

开发前准备

一、开发板订购

1、进入开发板申请页面



2、点击购买



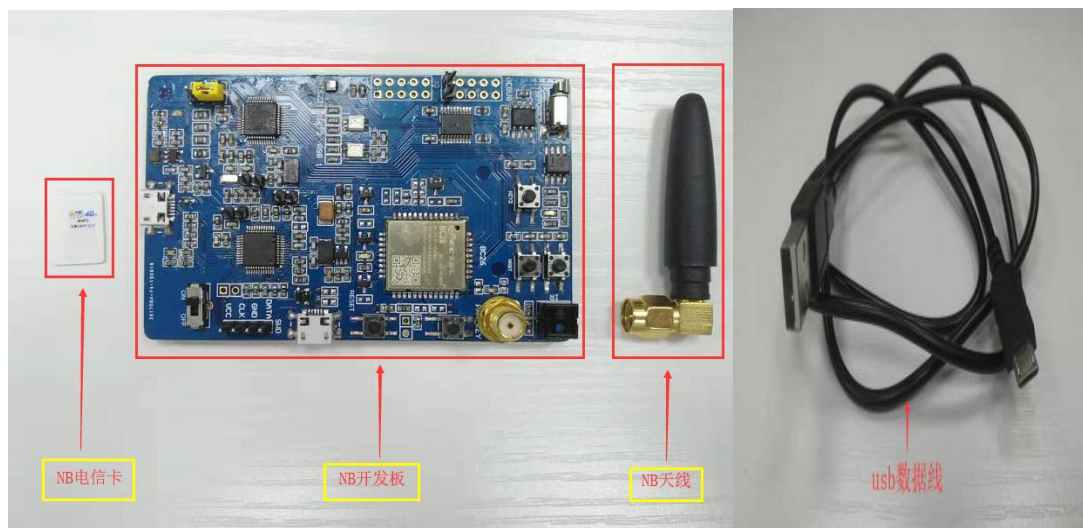
3、根据自身需求选择配件及开发板数量后，点击“订购”，之后填写收货地址、发票信息、订单备注等信息，点击“确认下单”即可。



二、清单列表

硬件:

电信开发套件: NB 电信卡、NB 开发板 (CTWINGSKIT_BC28)、NB 天线, usb 数据线;



软件:

电信软件套件: 开发板工程代码包 (skit_ctnb_st.zip)、SKIT 串口调试助手;

自备 软件工具: ST-Link 驱动、串口工具 SKIT 或其它串口工具、Keil5 软件、Keil.STM32F1xx_DFP.2.2.0 库包;

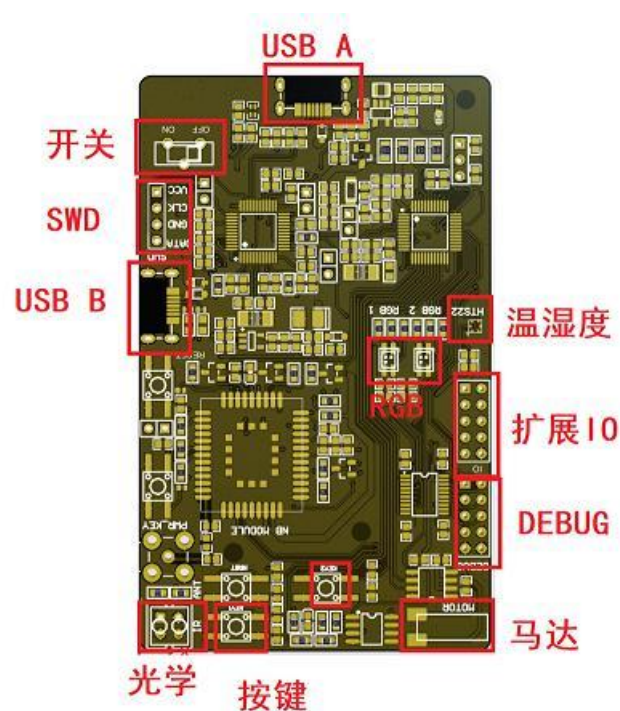
文档:

自备 《Quectel_BC35-G&BC28&BC95 R2.0_AT_Commands_Manual_V1.5》

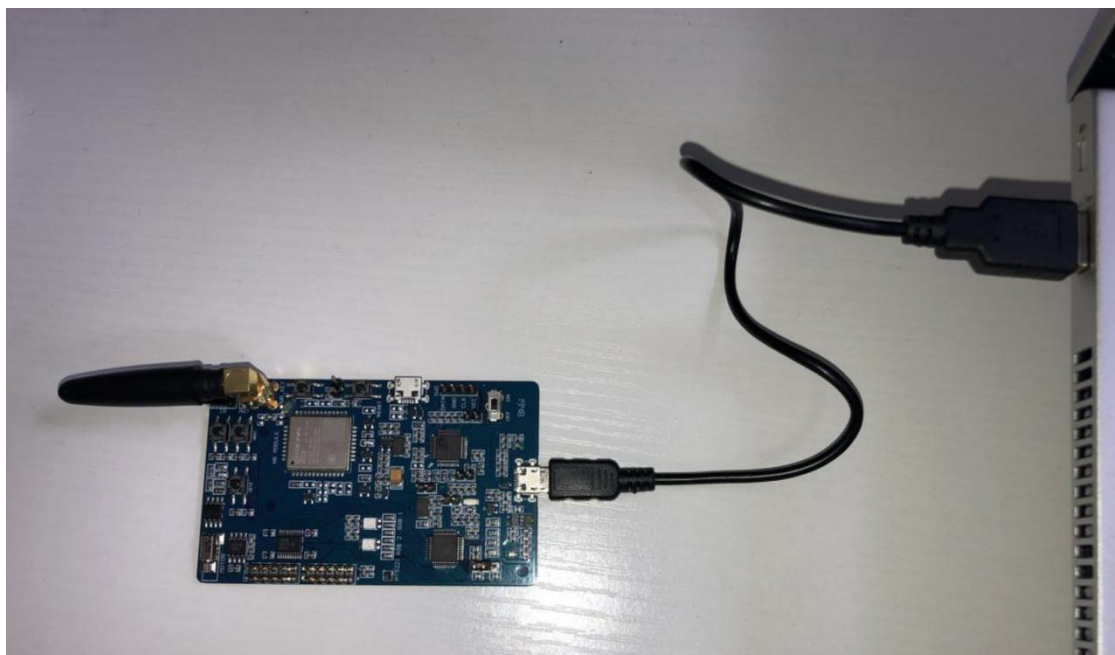
《基于开发板的开发指南_SKIT_NB_BC28_V1.1》

《SKIT_NB_BC28_V1.1 样例程序代码下载》

三、开发板介绍



NB 开发板套件使用 NB 开发板，通过板载 ST-Link 工具下载 mcu 程序，通过 ST-Link 提供的模拟串口输出调试日志，并且通过 sub 供电。在断电条件下插入电信 **NB-IOT 通信卡 (必须事先实名认证)**，将 usb 数据线一端插入计算机 USB 口，另一端插入开发板上的 microusb 接口，如下图所示：




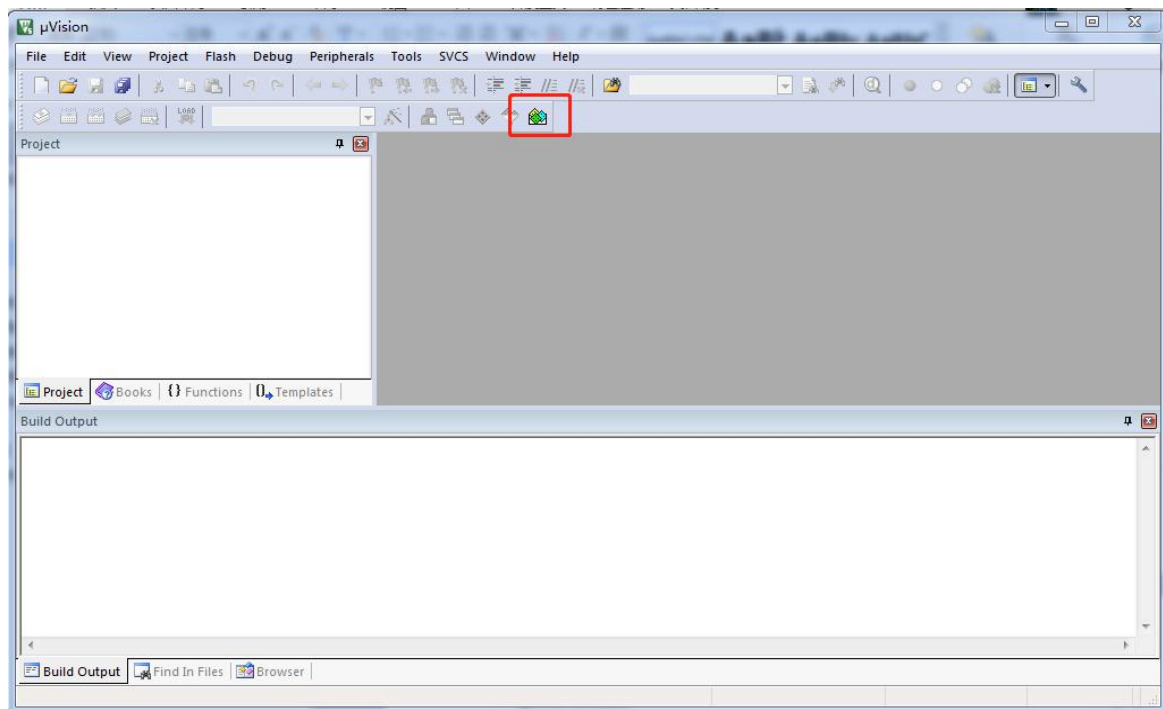
四、开发软件安装

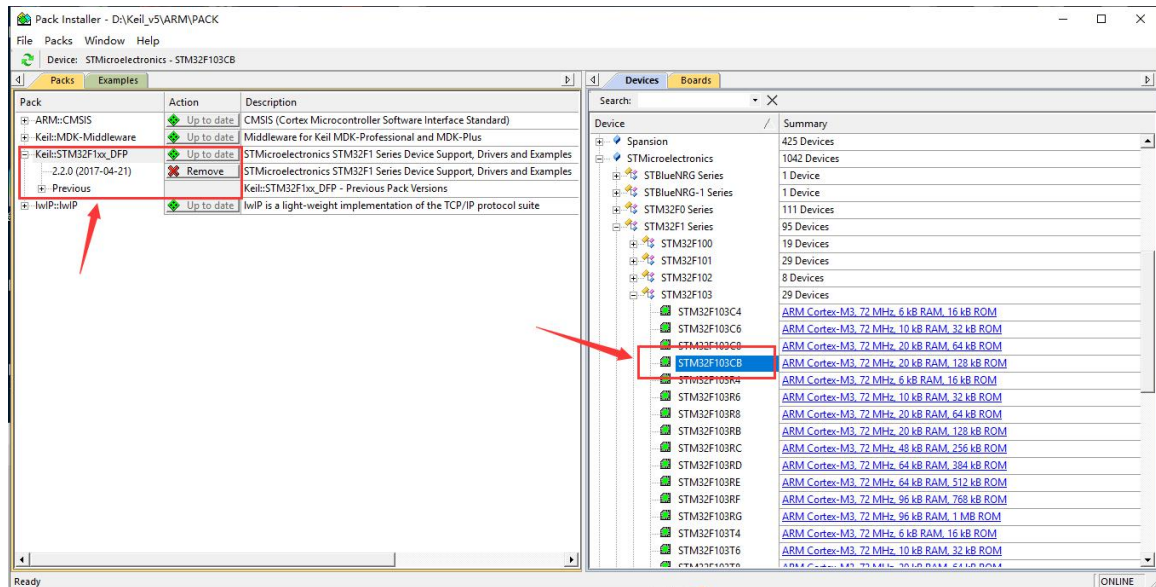
1. 安装 STLink 驱动(自备)；
2. 如果需要 Debug 工具，请安装 SKIT.exe 专用工具或其它串口工具 ；
3. 安装 Keil5 软件(自备)；
4. Keil5 导入库包 Keil.STM32F1xx_DFP.2.2.0 或最新库包(自备)；

导入库包 Keil.STM32F1xx_DFP.2.2.0 可选择以下两种方法中的一种：


方法一：

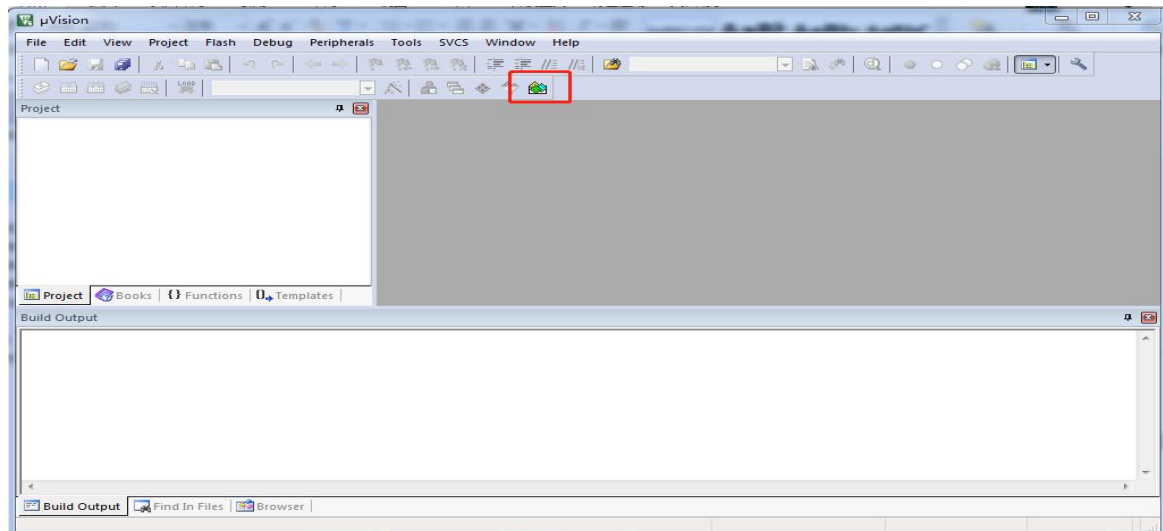
打开工程，点击‘Pack Installer’按钮，然后在 pack installer 界面从 Packs 树形图中选择 STM32F1xx_DFP，点击‘Install’按钮，安装 Keil.STM32F1xx_DFP 库包，安装完成后应该从右边 device 树状列表中就能查到设备型号“STM32F103CB”。

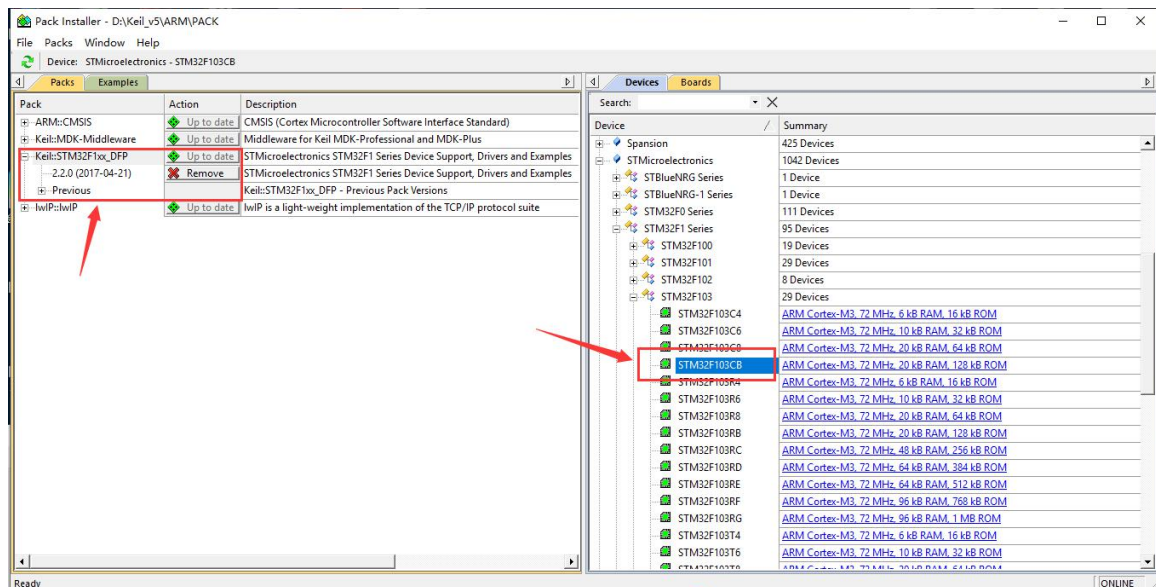
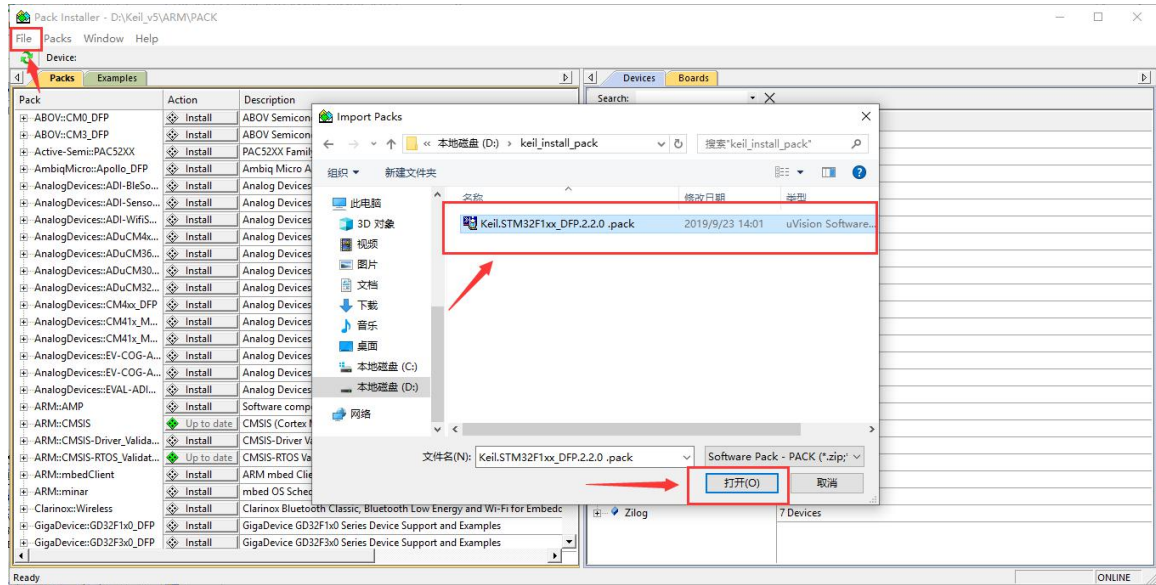




方法二：

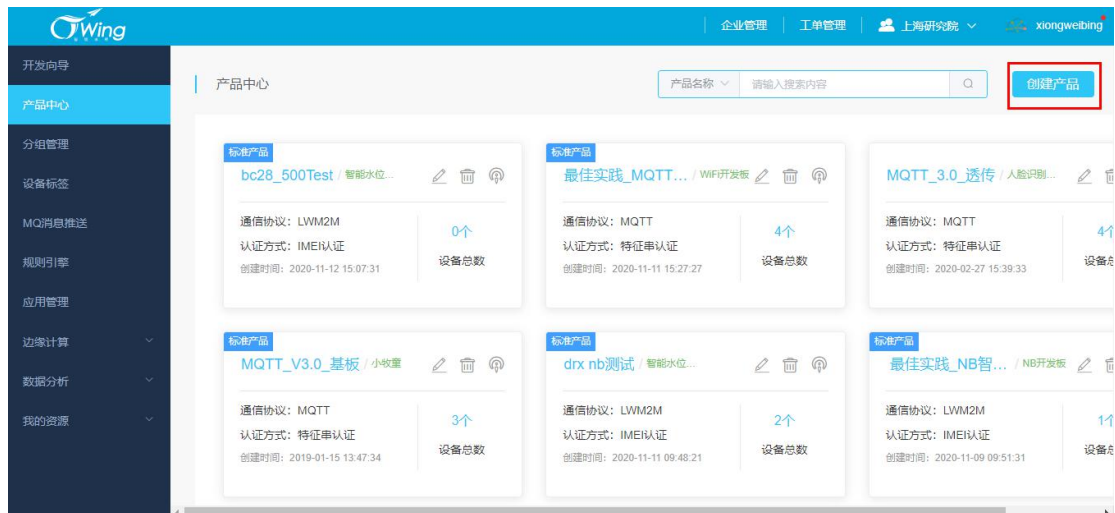
下载 Keil.STM32F1xx_DFP.2.2.0 包，下载地址为：<http://www.keil.com/pack>，打开工程，点击‘Pack Installer’按钮，然后在 pack installer 界面点击菜单 file，选择 import 菜单，选择下载的安装包进行导入操作，安装完成后应该从右边 device 树状列表中就能查到设备型号“STM32F103CB”。





创建产品

登录平台，进入开发者中心



创建产品

×

* 产品名称

1-64个字符,必须包含数字或字母

* 产品分类

智慧农

最佳实践

NB-IoT

功能定义

* 节点类型

设备

网关

* 接入方式

设备直连

* 网络类型

NB-IoT

* 通信协议

LWM2M

* 数据加密方式

明文

* 认证方式

IMEI认证

* Endpoint格式

imei

* 是否已有电信官方认证

是

否

通过的profile

是

否

* 设备型号

请输入设备型号

请输入设备型号

是否遗传

是

否

* 消息格式

紧凑型二进制

* 省电模式

PSM

产品描述

输入产品描述

确定

取消

创建好产品后，进入添加好的产品，进入设备管理，添加设备（添加设备时，注意 IMEI 号，在开发板的 BC28 芯片上）

添加设备

* 设备名称

1-64个字符，必须包含一个字母、数字或汉字

* IMEI号

在开发板的BC28芯片上

请根据产品Endpoint格式输入IMEI号

IMSI号

请根据产品Endpoint格式输入IMSI号

是否开启自动订阅

☒ 是

☐ 否

确定

取消

终端软件开发

开始进入软件开发前，请先熟悉 NB 开发板相关 AT 指令文档，从中国电信物联网开放平台下载代码源文件（https://www.ctwing.cn/page.html#/kit_2）。

代码结构介绍

1.1 下载开发板程序文件 skit_ctnb_st.zip，解压文件到自己工作目录中（建议直接解压在根目录下，如：D:\skit_ctnb_st）；

文件目录结构如下：

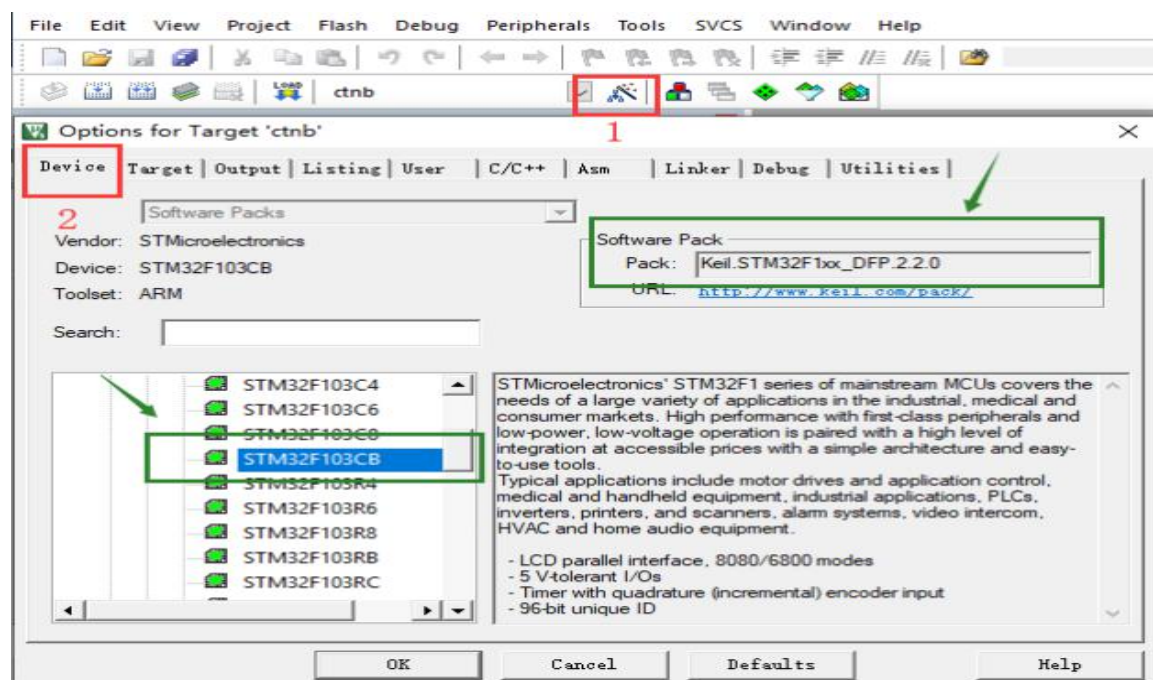
名称	修改日期	类型	大小
bsp	2019/9/23 16:41	文件夹	
Drivers	2019/9/23 16:41	文件夹	
MDK-ARM	2019/9/23 16:41	文件夹	
RTOS	2019/9/23 16:41	文件夹	
system	2019/9/23 16:41	文件夹	
user	2019/9/23 16:41	文件夹	
.mxproject	2019/9/23 16:37	MXPROJECT 文件	5 KB
ctnb.ioc	2019/9/23 16:37	STM32CubeMX	6 KB
mx.scratch	2019/9/23 16:37	SCRATCH 文件	5 KB

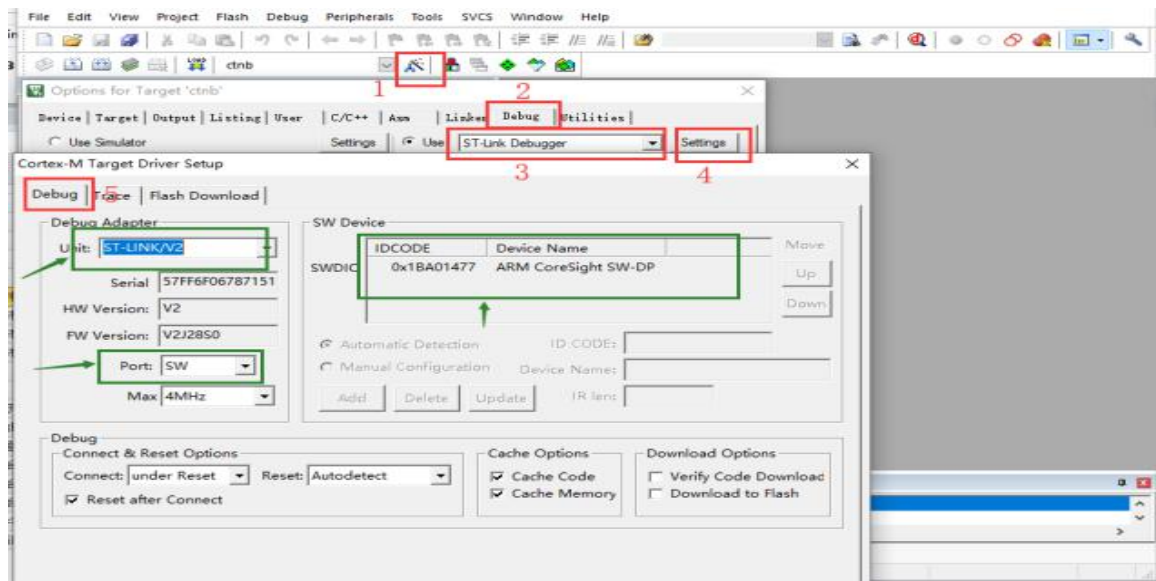
- bsp:** 存放 BSP 驱动文件（如 LED，电机，温湿度传感器等），共用工具类文件（如 MD5，字符串处理函数等）；
- Drivers:** 存放 STM32 平台的 HAL 库文件和 CMSIS 接口文件；
- MDK-ARM:** 存放 Keil 工程启动文件；
- RTOS:** 存放操作系统 RTOS 相关文件（如 OS 线程，日志等）；
- system:** 存放 main 函数入口文件和是 STM32 系统接口文件（如中断，系统时钟初始化、HAL 初始化等）；
- user:** 存放用户线程文件（如 与通信模组的 AT 操作和业务实现函数等）；

1.2 开发板程序是基于 MDK IDE 编写，使用 Keil5 打开工程：进入 MDK-ARM 目录，双击文件 ctnb.uvprojx，或者打开 Keil 软件后选择 project 菜单中的 open project，即可打开代码工程。

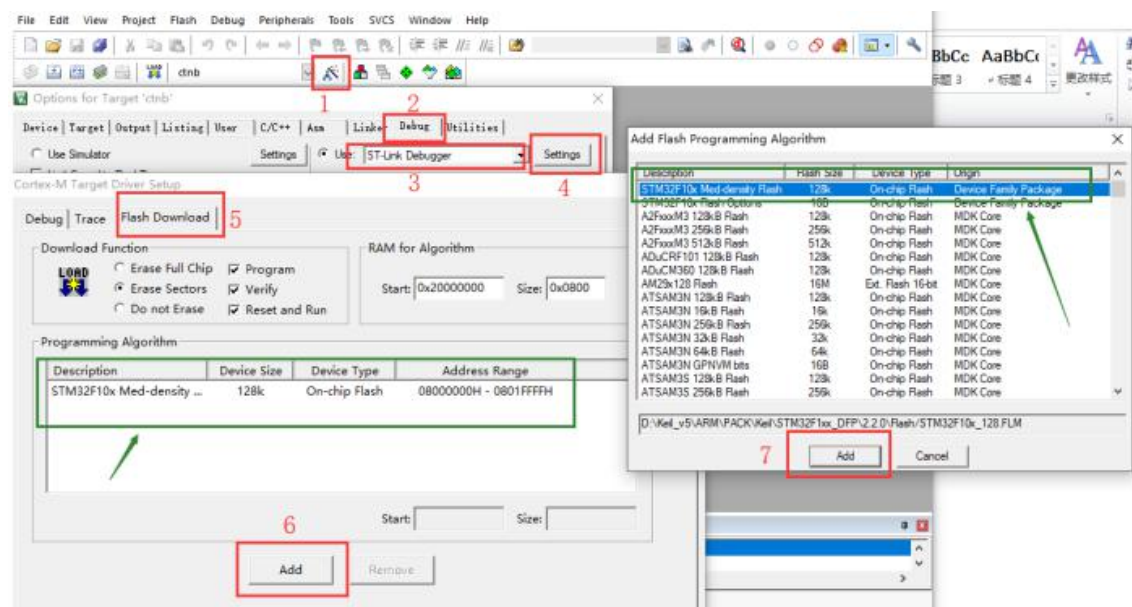
确认配置信息（按照红色步骤点击菜单，确认绿色内容）：

ST-Link 如下图：

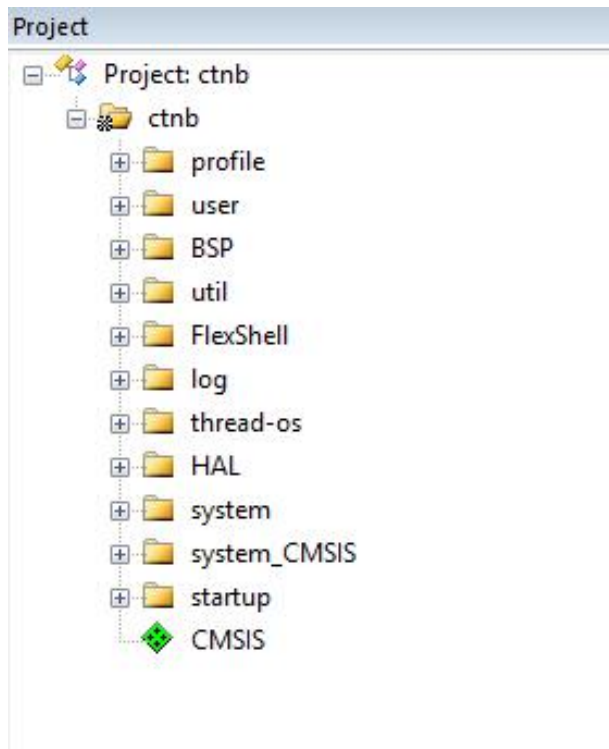




如果没有下载配置内容，可以自行点击‘Add’添加下载配置，如下图：



工程架构介绍：



- profile:** 存放编解码文件;
- user:** 业务实现的具体存放区, 目前里面已经实现了 nb_thread, bsp_thread, ctnb_thread 等, 如果用户想创建自己的业务逻辑可以参考这里的 thread。
- BSP:** 存放 BSP 驱动文件 (如 LED, 电机, 温湿度传感器、GPIO、SPI, IIC, 红外传感器、BSP 通用接口等);
- util:** 共用工具类文件 (如 MD5, 字符串处理函数等);
- FlexShell:** 命令行实现的具体文件, 用户想自己创建测试命令, 可以在 FlesShellUser.c 文件里追加。
- log:** 日志打印文件, 用户可以选择 UART 或者 RTT 进行日志数据的打印。
- thread-os:** 一个基于线程的 RTOS。用户不需要关心该文件。
- HAL:** st 公司提供的库文件。
- system:** 存放 main 函数入口文件和是 STM32 系统接口文件 (如中断, 系统时钟初始化、HAL 初始化等);
- system_CMSIS:** 系统时钟配置文件。
- startup:** stm32 启动文件。如果用户想修改栈大小, 可以修改该文件

备注:如需使用日志、定时器、线程、shell 命令详见附件

开发开发板程序

基于 NB 开发板的样例程序通过串口使用 AT 指令与 NB 模组通信, 所有 AT 指令集参数说明详见《Quectel_BC35-G&BC28&BC95 R2.0_AT_Commands_Manual_V1.5》

1、连接好开发板，搭建好开发环境；

2、产品在平台创建完成后，用户获取并保存如下所需信息：

- a. 接入 IP 地址 (hostIP)，接口端口(hostPort)，
- b. 服务 ID (用于设置数据上报或下行的 DatasetID)；

备注：

属性和服务可以使用导入物模型（从其它产品导出的物模型）方式快速生成，创建成功后支持导出物模型（供其它产品使用）和生成设备侧编解码文件功能；

3、获取代码包，使用 Keil 软件打开代码工程；

主要关注 user 目录中的 userconfig.h 和 profile 目录中的 app_aep_profile.c。

userconfig.h 用于配置平台接入参数。

app_aep_profile.c 是具体应用文件（如电机控制等）。

如果需要使用自定义物模型编解码代码，请使用前面生成的设备侧编解码文件替换 profile 目录中的两个文件：

AepServiceCodes.c //编解码源代码文件

AepServiceCodes.h //编解码头文件

并且修改 app_aep_profile.c 中的参数调用和对应的物模型编解码函数。

4、修改接入配置信息（userconfig.h）

```
1  #ifndef  __USERCONFIG_H__
2  #define  __USERCONFIG_H__
3
4
5  /*****
6   *   server params config
7   *
8   *****/
9
10
11  #define CTIOT_INIT_IP      "221.229.214.202"
12  #define CTIOT_INIT_PORT    5683
13  #define CTIOT_REG_LIFETIME 3600
14
15
16
17 #endif
18
19
```

编译开发板程序



连接好开发板，点击左上角编译图标进行代码编译，确认编译结果没有 error 出现，编译成功后输出如下：

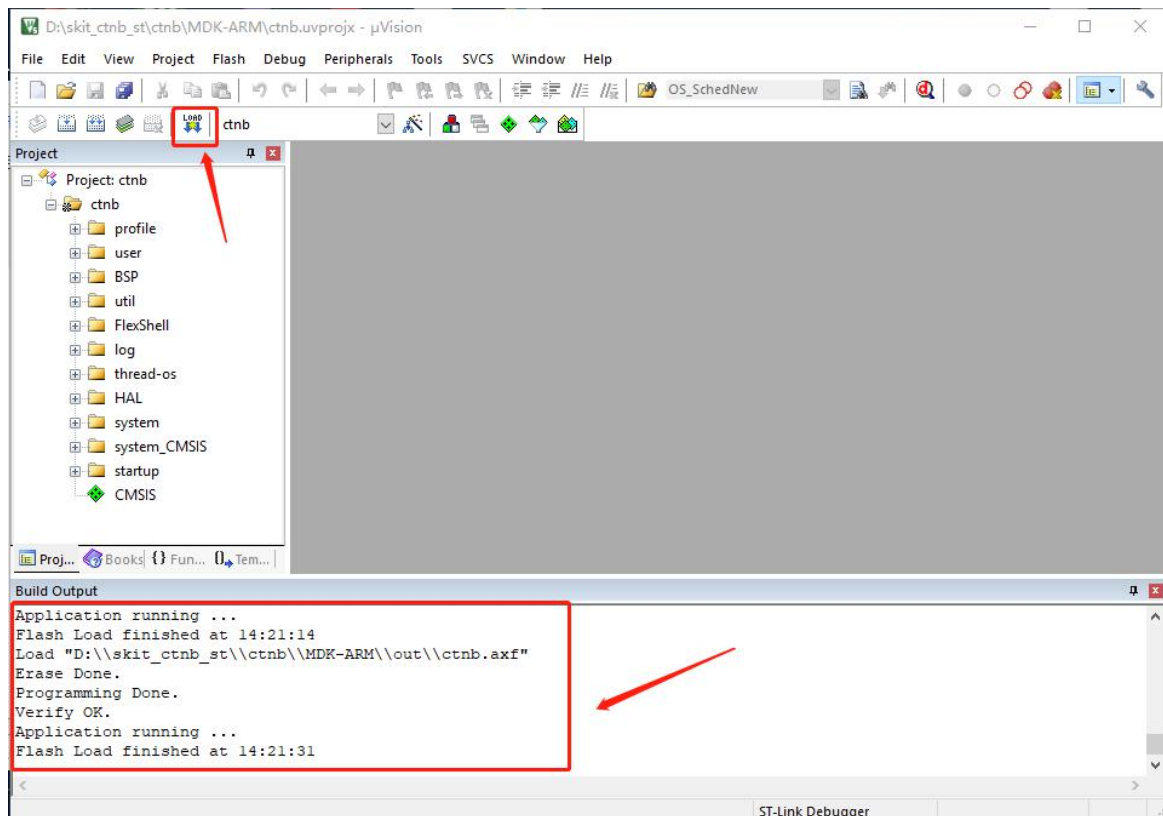

```
Build Output
..\system\Src\main.c(236): warning: #177-D: function "MX_USART2_UART_Init" was declared but never referenced
static void MX_USART2_UART_Init(void)
..\system\Src\main.c(201): warning: #177-D: function "MX_RTC_Init" was declared but never referenced
static void MX_RTC_Init(void)
..\system\Src\main.c: 3 warnings, 0 errors
compiling stm32flxx_hal_msp.c...
compiling stm32flxx_it.c...
compiling system_stm32flxx.c...
assembling startup_stm32f103xb.s...
linking...
Program Size: Code=38854 RO-data=2546 RW-data=3292 ZI-data=5644
".\out\ctnb.axf" - 0 Error(s), 7 Warning(s).
Build Time Elapsed: 00:00:55
```

ST-Link Debugger

下载开发板程序



编译成功后点击 load 图标 进行下载调试，下载成功后输出如下图：



数据上下行测试

产品概况

服务定义

设备管理

事件上报

数据查看

指令下发日志

订阅管理

远程升级管理

消息跟踪

导出任务

设备影子

9393671527994bf2bdb1c9e3c5e33a7b

2020-11-13 00:00:00 至 2020-11-13 23:59:59

设备ID	上报时间	数据	操作
9393671527994bf2bdb1c9e3c5e33a7b	2020-11-13 08:38:00	{"sinr":174,"rsrp":-743,"pci":149,"ecr":0,"cell_id":186167507}	🔍
9393671527994bf2bdb1c9e3c5e33a7b	2020-11-13 08:37:57	{"software_version":"V1.1","hardware_version":"V1.0","IMEI":"86...	🔍

< 最佳实践_NB智慧农业

产品概况

服务定义

设备管理

事件上报

数据查看

指令下发日志

订阅管理

远程升级管理

消息跟踪

导出任务

设备影子

请输入IMEI、设备ID

开始日期 至 结束日期

请选择事件类型

IMEI	设备ID	事件类型	事件内容	上报时间	操作
861410048952120	ae83b0071cd441f0b671d4...	信息	{"ir_sensor_data":1}	2020-11-11 16:12:39	🔍
861410048952120	ae83b0071cd441f0b671d4...	信息	{"ir_sensor_data":0}	2020-11-11 16:12:37	🔍
861410048952120	ae83b0071cd441f0b671d4...	信息	{"ir_sensor_data":1}	2020-11-11 16:11:13	🔍
861410048952120	ae83b0071cd441f0b671d4...	信息	{"ir_sensor_data":0}	2020-11-11 16:11:10	🔍
861410048952120	ae83b0071cd441f0b671d4...	信息	{"ir_sensor_data":1}	2020-11-11 16:00:18	🔍
861410048952120	ae83b0071cd441f0b671d4...	信息	{"ir_sensor_data":0}	2020-11-11 16:00:13	🔍
861410048952120	ae83b0071cd441f0b671d4...	信息	{"ir_sensor_data":1}	2020-11-10 11:06:42	🔍
861410048952120	ae83b0071cd441f0b671d4...	信息	{"ir_sensor_data":1}	2020-11-10 11:06:39	🔍

下行指令测试：

产品概况

服务定义

设备管理

事件上报

数据查看

指令下发日志

订阅管理

远程升级管理

消息跟踪

导出任务

设备影子

请输入设备名称、设备ID、IMEI

在线状态

设备状态

导入模板下载

添加设备

批量删除

导入

导出

设备名称	设备ID	IMEI	IMSI	创建时间	最后上线时间	最后离线时间	状态	操作
8614100...	9393671...	8614100...		2020-11-13 08:...	2020-11-13 08:...	--	已激活	<div><div>🔍</div><div>🔧</div><div>📄</div><div>🗑️</div><div>🔄</div><div>📶</div><div>📡</div><div>📶</div><div>📡</div><div>📶</div><div>📡</div></div>

点击下行指令按钮，选择下行指令的服务标识，填入指令控制值

指令下发

* 服务标识

motor_control_cmd

motor_control_cmd

set_auto_control

set_report_period

* motor_control (电机控制)

请选择motor_contro...的值

确定

取消

指令执行完成后，进入指令下发日志查看指令下发情况

产品概况

服务定义

设备管理

事件上报

数据查看

指令下发日志

订阅管理

远程升级管理

消息跟踪

导出任务

设备影子

设备日志 / 分组日志

9393671527994bf2b

请输入指令ID

搜索

开始日期

至

结束日期

请选择下发状态

设备ID	IMEI号	指令ID	指令下发状态	指令下发时间	指令完成时间	指令级别	操作员	操作
939367152799...	861410048952...	2	指令已完成	2020-11-13 09:22...	2020-11-13 09:22...	设备级	xiongw...	<div>点击查看详情</div>
939367152799...	861410048952...	1	指令已完成	2020-11-13 09:21...	2020-11-13 09:21...	设备级	xiongw...	

详情

指令ID2

指令内容{"motor_control": "0"}

返回内容{"motor_control": 0, "act_result": 0}

ttl7200

关闭

实践样例说明

(1) 终端开发板只在成功登录平台时发送一次“信号数据上报”和“设备信息上报”。

- 1、终端初始化，初始化采集线程，初始化 at 线程（包括 at 通知处理线程、bsp 初始化线程、bsp 事件处理线程、aep 数据处理线程、at 下行处理线程及 at 初始化线程）及 aep 业务处理线程
- 2、在数据采集线程进行周期性的温湿度采集
- 3、Bsp 初始化线程（BC28_init_process）进行芯片状态查询及 lwm2m 参数配置及登录
- 4、登录成功后，在 aep 业务处理线程（app_process）进行版本信息及网络信息的上报

```

PROCESS_THREAD(app_process, ev, pdata)
{
    int8_t ret;
    static uint8_t IR_status = true;

    THREAD_BEGIN()

    getStatus(IR, &BSP_status);
    IR_status = BSP_status.IR_status;
    if( IR_status == 1 )
    {
        setStatus(LED0_DEVICE_ID, LED_BLUE);
    }

    //send device info
    send_info_report_data();
    THREAD_OS_DELAY(3000);
    //send network signal info
    send signal report data();
}

```

(2) 门禁告警

在 aep 业务处理线程（app_process）周期性采集红外线状态，在红外线状态变更时，主动上报红外遮挡告警事件。

```

    getStatus(IR, &BSP_status);
    if( BSP_status.IR_status != IR_status )
    {
        THREAD_OS_DELAY(500);

        getStatus(IR, &BSP_status);
        if( BSP_status.IR_status != IR_status )
        {
            os_log("IR status change\r\n");
            if( BSP_status.IR_status == 1 )
            {
                setStatus(LED0_DEVICE_ID, LED_BLUE);
            }
            else
            {
                setStatus(LED0_DEVICE_ID, LED_RED);
            }
            IR_status = BSP_status.IR_status;
            send_ir_sensor_data();
        }
    }

    THREAD_OS_DELAY(200);
}

```

发送红外事件

(3) 温湿度实时监控

- 1、At 接收到下行指令后，交给 aep 数据处理线程（nb_rcvdata_process 下的 decode_aep_data 过程）进行处理，
- 2、如果是温湿度实时监控的开启指令（set_auto_control），则启动定时器（nb_timer_cb），进行定时数据上报，同时回复数据处理结果。

```

,
else if( strcmp(result.serviceIdentifier, "set_auto_control") == 0 )
{
    set_auto_control_resp ack;
    AepString rsp;
    uint8_t value;

    value = *((uint8_t *)result.data);
    app_log("get value:%d\r\n",value);
    if(value == 0)
    ,

static void nb_timer_cb(void *args)
{
    int8_t ret;
    getStatus(HUM_TEMP, &BSP_status);

    data_report reportFlag;
    AepString      reportstr;

    reportFlag.temperature_data = ((float)((int)(BSP_status.temp*10)))/10;
    reportFlag.humidity_data = BSP_status.hump;
    reportstr = data_report_CodeDataReport(reportFlag);
    app_log(reportstr.str);
    ret = send_msg(reportstr.str, reportstr.len, SEND_MODE CON);
    if( ret < 0 )
    {
        app_log("send temperatute humidity data err\r\n");
    }
    free(reportstr.str);

    soft_timer_start(nb_timer, reportPeriod);
}

```

(4) 手动控制通风装置的开启和关闭

- 1、At 接收到下行指令后, 交给 aep 数据处理线程(nb_rcvdata_process 下的 decode_aep_data 过程) 进行处理,
- 2、如果是电机控制指令 (motor_control_cmd), 根据指令要求设置电机状态 (setStatus), 回复处理结果。

```

void decode_aep_data(char *data)
{
    int8_t ret;
    AepCmdData result;

    result = decodeCmdDownFromStr(data);
    if( result.code == AEP_CMD_SUCCESS )
    {
        if( strcmp(result.serviceIdentifier, "motor_control_cmd") == 0 )
        {
            motor_control_resp ack;
            AepString rsp;
            uint8_t value;

            value = *((uint8_t *)result.data);

```


附录：实践样例 API 说明

简介	程序主入口（app_thread_init）
函数名	void app_thread_init(void)
参数	无
返回	无

简介	采集线程初始化（bsp_thread_init）
函数名	void bsp_thread_init(void)
参数	无
返回	无

简介	At 线程初始化（nb_thread_init）
函数名	void nb_thread_init(void)
参数	无
返回	无

简介	At 线程初始化（aep_thread_init）
函数名	void aep_thread_init(void)
参数	无
返回	无

1) 数据采集线程

简介	定时数据采集（bsp_process）
函数名	PROCESS_THREAD(bsp_process, ev, pdata)
参数	ev: 线程执行 event 参数 pdata: 线程执行 data 参数
返回	无

简介	采集温度数据（GetTemperature）
函数名	HTS221_StatusTypeDef HTS221_GetTemperature(float *pfData)
参数	float *pfData, 输出采集到的温度数据
返回	采集操作结果： HTS221_OK = 0x00U, //成功 HTS221_ERR = 0x01U, //错误 HTS221_BUSY = 0x02U, //设备忙 HTS221_TIMEOUT = 0x03U, //处理超时

简介	采集湿度数据（GetHumidity）
函数名	HTS221_StatusTypeDef HTS221_GetHumidity(uint16_t *pfData)
参数	uint16_t *pfData, 输出采集到的湿度数据

返回	采集操作结果： HTS221_OK = 0x00U, //成功 HTS221_ERR = 0x01U, //错误 HTS221_BUSY = 0x02U, //设备忙 HTS221_TIMEOUT = 0x03U, //处理超时
----	--

2) At 处理线程

A. at 初始化线程：初始化环境，驻网，获取网络参数

简介	At 初始化线程 (nb_init_process)
函数名	PROCESS_THREAD(nb_init_process, event, pdata)
参数	ev: 线程执行 event 参数 pdata: 线程执行 data 参数
返回	无

通过 at 配置服务器 ip 地址和端口

简介	配置服务器 ip 地址和端口 (at_config_IP_port)
函数名	static int8_t at_config_IP_port(char *serverIP, uint16_t port)
参数	char *serverIP: 服务器地址 uint16_t port: 服务器端口
返回	配置结果：

配置 lwm2m lifetime

简介	配置 lwm2m lifetime (at_config_lifetime)
函数名	static int8_t at_config_lifetime(uint32_t lifetime)
参数	uint32_t lifetime: lwm2m lifetime
返回	配置结果：

配置 lwm2m endpointname

简介	配置 lwm2m endpointname (at_config_endpoint_name)
函数名	static int8_t at_config_endpoint_name(const char *IMEI)
参数	const char *IMEI: 芯片 imei 号
返回	配置结果：

登录 lwm2m 服务器

简介	登录 lwm2m 服务器 (at_lwm2m_register)
函数名	static int8_t at_lwm2m_register(void)
参数	无
返回	登录指令 at 发送结果：

登出 lwm2m 服务器

简介	登出 lwm2m 服务器 (at_lwm2m_deregister)
----	------------------------------------

函数名	static int8_t at_lwm2m_deregister(void)
参数	无
返回	登出指令 at 发送结果:

查询 lwm2m 服务器登录状态

简介	查询 lwm2m 服务器登录状态 (at_lwm2m_get_status)
函数名	static int8_t at_lwm2m_get_status(void)
参数	无
返回	当前登录状态

发送 lwm2m 数据

简介	发送 lwm2m 数据 (at_send_data)
函数名	static int8_t at_send_data(void *data_type)
参数	void *data_type: 需要发送的数据
返回	发送指令操作结果

获取 CON 报文发送结果

简介	CON 报文发送结果 (at_get_CON_result)
函数名	int8_t at_get_CON_result(void *thread)
参数	void *thread: 发送报文的原始线程
返回	当前登录状态

BC28 参数获取及 lwm2m 参数设置

简介	BC28 参数获取及 lwm2m 参数设置 (BC28_init_process)
函数名	PROCESS_THREAD(BC28_init_process, event, pdata)
参数	ev: 线程执行 event 参数 pdata: 线程执行 data 参数
返回	无

BC28 事件处理

简介	BC28 事件处理 (BC28_event_process)
函数名	PROCESS_THREAD(BC28_event_process, event, pdata)
参数	ev: 线程执行 event 参数 pdata: 线程执行 data 参数
返回	无

B. at 下行处理线程: 处理 at 响应结果及芯片主动发送的指令

简介	At 下行处理线程 (nb_rsp_process)
函数名	PROCESS_THREAD(nb_rsp_process, ev, pdata)
参数	ev: 线程执行 event 参数 pdata: 线程执行 data 参数
返回	无

处理 at 应答

简介	处理 at 应答 (at_process_rsp)
函数名	static int8_t at_process_rsp(void *data)
参数	void *data: at 下行的结果数据
返回	无

处理 mcu at 通知

简介	处理 mcu at 通知 (at_process_URC)
函数名	static void at_process_URC(void *data)
参数	void *data: at 下行的通知
返回	无

C. aep 接收指令、数据处理线程：处理 at 指令下行的有关 aep 的下行的指令及数据

简介	aep 接收指令、数据处理线程 (nb_recvdata_process)
函数名	PROCESS_THREAD(nb_recvdata_process, event, pdata)
参数	ev: 线程执行 event 参数 pdata: 线程执行 data 参数
返回	无

对 aep 下行指令进行解码并处理应答

简介	处理 aep 下行指令 (decode_aep_data)
函数名	void decode_aep_data(char *data)
参数	char *data: at 下行的 aep 指令
返回	无

定时上报

简介	定时数据上报 (nb_timer_cb)
函数名	static void nb_timer_cb(void *args)
参数	void *args: 未用
返回	无

3) Aep 业务处理线程：上报版本信息、网络信息，处理红外遮挡事件

简介	定时数据采集 (bsp_process)
函数名	PROCESS_THREAD(app_process, ev, pdata)
参数	ev: 线程执行 event 参数 pdata: 线程执行 data 参数
返回	无

简介	获取硬件状态 (getStatus)
函数名	uint8_t getStatus(BSP_DEVICE_ID device_id, BSP_status_t *bsp_status)

参数	BSP_DEVICE_ID device_id: LED0_DEVICE_ID =0, //LED 0 LED1_DEVICE_ID = 1, //LED 1 MOTOR1 = 2, //马达 HUM_TEMP = 3, //温湿度 KEY1 = 4, //按键 1 KEY2 = 5, //按键 2 IR = 6, //红外线 BSP_ALL BSP_status_t *bsp_status: typedef struct { uint8_t led0_status; //LED 0 状态 uint8_t led1_status; //LED 1 状态 uint8_t motor_status; //马达状态 float temp; //温度状态 uint16_t hump; //湿度状态 uint8_t key_value; //按键状态 uint8_t IR_status; //红外线状态 }BSP_status_t;
返回	处理结果：0-成功，其它-失败

简介	设置硬件状态（setStatus）
函数名	void setStatus(BSP_DEVICE_ID device_id,unsigned char status)
参数	BSP_DEVICE_ID device_id: LED0_DEVICE_ID =0, //LED 0 LED1_DEVICE_ID = 1, //LED 1 MOTOR1 = 2, //马达 HUM_TEMP = 3, //温湿度 KEY1 = 4, //按键 1 KEY2 = 5, //按键 2 IR = 6, //红外线 unsigned char status
返回	处理结果：0-成功，其它-失败

简介	发送软硬件版本、iccid、imei 等信息（ send_info_report_data）
函数名	static void send_info_report_data(void)
参数	无
返回	无

简介	发送网络信号状态信息（send_signal_report_data）
函数名	static void send_signal_report_data(void)
参数	无

返回	无
----	---

简介	发送红外状态信息（ send_ir_sensor_data）
函数名	static void send_ir_sensor_data(void)
参数	无
返回	无