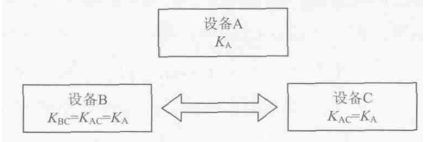
# 材料说明

## 蓝牙中间人攻击原理

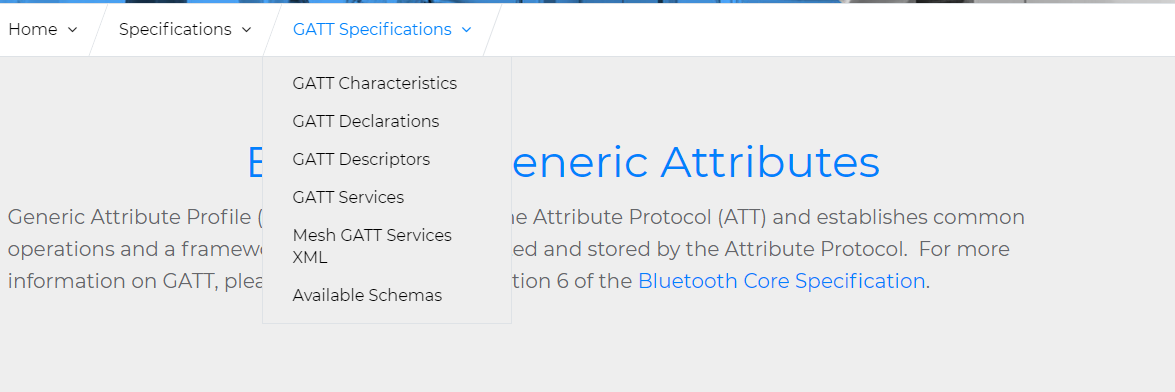
在两个设备之间的攻击者截获数据一方发送的数据后再转发给另一方，可在不影响双方通信的情况下获得双方通信的内容，是一种广泛应用于无线网络的攻击方式。蓝牙 4.0 版本的低功耗蓝牙技术（Bluetooth Low Energy，BLE）在设计初始时有防范中间人攻击的安全措施，但是在产品阶段考虑到产品功耗成本等因素，这方面并没有得到足够的重视，依然容易受到攻击。最常见的是用软硬件结合的蓝牙攻击设备伪造BLE 通信进行中间人攻击。中间人攻击示意图下图所示。



## 蓝牙数据包简析

### 3.1 GATT层

关于GATT的一些服务的名称，可通过下面的网址进行查阅：

<https://www.bluetooth.com/specifications/gatt/>

蓝牙协议中包含了许多种GATT规范，每个规范适配一种用户案例，比如FindMe规范适配查找物件的场景，心率传感器规范适配心率测量场景。

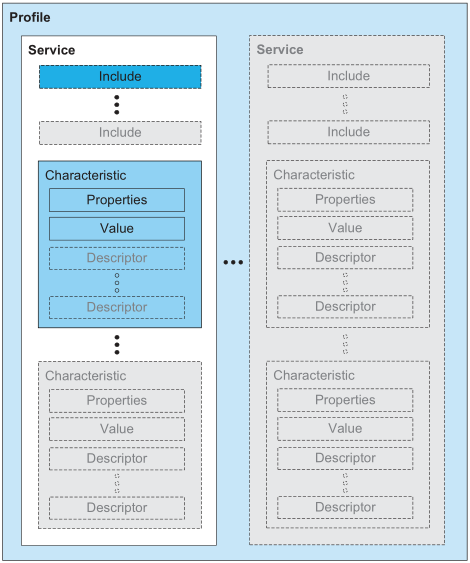
每个规范均中均有若干服务项和特征项，服务项和特征项都属于属性实体，它们携带了通信中传输的数据。

服务项分为主要服务和次要服务，主要服务可以引用（Include）另一个主要服务或次要服务，客户端设备可以通过“主要服务发现过程”获取主要服务信息。

特征项包括一个声明、配置、数据和描述符。描述符用于描述特征项的数据如何被访问和展示。

规范、服务项和特征项之间有明确的包含关系，一个GATT规范中可以包括多个服务项，一个服务项中可以包括多个特征项。

GATT的规范结构框图如下：



#### 2.2.1属性

一个属性包含句柄、UUID(类型)、值，句柄是属性在GATT表中的索引，在一个设备中每一个属性的句柄都是唯一的。UUID包含属性表中数据类型的信息，它是理解属性表中的值的每一个字节的意义的关键信息。在一个GATT表中可能有许多属性，这些属性能可能有相同的UUID。

#### 2.2.2 特性

一个特性至少包含2个属性：一个属性用于声明，一个属性用于存放特性的值。

所有通过GATT服务传输的数据必须映射成一系列的特性，可以把特性中的这些数据看成是一个个捆绑起来的数据，每个特性就是一个自我包容而独立的数据点。 例如，如果几块数据总是一起变化，那么我们可以把它们集中在一个特性里。

#### 2.2.3描述符

任何在特性中的属性不是定义为属性值就是为描述符。描述符是一个额外的属性以提供更多特性的信息，它提供一个人类可识别的特性描述的实例。

然而，有一个特别的描述符值得特别地提起：客户端特性配置描述符(Client Characteristic Configuration Descriptor，CCCD)，这个描述符是给任何支持通知或指示功能的特性额外增加的。

在CCCD中写入“1”使能通知功能，写入“2”使能指示功能，写入“0”同时禁止通知和指示功能。

#### 2.2.4服务

一个服务包含一个或多个特性，这些特性是逻辑上相关的集合体。

GATT服务一般包含几块具有相关的功能，比如特定传感器的读取和设置，人机接口的输入输出。组织具有相关的特性到服务中既实用又有效，因为它使得逻辑上和用户数据上的边界变得更加清晰，同时它也有助于不同应用程序间代码的重用。GATT基于蓝牙技术联盟(SIG)官方而设计，SIG建议根据它们的规范设计自己的profile。

#### 2.2.5 profile（数据配置文件）

一个profile文件可以包含一个或者多个服务，一个profile文件包含需要的服务的信息或者为对等设备如何交互的配置文件的选项信息。设备的GAP和GATT的角色都可能在数据的交换过程中改变，因此，这个文件应该包含广播的种类、所使用的连接间隔、所需的安全等级等信息。

需要注意的是： 一个profile中的属性表不能包含另一个属性表。

#### 2.2.6 标准的定制服务和特性

蓝牙技术联盟(SIG)已经定义一些profile、服务、特性和根据协议栈的GATT层定义的属性。但是，协议栈中只实现了一部分应用的BLE服务，那就意味着，只要协议栈支持GATT，就可能为一个应用建立一个它需要的profile和服务。

既然在一个应用中可以支持profile和服务，那么就可以在这个应用中建立一个定制的服务。

#### 2.2.7 UUID

“GATT层”中定义的所有属性都有一个UUID值，UUID是全球唯一的128位的号码，它用来识别不同的特性。

1. 蓝牙技术联盟 UUID

蓝牙核心规范制定了两种不同的UUID，一种是基本的UUID，一种是代替基本UUID的16位UUID。

所有的蓝牙技术联盟定义UUID共用了一个基本的UUID：

0x0000xxxx-0000-1000-8000-00805F9B34FB

为了进一步简化基本UUID，每一个蓝牙技术联盟定义的属性有一个唯一的16位UUID，以代替上面的基本UUID的‘x’部分。例如，心率测量特性使用0X2A37作为它的16位UUID，因此它完整的128位UUID为：

0x00002A37-0000-1000-8000-00805F9B34FB

虽然蓝牙技术联盟使用相同的基本UUID，但是16位的UUID足够唯一地识别蓝牙技术联盟所定义的各种属性。

蓝牙技术联盟所用的基本UUID不能用于任何定制的属性、服务和特性。对于定制的属性，必须使用另外完整的128位UUID。

1. 供应商特定的UUID

SoftDevice 根据蓝牙技术联盟定义UUID类似的方式定义UUID：先增加一个特定的基本UUID，再定义一个16位的UUID（类似于一个别名），再加载在基本UUID之上。这种采用为所有的定制属性定义一个共用的基本UUID的方式使得应用变为更加简单，至少在同一服务中更是如此。

使用软件nRFgo Studio非常容易产生一个新的基本UUID：

例如，在LED BUTTON示例中，采用0x0000xxxx-1212-EFDE-1523-785FEABCD123作为基本UUID。

蓝牙核心规范没有任何规则或是建议如何对加入基本UUID的16位UUID进行分配，因此你可以按照你的意图来任意分配。

例如，在LED BUTTON示例中，0x1523作为服务的UUID，0x1524作为LED特性的UUID，0x1525作为按键状态特性的UUID。

#### 2.2.8 空中操作和性质

大部分的空中操作事件都是采用句柄来进行的，因为句柄能够唯一识别各个属性。如何使用特性依据它的性质，特性的性质包括：

1）写

2）没有回应的写

3）读

4）通知

5）指示

写和没有回应的写

写和没有回应的写允许GATT客户端写入一个值到GATT服务器的一个特性中。它们之间不同的地方在于没有回应的写事件没有任何应用层上的确认或回应。

读

读性质表明一个GATT客户端可以读取在GATT服务器中特性的值。

通知和指示

通知和指示性质允许GATT服务器在其某个特性改变的时候对GATT客户端进行提醒，通知和指示之间不同之处在于指示有应用层上的确认，而通知没有。

## 针对BLE设备中间人攻击

这是一个有趣的项目，其中介绍了如何对使用蓝牙低功耗（BLE）进行数据交换的测温枪进行中间人攻击。由于测温枪使用低功耗蓝牙（Bluetooth Low Energy）（是蓝牙4.0的子集），因此从理论上讲任何BLE设备都可以控制灯泡。但是要控制灯泡，必须了解与灯泡进行通信的协议，并且本文将向您展示如何使用Bluefruit LE嗅探器和其他工具对蓝牙低功耗小工具进行中间人攻击。

我们需要如下的一些硬件：

* 蓝牙4.0 USB模块。确保模块支持低功耗蓝牙。 4.0版之前的旧版蓝牙不支持BLE！
* PC计算机。最好使用带有Linux系统的计算机，并保证计算机带有USB口，在Linux系统下能够使用一些很好用的小工具，十分便于我们调试，USB口则是其他硬件设备与计算机连接的必备接口。
* 手机（Andriod或IOS），通过手机端的软件与测温枪相连。
* 测温枪，能够测量身体的温度数据，并通过蓝牙连接到app传输温度数据给手机端。
* Bluefruit LE Sniffer。 Bluefruit LE Sniffer是Bluefruit LE friend的特殊版本，可以允许查看来自其他设备的BLE数据包。

### GATT服务初探

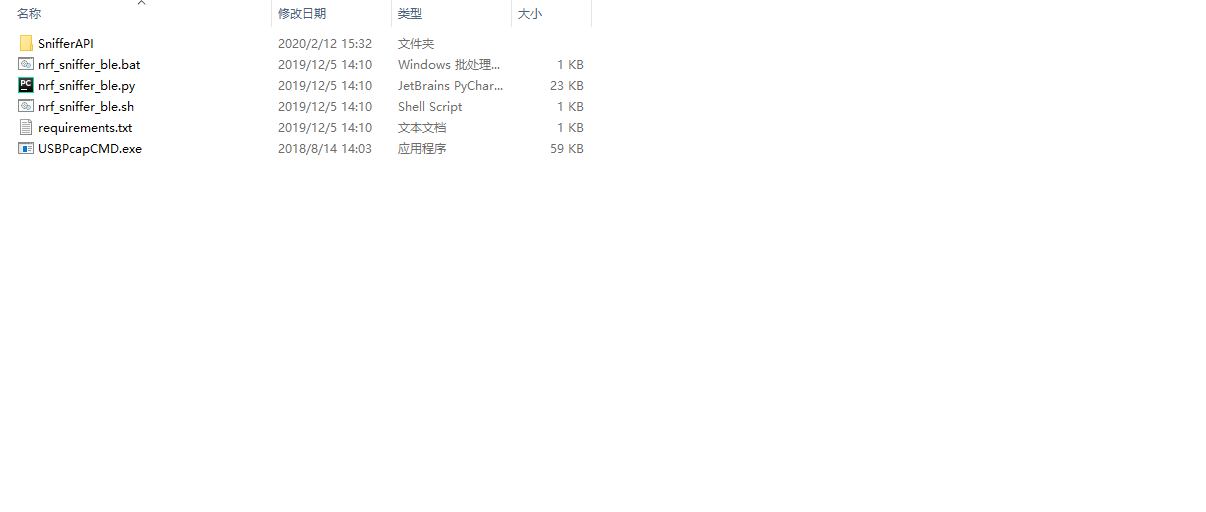
我使用了IOS上的一款名为LightBlue的蓝牙调试软件进行初步测试，这是测试图：



从这张图我们可以看到我们的测温枪的设备名和使用的服务名以及设备的一些信息以及使用该服务发送的一些值。

### WireShrak抓包

我们继续使用WireShark软件对测温枪的发送数据进行探究，要注意的是，在进行这些活动时，测温枪不能连接其专用的APP，否则将会无法抓包。同时，我在github上上传了名为nrf\_sniffer\_for\_bluetooth\_le\_3.0.0\_129d2b3的wireshark插件，将其中的extcap文件夹复制到wireshark对应的文件夹，即可在wireshark中使用低功耗蓝牙抓取的插件，使抓包过程更为便捷：



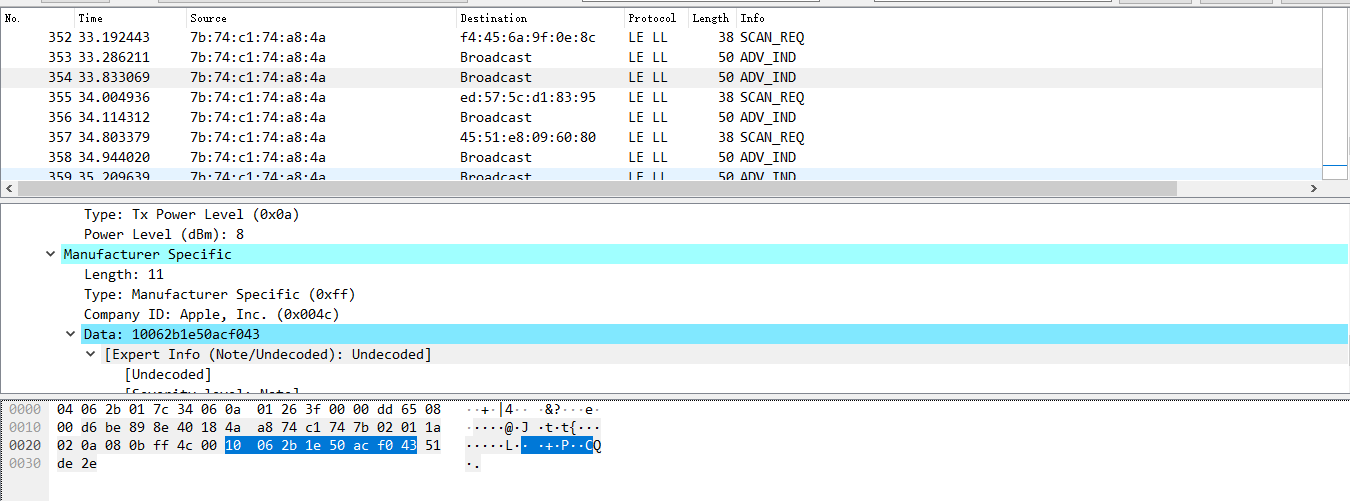
复制后的文件夹



插件示意

下面我们正式开始使用sniffer抓取BLE包，在使用Bluefruit LE sniffer嗅探BLE流量之前，需要注意的一件事是，它可能对来自其他BLE设备的噪声敏感。尝试关闭附近的所有其他BLE设备，例如平板电脑，电话等。还要确保PC上正在运行的程序尽可能少，因为该工具需要尽快从Bluefruit嗅探器获取数据以防止数据包丢失。

Wireshark加载后，很快就会看到大量广告包，例如：

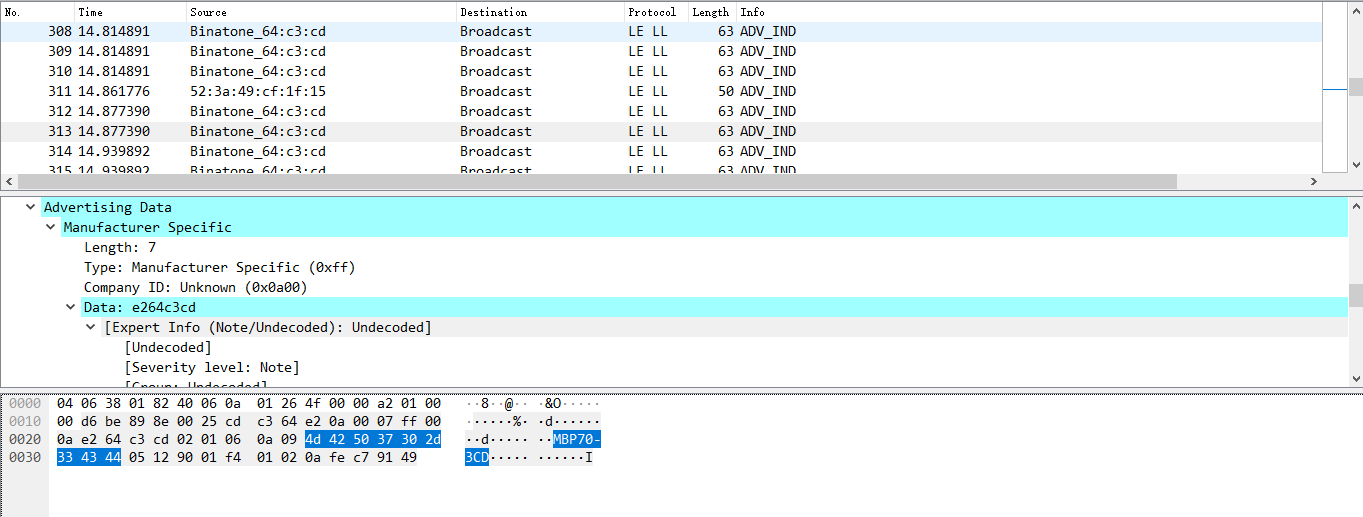


上图所示的窗口可以分为三个部分：

* 前三分之一是已捕获的数据包列表。当wireshark捕获数据包时，此列表将迅速增长。您可以上下滚动查看收到的数据包，也可以单击数据包以查看有关它的更多信息。
* 中间三分之一是已从数据包解码的信息。您可以深入查看数据包中的特定帧，以查看其作用。在这一层我们可以看到一些能够阅读的详细的数据包信息。
* 底部的三分之一是数据包数据的原始十六进制和ASCII表示形式。有趣的是，当您单击中间窗格中的信息时，您会在底部窗格中看到原始信息高亮表示。

在上图中，我选择了一个手机的广告包，并在中间窗格中突出了一些有趣的细节。您可以看到此数据包来自手机，他在不间断的向外发送广播包，同时广播包里面带上了一些自定义的数据。在上面的窗格中我们还可以看到，在app上点击搜索的时候，手机向一些固定的mac地址发送了SCAN\_REQ的广播包。

然后我们启动手机端的app，连接到测温枪，并在这个过程中一直启动sniffer，抓包结果如下图：



可以看到，测温枪一直在发送同一段广播包，里面包含了设备名，以及一段自身的mac地址，但是从协议端可以看到，我们只能观察到LE LL协议的内容，一直没有抓取到ATT类型的包，而只有那种的包才能识别出对我们有用的信息。到这里wireshark抓包就告一段落了。

### 进行MITM攻击

#### 3.3.1接收

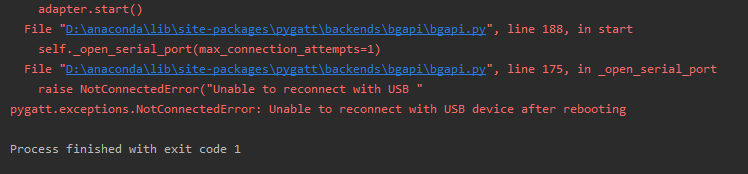
从上面的分析可知，我们这里可以实施较为初级的攻击，即首先将测温枪连接到自己的电脑，同时读取数据，得到测温枪发送的数据格式，但是这一步并不是最重要的。

因为在我们的项目中，测温枪发送温度这一最重要的步骤我们是无法探知的，这不同于其他一些蓝牙设备的交互逻辑，在我们的项目中，从设备（即测温枪）是数据的发送方，所以他发送的数据必须要连接上主设备后才开始发送，所以我们不能达到他发送温度数据的格式。但是如果是一个使用手机蓝牙控制小灯这样的应用中，就算没有连接上小灯，我们仍然可以通过手机发送命令，这就给了我们探究该命令数据格式的机会。但是所幸我们的项目发送数据必须使用规定的服务方式，所以理论上数据格式是固定的。

然后这一步也是没有成功，因为这一步需要使用蓝牙接收器来连接上测温枪接收数据并分析，但是我的电脑当蓝牙接收器连接上的时候没办法识别COM口，所以就没办法搜索设备。：



蓝牙接收器adapter



代码运行失败截图

这个部分属于硬件问题，可能出问题的地方是这个蓝牙接收器、或者是windows下这个代码的问题，因为我现在没有Linux系统的环境，所以这个问题暂时是无解的。

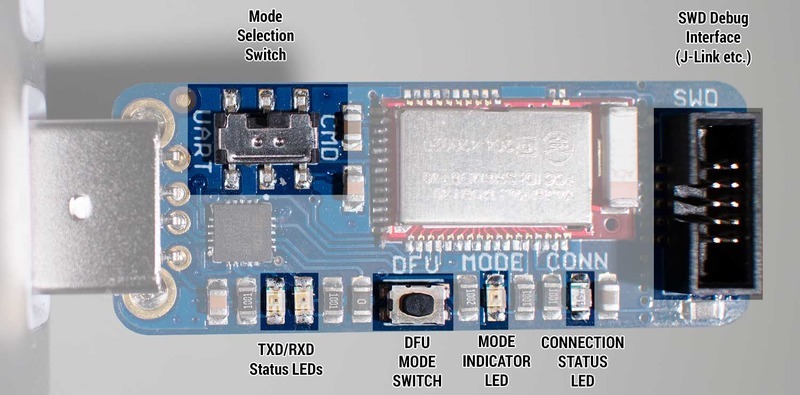
#### 3.3.2发送

接下来我们在Linux上做一下这些动作，下面的操作使用了LE friend ，因为按照官方的教程，直接使用LE friend也是可以的，所以我想试一试。

1. LE friend介绍

这是LE friend的硬件示意图，从上面我们可以清楚的看到各个指示灯的含义

后面的那个口（即SWD Debug）我们不需要用，本身硬件也没有，所以不用管，DFU是用于烧写LE friend固件的，也不用管。



Mode Selection Switch

可以在“ CMD”（命令模式）和“ UART”（数据模式）之间切换此开关，这将改变设备在终端仿真器中的行为方式。

TXD/RXD Status LEDs

提供这两个LED主要用于调试目的，以帮助您可视化USB CDC接口上的传入和传出字符。Mode Indicator LED

该指示灯用于指示设备当前正在运行的模式（数据，命令或DFU）。

Connection Status LED

当BLEFriend已成功与另一个BLE设备建立连接时，此LED将启用，并且对于调试目的很有用。

1. 操作模式

数据模式

数据模式使用BLE UART服务，并将BLEFriend变成BLE中央设备（您的手机或平板电脑）与您的PC或支持USB的设备之间的HW UART桥接器。 要使用数据模式，只需将BLEFriend模块连接到PC的USB端口，将模式选择开关设置为UART，然后使用您喜欢的终端软件以9600 bps的速度开始发送或接收数据。

如果MODE LED闪烁两次，然后延迟三秒钟，则您处于数据模式：

（视频演示了如何亮灯）

命令模式用于将配置命令发送到模块或检索有关模块本身或BLE连接另一端连接的设备的信息。 要使用命令模式，请确保将模式选择开关设置为CMD，然后使用您喜欢的终端仿真器以9600 bps的速度输入有效的Hayes AT样式命令（例如，使用``ATI''显示有关该模块的一些基本信息）。 如果MODE LED闪烁三下，然后延迟三秒钟，则您处于命令模式：



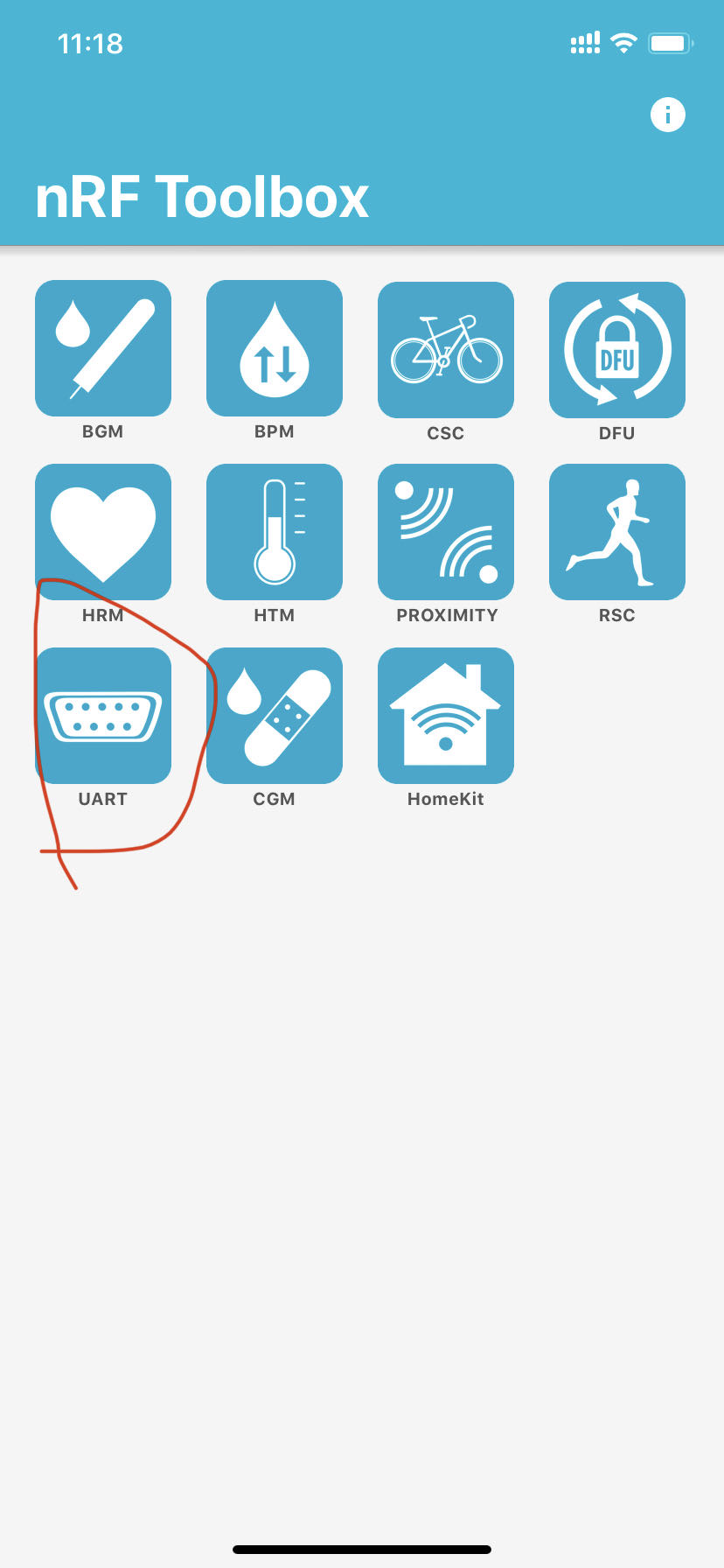
1. 测试操作（这一步只是进行测试，和我们最后做的没关系，所以这些工具可以只是作为了解或是写在论文中，不必强求能够运行）

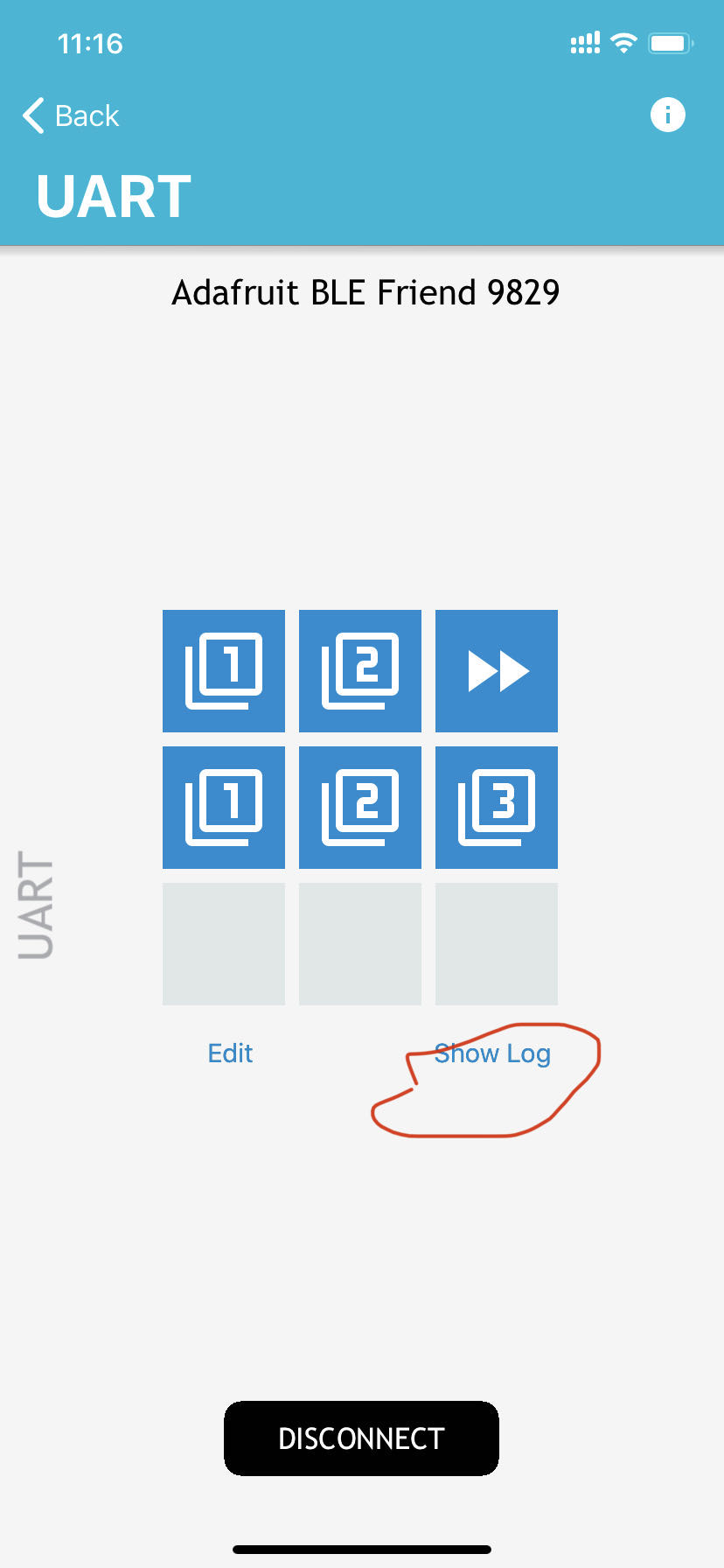
首先需要在电脑端安装CoolTerm，这是一个免费且功能齐全的终端仿真器软件包。<http://freeware.the-meiers.org/>

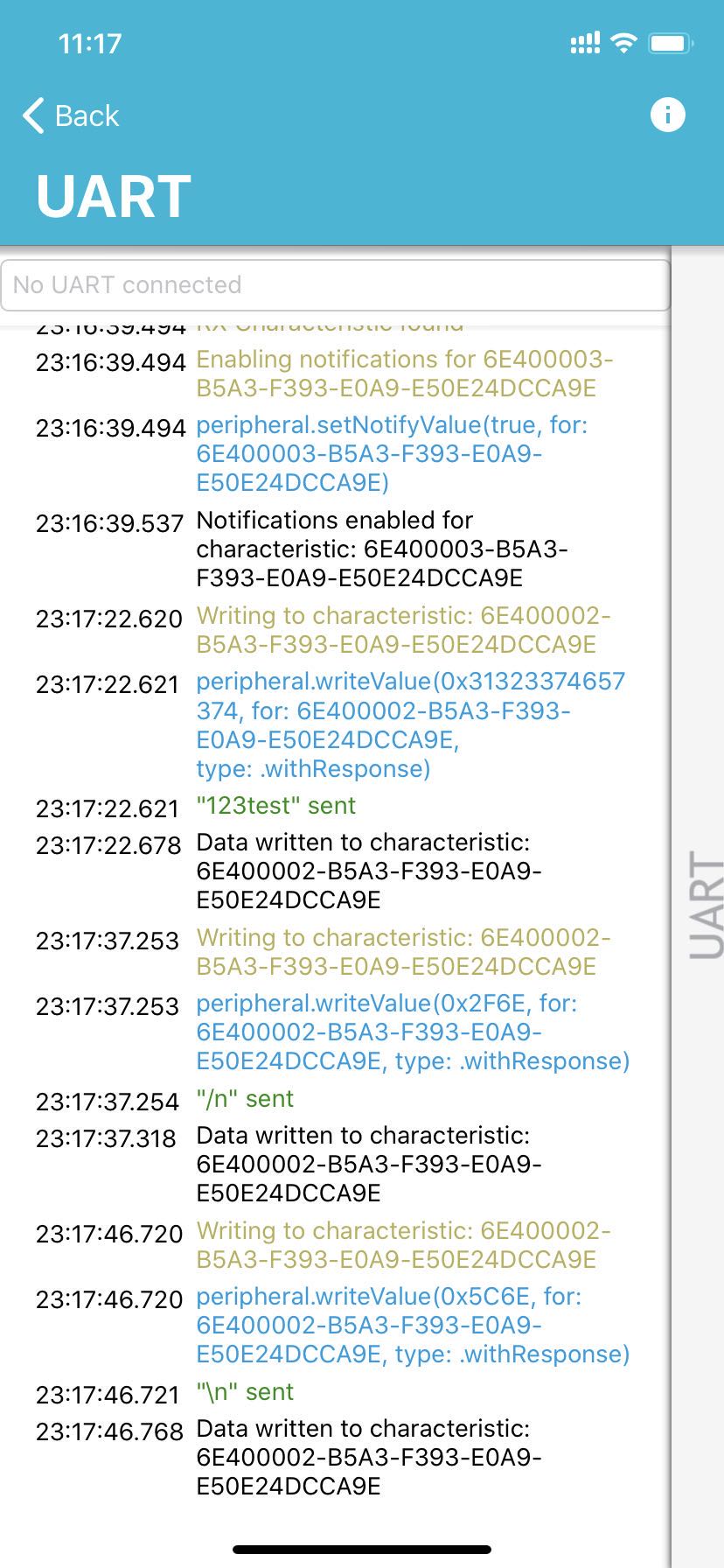
这里可以进行下载

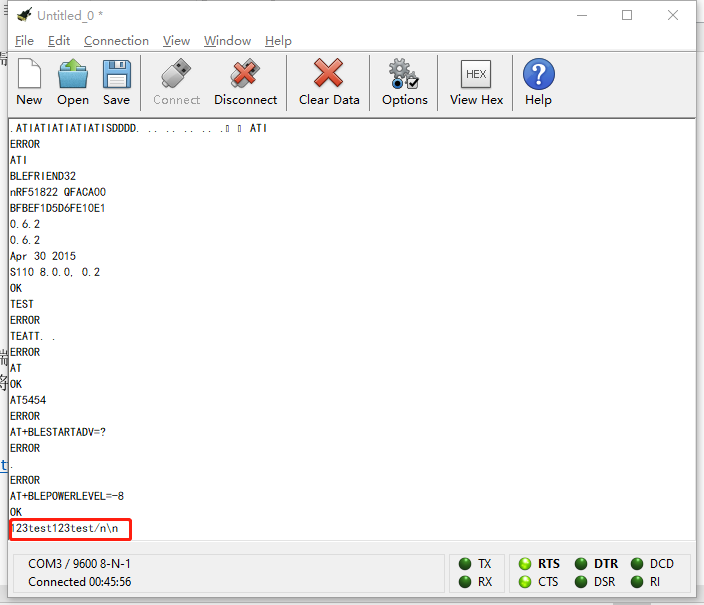
手机端需要安装

然后将LE friend设置为数据模式，手机端发送的数据就能在电脑端看到了：









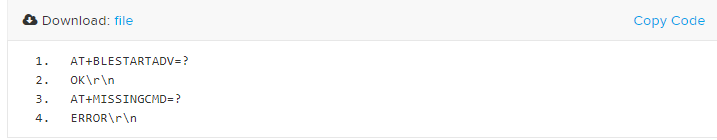
将LE friend调至cmd模式，输入命令可以看到上面的一些消息打印出来

**Hayes/AT Commands简介**

AT样式命令集的优点是它易于在机器间通信中使用，同时仍然对用户友好。

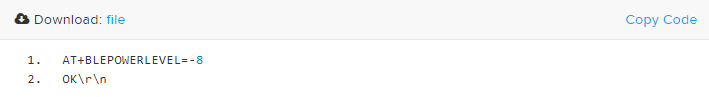
Test Command Mode '=?'

“测试”模式用于检查系统上是否存在指定的命令。 某些固件版本或配置可能包含也可能不包含特定命令，您可以通过使用命令名称并附加'=？'来确定该命令是否存在。如下图所示：



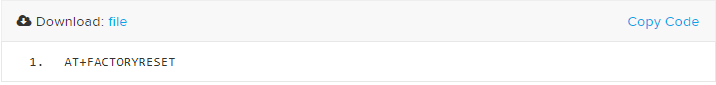
Write Command Mode '=xxx'

“写入”模式用于为命令分配特定的值，例如使用我们上面使用的命令来更改无线电的发射功率电平。 要向命令中写入值，只需在命令后附加一个'='符号，然后加上您要写入的任何参数（除了一个单独的'？'字符，它将被解释为tet模式）：



Execute Mode

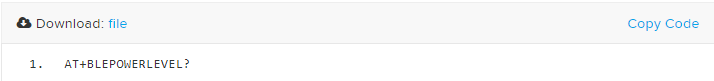
“执行”模式将运行合法的特定命令，并且在输入命令名称而没有其他参数时将使用该模式。



您可以使用执行模式来执行出厂重置，就像上面那样通过执行AT + FACTORYRESET命令来实现。

Read Command Mode '?'

“读取”模式用于读取命令的当前值。 并非每个命令都支持读取模式，但是通常您可以通过添加'？'来使用它来检索信息，例如无线电的当前发射功率水平。如下所示：



1. 模拟蓝牙

通过上面的测试我们能够知道，将LE FRIEND设置为cmd模式，我们就可以通过输入命令的方式模拟一个蓝牙设备。

这些代码是通过AT+命令的形式来将LE friend模拟为一个心率设备的，下面是代码的详细信息：

# Perform a factory reset to make sure we get a clean start

AT+FACTORYRESET

OK

# Add the Heart Rate service entry

AT+GATTADDSERVICE=UUID=0x180D

1

OK

# Add the Heart Rate Measurement characteristic

AT+GATTADDCHAR=UUID=0x2A37, PROPERTIES=0x10, MIN\_LEN=2, MAX\_LEN=3, VALUE=00-40

1

OK

# Add the Body Sensor Location characteristic

AT+GATTADDCHAR=UUID=0x2A38, PROPERTIES=0x02, MIN\_LEN=1, VALUE=3

2

OK

# Create a custom advertising packet that includes the Heart Rate service UUID

AT+GAPSETADVDATA=02-01-06-05-02-0d-18-0a-18

OK

# Reset the device to start advertising with the custom payload

ATZ

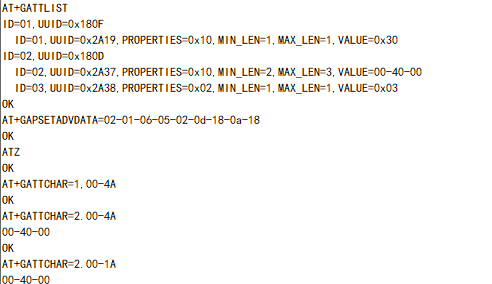
OK

# Update the value of the heart rate measurement (set it to 0x004A)

AT+GATTCHAR=1,00-4A

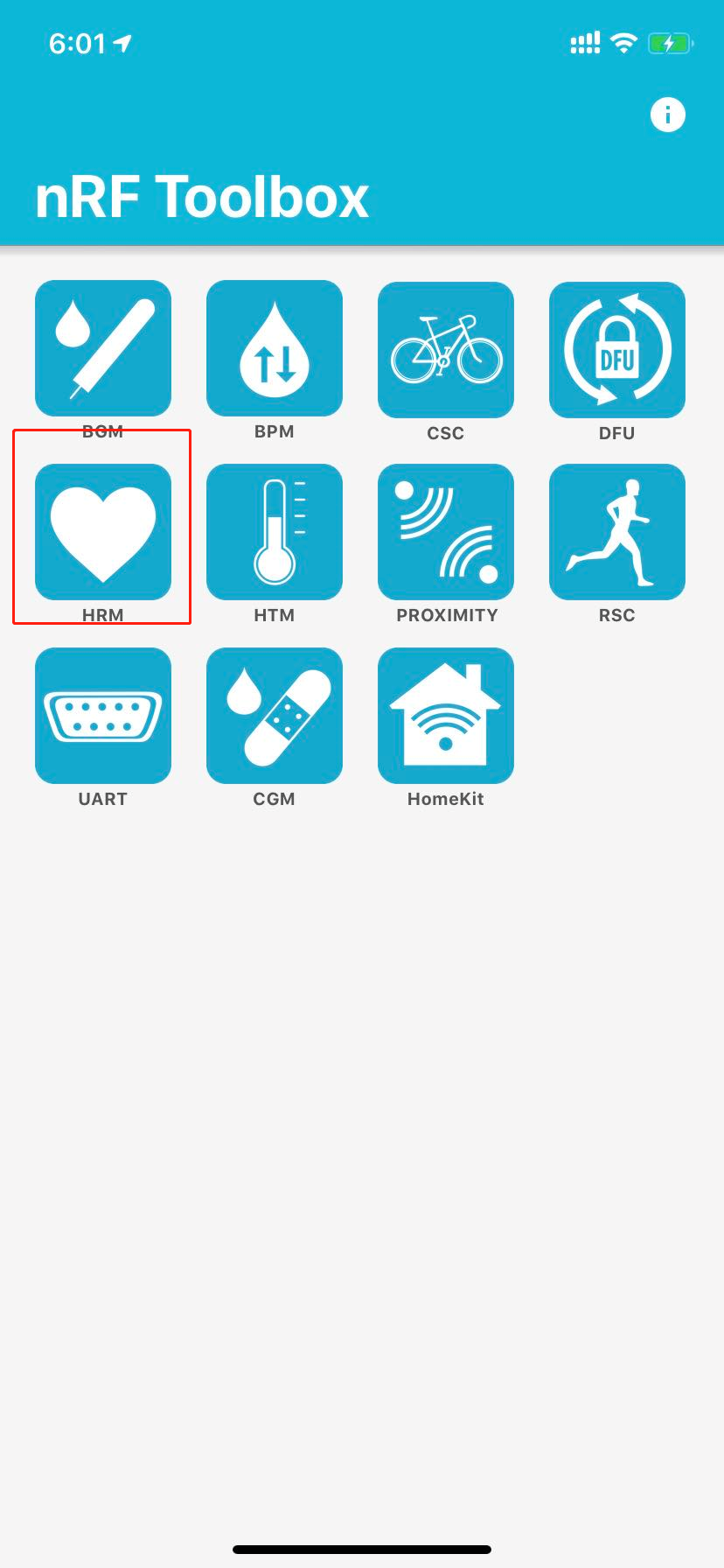
OK

这是我在CoolTerm上输入的部分指令：

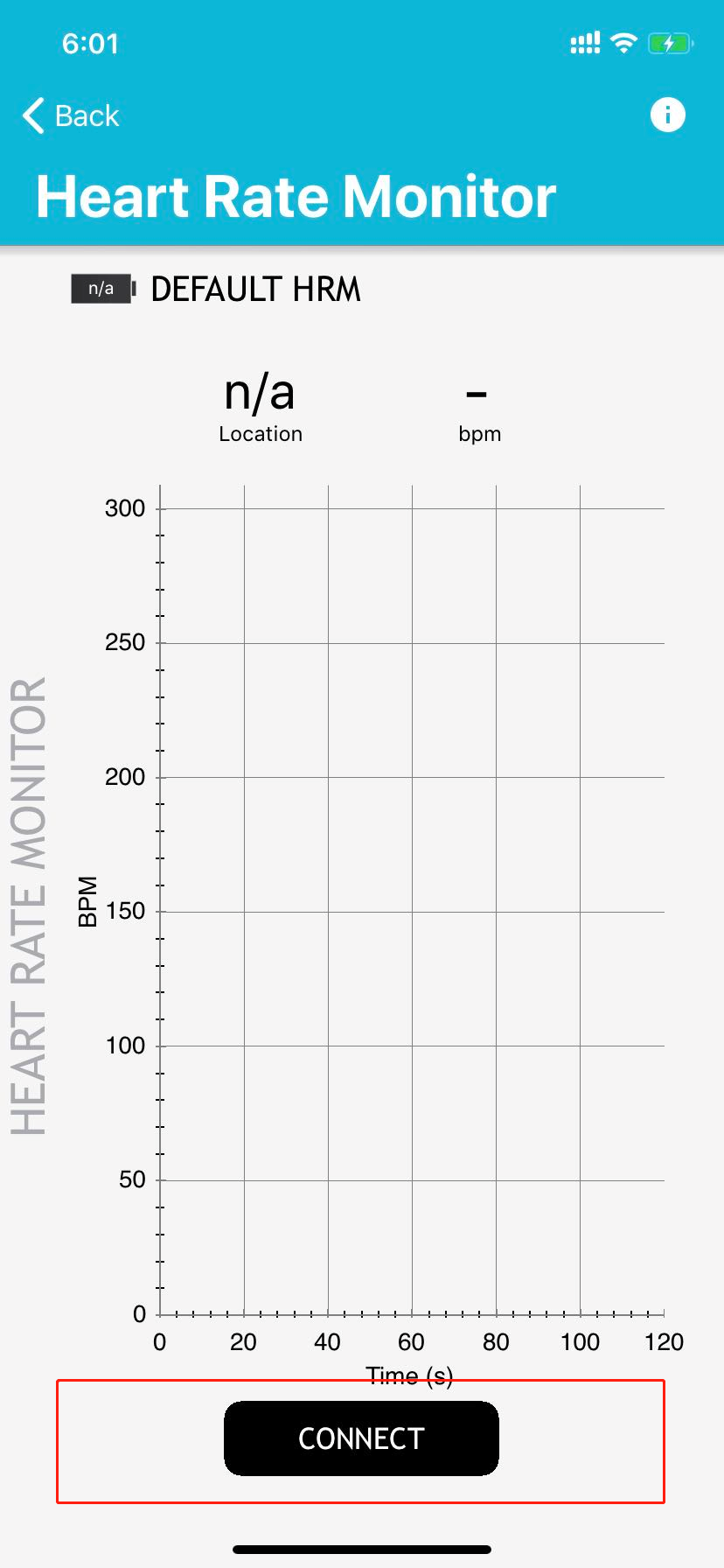


这是运行的效果图：

首先打开软件，进入心率测量的选项：



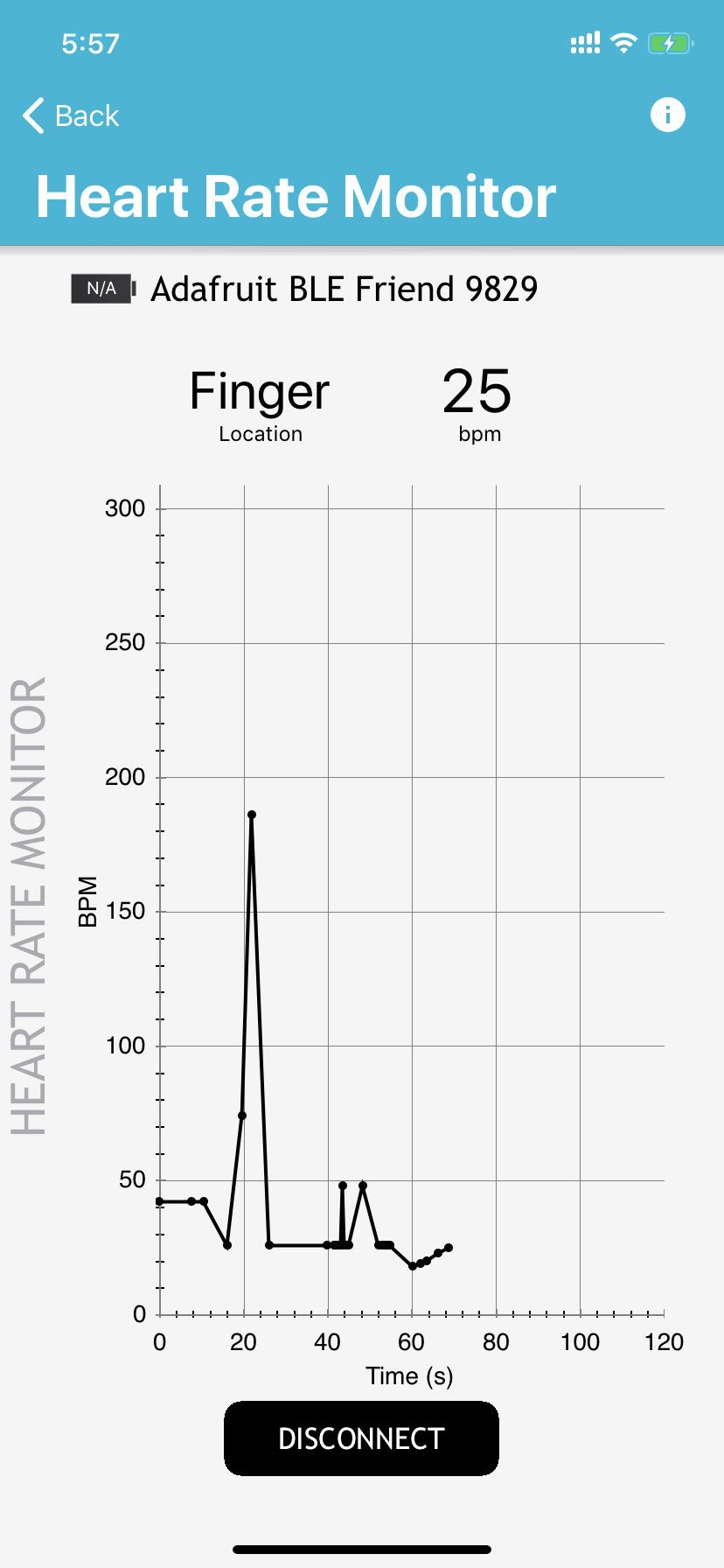
点进去后点connect：



找到你的LE friend，连接上：（这个名字也是可以改的）



输入上面的命令之后就可以看到效果了：



通过修改服务类型和广播包，我们很容易就能模拟一个温度计：(这里的东西就是实验步骤和结果，也就是说发端我现在已经成功的模拟出一个温度计了，但是由于我还不知道测温枪和测温枪app之间的数据交换过程，所以这个地方的假的设备只能被像你给我的那个文章中的BLEscanner或者我用的nrf Toolbox这样的app搜索到，我之前觉得这个不是最终的结果可以不放，但是我看了下你给我的文章，他其实也把这个部分的东西放进去了，所以我们可以先放这个进文章，要是你实在需要交论文需要一个最终的截图你可以将正常的测温枪连接到app的图放到里面。然后后面再补软件截图什么的。

然后那个文章里的工具所用到的原理都是一样的，都是根据收到的测温枪的数据复制过来，然后在设备里根据数据伪造一个假设备，因为设备的功能就是发送蓝牙广播包嘛，所以我就是在LE friend里写了很多命令，规定了发送广播包的格式，然后那个文章就是直接在电脑上将收到的广播包复制再发送出去，就是相当于我们这个是手动写的命令，然后因为它app可能有一些连接的包我没写进去或者没写完，所以现在还没法连接app，但是理论上是可行的，所以你先用着这个，然后后面直接我们就试试文章中的这个行不行，如果可以的话就用他的工具，但是我这个你也可以加的，我觉得都没问题的)

命令解析：

（下面用到的UUID的含义在这篇文章中有讲，这个是一个规定，如果你要引用的话最好参考官方网站，这个只是开发的一个参考：<https://blog.csdn.net/qq_15003505/article/details/75315049>）

将LE friend的参数重置

AT+FACTORYRESET

OK

# 添加一个温度计服务

AT+GATTADDSERVICE=UUID=0x1809

1

OK

# 添加一个温度测量特性

AT+GATTADDCHAR=UUID=0x2A1C, PROPERTIES=0x20, MIN\_LEN=2, MAX\_LEN=3, VALUE=00-40

1

OK

# 添加温度测量类型特性

AT+GATTADDCHAR=UUID=0x2A1D, PROPERTIES=0x02, MIN\_LEN=1, VALUE=00-02

2

OK

# 添加测量温度间隔的特性

AT+GATTADDCHAR=UUID=0x2A21, PROPERTIES=0x20, MIN\_LEN=1, VALUE=00-00

# 创建一个广播包

AT+GAPSETADVDATA=02-01-06-05-02-09-18-0a-18

OK

# 重启LE friend 服务

ATZ

OK

# 根据刚才设定的特性发送你想要发送的值

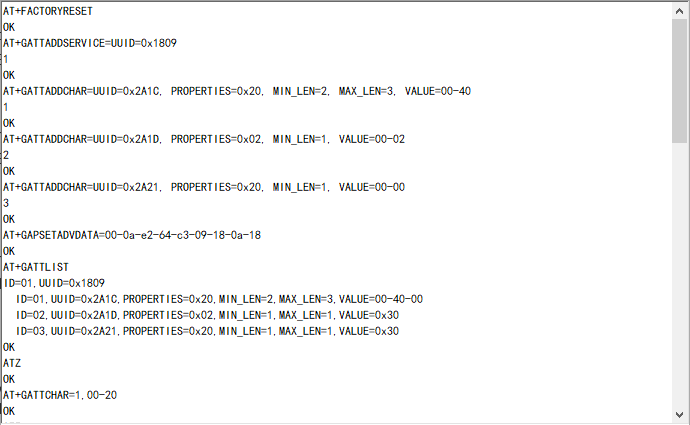
AT+GATTCHAR=1,00-2A

OK

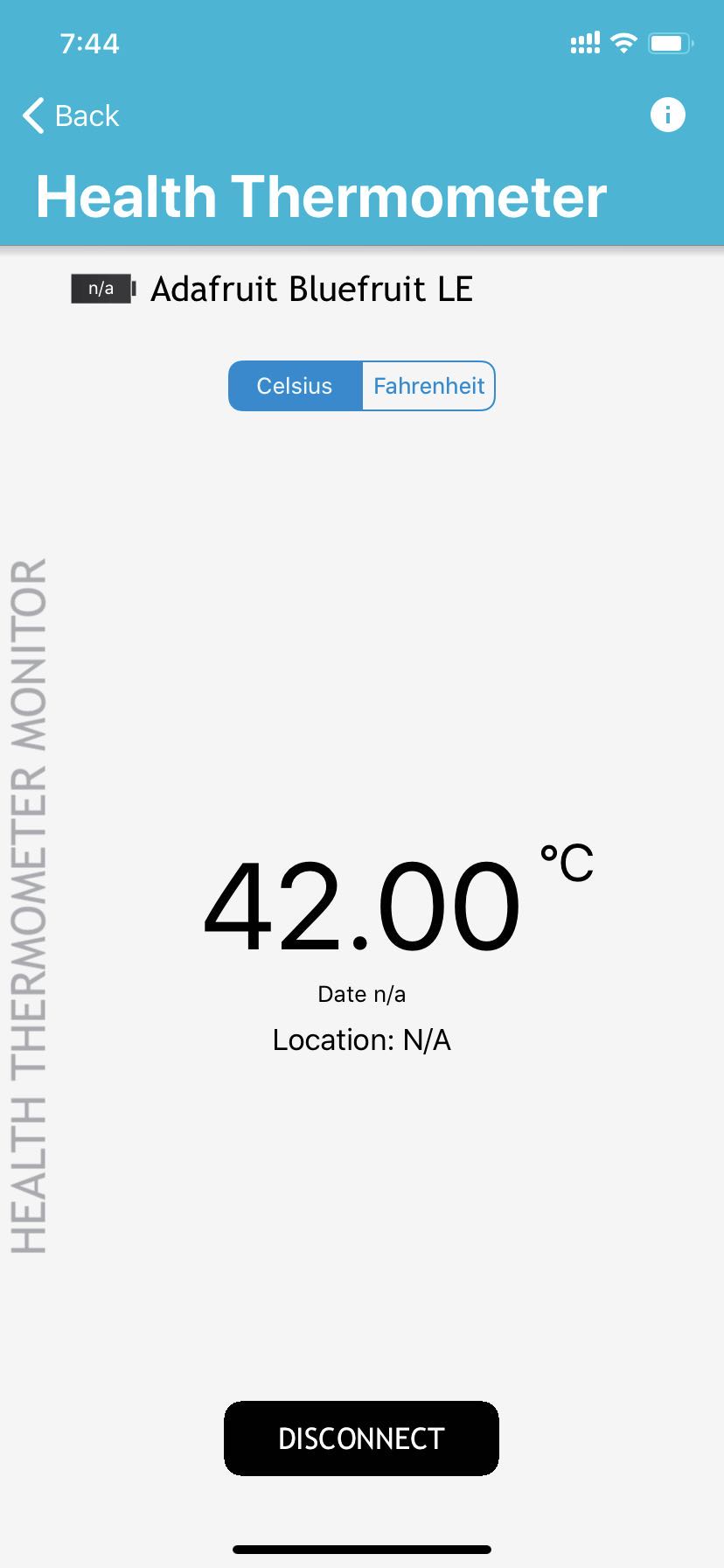
（关于这些命令的具体设定，可以参考这个网址，上面的东西可以直接复制

<https://learn.adafruit.com/introducing-adafruit-ble-bluetooth-low-energy-friend/ble-gatt>）

在coolterm中输入这些命令：



这是运行的截图：



#### 3.3.3使用gattacker

在正式开始之前，我们必须具备以下条件：

**硬件**

* 笔记本电脑
* BLE Dongles

**软件**

* Gattacker

****安装Gattacker并对其进行配置：****

**执行步骤**

**Step 1**：安装Gattacker，你需要最新版本的node和npm。可以使用以下命令安装：

sudo apt-**get** install bluetooth bluez libbluetooth-dev libudev-dev

**Step 2**：接下来，我们需要安装bleno，命令如下：

npm install bleno

npm install noble

如果出现错误，请确保你之前已正确安装了node和npm软件包

**Step 3**：现在使用该命令安装Gattacker

npm install gattacker

**Step 4**：在另一台虚拟机（或系统）上重复相同的步骤，因为我们需要两台机器一台用于主机，另一台用于从机。（这个地方我是克隆了一台虚拟机，但是好像不是必须要有两台虚拟机，但是两台会好一些）

**Step 5**：完成后，插入ble适配器并使用sudo hciconfig确保其已插入。

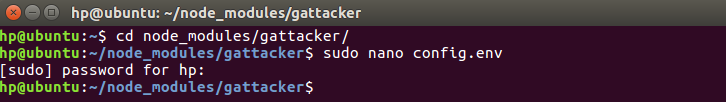
然后导航到gattacker文件夹

cd node\_modules/gattacker

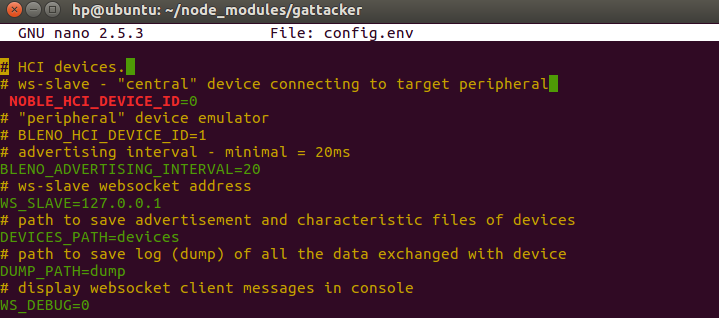
**Step 6**：

接下来，我们需要编辑config.env文件，配置gattacker。

**sudo** **nano** **config**.env



**Step 7**：取消**NOBLE\_HCI\_DEVICE\_ID**注释，然后将其替换为hciX，其中（X是我们之前通过hciconfig找到的值）并保存文件



**Step 8**：现在在主机中，插入BLE适配器并按照上述步骤操作。对于config.env，请按照以下步骤操作：

* 取消**NOBLE\_HCI\_DEVICE\_ID**注释
* 取消**BLENO\_HCI\_DEVICE\_ID**注释

将它们分配给hciX值。

完成后，在WS\_SLAVE中，将其中的IP地址替换成从机的地址：

使用Gattacker扫描和存储设备信息：

**执行步骤**

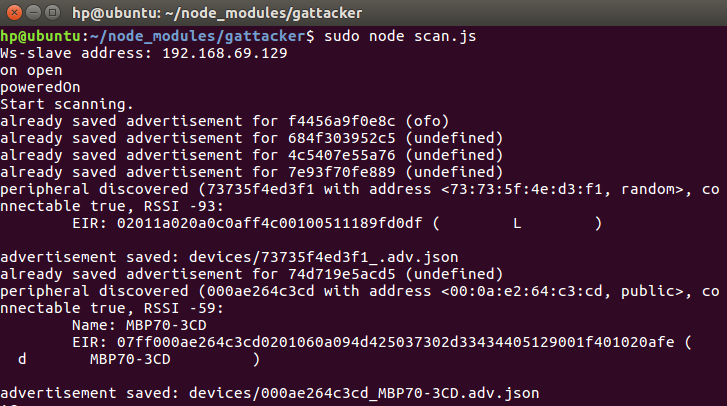
**Step 1**：打开从机VM并启动ws-slave.js，如下所示：

**sudo** **node** **ws-slave**.js

**Step 2**：现在在主机分支的gattacker文件夹下并启动扫描，如下所示

**sudo** **node** **scan**.js

现在打开测温枪，然后按Enter键以查看如下消息：



sudo node scan 74d719e5acd5其中74d719e5acd5是由gattacker保存的测温枪的peripheral名称。

使用linux系统

我们在这里想要使用bluez平台，接下来我们将纤细介绍这个工具，它包含了许多实用的蓝牙工具并且便于使用。不幸的是，没有跨Windows，Mac，Linux等平台的跨平台蓝牙协议栈或API。因此，如果想使用其他平台，则需要查看该平台的蓝牙低能耗协议栈和API。

Bluez

BlueZ是Linux官方蓝牙协议栈。它是一个基于GNU General Public License (GPL)发布的开源项目，从Linux2.4.6开始便成为Linux 内核的一部分。其基础代码均是由就职于Qualcomm(高通)的Maxim Krasnyansky完成的。包括：HCI，L2CAP，RFCOMM和基本socket的实现。Marcel Holtmann开发层的协议和应用，包括：BNEP, CMTP等。当然，这些中也有Maxim Krasnyansky的参预。有部分代码由Nokia提供的。

1. BlueZ支持蓝牙核心层和协议，它灵活、高效，以模块化方式实现，具有以下特点：

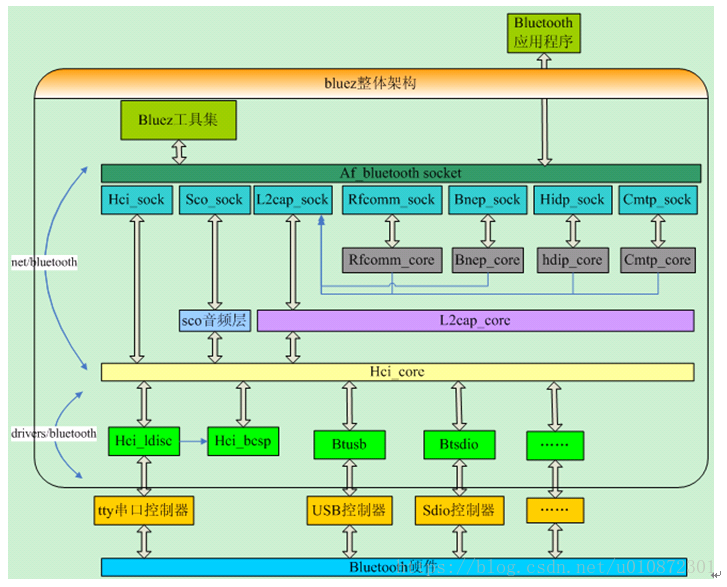
* 完整的模块化实现
* 均衡的多处理安全
* 支持多线程数据处理
* 支持多个蓝牙设备
* 硬件抽象
* 提供所有层的标准socket接口
* 支持设备级和服务级安全保证

1. BlueＺ包含多个相互独立的模块：

* Linux内核蓝牙子系统核心
* L2CAP 和 SCO 音频内核层
* RFCOMM, BNEP, CMTP 和 HIDP内核实现
* HCI UART, USB, PCMCIA 和虚拟设备驱动
* 通用的蓝牙和SDP库及守护进程
* 配置和测试小工具
* 协议解析和分析工具

我们主要使用了上面标红的这些模块

下图是bluez的一个架构示意图：



Bluez的安装

在ubuntu14.04 下编译安装Bluez-5.52

1、安装相关依赖库：

sudo apt-get install libusb-dev libdbus-1-dev libglib2.0-dev automake libudev-dev libical-dev libreadline-dev

2、下载Bluez-5.28压缩包并解压:

wget http://www.kernel.org/pub/linux/bluetooth/bluez-5.52.tar.xz

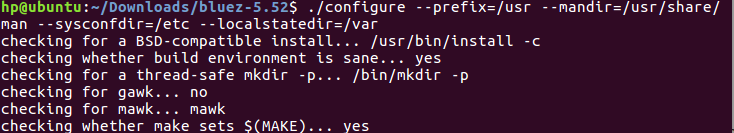
tar xvf bluez-5.52.tar.xz

进入解压后的目录：

cd bluez-5.52

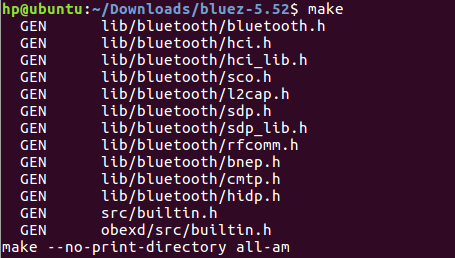
3、执行配置文件

./configure --prefix=/usr --mandir=/usr/share/man --sysconfdir=/etc --localstatedir=/var

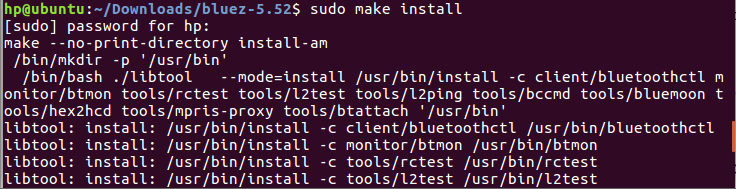


4、执行make和install

make



sudo make install



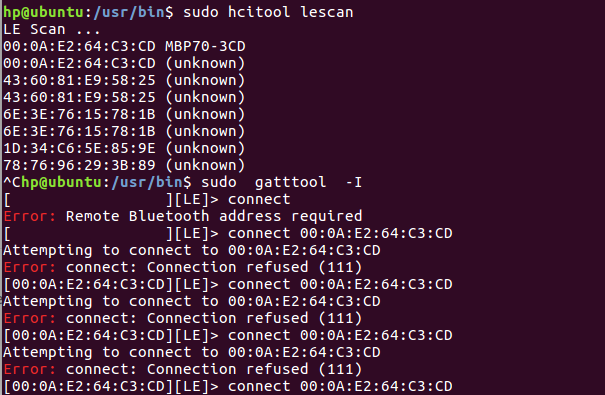
没有提示任何错误的话，安装完毕。

Bluez的使用

1. lescan扫描低功耗蓝牙设备

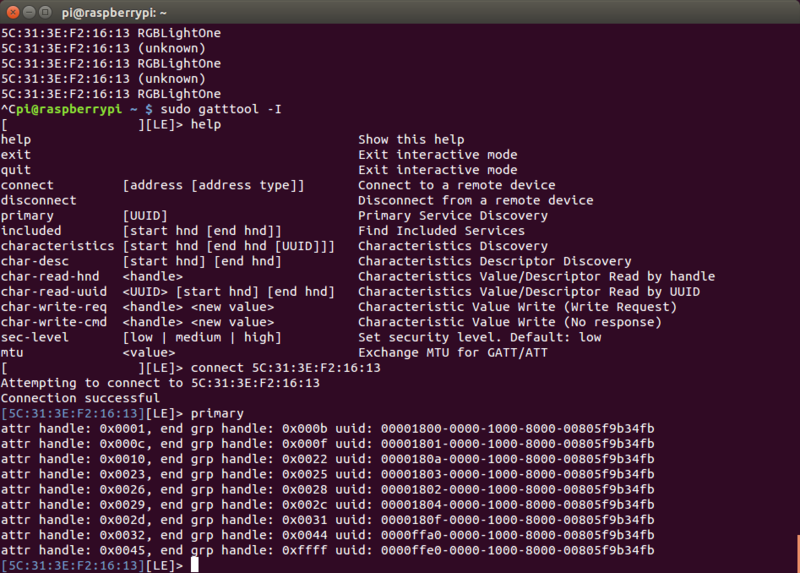


1. connect   //连接低功耗设备



1. primary

在连接成功后输入这个命令，可以看到该低功耗蓝牙设备能够支持的服务，和uuid



1. char-desc 0x0028 0x0028

可以运行的另一个命令是char-desc命令，以获取有关特定特征的详细信息。我运行以下命令来查询0x0028特性：

