

蓝牙鼠标

基于 CC2541SmartRF 开发板的蓝牙鼠标

刘雨

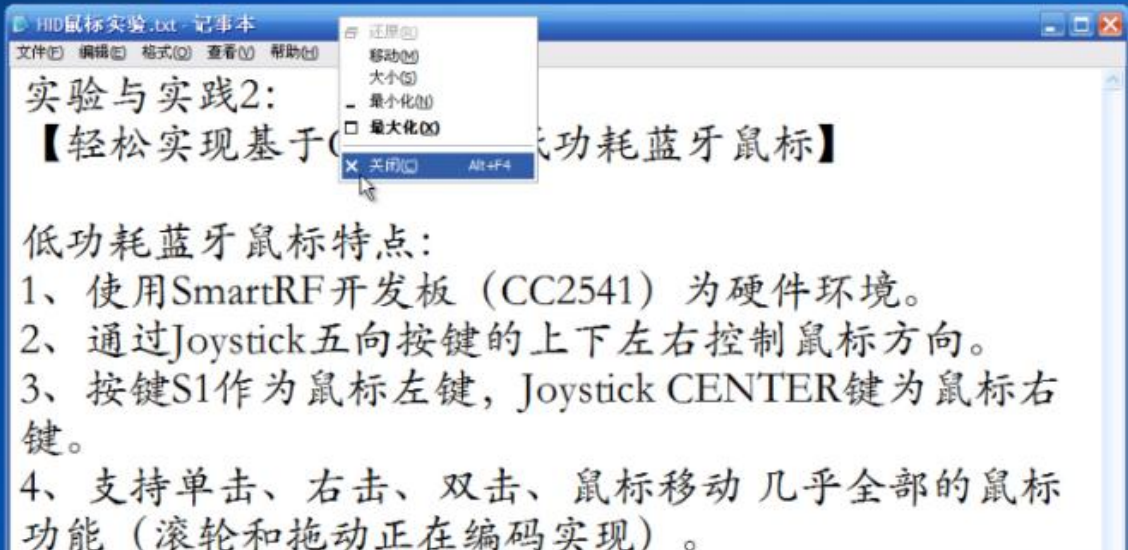
2013-09-12

[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。]

实验与实践 2：轻松实现基于 CC2541 的低功耗蓝牙鼠标

低功耗蓝牙鼠标特点：

- 1、使用 SmartRF 开发板（目前为 CC2541）为硬件环境。
- 2、通过 Joystick 五向按键的上下左右控制鼠标方向。
- 3、按键 S1 作为鼠标左键，Joystick CENTER 键为鼠标右键。
- 4、支持单击、右击、双击、鼠标移动 几乎全部的鼠标功能（滚轮和拖动正在编码实现）。
- 5、可扩展键盘功能。
- 6、支持 XP、WIN7 等系统，为标注 HID 设备



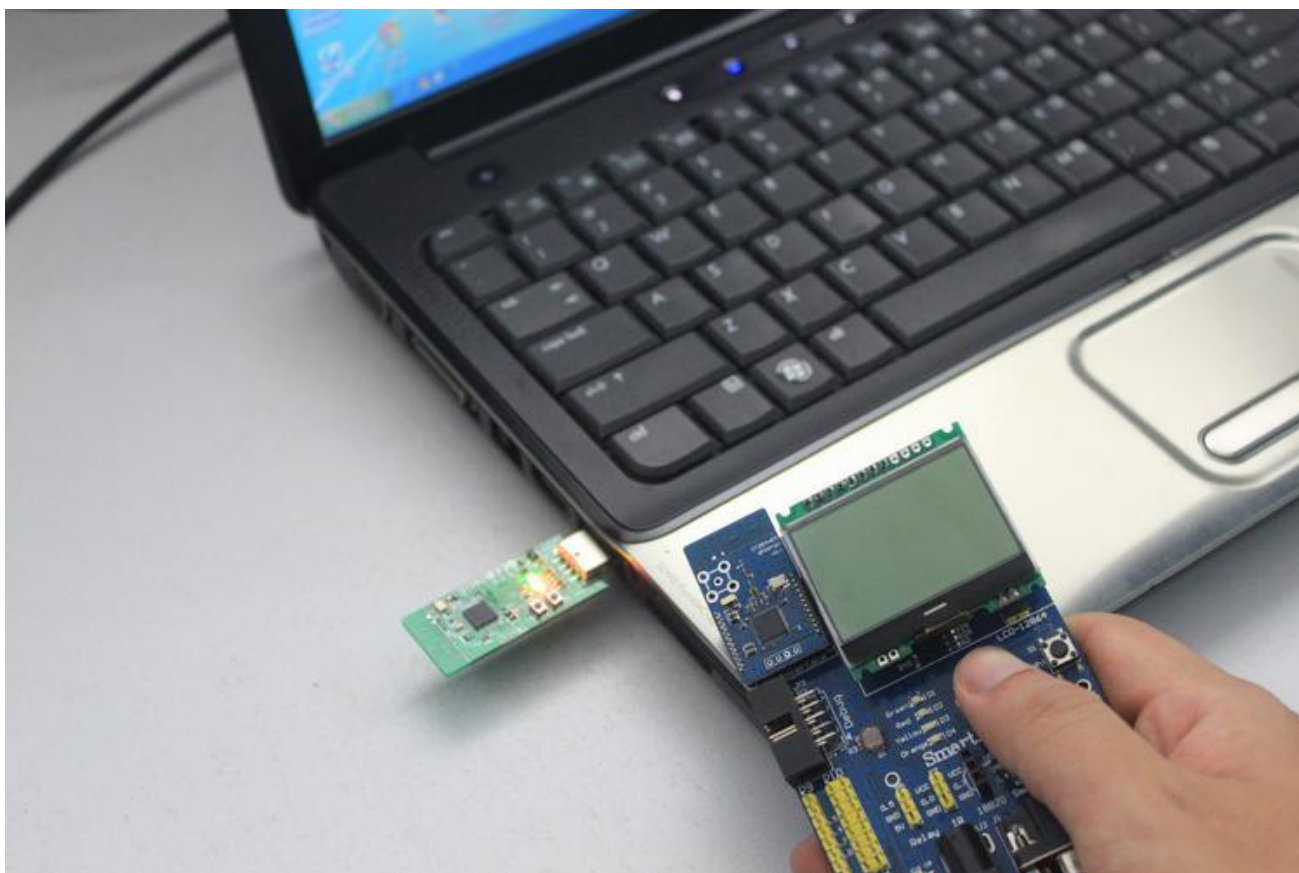
实验与实践2：
【轻松实现基于CC2541的低功耗蓝牙鼠标】

低功耗蓝牙鼠标特点：

- 1、使用SmartRF开发板（CC2541）为硬件环境。
- 2、通过Joystick五向按键的上下左右控制鼠标方向。
- 3、按键S1作为鼠标左键，Joystick CENTER键为鼠标右键。
- 4、支持单击、右击、双击、鼠标移动 几乎全部的鼠标功能（滚轮和拖动正在编码实现）。
- 5、可扩展键盘功能。
- 6、标准HID设备，支持XP、WIN7等系统，为标注 HID 设备

支持单击、右击、双击等操作

蓝牙鼠标



本文内容

1、蓝牙鼠标原理

2、材料准备

3、开发基于 SmartRF (当前为 CC2541) 开发板的蓝牙鼠标程序

4、测试

正文：

1、蓝牙鼠标原理。

在本次实践中，我们所实现的蓝牙鼠标和常规的无线鼠标非常类似，有一个 PC 端的 USB 适配器 (USBdongle)，一个无线鼠标 (SmartRF CC2541)。

我们这里的适配器和鼠标之间是通过低功耗蓝牙来实现，usb dongle 上实现的功能有两个，一个是实现标准的 hid 设备，第二个是接收通过 ble 传送到鼠标数据（单击、右击、x 轴、y 轴等）。

2、材料准备

一个 CC2540USBdongle，用来作为 PC 端的适配器

一个 SmartRF（CC2541）开发板，用来作为模拟无线鼠标

SmartRF 开发板上有一个 Joystick，以及一个 Button，这样，正好利用 joystick 的上下左右键模拟鼠标的上下左右移动，剩下的 center 键和 button S1 按键来模拟鼠标的右击和左击。

另外还需要 CC-debugger 仿真器以及 BLE 的开发板环境。

3、开发基于 SmartRF（当前为 CC2541）开发板的蓝牙鼠标程序

与蓝牙台灯类似，我们依然是修改现有的协议栈 demo，来实现我们的功能，这是常规的开发方式。

TI 的 BLE 协议栈，从 1.3.1 开始添加了新的硬件环境：CC2541ARC，一个基于 2541 的万能遥控器，带有三轴三速度计以及阵列按键，该 CC2541ARC，可以用来做体感游戏手柄。也可以作为常规的鼠标和键盘。我们现在就来将这个 demo 移植到我们的 SmartRF 开发板上，来实现无线鼠标功能。该移植意义非常重大，读者可以按照这种方法，实现自己的蓝牙键盘或者蓝牙鼠标。至于硬件环境嘛，可以任意修改。至于接收端的适配器，直接使用现有的 demo，无线修改，烧写到我们的 CC2540USBdongle 中即可

开发环境：

协议栈版本：1.3.2

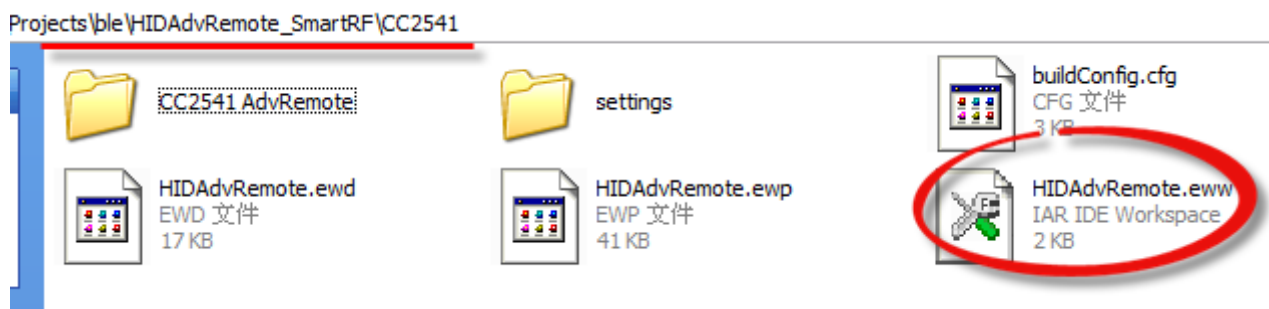
IAR 版本：IAR for 8051 8.10

原鼠标 demo 例程：HIDAdvRemote

原适配器 demo 例程：HIDAdvRemoteDongle

首先是需要我们修改的是 HIDAdvRemote，也就是无线鼠标 demo。

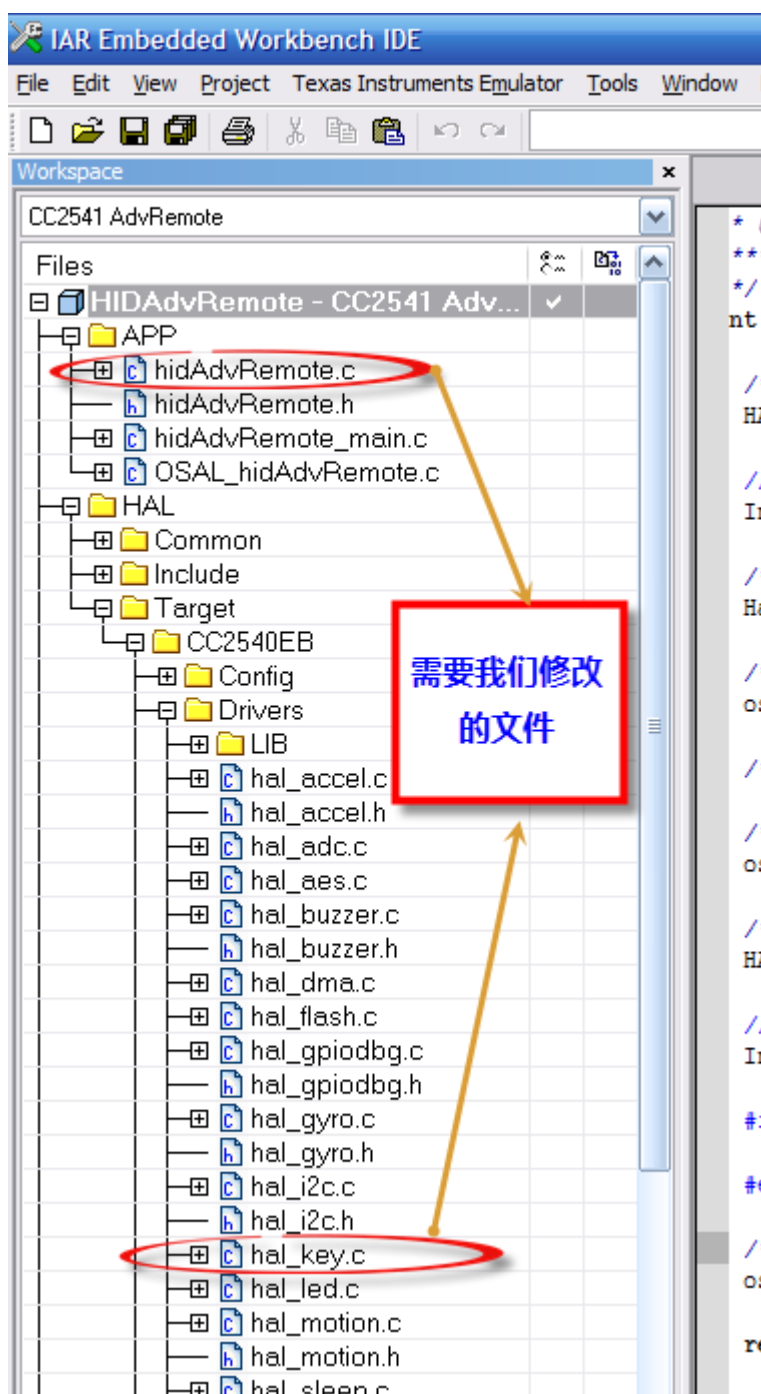
复制 HIDAdvRemote 副本为 HIDAdvRemote_SmartRF，这是一个很好的习惯，在修改任何程序之间，我们需要新建一个原 demo 的副本。



打开 IAR 工程，需要我们修改的有以下两个源文件：

1、hidAdvRemote.c

2、hal_key.c



hidAdvRemote.c 是蓝牙鼠标的程序主体，协调按键，三轴传感器等一起实现无线鼠标的功能，由于原本的 demo 使用的是一个遥控器的硬件环境，而我们需要将其修改到 SmartRF 开发板上，所以除了修改按键部分还要修改这个文件里的相关内容

hal_key.c 每个硬件环境，比如 SmartRF、usbdongle 等只要有按键的，都有一个配套的 hal_key.c 这里同样不例外，TI 遥控器 ARC 的 keys 更加特殊，使用了整

列扫描，由于端口资源有限，还加入了移位的芯片。因为我们要重新修改代码，以驱动 SmartRF 开发板上的 key

hidAdvRemote.c 修改 key 回调函数

```
00635: /*****
00636:  *
00637:  * @fn      hidAdvRemoteKeyCback
00638:  *
00639:  * @brief   Callback service for keys
00640:  *
00641:  * @param   keys - key that was pressed (i.e. the scanned row/col index)
00642:  *          state - shifted
00643:  *
00644:  * @return  void
00645:  */
00646: #define GHOSTYU_CODE_NOKEY 0x00
00647: static void hidAdvRemoteKeyCback(uint8 keys, uint8 state)
00648: {
```

```

00692:      // Process SW1 here
00693:      if ( keys & HAL_KEY_SW_1 )    // up
00694:      {
00695:          hidMouseSendReport( 0, 0, -keyRepeated++ );
00696:          HalGpioUartSendString("key UP\r\n",8);
00697:      }
00698:      // Process SW2 here
00699:      if ( keys & HAL_KEY_SW_2 )    // right
00700:      {
00701:          hidMouseSendReport( 0, keyRepeated++, 0 );
00702:          HalGpioUartSendString("key RT\r\n",8);
00703:      }
00704:      // Process SW3 here
00705:      if ( keys & HAL_KEY_SW_3 )    // down
00706:      {
00707:          hidMouseSendReport( 0, 0, keyRepeated++ );
00708:          HalGpioUartSendString("key DN\r\n",8);
00709:      }
00710:      // Process SW4 here
00711:      if ( keys & HAL_KEY_SW_4 )    // left
00712:      {
00713:          hidMouseSendReport( 0, -keyRepeated++, 0 );
00714:          HalGpioUartSendString("key LT\r\n",8);
00715:      }
00716:      // Process SW5 here
00717:      if ( keys & HAL_KEY_SW_5 )    // center,here is HID_MOUSE_BUTTON_RIGHT
00718:      {
00719:          mouseButtonStates |= 0x02;
00720:          sendMouseReport = TRUE;
00721:          HalGpioUartSendString("key MR\r\n",8);
00722:      }
00723:      // Process SW6 here
00724:      if ( keys & HAL_KEY_SW_6 )    // s1,here is HID_MOUSE_BUTTON_LEFT
00725:      {
00726:          mouseButtonStates |= 0x01;
00727:          sendMouseReport = TRUE;
00728:          HalGpioUartSendString("key ML\r\n",8);
00729:      }

```

hidAdvRemoteKyeCback 是按键的回调函数，当有按键触发时，会调用此函数来处理是哪个按键按下然后执行相应动作。

我们分别在每个按键处理下面天下之前定义的动作，joystick 的上下左右代表鼠标方向，joystick 的 center 代表右击，Button S1，代表左击。如上述截图。

那么鼠标上传的数据是什么样的呢，如下图：

是一个四个字节的数组，

index0 代表，鼠标点击的状态，0x01 表示左击，0x02 表示右击

index1 代表鼠标 x 轴移动的距离，坐标原点是屏幕的左上角，因此，x 轴为负数，表示向左，正数表示向右

index2 代表鼠标 y 轴移动的距离，与 x 轴类似。

index3 代表滚轮，不讨论。

```
00935:  * @param  mouseStates - bitmap of left/middle/right mouse bu
00936:  * @param  mickeysX - amount of mouse movement along X-axis i
00937:  * @param  mickeysY - amount of mouse movement along Y-axis
00938:  *
00939:  * @return  none
00940:  */
00941: static void hidMouseSendReport( uint8 mouseStates, int8
00942: {
00943:     uint8 buf[HID_MOUSE_IN_RPT_LEN];
00944:
00945:     // No need to include Report Id
00946:     buf[0] = mouseStates;           // Buttons
00947:     buf[1] = mickeysX;              // X
00948:     buf[2] = mickeysY;              // Y
00949:     buf[3] = 0;                    // Wheel
00950:
00951:     HidDev_Report( HID_RPT_ID_MOUSE_IN, HID_REPORT_TYPE_INPUT,
00952:                   HID_MOUSE_IN_RPT_LEN, buf );
00953: }
00954:
```

到目前为止，我们完成了按键上层代码的修改，底层按键也需要我们修改，以适应 SmartRF，按键部分修改的较多，具体代码请参考我们提供的代码。在开发资料中的【实验与实践】目录下的同名文件夹下。

另外还有无线鼠标的接收端，usb 适配器，使用 CC2540USBdongle，这里面的程序无需修改，保持原样即可，工程为：HIDAdvRemoteDongle

4、测试

(1) 程序烧写

将刚刚修改的 HIDAdvRemote 工程重新编译，然后使用 CC-Debugger 下载到 SmartRF (CC2541) 中，注意 smartrf04eb 不支持 CC2541

将 HIDAdvRemoteDongle 重新编译烧写到 CC2540USBdongle 中。

(2) 连接

将 CC2540USBdongle 插到电脑 USB 上，如果程序烧写正确的话，会跳出 hid 设备驱动的安装，过一会自动安装成功。

这时的 usbdongle 的 D1 红灯将常亮，然后按一下 usbdongle 上的 S2 按键，伴随着 D1 的闪烁开始搜索蓝牙鼠标。注意，搜索只持续几秒，需要在这几秒内，按 SmartRF 开发板上的任意按键，这样两者才能建立连接。而且 usbdongle 的 D2 绿色会变量，我们再按 SmartRF 开发板的按键时，usbdongle 的 D1 红灯会随着 SmartRF 的按键按下而亮。

此时，SmartRF 就是一个无线鼠标拉，enjoy it !