泰芯半导体
TaiXin Semiconductor

泰芯 TXW8xx SDK BLE 配网开发指南

珠海泰芯半导体有限公司
Zhuhai Taixin Semiconductor Co.,Ltd

珠海市高新区港湾一号科创园港 11 栋 3 楼

责任与版权

责任限制

由于产品版本升级或者其他原因，本文档会不定期更新。除非另行约定，泰芯半导体有限公司对本文档所有内容不提供任何担保或授权。

客户应在遵守法律、法规和安全要求的前提下进行产品设计，并做充分验证。泰芯半导体有限公司对应用帮助或客户产品设计不承担任何义务。客户应对其使用泰芯半导体有限公司的产品和应用自行负责。

在适用法律允许的范围内，泰芯半导体有限公司在任何情况下，都不对因使用本文档相关内容及本文档描述的产品而产生的损失和损害进行超过购买支付价款的赔偿（除在涉及人身伤害的情况中根据适用的法律规定的损害赔偿外）。

版权申明

泰芯半导体有限公司保留随时修改本文档中任何信息的权利，无需提前通知且不承担任何责任。

未经泰芯半导体有限公司书面同意，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。除非获得相关权利人的许可，否则，任何人不能以任何形式对前述软件进行复制、分发、修改、摘录、反编译、反汇编、解密、反向工程、出租、转让、分许可等侵犯本文档描述的享有版权的软件版权的行为，但是适用法禁止此类限制的除外。

修订记录

日期	版本	描　　述	修订人
2023-11-2	V1.0	初始版本	DY

目录

1. 概述

泰芯 TXW8xx SDK 支持 BLE 配网功能，支持 3 种配网方式：

- **BLE广播配网：**设备以广播方式进行通信，无需建立连接，而且不可扫描。SDK Demo代码对接了泰芯微信小程序：TXBLE配网，使用该小程序可以直接发送广播包对设备进行参数设置。
- **BLE可扫描广播配网：**支持BLE协议的广播/扫描功能，手机端可以扫描发现设备，但是不能进行连接。手机扫描发现设备后，可发送特定的广播包对设备进行参数设置。SDK Demo代码对接了泰芯微信小程序：TXBLE配网，使用该小程序可以直接发送广播包对设备进行参数设置。
- **BLE连接配网：**支持BLE协议连接，手机App扫描发现设备并进行连接，设备提供了配网服务。App通过自定义的属性特征进行参数设置。

SDK 包含了 3 种配网方式的 demo 代码，代码文件：sdk/lib/ble/ble_demo.c

使用此模块需声明以下宏定义：

- #define BLE_EN
- #define BLE_DEMO_MODE 0

BLE_DEMO_MODE 有以下取值：

- 0：使用通用的集成接口，传入参数需包含使能的模式；
- 1：BLE广播配网；
- 2：BLE可扫描广播配网；
- 3：BLE连接配网；

2. BLE 广播配网

2.1. BLE 广播配网初始化

使用 BLE 广播配网之前，需要对模块进行初始化，可直接使用 ble_demo.c 提供的示例代码，如下图所示。

```
int32 ble_demo_mode1_init(struct bt_ops *bt_ops)
{
    ble_adv_init(bt_ops);

    return ble_ll_open(bt_ops, 0, 38);
}
```

ble_adv_init 对模块进行初始化会为设备接收到的广播数据声明回调处理函数。

初始化完成后，使用宏 ble_ll_open 打开 BLE 广播配网模式。ble_ll_open 参数说明如下：

ble_ll_open(ops, type, chan)

参数说明：

- ops： bt_ops
- type： 0 - BLE广播配网模式
- chan： 该参数固定输入 38

返回值：

- 返回 0： 开启成功
- 非 0： ERROR

2.2. 解析小程序发送的参数

TXW8xx SDK 支持泰芯 TXBLE 配网微信小程序进行配网。使用泰芯 TXBLE 配网微信小程序对设备进行配网操作流程如下图所示。

进行上述操作后，手机端开始持续广播带联网信息数据的广播包，而设备端进入 BLE 广播配网模式后，则开始接收范围内的各种广播数据。广播数据会进行过滤和接收超时处理，从而筛选出目标的广播数据。

设备获取到目标的广播数据后，则会交给 ble_adv_parse_param 解析，由此获取到必要的配网信息。解析数据过程如下。

```c
void ble_adv_parse_param(uint8 *data, int len)
{
    uint8 *ptr = data;
    extern struct sys_config sys_cfgs;
#if 1 //sample code
    uint8 buff[33];
    while (ptr < data + len) {
        switch (ptr[0]) {
            case 1: //SSID
                os_memset(buff, 0, sizeof(buff));
                os_memcpy(buff, ptr + 2, ptr[1]);
                os_printf("SET ssid:%s\r\n", buff);
                os_memcpy(sys_cfgs.ssid, buff, sizeof(buff));
                break;
            case 2: //PassWord
                os_memset(buff, 0, sizeof(buff));
                os_memcpy(buff, ptr + 2, ptr[1]);
                os_printf("SET passwd:%s\r\n", buff);
                os_memcpy(sys_cfgs.passwd, buff, sizeof(buff));
                break;
            case 3: //Keymgmt
                os_printf("SET keymgmt:%s\r\n", ptr[2] ? "WPA-PSK" : "NONE");
                sys_cfgs.key_mgmt = ptr[2] ? WPA_KEY_MGMT_PSK : WPA_KEY_MGMT_NONE;
                break;
            case 4: //auth
                os_printf("AUTH %d\r\n", ptr[2]);
                break;
            default:
                os_printf("Unsupport ID:%d\r\n", ptr[0]);
                break;
        } « end switch ptr[0] »
        ptr += (ptr[1] + 2);
    } « end while ptr<data+len »

    syscfg_dump();

    ble_network_configured();
    SYSEVT_NEW_BLE_EVT(SYSEVT_BLE_NETWORK_CONFIGURED, 0);
#endif
} « end ble_adv_parse_param »
```

设置 SSID

设置 Password

设置 Keymgmt

设备联网

回抛 BLE 事件

ble_network_configured 会对解析到的联网参数进行保存。由于 TXW8xx 的 BLE 功能和 WiFi 功能不能同时工作，所以这里需要关闭蓝牙 BLE 模式，进入 WIFI 模式进行联网。

```c
static int32 ble_network_configured(void)
{
    extern void *g_ops;

    struct lmac_ops *lops = (struct lmac_ops *)g_ops;
    struct bt_ops *bt_ops = (struct bt_ops *)lops->btops;

    ble_demo_stop(bt_ops);

    wpa_passphrase(sys_cfgs.ssid, sys_cfgs.passwd, sys_cfgs.psk);
    ieee80211_conf_set_ssid(WIFI_MODE_STA, sys_cfgs.ssid);
    ieee80211_conf_set_psk(WIFI_MODE_STA, sys_cfgs.psk);
    ieee80211_conf_set_keymgmt(WIFI_MODE_STA, sys_cfgs.key_mgmt);
    sys_cfgs.wifi_mode = WIFI_MODE_STA;
    syscfg_save();

    ieee80211_iface_stop(WIFI_MODE_AP);
    wificfg_flush(WIFI_MODE_STA);
    netdev_set_wifi_mode((struct netdev *)dev_get(HG_WIFI0_DEVID), WIFI_MODE_STA);
    ieee80211_iface_start(WIFI_MODE_STA);

    return RET_OK;
} « end ble_network_configured »
```

退出蓝牙 BLE 模式

计算联网密钥

联网参数保存

3. BLE 可扫描广播配网

3.1. BLE 可扫描广播配网初始化

使用 BLE 可扫描广播配网之前，需要对模块进行初始化，可直接使用 ble_demo.c 提供的示例代码，如下图所示。

```
int32 ble_demo_mode2_init(struct bt_ops *bt_ops)
{
    ble_adv_init(bt_ops);

    uint8 adv_data[] = {    0x02, 0x01, 0x06,

                            0x03, 0x02, 0x01, 0xA2,

                            0x14, 0x16, 0x01, 0xA2, 0x01, 0x6B, 0x65, 0x79, 0x79, 0x66, \
                            0x67, 0x35, 0x79, 0x33, 0x34, 0x79, 0x71, 0x78, 0x71, 0x67, 0x64};

    uint8 scan_resp[] = {   0x04, 0x09, 0x53, 0x53, 0x53,

                            0x19, 0xFF, 0xD0, 0x07, 0x01, 0x03, 0x00, 0x00, 0x0C, 0x00, \
                            0x88, 0xD1, 0xC4, 0x89, 0x2B, 0x56, 0x7D, 0xE5, 0x65, 0xAC, \
                            0xA1, 0x3F, 0x09, 0x1C, 0x43, 0x92};

    ble_ll_set_advdata(bt_ops, adv_data, sizeof(adv_data));
    ble_ll_set_scan_rsp(bt_ops, scan_resp, sizeof(scan_resp));
    ble_ll_set_adv_en(bt_ops, 1);

    return ble_ll_open(bt_ops, 1, 38);
} « end ble_demo_mode2_init »
```

Length UUID Data

设备名称

相较于 BLE 广播配网模式，BLE 可扫描广播配网需要设置广播数据和扫描响应数据，同时还需要打开设备 keep RX 和广播功能，具体操作如下：

1) 设置设备的BLE广播数据，该广播数据用于手机发现设备。

广播数据内容为 AdvData 部分，如下图所示。

Preamble (1 octet)	Access Address (4 octet)	Adv Header (2 octet)	AdvA (6 octets)	AdvData (0-31 octets)	CRC (3 octets)

设备将以 ADV_IND 类型发送广播数据。

设置广播数据的宏为：

ble_ll_set_advdata(ops, adv_data, len)

参数说明：

- ops: btops
- adv_data: 广播数据
- len: 广播数据的长度

返回值：

- 返回 0： 设置成功
- 非 0： ERROR

2) 设置设备的BLE扫描响应数据，该数据用于回应手机扫描请求，并携带设备信息。

响应数据的内容也为 AdvData 部分，如上图所示。

设置扫描响应数据的宏为：

```
ble_ll_set_scan_rsp(ops, scan_resp, len)
```

参数说明：

- ops： btops
- scan_data： 扫描响应数据
- len： 扫描响应数据的长度

返回值：

- 返回 0： 设置成功
- 非 0： ERROR

3) 使能设备keep RX和广播功能，芯片默认关闭此项功能。

开启此项功能后，BLE 可扫描广播配网模式才能保持发送和接收广播数据，若未使能此项功能，手机端无法扫描到设备。

开启此功能的宏为：

```
ble_ll_set_adv_en(ops, start)
```

参数说明：

- ops： btops
- start： 0 - 关闭，1 - 开启

返回值：

- 返回 0： 设置成功
- 非 0： ERROR

初始化完成后，使用宏 ble_ll_open 打开 BLE 可扫描广播配网模式。ble_ll_open 参数说明如下：

ble_ll_open(ops, type, chan)

参数说明：

- ops： bt_ops
- type： 1 - BLE可扫描广播配网模式
- chan： 该参数固定输入 38

返回值：

- 返回 0： 开启成功
- 非 0： ERROR

3.2. 解析小程序发送的参数

BLE 可扫描广播配网接收广播数据流程与 BLE 广播配网一致，请参考 2.2 章节。

3.3. ATT Table 介绍

进行 BLE 连接配网功能开发时，核心数据是 att_table，att_table 是设备配置的 GATT 服务信息。定义不同的 att_table，设备就会提供不同的服务，在进行自定义开发时根据实际需求配置服务信息。

TXW8xx SDK 中 ble_demo.c 对 att_table 声明如下图所示。

```
static const struct uble_gatt_data uble_demo_att_table[] = {
    /*Primary service 1: Generic Access service*/
    {0x2800/*Primary service*/, 0, 0x1800 /*Generic Access service*/},

    {0x2803/*Characteristic*/, UBLE_GATT_CHARAC_READ, 0x2A00 /*Device Name*/},
    {0x2A00/*Device Name*/, 0, NULL},

    {0x2803/*Characteristic*/, UBLE_GATT_CHARAC_READ, 0x2A01 /*Appearance*/},
    {0x2A01/*Appearance*/, 0, NULL},

    /*Primary service 2: app demo service*/
    {0x2800/*Primary service*/, 0, 0x1910 /*demo service*/},

    {0x2803/*Characteristic*/, UBLE_GATT_CHARAC_WRITE_WITHOUT_RESPONSE, 0x2b11 /*app write */},
    {0x2b11/*tuya app write*/, 0, (uint32) &uble_demo_values[0]},

    {0x2803/*Characteristic*/, UBLE_GATT_CHARAC_NOTIFY, 0x2b10 /*notify to app*/},
    {0x2b10/*notify to tuya app*/, 0, NULL},
    {0x2902/*Characteristic CCCD*/, 0, NULL},

    {0x2803/*Characteristic*/, UBLE_GATT_CHARAC_READ, 0x2b12 /*SSID read*/},
    {0x2b12/*SSID read*/, 0, (uint32) &uble_demo_values[1]},

    {0x2803/*Characteristic*/, UBLE_GATT_CHARAC_READ, 0x2b13 /*PASSWD read*/},
    {0x2b13/*PASSWD read*/, 0, (uint32) &uble_demo_values[2]},
};
```

由上图可见,att_table 包含了两项服务,分别是默认服务(0x1800)和自定义配网服务(0x1910)。自定义配网服务是我们需要关心的，其包含了四个特征，具体如下：

- Write特征（0x2b11）：用于手机端向设备进行写入数据；
- Notify特征（0x2b10）：用于设备向手机端进行信息通知；
- Read特征 1（0x2b12）：用于手机端向设备进行读取数据，这里是读取设备的SSID；
- Read特征 2（0x2b13）：用于手机端向设备进行读取数据，这里是读取设备的Password；

BLE 连接配网就是通过手机端对设备的写入实现将联网信息传输给设备的，设备接收数据的 callback 就由 att_table 中的 uble_demo_values 指定，ble_demo.c 对 uble_demo_values 的定义如下图所示。

```
static const struct uble_value_entry uble_demo_values[] = {
    {.type = UBLE_VALUE_TYPE_HDL, .value = uble_test_hdlval, .size = 0, .bitoff = 0, .maskbit = 0,},
    {.type = UBLE_VALUE_TYPE_STRING, .value = sys_cfgs.ssid, .size = 21, .bitoff = 0, .maskbit = 0,},
    {.type = UBLE_VALUE_TYPE_STRING, .value = sys_cfgs.passwd, .size = 21, .bitoff = 0, .maskbit = 0,},
};
```

由上图可见，uble_demo_values 指明了不同特征属性值的处理，协议代码便可以直接读写，不需要再写额外代码。UBLE_VALUE_TYPE 已经枚举了常用的数据类型，如下图所示，如果没有合适的数据类型还可以采用 UBLE_VALUE_TYPE_HDL 来定义通用的类型，其 value 可以是回调函数用于处理特殊的数据。

```
enum UBLE_VALUE_TYPE {
    UBLE_VALUE_TYPE_UINT8,
    UBLE_VALUE_TYPE_UINT16,
    UBLE_VALUE_TYPE_UINT32,
    UBLE_VALUE_TYPE_UINT8_BIT,
    UBLE_VALUE_TYPE_UINT16_BIT,
    UBLE_VALUE_TYPE_UINT32_BIT,
    UBLE_VALUE_TYPE_BYTES,
    UBLE_VALUE_TYPE_STRING,
    UBLE_VALUE_TYPE_HDL,
};
```

在使用 TXW8xx SDK 进行 BLE 连接配网开发时，需要修改的也就是上面两个表。下面简单介绍一下两个表的结构。

uble_demo_att_table 表结构如下：

- att_type:　　UUID

- properties： 特征
- att_value： type或者character value关联的value值

```
struct uble_gatt_data {
    uint16 att_type, properties;
    uint32 att_value;
};
```

uble_demo_values 表结构如下:

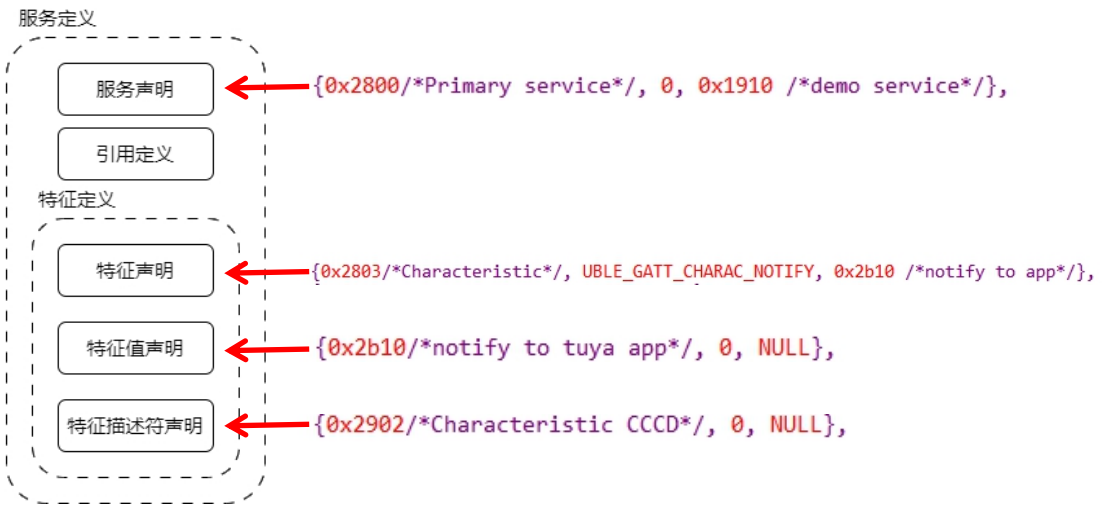
- type： 数据类型
- size： 数据大小
- bitoff： 偏移量
- maskbit： 位掩码
- *value： 用于关联任意参数

```
struct uble_value_entry {
    uint8 type, size, bitoff, maskbit;
    void *value;
};
```

根据 BLE 协议可以找到服务与特征的 UUID 定义，如下图所示。

Assigned Number	Name	Uniform Type Identifier	Specification
0x2800	Primary Service	org.bluetooth.attribute.gatt.primary_service_declaration	GSS
0x2801	Secondary Service	org.bluetooth.attribute.gatt.secondary_service_declaration	GSS
0x2802	Include	org.bluetooth.attribute.gatt.include_declaration	GSS
0x2803	Characteristic Declaration	org.bluetooth.attribute.gatt.characteristic_declaration	GSS

了解了两个表的结构之后，就可以自定义 GATT 服务了。创建的自定义 GATT 服务与 att_table 对应如下图所示（以 Notify 为例）。



一般的，一个配网服务只完成联网信息交互的话，只需要一个 Write 的特征，手机端便可将联网信息写入设备，设备读取和解析数据后进而连上网。

TXW8xx SDK 提供的 BLE 连接配网 demo，实现了联网信息的写入和设备 SSID 和 Password 的读取，下文会进行操作的说明和演示。

3.4. BLE 连接配网初始化

使用 BLE 连接配网之前，需要对模块进行初始化，可直接使用 ble_demo.c 提供的示例代码，如下图所示：

```
int32 ble_demo_mode3_init(struct bt_ops *bt_ops)
{
    uble_init(bt_ops, uble_demo_att_table, ARRAY_SIZE(uble_demo_att_table));

    uint8 adv_data[] = {    0x02, 0x01, 0x06,

                            0x03, 0x02, 0x01, 0xA2,

                            0x14, 0x16, 0x01, 0xA2, 0x01, 0x6B, 0x65, 0x79, 0x79, 0x66, \
                            0x67, 0x35, 0x79, 0x33, 0x34, 0x79, 0x71, 0x78, 0x71, 0x67, 0x64};

    uint8 scan_resp[] = {   0x04, 0x09, 0x53, 0x53, 0x53,

                            0x19, 0xFF, 0xD0, 0x07, 0x01, 0x03, 0x00, 0x00, 0x0C, 0x00, \
                            0x88, 0xD1, 0xC4, 0x89, 0x2B, 0x56, 0x7D, 0xE5, 0x65, 0xAC, \
                            0xA1, 0x3F, 0x09, 0x1C, 0x43, 0x92};

    ble_ll_set_advdata(bt_ops, adv_data, sizeof(adv_data));
    ble_ll_set_scan_rsp(bt_ops, scan_resp, sizeof(scan_resp));

    return ble_ll_open(bt_ops, 2, 38);
} « end ble_demo_mode3_init »
```

Length UUID Data
设备名称

uble_init 对 BLE 协议模块进行初始化，需要使用到定义的 att table。

BLE 连接配网需要设置广播数据和扫描响应数据，具体操作如下。

1) 设置设备的BLE广播数据，该广播数据用于手机发现设备。

广播数据内容为 AdvData 部分，如下图所示：

Preamble (1 octet)	Access Address (4 octet)	Adv Header (2 octet)	AdvA (6 octets)	AdvData (0-31 octets)	CRC (3 octets)

设备将以 ADV_IND 类型发送广播数据。

设置广播数据的宏为：

ble_ll_set_advdata(ops, adv_data, len)

参数说明：

- ops：　　btops
- adv_data：广播数据
- len：　　广播数据的长度

返回值：

- 返回 0： 设置成功
- 非 0：　ERROR

2)　设置设备的BLE扫描响应数据，该数据用于回应手机扫描请求，并携带设备信息。

响应数据的内容也为 AdvData 部分，如上图所示。

设置扫描响应数据的宏为：

```
ble_ll_set_scan_rsp(ops, scan_resp, len)
```

参数说明：

- ops：　　btops
- scan_data：　扫描响应数据
- len：　　扫描响应数据的长度

返回值：

- 返回 0： 设置成功
- 非 0：　ERROR

初始化完成后，使用宏 ble_ll_open 打开 BLE 连接配网模式。ble_ll_open 参数说明如下：

```
ble_ll_open(ops, type, chan)
```

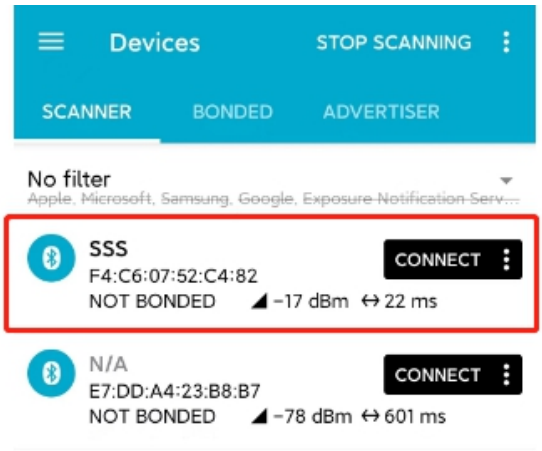
参数说明：

- ops：　bt_ops
- type：　2 - BLE连接配网模式
- chan：　该参数固定输入 38
- 返回 0： 开启成功
- 非 0：　ERROR

3.5. 接收 App 发送数据

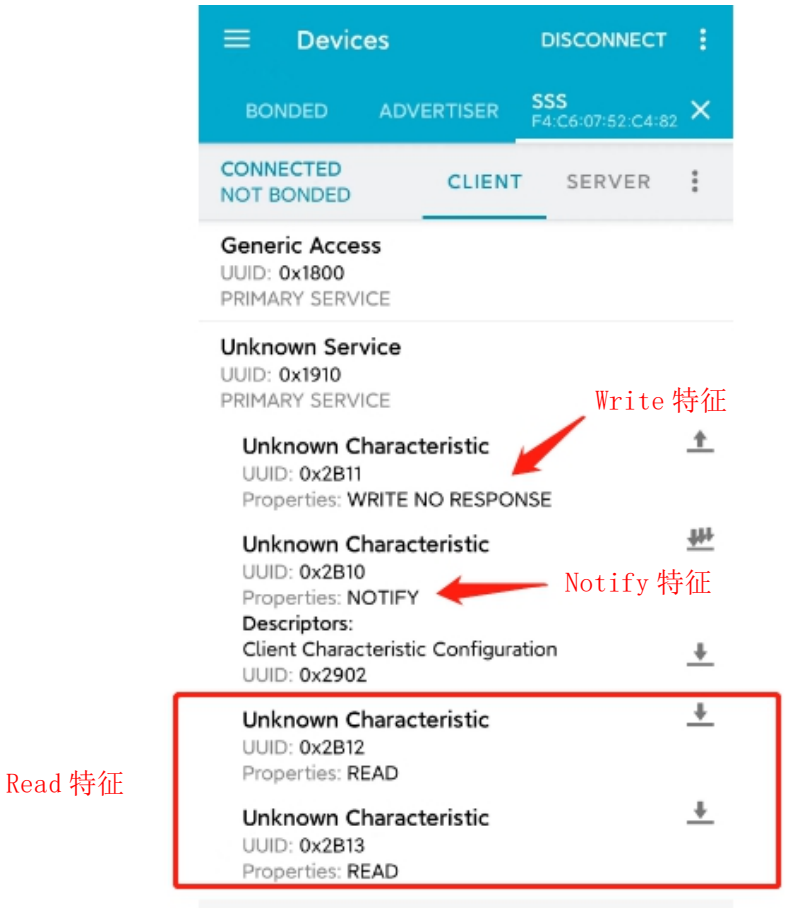
TXW8xx BLE 连接配网可使用 nRF Connect 进行测试，操作流程如下。

使用 nRF Connect 扫描设备。



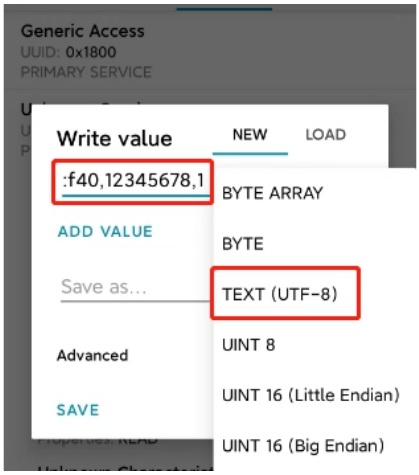
手机端点击 CONNECT 后，设备会收到连接请求。

建立连接后，手机端可以查看设备的所有 GATT 服务，即 uble_demo_att_table 表定义的服务。



为满足蓝牙 BLE 配网需求，TXW8xx BLE 连接配网实现了对设备进行 Write 操作，方便将联网信息发送给设备。

手机端使用 nRF Connect 配合 demo 代码进行测试，对设备进行写入操作如下：



发送的联网信息需遵循以下规则：

:SSID,PASSWD,0/1

注：全英字符，0 表示不加密，1 表示加密。字符小于 21byte

手机端发送后，此时设备将接收并进行处理，最终调用 uble_test_hdlval 进行数据解析。

```
static int32 uble_test_hdlval(const struct uble_value_entry *entry, uint8 read, uint8 *buff, int32 size)
{
    char *ptr1, *ptr2;
    os_printf("buff=%s size = %d\r\n", buff, size);
    if (read) { /*get value*/

        //buff: used to store response data.
        //size: buff's size.

        //return response len.
        return 0;

    } else { /*set value*/
            if (buff[0] == ':') {

            ptr1 = os_strchr(buff, ',');
            if (ptr1 == NULL) { return UBLE_ATT_NOT_SUPPORT; }
            *ptr1++ = 0;
            os_strcpy(sys_cfgs.ssid, buff + 1);
            os_printf("SET ssid:%s\r\n", buff + 1);

            ptr2 = os_strchr(ptr1, ',');
            if (ptr2 == NULL) { return UBLE_ATT_NOT_SUPPORT; }
            *ptr2++ = 0;
            os_strcpy(sys_cfgs.passwd, ptr1);
            os_printf("SET passwd:%s\r\n", ptr1);

            sys_cfgs.key_mgmt = ptr2[0] == '1' ? WPA_KEY_MGMT_PSK : WPA_KEY_MGMT_NONE;
            os_printf("SET passwd:%c\r\n", ptr2[0]);
            os_printf("SET keymgmt:%s\r\n", ptr2[0] == '1' ? "WPA-PSK" : "NONE");

            syscfg_dump();

            ble_network_configured();
            SYSEVT_NEW_BLE_EVT(SYSEVT_BLE_NETWORK_CONFIGURED, 0);
        } « end if buff[0]==':' »
        return 0;
    } « end else »
} « end uble_test_hdlval »
```

设置 SSID

设置 Password

设置 Keymgmt

设备联网

回抛 BLE 事件

最后关闭蓝牙 BLE 模式，进入 WIFI 模式进行联网。

3.6. 向 App 发送数据

设备向 App 发送数据有 2 种形式：

- **Read**：App发送读取指令，设备反馈数据；
- **Notify**：设备主动发送数据通知App；

TXW8xx BLE 连接配网提供了设备 Notify 的 API，以及对设备已保存的联网信息的 Read 操作，包括对设备的 SSID 和 Password 的读取，详细说明如下。

1) Notify

设备与手机端建立 BLE 连接后，设备可以主动发送数据通知 App。att_table 有关 Notify 特征定义如下。

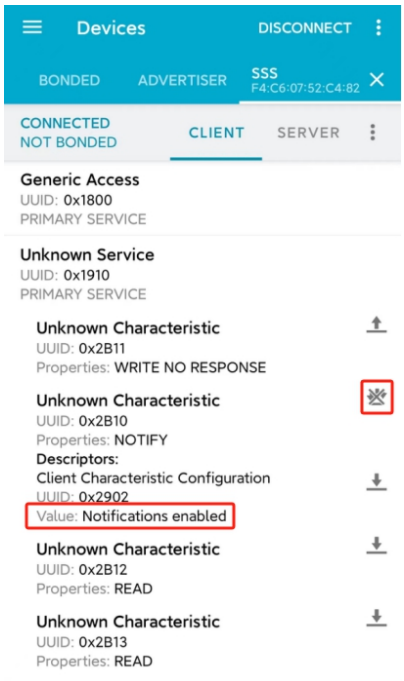
```
{0x2803/*Characteristic*/, UBLE_GATT_CHARAC_NOTIFY, 0x2b10 /*notify to app*/},
{0x2b10/*notify to tuya app*/, 0, NULL},
{0x2902/*Characteristic CCCD*/, 0, NULL},
```

这里需要注意 Notify 特征需要声明 CCCD 描述符，否则 App 监听失败。

TXW8xx BLE 连接配网开放 uble_gatt_notify 接口可以实现通知功能，其中参数 data 即为通知的内容。

```
extern int32 uble_gatt_notify(uint16 att_hdl, uint8 *data, int32 len);
```

若使用 nRF Connect 进行调试，当点击 Notify 时，手机端会开始监听数据，如下图所示。



2) Read

设备与手机端建立 BLE 连接后，当手机端点击 Read 请求时，设备将向手机端发送数据，发送的数据与 att_table 表有关，设备会根据手机端发送请求的 UUID 找到 att_table 表中对应的特征，该特征对应的 att_value 即为返回的数据，如下图所示。

- 0x2b12 返回设备SSID，大小为 21byte；
- 0x2b13 返回设备Password，大小为 21byte；

```
{0x2803/*Characteristic*/, UBLE_GATT_CHARAC_READ, 0x2b12 /*SSID read*/},
{0x2b12/*SSID read*/, 0, (uint32) &uble_demo_values[1]},

{0x2803/*Characteristic*/, UBLE_GATT_CHARAC_READ, 0x2b13 /*PASSWD read*/},
{0x2b13/*PASSWD read*/, 0, (uint32) &uble_demo_values[2]},
```

```
static const struct uble_value_entry uble_demo_values[] = {
    {.type = UBLE_VALUE_TYPE_HDL, .value = uble_test_hdlval, .size = 0, .bitoff = 0, .maskbit = 0,},
    {.type = UBLE_VALUE_TYPE_STRING, .value = sys_cfgs.ssid, .size = 21, .bitoff = 0, .maskbit = 0,},
    {.type = UBLE_VALUE_TYPE_STRING, .value = sys_cfgs.passwd, .size = 21, .bitoff = 0, .maskbit = 0,},
};
```

若使用 nRF Connect 进行调试，nRF Connect 同时会将 ASCLL 码转成对应的字符。如下图所示。