Kw36多连接（服务端）

两块以上KW36

1. 首先导出temp\_sense的freertos例程

2 修改app\_preinclude.h，作如下修改

**#define** gAppMaxConnections\_c 8

**#define** gTmrStackTimers\_c (6 + gAppMaxConnections\_c)

不使用低功耗，则

**#define** cPWR\_UsePowerDownMode 0

1. 修改temperature\_interface.h

修改结构体为

**typedef struct tmsConfig\_tag**

{

uint16\_t serviceHandle;

int16\_t temperature;

bool\_t\* aValidSubscriberList; //表示哪些注册设备是可使用的

uint8\_t validSubscriberListSize; //可以连接设备的数量

} tmsConfig\_t;

将以下声明函数修改，添加参数

bleResult\_t **Tms\_Subscribe**(tmsConfig\_t \*pServiceConfig, deviceId\_t deviceId);

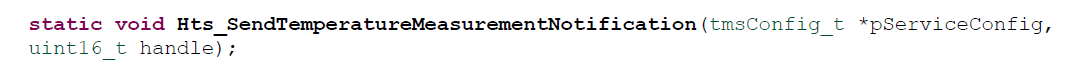
bleResult\_t **Tms\_Unsubscribe**(tmsConfig\_t \*pServiceConfig, deviceId\_t deviceId);

bleResult\_t **Tms\_RecordTemperatureMeasurement** (tmsConfig\_t \*pServiceConfig);

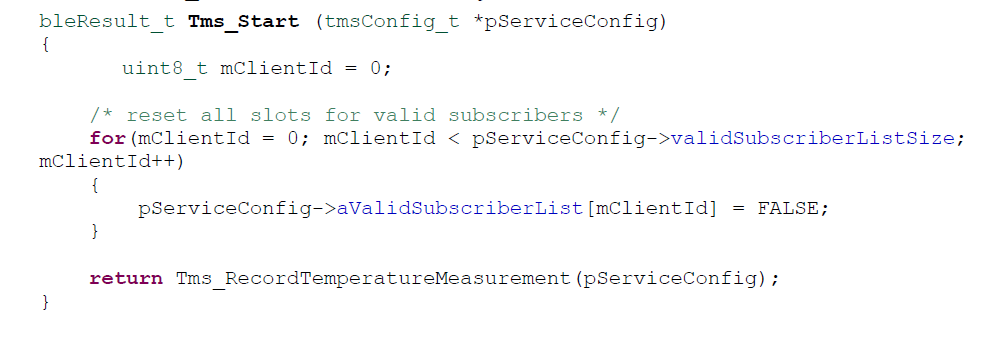
1. 修改temperature\_service.c

将static deviceId\_t mTms\_SubscribedClientId; 注释

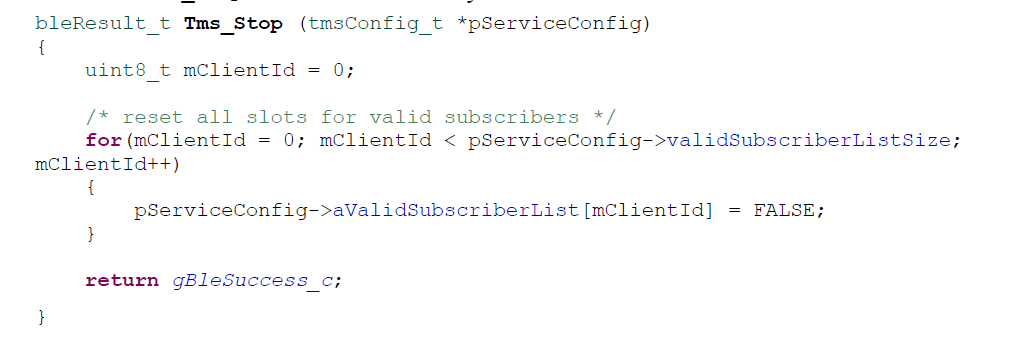
修改下面函数参数定义

****

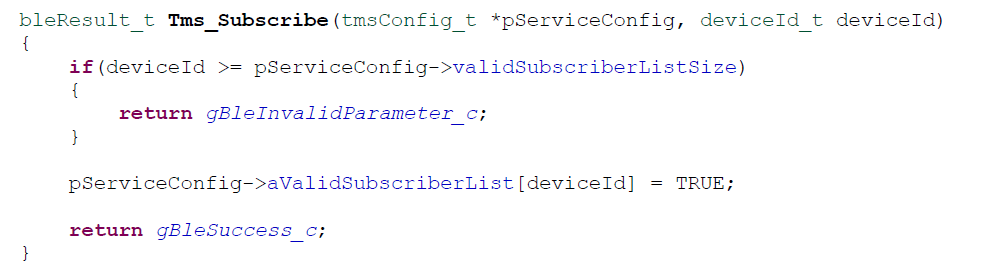
4.1 修改函数，**Tms\_Start**

****

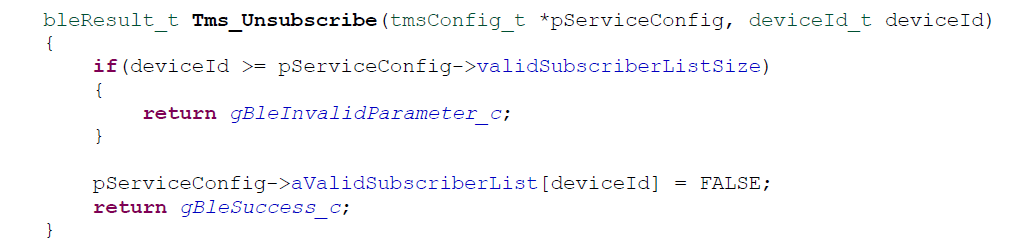
4.2 修改函数**Tms\_Stop**



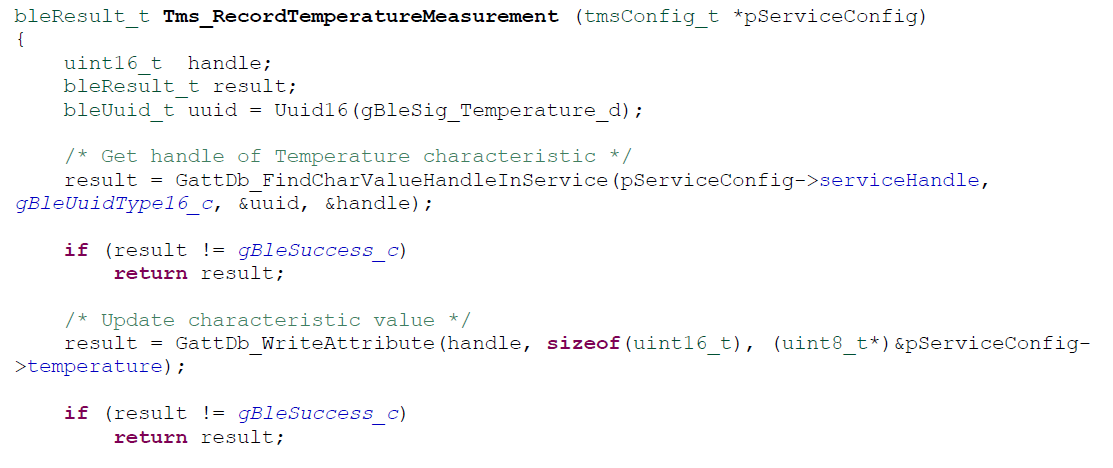
4.3 修改函数**Tms\_Subscribe**

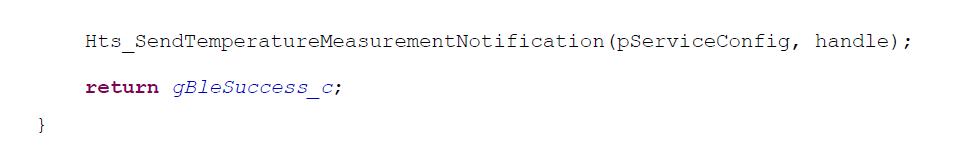


4.4 修改函数**Tms\_Unsubscribe**

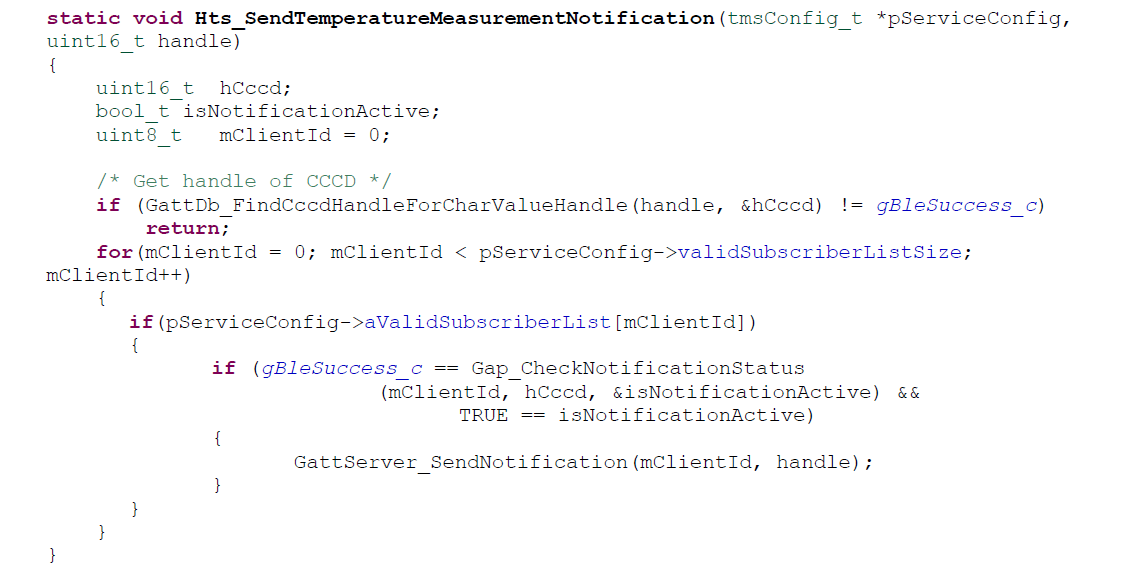


4.5 修改函数**Tms\_RecordTemperatureMeasurement**





4.6 修改**Hts\_SendTemperatureMeasurementNotification**



1. 修改temperature\_sensor.c

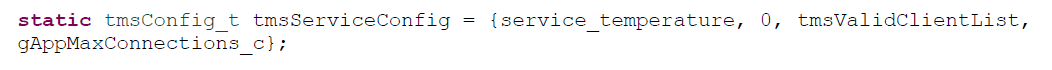
5.1 找到mPeerDeviceId 修改为

**static** deviceId\_t mPeerDeviceId[gAppMaxConnections\_c] ={gInvalidDeviceId\_c};

5.2 定义uint8\_t mActiveConnections = 0;

**static** bool\_t tmsValidClientList[gAppMaxConnections\_c] = {FALSE};

5.3 修改tmsServiceConfig ，为

****5.4 找到DisconnectTimerCallback 函数注释掉

static void DisconnectTimerCallback(void\* );

5.5 找到BleApp\_Config，修改

tmsServiceConfig.temperature = 100 \* BOARD\_GetTemperature();

5.6 找到BleApp\_ConnectionCallback，

5.6.1 在case gConnEvtConnected\_c，修改

1) mPeerDeviceId[mActiveConnections] = peerDeviceId;

2) Tms\_Subscribe(&tmsServiceConfig, peerDeviceId);

在 Tms\_Subscribe 函数后mActiveConnections++;

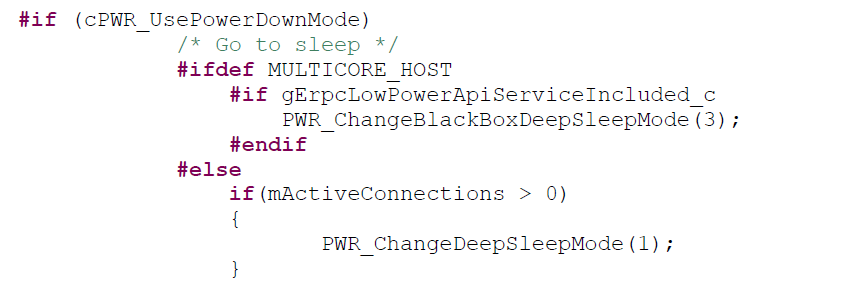
5.6.2 在case gConnEvtDisconnected\_c,

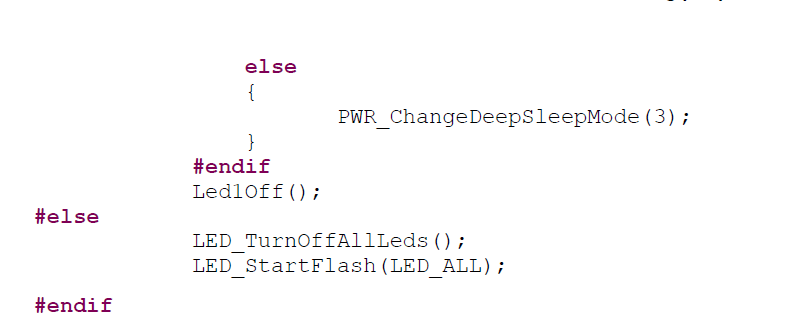
1) mPeerDeviceId[peerDeviceId] = gInvalidDeviceId\_c;

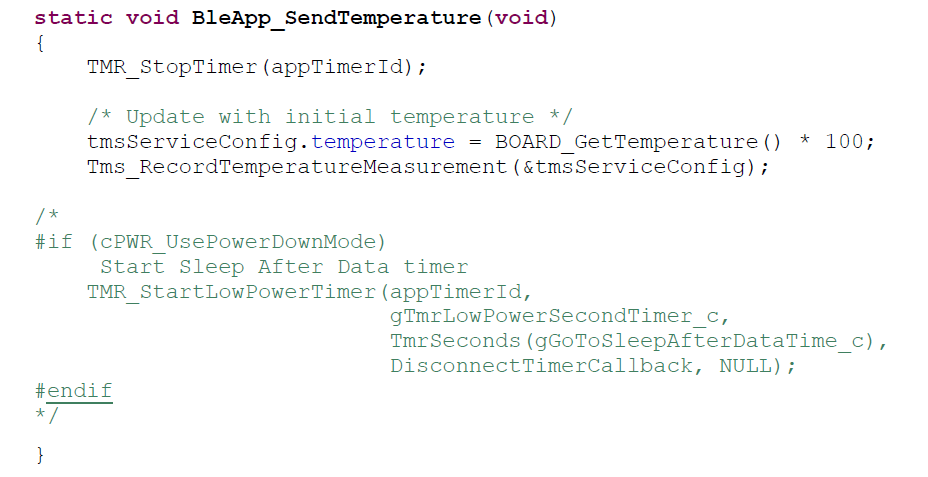
2) Tms\_Unsubscribe(&tmsServiceConfig, peerDeviceId);

在 Tms\_Unsubscribe函数后mActiveConnections--;

找到cPWR\_UsePowerDownMode，修改为



* 1. 
  2. 将DisconnectTimerCallback 函数注释掉
  3. 修改BleApp\_SendTemperature

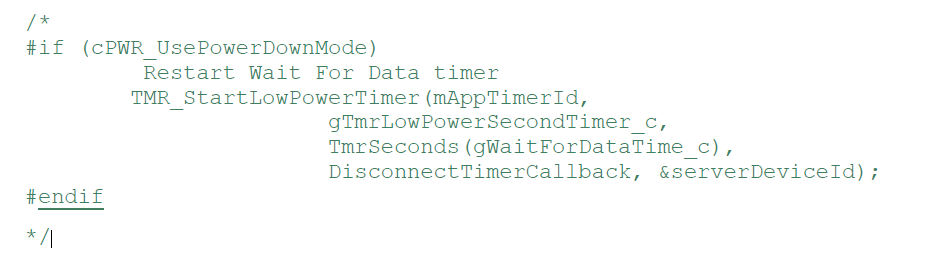


5.9 导出temp\_coll工程

5.9.1 可以将低功耗关闭，如上个工程一样

5.9.2 找到BleApp\_GattNotificationCallback

注释掉



6 分别将程序烧入两块板子，打开两个串口，一个板子按sw2开启广播，另一个按下sw2开启扫描，一会儿两个板子就可以连接成功，串口打印出消息。大约等5s之后获得第一条数据。随后就不再有输出了，客户端也不再获得服务端的数据。服务端可以再按下sw2，可以再次开启广播，来连接另外一个开启扫描的板子。

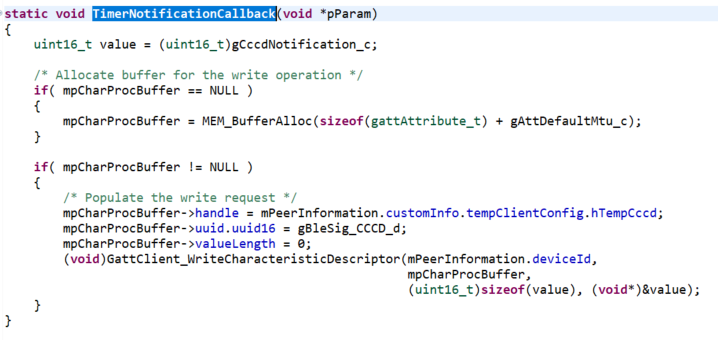
若开启低功耗，则按下llwu后唤醒两个设备，两个设备自动一个开启广播，一个开启扫描，并自动连接，客户端约5s获得一次消息。然后两个设备进入休眠，可以通过按下llwu再次唤醒两个设备，这样服务端会再发消息，并且不再需要等待5s。

7 在不进入低功耗情况下，服务端只能发送一次数据，可以做以下修改使得客户端来不断获得服务器数据。因为这两个例程，简单来说就是，客户端发送notification，服务端接收到这个请求，发送数据给客户端，那么只要我们定时发送这个notification，就可以不断获得数据。

7.1 在temp\_coll工程里，找到temperature\_collector.c文件，定义定时器ID

**static** tmrTimerID\_t mNotificationId;

7.2 定义该ID的回调函数



7.3 找到BleApp\_StateMachineHandler，找到case为*mAppRunning\_c*

在mpCharProcBuffer = NULL;后面开启定时器 TMR\_StartLowPowerTimer(mNotificationId, gTmrLowPowerIntervalMillisTimer\_c,TmrSeconds(1),TimerNotificationCallback,NULL);

效果如图

