | 文件夹 | |
| Startup | 2019/7/21 14:33 | 文件夹 | |
| User | 2019/7/21 14:33 | 文件夹 | |
| Utils | 2019/7/21 14:33 | 文件夹 | |

代码移植说明;

1. 实现与模组通信串口驱动（中断收数据写入环形缓冲区；实现 `uartWrite()` 串口发送函数）
2. 实现串口打印函数 `printf()`
3. 实现 ms 定时器，`gizTimerMs()` 维护系统时间
4. 实现 MCU 复位函数，模组可请求 MCU 复位
5. 实现配置入网功能，调用 `gizwitsSetMode()` 函数实现模组配网功能
6. 实现 `userHandle()` 数据的采集（上行逻辑）
7. 实现 `gizwitsEventProcess()` 控制命令的具体执行（下行逻辑）
8. 实现 `mcuRestart()` 复位函数

WIFI 串口接收数据写入缓冲区

将 wifi 串口接收到数据写入到缓冲区中，而写入缓冲区的函数为 gizPutData(), 函数位置在 gizwits_protocol.c 文件下

```
void USART3_IRQHandler(void)
{
    u8 res;
    if(USART_GetITStatus(USART3, USART_IT_RXNE) != RESET) //接收到数据
    {
        res = USART_ReceiveData(USART3);
        gizPutData(&res, 1); // 数据写入到缓冲区
    }
}
```

实现 uartWrite() 串口发送数据

在 uartWrite()函数中，协议通信会用到这个函数，而这函数会涉及数据帧数据的发送，这里需我们提供串口的发送函数，uartWrite()函数具体位置在 gizwits_product.c 文件下

```
int32_t uartWrite(uint8_t *buf, uint32_t len)
{
    uint32_t i = 0;

    if(NULL == buf)
    {
        return 0;
    }

    #ifdef PROTOCOL_DEBUG
    for(i=0; i<len; i++)
    {
        //USART_SendData(UART, buf[i]); //STM32 test demo
        //Serial port to achieve the function, the buf[i] sent to the module
        USART_SendData(USART3, buf[i]);
        while(USART_GetFlagStatus(USART3, USART_FLAG_TC) == RESET);
        if(i >= 2 && buf[i] == 0xFF)
        {
            //Serial port to achieve the function, the 0x55 sent to the module
            //USART_SendData(UART, 0x55); //STM32 test demo
            USART_SendData(USART3, 0x55);
            while(USART_GetFlagStatus(USART3, USART_FLAG_TC) == RESET);
        }
    }
    return len;
}
```

实现毫秒定时

```
*****/
void TIM4_IRQHandler(void)    //TIM4中断
{
    if (TIM_GetITStatus(TIM4, TIM_IT_Update) != RESET)    //检查TIM4更新中断发生与否
    {
        gizTimerMs();

        if(ntp_ms <= 1000)
        {
        }
        else
        {
            if(wifi_ms <= 15000)
            {
            }
            else
            {
                Iwdg_Feed();    //喂狗
            }
        }
        TIM_ClearITPendingBit(TIM4, TIM_IT_Update );    //清除TIM4更新中断标志
    }
}
```

协议层的运行需要一个系统时间，事件单位为毫秒，所以我们需要实现毫秒定时器（必须是 1ms 的精确定时，若不准确，会影响到超时重发、定时上报等处理），gizTimerMs()函数具体位置在 gizwits_protocol.c 文件下

配置 WIFI 接入

```
printf("-----物联网 IOT-LED控制头灯-----\r\n");
printf("KEY1:AirLink 连接模式\t KEY_UP:复位\r\n\r\n");
while(1)
{
    if(KEY0 == RESET)    按键配网
    {
        DelayMs(10);
        if(KEY0 == RESET)
        {
            gizwitsSetMode(WIFI_AIRLINK_MODE);
        }
        while(!KEY0);
    }
}
```

WIFI 设备与云端通信前需要配置接口模式，而配置接口模式是调用了 gizwitsSetMode()函数，其函数可实现模组配网功能或复位、产测和绑定功能，gizwitsSetMode()函数具体位置在 gizwits_protocol.c 文件下，它共有五种配置模式，复位、SoftAP、AirLink 模式、产测模式和允许用户绑定设备模式

实现 userHandle() 数据的采集（上行逻辑）

userHandle()函数位置在 main.c 中，它主要目的让开发者可以在这函数中实现对传感器等其他设备采集，然后将需要上传的数据存到其设备状态结构体 currentDataPoint 结构体成员中（注意：上传数据要和产品的数据点是有关联的，不然上传也没有意义），设备状态结构体 currentDataPoint 变量定义在 main.c

中


```
65
66 dataPoint_t currentDataPoint;
67
68 /*****
31 void userHandle(void)
32 {
33     if( wifi_sta )
34     {
12     if(short_key_flag == 1)
13     {
54     else if(long_key_flag == 1)
55     {
61 }
62
```

协议处理

根据采集到的数据，然后调用函数 gizwitsHandle()上报，函数位置在 gizwits_protocol.c 文件中，该函数主要完成协议数据的处理及数据主动上报的相关操作。

```
int main(void)
{
    Hardware_Init();

    while(1)
    {
        userHandle();
        scanf_key();
        gizwitsHandle((dataPoint_t *)&currentDataPoint); //协议处理
    }
}
```



实现 gizwitsEventProcess() 控制命令的具体执行（下行逻辑）

在 gizwitsEventProcess() 事件处理函数中需要用户处理数据点的事件位置在 gizwits_product.c 文件中。这里我们需在 EVENT_led1, LED 开关事件中，增加对 LED 开关的逻辑

```
int8_t gizwitsEventProcess(eventInfo_t *info, uint8_t *gizdata, uint32_t len)
{
    uint8_t i = 0;
    dataPoint_t *dataPointPtr = (dataPoint_t *)gizdata;
    moduleStatusInfo_t *wifiData = (moduleStatusInfo_t *)gizdata;
    protocolTime_t *ptime = (protocolTime_t *)gizdata;
    #if MODULE_TYPE
    if((NULL == info) || (NULL == gizdata))
    {
        for(i=0; i<info->num; i++)
        {
            switch(info->event[i])
            {
                case EVENT_led1:
                    currentDataPoint->valueled1 = dataPointPtr->valueled1;
                    GIZWITS_LOG("Evt: EVENT_led1 %d \n", currentDataPoint->valueled1);
                    if(0x01 == currentDataPoint->valueled1)
                    {
                        //user handle
                        LED1_ON;
                        key1_flag = 1;
                    }
                    else
                    {
                        //user handle
                        LED1_OFF;
                        key1_flag = 0;
                    }
                }
            }
        }
    }
}
```

用户事件

实现 mcuRestart() 复位函数

WIFI 在通信过程中可能会对 MCU 进行请求复位，而复位调用函数为 mcuRestart(), 该函数是空的，我们需在函数中实现软复位，函数位置在 gizwits_product.c 中

```
void mcuRestart(void)
{
    __set_FAULTMASK(1); // 关闭所有中断
    NVIC_SystemReset(); // 复位
}
/**@} */
```

初始化

```
void Gizwits_Init(void)
{
    TIM4_Int_Init(9, 7199);
    usart3_init(9600); // wifi初始化 波特率必须为9600
    memset((uint8_t *)&currentDataPoint, 0, sizeof(dataPoint_t));
    gizwitsInit();
}
```

验证

[下载工程、配网](#)