BlueZ Tools

Dispositivos y Sistemas Inalámbricos

Francisco M. Delicado Martínez[*]

Depto. Sistemas Informáticos

Escuela Superior de Ingeniería Informática de Albacete

Universidad de Castilla-La Mancha

Índice

[*]francisco.delicado@uclm.es

1. ¿Qué es BlueZ?

BlueZ[4] es la implementación en Linux de la pila de protocolos Bluetooth Low Energy, la cual se incluye oficialmente en el núcleo de Linux a partir de la versión 2.4.6.

1.1. Instalación de BlueZ

BlueZ está incluido en el núcleo de Linux a partir de la version 2.4.6 del mismo. Por ello para la mayoría de las distribuciones de nueva instalación, o que mantienen su núcleo actualizado, no es necesaria la instalación explícita del BlueZ.

Dependiendo de la distribución Linux sobre la que se desée instalar BlueZ, puede que existan paquetes precompilados de BlueZ. En caso de no existir dichos paquetes precompilados la información para instalar BlueZ a partir de los fuentes puede encontrarse en [4].

Instalación en Debian/Ubuntu/Raspbian.
Existen paquetes .deb con la implementación precompilada de BlueZ. La instalación es similar a la instalación de cualquier paquete .deb, teniendo en cuenta que en esta distro el paquete bluez.deb depende de los paquetes bluetooth.deb, libbluetooth-dev.deb y libudev-dev.deb. Por ello la instalación se realiará con los siguientes comandos:

```
$ sudo apt-get update
$ sudo apt-get install bluetooth bluez libbluetooth-dev
    libudev-dev
```

Instalación en Fedora En Fedora la dependencia del paquete bluez es con los paquetes bluez-libs y bluez-libs-devel; instalandose todos los paquetes con el siguiente comando:

```
$ sudo yum install bluez bluez-libs bluez-libs-devel
```

1.2. Herramientas incluidas en BlueZ

Además de la especificación de la pila de protocolos Bluetooth, BlueZ, también incluye una serie de herramientas que permiten la comunicación

con dispositivos Bluetooth. A continuación describimos alguna de dichas herramientas.

1.2.1. hciconfig

La aplicación `hciconfig` se utiliza, de manera similar a la aplicación `ifconfig`, para ver el estado y configurar los dispositivos Bluetooth que hay instalados en una máquina Linux.

A continuación se especifican algunas de las acciones más comunes que se pueden realizar con esta herramienta.

Ver estado de las interfaces.

Para ver el estado de todas las interfaces Bluetooth se utilizará el comando:

```
$ hciconfig
```

que muestra una información reducida del estado de todas las interfaces Bluetooth:

```
hci1:   Type: BR/EDR  Bus: USB
        BD Address: 00:1A:7D:DA:71:13  ACL MTU: 310:10
           SCO MTU: 64:8
        DOWN
        RX bytes:889 acl:0 sco:0 events:40 errors:0
        TX bytes:951 acl:0 sco:0 commands:40 errors:0

hci0:   Type: BR/EDR  Bus: USB
        BD Address: FC:F8:AE:30:FD:35  ACL MTU: 1021:5
           SCO MTU: 96:5
        UP RUNNING PSCAN
        RX bytes:11765 acl:10 sco:0 events:275 errors:0
        TX bytes:22377 acl:10 sco:0 commands:184 errors:0
```

Donde podemos ver que el dispositivo consta de dos interfaces Bluetooth: `hci0` y `hci1`. Estando `hci0` activa, de ahí la salida (`UP RUNNING PSCAN`); mientras que la interfaz `hci1` está inactiva (estado `DOWN`).

Las interfaces son nombradas como `hciX`, ya que el sistema no accede directamente a la interfaz física de emisión Bluetooth, sino que solo puede acceder al `HCI` (*Host Controller Interface*) de la interfaz Bluetooth.

Si al comando anterior le damos el nombre de la interfaz Bluetooth, solo obtendremos información sobre dicha interfaz:

```
$ hciconfig hci1
hci1:   Type: BR/EDR  Bus: USB
        BD Address: 00:1A:7D:DA:71:13  ACL MTU: 310:10
           SCO MTU: 64:8
        DOWN
        RX bytes:889 acl:0 sco:0 events:40 errors:0
        TX bytes:951 acl:0 sco:0 commands:40 errors:0
```

Como se puede ver la información sobre la interfaz es muy escasa. Para obtener una información más detallada sobre la interfaz: nombre de la misma, clase de interfaz, si es de un smartphone, un PC u otro dispositivo, fabricante, versión de `HCI`, etc. podemos añadir la opcion `-a`, al comando:

```
$ hciconfig -a
hci1:   Type: BR/EDR  Bus: USB
        BD Address: 00:1A:7D:DA:71:13  ACL MTU: 310:10
           SCO MTU: 64:8
        DOWN
        RX bytes:2049 acl:0 sco:0 events:82 errors:0
        TX bytes:1667 acl:0 sco:0 commands:82 errors:0
        Features: 0xff 0xff 0x8f 0xfe 0xdb 0xff 0x5b 0x87
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH HOLD SNIFF PARK
        Link mode: SLAVE ACCEPT

hci0:   Type: BR/EDR  Bus: USB
        BD Address: FC:F8:AE:30:FD:35  ACL MTU: 1021:5
           SCO MTU: 96:5
        UP RUNNING PSCAN
        RX bytes:12596 acl:10 sco:0 events:284 errors:0
        TX bytes:22404 acl:10 sco:0 commands:193 errors:0
        Features: 0xff 0xfe 0x0f 0xfe 0xdb 0xff 0x7b 0x87
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH HOLD SNIFF
        Link mode: SLAVE ACCEPT
        Name: 'pelijes-laptop'
        Class: 0x00010c
        Service Classes: Unspecified
        Device Class: Computer, Laptop
        HCI Version: 4.0 (0x6)  Revision: 0x500
        LMP Version: 4.0 (0x6)  Subversion: 0x500
        Manufacturer: Intel Corp. (2)
```

Como en el caso anterio, si añadimos el nombre de la interfaz, solo obtendremos la salida sobre dicha interfaz:

```
$ hciconfig -a hci0
hci0:   Type: BR/EDR  Bus: USB
        BD Address: FC:F8:AE:30:FD:35  ACL MTU: 1021:5
           SCO MTU: 96:5
        UP RUNNING PSCAN
        RX bytes:12873 acl:10 sco:0 events:287 errors:0
        TX bytes:22413 acl:10 sco:0 commands:196 errors:0
        Features: 0xff 0xfe 0x0f 0xfe 0xdb 0xff 0x7b 0x87
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH HOLD SNIFF
        Link mode: SLAVE ACCEPT
        Name: 'pelijes-laptop'
        Class: 0x00010c
        Service Classes: Unspecified
        Device Class: Computer, Laptop
        HCI Version: 4.0 (0x6)  Revision: 0x500
        LMP Version: 4.0 (0x6)  Subversion: 0x500
        Manufacturer: Intel Corp. (2)
```

Activación de una interfaz interfaz.

La activación de una interfaz se realiza mediante el comando:

```
  $ hciconfig <hciX> up
```

donde `<hciX>` es el nombre de la interfaz Bluetooth a activar, y la activación se realiza con la opción `up`. Un ejemplo de uso se muestra a continuación:

```
$ hciconfig
hci0:   Type: BR/EDR  Bus: USB
        BD Address: FC:F8:AE:30:FD:35  ACL MTU: 1021:5
           SCO MTU: 96:5
        DOWN
        RX bytes:13150 acl:10 sco:0 events:290 errors:0
        TX bytes:22422 acl:10 sco:0 commands:199 errors:0
```

Como se puede ver la interfaz `hci0` está inactiva, por lo que para activarla deberemos ejecutar:

```
$ sudo hciconfig hci0 up
$ hciconfig
hci0: Type: BR/EDR  Bus: USB
      BD Address: FC:F8:AE:30:FD:35  ACL MTU: 1021:5
         SCO MTU: 96:5
      UP RUNNING PSCAN
      RX bytes:13750 acl:10 sco:0 events:325 errors:0
      TX bytes:23114 acl:10 sco:0 commands:234 errors:0
```

pasando la interfaz a estar activa; lo que se indica con el mensaje `UP RUNNING PSCAN`.

Desactivación interfaz.
Se realiza de manera similar a como se hace la activación, solo que ahora ejecutamos el comando `down`:

```
$ hciconfig <hciX> down
```

donde como antes `<hciX>` indica el nombre de la interfaz Bluetooth a desactivar, y que previamente debería estar activada.

```
$ sudo hciconfig hci0 down
$ hciconfig
hci0: Type: BR/EDR  Bus: USB
      BD Address: FC:F8:AE:30:FD:35  ACL MTU: 1021:5
         SCO MTU: 96:5
      DOWN
      RX bytes:13750 acl:10 sco:0 events:325 errors:0
      TX bytes:23114 acl:10 sco:0 commands:234 errors:0
```

La herramienta `hcitool` permite una gran cantidad de operaciones sobre la interfaz Bluetooth, que van desde escanear dispositivos, hasta establecer una conexión. Para una descripción mas detallada de esta herramienta se puede consultar: la página `man` de la aplicación, o los siguientes enlaces [6] y [7]

1.2.2. `hcitool`

Esta herramienta permite configurar conexiones Bluetooth, así como enviar algunos comandos a los dispositivos Bluetooth que están en tu área de cobertura.

Esta herramienta se puede utilizar tanto para comunicarse con dispositivos Bluetooth, como con dispositivos que soportan Bluetooth LE. En esta sección nos centraremos en comentar algunos comandos a utilizar para interactuar con Bluetooth LE, el resto de información se puede obtener consultando la página `man` de la aplicación. También se puede obtener la ayuda en línea con la opción `-help`.

```
$ hcitool --help
```

Que puede usarse con cada comando para obtener la ayuda sobre los parámetros que dicho comando acepta. Por ejemplo, para ver las opciones del parámetro `lescan` ejecutaremos:

```
$ hcitool lescan --help
Usage:
        lescan [--privacy] enable privacy
        lescan [--passive] set scan type passive (default
            active)
        lescan [--whitelist] scan for address in the
           whitelist only
        lescan [--discovery=g|l] enable general or
           limited discoveryprocedure
        lescan [--duplicates] don't filter duplicates
```

Ahora se muestran las acciones más frecuentes que se pueden realizar con el comando `hcitool`.

Escaneado pasivo.
El escaneado pasivo de los dispositivos que están en el área de cobertura de nuestro dispositivo Bluetooth LE con nombre de interfaz `<hciX>` se obtendrá con el siguiente comando:

```
$ sudo hcitool -i <hciX> lescan
```

Un ejemplo de uso para el caso de la interfaz `hci0` sería:

```
$ sudo hcitool -i hci0 lescan
LE Scan ...
C4:BE:84:E5:20:56 (unknown)
C4:BE:84:E5:20:56 LightBlueBean-3
FE:0A:74:24:CC:00 (unknown)
FE:0A:74:24:CC:00 EST
```

```
DB:AC:3C:29:6B:09 (unknown)
DB:AC:3C:29:6B:09 EST
C4:BE:84:E5:69:9D (unknown)
C4:BE:84:E5:69:9D LightBlueBean-A
```

Se observa como por cada uno de los dispositivos Bluetooth aparecen dos líneas (las dos con la misma dirección MAC), indicándonos en una de ellas el nombre del dispositivo Bluetooth.

Como se puede observar el parámetro `-i hci0` indica la interfaz Bluetooth a utilizar para el escaneo.

Establecimiento de una conexión.

Una de las funcionalidades de `hcitool` es el establecimiento de una conexión Bluetooth LE. Para ello necesitamos obtener la dirección MAC Bluetooth del dispositivo al que queremos conectarnos. Algo que como hemos visto antes se obtiene con el comando `hcitool -i hciX lescan`.

Una vez obtenida la dirección del dispositivo Bluetooth, la conexión con él se realiza con el comando:

```
$ sudo hcitool -i <hciX> lecc <MAC_dispositivo>
```

donde `<hciX>` indica el nombre de la interfaz Bluetooth utilizada para conectarnos con el dispositivo Bluetooth con MAC `<MAC_dispositivo>`

Un ejemplo de uso se muestra a continuación:

```
$ sudo hcitool -i hci0 lecc C4:BE:84:E5:69:9D
Connection handle 3585
```

Que como se ve, devuelve el *handle* de la conexión. Este se deberá usar para acceder a la conexión en posteriores llamadas a esta aplicación.

Cierre de la conexión.

`hcitool` también se puede utilizar para el cierre de una conexión Bluetooth, si disponemos del *handle* de la conexión a cerrar. El comando a utilizar es:

```
$ sudo hcitool -i <hciX> ledc <handle> [reason]
```

donde como en el caso anterior `<hciX>` es la interfaz Bluetooth a través de la que se estableció la conexión, `<handle>` es el valor del *handle* de la

conexión a cerrar. Y `reason` es un parámetro opcional que indica al otro extremo de la conexión la causa del cierre.

Así si seguimos con la conexión establecida en la sección anterior, el comando para cerrarla sería:

```
$ sudo hcitool -i hci0 ledc 3585
```

Como ya se ha comentado el potencial de la herramienta `hcitool` es mucho mayor del aquí indicado, por lo que se remite al lector a la página `man` de la herramienta, o a su ayuda en línea, para una mayor información.

1.2.3. `gatttool`

`gatttool` es una herramienta que permite la interacción con un dispositivo Bluetooth. La herramienta puede establecer una conexión, descubrir los servicios y características del servidor, así como la leer y/o escribir valores asociados a las características de los servicios del dispositivo Bluetooth.

Cada una de las acciones anteriores se puede realizar mediante líneas de comando, pero también dispone de un **modo interactivo**, que será sobre el que nos centremos en esta sección. Para una descripción más detallada de la herramienta, y su uso mediante línea de comandos, se remite al lector a la ayuda en línea de la aplicación la cual es accesible mediante el comando:

```
$ gatttool --help
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

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
```

Para lanzar en **modo interactivo** la aplicación `gatttool` se deberá ejecutar el siguiente comando:

```
$ sudo gatttool -i <hciX> -b <MAC_dispositivo> -I
```

donde `<hciX>` indica la interfaz Bluetooth utilizada para interactuar con el dispositivo Bluetooth, `<MAC_dispositivo>` indica la dirección MAC del dispositivo Bluetooth al que nos queremos conectar. Esta dirección MAC se pude obtener mediante la realización de un escaneo; por ejemplo, usando la herramienta `hcitool -i <hciX>lescan`.

La entrada en el modo interactivo también se pude hacer sin la especificación del parámetro `-b <MAC_dispositivo>`, pero en este caso deberemos indicar explícitamente la dirección MAC del dispositivo Bluetooth con el que queramos interactuar dentro del modo interactivo de `gatttool`. Por ello, es más habitual, seleccionar antes de lanzar `gatttool` en modo interactivo, el dispositivo Bluetooth con el que trabajaremos y así indicar su MAC a la hora de ejecutar la herramienta `gatttool`. De esta forma un ejemplo de uso común sería:

```
$ sudo gatttool -i hci0 -b C4:BE:84:E5:69:9D
[DB:AC:3C:29:6B:09][LE]>
```

Como se puede ver, la aplicación nos devuelve un *prompt* donde nos indica la MAC del dispositivo por defecto con el que interactuará, y que dicho dispositivo el `LE`.

Aunque se ha llamado a `gatttool` con la opción `-b <MAC_dispositivo>`, esto no implica que se haya establecido una conexión con el dispositivo, co-

mo ya se ha dicho. La conexión se ha de solicitar explícitamente una vez inicializado el método interactivo.

Ahora pasaremos a detallar las acciones más habituales a realizar con la herramienta `gatttool` en modo interactivo.

Conexión con el dispositivo

Para el establecimiento de la conexión se utiliza el comando interactivo:

```
connect [<MAC_address>]
```

Si el modo interactivo se ha lanzado con la opción `-b <MAC_dispositivo>`, y nos queremos conectar a dicho dispositivo, no es necesario el indicar el parámetro `<MAC_address>`. Sí será necesario indicarlo si: queremos conectarnos a un dispositivo distinto, o si hemos iniciado el modo interactivo sin la opción `-b <MAC_dispositivo>`. A continuación se muestran ejemplos del uso de este comando:

```
[DB:AC:3C:29:6B:09][LE]> connect C4:BE:84:E5:69:9D
Attempting to connect to C4:BE:84:E5:69:9D
Connection successful
[C4:BE:84:E5:69:9D][LE]>
```

donde nos hemos conectado a un dispositivo distinto al que utilizamos para comenzar la sesión interactivo. Y usando el mismo dispositivo sería:

```
[C4:BE:84:E5:20:56][LE]> connect
Attempting to connect to C4:BE:84:E5:20:56
Connection successful
[C4:BE:84:E5:20:56][LE]>
```

Una vez que hemos establecido la conexión con un servidor, podremos interactuar con él y sus servicios.

Descubrimiento de servicios

El descubrimiento de servicios se realiza con el comando `primary`, como se puede ver en el siguiente ejemplo:

```
[C4:BE:84:E5:20:56][LE]> primary
attr handle: 0x0001, end grp handle: 0x0009 uuid:
   f000ffc0-0451-4000-b000-000000000000
attr handle: 0x000a, end grp handle: 0x0014 uuid:
   00001800-0000-1000-8000-00805f9b34fb
```

```
attr handle: 0x0015, end grp handle: 0x0018 uuid:
   00001801-0000-1000-8000-00805f9b34fb
attr handle: 0x0019, end grp handle: 0x002b uuid: 0000180
   a-0000-1000-8000-00805f9b34fb
attr handle: 0x002c, end grp handle: 0x0030 uuid:
   a495ff10-c5b1-4b44-b512-1370f02d74de
attr handle: 0x0031, end grp handle: 0x0045 uuid:
   a495ff20-c5b1-4b44-b512-1370f02d74de
attr handle: 0x0046, end grp handle: 0xffff uuid: 0000180
   f-0000-1000-8000-00805f9b34fb
[C4:BE:84:E5:20:56][LE]>
```

El cual devuelve una lista con cada uno de los servicios primarios que ofrece el dispositivo Bluetooth. Por cada servicio (línea) se especifica el *handle* inicial y final que dicho servicio tiene asignados y su UUID.

Dichos UUID se pueden utilizar para averiguar de que servicio se trata. Ésto solo se puede hacer con aquellos servicios estandarizados, los cuales se identifican por tener un UUID de 16-bits: todos los que acaban en `UUID-0000-1000-8000-00805f9b34fb`. Siendo en dicho caso el valor de `UUID` el que habría que buscar en [2] para determinar de qué servicio se trata. Para los servicios no estandarizados habría que ver las especificaciones del fabricante del dispositivo Bluetooth para descubrir el tipo de servicio de que se trata.

El comando `primary` también se puede llamar dándole como parámetro el `UUID` del servicio a descubrir. En dicho caso el comando solo devolverá información sobre los *handle* inicial y final que el servicio tiene asignados. Para un UUID de 16-bits (estandarizado) sería:

```
[C4:BE:84:E5:20:56][LE]> primary 1800
Starting handle: 0x000a Ending handle: 0x0014
[C4:BE:84:E5:20:56][LE]>
```

en el caso de un UUID de 128-bits, no estandarizado y asignado por el fabricante, el comando sería:

```
  [C4:BE:84:E5:20:56][LE]> primary f000ffc0-0451-4000-
     b000-000000000000
  Starting handle: 0x0001 Ending handle: 0x0009
  [C4:BE:84:E5:20:56][LE]>
```

Descubrimiento de características

Las características asociadas a un servicio se descubren mediante el comando:

```
  characteristics [start_hnd] [end_hnd [UUID]]]
```

donde, los valores `start_hnd` y `end_hnd` indican el rango de *handles* entre los que quiero descubrir características. En el caso de no especificar dichos valores el comando tomará `start_hnd=0x0001` y `end_hnd=0xffff`.

Lo normal es que se utilicen como valores para `start_hnd` y `end_hnd`, los obtenidos en el descubrimiento de servicios. De esta forma, descubrimos las características asociadas a un servicio en concreto. Por ejemplo, para el caso anterior y el servicio con `UUID=0x180a`, cuyo *handles* en el intervalo `[0x0019, 0x002b]`, sus características se obtendrían con el siguiente comando:

```
 [C4:BE:84:E5:20:56][LE]> characteristics 0019 002b
handle: 0x001a, char properties: 0x02, char value handle:
    0x001b, uuid: 00002a23-0000-1000-8000-00805f9b34fb
handle: 0x001c, char properties: 0x02, char value handle:
    0x001d, uuid: 00002a24-0000-1000-8000-00805f9b34fb
handle: 0x001e, char properties: 0x02, char value handle:
    0x001f, uuid: 00002a25-0000-1000-8000-00805f9b34fb
handle: 0x0020, char properties: 0x02, char value handle:
    0x0021, uuid: 00002a26-0000-1000-8000-00805f9b34fb
handle: 0x0022, char properties: 0x02, char value handle:
    0x0023, uuid: 00002a27-0000-1000-8000-00805f9b34fb
handle: 0x0024, char properties: 0x02, char value handle:
    0x0025, uuid: 00002a28-0000-1000-8000-00805f9b34fb
handle: 0x0026, char properties: 0x02, char value handle:
    0x0027, uuid: 00002a29-0000-1000-8000-00805f9b34fb
handle: 0x0028, char properties: 0x02, char value handle:
    0x0029, uuid: 00002a2a-0000-1000-8000-00805f9b34fb
handle: 0x002a, char properties: 0x02, char value handle:
    0x002b, uuid: 00002a50-0000-1000-8000-00805f9b34fb
[C4:BE:84:E5:20:56][LE]>
```

Analizando la salida, podemos ver que se obtienen los siguientes datos:

- `handle`: es el *handle* asociado a la característica.
- `char properties`: son las propiedades asignadas a la característica. Una interpretación de este campo se puede hacer utilizando la Tabla

3.5 de [3]-Parte G, sección 3.3.1.1.

- `char value handle`: es el *handle* a utilizar si queremos acceder al valor del dato asociado a la característica.

- `uuid`: es el valor del UUID asociado a la característica.

  Este UUID, al igual que en el caso de los servicios, se puede usar para descubrir que tipo de característica es la descubierta. Ésto solo se puede hacer con aquellas características estandarizadas, las cuales se identifican por tener un UUID de 16-bits, que son todos los que acaban en `UUID-0000-1000-8000-00805f9b34fb`. Siendo en dicho caso el valor de `UUID` el que habría que buscar en [1] para determinar de qué característica se trata. Para el caso de características no estandarizadas, habría que consultar la documentación del fabricante del dispositivo Bluetooth, para buscar en ella el UUID de la característica.

Cabe la posibilidad de llamar al comando anterior especificando una valor de UUID de característica. En dicho caso, la salida anterior será filtrada por dicho UUID, siendo presentada por pantalla solo aquella que tenga dicho UUID. Por ejemplo:

```
[C4:BE:84:E5:20:56][LE]> characteristics 0019 002b 2a28
handle: 0x0024, char properties: 0x02, char value handle:
    0x0025, uuid: 00002a28-0000-1000-8000-00805f9b34fb
[C4:BE:84:E5:20:56][LE]>
```

Lectura de valores

Para la lectura de los valores asociados a las características, lo primero que hay que hacer es comprobar que dicho valor tiene derecho de lectura. Para ello hay que analizar el campo `char properties` (descrito en la sección anterior), devuelto en el proceso de descubrimiento de la característica, según las indicaciones de [3]-Parte G, sección 3.3.1.1.

Una vez comprobado que el valor es accesible para lectura, el acceso a dicho valor puede realizarse utilizando el `handle` o el `uuid` devuelto durante el proceso de descubrimiento.

Lectura mediante *handle*: el comando a utilizar es:

```
char-read-hnd <handle>
```

siendo <handle> el valor del *handle* asociado a la característica en cuestión. Un ejemplo de salida se puede ver a continuación:

```
[C4:BE:84:E5:20:56][LE]> char-read-hnd 002b
Characteristic value/descriptor: 01 0d 00 00 00 10 01
[C4:BE:84:E5:20:56][LE]>
```

el valor es devuelto en formato hexadecimal, por lo que para su interpretación se tendrá que tener en cuenta el tipo de característica de que se trata, bien por haber consultado [1] en el caso de tener un UUID de 16 bits; o por consultar la documentación del fabricante si el UUID de la característica es de 128 bits.

Lectura mediante `uuid`: en este caso el comando a usar es:

```
[C4:BE:84:E5:20:56][LE]> char-read-uuid <UUID>
```

para <UUID> el valor del UUID de la característica en cuestión.

Un ejemplo de uso es:

```
[C4:BE:84:E5:20:56][LE]> char-read-uuid 2a2a
handle: 0x0029   value: fe 00 65 78 70 65 72 69 6d 65 6e
   74 61 6c
[C4:BE:84:E5:20:56][LE]>
```

que como vemos tiene una salida similar al caso anterior, solo que ahora especifica el `handle` del valor en la salida para su posterior uso.

Escritura de valores

La escritura de valores se podrá realizar si dichos valores poseen la propiedad de ser modificados. Determinar si un valor puede ser escrito o no se debe realizar antes de intentar la escritura. Para ello se analizará el valor del parámetro `char properties`, devuelto en el proceso de descubrimiento de la característica, siguiendo las indicaciones de [3]-Parte G, sección 3.3.1.1.

Si una característica se puede escribir, las operaciones de escritura se pueden realizar de dos formas, con confirmación o sin ella. La diferencia es que si realizamos una escritura sin confirmación, el servidor no nos indicará el éxito o fracaso de la operación. Mientras que una escritura con confirmación va acompañada de un mensaje de éxito o error de la operación.

Escritura sin confirmación: Para realizarla se utiliza el siguiente comando:

```
[C4:BE:84:E5:20:56][LE]> char-write-cmd <handle> <
    hex_value>
```

siendo `<handle>` el *handle* asociado a la característica a escribir, y `hex_value` el valor a escribir, que deberá de ir en formato hexadecimal; por lo que en la mayoría de los casos se requiere una conversión del valor a escribir a formato hexadecimal. Esta conversión se debe realizar por el usuario, ya que la aplicación no la hace.

Un ejemplo de uso se muestra a continuación:

```
[C4:BE:84:E5:20:56][LE]> char-write-cmd 0x0033 00ff00
[C4:BE:84:E5:20:56][LE]>
```

Como se observa al comando se le pasa como primer parámetro el *handle* de acceso al valor a escribir, y como segundo el valor en hexadecimal del dato a escribir. Como en el caso de la lectura de valores, el *handle* se obtiene del parámetro `handle` obtenido durante el proceso de descubrimiento de la característica.

Al ser una escritura sin confirmación, el servidor no indica nada al cliente, como se ve, no existe ningún mensaje de error o éxito tras la operación de escritura.

Escritura con confirmación: En este caso, similar al anterior, el comando a ejecutar es:

```
[C4:BE:84:E5:20:56][LE]> char-write-req <handle> <
    hex_value>
```

que como se ve posee los mismos parámetros que el comando para la realización de una escritura sin confirmación visto en la sección anterior. A continuación se muestra un ejemplo de uso:

```
[C4:BE:84:E5:20:56][LE]> char-write-req 0x0033 00ff00
Characteristic value was written successfully
[C4:BE:84:E5:20:56][LE]>
```

Se aprecia como se indica tanto el *handle* del valor a modificar, como el dato a escribir (`00ff00` en hexadecimal). Y como se puede apreciar, tras la realización de la escritura se muestra un mensaje relacionado con el éxito o

fracaso de la escritura del dato. El cual, procede de la confirmación enviada por el servidor.

1.2.4. `hcidump`

La aplicación `hcidump` nos permite obtener y analizar la actividad Bluetotth que una interfaz en concreto. Una descripción detallada de la misma se puede encontrar en la página `man` de la aplicación.

Normalmente la aplicación se utiliza para obtener las trazas del tráfico generado y recibido en una interfaz determinada, para ello se utilizará el siguiente comando:

```
$ sudo hcidump -i <hciX>
```

donde `<hciX>` indica la interfaz Bluetooth que analizar. En este caso la aplicación mostrará por pantalla cada uno de los mensajes Bluetooth emitidos y/o recibidos por dicha interfaz.

Es habitual, que lo que nos interese es el análisis de dichos mensajes. Por ello es conveniente el guardar todo lo emitido y/o recibido por la interfaz en un fichero, el cual más tarde puede ser analizado utilizando una herramienta de análisis de protocolos como `wireshark` [5]. El almacenamiento de los mensajes emitidos y recibidos se realiza con el uso del parámetro `-w <file>`, quedando el comando:

```
$ sudo hcidump -i hciX -w <file>
```

donde, al igual que antes, `hciX` indica la interfaz a analizar, y `<file>` es el fichero donde guardar lo capturado por la interfaz.

Referencias

[1] Bluetooth SIG, Inc. Characteristics | Bluetooth Development Portal. `https://developer.bluetooth.org/gatt/characteristics/Pages/CharacteristicsHome.aspx`. Accedido: 2016-03-16.

[2] Bluetooth SIG, Inc. Services | Bluetooth Development Portal. `https://developer.bluetooth.org/gatt/characteristics/Pages/ServicesHome.aspx`. Accedido: 2016-03-16.

[3] Bluetooth SIG, Inc. Specification of the Bluetooth System. Covered Core Package version: 4.2. Vol.3 - Core System Package [Host volume]. Technical report, Bluetooth SIG, Inc., 2014.

[4] BlueZ Project. BlueZ - Official Linux Bluetooth protocol stack. `www.bluez.org`. Accedido: 17-02-2016.

[5] Wireshark Fundation. Wireshark - Go Deep. `www.wireshark.org`. Accedido: 21-03-2016.

[6] Libelium Comunicaciones Distribuidas, S.L. Bluetooth Extreme USB Dongle. https://www.cooking-hacks.com/documentation/tutorials/bluetooth... Accedido: 17-02-2016.

[7] Zahid Adeel. Anonymize Bluetooth Devices On Linux Using hciconfig Utility. http://exploiterz.blogspot.com.es/2013/08/how-to... Accedido: 17-02-2016.