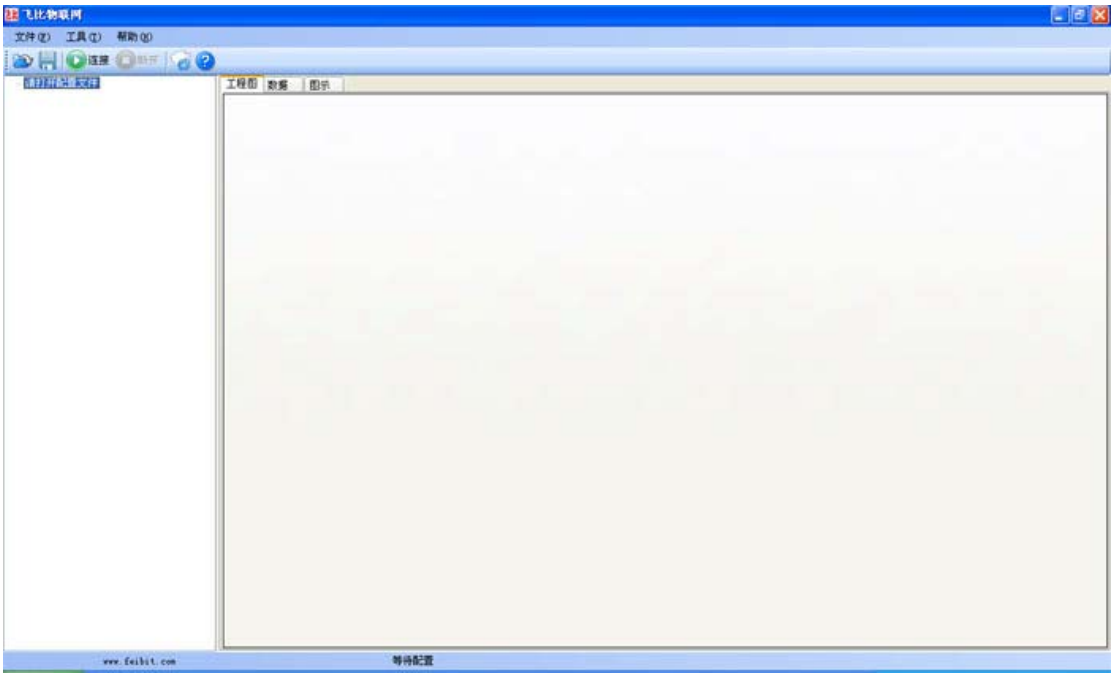
快速开始之二、用物联网浏览器构建应用场景

上篇内容实现了本地的采集和控制，实现的工具仅仅是用的“串口助手”，当然这个工具对调试非常有帮助，但这明显是给“工程师”用的软件，不是给“客户”用的。如果你敢让你的客户为了看一个电灯是亮的还是灭的，到一堆数据里找一个 0 还是 1，你的老板会非常贴心地给你放一个长长的假期… …

本文主要介绍如何通过物联网浏览器（FIT Exploer 软件），构建应用场景，并在本地“可视化”地监控数据，并发送指令。

先互相认识下吧，到 http://bbs.feibit.com/thread-4370-1-1.html 这里把它下载下来，和普通 windows 软件一样安装，然后打开。一个简单得像张白纸一样的界面出现在你面前：



没错，它就是一张白纸，却可以画出你想得到的所有东西，而且它可以“动”，动的不但是界面的图片，关键是那些动画下面所连接的是真实的世界，它能告诉你今天风力几级，海水温度多少；你也能通过它来打开家里的空调，为你的宠物喂水… …

这就是 IoT Explorer 和我们常用的 Internet Explorer 的区别，我们已经把它实现了，并且免费供大家使用，接下来我们能做的，只是做一些 Demo，然后告诉您怎么用。但您的想像力才是它的灵魂，有好的作品，一定记得和我们分享!

好了，继续我们原来的工作，把我们的电灯装上来!

在开始这个工作之前，要做一个准备工作，就像一个画家要画一幅画，第一步是要构思，就是先想象一下，理想中的场景是怎样的？作为一个最简单的例子，我们想象有个屋

子，客厅里放着一盏电灯，电灯旁边一个开关，当点击开关时，zigbee 节点所控制的电灯动作，并且将结果反馈回控制界面，通过亮、灭两张不同的图片来表示。

按照这个构思，我们先选好素材：

1、背景（智能家居.jpg）

2、电灯（开灯.png/关灯.png）

3、开关按钮（按钮开.png/按钮关.png）

准备工作基本就绪了，但是在 FIT Explorer 中到底是如果将这些元素组合成一个场景的呢？这要借助于 xml 语言，请看下面一段简单的例子：

```
<?xml version="1.0" encoding="GB2312"?>
<Title Name="我家的远程电灯" Background="NULL" ComPort="COM3" ComBandrate="38400" NetPort="8090" AddressMode="2" DataValidTime="20" ConnectType="0" UptoFbCloud="0" GetCTLFbCloud="0" AreaCNT="1" SensorCNT="0">
  <Area0>
    <AreaAttrib AreaName="我的家" Background="pic\智能家居.jpg" NodeCNT="0" />
  </Area0>
</Title>
```

<Title>开始，</Title>结束，<Area0>开始，</Area0>结束，看过 html 代码的朋友，可能会觉得有点亲切。至于 xml 语言是什么，怎么写，这里不去深究，有兴趣的朋友可以专门了解下。我们只要知道我们想改的东西在哪里就可以了。

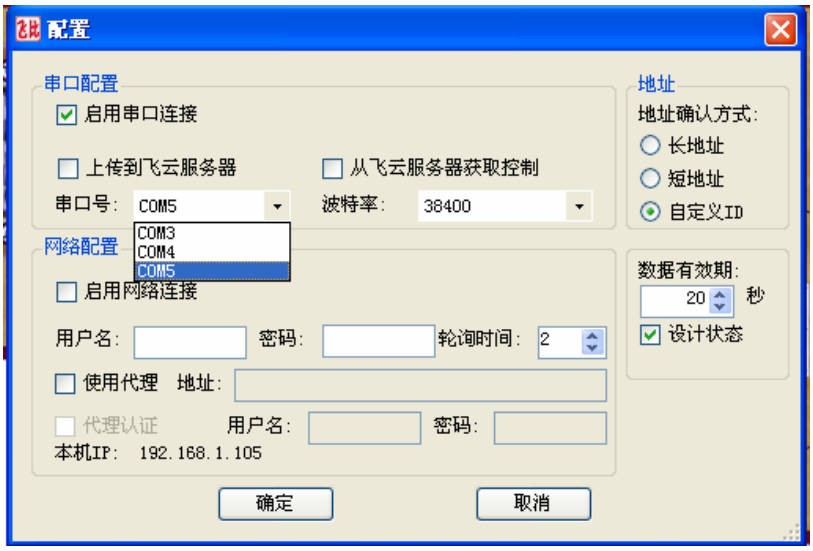
把上面一段代码复制到记事本，保存成“我家的远程电灯.xml”的文件（记得不是“我家的远程电灯.xml.txt”哦），然后在这个 xml 文件所在的目录下建立新的文件夹，命名为“pic”，并将前面的所有图片都复制到这个文件夹下。

此时打开 FIT Explorer 软件，点击  ，选择刚刚建立的“我家的远程电灯.xml”文件，看看“白纸”上是不是已经有些变化了？

看到这里，是不是就知道前面 xml 代码中的红色字体内容，分别是做什么的了？

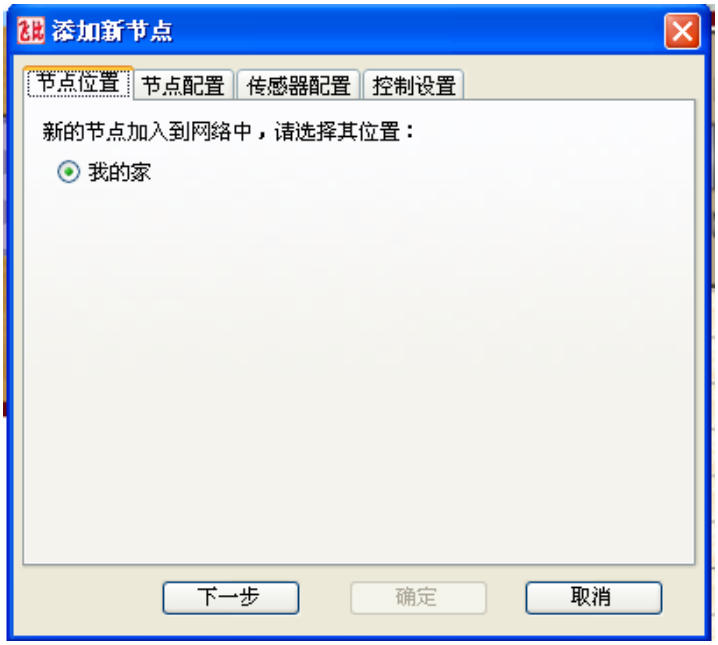
但是电灯、开关在哪里呢？不用着急，听我一一道来。在前一节的内容中，我们已经通过“串口助手”看到过传感器数据，紧接着，我们就要把这个数据以可视化的方式，放到“物联网浏览器”里来：

如果您已经跟着上一篇的内容做完实验，现在要做的就是关闭串口助手，然后打开 FIT Explorer 软件，否则会出现串口冲突的情况。接着点击  ，进入设置界面：



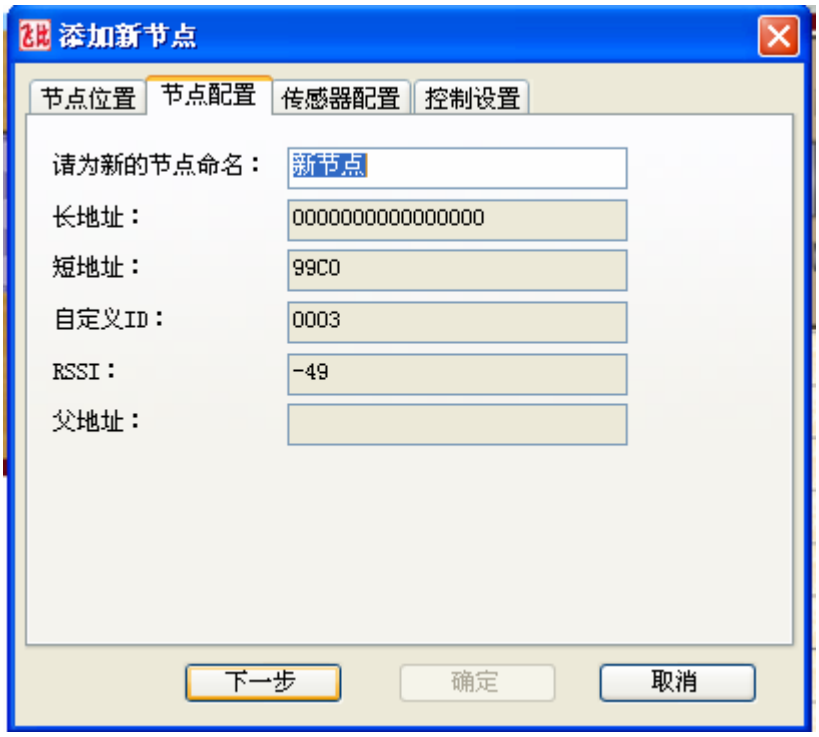
按照上图做好设置后，点“确定”回到主界面，然后点 

看看出现什么了？是不是弹出了如下一个对话框：



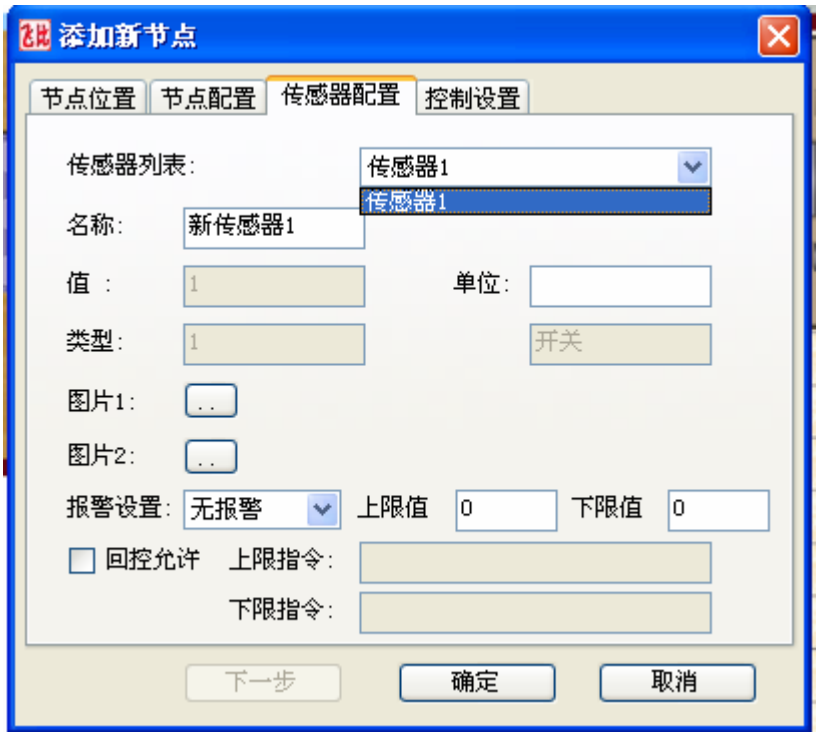
这说明 FIT Explorer 软件已经从 Zigbee 协调器的串口上收到了一组完整的传感数据，在等待您给传感器命名。

点击“下一步”：

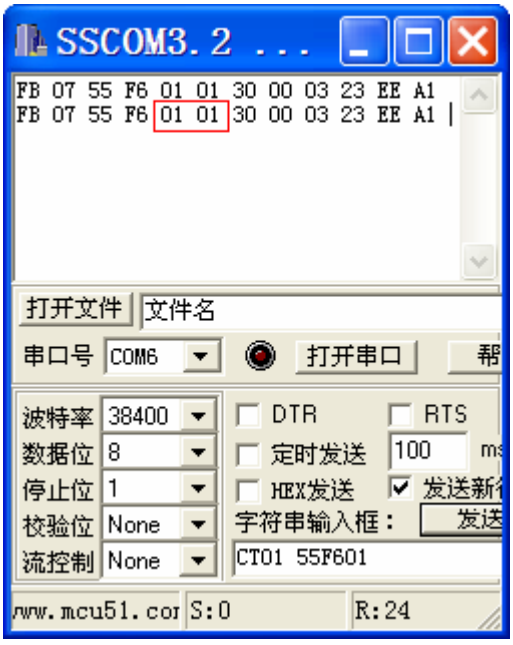


我们把这个节点命名为“Zigbee 电灯”，在下面的数据中，我们可以看到更多的信息：长地址，即 IEEE 地址是空的，说明这组数据中没有带长地址；短地址是 99C0；自定义 ID，即 UserID 为 0201（详情还放在后面解释）；RSSI，即信号质量为-49dbm。

继续下一步：

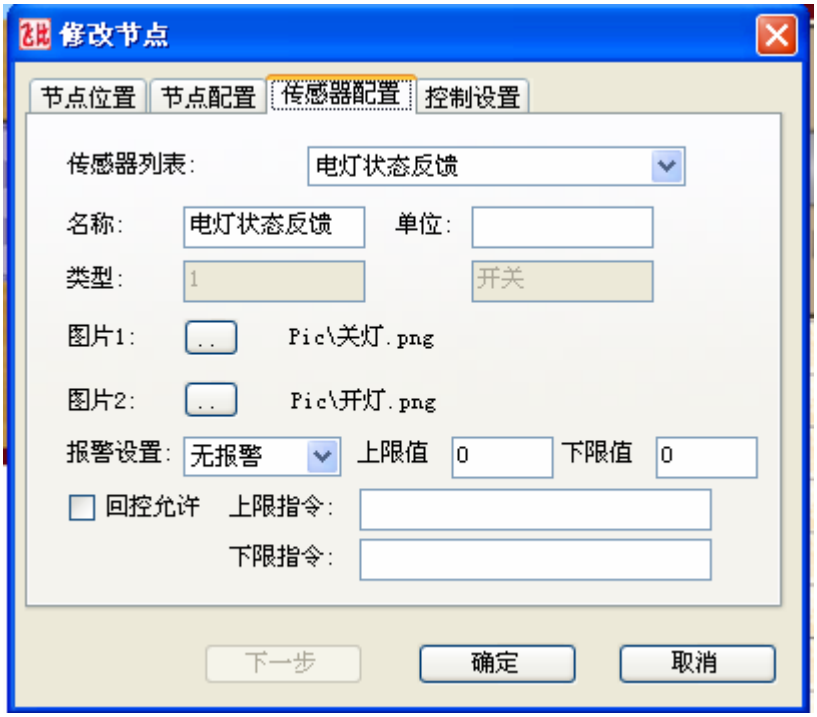


从传感器的下拉表中，只有一个选项，说明收到的一组数据中只有一种传感器，还记得上一篇中的数据“0101”吗？



实际上这一串数据，只有这一个“传感器数据”，那就是电灯状态的“反馈”信号，其它的数据都是辅助数据。

我们把传感器名称命名为“电灯状态反馈”，然后“图片 1”和“图片 2”分别指：反馈数据为 0 和 1 时，界面中所显示的图片。这里我们指定为素材中的那两个灯泡，按下图进行设置：



点“确定”后，发现主界面上多了一个灯泡：

但灯泡的位置和大小似乎不是太满意，那我们看下用 FIT Explorer 是如何实现可视化的场景布置的：

1、首先确认设置界面中的 ☑ 设计状态 是否已经勾选的

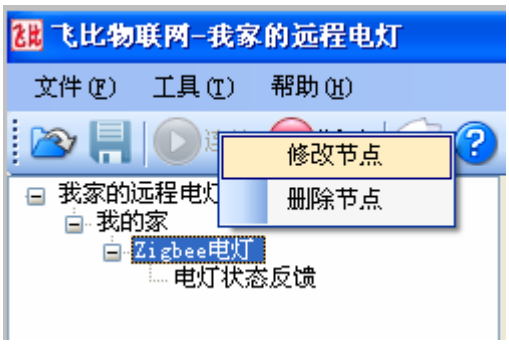
2、通过鼠标点击、选中灯泡后，出现 8 个缩放点，如下图：

3、此时，用鼠标进行大小的缩放、位置的拖动，这一点和普通 windows 软件没什么区别，不再啰嗦。

至此，我们完成了灯的状态的可视化采集，为检测我们的成果，您可以将采集 IO 口的杜邦线分别连接板上的 3V 或者地，看下灯的状态是否发生了变化。

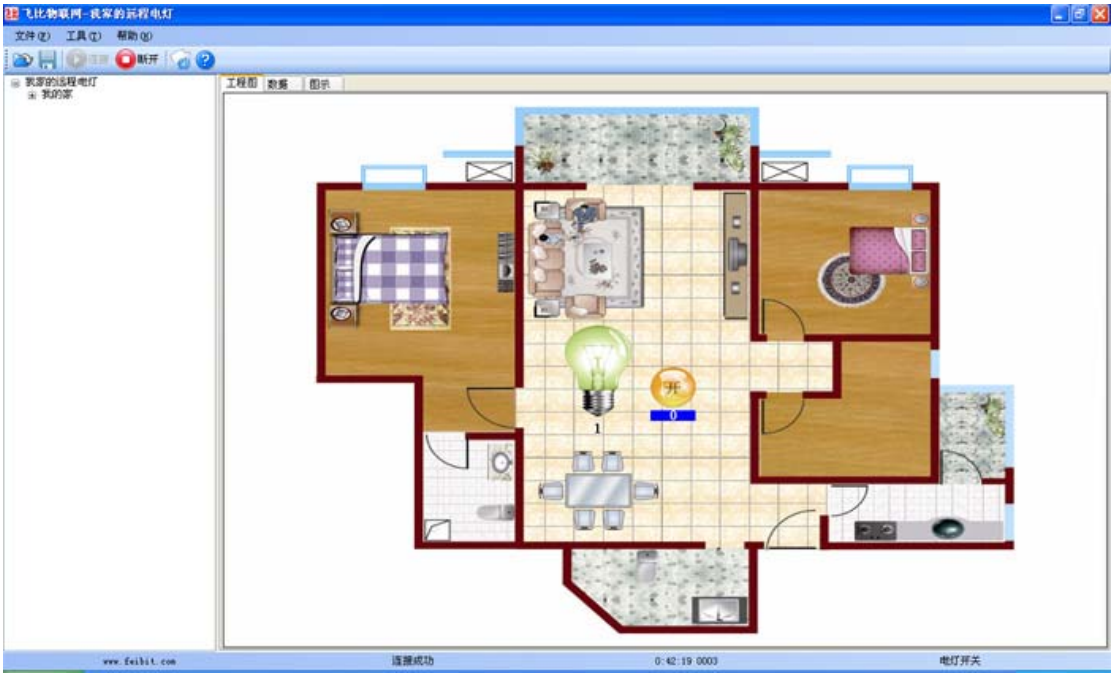
接下来，我们一起看下如果增加“开关”这样的控制对象：

在左侧对象列表中选择“Zigbee 电灯”，并点击右键，选择“修改节点”，如下图：



然后按如下设置进行配置：

在这里，我们启用了一个可控开关，取名为“电灯开关”，设置了“打开”与“关闭”动作所对应的不同的图片。经过调整后，工程图如下所示：



下面我们来看下在点击这两个不同的按钮时，如何设置其向串口发送的控制指令：

首先，我们看下“我家的远程电灯.xml”文件，在我们前面做了一系列的设置后，内容发生了什么变化：

```
<?xml version="1.0" encoding="GB2312"?>
<Title Name="我家的远程电灯" Background="NULL" ComPort="COM5" ComBandrate="38400" NetPort="8090" AddressMode="2" DataValidTime="20" ConnectType="0" UptoFbCloud="0" GetCTLFbCloud="0" AreaCNT="1" SensorCNT="2">
  <Area0>
    <AreaAttrib AreaName="我的家" Background="pic\智能家居.jpg" NodeCNT="1" />
    <Zigbee 电灯>
      <NodeAttrib LongAddress="0000000000000000" ShortAddress="99C0" UserID="0003" NodePicture="" NodeX="0" NodeY="0" NodeType="0" SensorCNT="2" />
      <电灯状态反馈 Type="1" PicName="Pic\关灯.png" PicName2="Pic\开灯.png" Unit="" SensorX="386" SensorY="420" SensorWidth="76" SensorHeight="211" AlarmType="0" AlarmUpper="0" AlarmDown="0" AEnable="0" UpAlarm="" DownAlarm="" />
      <电灯开关 Type="4097" PicName="Pic\按钮开.png" PicName2="Pic\按钮关.png" Unit="" SensorX="484" SensorY="496" SensorWidth="52" SensorHeight="101" AlarmType="0" AlarmUpper="0" AlarmDown="0" AEnable="0" UpAlarm="" DownAlarm="" />
    </Zigbee 电灯>
  </Area0>
</Title>
```

是不是自动增加了很多内容？这里我们不一一进行讲解了，有兴趣的朋友可以读一下这段代码，还是不难理解的。我们重点来看下“电灯开关”这个对象，因为我们下面要对

它的控制指令进行设置：

```
<电灯开关 Type="4097" PicName="Pic\按钮开.png" PicName2="Pic\按钮关.png" Unit="" SensorX="484" SensorY="496" SensorWidth="52" SensorHeight="101" AlarmType="0" AlarmUpper="0" AlarmDown="0" AEnable="0" UpAlarm="" DownAlarm="" />
```

这段代码中的 UpAlarm="" DownAlarm=""，引号中间的内容，分别为点击“打开”与“关闭”两个按钮时，向协调器串口发送的指令字符。

回顾上一篇的内容，如果我们要控制某节点的 IO 口为高电平，可以向协调器串口发送“CTO1 HHLL01”这样的指令，其中 HHLL 代表的是节点的“短地址”。

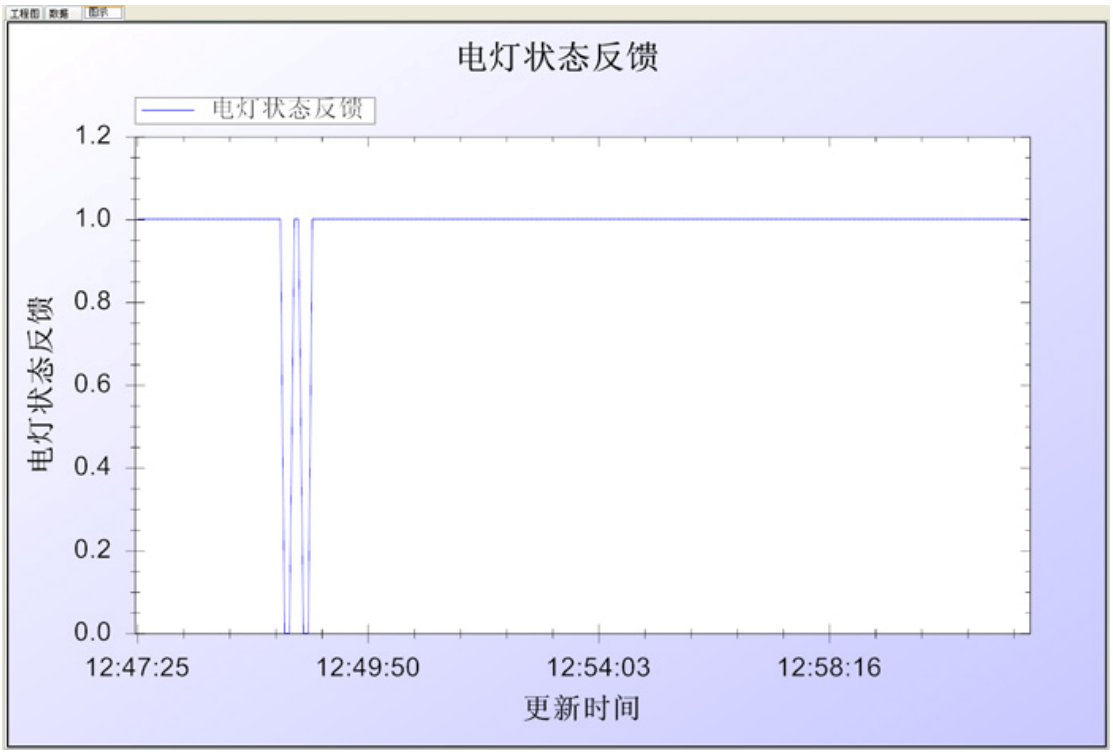
但 zigbee 短地址，有可能在某些条件下发生变化。为了避免这种情况，而且又不用使用过长的 IEEE 地址，我们的默认设置采用了用户自定义 ID，即 UserID 的方式来区分节点。而且 FIT Explorer 软件已经实现了 UserID 与短地址的自动转换，比如 UserID 为 0201 的节点，其短地址为 99C0，如果我们想发送“CTO1 99C001”这样的控制指令，可以转换为“CTO1 &uid0201&01”。这样，当节点短地址变动时，软件将自动获取新的短地址，并替换 &uid0201&，这样用户就可以不用关注 zigbee 短地址，只关注自己设置的 UserID 号，即可区分不同的设备。

在 xml 语言中，“&”这个符号是用&符号代表的，所以最终的控制代码如下：

```
UpAlarm="CTO1 &amp;uid0201&amp;01" DownAlarm="CTO1 &amp;uid0201&amp;00"
```

将这段代码替换到原 xml 文件中，保存，然后用 FIT Explorer 重新打开，点击界面里的开关按钮，看看节点上的灯是不是按照预期进行亮灭，并且将状态反馈上来了呢？

顺便看下 FIT Explorer 的两个小功能：数据列表和曲线



至此，我们已经成功得通过 FIT Explorer 软件，构建了一个简单的应用场景，当然软件本身的功能远不只这些，随着后续教程的深入，我们慢慢从应用中进行了解。