Universidad Carlos III de Madrid

Ingeniería de Telecomunicación

Manual de Usuario

BLUC3M

Autor: William D. Wallace San Paulo

Julio de 2006

Tutor: Dr. José Ignacio Moreno Novella

Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid

ÍNDICE:

I. Introducción:

El objetivo de BLUC3M es proporcionar una librería de alto nivel para la comunicación entre dispositivos Bluetooth en un entorno GNU/Linux, evitando tener que tratar directamente con las librerías de desarrollo proporcionadas por BlueZ, que requieren del uso intensivo de programación con sockets.

Uno de los criterios de diseño ha sido la integración con la librería Linpal, creada para la comunicación de dispositivos a través de Ethernet. De hecho, en combinación con Linpal, BLUC3M permite la comunicación de terminales Bluetooth con terminales de una LAN o de Internet de una forma transparente y sin añadir demasiada complejidad. Actualmente Blucem sólo ofrece comunicaciones sobre L2CAP.

II. Dependencias:

Para el funcionamiento de BLUC3M es necesario tener instalada la librería BlueZ. Esta librería se encuentra ya en versiones recientes del kernel de Linux, por lo tanto antes de su instalación se recomienda verificar si ya se encuentra disponible. La versión más reciente de esta librería puede obtenerse a través de http://www.bluez.org/ , aunque se recomienda la instalación mediante paquetes precompilados:

http://www.bluez.org/packages.html

III. Compilación:

Todavía no se ha migrado la librería a un paquete instalable, por lo tanto será necesario incluir en la cabecera un #include “bluc3m.h”, donde bluc3m.h es el fichero de cabecera de la librería y que debería encontrarse al mismo nivel que el resto del código, al igual que el fichero bluc3m.c que contiene el código fuente de la librería. Para compilar:

gcc -Wall –o ejecutable fichero_main.c -lbluc3m –lbluetooth

IV. Man Page:

Todavía no disponible.

V. Comandos proporcionados por BlueZ:

BlueZ pone a nuestra disposición varios comandos para la configuración de los dispositivos Bluetooth conectados a nuestro Terminal, a continuación se listan los más importantes junto con una pequeña descripción:

Para la ejecución de los comandos hciconfig es necesario utilizar una consola con privilegios de superusuario.

HCICONFIG

hciconfig	Lista los adaptadores conectados
hciconfig –a	Lista los adaptadores conectados de una forma más detallada.
hciconfig hciX	Realiza el comando sobre el dispositivo hciX, donde X es su número de identificación. Estos números son asignados en orden consecutivo desde el cero hasta el número de dispositivos conectados menos uno.
hciconfig hciX up	Abre e inicializa el dispositivo.
hciconfig hciX down	Apaga el dispositivo.
hciconfig hciX reset	Vuelve a la configuración por defecto del dispositivo.
hciconfig hciX rstat	Pone a cero las estadísticas del adaptador.
hciconfig hciX auth	Habilita la autenticación.
hciconfig hciX noauth	Deshabilita la autenticación.
hciconfig hciX encrypt	Habilita la encriptación de las comunicaciones
hciconfgi hciX noencrypt	Deshabilita la encriptación de las comunicaciones.
hciconfig hciX iscan	Permite el rastreo de dispositivos mediante Inquiry.
hciconfig hciX noscan	Deshabilita el rastreo de dispositivos.
hciconfig hciX lm MODE	Donde MODE puede tomar dos valores: MASTER o SLAVE. Configura el adaptador al modo elegido.
hciconfig hciX name NOM	Donde NOM es el nombre que le queremos dar al dispositivo, si no indicamos ninguno nos devolverá el nombre asignado.

Para una lista completa de los comandos hciconfig se puede ejecutar hciconfig –help.

Es siempre necesario que haya al menos un dispositivo MASTER, y que este esté en funcionamiento (UP).

Para la ejecución de comandos hcitool no es necesario poseer privilegios de superusuario. A continuación se listan algunos, aunque la propia librería permite ejecutar la mayoría a través de distintos métodos.

HCITOOL

hcitool –i dev	Ejecuta el comando sobre el dispositivo dev (hciX)
hcitool dev	Lista los adaptadores disponibles.
hictool scan	Devuelve una lista con los dispositivos remotos con los que podemos conectarnos.
hcitool name	Devuelve el nombre de un dispositivo remoto.
hcitool info	Devuelve información detallada sobre un dispositivo remoto.
hictool con	Lista las conexiones activas.
hcitool cc	Establece una conexión con otro dispositivo.
hcitool sr	Cambia entre el modo MAESTRO y ESCLAVO.
hcitool auth	Pide la autenticación.
hcitool enc	Establece encriptación de datos en el enlace.
hcitool key	Cambia la clave de autenticación.

Para un resumen completo sobre los comandos hcitool se puede ejecutar **hcitool** ó **hcitool -- help**.

Sobre ambos comandos existe abundante información en Internet, se recomienda visitar la dirección http://holtmann.org/linux/bluetooth para acceder a diversos artículos relacionados.

VI. Resumen de tipos de datos:

Información sobre un dispositivo remoto, incluye solamente dirección en formato binario y nombre.

```
struct BlucemPeerInfo
{
        char addr [Blucem_MAC_BIN_LENGTH];
        char name [Blucem_MAX_NAME_LENGTH];

} __attribute__ ((packed));
```

Estructura de datos que encapsula la información sobre un Adaptador, incluye un entero como identificador del dispositivo, el nombre asignado por el Terminal (hciX, a no confundir con el nombre presentado a dispositivos remotos) y su dirección MAC en formato binario.

```
struct BlucemAdapter
{
        unsigned short int ident;
        char name[8];
        char addr[Blucem_MAC_BIN_LENGTH];
}  __attribute__ ((packed));
```

Estructura que representa un paquete de datos de forma muy simplificada, sólo se tienen en cuenta los campos de dirección MAC origen, dirección MAC destino (ambas en formato binario) y datos a enviar. Es importante destacar que será necesario reservar memoria antes de usar esta estructura mediante la función malloc.

```
struct BlucemDataPacket
{
        char dest_addr [Blucem_MAC_BIN_LENGTH];
        char source_addr [Blucem_MAC_BIN_LENGTH];
        char* data;
}  __attribute__ ((packed));
```

VII. Resumen de Constantes:

Número máximo de dispositivos con los que podemos mantener una conexión abierta:
#define BLUC3M_MAX_PEERS 50

Tamaño de direcciones Binarias MAC y de la dirección de su representación en cadena de caracteres.
#define Blucem_MAC_BIN_LENGTH 6
#define Blucem_MAC_STR_LENGTH 19

Longitud de la cadena de caracteres que representa el nombre de un dispositivo.
#define Blucem_MAX_NAME_LENGTH 250

Número máximo de adaptadores permitidos en un solo terminal (no se recomienda en ningún caso superar los dos adaptadores por Terminal).
#define Blucem_MAX_ADAPTERS 10

Constantes para obtener la estadística deseada de la función BLUC3MLocalAdapterGetStat:

```
#define BLUC3M_RXERRORS 0
#define BLUC3M_TXERRORS 1
#define BLUC3M_CMDTX 2
#define BLUC3M_EVTRX 3
#define BLUC3M_ACLTX 4
#define BLUC3M_ACLRX 5
#define BLUC3M_SCOTX 6
#define BLUC3M_SCORX 7
#define BLUC3M_BYTERX 8
#define BLUC3M_BYTETX 9
```

Punteros a las estructuras de adaptador, información sobre dispositivo remoto y paquete de datos:

```
#define BPADAPTER struct BlucemAdapter *
#define BPPEERINFO struct BlucemPeerInfo *
#define BPDATAPACKET struct BlucemDataPacket *
```

VIII. Resumen de funciones:

8.1 Configuración e información del adaptador:

`int BlucemGetLocalAdapter ( BPAdapter adapter )`

Descripción:	Deja en la estructura a la que apunta el puntero pasado por parámetro los datos del adaptador que se está utilizando.
Entrada:	Puntero a una estructura de adaptador.
Salida:	1 en caso de que el adaptador estuviese configurado correctamente, -1 en caso de error.

`int BlucemGetLocalAdapters ( struct BlucemAdapter adapters [Blucem_MAX_ADAPTERS] )`

Descripción: Devuelve los datos de los adaptadores disponibles.

Entrada: Array de estructuras BlucemAdapter dónde se dejarán los datos de los adaptadores disponibles.

Salida: Devuelve el número de adaptadores encontrados, no hay errores.

int BlucemPrintLocalAdapters ( )

Descripción: Imprime por pantalla una lista de todos los adaptadores disponibles, incluyendo identificador, nombre (hciX) y dirección MAC.

Salida: Devuelve el número de adaptadores encontrados, no hay errores.

int BlucemSelectAdapter ( )

Descripción: Imprime por pantalla la lista de adaptadores disponibles, pide al usuario que seleccione uno y devuelve el identificador del adaptador elegido.

Salida: Devuelve el identificador del adaptador elegido, -1 en caso de que la opción elegida por el usuario no sea correcta.

int BlucemSetupLocalAdapter (int dev_id,
BPADAPTER adapter)

Descripción: Configura el adaptador elegido como dispositivo predeterminado para las comunicaciones con otros terminales.

Entrada: dev_id: Identificador del adaptador.
adapter: puntero a una estructura adaptador donde se dejarán los datos del adaptador seleccionado.

Salida: 1 en caso de éxito, -1 en caso de error.

int BlucemGetLocalAddres (char *dest)

Descripción: Devuelve la dirección del dispositivo configurado en SetupLocalAdapter en formato binario.

Entrada: Puntero a una posición de memoria en la que dejar la dirección en formato binario.

Salida: 1 en caso de éxito, -1 en caso de error.

int BlucemLocalAdapterGetStat (int dev_id,
int opt,
unsigned long *value)

Descripción: Devuelve la estadística pedida del adaptador elegido.

Entrada: dev_id: Identificador del adaptador.
opt: entero que representa la estadística elegida (Ver constantes).
*value: puntero a un entero de 32 bits sin signo donde dejar la estadística pedida.

Salida: 1 en caso de éxito, -1 en caso de error.

int BlucemCloseAdapter ( )

Descripción: Cierra el puerto de escucha si estuviese abierto, termina las conexiones con otros equipos y finalmente pone a cero los valores del adaptador usado para comunicaciones (NO apaga el adaptador, otras aplicaciones pueden seguir usándolo).

Salida: 1 siempre, aunque no hubiese ningún adaptador configurado.

8.2 Configuración de las comunicaciones:

int BlucemOpenPort (short int port)

Descripción: Abre el puerto seleccionado del adaptador que se ha configurado para las comunicaciones.

Entrada: Puerto que se desea abrir, es necesario investigar antes qué puertos están ya ocupados por otros servicios.

Salida: 1 en caso de éxito, -1 en caso de error.

int BlucemSetMTU (int output_mtu, int input_mtu)

Descripción: Establece la MTU de datos de las comunicaciones salientes y entrantes. Es necesario haber configurado el puerto antes de ejecutar este método. Si no se especifica ningún valor de MTU mediante este método, se tomará el valor por defecto de 672 bytes.
La MTU puede tomar valores entre 48 y 65 535 bytes, aunque no se recomienda exceder los 2500 bytes.

Entrada: Mtu de salida y de entrada respectivamente.

Salida: 1 en caso de éxito, -1 en caso de error.

int BlucemClosePort ( )

Descripción: Cierra todas las conexiones activas y cierra el puerto de escucha. También reestablece la MTU a su valor por defecto (672 bytes).

Salida: 1 en caso de éxito, -1 en caso de error.

8.3 Información sobre otros dispositivos:

int BlucemGetAllPeers (struct BlucemPeerInfo peers [Blucem_MAX_PEERS])

Descripción: Devuelve una lista con todos los dispositivos encontrados en nuestro radio de alcance.

Entrada: Puntero a un array donde dejar la información sobre los equipos encontrados.

Salida: Número de dispositivos encontrados, -1 en caso de error.

int BlucemGetPeerName (char addr [Blucem_MAC_BIN_LENGTH], char *name)

Descripción: Devuelve el nombre de un dispositivo remoto.

Entrada: addr: Array en el que figura la dirección del dispositivo en formato binario.
*name: Puntero a una posición de memoria dónde dejar el nombre del dispositivo.

Salida: 1 en caso de éxito, -1 en caso de error.

int BlucemSelectPeer (BPPEERINFO selected_peer)

Descripción: Imprime por pantalla los dispositivos que se encuentran dentro de nuestro radio de alcance y pide la entrada por teclado del usuario para elegir uno.

Entrada: Puntero a una estructura donde dejar los datos del dispositivo elegido.

Salida: Número de dispositivos encontrados, -1 en caso de error.

int BlucemPrintAllPeers ( )

Descripción: Imprime por pantalla una lista de todos los dispositivos dentro del radio de alcance.

Salida: Número de dispositivos encontrados, -1 en caso de error.

8.4 Conexión y Desconexión de equipos remotos:

int BlucemConnect (char *dest, short int port)

Descripción: Establece una conexión con un equipo remoto.

Entrada: *dest: Puntero a una posición de memoria con la MAC en formato binario del equipo remoto.
port: Puerto al que conectarse.

Salida: 1 en caso de éxito, -1 en caso de fallo.

int BlucemListen ( )

Descripción: Escucha nuevas peticiones de conexión, desconexión e intercambio de datos.

Salida:

- 2 en caso de petición de desconexión o de envío de datos (es necesario llamar a BlucemRead para gestionar este evento).
- 1 en caso de petición de conexión, gestionada automáticamente.
- 0 en caso de que no haya habido eventos.
- -1 en caso de error.

int BlucemDisconnectFromPeers ( )

Descripción: Se desconecta de todos los equipos remotos pero se mantiene a la escucha de nuevas conexiones.

Salida: 1 en caso de éxito, -1 en caso de error.

int BlucemDisconnectFromPeer (char *addr)

Descripción: Se desconecta del equipo remoto elegido.

Entrada: Puntero a una posición de memoria que contenga la dirección MAC en formato binario del dispositivo remoto del que nos queremos desconectar.

Salida:

- -1 en caso de que el adaptador y/o el puerto no estuviesen configurados.
- 0 en caso de no haber encontrado el dispositivo.
- 1 en caso de éxito.

8.5 Intercambiado información:

int BlucemRead (BPDATAPACKET bp_packet)

Descripción: Lee la información enviada por otro dispositivo o gestiona su petición de desconexión de forma automática.

Entrada: Puntero a un paquete de datos donde dejar la información leída o la dirección del dispositivo que se ha desconectado.

Salida:

- -1 en caso de error.
- 0 en caso de haber recibido una petición de desconexión, dejando la dirección del equipo en formato binario en memoria.
- 1 en caso de haber recibido datos.

int BlucemSend (BPDATAPACKET bp_packet, int byte_length)

Descripción: Envía datos al último dispositivo con el que hemos intercambiado información, ya sea una conexión, recepción o envío de datos.

Entrada: Puntero a un paquete de datos, el tamaño del campo de datos no debe exceder la MTU de salida.
Entero que representa la cantidad de bytes de datos a enviar.

Salida:

- -1 en caso de error.
- 0 en caso de que no se haya enviado nada.
- 1 en caso de que se hayan enviado datos.

8.6 Información sobre dispositivos conectados:

int BlucemGetConnectedPeers (struct BlucemPeerInfo peers [Blucem_MAX_PEERS])

Descripción: Devuelve una lista con los dispositivos con los que tenemos una conexión activa.

Entrada: Array donde dejar los datos sobre los dispositivos conectados.

Salida: Devuelve el número de dispositivos conectados, -1 en caso de error.

int BlucemGetCurrentPeerAddress (char *dest)

Descripción: Devuelve la dirección en formato binario del último dispositivo con el que hemos intercambiado información, al que nos hemos conectado o que se ha conectado a nosotros.

Entrada: Puntero a un array donde dejar la dirección en formato binario del dispositivo.

Salida:

- -1 en caso de error.
- 0 en caso de que no haya dispositivos con los que nos hayamos comunicado últimamente (o la última petición haya sido de desconexión).
- 1 en caso de éxito.

int BlucemPrintConnectedPeers ( )

Descripción: Imprime por pantalla una lista de los dispositivos conectados.

Salida: Devuelve el número de dispositivos conectados, o un número negativo en caso de error.

8.7 Imprimir información por pantalla:

int BlucemPrintError (char *msg)

Descripción: Imprime un mensaje de error pasado por parámetro, seguido de los detalles de ese error. Todos los métodos que devuelven -1 en caso de error establecen una variable interna con los detalles del error ocurrido.

Entrada: Puntero al mensaje a imprimir, que irá seguido de los detalles del error.

Salida: Siempre 1.

int BlucemPrintCurrentPeerAddress ( )

Descripción: Imprime por pantalla, sin ningún retorno de carro antes o después, la dirección del último dispositivo con el que intercambiamos información, al que nos conectamos o que se conectó a nosotros.

Salida: Siempre 1.

int BlucemPrintLocalAddress ( )

Descripción: Imprime por pantalla, sin ningún retorno de carro antes o después, la dirección del adaptador que estamos utilizando para comunicarnos con los dispositivos remotos.

Salida: Siempre 1.

8.8 Métodos Auxiliares:

int BlucemStr2MAC (char *str, char *mac)

Descripción: Convierte una cadena de caracteres representando una dirección a su equivalente en binario.

Entrada: *str: Puntero a la cadena de caracteres que representa la dirección MAC en formato legible.
*mac: Puntero a la cadena que representa la dirección MAC en formato binario.

Salida: Negativa en caso de error, positiva en caso de éxito.

int BlucemMAC2Str (char *mac, char *str)

Descripción: Convierte una dirección MAC en formato binario a una cadena de caracteres legible.

Entrada: *mac: Puntero a la cadena que representa la dirección MAC en formato binario.
*str: Puntero a la cadena de caracteres que representa la dirección MAC en formato legible.

Salida: Negativa en caso de error, positiva en caso de éxito.

IX. Librerías ya incluidas en el paquete:

```
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <bluetooth/bluetooh.h>
#include <bluetooth/l2cap.h>
#include <bluetooth/hci.h>
#include <bluetooth/hci_lib.h>
```

X. Bibliografía:

10.1 Programación con BlueZ:

- **An Introduction to Bluetooth programming in GNU/Linux**
  - ***Albert Huang, MIT***
    - http://people.csail.mit.edu/albert/bluez-intro/

- **Documentación general sobre BlueZ**
  - ***Marcel Holtmann y otros.***
    - http://www.holtmann.org/linux/bluetooth/

- **Lista de Correo BlueZ-devel**
  - ***Marcel Holtmann y otros.***
    - http://www.bluez.org/lists.html

- **Página web con todo el código fuente de BlueZ, que permite una mejor comprensión de este entorno de programación:**
  - ***Desconocido.***
    - http://www-md.e-technik.uni-rostock.de/ma/hm27/uebung_hnp_2004/documentation/bluez-libs-2.10-html/

10.2 Uso general del lenguaje C:

- **Varias páginas web con código fuente, ficheros de cabecera y descripción de los principales métodos y estructuras, entre ellas:**
    - www.cplusplus.com
    - http://man.he.net/

10.3 Programación con Sockets:

- **Guía extremadamente útil y completa de programación con sockets en lenguaje C.**
  - ***Desconocido.***
    - http://beej.us/guide/bgnet/
    - http://www.arrakis.es/~dmrq/beej/home.htm (Español)