蓝牙台灯

基于 CC254xSmartRF 开发板的蓝牙台灯制作

刘雨

2013-09-12

[在此处键入文档摘要。摘要通常为文档内容的简短概括。在此处键入文档摘要。摘要通常为文档内容的简短概括。]

学习任何一种知识都必须将其用到生活中去，否则就不要去学他。

今天就展示一下，蓝牙台灯的制作全过程，基于我们的 CC254x-SmartRF 开发板

本文简要目录

正文：

1、为什么要做蓝牙台灯？

我想只要学过 zigbee 或者低功耗蓝牙 ble 的人都有一种冲动，想把家里的一切只要带电的都想改成无线的，然后用手机控制。

但是大多数人每天都要忙于工作和学习，各种苦命，很少有时间，像老外一样 DIY 各种各样好玩的东西。

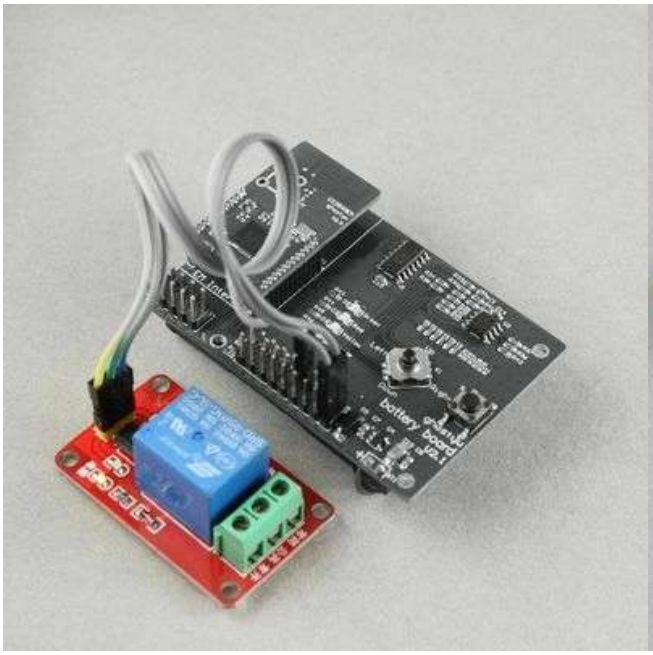
今天我们就甩掉所有事情，把想法变成行动！

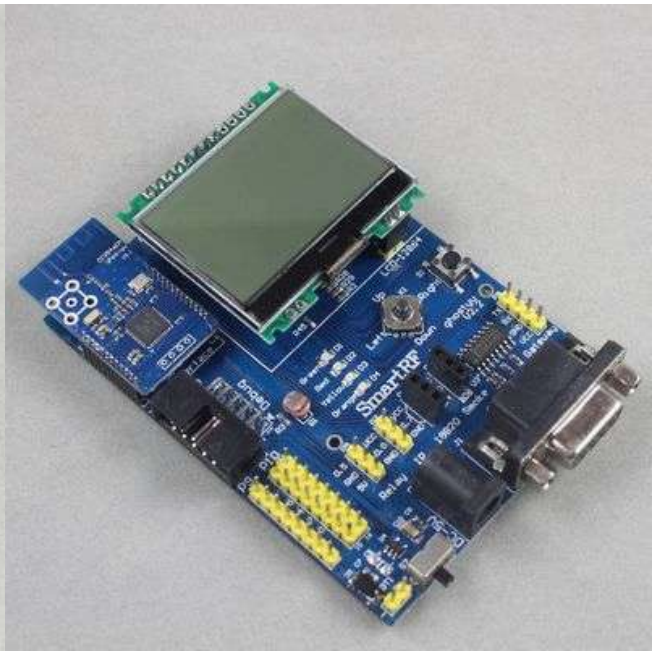
通过这个蓝牙台灯，希望不仅仅教会大家如何修改现有的 TI BLE 协议栈例程，更是希望能让读者重拾往日的激情与冲动！

2、蓝牙台灯原型与材料准备

台灯原型是我随便在网上找的一个，就是下面这个样子

除了台灯，我们还需要一个继电器，一个 BLE-SmartRF 开发板，以及一些杜邦线。

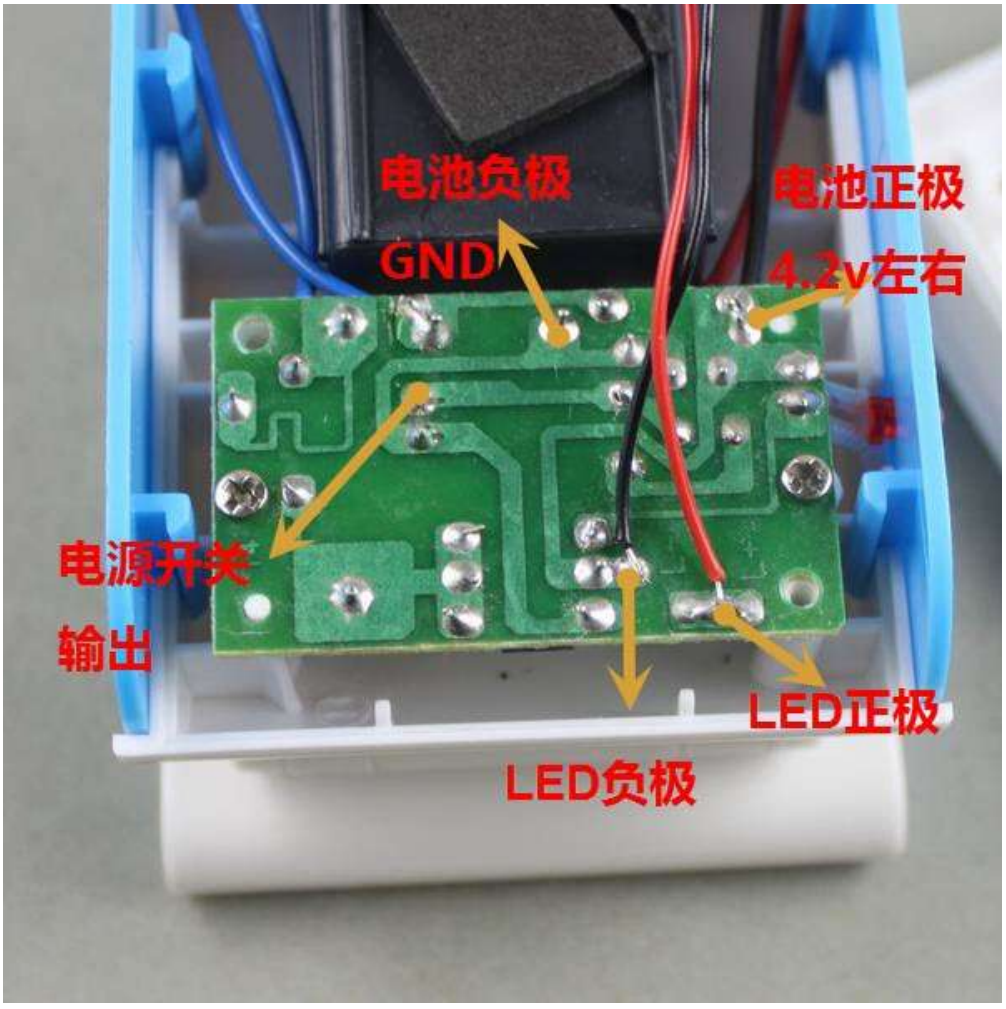




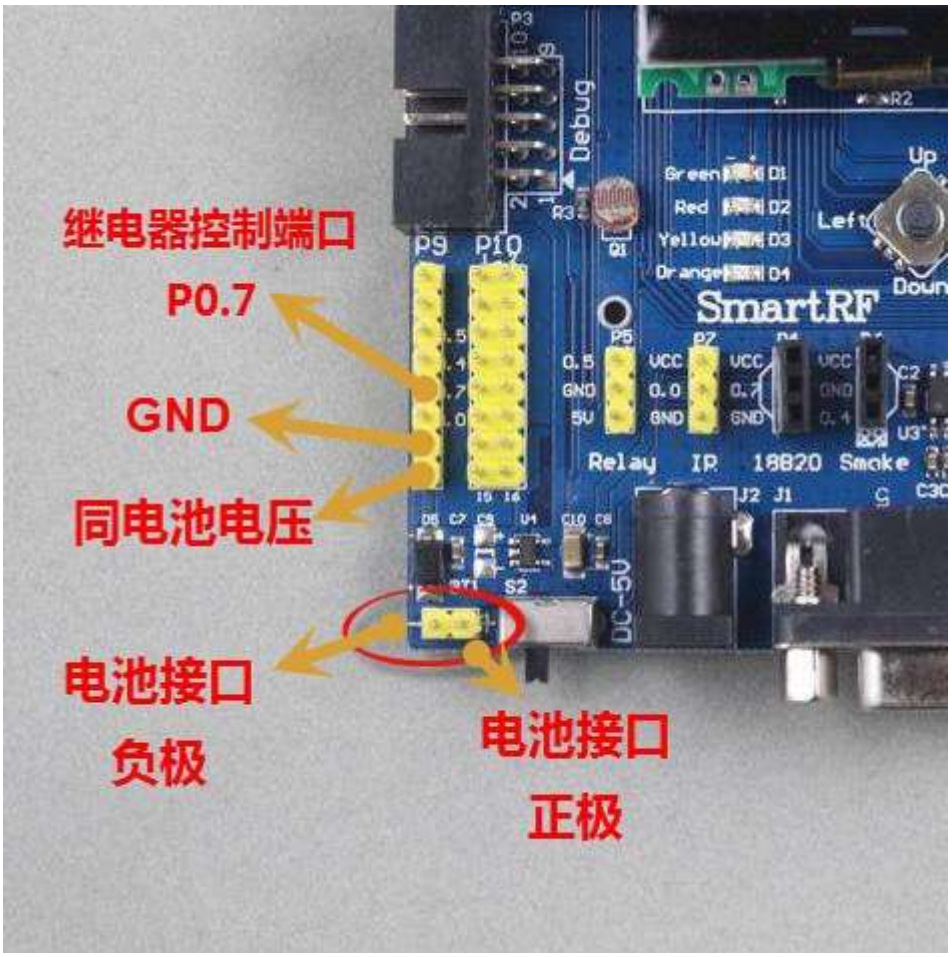
3、拆解分析

拆开后，结构非常简单，一块电池，一个简单的控制电路。

使用万用表简单的测一下，得出如下结论

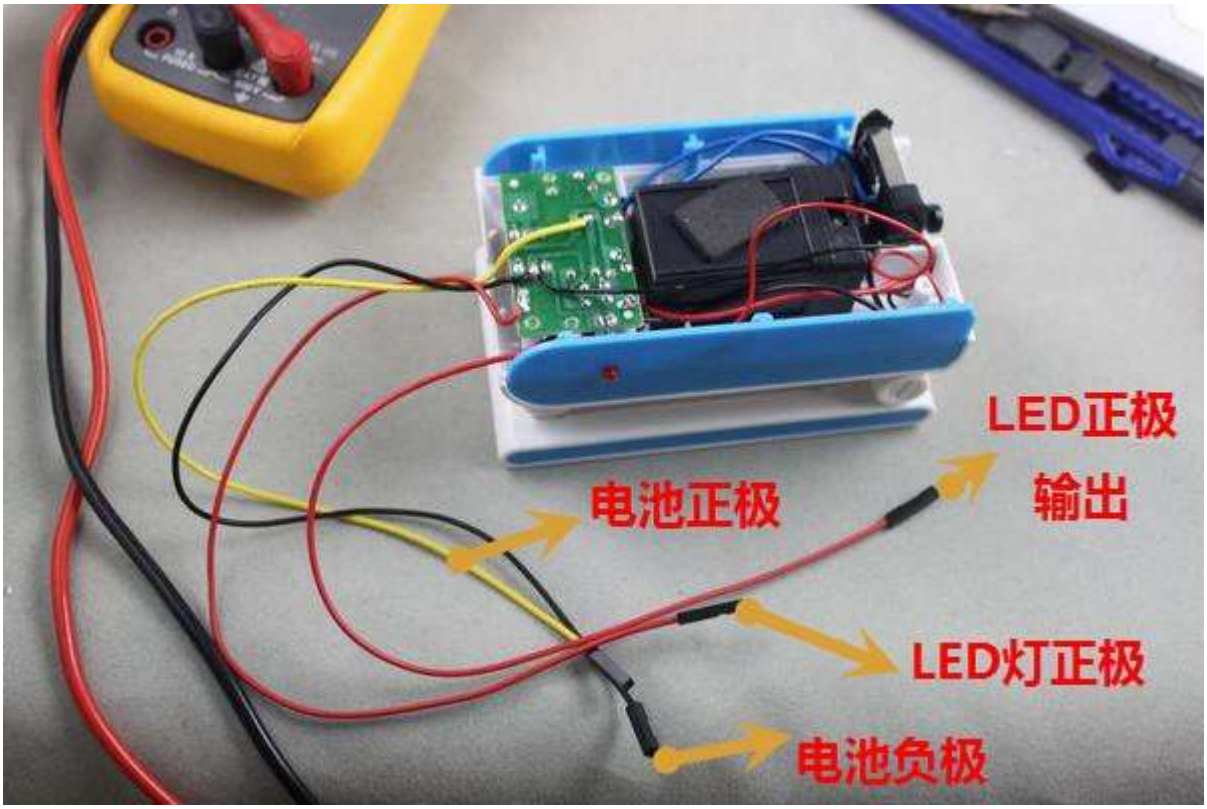


这样就好办了，将电源开关输出接到 SmartRF 开发板的电池接口上，将 LED 正极从 PCB 板上断开，分开后的两根线，分别连接继电器，让继电器控制通断。然后从开发板上引出继电器工作时需要的 VCC 和 GND，然后将继电器的控制端接到开发板上的 P0.7 GPIO 端口，需要注意的事，SmartRF 开发板上有一颗 3.3V 的 LDO 芯片，台灯电池接到 SmartRF 开发板上后，通过 LDO 降压芯片（RT9013）稳压到 3.3V，为 CC254x 提供稳定的工作电压，请勿将电池直接给芯片供电，CC254x 能够容忍的最大电压不能超过 4V。另外，使用的继电器为 5V 驱动，电池电压 4.2v 左右，也可以驱动，因此，台灯电压经过 SmartRF 开发板上的 D5 二极管（反向保护肖特基二极管）后直接给继电器提供工作电压。如下图所示



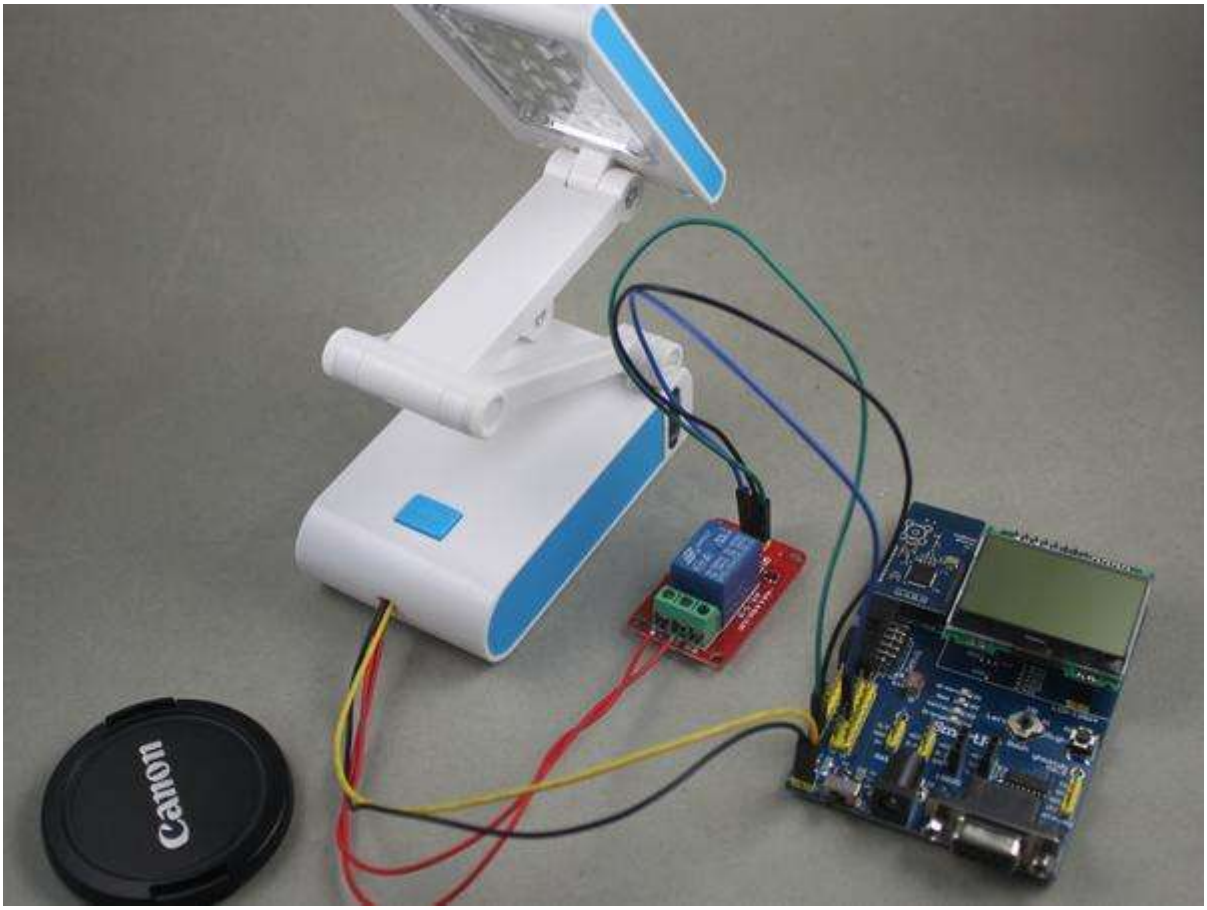
4、改装

我们按照 3 中的分析结果，进行改装，需要大家使用烙铁引出一些电源、信号线。如下图



线焊好之后 ,需要将杜邦那个线接到台灯外壳的外面去 ,需要将外壳刻出一点缝隙，让杜邦线穿过。然后重新将外壳安装好。

根据 3 中分析的结果将台灯引出来的线和外面的继电器，SmartRF 开发板连接好，连接继电器时注意，我们使用的继电器低电平驱动，也就是说，当有低电平触发时，继电器输出端口的【常开】端将和【公共端】闭合，而【常闭】端将和【公共端】断开。台灯的使用习惯，不用是断开电源，使用的时候才会打开电源，因此将台灯的 PCB 板上的【LED 正极输出】接到继电器的【常开】端，【LED 灯正极】接到继电器的【公共端】，这样，当继电器有低电平信号触发时，继电器工作，台灯的 LED 将通过继电器导通，从而被点亮。连接好的样子如下图。



5、编写基于 SmartRF(CC2540 或者 CC2541)开发板的蓝牙控制程序

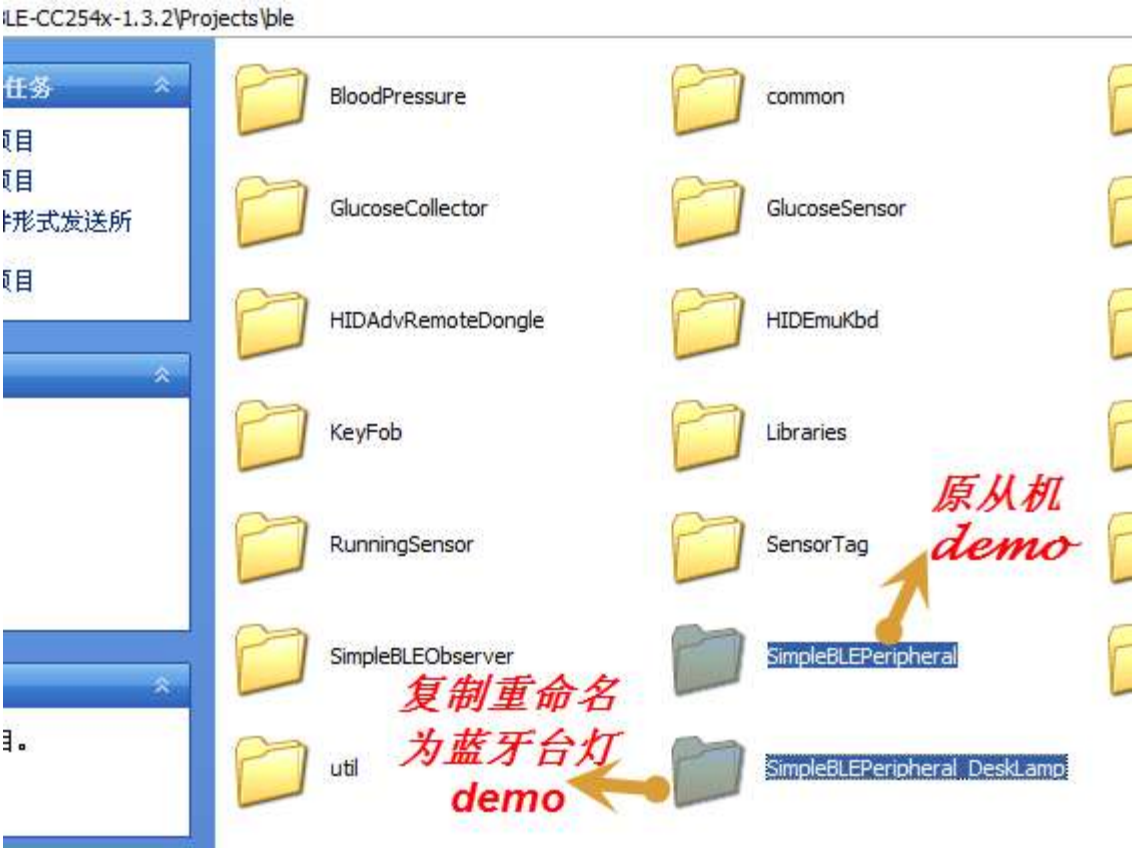
该节是本次 DIY 的重点，我们将修改 TI 的 SimpleBLEPeripheral 从机程序，添加继电器的控制，从而控制继电器的通断，SimpleBLEPeripheral 从机程序的蓝牙功能已经实现，如果有需要，做一些改动即可，读者们可以注意下，我们开发 CC2540

程序时，通常会有 BLE 协议栈原有例程为样板，然后修改成我们所需要的，这是协议栈开发板的最佳道路。我们平时的 BLE 协议栈学习，也是精良搞懂每一个例子。这样在需求来临时，可以迅速的选择 BLE 现有例程，然后修改成自己想要的。、

（1）新建 SimpleBLEPeripheral_DeskLamp 工程

进入 BLE-CC254x-1.3.2 协议栈的工程目录。我的协议栈安装目录为：E:\TI\BLE-CC254x-1.3.2\

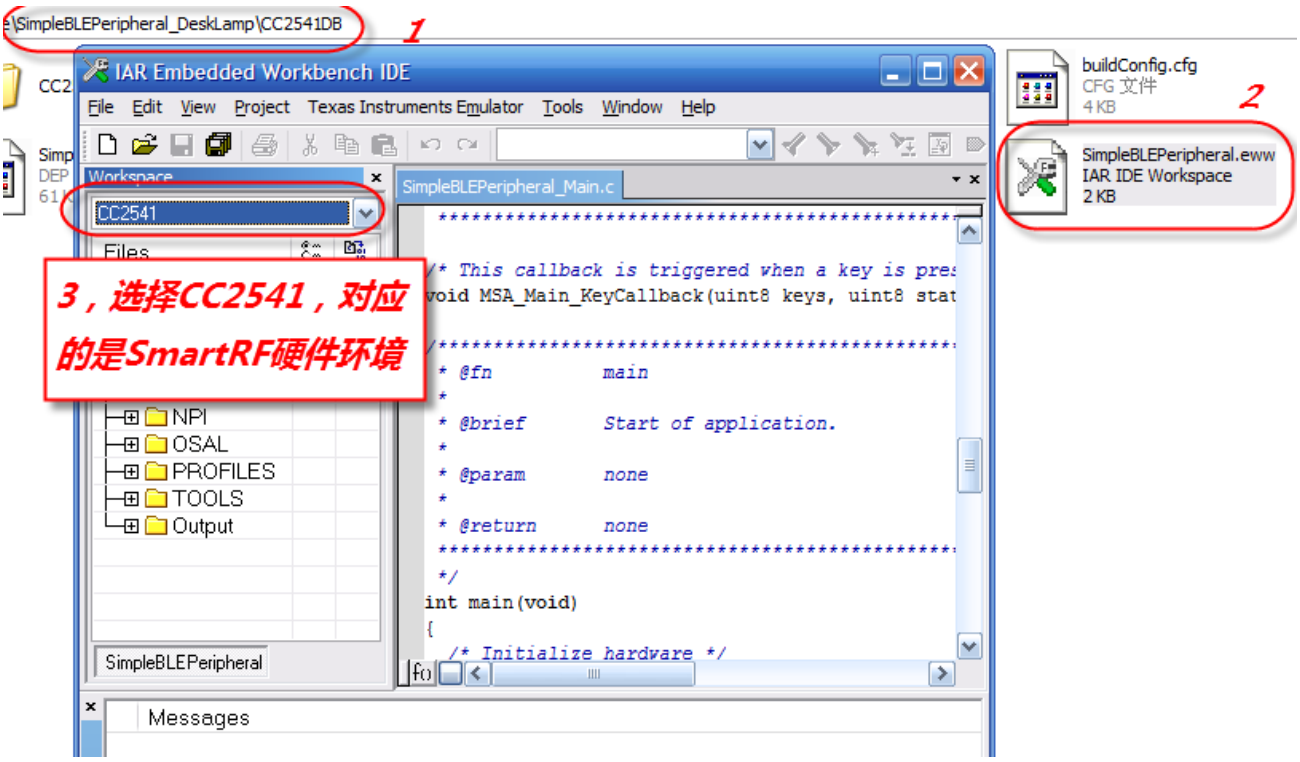
工程目录为：E:\TI\BLE-CC254x-1.3.2\Projects\ble，这里面是所有协议栈 demo 所在的位置，所有例程都必须在这个目录里编译，否则会因为找不到组件会出现大量错误，所以大家要写自己的 demo，就在当前目录复制粘贴重命名为：SimpleBLEPeripheral_DeskLamp。



（2）打开 SimpleBLEPeripheral_DeskLamp IAR 工程

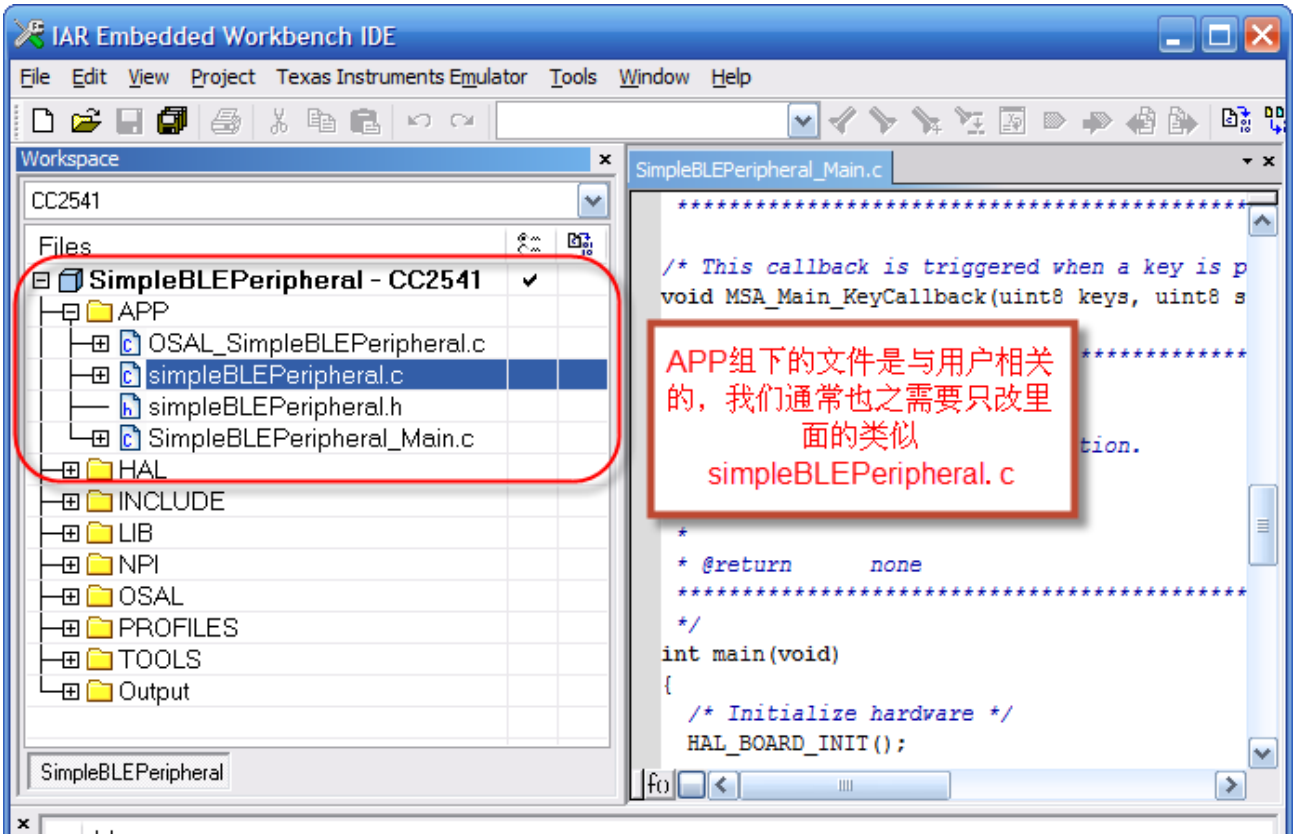
我使用的是 CC2541EM，所以需要进入 SimpleBLEPeripheral_DeskLamp\CC2541DB，打开 SimpleBLEPeripheral.eww 工程

注意选择 Configuration，跟自己所使用的硬件换进对应即可，我所使用的是 SmartRF 硬件，所以选择 CC254x.



（3）修改原有程序

在 APP 组下面有几个源文件，其中需要我们经常修改的是类似 simpleBLEPeripheral.c 这样的文件，而 OSAL_xx.c 或者 xxx_Main.c 通常不需要修改。



添加继电器初始化代码

我们把继电器初始化代码放到了任务函数里（simpleBLEPeripheral.c），其实，通常的做法，是放到任务函数对应的 init 里，之所以放到这里，只是简单起见，并且能够完全控制该 GPIO，不用其他地方还在使用。

GPIO 初始化比较简单，设置 PxDIR 具体端口的输入输出方向，置 1 是输出，清 0 是输入；然后设置 PxSEL 端口的功能，置 1 是外设，清 0 是 GPIO。具体如下图。

```
uint16 SimpleBLEPeripheral_ProcessEvent( uint8 task_id, uint16
{

  VOID task_id; // OSAL required parameter that isn't used in this function

  if ( events & SYS_EVENT_MSG )
  {
    uint8 *pMsg;

    if ( (pMsg = osal_msg_receive( simpleBLEPeripheral_TaskID )) != NULL
    {
      simpleBLEPeripheral_ProcessOSALMsg( (osal_event_hdr_t *)pMsg );

      // Release the OSAL message
      VOID osal_msg_deallocate( pMsg );
    }

    // return unprocessed events
    return (events ^ SYS_EVENT_MSG);
  }

  if ( events & SBP_START_DEVICE_EVT )
  {
    //初始化台灯控制所使用的gpio引脚，为什么放到这里初始化？
    //由于我们使用的GPIO可能会被系统占用，所以在系统运行后，强行将该GPIO
    //这里使用P0_7，继电器低电平触发，因此初始化 输出为高
    P0_7 = 1;  //防止继电器跳变，初始化为高
    P0DIR |= BV(7); //设置为输出
    P0SEL &=~BV(7); //设置该脚为普通GPIO

    // Start the Device
    VOID GAPRole_StartDevice( &simpleBLEPeripheral_PeripheralCBs );
```

BLE 通信概述

在讲如何接收蓝牙数据之前，我们看一下是谁发送的这个数据，在 BLE 里面，一个从机设备有多个 Services，每个 Service 下面又会有多个 Characteristics，我们 BLE 通信时，其实是通过每个具体的 Characteristic。可以把 Characteristic 理解为具体的端口，通过具体的端口将数据发送出去即可。

那怎样区分不同的 Services 和 Characteristics 呢，就是 UUID，在同一个从机设备中每个 Services 和 Characteristics 都会有唯一的 UUID ,在 CC254x 的程序里，UUID 由两个字节表示 ,例如 0xFFF1 ,在而智能机例如 Android 或者 iOS 编程里，

通常是完整的 128 位的 UUID。两者之间有公式可以转换。

主机程序通过 GATT_WriteCharValue 函数向从机发送数据，如下图，SimpleBLECentral 中的写 characteristic 代码。是不是没有看到 uuid？这是因为有 uuid 对应的 handle，在 cc2540 编程是通过 handle 来代替 uuid。

```
// Do a write
attWriteReq_t req;

req.handle = simpleBLECharHdl;
req.len = 1;
req.value[0] = simpleBLECharVal;
req.sig = 0;
req.cmd = 0;
status = GATT_WriteCharValue( simpleBLEConnHandle, &req, simpleBLETaskId );
```

当主机调用上述函数发送数据后，SimpleBLEPeripheral，也就是我们的从机，会通知用户层，有数据到，快来接收。

```
static void simpleProfileChangeCB( uint8 paramID )
{
  uint8 newValue;

  switch( paramID )
  {
    case SIMPLEPROFILE_CHAR1:
      SimpleProfile_GetParameter( SIMPLEPROFILE_CHAR1, &newValue );
      //这里是char1的数据接收，当主机通过writechar想uuid 为fff1发送的数据将在这里接收到
      //我们定义一下简单的控制命令
      //点亮台灯：0x01，熄灭台灯：0x00
      //可使用btool或者lightblue，或者我们提供的ios均可以发送上述命令来控制
      if(newValue==0x00){//熄灭台灯
        //台灯的led电源连接继电器的常开两端，当继电器被低电平触发时，常开将闭合，LED通电，被点亮。
        P0_7=1;
      }else if(newValue==0x01){//点亮台灯
        //台灯的led电源连接继电器的常开两端，此时GPIO输出高电平，继电器将保持常开，LED断电，被熄
        P0_7=0;
      }
      #if (defined HAL_LCD) && (HAL_LCD == TRUE)
        HalLcdWriteStringValue( "Char 1:", (uint16)(newValue), 10,  HAL_LCD_LINE_3 );
      #endif // (defined HAL_LCD) && (HAL_LCD == TRUE)

      break;

    case SIMPLEPROFILE_CHAR3:
```

当接收到数据后，协议栈会调用改函数，让用户接收 ble 数据。注意下该函数名称 xxxCB，表示的是回调函数，回调函数，是在初始化阶段，将自己的函数指针传递给了协议栈，这样协议栈就可以在需要的时候，通过函数指针来调用改函数，这就叫做回调函数。来看一下，这个函数指针是在何时注册到协议栈内的。在 init 函数

初始化的时候通过 SimpleProfile_RegisterAppCBs 函数注册，如下图

```
00257: // Simple GATT Profile Callbacks
00258: static simpleProfileCBs_t simpleBLEPeripheral_SimpleProfileCBs =
00259: {
00260:   simpleProfileChangeCB     // Charactersitic value change callback
00261: };

00417:   // Register callback with SimpleGATTprofile
00418:   VOID SimpleProfile_RegisterAppCBs( &simpleBLEPeripheral_SimpleProfileCBs );
```

解析数据并添加继电器控制命令

我们通过 SimpleProfile_GetParameter 函数接收主机发送的数据，并将数据保存到 newValue 中

我们约定 0x00 表示熄灭，0x01 表示点亮，当然，大家可以添加更多的命令。

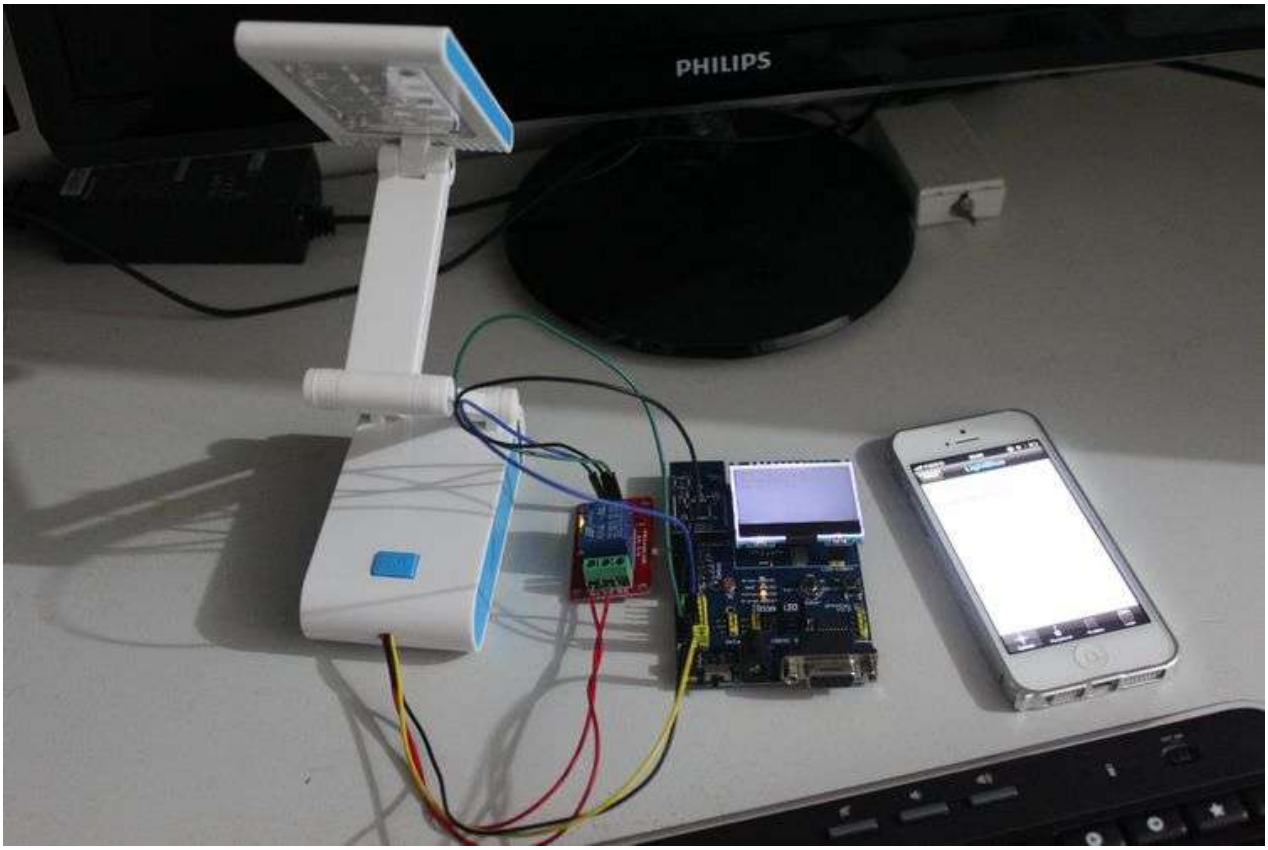
当收到的等于 0x00 时 ,将继电器控制端置 1 ,等于 0x01 时 ,将继电器控制端清零，触发继电器动作

```
case SIMPLEPROFILE_CHAR1:
  SimpleProfile_GetParameter( SIMPLEPROFILE_CHAR1, &newValue );
  //这里是char1的数据接收，当主机通过writechar想uuid 为fff1发送的数据将在这里接收到
  //我们定义一下简单的控制命令
  //点亮台灯：0x01，熄灭台灯：0x00
  //可使用btool或者lightblue，或者我们提供的ios均可以发送上述命令来控制
  if(newValue==0x00){//熄灭台灯
    //台灯的led电源连接继电器的常开两端，当继电器被低电平触发时，常开将闭合，LED通电，被点亮.
    P0_7=1;
  }else if(newValue==0x01){//点亮台灯
    //台灯的led电源连接继电器的常开两端，此时GPIO输出高电平，继电器将保持常开，LED断电，被熄
    P0_7=0;
  }
```

至此，我们的 254x 上的程序就编写结束了，我们编译然后下载到开发板中测试。

6、使用 iPhone 上的 LightBlue 软件测试

连接开发板、继电器与台灯，然后按台灯原有的电源开关，给开发板和继电器等供电，SmartRF 开发板上电后自动广播

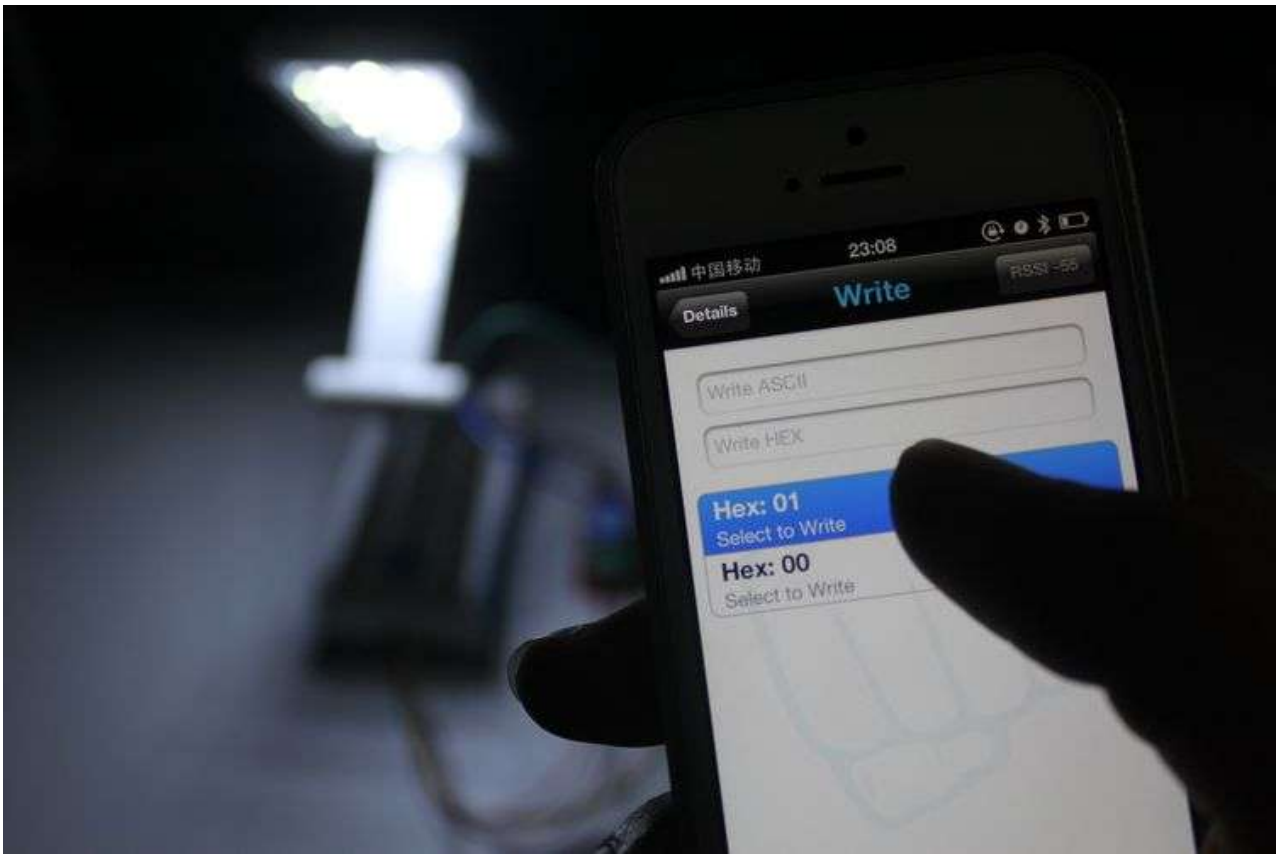


打开 LightBlue 软件，依次找到 uuid 为 0xFFF0 的 Service，然后在找到 UUID 为 0xFFF1 的 Characteristic，通过该 Characteristic 发送控制命令

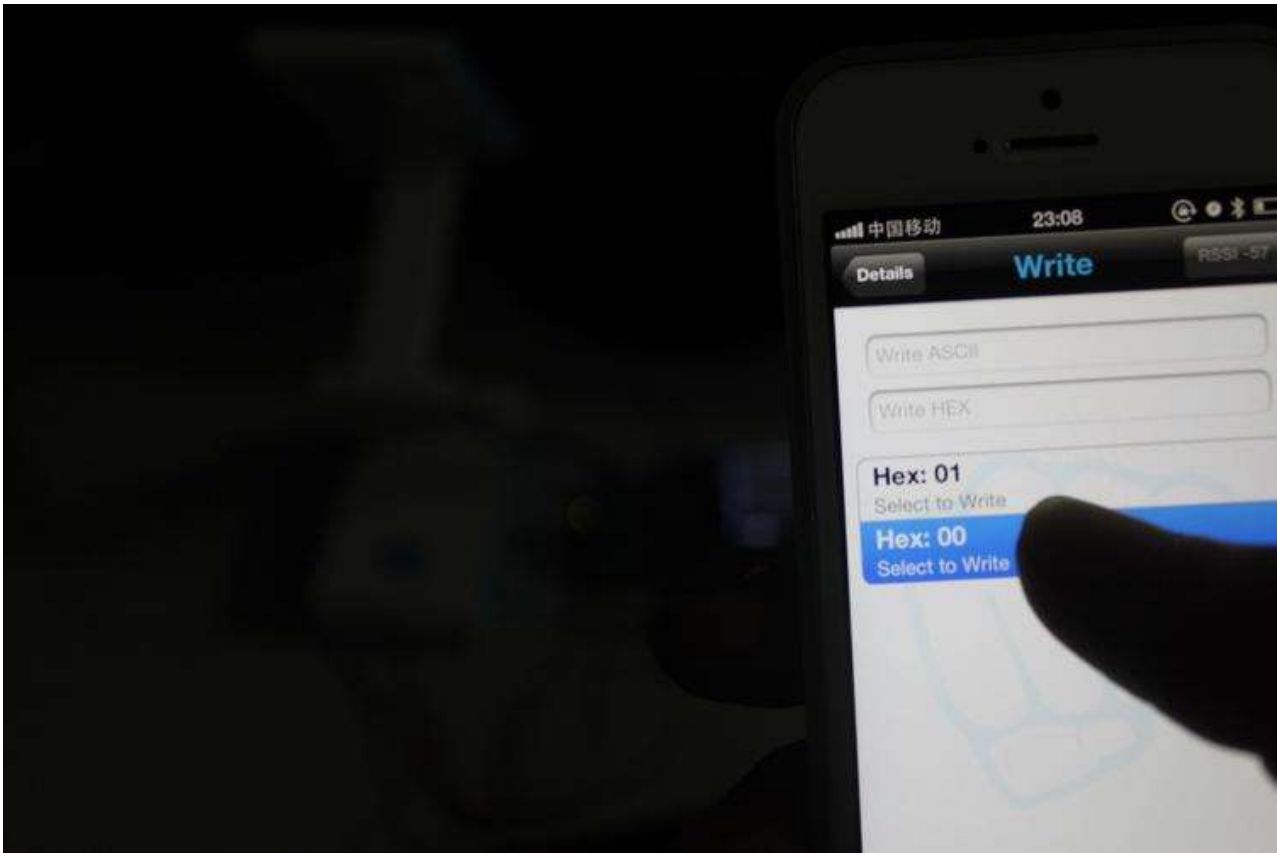
向 Characteristic 发送十六进制数：0x00，熄灭 LED

向 Characteristic 发送十六进制数：0x01，点亮 LED

控制蓝牙台灯开：


中国移动
23:08
Details
Write
RSSI -55
Write ASCII
Write HEX
Hex: 01
Select to Write
Hex: 00
Select to Write

控制蓝牙台灯关：


中国移动
23:08
Details
Write
RSSI -57
Write ASCII
Write HEX
Hex: 01
Select to Write
Hex: 00
Select to Write

成功 !!

7、开发 iOS 配套 APP 程序

8、开发 Android 配套 APP 程序

附录：SimpleBLEPeripheral_DeskLamp 程序下载

SimpleBLEPeripheral_DeskLamp.rar (236.47 KB，下载次数: 0)