Porting Realtek Bluetooth driver and Bluez into Linux OS Guide

Date: 2015/02/06

Version: 4.0

目录

1 概述

这篇文档将介绍如何在 linux 系统中支持 Realtek 蓝牙模块，主要是在系统中移植 bluez 以及蓝牙驱动的说明。

2 平台信息确认

Realtek 有不同的蓝牙模块方案，首先需要确认模块的特性是否和客户平台相匹配，主要有以下几个方面。

2.1 模块接口及 efuse 设定

Realtek 蓝牙模块接口有 USB/UART/PCM，对于 uart 接口可以配置成 H4/H5。同时可以支持单天线/双天线，这些设定都是在 efuse 中固定的，需要确认模块的设置是否正确。另外有些配置是可以通过 driver 里面的 config 文件来修改，如串口波特率，PCM 参数，32k clock 配置等，详细设定可以参考《**UART interface BT controller initial guide**》。

2.2 安装驱动

如果客户使用的系统已经安装了 Bluez，并且 kernel 关于 net/bluetooth 的选项已经打开，那么可以通过 command line 方式直接安装我们的驱动，详细指令参考 driver 内部的 readme 说明，本文档以下说明可以略过。

2.3 移植 Bluez

如果客户系统并没有移植过蓝 bluez lib 或者需要更新 bluez 版本，首先需要从官网下载需要的版本，然后可以把我们提供的 tool 及 driver 直接修改到 bluez 及 kernel source code，然后编译生成 image。

2.4 开发支持其他 stack 的 driver

如果客户希望使用 bluez 之外的其他 stack，那么需要自行开发驱动，本文档以下说明不适用，请参考文件《**UART interface BT controller initial guide**》.

3 Kernel 设定

3.1 bluez 协议栈设定

通过“make menuconfig”打开 net/bluetooth 的选项设定，也可以直接修改平台的 config 文件，确认 kernel 配置中蓝牙有支持，这部分对应的 code 在 **kernel/net/bluetooth** 目录。

```
CONFIG_BT=y
```

3.2 UART 驱动设定

这部分是 UART 接口驱动，使用 usb 接口可以直接略过，参考 3.3 USB 驱动设定。

将 Realtek BT UART driver 包中 bluetooth_uart_driver 文件夹内的.h 和.c 文件都 copy 至 **kernel/drivers/bluetooth** 目录。

修改 **kernel/drivers/bluetooth** 目录下 Kconfig 和 Makefile 文件增加对 Realtek H5 的支持。

```
//add RTKH5 in Kconfig
config BT_HCIUART_RTKH5
    bool "Realtek H5 protocol support"
    depends on BT_HCIUART
    help
      Realtek H5 is serial protocol for communication
      between Realtek Bluetooth device and host. This protocol is required for
      Realtek uart h5 bluetooth controller

      Say Y here to compile support for Realtek HCI H5 protocol.

//add RTKH5 in Makefile
hci_uart-$(CONFIG_BT_HCIUART_H4)    += rtk_coex.o

hci_uart-$(CONFIG_BT_HCIUART_RTKH5) += hci_rtk_h5.o
hci_uart-$(CONFIG_BT_HCIUART_RTKH5) += rtk_coex.o
```

通过“make menuconfig”或者直接修改 kernel 的配置文件，确认 RTK H5 driver 有支持。

```
CONFIG_BT_HCIUART=y
CONFIG_BT_HCIUART_H4 =y
CONFIG_BT_HCIUART_RTKH5=y
```

3.3 USB 驱动设定

这部分是 USB 接口驱动，使用 UART 接口可以直接略过，参考 3.2 UART 驱动设定。

将 Realtek BT USB driver 包中 bluetooth_usb_driver 文件夹下所有.c 和.h 文件 copy 至 **kernel/driver/bluetooth** 目录。

修改 **kernel/driver/bluetooth 目录下** Kconfig 和 Makefile 增加对 Realtek BT USB 的支持。

```
//add RTK_BTUSB in Kconfig
config BT_RTKBTUSB
        tristate "RTK HCI USB driver"
        depends on USB
        help
              RTK Bluetooth HCI USB driver.

//add RTK_BTUSB in Makefile and remove generic driver
obj-$( CONFIG_BT_RTKBTUSB) += rtk_btusb.o
rtk_btusb-objs := rtk_bt.o rtk_coex.o
// obj-$(CONFIG_BT_HCIBTUSB)     += btusb.o
```

通过“make menuconfig”或者直接修改 kernel 的配置文件，确认 BT USB 有支持。

```
CONFIG_BT_RTKBTUSB=y
```


3.4 AVRCP support

通过“make menuconfig”或者直接修改 kernel 的配置文件。

```
CONFIG_INPUT_UINPUT=y      # User level driver support
CONFIG_INPUT_MISC=y
```

4 Bluez 移植

4.1 Bluez 编译

从 bluez 官网（www.bluez.org）下载 bluez source code 进行编译移植，编译过程中还需要移植 bluez 依赖的库，如 dbus，glib 等。如果不需要支持 LE 功能，推荐使用 **bluez4.101**；如果需要支持 LE 功能，要求 **kernel 版本 3.10 及以上，bluez5.0 及以上**。RTK BT driver 支持 kernel 版本 2.6.32 以上。

bluez 的编译可以参考 bluez/README，首先通过 configure 进行配置，然后编译。可以通过 configure 增减一些功能，例如：--enable-test 可以编译 bluez/test 下的一些 test tool；对于 bluez5.0+，有些平台需要添加--disable-systemd 以避免出现 configure error，添加 --enable-experimental 增加 heartrate 等 plugin，具体可以看 Makefile.plugins 文件，含有 **if EXPERIMENTAL** 的都受其控制，另外对于 bluez 5.0+，需要编译出 bluetoothctl，用于测试，编译参数--enable-client。

bluez 中有两个重要的地址与 configure 配置有关：**CONFIGDIR**、**STORAGEDIR**。CONFIGDIR 是 bluez 中 config 文件的存放位置，STORAGEDIR 是 bluez 存放每一个 adapter 相关信息的位置。

```
->>>if (test "$sysconfdir" = '${prefix}/etc'); then
->>>->>>configdir="${prefix}/etc/bluetooth"
->>>else
->>>->>>configdir="${sysconfdir}/bluetooth"
->>>fi

->>>if (test "$localstatedir" = '${prefix}/var'); then
->>>->>>storagedir="${prefix}/var/lib/bluetooth"
->>>else
->>>->>>storagedir="${localstatedir}/lib/bluetooth"
->>>fi
```

若按照 README 设置 (./configure --prefix=/usr --mandir=/usr/share/man --sysconfdir=/etc --localstatedir=/var --libexecdir=/lib)，CONFIGDIR 就为/etc/bluetooth，STORAGEDIR 就为/var/lib/bluetooth。bluez 启动后会在 CONFIGDIR 中查找 main.conf、audio.conf 等文件进行配置，main.conf 中可以设置 name，class 等信息，bluez4.101 还会有 audio.conf 可以对 a2dp/hfp 进行配置，移植后要将这些 conf 文件放在对应的目录下。

bluez4.101 会为每一个 adapter 在 STORAGEDIR 下以 address 创建文件夹，里面会包含 config、linkkeys 等内容，config 会存储 name 等信息，linkkeys 会保存和这个 adapter 配对的 link key。

```
root@kyle-OptiPlex-380:/var/lib/bluetooth# ls
00:E0:4C:23:D7:C3
root@kyle-OptiPlex-380:/var/lib/bluetooth# cd 00\:E0\:4C\:23\:D7\:C3/
root@kyle-OptiPlex-380:/var/lib/bluetooth/00:E0:4C:23:D7:C3# ls
classes  config  did  eir  lastseen  lastused  linkkeys  names  profiles  sdp  trusts
```

bluez5.0+在 STORAGEDIR 下 adapter address 文件夹内会为每一个 remote device 以 address 创建文件夹，其中的 info 文件会存储该 device 的相关信息，如对于 LE device，会有 remote signature key、local signature key、LTK 等。

```
root@kyle-OptiPlex-380:/var/lib/bluetooth/00:E0:D1:2D:D8:D8# ls
78:C5:E5:73:54:82  94:DB:C9:1D:9B:EA  cache  settings
root@kyle-OptiPlex-380:/var/lib/bluetooth/00:E0:D1:2D:D8:D8# cat 78\:C5\:E5\:73\:54\:82/info
[RemoteSignatureKey]
Key=2279E998754E78A71821E68915CBF64F

[LocalSignatureKey]
Key=E982747E3ABC629712B170CFBD88BECB

[LongTermKey]
Key=BF555432123C060CE383E58BE58B850B
Authenticated=0
EncSize=16
EDiv=11107
Rand=16378185036631317833

[SlaveLongTermKey]
Key=C4FEDC481664C16B14B54677DA496D50
Authenticated=0
EncSize=16
EDiv=42963
Rand=5478683869407214074

[General]
Name=BLE Chest Strap
AddressType=public
SupportedTechnologies=LE;
Trusted=true
Blocked=false
Services=00001800-0000-1000-8000-00805f9b34fb;00001801-0000-1000-8000-00805f9b34fb;0000180a-0000-1000-8000-00805f9b34fb;0000180d-0000-
1000-8000-00805f9b34fb;0000180f-0000-1000-8000-00805f9b34fb;

[ConnectionParameters]
MinInterval=400
MaxInterval=800
Latency=0
Timeout=600
```

4.2 hciattach tool

hciattach tool 是 bluez 为 UART 接口 BT 提供的初始化工具，对于 USB 接口的蓝牙模块直接跳过这部分的设置。

对于 Realtek UART BT，请使用 driver 包 rtk_hciattach 目录下文件编译生成的 rtk_hciattach，不要使用 bluez 编出的 hciattach。初始化时通过 rtk_hciattach -n -s 115200 ttyS1 rtk_h5 或 rtk_hciattach -n -s 115200 ttyS1 rtk_h4 命令执行，其中串口号在各平台会有不同。

4.3 加载固件文件

蓝牙模块初始化过程需要加载固件文件，对于 UART 接口的可在 hciattach_rtk.c 中自行定义固件文件所在的目录。

```
#define FIRMWARE_DIRECTORY "/system/etc/firmware/rtlbt/"
#define BT_CONFIG_DIRECTORY "/system/etc/firmware/rtlbt/"
```

```
sprintf(firmware_file_name, FIRMWARE_DIRECTORY"rtlbt_fw");
sprintf(bt_config_file_name, BT_CONFIG_DIRECTORY"rtlbt_config");
```

对于 USB 接口，driver 中通过 request_firmware 这个系统函数获取固件，这个函数查找文件的目录是与平台相关的，一般在/lib/firmware 目录，也有平台可能在/user/firmware，需要客户确认所使用的平台对应的位置并将固件文件 copy 至目录。

4.4 bluez 启动

在使用蓝牙之前需要完成以下几步配置：

1、确认 bluetooth.conf 已放置在/etc/dbus-1/system.d 目录下，启动 bluez 进程：bluetoothd -n -d；对于 Bluez 5，可以添加-C 参数，提供 deprecated command line interfaces，如 sdptool browse local。

如果 bluetoothd 启动失败，出现以下 log：

D-Bus setup failed: Connection ":1.12" is not allowed to own the service "org.bluez" due to security policies in the configuration file

则可能是 dbus 权限的问题，可以在 /etc/dbus-1/system.d/bluetooth.conf 中添加

```
<policy user="root">
    <allow own="org.bluez"/>
    <allow send_destination="org.bluez"/>
    <allow send_interface="org.bluez.Agent1"/>
    <allow send_interface="org.bluez.MediaEndpoint1"/>
    <allow send_interface="org.bluez.MediaPlayer1"/>
    <allow send_interface="org.bluez.ThermometerWatcher1"/>
    <allow send_interface="org.bluez.AlertAgent1"/>
    <allow send_interface="org.bluez.Profile1"/>
    <allow send_interface="org.bluez.HeartRateWatcher1"/>
    <allow send_interface="org.bluez.CyclingSpeedWatcher1"/>
    <allow send_interface="org.bluez.GattCharacteristic1"/>
    <allow send_interface="org.bluez.GattDescriptor1"/>
    <allow send_interface="org.freedesktop.DBus.ObjectManager"/>
    <allow send_interface="org.freedesktop.DBus.Properties"/>
    <!-- for bluez 4 -->
    <allow send_interface="org.bluez.Agent"/>
    <allow send_interface="org.bluez.HandsfreeAgent"/>
    <allow send_interface="org.bluez.MediaEndpoint"/>
    <allow send_interface="org.bluez.MediaPlayer"/>
    <allow send_interface="org.bluez.Watcher"/>
    <allow send_interface="org.bluez.ThermometerWatcher"/>
    <allow send_type="method_call"/>
</policy>
```

如果不是以 root 运行 bluetoothd，则需要添加相应 user 的 policy，和 root policy 类似，复制上面的内容，然后把第一行修改成<policy user=”xxx”>

不过请在 root 权限下运行 bluetoothd。

2、如果使用的是 uart 的卡片，需要将 BT DIS pin 拉 high 并通过 rtk_hciattach 完成初始化和配置：

rtk_hciattach -n -s 115200 ttyUSB0 rtk_h5；

3、将蓝牙带起来：hciconfig hci0 up。如果是 Bluez 5+，可以运行 bluetoothctl，然后输入 power on 命令。

4.5 bluez 测试及 debug

在 porting 完成之后可以使用 bluez 提供的一些 test tool 和 test script 进行测试。

4.5.1 test tools

bluez 有提供一些 tool 可以测试基本功能，具体代码在 bluez\tools 目录下，下面对其中一些常用 tool 进行简单介绍。

(a) hciconfig: HCI 配置工具

hciconfig hci0 up /*Open and initialize HCI device */

hciconfig hci0 iscan /*can be seen by other bluetooth device */

其它 hciconfig cmd 可用 hciconfig --help 获得或查看 tools/ hciconfig.c 文件。

```
} command[] = {
    { "up",     cmd_up,     0,      "Open and initialize HCI device" },
    { "down",   cmd_down,   0,      "Close HCI device" },
    { "reset",  cmd_reset,  0,      "Reset HCI device" },
    { "rstat",  cmd_rstat,  0,      "Reset statistic counters" },
    { "auth",   cmd_auth,   0,      "Enable Authentication" },
    { "noauth", cmd_auth,   0,      "Disable Authentication" },
    { "encrypt",    cmd_encrypt,    0,      "Enable Encryption" },
    { "noencrypt",  cmd_encrypt,    0,      "Disable Encryption" },
    { "piscan", cmd_scan,   0,      "Enable Page and Inquiry scan" },
    { "noscan", cmd_scan,   0,      "Disable scan" },
    { "iscan",  cmd_scan,   0,      "Enable Inquiry scan" },
    { "pscan",  cmd_scan,   0,      "Enable Page scan" },
    { "ptype",  cmd_ptype,  "[type]",   "Get/Set default packet type" },
    { "lm",     cmd_lm,     "[mode]",   "Get/Set default link mode"   },
    { "lp",     cmd_lp,     "[policy]", "Get/Set default link policy" },
    { "name",   cmd_name,   "[name]",   "Get/Set local name" },
    { "class",  cmd_class,  "[class]",  "Get/Set class of device" },
    { "voice",  cmd_voice,  "[voice]",  "Get/Set voice setting" },
    { "iac",    cmd_iac,    "[iac]",    "Get/Set inquiry access code" },
    { "inqtpl", cmd_inq_tpl,    "[level]",  "Get/Set inquiry transmit power level" },
    { "inqmode",    cmd_inq_mode,   "[mode]",   "Get/Set inquiry mode" },
    { "inqdata",    cmd_inq_data,   "[data]",   "Get/Set inquiry data" },
    { "inqtype",    cmd_inq_type,   "[type]",   "Get/Set inquiry scan type" },
    { "inqparms",   cmd_inq_parms,  "[win:int]",    "Get/Set inquiry scan window and interval" },
    { "pageparms",  cmd_page_parms, "[win:int]",    "Get/Set page scan window and interval" },
    { "pageto", cmd_page_to,    "[to]",     "Get/Set page timeout" },
    { "afhmode",    cmd_afh_mode,   "[mode]",   "Get/Set AFH mode" },
    { "sspmode",    cmd_ssp_mode,   "[mode]",   "Get/Set Simple Pairing Mode" },
    { "aclmtu", cmd_aclmtu, "<mtu:pkt>",    "Set ACL MTU and number of packets" },
    { "scomtu", cmd_scomtu, "<mtu:pkt>",    "Set SCO MTU and number of packets" },
    { "delkey", cmd_delkey, "<bdaddr>", "Delete link key from the device" },
    { "oobdata",    cmd_oob_data,   0,      "Get local OOB data" },
    { "commands",   cmd_commands,   0,      "Display supported commands" },
    { "features",   cmd_features,   0,      "Display device features" },
    { "version",    cmd_version,    0,      "Display version information" },
    { "revision",   cmd_revision,   0,      "Display revision information" },
    { "block",  cmd_block,  "<bdaddr>", "Add a device to the blacklist" },
    { "unblock",    cmd_unblock,    "<bdaddr>", "Remove a device from the blacklist" },
    { "lerandaddr", cmd_le_addr,    "<bdaddr>", "Set LE Random Address" },
    { "leadv",  cmd_le_adv, "[type]",   "Enable LE advertising"
      "\n\t\t\t0 - Connectable undirected advertising (default)"
      "\n\t\t\t3 - Non connectable undirected advertising"},
    { "noleadv",    cmd_no_le_adv,  0,      "Disable LE advertising" },
    { "lestates",   cmd_le_states,  0,      "Display the supported LE states" },
    { NULL, NULL, 0 }
```

(b) hcitool: HCI 工具

hcitool scan /*scan other bluetooth device*/.

hcitool cc xx:xx:xx:xx:xx:xx /*create a ACL connection to remote device, for it is not initiated by upper profiles , this connection will disconnect very soon */

hcitool lescan /*scan other le bluetooth device*/.

hcitool lewladd <address> /*add device to LE White List*/

其它 hcitool cmd 可用 hcitool --help 获得或查看 tools/hcitool.c 文件。

```
} command[] = {
    { "dev",      cmd_dev,     "Display local devices"                },
    { "inq",      cmd_inq,     "Inquire remote devices"               },
    { "scan",     cmd_scan,    "Scan for remote devices"              },
    { "name",     cmd_name,    "Get name from remote device"          },
    { "info",     cmd_info,    "Get information from remote device"   },
    { "spinq",    cmd_spinq,   "Start periodic inquiry"               },
    { "epinq",    cmd_epinq,   "Exit periodic inquiry"                },
    { "cmd",      cmd_cmd,     "Submit arbitrary HCI commands"        },
    { "con",      cmd_con,     "Display active connections"           },
    { "cc",       cmd_cc,      "Create connection to remote device"   },
    { "dc",       cmd_dc,      "Disconnect from remote device"        },
    { "sr",       cmd_sr,      "Switch master/slave role"             },
    { "cpt",      cmd_cpt,     "Change connection packet type"        },
    { "rssi",     cmd_rssi,    "Display connection RSSI"              },
    { "lq",       cmd_lq,      "Display link quality"                 },
    { "tpl",      cmd_tpl,     "Display transmit power level"         },
    { "afh",      cmd_afh,     "Display AFH channel map"              },
    { "lp",       cmd_lp,      "Set/display link policy settings"     },
    { "lst",      cmd_lst,     "Set/display link supervision timeout" },
    { "auth",     cmd_auth,    "Request authentication"               },
    { "enc",      cmd_enc,     "Set connection encryption"            },
    { "key",      cmd_key,     "Change connection link key"           },
    { "clkoff",   cmd_clkoff,  "Read clock offset"                    },
    { "clock",    cmd_clock,   "Read local or remote clock"           },
    { "lescan",   cmd_lescan,  "Start LE scan"                        },
    { "lewladd",  cmd_lewladd, "Add device to LE White List"          },
    { "lewlrm",   cmd_lewlrm,  "Remove device from LE White List"     },
    { "lewlsz",   cmd_lewlsz,  "Read size of LE White List"           },
    { "lewlclr",  cmd_lewlclr, "Clear LE White list"                  },
    { "lecc",     cmd_lecc,    "Create a LE Connection"               },
    { "ledc",     cmd_ledc,    "Disconnect a LE Connection"           },
    { "lecup",    cmd_lecup,   "LE Connection Update"                 },
    { NULL, NULL, 0 }
```

l2ping: l2cap 测试指令

l2ping <address> /*if can ping through*/

```
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/test# l2ping 94:DB:C9:1D:9B:EA
Ping: 94:DB:C9:1D:9B:EA from 00:E0:4C:23:D7:C3 (data size 44) ...
44 bytes from 94:DB:C9:1D:9B:EA id 0 time 5.91ms
44 bytes from 94:DB:C9:1D:9B:EA id 1 time 32.89ms
44 bytes from 94:DB:C9:1D:9B:EA id 2 time 30.02ms
44 bytes from 94:DB:C9:1D:9B:EA id 3 time 28.94ms
44 bytes from 94:DB:C9:1D:9B:EA id 4 time 38.74ms
44 bytes from 94:DB:C9:1D:9B:EA id 5 time 34.83ms
```


4.5.2 test script

bluez 提供 D-Bus 及 socket 接口给上层应用调用，bluez\test 目录下面有脚本用例提供接口调用方法；D-Bus 接口的测试一般是 python 脚本，socket 接口是 C 代码；可以参考这些测试文件来实现上层应用开发。在测试 bluez 之前首先要保证 bluetoothd 进程已经正常运行并且卡片已经成功带起来。可用 hciconfig –a 确认。

```
root@kyle-OptiPlex-380:/home/kyle# hciconfig -a
hci0:   Type: BR/EDR  Bus: USB
        BD Address: 00:E0:4C:23:D7:C3  ACL MTU: 1021:8  SCO MTU: 255:16
        UP RUNNING
        RX bytes:4328 acl:63 sco:0 events:136 errors:0
        TX bytes:3905 acl:64 sco:0 commands:55 errors:0
        Features: 0xff 0xff 0xff 0xfe 0xdb 0xff 0x7b 0x87
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH HOLD SNIFF PARK
        Link mode: SLAVE ACCEPT
        Name: 'kyle-OptiPlex-380-0'
        Class: 0x7c0100
        Service Classes: Rendering, Capturing, Object Transfer, Audio, Telephony
        Device Class: Computer, Uncategorized
        HCI Version: 4.0 (0x6)  Revision: 0xe4c
        LMP Version: 4.0 (0x6)  Subversion: 0x79ca
        Manufacturer: Realtek Semiconductor Corporation (93)
```

4.5.3 adapter 信息查看和修改

如果平台支持 python，除了用 hciconfig -a 外也可以用 test script 中的 test-adapter 查看 adapter 相关信息。

```
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/test# ./test-adapter
Usage: ./test-adapter <command>

  address
  list
  name [name]
  powered [on/off]
  pairable [on/off]
  pairabletimeout [timeout]
  discoverable [on/off]
  discoverabletimeout [timeout]
  discovering
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/test# ./test-adapter address
00:E0:4C:23:D7:C3
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/test# ./test-adapter list
 [ /org/bluez/637/hci0 ]
    Name = kyle-OptiPlex-380-0
    Powered = 0
    Devices = dbus.Array([dbus.ObjectPath('/org/bluez/637/hci0/dev_48_C1_AC_4D_0E_7B'), dbus.ObjectPath('/org/bluez/637/hci0/dev_0C_12
_62_27_8B_29')], signature=dbus.Signature('o'), variant_level=1)
    DiscoverableTimeout = 0
    PairableTimeout = 0
    Discoverable = 0
    Address = 00:E0:4C:23:D7:C3
    Discovering = 0
    Pairable = 1
    Class = 0x000000
    UUIDs = dbus.Array([dbus.String(u'00001000-0000-1000-8000-00805f9b34fb'), dbus.String(u'00001001-0000-1000-8000-00805f9b34fb'), db
us.String(u'00001112-0000-1000-8000-00805f9b34fb'), dbus.String(u'0000111f-0000-1000-8000-00805f9b34fb'), dbus.String(u'0000111e-0000-
1000-8000-00805f9b34fb'), dbus.String(u'0000110c-0000-1000-8000-00805f9b34fb'), dbus.String(u'0000110e-0000-1000-8000-00805f9b34fb'),
dbus.String(u'0000110a-0000-1000-8000-00805f9b34fb'), dbus.String(u'0000110b-0000-1000-8000-00805f9b34fb')], signature=dbus.Signature(
's'), variant_level=1)
```

bluez 在 main.conf 中可以设置 name、class 等信息。如果要修改 adapter 在手机等远端设备上的显示图标可以修改 class，例如要显示成耳机，可修改成 0x210404。(Only the major and minor device class bits are considered)

如果在 STORAGEDIR/<local address>/config 文件中已经有 name 等信息，则 bluez 会以 config 文件中的信息为准，修改 main.conf 并重启也是无效的。此时可通过 adapter 的 SetProperty 这个 dbus 调用设置，可参考 test-adapter，例如用 test-adapter name xxx 可以重新设置名字并将其保存在 STORAGEDIR/<local address>/config 文件内。

4.5.4 搜索

对 br/edr 设备可以用 hcitool inq 和 hcitool scan 进行搜索，这两个命令都会进行 10s 左右的 inquiry，hcitool scan 在 inquiry 之后还会进行 remote name request。

```
root@kyle-OptiPlex-380:/home/kyle# hcitool inq
Inquiring ...
        00:E0:4C:83:F0:0E       clock offset: 0x71f5    class: 0xbe010c
        22:22:CC:E5:6E:75       clock offset: 0x4616    class: 0x5a011c
```

```
root@kyle-OptiPlex-380:/home/kyle# hcitool scan
Scanning ...
        00:E0:4C:83:F0:0E       OPEN
        22:22:CC:E5:6E:75       Softwinner-polaris-evb
```

如果需要搜索 LE 设备，可以使用 hcitool lescan，le scan 会一直持续进行直到手动停止。

```
root@kyle-OptiPlex-380:/home/kyle# hcitool lescan
LE Scan ...
B4:99:4C:67:39:84 (unknown)
00:11:67:01:17:7D BLE Mouse
78:C5:E5:73:54:82 BLE Chest Strap
```

在 bluez/test 目录下 test-discovery 脚本也可以执行搜索的操作。

```
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/test# ./test-discovery
[ 94:DB:C9:1D:9B:EA ]
    Paired = 0
    LegacyPairing = 0
    Alias = 94-DB-C9-1D-9B-EA
    Address = 94:DB:C9:1D:9B:EA
    RSSI = -96
    Class = 0x58020c
    Trusted = 0
    Icon = phone

[ 00:E0:4C:83:F0:0E ]
    Name = OPEN
    Paired = 0
    LegacyPairing = 0
    Alias = OPEN
    Address = 00:E0:4C:83:F0:0E
    RSSI = -86
    Class = 0xbe010c
    Trusted = 0
    Icon = computer

[ 88:CB:87:62:69:C0 ]
    Paired = 0
    LegacyPairing = 0
    Alias = 88-CB-87-62-69-C0
    Address = 88:CB:87:62:69:C0
    RSSI = -96
    Class = 0x7a020c
    Trusted = 0
    Icon = phone
```

如果需要能被其它设备搜索到需要打开 inquiry scan：hciconfig hci0 piscan。这个 tool 的参数含义如下：

piscan：将 page scan 和 inquiry scan 都打开；

iscan：仅开 inquriy scan；

pscan：仅开 page scan；

noscan：将 page scan 和 inquiry scan 都关闭。

如果平台支持 python 也可以用 test-adapter discoverable/pairable 进行设置。

用 hciconfig hci0 leadv [type]可以控制 adapter 发 LE adverting，type 参数为 advertising type（0～4）；用 hciconfig hci0 noleadv 停止 advertising。

4.5.5 配对

bluez 的配对过程是基于 agent 的，bluez4.101 提供了 agent 这个 test tool。

要使用这个 tool 需要在编译过程 configure 时加上--enable-test，并且移植 check，移植过程可以参照：http://check.sourceforge.net/web/install.html；

编译出 agent tool 后可以使用 agent 0000 <address>发起和对端设备的配对过程，这种方式注册的是 device agent，配对完成后 agent 就会被释放。

```
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/test# ./agent 0000 94:DB:C9:1D:9B:EA
Confirmation request of 499469 for device /org/bluez/650/hci0/dev_94_DB_C9_1D_9B_EA
Agent has been released
```

也可以使用 agent 0000 注册 adapter agent，然后从手机端发起配对过程。Adapter agent 会一直存在。

```
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/test# ./agent 0000
Confirmation request of 317445 for device /org/bluez/650/hci0/dev_94_DB_C9_1D_9B_EA
```

如果平台支持 python，也可以用 simple-agent 实现。用 simple-agent hci0 <address>发起到和对端的配对，在 confirm passkey 时输入 yes。这种方式注册的也是 device agent，在配对

完成后会被释放。

```
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/test# ./simple-agent hci0 94:DB:C9:1D:9B:EA
RequestConfirmation (/org/bluez/650/hci0/dev_94_DB_C9_1D_9B_EA, 994027)
Confirm passkey (yes/no): yes
Release
New device (/org/bluez/650/hci0/dev_94_DB_C9_1D_9B_EA)
```

同样 simple-agent 也可以注册 adapter agent，然后由对端发起配对过程。

```
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/test# ./simple-agent
Agent registered
RequestConfirmation (/org/bluez/650/hci0/dev_94_DB_C9_1D_9B_EA, 195999)
Confirm passkey (yes/no): yes
```

配对完成后用 **test-device trusted <address> yes** 将以配对过的设备设置为 trusted 以避免出现设备连接时出现没有 agent 导致的错误。

```
root@kyle-OptiPlex-380:/home/kyle/bluez-5.24/test# ./test-device trusted 94:DB:C9:1D:9B:EA yes
root@kyle-OptiPlex-380:/home/kyle/bluez-5.24/test# ./test-device trusted 94:DB:C9:1D:9B:EA
1
```

4.5.6 搜索服务

如果要查看本地或远端支持的 profile，可以使用 sdptool 实现。

查看本地支持的 profile：sdptool browse local

NOTE: 如果是 Bluez 5.x，运行 bluetoothd 时要加-C 参数

```
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/test# sdptool browse local
Browsing FF:FF:FF:00:00:00 ...
Service Name: Headset Audio Gateway
Service RecHandle: 0x10000
Service Class ID List:
  "Headset Audio Gateway" (0x1112)
  "Generic Audio" (0x1203)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 12
Profile Descriptor List:
  "Headset" (0x1108)
    Version: 0x0102

Service Name: Hands-Free Audio Gateway
Service RecHandle: 0x10001
Service Class ID List:
  "Handsfree Audio Gateway" (0x111f)
  "Generic Audio" (0x1203)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 13
Profile Descriptor List:
  "Handsfree" (0x111e)
    Version: 0x0105

Service Name: AVRCP TG
Service RecHandle: 0x10002
Service Class ID List:
  "AV Remote Target" (0x110c)
Protocol Descriptor List:
  "L2CAP" (0x0100)
    PSM: 23
  "AVCTP" (0x0017)
    uint16: 0x103
Profile Descriptor List:
  "AV Remote" (0x110e)
    Version: 0x0104
```

用 sdptool browse <address>可以查询远端设备支持的 profile。

```
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/test# sdptool browse 94:DB:C9:1D:9B:EA
Browsing 94:DB:C9:1D:9B:EA ...
Service RecHandle: 0x10000
Service Class ID List:
  "PnP Information" (0x1200)
Profile Descriptor List:
  "PnP Information" (0x1200)
    Version: 0x0102

Service Name: Audio Source
Service RecHandle: 0x10001
Service Class ID List:
  "Audio Source" (0x110a)
Protocol Descriptor List:
  "L2CAP" (0x0100)
    PSM: 25
  "AVDTP" (0x0019)
    uint16: 0x102
Profile Descriptor List:
  "Advanced Audio" (0x110d)
    Version: 0x0102

Service Name: AVRCP TG
Service RecHandle: 0x10002
Service Class ID List:
  "AV Remote Target" (0x110c)
Protocol Descriptor List:
  "L2CAP" (0x0100)
    PSM: 23
  "AVCTP" (0x0017)
    uint16: 0x103
Profile Descriptor List:
  "AV Remote" (0x110e)
    Version: 0x0100
```

4.5.7 连接

用 hcitool cc <address>可以发起和远端设备的 ACL 连接，连接完成之后不会进行 service 的连接，所以很快就会断开。bluez 也提供了一些 test script 可以连接某些 profile。

a) a2dp source/hfp ag

在与 a2dp/hfp 耳机配对之后可以用 test-audio(Bluez 4.101)进行连接，但该 tool 仅仅是实现 profile 的连接，并没有实现 audio 相关内容，所以无法测试 audio stream 或 SCO。

测试前先用 sdptool browse local 确认支持 a2dp source 或 hfp/hsp ag。

```
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/test# ./simple-agent hci0 48:C1:AC:4D:0E:7B
Release
New device (/org/bluez/648/hci0/dev_48_C1_AC_4D_0E_7B)
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/test#
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/test#
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/test#
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/test# ./test-audio connect 48:C1:AC:4D:0E:7B
```

b) hid

bluez 4.101 提供了 hidd tool 可以和 hid device 进行连接，如果要编译该 tool，需要在 configure 时加上--enable-hidd，编译生成的 hidd tool 在 compat 文件夹中。

hidd --search 会搜索周围的 hid 设备，在搜索完成后会自动连接（确认 hid 设备处于可连接状态）；

```
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/compat# ./hidd --search
Searching ...
        Connecting to device 00:22:48:DC:89:0F
```

hidd --connect <address>去连接指定 hid 设备。

```
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/compat# ./hidd --connect 00:22:48:DC:89:0F
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/compat#
```

很多 hid 设备例如遥控器等为了省电在没有操作一段时间之后会自动断开连接，hidd tool 只有连接没有配对过程，所以断开后从 device 端也无法发起重连，可以先用 simple-agent 与 hid 设备进行配对，配对完成后将其设置为 trusted，这样在连接断开之后也可以从 device 端发起重连。

```
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/compat# ../test/simple-agent hci0 00:22:48:DC:89:0F
RequestPinCode (/org/bluez/648/hci0/dev_00_22_48_DC_89_0F)
Enter PIN Code: 0000
Release
New device (/org/bluez/648/hci0/dev_00_22_48_DC_89_0F)
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/compat# ../test/test-device trusted 00:22:48:DC:89:0F yes
root@kyle-OptiPlex-380:/home/kyle/bluez-4.101/compat# ./hidd --connect 00:22:48:DC:89:0F
```

c) HOGP (Bluez 5.x)

使用 HOGP 前确认平台加载了 uhid driver。

HOGP device 在配对完成后可以直接使用，所以使用前面介绍的配对 test script 就可以：

test-discovery

simple-agent hci0 <address>

```
root@kyle-OptiPlex-380:/home/kyle/bluez-5.24/test# ./test-discovery
[ C9:DE:4C:82:B0:1D ]
    Name = Lenovo Mice N700
    Paired = 0
    Adapter = /org/bluez/hci0
    Appearance = 962
    LegacyPairing = 0
    Alias = Lenovo Mice N700
    Connected = 0
    UUIDs = dbus.Array([dbus.String(u'00001812-0000-1000-8000-00805f9b34fb')], signature=dbus.Signature('s'), variant_level=1)
    Address = C9:DE:4C:82:B0:1D
    RSSI = -72
    Blocked = 0
    Trusted = 0
    Icon = input-mouse

[ 94:DB:C9:1D:9B:EA ]
    Name = ZTE-T
    Alias = ZTE-T
    Modalias = usb:v000Ap0000d0000
    Adapter = /org/bluez/hci0
    LegacyPairing = 0
    Paired = 1
    Connected = 0
    UUIDs = dbus.Array([dbus.String(u'00001105-0000-1000-8000-00805f9b34fb'), dbus.String(u'00001106-0000-1000-8000-00805f9b34fb'), db
us.String(u'0000110a-0000-1000-8000-00805f9b34fb'), dbus.String(u'0000110c-0000-1000-8000-00805f9b34fb'), dbus.String(u'00001112-0000-
1000-8000-00805f9b34fb'), dbus.String(u'0000111f-0000-1000-8000-00805f9b34fb'), dbus.String(u'0000112f-0000-1000-8000-00805f9b34fb'),
dbus.String(u'00001132-0000-1000-8000-00805f9b34fb'), dbus.String(u'00001200-0000-1000-8000-00805f9b34fb')], signature=dbus.Signature(
's'), variant_level=1)
    Address = 94:DB:C9:1D:9B:EA
    RSSI = -88
    Icon = phone
    Class = 0x58020c
    Trusted = 1
    Blocked = 0

^CTraceback (most recent call last):
  File "./test-discovery", line 158, in <module>
    mainloop.run()
  File "/usr/lib/python2.7/dist-packages/gi/overrides/GLib.py", line 526, in run
    raise KeyboardInterrupt
KeyboardInterrupt
root@kyle-OptiPlex-380:/home/kyle/bluez-5.24/test# ./simple-agent hci0 C9:DE:4C:82:B0:1D
Agent registered
Device paired
```

d) heartrate

Bluez 5.x 提供了 test-heartrate test script 可以测试 LE 心率带，使 LE 心率带处于可发现模式后搜索配对，然后运行 test-heartrat 就可以在 terminal 上周期性显示心率值。

test-discovery

simple-agent hci0 <address>

test-heartrate -b <address>

```
root@kyle-OptiPlex-380:/home/kyle/bluez-5.24/test# ./simple-agent hci0 78:C5:E5:73:54:82
Agent registered
Device paired
root@kyle-OptiPlex-380:/home/kyle/bluez-5.24/test# ./test-heartrate -b 78:C5:E5:73:54:82
Sensor location: chest
Measurement received from /org/bluez/hci0/dev_78_C5_E5_73_54_82
Value:  0
Contact:  1
Measurement received from /org/bluez/hci0/dev_78_C5_E5_73_54_82
Value:  0
Contact:  1
Measurement received from /org/bluez/hci0/dev_78_C5_E5_73_54_82
Value:  0
Contact:  1
Measurement received from /org/bluez/hci0/dev_78_C5_E5_73_54_82
Value:  0
Contact:  1
Measurement received from /org/bluez/hci0/dev_78_C5_E5_73_54_82
Value:  0
Contact:  1
Measurement received from /org/bluez/hci0/dev_78_C5_E5_73_54_82
Value:  0
Contact:  1
Measurement received from /org/bluez/hci0/dev_78_C5_E5_73_54_82
Value:  0
Contact:  1
Measurement received from /org/bluez/hci0/dev_78_C5_E5_73_54_82
Value:  0
Contact:  1
Measurement received from /org/bluez/hci0/dev_78_C5_E5_73_54_82
Value:  0
Contact:  1
Measurement received from /org/bluez/hci0/dev_78_C5_E5_73_54_82
Value:  73
Contact:  1
Measurement received from /org/bluez/hci0/dev_78_C5_E5_73_54_82
Value:  75
Contact:  1
Interval:  684
Measurement received from /org/bluez/hci0/dev_78_C5_E5_73_54_82
Value:  76
Contact:  1
Interval:  813
```

4.6 Bluez 5.x 测试

4.6.1 连接远端设备

在 root 权限下运行 bluetoothctl，进入一个内部 command prompt，[bluetooth]#。

```
[bluetooth]# show //查看是否 Power: yes，如果 Power 为 no，则运行 power on
[bluetooth]# agent NoInputNoOutput //可以设置其他 IO caps，如 KeyboardDisplay
[bluetooth]# default-agent
[bluetooth]# scan on //扫描到对应的设备后，可使用 scan off 关闭 scan。
[bluetooth]# pair 00:22:48:DC:89:0F //配对远端设备，可能需要输入 PIN code。
Request PIN code
[agent] Enter PIN code: 1234
[bluetooth]# connect 00:22:48:DC:89:0F
```

4.6.2 A2DP test with PulseAudio

a. (交叉)编译

以 bluez 5.25 和 PulseAudio 6.0 为例，首先需要移植和 PulseAudio 相关的 intltool, libtool, json-c, libsndfile, sbc. 如果在 PC Linux(Ubuntu or Gentoo or etc)下测试，先确认这些 packages 是否已经安装。

编译 pulseaudio，参数(for example)如下：

```
./configure --with-gnu-ld \
        --enable-shared \
        --enable-dbus \
        --disable-static \
        --disable-x11 \
        --disable-oss-output \
        --disable-oss-wrapper \
        --disable-coreaudio-output \
        --disable-esound \
        --disable-solaris \
        --disable-waveout \
        --disable-gtk3 \
        --disable-gconf \
        --disable-jack \
        --disable-asyncns \
        --disable-tcpwrap \
        --disable-lirc \
        --disable-hal-compat \
        --disable-openssl \
        --disable-xen \
        --disable-systemd \
        --disable-systemd-journal \
        --disable-manpages \
        --disable-per-user-esound-socket \
        --disable-neon-opt \
        --disable-atomic-arm-linux-helpers \
        --disable-ipv6 \
        --disable-glib2 \
        --without-speex \
        --disable-systemd-daemon \
        --disable-systemd-login \
        --disable-systemd-journal \
        --with-system-user=root \
        --with-system-group=root \
        --with-access-group=root \
        --prefix=/usr \
        --sysconfdir=/etc \
```

```
--localstatedir=/var \
--libdir=/lib \
--datarootdir=/usr/share \
--disable-bluez4 \
--enable-bluez5 \
--enable-alsa \
--enable-udev \
--enable-bluez5-native-headset \
--disable-bluez5-ofono-headset
```

如果平台是一个嵌入式平台，则需要根据 CPU Type 添加**--host=xxx-linux，如 arm-linux, mips-linux 等。**

如果想支持 bluez 4，把--disable-bluez4 改成--enable-bluez4，--enable-bluez5 改成 --disable-bluez5。

如果平台没有 udev，则把--enable-udev 改成--disable-udev

如果不需要支持 alsa，把--enable-alsa 改成--disable-alsa。

在编译 PulseAudio 前请正确设置 LIBJSON_CFLAGS, LIBJSON_LIBS, LIBSNDFILE_CFLAGS, LIBSNDFILE_LIBS (对于 PC Linux，不需要关心这些 environments，一般系统都已设置好)

b. 运行 PulseAudio

运行命令：/usr/bin/pulseaudio -n --log-level=3 --dl-search-path=/lib/pulse-6.0/modules --file=/etc/pulse/default.pa

OR

/usr/bin/pulseaudio -n --log-level=3 --dl-search-path=/lib/pulse-6.0/modules --file=/etc/pulse/system.pa –system

前者以普通用户运行 PulseAudio，PulseAudio 官方建议用这种方式运行。

后者以 system-wide 方式运行 PulseAudio，请在 root 权限下运行。一般嵌入式平台会以 root 方式运行 PulseAudio

Log level:

- error log
- error + warn log
- error + warn + info log
- error + warn + info + debug log.

一开始建议只打开 error + warn log。

如果出现类似的错误：“main.c: Failed to acquire org.pulseaudio.Server: org.freedesktop.DBus.Error.AccessDenied: Connection ":1.46" is not allowed to own the service "org.pulseaudio.Server" due to security policies in the configuration file”，需要在 /etc/dbus-1/system.d/pulseaudio-system.conf 中添加如下内容。如果以普通用户运行 pulseaudio，在这个文件中为普通用户添加一个 policy，内容和 root 用户类似。

```
<!DOCTYPE busconfig PUBLIC
 "-//freedesktop//DTD D-BUS Bus Configuration 1.0//EN"
 "http://www.freedesktop.org/standards/dbus/1.0/busconfig.dtd">
<busconfig>
    <policy user="root">
```

```
        <allow own="org.pulseaudio.Server"/>
        <allow send_destination="org.bluez"/>
        <allow send_interface="org.bluez.Manager"/>
    </policy>
    <policy user="pulse">
        <allow own="org.pulseaudio.Server"/>
        <allow send_destination="org.bluez"/>
        <allow send_interface="org.bluez.Manager"/>
    </policy>
    <policy context="default">
        <deny own="org.pulseaudio.Server"/>
        <deny send_destination="org.bluez"/>
        <deny send_interface="org.bluez.Manager"/>
    </policy>
</busconfig>
```

如果提示/etc/pulse 下不存在 system.pa，可在/etc/pulse 下创建链接 system.pa，链接到 default.pa。（cd /etc/pulse;ln -sf default.pa system.pa）

修改配置文件 system/default.pa，加载 module-bluetooth-policy, module-bluetooth-discover.

```
load-module module-bluetooth-policy
load-module module-bluetooth-discover
```

c. 测试 A2DP

参考 4.6.1，运行 bluetoothctl，连接蓝牙耳机。

pactl list modules //确保 module-bluez5-discover 和 module-bluetooth-policy 已加载

pactl list cards //查看 card

```
Card #0
    Name: bluez_card.00_18_91_00_24_57
    Driver: module-bluez5-device.c
    Owner Module: 12
    Properties:
        device.description = "WOOWI HERO"
        device.string = "00:18:91:00:24:57"
        device.api = "bluez"
        device.class = "sound"
        device.bus = "bluetooth"
        device.form_factor = "headset"
        bluez.path = "/org/bluez/hci0/dev_00_18_91_00_24_57"
        bluez.class = "0x240404"
        bluez.alias = "WOOWI HERO"
        device.icon_name = "audio-headset-bluetooth"
        device.intended_roles = "phone"
    Profiles:
        headset_head_unit: Headset Head Unit (HSP/HFP) (sinks: 1, sources: 1, priority: 20,
```

```
available: yes)
        a2dp_sink: High Fidelity Playback (A2DP Sink) (sinks: 1, sources: 0, priority: 10, available: no)
        off: Off (sinks: 0, sources: 0, priority: 0, available: yes)
    Active Profile: off
    Ports:
        headset-output: Headset (priority: 0, latency offset: 0 usec)
            Part of profile(s): headset_head_unit, a2dp_sink
        headset-input: Headset (priority: 0, latency offset: 0 usec)
            Part of profile(s): headset_head_unit
```

pactl list sinks

```
Sink #1
    State: SUSPENDED
    Name: bluez_sink.00_18_91_00_24_57
    Description: WOOWI HERO
    Driver: module-bluez5-device.c
    Sample Specification: s16le 1ch 8000Hz
    Channel Map: mono
    Owner Module: 20
    Mute: no
    Volume: mono: 43691 /    67%
            balance 0.00
    Base Volume: 65536 / 100%
    Monitor Source: bluez_sink.00_18_91_00_24_57.monitor
    Latency: 0 usec, configured 0 usec
    Flags: HARDWARE HW_VOLUME_CTRL LATENCY
    Properties:
        bluetooth.protocol = "headset_head_unit"
        device.intended_roles = "phone"
        device.description = "WOOWI HERO"
        device.string = "00:18:91:00:24:57"
        device.api = "bluez"
        device.class = "sound"
        device.bus = "bluetooth"
        device.form_factor = "headset"
        bluez.path = "/org/bluez/hci0/dev_00_18_91_00_24_57"
        bluez.class = "0x240404"
        bluez.alias = "WOOWI HERO"
        device.icon_name = "audio-headset-bluetooth"
    Ports:
        headset-output: Headset (priority: 0)
    Active Port: headset-output
    Formats:
        pcm
```

pactl set-card-profile bluez_card.00_18_91_00_24_57 a2dp_sink
如果出现 Failure: No such entity 查看 connection 是否断开.
pactl upload-sample /usr/share/sounds/alsa/Front_Center.wav sp1
pactl play-sample sp1 bluez_sink.00_18_91_00_24_57

4.6.3 Test HSP with PulseAudio

假设 Bluetooth Headset 地址为 00:0D:3C:25:89:F0

a. 修改 config files

首先打开/etc/pulse/default.pa(如果以 system wide 方式运行 pulseaudio，则修改 /etc/pulseaudio/system.pa)，找到 load-module module-bluetooth-discover，在尾部添加 headset=native:

load-module module-bluetooth-discover **headset=native**

然后重新启动 pulseaudio。

为了得到更多的 pulseaudio debug，重启 pulseaudio 前可以修改/etc/pulse/daemon.conf，添加 log-level = debug

b. 连接 Bluetooth headset

方法和连接普通蓝牙设备一样，在 bluetoothctl 运行 scan on，找到设备后运行 connect <bdaddr> 连接蓝牙耳机。

c. 检查连接是否成功

$ pactl list cards
Card #7
    Name: bluez_card.00_0D_3C_25_89_F0
    Driver: module-bluez5-device.c
    Owner Module: 30
    Properties:
        device.description = "MyVoice 312"
        device.string = "00:0D:3C:25:89:F0"
        device.api = "bluez"
        device.class = "sound"
        device.bus = "bluetooth"
        device.form_factor = "headset"
        bluez.path = "/org/bluez/hci0/dev_00_0D_3C_25_89_F0"
        bluez.class = "0x200404"
        bluez.alias = "MyVoice 312"
        device.icon_name = "audio-headset-bluetooth"
        device.intended_roles = "phone"
    Profiles:
        headset_head_unit: Headset Head Unit (HSP/HFP) (sinks: 1, sources: 1, priority: 20, available: yes)

```
        off: Off (sinks: 0, sources: 0, priority: 0, available: yes)
    Active Profile: headset_head_unit
    Ports:
        headset-output: Headset (priority: 0, latency offset: 0 usec)
            Part of profile(s): headset_head_unit
        headset-input: Headset (priority: 0, latency offset: 0 usec)
            Part of profile(s): headset_head_unit
```

d. 切换 Active profile 到 headset_head_unit

$ pactl set-card-profile bluez_card.00_0D_3C_25_89_F0 headset_head_unit

如果通过 pactl list cards 得到 Active Profile 已经为 headset_head_unit，则忽略这一步。

e. 查看 HSP 对应的 source

$ pactl list sources

```
Source #18
    State: SUSPENDED
    Name: bluez_source.00_0D_3C_25_89_F0
    Description: MyVoice 312
    Driver: module-bluez5-device.c
    Sample Specification: s16le 1ch 8000Hz
    Channel Map: mono
    Owner Module: 30
    Mute: no
    Volume: mono: 65536 / 100%
            balance 0.00
    Base Volume: 65536 / 100%
    Monitor of Sink: n/a
    Latency: 0 usec, configured 0 usec
    Flags: HARDWARE HW_VOLUME_CTRL LATENCY
    Properties:
        bluetooth.protocol = "headset_head_unit"
        device.intended_roles = "phone"
        device.description = "MyVoice 312"
        device.string = "00:0D:3C:25:89:F0"
        device.api = "bluez"
        device.class = "sound"
        device.bus = "bluetooth"
        device.form_factor = "headset"
        bluez.path = "/org/bluez/hci0/dev_00_0D_3C_25_89_F0"
        bluez.class = "0x200404"
        bluez.alias = "MyVoice 312"
        device.icon_name = "audio-headset-bluetooth"
    Ports:
        headset-input: Headset (priority: 0)
```

Active Port: headset-input
Formats:
pcm

f. 通过 HSP 录音

$ parecord --no-remix --no-remap --device="bluez_source.00_0D_3C_25_89_F0"
--file-format=wav /tmp/3.wav
运行以上 command 后，对着蓝牙 MIC 说话。
录音的时候可以看到以下 log，如果 submit SCO RX urb 失败，则无法录到声音，3.wav 会很小。
rtk_btusb: btusb_notify: hci0 evt 1
rtk_btusb: btusb_notify: Update sco num 1
rtk_btusb: btusb_work: sco num 1
rtk_btusb: set isoc interface: alt 2
rtk_btusb: submit SCO RX urb.

运行 CTRL + c 结束 parecord

5 上层应用开发

Bluez 提供 D-Bus 及 socket 接口给上层应用调用，用户可以自行开发，也可以直接采用一些开源的库和程序进行整合。

一些 Linux based 系统如 Ubutnu/Debian 发行版本已经自带 bluez 及应用和 UI，可以在这些系统进行基本测试来了解蓝牙的使用方法和应用场景。

5.1 A2dp sink

Bluez 有实现 a2dp 协议层，客户需要开发应用来控制链路的管理及 audio data 的解码及输出等，PulseAudio 有支持这部分功能，可至官网（http://pulseaudio.org）下载移植。

Bluez 中要支持 a2dp sink，需要在/etc/bluetooth/audio.conf 中添加：**Enable = Source**。

Bluez 提供的 a2dp audio 相关的 dbus 接口可以查看 doc/media-api.txt。

移植 pulseaudio 时需在 configure 过程中添加--enable-bluez，对于高版本可能会是--enable-bluez4，具体可参见 pulseaudio 的 configure 文件。

pulseaudio 中 bluetooth 相关的 code 在 src/modules/bluetooth 中，以 pulseaudio 5.0 为例，使用 bluez4 时在 bluez4-util.c 文件的 register_endpoint 函数注册 endpoint，通过 pa_bluez4_transport_acquire 函数获取 stream fd，在 module-bluetooth-bluez4.c 文件 a2dp_process_push 函数中获取 a2dp media data 并进行 decode。

5.2 AVRCP CT

bluez4.101 提供的 AVRCP CT 相关的 dbus 接口可以查看 doc/control-api.txt。具体实现的 code 在 audio/control.c 文件中。4.101 提供的 avrcp cmd 只有 volume up 和 volume down，不过大部分的 TG 端并不支持，可以参照这两个函数实现其它的 AVRCP 操作，只需要将 avctp_send_passthrough(control->session, VOL_UP_OP)中 VOL_UP_OP 换成其它的 op code 即可，例如 PLAY_OP，PAUSE_OP 等。

bluez5.x 中提供的 AVRCP CT 相关的 dbus 接口可以查看 doc/media-api.txt，interface 为 org.bluez.MediaControl1。bluez5 中提供了更多的 AVRCP 操作，包括 play，pause 等，实现在 profile/audio/control.c 文件中，如果需要也可参照添加其它操作。

5.3 HFP HF

Bluez4.101 中要支持 HFP HF，需要在 audio.conf 中添加：**Enable = Gateway**。

客户需要开发上层应用实现 AT cmd 的控制和 sco data 的处理等，oFono 有支持这部分功能，客户可以到官网（https://www.kernel.org/pub/linux/network/ofono）上下载移植。移植 oFono 时需在 configure 过程中添加--enable-bluez，对于高版本可能会是--enable-bluez4，具体可参见 oFono 的 configure 文件。

oFono 实现 HFP HF 功能需要修改以下几个地方（以 oFono 1.14 code 为例，其它版本可参照看是否需要修改。）：

1、 oFono 启动后会监听 bluez 的 Newconnection 的 dbus 消息，在收到该消息后会根据对端的 version 注册 feature，bluez4.101 只支持 HFP 1.5，所以要修改 plugins/hfp-hf-bluez4.c 中 hfp_agent_new_connection 函数 hfp_slc_info_init(&hfp_data->info, version) 为 hfp_slc_info_init(&hfp_data->info, HFP_VERSION_1_5)；

2、 bluez 中不要 enable sap，否则会和部分手机出现 IOT 问题；

3、 将 drivers/hfpmodem/hfpmodem.c 中 hfpmodem_init 时 hfp_siri_init 去掉，hfpmodem_exit 中 hfp_siri_exit 也去掉，小米 3 手机在回复这个 feature 的 AT cmd 的 response 会有错误，导致 ofono 状态有误，outgoing 会没有声音；

```
static int hfpmodem_init(void)
{
    …..
    //hfp_siri_init();
    return 0;
}

static void hfpmodem_exit(void)
{
    ……
    //hfp_siri_exit();
}
```

4、 修改 bluez4.101 中 audio/manager.c 文件 hfp_hs_record 函数，添加注册 sdp record 中

supported feature，否则会与一些手机出现无法连接的问题；

```
static sdp_record_t *hfp_hs_record(uint8_t ch)
{
      ……
       sdp_data_t *channel, *features;
       uint16_t sdpfeat;

      ……
       sdpfeat = 0x09;
       features = sdp_data_alloc(SDP_UINT16, &sdpfeat);
       sdp_attr_add(record, SDP_ATTR_SUPPORTED_FEATURES, features);

       aproto = sdp_list_append(0, apseq);
       sdp_set_access_protos(record, aproto);
      ……
}
```

5、 ofono 对+VGS 的响应在 src/call-volume.c 文件的 ofono_call_volume_set_speaker_volume 函数中，收到该 cmd 后会在 org.ofono.CallVolume 接口产生 SpeakerVolume signal，客户可监听该 signal 设置平台的 speaker volume；对 microphone volume 的处理也在该函数中，对应 signal 为 MicrophoneVolume。

对于 SCO over PCM 的方案，可以参考 tools/huawei-audio.c 实现语音数据的处理；对于 SCO over HCI 的方案，可以移植 PulseAudio 实现语音数据的处理，以 pulseaudio5.0 为例，语音数据的收发处理在 module-bluez4-device.c 文件的 hsp_process_render 和 hsp_process_push 函数。

5.4 Bluez-alsa

对于嵌入式的应用，往往受限于 Code Size Limit。PulseAudio 是一个比较庞大的声音系统(Sound Server for Sound Routing)，对于 flash 比较小的嵌入式环境不是很好的选择。按照 bluez-alsa 的说明，This project[Bluez-alsa] is a rebirth of a direct integration between Bluez and ALSA。Bluez-4.xx 内部集成了 BT 到 ALSA 的 sound routing，但是在 Bluez-5.xx 中，这种内建的集成被移除，BT audio routing 转而寻求第三方的 audio app,比如 PulseAudio。所以目前的状态是如果使用 BT audio，就必须用 PulseAudio 或者使用 Bluez-4.xx。Bluez-4.xx 已经过时，官方也不再维护，Bluez 官方推荐使用 Bluez-5.xx。所以安装 PulseAudio 是唯一的办法。

Bluez-alsa 可以替代 PulseAudio，而且依赖更少。Bluez-alsa 注册所有的 Bluez 所支持的 audio profile。Bluez-alsa 的实现基于 ALSA PCM plugin。

5.4.1 Dependencies

- Alsa-lib
- Bluez-5+
- Glib with GIO support
- Sbc

5.4.2 编译安装

下载 bluez-alsa 源代码：

```
$ git clone https://github.com/Arkq/bluez-alsa.git
$ autoreconf --install
$ mkdir build && cd build
$ ../configure --enable-aac --enable-debug
$ make && make install
```

Bluealsa类似于bluez运行于mainloop中，bluez运行的时候会通过dbus提供一些service：其中一个是 RegisterProfile()，bluealsa 初始化时首先注册所有支持的 audio profile:

```
me@mybox:~/bluez-alsa-master$ sudo ./src/bluealsa
[sudo] password for me:
./src/bluealsa: bluez.c:699: Registering endpoint: 0000110A-0000-1000-8000-00805
F9B34FB: /MediaEndpoint/A2DPSource
./src/bluealsa: bluez.c:699: Registering endpoint: 0000110B-0000-1000-8000-00805
F9B34FB: /MediaEndpoint/A2DPSink
./src/bluealsa: bluez.c:935: Registering profile: 00001108-0000-1000-8000-00805F
9B34FB: /Profile/HSPHeadset
./src/bluealsa: bluez.c:935: Registering profile: 00001112-0000-1000-8000-00805F
9B34FB: /Profile/HSPAudioGateway
./src/bluealsa: ctl.c:489: Starting controller loop
./src/bluealsa: bluez.c:935: Registering profile: 0000111E-0000-1000-8000-00805F
9B34FB: /Profile/HFPHandsFree
./src/bluealsa: bluez.c:935: Registering profile: 0000111F-0000-1000-8000-00805F
9B34FB: /Profile/HFPAudioGateway
./src/bluealsa: main.c:226: Starting main dispatching loop
```

5.4.3 A2DP

1. A2DP Sink

   在 bluetoothctl 中

```
$connect xx:xx:xx:xx:xx
```

   在另外的 terminal:

```
$bluealsa-aplay xx:xx:xx:xx:xx
```

2. A2DP SRC

```
$aplay -D bluealsa:HCI=hci0,DEV=xx:xx:xx:xx:xx:xx,PROFILE=a2dp xxx.wav
```

   即可

5.4.4 HFP

1. Hand-free

在 bluetoothctl 中

```
$connect xx:xx:xx:xx:xx
```

在另外的 terminal:

```
$bluealsa-aplay xx:xx:xx:xx:xx
```

3. Audio Gateway

```
$aplay -D bluealsa:HCI=hci0,DEV=xx:xx:xx:xx:xx:xx,PROFILE=sco xxx.wav
```

即可

5.5 LE profile

对于 Bluez 5.x 中已有的 LE profile 可以用 test 文件夹中提供的 test script 进行测试并参考开发。

5.5.1 GATT API

如果需要实现新的 LE profile，可以使用 bluez 提供的 GATT 接口进行开发。

bluez 实现的 GATT 接口在 attrib/gatt.c 及 gattrib.c 文件中，主要实现的 GATT api 有：

a) guint gatt_discover_primary(GAttrib *attrib, bt_uuid_t *uuid, gatt_cb_t func,
gpointer user_data)

搜索 UUID 为指定 uuid 的 primary service，如果 uuid 为 NULL，则搜索所有的 primary service。

b) unsigned int gatt_find_included(GAttrib *attrib, uint16_t start, uint16_t end,
gatt_cb_t func, gpointer user_data)

搜索 handle 在 start 和 end 之间的 included service。

c) guint gatt_discover_char(GAttrib *attrib, uint16_t start, uint16_t end,
bt_uuid_t *uuid, gatt_cb_t func,
gpointer user_data)

搜索 handle 在 start 和 end 之间的 UUID 为指定 uuid 的 characteristics，如果 uuid 为 NULL，则搜索所有的 characteristics

d) guint gatt_read_char_by_uuid(GAttrib *attrib, uint16_t start, uint16_t end,
bt_uuid_t *uuid, GAttribResultFunc func,
gpointer user_data)

读取 handle 在 start 和 end 之间的 Attribute Type 为 uuid 的 attributes。

e) guint gatt_read_char(GAttrib *attrib, uint16_t handle, GAttribResultFunc func,
gpointer user_data)

读取指定 handle 的 attribute。

f) guint gatt_discover_char_desc(GAttrib *attrib, uint16_t start, uint16_t end,
GAttribResultFunc func, gpointer user_data)

搜索 handle 在 start 和 end 之间的所有 characteristic descriptors。

g) guint gatt_write_char(GAttrib *attrib, uint16_t handle, uint8_t *value,
size_t vlen, GAttribResultFunc func, gpointer user_data)

对指定 handle 的 attribute 写入长度为 vlen 的数据 value。根据数据长短会自动调用 write characteristic 或 write long characteristic 方式。

h) guint gatt_write_cmd(GAttrib *attrib, uint16_t handle, uint8_t *value, int vlen,
GDestroyNotify notify, gpointer user_data)

对指定 handle 的 attribute 用 write cmd 的方式写入长度为 vlen 的数据 value。

i) guint gatt_exchange_mtu(GAttrib *attrib, uint16_t mtu, GAttribResultFunc func,
gpointer user_data)

与对端设备交换 mtu.

j) guint g_attrib_register(GAttrib *attrib, guint8 opcode, guint16 handle,
GAttribNotifyFunc func, gpointer user_data,
GDestroyNotify notify)

对指定的 handle 和 op code（notify/indicate 等）注册回调函数 func，在收到对应的 notification 或 indication 之后就会调用 func。

k) gboolean gatt_service_add(struct btd_adapter *adapter, uint16_t uuid,
bt_uuid_t *svc_uuid, gatt_option opt1, ...)

添加 GATT service，用于 GATT server 中。

bluez5.x 提供了 GATT test tool，位于 attrib 目录下，该 tool 提供了对远端设备的 attitude 的搜索和读写等功能。

```
root@kyle-OptiPlex-380:/home/kyle/Downloads/bluez-5.24/attrib# ./gatttool --help-all
Usage:
  gatttool [OPTION...]

Help Options:
  -h, --help                            Show help options
  --help-all                            Show all help options
  --help-gatt                           Show all GATT commands
  --help-params                         Show all Primary Services/Characteristics arguments
  --help-char-read-write                Show all Characteristics Value/Descriptor Read/Write arguments

GATT commands
  --primary                             Primary Service Discovery
  --characteristics                     Characteristics Discovery
  --char-read                           Characteristics Value/Descriptor Read
  --char-write                          Characteristics Value Write Without Response (Write Command)
  --char-write-req                      Characteristics Value Write (Write Request)
  --char-desc                           Characteristics Descriptor Discovery
  --listen                              Listen for notifications and indications

Primary Services/Characteristics arguments
  -s, --start=0x0001                    Starting handle(optional)
  -e, --end=0xffff                      Ending handle(optional)
  -u, --uuid=0x1801                     UUID16 or UUID128(optional)

Characteristics Value/Descriptor Read/Write arguments
  -a, --handle=0x0001                   Read/Write characteristic by handle(required)
  -n, --value=0x0001                    Write characteristic value (required for write operation)

Application Options:
  -i, --adapter=hciX                    Specify local adapter interface
  -b, --device=MAC                      Specify remote Bluetooth address
  -t, --addr-type=[public | random]     Set LE address type. Default: public
  -m, --mtu=MTU                         Specify the MTU size
  -p, --psm=PSM                         Specify the PSM for GATT/ATT over BR/EDR
  -l, --sec-level=[low | medium | high] Set security level. Default: low
  -I, --interactive                     Use interactive mode
```

例如：可用 gatttool -b <address> --characteristics 获取对方所有的 characteristics，也可加上 start handle 和 end handle：

gatttool –b <MAC address> --characteristics –s <start handle> -e <end handle>

```
root@kyle-HP-Pro-3330-MT:/home/kyle/bluez-5.13/attrib# ./gatttool -b 00:11:67:00:45:67 --primary
attr handle = 0x0100, end grp handle = 0x0117 uuid: 00001812-0000-1000-8000-00805f9b34fb
attr handle = 0x0118, end grp handle = 0x011b uuid: 0000180f-0000-1000-8000-00805f9b34fb
attr handle = 0x0150, end grp handle = 0x0162 uuid: 0000180a-0000-1000-8000-00805f9b34fb
attr handle = 0x0170, end grp handle = 0x0172 uuid: 00001813-0000-1000-8000-00805f9b34fb
root@kyle-HP-Pro-3330-MT:/home/kyle/bluez-5.13/attrib# ./gatttool -b 00:11:67:00:45:67 --char-desc
handle = 0x0100, uuid = 2800
handle = 0x0101, uuid = 2802
handle = 0x0102, uuid = 2803
handle = 0x0103, uuid = 2a4a
handle = 0x0104, uuid = 2803
root@kyle-HP-Pro-3330-MT:/home/kyle/bluez-5.13/attrib# ./gatttool -b 00:11:67:00:45:67 --characteristics
handle = 0x0102, char properties = 0x02, char value handle = 0x0103, uuid = 00002a4a-0000-1000-8000-00805f9b34fb
handle = 0x0104, char properties = 0x16, char value handle = 0x0105, uuid = 00002a33-0000-1000-8000-00805f9b34fb
handle = 0x0107, char properties = 0x02, char value handle = 0x0108, uuid = 00002a4b-0000-1000-8000-00805f9b34fb
handle = 0x0110, char properties = 0x12, char value handle = 0x0111, uuid = 00002a4d-0000-1000-8000-00805f9b34fb
handle = 0x0114, char properties = 0x04, char value handle = 0x0115, uuid = 00002a4c-0000-1000-8000-00805f9b34fb
handle = 0x0116, char properties = 0x06, char value handle = 0x0117, uuid = 00002a4e-0000-1000-8000-00805f9b34fb
handle = 0x0119, char properties = 0x12, char value handle = 0x011a, uuid = 00002a19-0000-1000-8000-00805f9b34fb
handle = 0x0151, char properties = 0x02, char value handle = 0x0152, uuid = 00002a29-0000-1000-8000-00805f9b34fb
handle = 0x0153, char properties = 0x02, char value handle = 0x0154, uuid = 00002a24-0000-1000-8000-00805f9b34fb
handle = 0x0155, char properties = 0x02, char value handle = 0x0156, uuid = 00002a25-0000-1000-8000-00805f9b34fb
handle = 0x0157, char properties = 0x02, char value handle = 0x0158, uuid = 00002a27-0000-1000-8000-00805f9b34fb
handle = 0x0159, char properties = 0x02, char value handle = 0x015a, uuid = 00002a26-0000-1000-8000-00805f9b34fb
handle = 0x015b, char properties = 0x02, char value handle = 0x015c, uuid = 00002a28-0000-1000-8000-00805f9b34fb
handle = 0x015d, char properties = 0x02, char value handle = 0x015e, uuid = 00002a23-0000-1000-8000-00805f9b34fb
handle = 0x015f, char properties = 0x02, char value handle = 0x0160, uuid = 00002a50-0000-1000-8000-00805f9b34fb
handle = 0x0161, char properties = 0x02, char value handle = 0x0162, uuid = 00002a2a-0000-1000-8000-00805f9b34fb
handle = 0x0171, char properties = 0x04, char value handle = 0x0172, uuid = 00002a4f-0000-1000-8000-00805f9b34fb
```

5.5.2 Plugin

要开发新的 profile，需要注册 plugin，首先在函数中添加如下方式：

BLUETOOTH_PLUGIN_DEFINE(name, VERSION,
BLUETOOTH_PLUGIN_PRIORITY_DEFAULT, init, exit)

然后修改 Makefile 将其加入到 bluetoothd 的 builtin plugin 中，在 Makefile.plugins 中添加

builtin_modules += name

builtin_sources +=<src file>

在 Bluez 5.x 目录下用 automake --add-missing 命令生成 Makefile.in 文件，如果出现 automake version error，需要先执行 aclocal，客户需要移植这些 tool 和一些相关的库，如 libtool 等。生成 Makefile 之后可按 4.1 中介绍的方式编译。

在 init 函数中需要将该 profile 添加到 adapter 的 profile 链表上，可以调用 btd_profile_register(&profile_table)实现，在 exit 时需要 unregister。

5.5.3 Client

GATT client 的开发可以参考 profiles/heartrate/heartrate.c 实现。

profile_table 中可以注册 adapter_probe 和 device_probe 函数及 remote_uuid。adapter_probe 会在 adapter 起来的时候被调用，可以在这里注册一些 D-Bus 函数提供给上层应用。device_probe 会在发现对端的 uuid 与 remote_uuid 相同时被调用，可以在这里开始搜索 primary service 并用 btd_device_add_attio_callback 注册 GATT 连接后的 callback 函数，在 callback 函数中实现 characteristics 的读写等。

5.5.4 Server

GATT server 的开发可以参考 profiles/time/server.c 实现。

开发 GATT server 最重要是将 profile 相关的内容写入 server 的 database 中，并注册一些 characteristics 读写时的处理。register service 可以通过调用 gatt_service_add 实现。以 time service 为例介绍该函数的使用。

```
static gboolean register_current_time_service(struct btd_adapter *adapter)
{
	bt_uuid_t uuid;
	bt_uuid16_create(&uuid, CURRENT_TIME_SVC_UUID);

	/* Current Time service */
	return gatt_service_add(adapter, GATT_PRIM_SVC_UUID, &uuid,
			/* CT Time characteristic */
			GATT_OPT_CHR_UUID16, CT_TIME_CHR_UUID,
			GATT_OPT_CHR_PROPS,
			ATT_CHAR_PROPER_READ | ATT_CHAR_PROPER_NOTIFY,
			GATT_OPT_CHR_VALUE_CB, ATTRIB_READ,
				current_time_read, adapter,

			/* Local Time Information characteristic */
			GATT_OPT_CHR_UUID16, LOCAL_TIME_INFO_CHR_UUID,
			GATT_OPT_CHR_PROPS, ATT_CHAR_PROPER_READ,
			GATT_OPT_CHR_VALUE_CB, ATTRIB_READ,
				local_time_info_read, adapter,

			GATT_OPT_INVALID);
}
```

gatt_service_add 函数第一个参数为 adapter，第二个参数是注册 service 类型的 UUID，例子中是 primary service，第三个参数就该 service 的 UUID，之后直至 GATT_OPT_INVALID 就是各个 characteristic 内容，每个 characteristic 以 GATT_OPT_CHR_UUID 或 GATT_OPT_CHR_UUID16 开始。GATT_OPT_CHR_UUID16 后面是该 characteristic 的 UUID，GATT_OPT_CHR_PROPS 为该 characteristic 的 property，GATT_OPT_CHR_VALUE_CB 后面有 3 个参数，第一个是类型，一般为 ATTRIB_READ 或 ATTRIB_WRITE，第二个是 callback function，当对该 characteristic 进行 read 或 write 时就会调用到，第三个是 user_data，会传递到 callback function 中。

6 Debug tool

6.1 hcidump

bluez 官方有提供 hcidump tool，可以使用 hcidump 录下 host stack 与 controller 之间的蓝牙数据交互流程。Bluez 4.101 中没有这个包含这个工具，需要在 bluez 官网下载移植；bluez5.x 版本中已经将 hcidump 包含在其中，在 tools 文件夹下。

hcidump 工具可以直接运行将 log 打印在 terminal，也可以用以下方式将 log 保存在文件：

hcidump -t > x.txt

hcidump -w x.cfa

6.2 bluez log

运行 bluetoothd 时加***-d -n*** 参数打开 bluez 中的 log，或者按如下方式修改 src/log.h 文件，然后重新编译。

```
#define DBG(fmt, arg...) do { \
        static struct btd_debug_desc __btd_debug_desc \
        __attribute__((used, section("__debug"), aligned(8))) = { \
                .file = __FILE__, .flags = BTD_DEBUG_FLAG_PRINT, \
        }; \
        if (__btd_debug_desc.flags & BTD_DEBUG_FLAG_PRINT) \
                btd_debug("%s:%s() " fmt,   __FILE__, __FUNCTION__ , ## arg); \
} while (0)
```

Bluez 的 log 是以 vsyslog 方式保存 log，所以要确定客户平台是否支持，如 ubuntu 系统 log 会存储在/var/log/vsyslog 文件中。客户可以移植 busybox 将 syslogd 打开，启动后在后台运行 syslogd，log 就会被保存在/var/log/message 文件中。

```
void btd_debug(const char *format, ...)
{
        va_list ap;

        va_start(ap, format);

        vsyslog(LOG_DEBUG, format, ap);

        va_end(ap);
}
```

6.3 Kernel log

可以通过 echo 7 > /proc/sys/kernel/prink 修改 printk 级别，将 net/bluetooth/相关文件的 log 打印出来，也可以增加 log 来定位问题， cat /proc/kmsg > x.txt。

6.4 FW log

为了更方便的抓取 FW 我们提供下发特殊 cmd 的方式 FW log 会随 event 返回
利用 bluez 自带的 hcitool 和 hcidump 工具首先在终端打开 hcidump
hcidump -w xx.cfa 这个命令是将输出的 log 重定向的 xx.cfa 然后打开一个新的终端输入命令 hcitool cmd_cmd 0x3F 0x27 0x0 0x0 0x0 0x1 这样 FW log 就会一直输出到 xx.cfa。

如果要关闭 FW log 的输出只需要 hcitool cmd_cmd 0x3F 0x27 0x0 0x0 0x0 0x0 即可。