



MIoT 智能插座 Demo 文档

Confidential

目录

1.小米模组工作原理.....	4
2.开发者平台创建产品.....	4
2.1 创建一个灯类产品.....	4
2.2 开发者平台基础配置.....	5
2.3 开发者平台功能定义.....	5
3.固件开发.....	6
3.1 开发者平台固件开发配置.....	6
3.2 MCU 端固件开发.....	6
3.2.1 下载 Demo 程序.....	7
3.2.2 打开 Demo 工程并编译.....	7
3.2.3 自动生成代码.....	7
3.2.4 编辑代码.....	9
4.高阶配置（自动化部分）.....	11
5.扩展程序开发.....	12
6.使用小爱控制.....	13
7.注意事项.....	13

版本信息（内部使用）

版本	内容	时间	作者
1.0.0	初始版本	2020.6.28	龚治威

本文概要

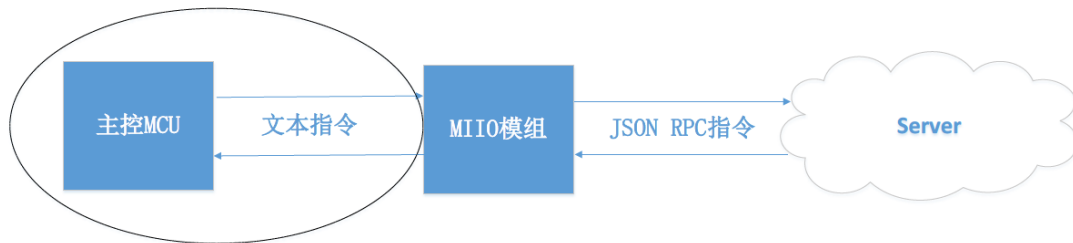
主要介绍了基于 WiFi 模组+MCU 的开发方式开发的一款智能插座 Demo，该智能插座 Demo 主要实现了远程独立开关、远程 OTA 升级、小爱语音控制、设备联动等功能。开发者可以通过参考本文的内容熟悉小米开发平台的开发步骤，快速的创建出产品。

参考

1. [小米 IoT 平台设备接入引导](#)
2. [通用模组标准协议开发指南](#)

Confidential

1.小米模组工作原理



目前 WiFi 模组与 WiFi+BLE Combo 模组，支持“标准通用模组+MCU”和“基于模组 SDK 二次开发”这两种开发模式。本文主要针对“标准通用模组+MCU”这种开发模式展开说明。对于“标准通用模组+MCU”的开发模式，主要关注模组与 MCU 之间的串口指令通信。

2.开发者平台创建产品

设备如何接入小米 IoT，参考：<https://iot.mi.com/new/doc/guidelines-for-access/direct-access/overview.html>

2.1 创建一个灯类产品



新建产品

* 产品名称: 插座demo

* 产品类型: 电源开关 / 插座 / 插排

* 产品型号: xxxx

* 联网方式: WiFi

备注(选填): 新建一个智能插座demo

复用产品(选填): 请选择源产品 请选择源产品功能定义

取消 确定

2.2 开发者平台基础配置

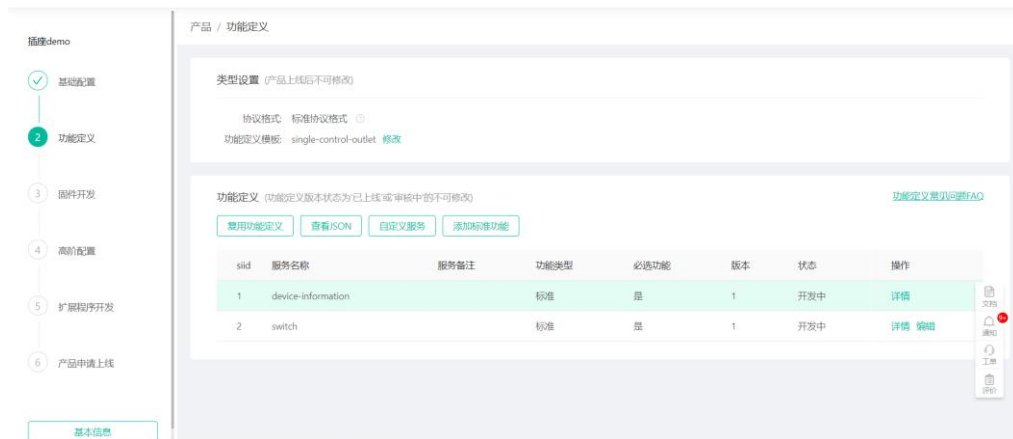


开发者需要按照要求上传产品相关图片，正确填写/选择相关信息。

2.3 开发者平台功能定义

根据产品功能，选择功能定义模板，模板不能满足需求时可以自定义相关功能。





3. 固件开发

3.1 开发者平台固件开发配置



3.2 MCU 端固件开发

模组与 MCU 的交互，log 的查看，OTA 打包，可参考：

<https://iot.mi.com/new/doc/embedded-development/wifi/standard-protocol.html>

此工程实现了模组+MCU 开发方式中 MCU 端的基本功能，阅读文件夹下的《MCU_Demo 二次开发手册 v1.2.pdf》，内有详细内容。

3.2.1 下载 Demo 程序

下载代码: [MCU Demo 程序下载](#)

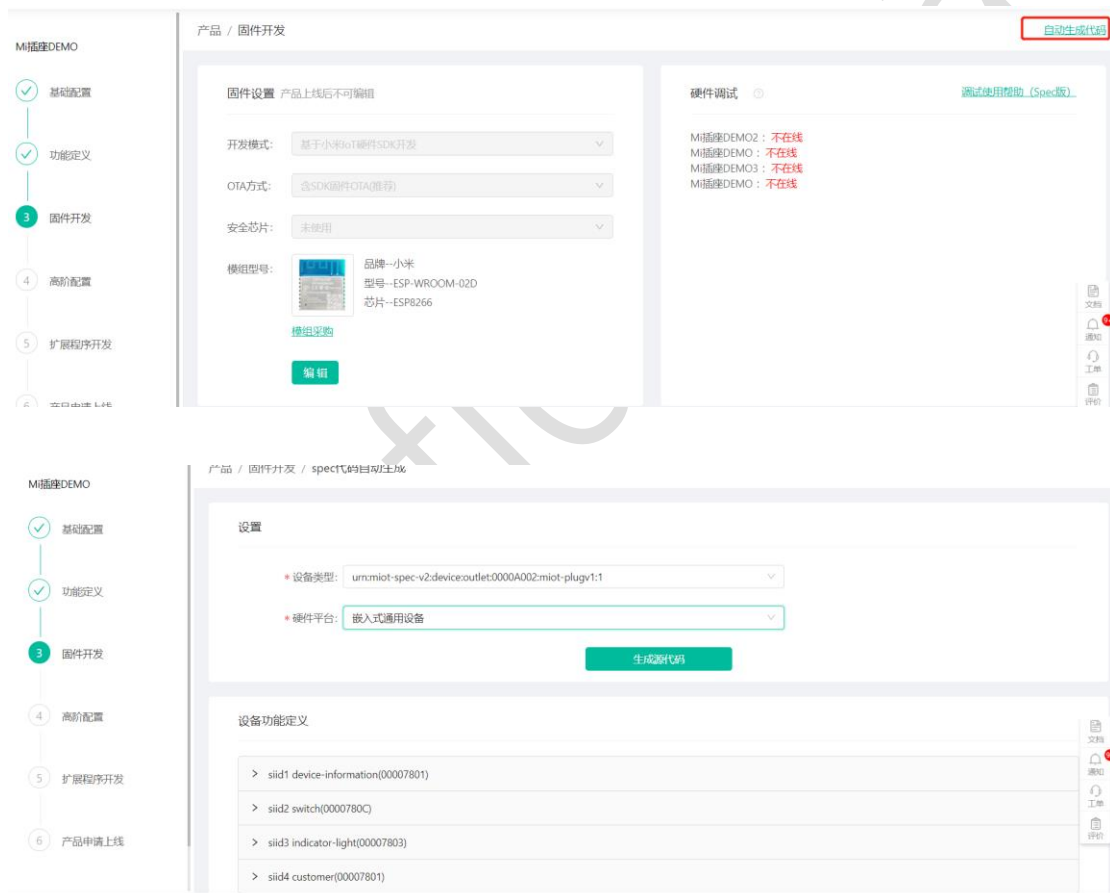
3.2.2 打开 Demo 工程并编译

进入 `mcu_demo\arch\stm32\MDK-ARM` 使用 Keil 集成开发工具打开 `mcu_demo.uvprojx`:

编译: 在 Keil 中, 选择 `project->Build Target` 即可开始编译工程。

3.2.3 自动生成代码

创建产品定义完功能后, 开发者可以使用平台提供的“代码自动工具”, 生成与功能定义符合的 spec 协议代码。可以节省 Coding 和调试时间。自动生成的代码适用于 **模组 SDK 二次开发**, 在 MCU_Demo 中使用需要将自动生成的代码做一些调整。



如下是自动生成代码的源码目录结构:

脑 > 下载 > miot_device_generic_outlet_20200622_152631 (2) > main > device >			
名称	修改日期	类型	大小
codec	2020/6/22 15:27	文件夹	
handler	2020/6/22 15:27	文件夹	
iid	2020/6/22 15:27	文件夹	
typedef	2020/6/22 15:27	文件夹	
utils	2020/6/22 15:27	文件夹	
operation_executor.c	2020/6/22 15:26	C 文件	10 KB
operation_executor.h	2020/6/22 15:26	H 文件	1 KB

生成源代码后，只需要将 handler 目录和 iid 目录中相关的内容复制到 MCU_Demo 源码路径 mcu_demo\user\handler\ 中（需要修改头文件包含路径，去掉 include 中的路径）例如：

```
#include "../typedef/action_operation.h" ---> #include "action_operation.h"
```

另外 xxx_doChange.c 文件中并未被调用的函数需要注释掉，例如 xxx_doChange_notify 函数和 xxx_doChange_cb 函数：

```
void P_2_4_Mode_doChange_notify(int newValue)
{
    arch_os_async_call(P_2_4_Mode_doChange_cb,(void *)newValue,500);
}

static void P_2_3_Status_doChange_cb(void * newValue)
{
    int value = (int) newValue;
    if (NULL == g_mio_instance_handle)
    {
        LOG_ERROR("please create mio instance first!\r\n");
        return;
    }
    P_2_3_Status_doChange(g_mio_instance_handle,value);
    return;
}
```

以下是复制完成后在 mcu_demo\user\handler\ 中的文件：

此电脑 > 本地磁盘 (D:) > work > code > test > user > handler				
名称	修改日期	类型	大小	
on_property_set.c	2020/5/25 17:28	C 文件	2 KB	
on_property_set.h	2020/5/25 17:28	H 文件	1 KB	
print_value.c	2020/5/25 17:28	C 文件	2 KB	
print_value.h	2020/5/25 17:28	H 文件	1 KB	
S_1_DeviceInformation_doGet.c	2020/5/25 17:28	C 文件	2 KB	
S_1_DeviceInformation_doGet.h	2020/5/25 17:28	H 文件	1 KB	
S_2_Switch_doChange.c	2020/5/25 17:28	C 文件	2 KB	
S_2_Switch_doChange.h	2020/5/25 17:28	H 文件	2 KB	
S_2_Switch_doGet.c	2020/5/25 17:28	C 文件	4 KB	
S_2_Switch_doGet.h	2020/5/25 17:28	H 文件	1 KB	
S_2_Switch_doSet.c	2020/5/25 17:28	C 文件	3 KB	
S_2_Switch_doSet.h	2020/5/25 17:28	H 文件	1 KB	
S_3_IndicatorLight_doChange.c	2020/5/25 17:28	C 文件	2 KB	
S_3_IndicatorLight_doChange.h	2020/5/25 17:28	H 文件	2 KB	
S_3_IndicatorLight_doGet.c	2020/5/25 17:28	C 文件	3 KB	
S_3_IndicatorLight_doGet.h	2020/5/25 17:28	H 文件	1 KB	
S_3_IndicatorLight_doSet.c	2020/5/25 17:28	C 文件	7 KB	
S_3_IndicatorLight_doSet.h	2020/5/25 17:28	H 文件	1 KB	
S_4_Customer_doAction.c	2020/5/25 17:28	C 文件	3 KB	
S_4_Customer_doAction.h	2020/5/25 17:28	H 文件	1 KB	
S_4_Customer_doChange.c	2020/5/25 17:28	C 文件	1 KB	
S_4_Customer_doChange.h	2020/5/25 17:28	H 文件	1 KB	
S_4_Customer_doEvent.c	2020/5/25 17:28	C 文件	2 KB	

3.2.4 编辑代码

更改产品的版本和产品 model:

```
mcu_demo\user\user_config.h

#define USER_MODEL "miot.plugin.plugv1" // Device Model
#define USER_MCU_VERSION "0001" // Device firmware version
```

配置 MCU 控制引脚 GPIO:

```
#define SWITCH GPIO_Pin_4 //GPIOA 4 插座开关控制引脚
#define RESTORE GPIO_Pin_5 //GPIOA 5 恢复出厂设置控制引脚
```

插座远程开关控制:

```
static void P_2_1_On_doSet(property_operation_t *o)
{
    // 判断数据格式是否正确, 如果错误, 返回代码: OPERATION_ERROR_VALUE
    if (o->value->format != PROPERTY_FORMAT_BOOLEAN)
    {
        o->code = OPERATION_ERROR_VALUE;
        return;
    }

    // TODO: 执行写操作: o->value->data.boolean.value;
    GPIO_WriteBit(GPIOA, SWITCH, o->value->data.boolean.value != false);

    // 如果成功, 返回代码: OPERATION_OK
    o->code = OPERATION_OK;

    return;
}
```

远程获取插座状态:

```
/**
 * 格式: property_value_new_boolean(true 或 false)
 * 取值: true 或 false
 */
static void P_2_1_On_doGet(property_operation_t *o)
{
    o->value = property_value_new_boolean(GPIO_ReadOutputDataBit(GPIOA, SWITCH) !=
    Bit_RESET);
    // TODO: 这里需要读到属性真正的值
}
```

插座状态改变主动上报:

```
int user_app_main(int argc, void *argv)
{
    ...
    if (old_status != GPIO_ReadOutputDataBit(GPIOA, SWITCH)) {
        old_status == GPIO_ReadOutputDataBit(GPIOA, SWITCH);
        P_2_1_On_doChange(argv, old_status != Bit_RESET);
    }
```

恢复出厂设置:

```
int user_app_main(int argc, void *argv)
{
    if (GPIO_ReadOutputDataBit(GPIOA, RESTORE) != Bit_RESET) {
        app_func_restore(argv);
    }
```

配置 MCU OTA 升级支持:

配置宏 USER_OTA_ENABLE 为 1, 则支持 MCU OTA 升级, 若配置为 0 则不支持 MCU OTA 升级

```
/* user ota configurations */
#define USER_OTA_ENABLE          (1)
#if (defined(USER_OTA_ENABLE) && USER_OTA_ENABLE)
#define XMODEM_PRINT_LOG        (1)
#endif
```

MCU OTA 升级 MCU 端写 flash 函数:

添加写 MCU flash 函数, 将下载的 OTA 数据包写到 MCU flash 中(涉及到 MCU boot 的流程, 此部分代码由开发者自行实现)

```
int arch_ota_func(unsigned char *pbuf, size_t len, size_t sum)
{
    /* trans data to MCU flash here */
    return MIIIO_OK;
```

```
}
```

4. 高阶配置（自动化部分）

高阶配置中消息推送、自动化、语控服务部分属于可选项，开发者可根据产品特性选择性配置，下面以自动化为例，在平台创建自动化后，在 app 端就可以创建相应的场景。





5. 扩展程序开发

可以使用 RN 开发自定义插件，也可以使用平台公版插件，自定义插件开发参考：

<https://iot.mi.com/new/doc/guidelines-for-access/direct-access/develop-extension.html>



拓展程序手机端显示效果



6. 使用小爱控制

功能定义使用标准的 spec 定义，产品即可支持小爱语控，详细可以参考 [《小爱语控与功能定义使用规范》](#)。

7. 注意事项

1. 自定义属性不支持小爱语控。
2. 自动化配置设备联动需要多台设备。