UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

ANALISIS Y DISEÑO BASADOS EN LA TECNOLOGIA BLUETOOTH

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

JONATAN ROBERTO GODINEZ GUTIERREZ

DIRECTOR DE TESIS: ING. ROBERTO MANDUJANO WILD

CIUDAD UNIVERSITARIA

AGOSTO DE 2004

CONTENIDO

AGRADECIMIENTOS

Este trabajo de tesis es un pequeño agradecimiento para todas las personas que han creído en mí.

Gracias a mis padres por haberme dado vida, amor y educación y al gran esfuerzo que han hecho para que pueda realizar todos mis sueños; gracias por su gran ejemplo de llevar una vida honrada y feliz. Gracias a mi papá por darme confianza, por darme apoyo incondicional en todas mis decisiones, por ser mi guía y mi ejemplo a seguir. Gracias a mi mamá por ser la crítica de mis acciones, por enseñarme a valorar mi vida, por hacerme reflexionar y por hacerme más fuerte.

A mi hermana Vanessa por darme amor y cariño, por ayudarme con mis deberes de la casa, por compartir algunos momentos de locura y sobre todo por recordarme que tengo que cuidar mis momentos de neürosis.

A mi hermano David por darme también amor y cariño, por aguantarme en la casa y por ser un motivo de ser un buen ejemplo a seguir.

A mi madrina Lucila y a mi tía Rosa, que han sido como unas madres para mí, por creer en mí y por todo su apoyo en todo sentido, gracias por su ejemplo y sobre todo, gracias por hacerme sentir especial.

A toda mi familia por haberme dado momentos tan felices y por demostrarme que son respaldo incondicional para mis problemas.

A mi gran amiga Vanessa, la enemiga de mi soledad; por escucharme y por compartir desde hace 7 años, los momentos más felices y tristes de nuestras vidas.

A Arely, Liliana, Yuria, Alejandro, Erick, Humberto, Joel, Juan, Mario, Pepe y a todos los que han sido mis amigos, por todas las aventuras y problemas que hemos pasado juntos.

A Orlando por ofrecerme su amistad y por todas las recomendaciones que me han sacado de muchos apuros.

A todos los que me ayudaron en diferentes aspectos para realizar la Tesis y aportaron ideas como Manuel Moreno y Edgar López.

Al Ingeniero Roberto Mandujano de la Facultad de Ingeniería por su tiempo y por ayudarme a realizar este trabajo de investigación.

Y gracias a Dios por ser una fuente de fe y por haberme permitido vivir en este mundo.

CAPÍTULO 1.
INTRODUCCIÓN

Cuando es necesario disponer de movilidad en las comunicaciones, depender de un enlace físico como es un cable (en cualquiera de sus modalidades), supone una seria restricción para conseguir la plena libertad de movimientos.

Para evitar las restricciones derivadas de la utilización de cables, las conexiones inalámbricas se convierten en la alternativa ideal.

1.1 Justificación

Gracias a los estándares industriales emergentes, el mayor rendimiento y los costos en constante disminución, resulta cada vez más fácil justificar el despliegue de la tecnología inalámbrica; la cual se ha empezado a ver desde hace relativamente poco tiempo, y la cual puede significar una revolución en el uso de las tecnologías de información como se conocen en la actualidad.

La comunicación sin cables ha estado disponible desde hace bastante tiempo (en **1950** se establece el primer enlace de comunicaciones vía Microondas, proviendo comunicaciones en una alto volumen a muy grandes distancias) siendo su principal aplicación las comunicaciones de voz. Hoy en día, millones de personas utilizan los sistemas de radio de dos vías para comunicaciones de voz punto a punto o multipunto, sin embargo, en lo que se refiere a la transmisión de datos binarios, sólo recientemente se han desarrollado servicios inalámbricos para éstos a gran escala. El mundo de los denominados datos inalámbricos incluye enlaces fijos de Microondas, Redes de Área Local Inalámbricas (WLAN, Wireless Local Area Network), enlaces mediante satélites, redes de transmisión digital, rayos infrarrojos difusos, Sistema de Posicionamiento Global (GPS, Global positioning system) y muchas otras.

Dentro de las tecnologías inalámbricas que se encuentran hoy en día en el mercado, encontramos la tecnología **Bluetooth**, tecnología que pretende unir el campo de las telecomunicaciones y la computación. El principal objetivo de esta tecnología, es la posibilidad de reemplazar los cables de los diferentes dispositivos y periféricos por medio de un enlace de radio universal de corto alcance; pudiendo encontrar aquí un sin fin de aplicaciones, como establecer comunicación entre teléfonos móviles y computadoras, PDAs, audífonos inalámbricos y módems.

Cualquier dispositivo electrónico en general es susceptible de usar esta tecnología, solo es necesario entender la tecnología, y saber como implementarse.

1.2 Objetivos y contribuciones

Debido a que **Bluetooth** es una tecnología reciente; es difícil encontrar información técnica y de diseño en México y en general en Latinoamérica; por lo que este trabajo de investigación pretende mostrar los elementos necesarios para comprender esta tecnología, y ser una fuente de información para futuros trabajos acerca del tema.

El objetivo de este trabajo de investigación es estudiar la especificación **Bluetooth** y elaborar un documento teórico-práctico para poder entender, diseñar e implementar un dispositivo que utilice esta tecnología, y poder realizar pruebas que permitan comprobar prácticamente los conceptos fundamentales de la misma.

1.3 Estructura del documento

En el capítulo 2, se revisarán los diferentes tipos de tecnologías inalámbricas para comunicaciones digitales tanto de voz como de datos, se realizará una comparación y se mencionará los campos donde se utiliza cada una de ellas.

Se describirá la historia y conceptos generales de la tecnología **Bluetooth**; y por último se mostrarán algunos ejemplos de productos **Bluetooth** que existen en la actualidad.

En el capítulo 3, se estudiará la especificación del protocolo **Bluetooth** donde se mostrarán las capas de las que consta, su interacción y funcionamiento de las mismas, con el fin de comprender el funcionamiento de la especificación.

En el capítulo 4, se estudiarán los tipos de comunicación de datos que serán necesarios para la comunicación entre el módulo **Bluetooth** y el Host a la hora de la implementación.

En el capítulo 5 se llevará a cabo la implementación de un módulo **Bluetooth** y se realizarán pruebas, para comprobar prácticamente su funcionamiento y rendimiento.

En el capítulo 6 y último, se analizarán los resultados obtenidos de las pruebas y se harán las conclusiones acerca de la investigación realizada y fundamentada.

Con todos estos elementos se puede tener una visión teórica-práctica de lo que es la especificación **Bluetooth**, y los elementos que hay que tomar en cuenta tanto de software como de hardware para su implementación.

CAPÍTULO 2.
COMPUTACIÓN MÓVIL Y TECNOLOGÍAS INALÁMBRICAS

Dentro del enorme horizonte de las comunicaciones sin cables y la informática móvil, las redes inalámbricas van ganando rápidamente adeptos como una tecnología madura y fiable, que permite resolver los inconvenientes derivados de la propia naturaleza del cable como medio físico de enlace en las comunicaciones, muchos de ellos de vital importancia en el entorno del trabajo habitual.

Este capítulo introduce algunos de los conceptos relacionados con las redes inalámbricas y la diversidad de tecnologías que se están desarrollando, como IEEE 802.11, IrDA (Infrared Data Association), y **Bluetooth**, dejando un tanto de lado las redes Inalámbricas de Área Global (WWAN, Wireless Wide Area Network) por su extensión y complejidad.

La primera empresa que desarrolló y comercializó un sistema inalámbrico de conexión a redes fue NCR, cuando a finales de 1990 lanzó un sistema que utilizaba ondas de radio para interconectar computadoras. Desde entonces, la comunicación por redes inalámbricas ha dejado de ser experimental, para convertirse en una solución real a problemas concretos.

Aunque esta tecnología puede ser una alternativa para redes fijas de área local, no pretende reemplazar al sistema tradicional con cable, sino complementar aquellas situaciones en que es difícil disponer de una conexión.

2.1 Ventajas y desventajas de las redes inalámbricas

Las redes inalámbricas ofrecen las siguientes ventajas en productividad, conveniencia y el costo sobre las redes cableadas convencionales:

- **Movilidad:** El no depender de una conexión física a la red significa que el usuario está libre para desplazarse prácticamente por cualquier zona de una casa, oficina o edificio.

- **Productividad:** El usuario puede trabajar según sus necesidades (en una sala de reuniones, en el salón de clases, en la cafetería) sin perder su conexión a la red.

- **Flexibilidad:** El usuario puede reordenar su oficina, desde su hogar o en la empresa, sin preocuparse por el tendido de cables para las computadoras que quiera conectar a la red.

- **Portabilidad:** Si existe la necesidad de cambiar de edificio o establecer una ubicación temporal (por ejemplo una reunión fuera de la oficina, al montar un stand en una feria, o en labores de recuperación de catástrofes), sólo se tiene que transportar los equipos necesarios, sin tener que planear una infraestructura compleja, como lo requeriría una red cableada.

- **Facilidad de instalación:** Añadir más computadoras a una red requiere un tiempo y un esfuerzo mínimos.

- **Ahorros en costos y en tiempo:** La costosa y lenta instalación de cable se puede sustituir, actualizar o ampliar mediante una solución inalámbrica, para que esté instalado y funcionando en cuestión de horas y no de días.

Algunas de las deventajas generales de las tecnologías inalámbricas son:

- **Covertura:** En general se puede decir que estas tecnologías tienen una covertura más pequeña que las redes con cables debido al medio de transmisión.

- **Interferencias:** Debido a que algunas de estas tecnologías utilizan las mismas frecuencias son muy suceptibles a las interferencias entre ellas y también interferencias ocacionadas por el clima o el medio ambiente.

- **Seguridad:** Al no tener la necesidad de que el usuario esté fisicamente conectado a la red, existe el riesgo de que alguien externo pueda infiltrarse en la misma; por lo cual cada tecnología tiene que implementar medidas de seguridad para evitar esta situación.

2.2 La capa física y la capa de enlace

Las redes inalámbricas se diferencian del resto principalmente en la capa Física y en la capa de Enlace de datos según el modelo OSI (Sistema de Interconexión Abierta), ya que sustituyen al cable típico por otros métodos de naturaleza similar, pero muy bien diferenciados en su comportamiento, la transmisión por **radiofrecuencia** y la **luz infrarroja**.

Los sistemas por infrarrojos pueden clasificarse, a su vez, en sistemas de **corta apertura**, también llamados de **línea de vista** (LOS-Line Of Sight) o de rayo dirigido y en sistemas de **gran apertura** pudiendo estos últimos ser reflejados o difusos.

Los sistemas de radiofrecuencia pueden clasificarse en sistemas de **banda estrecha** (narrow band), también llamados de **frecuencia dedicada** y en sistemas basados en **espectro disperso** o extendido (spread spectrum).

2.2.1 Sistemas por Infrarrojos

La tecnología de rayos infrarrojos cuenta con muchas características sumamente atractivas para utilizarse en redes inalámbricas. En principio los rayos infrarrojos tienen una longitud de onda cercana a la de la luz y, por lo tanto, un comportamiento similar, tanto en sus ventajas y sus desventajas.

Entre las principales ventajas cabría destacar:

- La necesidad de tener una línea de vista directa entre el transmisor y el receptor provee un seguro contra receptores no deseados.

- Debido a su alta frecuencia, presenta resistencia a las interferencias electromagnéticas artificiales radiadas por otros dispositivos, pudiendo además alcanzar grandes velocidades de transmisión.

- No requieren autorización especial en ningún país, excepto por los organismos que limitan la potencia de la señal transmitida.

- Utilizan componentes económicos y de bajo consumo de energía.

Entre las principales desventajas cabría destacar:

- No pueden atravesar objetos sólidos, como las paredes, lo que supone un serio freno a su capacidad de difusión.

- Son sumamente sensibles a objetos móviles que interfieran o perturben la comunicación entre emisor y receptor.

- Las restricciones de potencia de transmisión limitan su cobertura a unas cuantas docenas de metros.

- La luz solar directa, las lámparas incandescentes y otras fuentes de luz brillante pueden interferir seriamente con la señal.

Debido a estas características, las pocas redes que emplean la luz infrarroja como un medio de transmisión están limitadas por el espacio, utilizándose casi en exclusividad en redes, en la que los distintos dispositivos que se encuentran en una determinada área, como puede ser el ambiente doméstico. Sin embargo; algunas compañías que tienen oficinas en varios edificios realizan la comunicación colocando los receptores/emisores en las ventanas de dichos edificios.

Es por eso que lo más común es encontrar esta tecnología en los sistemas incorporados a computadoras portátiles, periféricos o PDAs (Personal Digital Assistance) asociados al estándar IrDA.

2.2.2 Sistemas por Radiofrecuencia

La Comisión Federal de Comunicaciones (FFC) permitió la operación sin licencia de dispositivos que utilicen hasta 1 watt de energía en tres bandas de frecuencias distintas: 902 a 928Mhz, 2400 a 2483.5Mhz y 5725 a 5850Mhz. Estas bandas de frecuencia son las denominadas bandas ICM (Industrial, Científico y Médico o ISM en inglés) limitadas, en principio, a su implantación en dispositivos para fines industriales, científicos y médicos.

En la actualidad algunas de estas frecuencias se están abriendo y numerosos dispositivos, como teléfonos inalámbricos, puertas de garaje automáticas, sensores remotos y microondas. Por esto las redes inalámbricas que operan en estas frecuencias deben ser diseñadas para trabajar bajo interferencias considerables, para ello utilizan, generalmente, una tecnología desarrollada en los años 40 para proteger las comunicaciones militares: *la técnica de espectro disperso.*

La idea es tomar una señal de banda convencional y distribuir su energía en un dominio más amplio de frecuencias. Así la densidad promedio de energía es menor en el espectro equivalente de la señal original, en aplicaciones militares el objetivo es reducir la densidad de energía por debajo del nivel de ruido ambiental de tal forma que la señal no fuese detectable; en cambio, esta técnica aplicada a las redes inalámbricas permite que la señal sea transmitida y recibida con un mínimo de interferencia.

Básicamente existen dos técnicas de modulación cuando se hace uso de la tecnología del espectro disperso:

- **Salto de frecuencia** (FHSS, Frecuency-Hopping Spread Spectrum). Los dispositivos saltan de una frecuencia a otra de manera síncrona según un patrón predeterminado. *Sólo aquellos dispositivos sincronizados pueden acceder a la información.*

- **Secuencia directa** (DSSS, Direct-Sequence Spread Spectrum). La información a transmitir se mezcla con un patrón pseudoaleatoreo de bits para extender los datos antes de que se transmitan. Cada bit transmitido se modula por medio de la secuencia de bits del patrón de referencia, extendiendo su ancho de banda, solo el receptor que tenga el mismo código de extensión será capaz de regenerar la información original, mientras que para cualquier otro receptor es ruido de baja potencia que resulta ignorado, esta técnica permite también corregir algunos de los errores que se puedan producir en la transmisión y requiere un procesador digital de señales (DSP) para correlacionar la señal de entrada.

La elección de la técnica dependerá de diversos factores relacionados con la aplicación de los usuarios y el entorno en el que opere el sistema.

Otro factor que se debe tener en cuenta, es la normativa acerca de la potencia de transmisión y las frecuencias utilizables, datos que varían entre EU, Europa y Japón.

2.3 El estándar IrDA

La asociación de datos por infrarrojos IrDA es una organización patrocinada por la industria y establecida en 1993 para crear estándares internacionales para el equipo y programas usados en los enlaces de comunicación por infrarrojos. En el apéndice B se muestra el logotipo de IrDA.

En esta forma de transmisión de radio, un haz enfocado de luz en el espectro de frecuencia infrarrojo, medido en terahertz o billones de hertz se modula con información y se envía de un transmisor a un receptor a una distancia relativamente corta.

La comunicación de datos por infrarrojos juega un papel importante en las comunicaciones de datos inalámbricas, debido a la popularidad de computadoras portátiles, PDAs, cámaras digitales, teléfonos móviles y otros dispositivos.
Entre los usos existentes o posibilidades razonables están:

- Enviar un documento de una computadora, cámara digital o PDA a una impresora.

- Intercambiar archivos entre computadoras.

- Coordinar agendas y libretas telefónicas entre computadoras de escritorio y portátiles.

- Enviar faxes desde una computadora portátil a un fax distante.

- Cámaras digitales que pueden enviar las imágenes a computadoras, teléfonos inalámbricos, impresoras y PDAs.

La comunicación infrarroja involucra un transceptor en los dos dispositivos que se comunican. Hay microchips especiales que proporcionan esta capacidad.

Adicionalmente, uno o ambos dispositivos pueden requerir software especial para que la comunicación pueda sincronizarse. En el estándar IrDA-1.1, el máximo tamaño de datos que se pueden transmitir es de 2048 bytes y la tasa máxima de transmisión es de 4 Mbps.

2.4 El estándar IEEE 802.11

El estándar IEEE 802.11 especifica los parámetros de las capas Físicas y de Control de Acceso al Medio (MAC) para redes inalámbricas.

La capa Física puede utilizar tanto enlaces por radiofrecuencia (FHSS y DSSS) en la banda de 2.4 GHz, modulando la señal por *posición de pulso*.

La capa MAC incluye el mecanismo CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance), donde un nodo se asegura de que el canal está libre antes de transmitir. El mecanismo de detección de colisiones usado en CSMA/CD no puede utilizarse en este caso debido a que un nodo no puede transmitir y escuchar el canal para detectar si otra estación lo hace al mismo tiempo. El mecanismo CSMA/CA no elimina las colisiones completamente, sólo minimiza su probabilidad.

Cuando un paquete está listo para transmitir y el canal está vacío, el nodo emite un paquete RTS (Ready to Send) y espera que el receptor le envíe un paquete CTS (Clear to Send). Tras este intercambio de paquetes el emisor envía su trama y si no hubo errores (utilizando un mecanismo basado en el uso del Código de Redundancia Cíclica CRC), el receptor envía un paquete ACK (Acuse de Recibo).

Los sistemas 802.11 son llamados generalmente "Wi-Fi". La alianza Wi-Fi es responsable de otorgar los logotipos de certificado Wi-Fi, que asegura la compatibilidad e interoperabilidad de los fabricantes de dispositivos con tecnología 802.11. En el apéndice B se muestra el logotipo de la alianza Wi-Fi.

El estándar original 802.11 se estableció en Junio de 1997, definiendo los sistemas a 2.4 GHz con una velocidad de transmisión máxima de 2 Mbps. En la actualidad existen dos categorías básicas de estándares para Redes de Área Local (WLAN) IEEE 802.11. La primera especifica los protocolos fundamentales para los sistemas Wi-Fi. Estos son conocidos como los estándares 802.11a, 802.11b, y 802.11g. Los segundos, son extensiones de especificaciones que proveen funciones adicionales a estos estándares y son: 802.11d, e, f, h, i y j. En la tabla 2-1 se muestran los tres estándares fundamentales, así como sus características principales.

Estándar	Banda de radio	Cobertura	Velocidad de transmisión	modulación
802.11b	2.4 GHz	100m/328ft	11 Mbps	DSSS
802.11a	5 GHz	50m/164ft	54 Mbps	OFDM
802.11g	2.4 GHz	100m/328ft	54 Mbps	OFDM

Tabla 2-1. Estándares fundamentales 802.11

2.4.1 El estándar 802.11b

El estándar 802.11b opera en la banda de radio de 2.4 GHz, tiene una velocidad máxima de conexión de 11 Mbps y cobertura de hasta 100 metros. Este estándar es práctico cuando no se tienen aplicaciones que requieran un amplio ancho de banda y se necesite tener una cobertura amplia; además una WLAN 802.11b cuesta en la actualidad como un cuarto del precio de una WLAN 802.11a, cubriendo la misma área y la misma velocidad de transmisión, y debido a que ocupa la banda de radio de 2.4 GHz al igual que otras tecnologías (como por ejemplo **Bluetooth** y teléfonos inalámbricos), puede implicar interferencia entre ellas.

2.4.2 El estándar 802.11a

El estándar 802.11a opera en la banda de radio de 5 GHz, tiene una velocidad máxima de conexión de 54 Mbps y cobertura de hasta 50 metros. Además de su mayor velocidad de conexión, especifica doce canales de radio mientras que la norma 802.11b específica tres canales. Proporciona así mayor protección contra posibles interferencias por múltiples puntos de acceso, este estándar es práctico cuando se requiere un amplio ancho de banda como por ejemplo aplicaciones de voz o video. La mayor desventaja es que no es compatible con 802.11b y su costo es muy elevado.

2.4.3 El estándar 802.11g

El estándar 802.11g opera en la banda de radio de 2.4 GHz, tiene una velocidad máxima de conexión de 54 Mbps y cobertura de hasta 100 metros. Dado que usa la misma frecuencia de radio que la norma 802.11b, ofrece compatibilidad hacia atrás con puntos de acceso 802.11b. Este estándar es práctico cuando se requiere un amplio ancho de banda y se necesita una cobertura amplia, sus principales desventajas son, que al compartir la misma red con 802.11b la velocidad de transmisión será la misma que la de 802.11b; además presenta los mismos problemas de interferencia por su frecuencia a 2.4 GHz.

2.4.4 Otros estándares 802.11

Las siguiente extensiones de 802.11 (excepto .11h y .11j) aplican a todas las variantes de Wi-fi:

- **802.11d** Estándar que permite el uso de la comunicación mediante el protocolo 802.11 en países que tienen restricciones sobre el uso de las frecuencias que éste es capaz de utilizar. De esta forma se puede usar en cualquier parte del mundo.

- **802.11e** Estándar que provee prioridades de acceso para ofrecer Calidad de Servicio (QoS).

- **802.11f** Es el llamado Inter Access Point Protocol (IAPP). Estándar que define una práctica recomendada de uso sobre el intercambio de información entre el Punto de Acceso y el Transmisor en el momento del registro a la red y la información que intercambian los Puntos de Acceso para permitir la interportabilidad. La adopción de esta práctica permitirá el Roamming entre diferentes redes.

- **802.11h** Estándar que sobrepasa al 802.11a al permitir la asignación dinámica de canales para permitir la coexistencia de éste con el HyperLAN. Además define el TPC (Transmit Power Control) para ajustar la potencia de transmisión.

- **802.11i** Estándar que define la encriptación y la autentificación para complementar y mejorar el WEP (Wired Equivalent Privacy). Es un estándar que mejorará la seguridad de las comunicaciones mediante el uso del Temporal Key Integrity Protocol (TKIP).

- **802.11j** Agrega un canal de 4.9 GHz a 5 GHz para 802.11a en Japón.

2.4.5 Seguridad

Una red con cable está dotada de una seguridad inherente en cuanto a que un posible ladrón de datos debe obtener acceso a la red a través de una conexión por cable, lo que normalmente significa el acceso físico a la red de cables, cuando la red ya no se sustenta con cables, la libertad que obtienen los usuarios también se hace extensiva al posible ladrón de datos.

Desde sus comienzos, 802.11 ha proporcionado algunos mecanismos de seguridad básicos para impedir que esta libertad mejorada sea una posible amenaza. Por ejemplo, los Puntos de Acceso (o conjuntos de puntos de acceso) 802.11 se pueden configurar con un identificador del conjunto de servicios (SSID). La tarjeta NIC también debe conocer este SSID para asociarlo al Punto de Acceso y así proceder a la transmisión y recepción de datos en la red.

Las especificaciones 802.11 proporcionan seguridad adicional mediante el algoritmo WEP que proporciona a 802.11 servicios de autenticación y cifrado. El algoritmo WEP define el uso de una clave secreta de 40 bits para la autenticación y el cifrado, y muchas implementaciones de IEEE 802.11.

También permiten claves secretas de 104 bits. Este algoritmo proporciona la mayor parte de la protección contra la escucha y atributos de seguridad física que son comparables a una red con cable.

Una limitación importante de este mecanismo de seguridad es que el estándar no define un protocolo de administración de claves para la distribución de las mismas. Esto supone que las claves secretas compartidas se entregan a la estación inalámbrica IEEE 802.11 a través de un canal seguro independiente del IEEE 802.11, para proporcionar un mecanismo mejor para el control de acceso y la seguridad, se incluye un protocolo de administración de claves en la especificación. 802.1X es un protocolo del estándar para el control de acceso a redes basado en puerto que se utiliza para proporcionar acceso a red autenticado para las redes Ethernet. Este control de acceso a red basado en puerto utiliza las características físicas de la infraestructura LAN conmutada para autenticar los dispositivos conectados a un puerto LAN. Si el proceso de autenticación no se realiza correctamente, se puede impedir el acceso al puerto.

Concretamente, en el caso de las conexiones inalámbricas, el punto de acceso actúa como autentificador para el acceso a la red y utiliza un Servidor del Servicio de Usuario de Acceso Telefónico de Autenticación Remota (RADIUS) para autenticar las credenciales del cliente.

Un servidor RADIUS puede realizar consultas en una base de datos de autenticación local si ello es adecuado para el escenario. O bien, la solicitud puede transmitirse a otro servidor para su validación, cuando RADIUS decide que se puede autorizar el equipo en esta red, vuelve a enviar el mensaje al punto de acceso y éste permite que el tráfico de datos fluya hacia la misma.

2.5 Bluetooth

Bluetooth es una tecnología utilizada para la conectividad inalámbrica de corto alcance entre dispositivos como PDAs, teléfonos celulares, teclados, faxes, computadoras de escritorio y portátiles, módems, proyectores, impresoras, etc. Está basada en un enlace de radio de bajo costo y corto alcance, implementado en un circuito integrado de 9 x 9 mm, proporcionando conexiones instantáneas (ad hoc) para entornos de comunicaciones tanto móviles como estáticas, el principal mercado es, la transferencia de datos y voz entre dispositivos y computadoras personales. El enfoque de **Bluetooth** es similar a la tecnología IrDA, sin embargo **Bluetooth**, es una tecnología de radiofrecuencia (RF) que utiliza la banda de espectro disperso de 2.4 GHz al igual que IEEE 802.11b y 802.11g. Aunque 802.11b/g ofrecen más velocidad de transmisión, necesitan más potencia y ofrecen menos opciones de conectividad que **Bluetooth** para el caso de aplicaciones de voz.

Bluetooth intenta proveer significantes ventajas sobre otras tecnologías inalámbricas similares tales como IrDA y 802.11, claros competidores en conexiones de PC a periféricos. IrDA es una tecnología muy popular para conectar periféricos, pero está limitada a conexiones de cortas distancias en rangos de un metro por la línea de vista requerida para la comunicación, debido a que **Bluetooth** funciona con RF no está sujeto a tales limitaciones. La distancia de conexión en **Bluetooth** puede ser de hasta 10 metros o más dependiendo del incremento de la potencia del transmisor, pero los dispositivos no necesitan estar en línea de vista ya que; las señales de RF pueden atravesar paredes y otros objetos no metálicos sin ningún problema.

La versión 1.0 de la especificación **Bluetooth** fue liberada en 1999, pero el desarrollo de esta tecnología empezó realmente 5 años atrás, en 1994, cuando la compañía Ericsson empezó a estudiar alternativas para comunicar los teléfonos celulares con otros dispositivos. El estudio demostró que el uso de enlaces de radio seria el más adecuado, ya que no es directivo y no necesita línea de vista, eran tan obvias estas ventajas con respecto a los enlaces vía infrarrojo que son utilizados para conectar dispositivos y teléfonos celulares. Existían muchos requerimientos para el estudio, los cuales incluían la manipulación tanto de voz como de datos, de tal manera se podrían conectar teléfonos a dispositivos de cómputo. Así es como nace la especificación de la tecnología inalámbrica conocida como **Bluetooth**. El origen del nombre de esta tecnología proviene de un Vikingo de origen Danés, Harald Blatand (**Bluetooth**) quien en el siglo décimo unificó Dinamarca y Noruega. El nombre fue adoptado por Ericsson, quien espera que **Bluetooth** unifique las telecomunicaciones y la industria del cómputo.

Antes de que los productos **Bluetooth** estuvieran en el mercado, varias compañías formaron el **Bluetooth** SIG (Special Interest Group), el cual proporcionó gran reconocimiento a nivel mundial de la marca **Bluetooth**. Cerca de 2500 miembros del SIG trabajan para promover, formar y definir la especificación y la posición de **Bluetooth** en el mercado.

El SIG se formó en 1998, con los principales promotores: Ericsson, Intel, IBM, Toshiba y Nokia. En Julio de 1999 los promotores publicaron la versión 1.0 de la especificación. Durante la última semana del mes de Marzo del 2002 la IEEE aprobó finalmente el estándar IEEE 802.15.1 compatible totalmente con la tecnología **Bluetooth** v1.1. En este estándar se definen las especificaciones de la capa Física y MAC para las redes WPANs, el nuevo estándar permitirá una mayor validez y soporte en el mercado de las especificaciones de **Bluetooth**, además es un recurso adicional para aquellos que implementen dispositivos basados en esta tecnología.

Las especificaciones **Bluetooth**, son especificaciones de comunicaciones inalámbricas y no tiene costo alguno bajarlas de Internet y usarlas. Estas especificaciones definen el funcionamiento mínimo de los dispositivos para que puedan operar aunque sean de distintos fabricantes. Esto significa:

- Definición de protocolos para la interoperabilidad
- El controlador de Interfaz del Host (Host Controller Interface)
- Portador de servicios para protocolos de las capas más altas
- Perfiles
- Calificación
- Prueba de productos
- El libro de la marca

La especificación de **Bluetooth** cubre desde el transceptor de radio hasta varias interfaces de protocolos basados tanto en hardware como en software, y define el concepto de **Red de Área Personal** (PAN, Personal Area Network). También es importante mencionar que no proporciona instrucciones de implementación ni aplicaciones de interfaces, o alguna especificación de software y hardware; aunque se puede encontrar una pequeña guía en la especificación de *perfiles*.

Bluetooth puede ser usado para aplicaciones en redes residenciales o en pequeñas oficinas, ambientes que son conocidos como WPANs (Wireless Personal Area Network). En el apéndice B se muestra el logotipo de **Bluetooth** el cual aparece en los productos certificados.

Los dispositivos **Bluetooth** son generalmente organizados en grupos de 2 a 8 llamados *piconets*: dos o más unidades **Bluetooth** que comparten un mismo canal forman una *piconet*. Todos los dispositivos tienen la misma implementación, para regular el tráfico en el canal, una de las unidades participantes se convertirá en *maestra*; por definición, la unidad que establece la piconet asume este papel y todos los demás serán *esclavos*. Los participantes podrían intercambiar los papeles si una unidad esclava quisiera asumir el papel de maestra, sin embargo; sólo puede haber un *maestro* en una *piconet* al mismo tiempo.

Las unidades que se encuentran en el mismo radio de cobertura pueden establecer potencialmente comunicaciones entre ellas. Sólo aquellas unidades que realmente quieran intercambiar información comparten un mismo canal creando la *piconet*. Este hecho permite que se creen varias *piconets* en áreas de cobertura superpuestas.

A un grupo de *piconets* que comparten dispositivos se le llama *scatternet*. En la figura 2-1 se muestra un ejemplo donde encontramos tres *Piconets* que a su vez forman una *scatternet*.

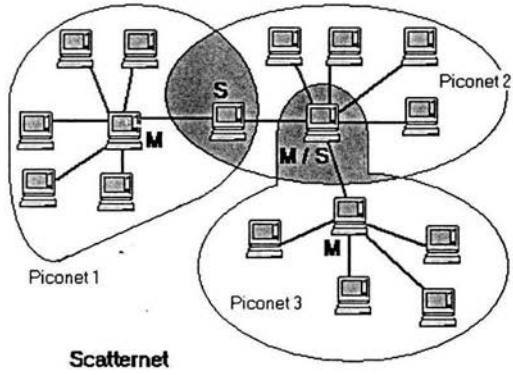

Figura 2-1. Tres Piconets formando una Scatternet

El principal objetivo de esta tecnología, es la posibilidad de reemplazar los cables de los diferentes dispositivos y periféricos; por ejemplo, la tecnología de radio **Bluetooth** implementada en el teléfono celular y en una computadora portátil reemplazaría el molesto cable utilizado hoy en día para conectar ambos aparatos.

La figura 2-2 muestra un modelo de usuario con conectividad inalámbrica local y sus posibles aplicaciones.

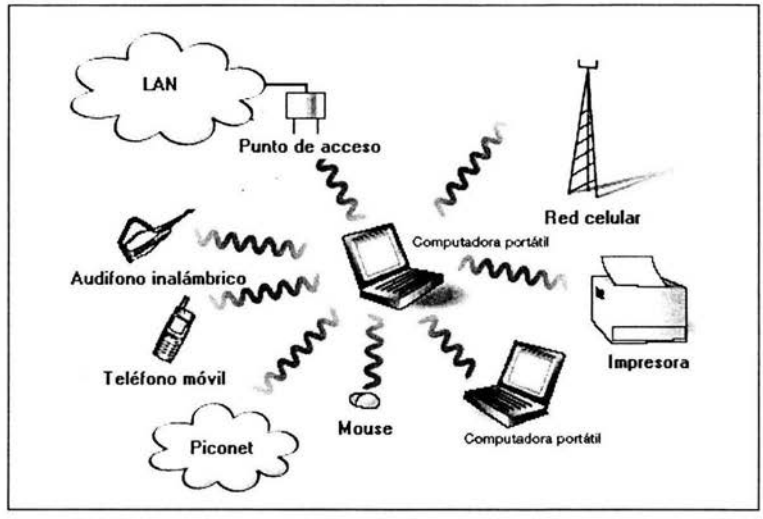

Figura 2-2. Modelo de posibles aplicaciones de Bluetooth.

El transmisor permite enviar voz y datos a una velocidad máxima de, 700 Kb/seg. y consume un 97% menos de potencia que un teléfono móvil; además, es inteligente, cuando el tráfico de datos disminuye, el transmisor adopta el modo bajo de consumo de energía.

2.6 Comparación de las tecnologías inalámbricas

Como se ha visto a lo largo de este capítulo, existen diferentes tipos de tecnologías inalámbricas, así que es importante saber como diferenciar cada una, y saber cuando ocuparlas según el escenario de aplicación.

Para entender el lugar que ocupa cada tecnología es importante mencionar los tres principales escenarios en la comunicación inalámbrica:

- Redes inalámbricas de área personal (WPAN, Wireless Personal Area Network). Se enfocan principalmente en la conectividad entre los diferentes dispositivos y periféricos que una persona puede tener a su disposición, un ejemplo podría ser un usuario que tenga conectados inalámbricamente su computadora portátil, su PDA, su teléfono móvil y su impresora con la cual podrá sincronizar agendas, compartir e imprimir archivos o conectarse a Internet.

- Redes Inalámbricas de Área Local (WLAN, Wireless Local Area Network). Al igual que las LANs, están enfocadas a la conexión de grupos de trabajo para compartir archivos y servicios.

- Redes Inalámbricas de Área Global WWAN. Están enfocadas a mantener una conexión mientras se viaja fuera de los límites de trabajo; actualmente se puede tener este tipo de enlaces por medio de tecnologías celulares o PC Cards con módems celulares.

La tabla 2-2 muestra los escenarios usuales de conexión inalámbrica por tecnología.

Estándar Wireless	Categoría de aplicación	Escenarios usuales
Bluetooth	Wireless Personal Area Network (WPAN)	• Conectar una computadora portátil a otra computadora portátil que cuente con **Bluetooth** para transferir archivos. • Elaborar un documento en grupo por medio de Microsoft NetMeeting, donde los participantes cuenten con conectividad inalámbrica **Bluetooth**. • Usando audífonos con **Bluetooth** escuchar música desde una computadora portátil mientras está guardada en su maletín. • Sincronizar la agenda del teléfono inalámbrico con una PDA o computadora portátil. • Conectar una impresora temporal a una computadora portátil sin necesidad de cargar los cables necesarios cuando se viaja.
802.11	Wireless Local Area Network (WLAN)	• Estar siempre conectado a la red corporativa mientras se mueve por todo el campus o en salas de conferencias, cafeterías y vestíbulos. • Accesar a Internet por medio de Hot Spots (puntos calientes) en aeropuertos, hoteles, cafeterías, etc. • Montar una red provisional para algún evento, sin tener que planear toda una infraestructura compleja.
Tecnologías celulares	Wireless Wide Area Network (WWAN)	• Accesar a Internet mientras se viaja de la oficina a la casa.

Tabla 2-2. Escenarios usuales de conexión inalámbrica por tecnología.

La tecnología **Bluetooth** actualmente se está convirtiendo en la solución más eficiente para las conexiones inalámbricas de tipo WPAN, en este campo, la única competencia en cuanto a tecnologías inalámbricas es IrDA, se espera que en cuanto los costos de **Bluetooth** bajen, disminuya el uso de la tecnología IrDA. En cuanto a las WLAN, 802.11 tiene un mayor uso que **Bluetooth** dado su alcance, costo y algunas otras características como el número de dispositivos que pueden coexistir en una misma área.

La tabla 2-3 muestra las principales ventajas y desventajas de las tecnologías inalámbricas.

Tecnología	Ventajas	Desventajas
Bluetooth	No se necesita línea de visión directa como IrDA. El bajo consumo de energía lo hace muy práctico en dispositivos que requieren uso de baterías. La frecuencia de radio de 2.4 GHz asegura la operatibilidad en todo el mundo. El programa de calificación para el uso del logotipo de **Bluetooth** asegura que ha sido sometido a diversas pruebas para satisfacer el estándar. Tiene aplicaciones no sólo en la industria de la computación, sino también en la industria de las telecomunicaciones y transporte.	Por ser una tecnología reciente, en la actualidad no se encuentra fácilmente información de ella. El hecho de ocupar la frecuencia de radio de 2.4 GHz, al igual que 802.11b y 802.11g y algunas otras tecnologías implica que puede haber interferencia, o que ambas tecnologías no podrán estar disponibles al mismo tiempo en el mismo lugar.
802.11	Cuenta con un mayor ancho de banda y cobertura que las demás tecnologías.	Requiere una planeación de la infraestructura y de hacer pruebas por posibles interferencias.
IrDA	Actualmente se encuentran en el mercado una gran diversidad de dispositivos que utilizan esta tecnología. Tiene una mayor velocidad de transmisión que la primera versión de **Bluetooth** (4 Mbps de IrDA contra 1 Mbps de **Bluetooth**). Actualmente los costos de IrDA son menores que **Bluetooth**. Debido a que utiliza tecnología de luz infrarroja, no tiene problemas de interferencia con aquellos dispositivos que utilizan radio frecuencia, como **Bluetooth** y 802.11.	El rango de cobertura es muy pequeño a comparación de las otras tecnologías. Necesita tener línea de vista directa entre los dispositivos que se van a comunicar, lo cual provoca que sea poco flexible y más difícil de usar.

Tabla 2-3. Principales ventajas y desventajas de las tecnologías inalámbricas.

Por último, la tabla 2-4 muestra la comparación entre las diferentes tecnologías inalámbricas.

	802.11	Bluetooth	IrDA
Mercado	WLAN	WPAN	WPAN
Tecnología	Radio Frecuencia 2.4 y 5 GHz DSSS y OFDM	Radio Frecuencia 2.4 GHz FHSS	Óptico
Potencia	Moderada 20 dBm	Baja 0/20 dBm	
Velocidad de transmisión	Alta 11/54 Mbps	Moderada 1 Mbps	Moderada 115 Kbps/4 Mbps
Distancia	Hasta 100 m	Hasta 100 m	Hasta 5 m
Topología	256 dispositivos punto-multipunto	8 dispositivos punto-multipunto	10 dispositivos punto-multipunto
Seguridad	WEP	Llave pública/privada y encriptación	En la capa de aplicación

Tabla 2-4. Comparación entre las diferentes tecnologías inalámbricas.

2.6.1 Bluetooth contra IrDA

Debido a que tanto **Bluetooth** como IrDA, están enfocadas a las WPANs, vale la pena analizar sus diferencias en el campo de las aplicaciones.

Tanto IrDA y **Bluetooth** consideran el intercambio de datos como una función imprescindible, el intercambio de datos puede ser tan simple como intercambiar una tarjeta de negocios desde un teléfono móvil a una PDA, o tan sofisticado como la sincronización de la información personal entre una PDA y una computadora.

Un escenario de intercambio común de datos es en el que se producirá el intercambio datos en un espacio donde se encuentran otros equipos: dos personas se encuentran para intercambiarse tarjetas de negocio cara a cara en una amplia sala de conferencias. Muchos otros llevan aparatos sin cable que también se encuentran en la habitación, posiblemente intentando hacer lo mismo, ésta es la situación en la que el IrDA puede tomar ventaja. El corto alcance y el estrecho ángulo del IrDA proporcionan una forma simple de seguridad ya que se dirigen ambos dispositivos y se establece la comunicación. Mientras que para **Bluetooth** con sus características omnidireccionales, tiene problemas para descubrir el dispositivo destino, no permite al usuario señalar simplemente al destino; un aparato **Bluetooth** puede realizar una operación de descubrimiento en el que encontrará a muchos de los otros aparatos en la habitación, la cercana proximidad del destino no ayudará. El usuario estará obligado a elegir entre un listado de aparatos descubiertos. Los dos usuarios intentando realizar un intercambio de tarjetas, usando **Bluetooth** necesitarían tomar además, medidas de seguridad.

A pesar de esto, **Bluetooth** tiene la habilidad de penetrar objetos sólidos y su capacidad de movilidad máxima en la piconet, permite aplicaciones de intercambio de datos que son imposibles o difíciles con la IrDA. Por ejemplo con **Bluetooth** una persona podría sincronizar el teléfono con una PC sin sacarlo del bolsillo.

Esto no es posible con el IrDA. El uso del **Bluetooth** para la sincronización no requiere que el teléfono permanezca en una localización fija. Si la persona lleva el teléfono en el bolsillo, la sincronización se puede producir mientras se mueve. Con el IrDA, el teléfono se puede colocar en la localización adecuada y permanecer estático mientras la sincronización se ejecuta.

Otra importante característica del **Bluetooth** y el IrDA es la habilidad de conexión a una LAN. Como no hay requerimientos de señal de línea para los aparatos de **Bluetooth**, sirve muy bien este tipo de aplicación. Los usuarios del **Bluetooth** tienen un alto nivel de flexibilidad cuando usan un punto de acceso LAN.

El IrDA especifica el protocolo de IrLAN mediante la conexión de un aparato IrDA a una red. Los requerimientos del IrDA para la señal de línea y distancias máximas de 1 metro, deben tenerse en cuenta cuando se instalan los aparatos de acceso del IrLAN, además una vez que el IrDA está conectado al LAN debe permanecer relativamente estático.

En cuanto a las aplicaciones de voz, una primitiva característica de las especificaciones del **Bluetooth** es la sincronización de los canales de voz. **Bluetooth** tiene la habilidad de reservar un ancho de banda para llevar los datos digitales de voz. **Bluetooth** puede soportar hasta tres conversaciones dúplex de voz con una piconet.

Un componente para las especificaciones de los infrarrojos de las comunicaciones móviles IrMC incluye RTCON lo que es una especificación para la transmisión de datos de voz dúplex sobre una unión de IrDA. RTCON consume la completa amplitud de banda de 115.2 Kbps, así que; la unión de IrDA multiplexando otros datos no está permitida.

En cuanto los aspectos de seguridad, la naturaleza direccional del IR tiene una bajo nivel de seguridad porque requiere una línea directa entre transmisor y receptor, sin embargo; es posible introducirse en una conversación detectando la luz reflectada e introduciéndose en el ruido ambiental de los alrededores. El IrDA no tiene capacidad de seguridad a un nivel de conexión como el **Bluetooth**, en su lugar confía en un nivel de protocolo mayor y mayores aplicaciones para proporcionar autentificación y/o encriptación. Como el **Bluetooth** es omnidireccional, puede ser visto por un aparato detector desde cualquier dirección incluyendo espacios escondidos, por lo que **Bluetooth** proporciona autentificación y encriptación en su protocolo de banda básico.

2.7 Productos Bluetooth

En la actualidad podemos encontrar una gran diversidad de productos **Bluetooth** que van desde material de implementación, como son los módulos y antenas; material de desarrollo, software; hasta productos implementados con esta tecnología, como pueden ser teléfonos celulares, computadoras portátiles, PDAs, audífonos, etc.

La tabla 2-5 muestra de manera muy resumida algunos productos que podemos encontrar en el mercado actual con tecnología **Bluetooth**.

Es importante aclarar que la mayoría de estos productos no se encuentran todavía de venta en México, teniendo su mayor mercado en Europa y Asia.

Sector o aplicación	Producto	Ejemplos
Telefonía	• Teléfonos móviles	• SmartPhone Sony Ericsson P900 • SmartPhone Nokia 7600
	• Auriculares	• Auricular Jabra BT250 FreeSpeak • Auricular **Bluetooth** Logitech
	• Sistemas de manos libres	• Manos libres **Bluetooth** de ECHO • Kit manos libres SuperTooth
Sistemas Integrados	• Computadoras portátiles	• Sony Vaio TR1MP • Apple iBook G4 12
	• PDAs	• PocketPC hp Ipaq 4150 • Palm Tungsten T3
	• Impresoras	• Impresora portátil hp deskjet 450Wbt • Impresora etiquetas portátil Zebra QL320
	• Periféricos	• Mouse y teclado Logitech Cordless Desktop MX BT • Mouse Microsoft **Bluetooth**
Adaptadores	• Para PC	• Adaptador USB TECOM BT-3035 • PC Card **Bluetooth** 3com
	• Para PDAs	• Tarjeta CF **Bluetooth** Conceptronic • Tarjeta Palm SD **Bluetooth**
	• Para impresoras	• Adaptador pueto paralelo/USB hp BT1300 • Anycom **Bluetooth** CF-2001 Printer Card
Electrónica	• Mother Board	• Mother Board PC Epox EP-8K5A2+B1
	• Módulos	• Ericsson ROK 101 007 • Taiyo Yuden EYMF2C
	• Antenas	• Antena omni-direccional Blue2space • Antena gigaAnt Limata
Desarrollo	• Kits de desarrollo	• Ericsson **Bluetooth** Application Tool Kit • Digianswer **Bluetooth** DemoCard for Windows CE
	• Software	• Bluez • Affix
Aplicaciones	• Acceso a Internet	• Módem inalámbrico **Bluetooth** Allnet ALL1560 • Módem/Router ADSL + Punto de Acceso **Bluetooth**
	• Acceso LAN	• Punto de Acceso LAN **Bluetooth** TECOM BT-3021 • Punto de acceso Siemens Blue2net

Tabla 2-5. Productos Bluetooth en el mercado.

2.8 Conclusiones del capítulo

Como se explicó a lo largo del capítulo, las tecnologías inalámbricas proveen gran libertad de movimiento y flexibilidad al no depender de cables para la conexión de dispositivos.

También se explicó que en la actualidad las tecnologías inalámbricas que tienen mayor auge en la mayoría del mundo son: IEEE 802.11, IrDA y **Bluetooth**. La primera se enfoca a las Redes Inalámbricas de Área Local WLAN debido a su velocidad de transmisión y la cobertura que puede tener; en cuanto a **Bluetooth** e IrDA, se enfocan principalmente a las Redes Inalámbricas de Área Personal y su principal diferencia es que **Bluetooth** utiliza la banda de radio ISM con radiofrecuencia lo que le permite atravesar paredes y no debe de tener línea de vista directa entre dispositivos que se están comunicando; mientras IrDA utiliza rayos infrarrojos y necesita una línea de vista directa entre dispositivos para que se pueda establecer la comunicación.

Por último **Bluetooth** es una tecnología reciente, pero cuenta con las características necesarias para tener una gran gama de aplicaciones en la industria del cómputo y las telecomunicaciones; sólo se tiene que dar a conocer más en el orden que los precios de integración bajen, ya que por ser nueva, todavía es cara su implementación a comparación de las demás tecnologías.

CAPÍTULO 3.
ESPECIFICACIONES DE LA TECNOLOGÍA BLUETOOTH

En este capítulo se estudiará la especificación **Bluetooth**, la cual está disponible en la página oficial de **Bluetooth**: **http://www.bluetooth.org**. La especificación se divide en dos partes:

- La *Especificación* ***Bluetooth***; la cual describe cómo trabaja la tecnología (por ejemplo la arquitectura de los protocolos).

- Los *Perfiles* ***Bluetooth***; los cuales describen cómo es usada la tecnología (por ejemplo, como las diferentes partes de la especificación pueden ser usadas para una función determinada en un dispositivo).

A lo largo del capítulo se estudiaran ambas partes, primero la *Especificación*, y después los *Perfiles*.

Aunque ya se encuentra disponible la versión 1.2, se analizará la versión 1.1, ya que es la que actualmente se utiliza para el diseño de dispositivos **Bluetooth**.

3.1 Pila de protocolos Bluetooth

Al igual que OSI, la Especificación **Bluetooth** hace uso de un protocolo estructurado en capas, aunque la arquitectura **Bluetooth** tiene su propia pila de protocolos de cuatro capas, hace reutilización de los protocolos existentes en las capas superiores; su objetivo fundamental es permitir que las aplicaciones descritas por la especificación puedan interoperar entre sí. La interoperabilidad se logra cuando las aplicaciones en los dispositivos remotos corren sobre pilas de protocolos idénticas, sin tener en cuenta la aplicación específica, la pila de protocolos utiliza siempre una capa Física y una de Enlace de datos común como se puede ver en la figura 3-1.

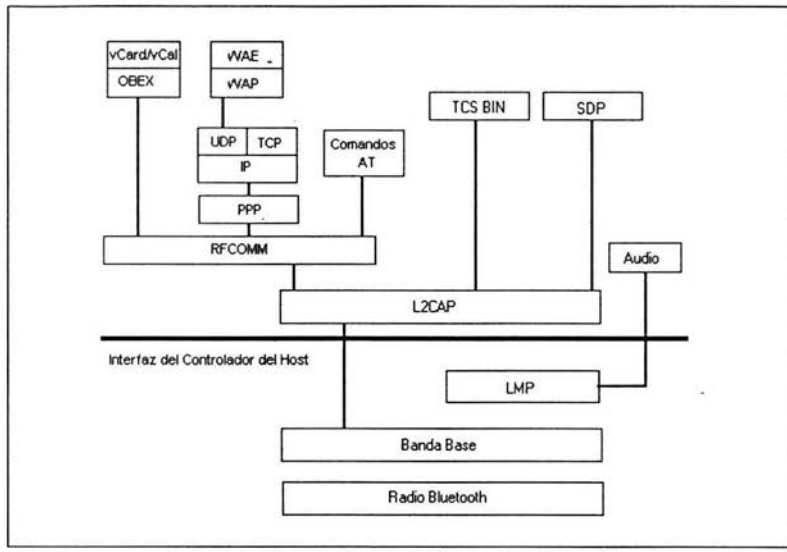

Figura 3-1. Pila de Protocolos Bluetooth

La pila completa incluye protocolos que son específicos de la tecnología **Bluetooth**, como el LMP (Protocolo del Administrador de Enlace) y el L2CAP (Logical Link Control and Adaptation Protocol), y aquellos que pueden usarse con muchas otras plataformas, como el Protocolo de Intercambio de Objetos (OBEX, Object Exchange Protocol), el Protocolo de Datagrama de Usuario (UDP, User Datagram Protocol), y el Protocolo de Aplicación Inalámbrica (WAP, Wireless Application Protocol).

3.2 Radio Bluetooth

La capa de Radio **Bluetooth**, es la capa más baja definida en la *Especificación* de **Bluetooth**, esta define los requerimientos del dispositivo transreceptor **Bluetooth**.

3.2.1 Bandas de frecuencia y convenios de canales

El sistema **Bluetooth** trabaja en la banda de radio para uso Industrial, Científica y Médica ISM (Industrial Scientific and Medical). En la mayor parte del mundo el rango de frecuencia es 2400 – 2483.5 Mhz. Algunos países tienen ciertas restricciones en este rango, el radio **Bluetooth** utiliza la técnica de espectro disperso y modulación de salto de frecuencia, en 79 saltos de 1 Mhz, empezando en 2.402 GHz y finalizando en 2.480 GHz. y en los países donde existen restricciones, este rango de frecuencias está reducido y se utiliza un sistema de 23 saltos.

3.2.2 Características del transmisor

Cada dispositivo, se puede clasificar en 3 clases de potencia:

- Clase 1: está diseñado para dispositivos de alto alcance (~100m), con una máximo de potencia de salida de 20dBm / 100mW,

- Clase 2: está diseñado para dispositivos de alcance medio (~10m), con un máximo de potencia de salida de 4dBm / 2.5mW,

- Clase 3: está diseñado para dispositivos de corto alcance (~10cm), con un máximo de potencia de salida de 0dBm / 1mW.

La interfaz de radio **Bluetooth** está basada en una potencia de salida de **0dBm** de la antena, cada dispositivo puede variar la potencia de transmisión; los equipos con capacidad de control de potencia pueden optimizar la potencia de salida en el enlace con comandos LMP. Esto se logra midiendo el Indicador de Intensidad de Señal Recibida (RSSI) y reportando si es necesario incrementar o aumentar la potencia.

Es importante mencionar que la Clase 1, no se debe de usar para transmitir paquetes de un dispositivo a otro que no soporte los mensajes de control de potencia, para esto es mejor pensar en utilizar un transmisor de Clase 2 o 3.

3.2.3 Características de Modulación

Se emplea como método de modulación un esquema de Codificación de Desplazamiento de Frecuencia Gaussiano (GFSK, Gaussian Frequency Shift Keying) con un factor **BT=0.5**. El factor BT (Bandwidth time product) es un parámetro característico del esquema GFSK, y se determina con base al producto entre el ancho de banda del filtro gaussiano y la duración del bit. Gracias a este parámetro se limita el ancho de banda que ocupa el espectro de la señal GFSK una vez que la misma es transmitida, cuestión que permite evitar la interferencia con otras portadoras que se ubiquen en un canal adyacente. El índice de modulación de frecuencia "**m**" (**$m = 2\Delta f/BW$**) debe estar comprendido entre **0.28** y **0.35**.

El uno binario es representado por una desviación de frecuencia positiva, mientras que un cero binario es representado por una desviación de frecuencia negativa como se muestra en la figura 3-2.

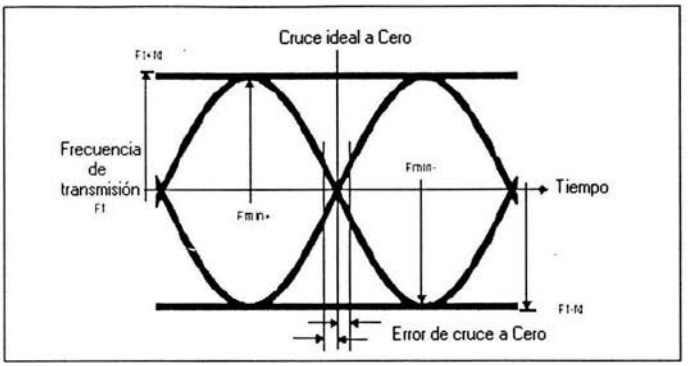

Figura 3-2. Esquema de modulación GFSK utilizada por Bluetooth.

Con este esquema y una tasa de símbolos para el canal de **1 Mbaudios** por segundo, se logra una tasa bruta de **1 Mbps**. La tasa de datos efectiva es significativamente inferior a este último valor debido a que cada paquete emplea un encabezado de tamaño considerable y una ventana de tiempo que se utiliza para permitirle al oscilador local de cada dispositivo sincronizarse en la nueva frecuencia de salto, además; otro elemento que reduce aún más la tasa en aplicaciones es el esquema de Solicitud de Repetición Automática (ARQ, Automatic Repeat Request), el cual lleva a cabo la retransmisión de paquetes ante la presencia de errores que pueden detectarse pero no corregirse.

3.2.4 Características del receptor

El receptor debe tener un nivel de sensibilidad para el cual se debe encontrar una tasa de error (BER) del **0.1%**. Un transreceptor que desee tomar parte del enlace de potencia controlada debe de poder medir su propia intensidad de señal que recibe y determinar si el transmisor del otro lado necesita aumentar o disminuir su potencia de salida, el RSSI hace esto posible, de esta forma el control de potencia tiene un *rango dorado de potencia recibida*, este rango es el umbral entre los límites máximos y mínimos de los niveles de potencia, el límite inferior corresponde a los **–56 dBm** y **6 dB** respecto a la sensibilidad actual del receptor, el límite superior es de **20 dB** y una exactitud de **+/-6dB**. Las instrucciones de alterar la potencia del transmisor TX son acarreadas en el enlace LMP.

3.3 Especificación Banda Base

*La Banda Base es la capa física de **Bluetooth***. Ésta administra los canales físicos y los enlaza a otros servicios como la corrección de errores, selección del salto y la seguridad de **Bluetooth**, la capa de Bandabase se encuentra en la parte superior de la capa de Radio de la Pila de **Bluetooth**, el protocolo Banda Base está implementado como el **Controlador de Enlace**, el cual trabaja con el **Administrador de Enlace** para transportar rutinas como la conexión de enlace y el control de potencia. También administra enlaces síncronos y asíncronos, paquetes manejables y maneja la requisición de acceso de dispositivos que se encuentran en el área.

El transreceptor ocupa el esquema de división de tiempo dúplex (TDD, Time Division Duplex) alternando para transmitir y recibir. Por lo cual, aparte de saltar en frecuencia, también el tiempo es dividido.

3.3.1 Canal Físico

En Estados Unidos de Norteamérica y Europa, está disponible una banda de **83.5Mhz** de ancho; en la cual, están definidos **79 canales de RF** espaciados por **1 Mhz**. En algunos países donde el ancho de banda está restringido, como se había mencionado anteriormente, se reduce el tamaño de la banda disponible; como es el caso de Francia, donde están disponibles **23 canales de RF** espaciados por **1 Mhz**.

El canal es representado por una secuencia pseudo-aleatoria de saltos a través de los 79 o 23 canales. Dos o más dispositivos **Bluetooth** que usan el mismo canal, forman una *piconet*. En cada piconet, pueden existir un *maestro* (master) y uno o más *esclavo*(s) (slaves). La secuencia de salto es única para cada piconet y es determinada por la dirección del dispositivo maestro (**BD_ADDR**), la fase en la secuencia de salto es determinada por el reloj del maestro. El canal está dividido en ranuras de tiempo, donde cada ranura corresponde a un salto de frecuencia de radiofrecuencia.

Las ranuras de tiempo, son de **625 μs** de longitud y están divididos de acuerdo al reloj del dispositivo maestro.

El esquema de Duplexación por División de Tiempo (TDD, Time Division Duplexing) es usado cuando el maestro y el esclavo transmiten alternándose, el maestro debe de comenzar su transmisión solamente en ranuras de tiempo pares y el esclavo deberá comenzar a transmitir solamente en ranuras de tiempo impares.

El paquete de inicio deberá estar alineado con el principio de la ranura de tiempo. La figura 3-3 ilustra esto:

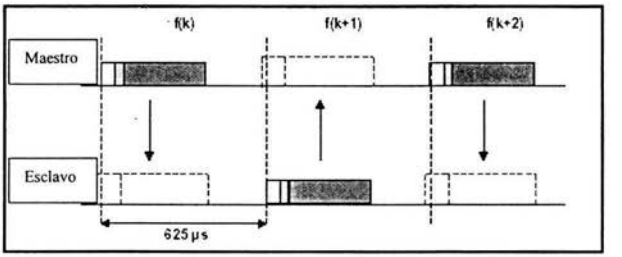

Figura 3-3. Transmisión en una piconet

3.3.2 Enlaces físicos

La Banda Base maneja dos tipos de enlace: Enlace Síncrono Orientado a la Conexión (SCO, Synchronous Connestion-Oriented) y Enlace Asíncrono sin Conexión (ACL, Asynchronous Connection-Less) El enlace SCO es un enlace simétrico punto a punto entre el maestro y un solo esclavo en la piconet, el maestro mantiene el enlace SCO usando ranuras reservados en intervalos regulares (tipo circuito switchado). Un enlace SCO se ocupa regularmente para la transmisión de voz, un dispositivo maestro puede soportar hasta tres enlaces SCO con uno o varios esclavos dentro de una misma piconet, como contraparte, un esclavo puede soportar hasta tres enlaces de este tipo con un mismo maestro, ó dos que se originen en dispositivos maestros diferentes, los paquetes SCO nunca son retransmitidos y son usados para transmitir conversaciones a **64 kB/s**.

En las ranuras no reservadas para enlaces SCO, el maestro puede intercambiar paquetes con cualquier esclavo dentro de la piconet, empleando para tal propósito una ranura exclusiva para cada uno de ellos, el enlace ACL provee una conexión de conmutación de paquetes entre el maestro y cualquier esclavo activo que se encuentre participando en la piconet. Sólo puede estar activo un único enlace ACL entre un maestro y un esclavo; sin embargo, el mismo puede utilizarse para soportar tanto servicios síncronos como asíncronos provenientes de capas superiores, para la mayoría de los paquetes ACL se aplica el proceso de retransmisión automática como una manera de asegurar la integridad de los datos.

A un esclavo se le permite devolver un paquete ACL en una ranura esclavo-maestro determinada sólo si el mismo ha sido colocado en la ranura maestro-esclavo previa. Si dicho esclavo falla en el intento de decodificar la dirección contenida en la cabecera del paquete anterior, el mismo no tiene autorización de transmitir.

3.3.3 Canales Lógicos

Bluetooth tiene cinco canales lógicos, los cuales hacen referencia a los diferentes tipos de canales que pueden correr sobre un enlace físico. El Canal de Control (LC, Control Chanel) y el Canal del Administrador de Enlace (LM, Link Manager) son usados a nivel de enlace, mientras los canales UA (User Asynchronous), UI (User Isochronous) y US (User Synchronous) son usados para transportar información del usuario asíncrona, isosíncrona y síncrona.

3.3.4 Direccionamiento de dispositivos

Existen cuatro tipos de direccionamiento para asignar unidades **Bluetooth**, **BD_ADDR**, **AM_ADDR**, **PM_ADDR** y **AR_ADDR.**

BD_ADDR: *Dirección del dispositivo **Bluetooth***: A cada transreceptor se le asigna una dirección de dispositivo única de 48 bits. La dirección está dividida en un campo LAP de 24 bits, un campo NAP de 16 bits y un campo UAP de 8 bits.

AM_ADDR: *Dirección de miembro activo*: Este es un número de 3 bits. Este número es válido únicamente en el período que el esclavo está activo en el canal. También es llamado en ocasiones como la dirección MAC de la unidad **Bluetooth**.

PM_ADDR: *Dirección de miembro estacionado*: Es una dirección del miembro (maestro-local) de 8 bits que separa a los esclavos estacionados. Esta solo es válida el período que el esclavo permanece estacionado.

AR_ADDR: *Dirección de demanda de acceso*: Esta es usada por el esclavo estacionado para determinar la ranura esclavo-maestro, y en la ventana de acceso se le permite mandar un mensaje de demanda de acceso. Es válida únicamente en el período que el esclavo está estacionado y no es necesariamente única.

3.3.5 Formato de los Paquetes

El formato de los paquetes, consiste en tres campos, el *Código de Acceso* (Access Code) de 72 bits, *la Cabecera* (Header) de 54 bits, y la *Carga Útil* (Payload) de 0 a 2745 bits, como se muestra en la figura 3-4.

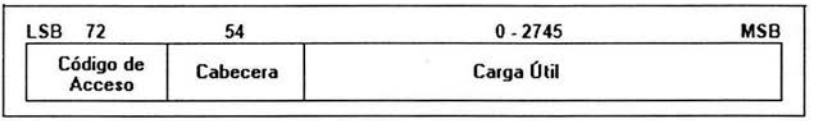

Figura 3-4. Formato de los paquetes.

Código de Acceso

Cada paquete comienza con un código de acceso, el cual se emplea para propósitos de señalización, éste código se subdivide en tres partes: Preámbulo (Preamble), Palabra de Sincronización (sync Word), y Cola (Trailer), como se muestra en la figura 3-5. El Preámbulo indica el arribo de un paquete al receptor. *La palabra de sincronización se utiliza para la sincronización temporal con el receptor*. La Cola se agrega a la Palabra de Sincronización en aquellos casos para los cuales se emplea una cabecera a continuación del Código de Acceso. El número de bits en el Código de Acceso puede variar, dependiendo de que si continúa o no una cabecera de paquete: Si sigue una cabecera de paquete, el código de acceso tiene **72 bits** de longitud: de otra forma tiene solamente **68 bits**.

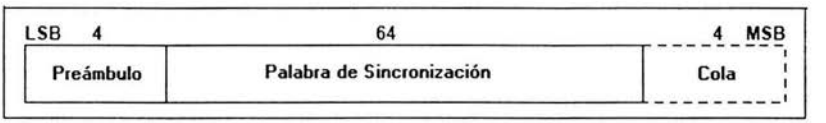

Figura 3-5. Formato del Código de Acceso.

Las funciones provistas por el Código de Acceso pueden ser diferentes, dependiendo del modo de operación del dispositivo. Existen tres tipos:

- *Código de Acceso al Canal* (CAC, Chanel Access Code): El CAC identifica una piconet, Este código está incluido en todos los paquetes intercambiados en el canal de la misma. Todos los paquetes enviados en una piconet comienzan en el mismo código de acceso al canal.

- *Código de Acceso al Dispositivo* (DAC, Device Access Code): El DAC se utiliza para procedimientos especiales de señalización, tales como un procedimiento de búsqueda o un procedimiento de respuesta a una búsqueda. Una búsqueda involucra la transmisión de una serie de mensajes con el objetivo de establecer un enlace de comunicación hacia una unidad activa dentro del área de cobertura. Cuando esa unidad responde a la búsqueda, se puede establecer el enlace.

- *Código de Acceso de Averiguación* (IAC, Inquiry Access Code): Existen dos tipos de códigos IAC: general y dedicado. El IAC general es común a todos los dispositivos, y se utiliza para descubrir otras unidades que se encuentren dentro del área. El IAC dedicado es propia a un grupo de unidades que comparten una característica común, y se utiliza para descubrir solamente aquellos dispositivos dedicados que se encuentren en rango.

Cabecera

Si se utiliza, contiene información de control del enlace y consiste de seis campos que totalizan 18 bits:

- *Dirección de Miembro Activo* (AM_ADDR) de 3 bits.

- *Tipo* (Type) de 4 bits: Se utiliza para alojar un código que especifica el tipo de paquete (SCO o ACL). Hay cuatro tipos de paquetes SCO y siete tipos de paquetes ACL, dentro del código *Tipo* se especifica también el número de ranuras de tiempo que ocupará el paquete en cuestión. Esto permite a los receptores no seleccionados abstenerse de escuchar el canal durante la duración de los restantes **TS** ante la presencia de paquetes multiranura.

- *Flujo* (Flow) de 1 bit: Se emplea para el control de flujo de los paquetes sobre un enlace ACL, cuando el buffer del receptor en un enlace ACL está lleno, se devuelve una indicación *detener* (stop) para interrumpir temporalmente la transmisión de datos, cuando el buffer del receptor se vacía, se devuelve una indicación de *seguir* (go). Cuando no se reciben paquetes, se asume la presencia de la indicación *seguir*.

- *Solicitud de Repetición Automática* (ARQ, Automatic Repeat Request) de 1 bit: Este campo se utiliza para indicar al transmisor sobre la transferencia exitosa de los datos. El éxito de la recepción se verifica por medio de un código de Verificación de Redundancia Cíclica (CRC, Cyclic Redundancy Check). La notificación puede ser un acuse de recepción positiva (ACK, Acknowledgment) o un acuse de recepción negativa (NAK, Negative Acknowledment).

- *Número de Secuencia* (SEQN, Sequence Number) de 1 bit: Este campo provee un esquema de numeración secuencial para colocar al flujo de paquetes de datos en el orden correcto durante la recepción, por cada nuevo paquete transmitido, se invierte el bit del número de secuencia con la idea de filtrar en el destino las posibles retransmisiones.

- *Verificación de Error en la Cabecera* (Header Error Check) de 8 bits: Este campo se utiliza para verificar la integridad de la Cabecera.

La figura 3-6 muestra el formato de la Cabecera.

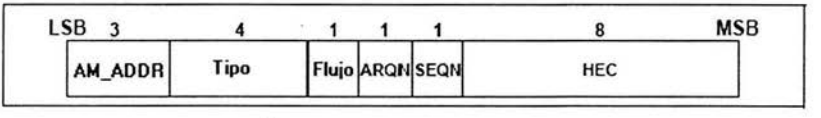

Figura 3-6. Formato de la Cabecera.

Carga útil

En esta parte, existen dos tipos de campo: el campo de voz (síncrono) y el campo de datos (asíncrono). El campo de datos está compuesto por tres segmentos:

- *Cabecera de la Carga Útil* (Payload Header) de 8 bits: Solo los campos de datos tienen esta cabecera. La misma tiene una longitud de uno o dos bytes; especifica el canal lógico, ejecuta el control de flujo en el mismo, y tiene un indicador de longitud de carga útil. Este último especifica el número de bytes en el cuerpo de carga útil, excluyendo la cabecera y el código CRC.

- *Cuerpo de la Carga Útil* (Payload Body) de 0 a 2721 bits: Este segmento incluye la información del usuario. La longitud del mismo está especificada en el indicador de longitud de la cabecera de la carga útil.

- *Verificación de Redundancia Cíclica* (Cyclic Redundancy Check) de 16 bits: Antes de iniciarse el generador CRC, se calcula un código de verificación de redundancia de 16 bits sobre la información a transmitirse y se añade a la misma.

3.3.5 Tipos de paquetes

Los paquetes comunes son aquellos que pueden usarse tanto en enlaces ACL como SCO, hay cinco tipos:

- Paquete **ID**: Consiste en el código DAC o IAC sin cabecera ni carga útil, se utiliza generalmente como respuesta a las solicitudes de *búsqueda* (page) o *averiguación* (inquiry) y es de 68 bits.

- Paquete **NULL**: consiste en el código CAC y la cabecera del paquete, pero sin carga útil. Se utiliza para devolver los valores de los campos ARQN y FLOW.

- Paquete **POLL**: es similar al paquete NULL. Un dispositivo maestro podría emplear este tipo de paquete para realizar una operación de sondeo de un dispositivo esclavo que necesite responder con un acuse de recibo, sin importar si el último tiene o no datos para transmitir.

- Paquetes **FHS**: Se usa para controlar y revelar información de la piconet a los dispositivos miembros como la dirección BD_ADDR y los datos de temporización del sistema. Para proteger la carga útil, se utiliza un código de corrección de error. El paquete contiene en total 240 bits.

- Paquete **DM1**: Puede transportar datos regulares, pero tiene la capacidad de interrumpir un enlace síncrono con el propósito de enviar información de control.

También existen paquetes SCO, y están reservados para transmitir voz, por lo que no contienen código CRC ni implementan mecanismos de retransmisión:

- Paquete **HV1** (High-quality voice): Este paquete puede transportar **1.25 ms** de voz codificada a **64 kps** y se requiere enviar un paquete HV1 cada dos ranuras de tiempo para mantener un enlace SCO.

- Paquete **HV2**: es similar al HV1. Soporta hasta **2.5 ms** de voz.

- Paquete **HV3**: soporta hasta **3.75 ms** de voz.

- Paquete **DV**: contiene una mezcla de voz y datos (**80 bits** de voz y hasta **150 bits** de datos).

El último tipo de paquetes, son los paquetes ACL para enlaces asíncronos, y están reservados para transmitir datos. A pesar de que el paquete DM1 está asignado como paquete común, se considera como paquete ACL. Los paquetes DM1 se utilizan en enlaces SCO para transportar información de control.

- Paquete **DM1** Datos de Tasa Media 1 (Data Medium Rate 1): La carga útil contiene **18 bytes** de información y un código CRC de 16 bits y ocupan una ranura de tiempo. Se codifica con un código de corrección de error con tasa 2/3.

- Paquete **DM3**: Ocupan 3 ranuras de tiempo y pueden transmitir **123 bytes** de datos.

- Paquete **DM5**: Ocupa 5 ranuras de tiempo y puede transportar **226 bytes** de información.

- Paquete **DH1** Datos de Tasa Alta 1 (Data High Rate 1): A diferencia de la serie DM, no transportan código de corrección de error, solo utilizan el código CRT combinado con un esquema ARQ. Pueden transportar hasta **28 bytes** de información.

- Paquetes **DH3**: Ocupan 3 ranuras de tiempo y pueden transporta hasta **185 bytes** de información.

- Paquetes **DH5**: pueden transportar hasta **341 bytes** de información y ocupan 5 ranuras de tiempo.

- Paquete **AUX1**: Es una variante del paquete DH1 y ocupan una ranura de tiempo y no utilizan código CRC para la carga útil; pudiendo transportar **30 bytes** de información y no es posible la retransmisión.

3.3.6 Estados de Bluetooth

Hay dos estados primarios soportados por el protocolo LMP: *Espera* (Standby) y *Conexión* (Connection). Aparte de estos dos, hay siete estados que se utilizan para agregar un nuevo dispositivo esclavo a una piconet determinada. Ellos son:

- Estado *Búsqueda* (Page)
- Estado *Exploración de Búsqueda* (Page Scan)
- Estado *Averiguación* (Inquiry)
- Estado *Exploración de Averiguación* (Inquiry Scan)
- Estado *Respuesta del Maestro* (Master Response)
- Estado *Respuesta del Esclavo* (Slave Response)
- Estado *Respuesta de Averiguación* (Inquiry Response)

En la figura 3-7 se muestra un diagrama que ilustra los diferentes estados utilizados por un dispositivo **Bluetooh**.

Estado *Espera*

El estado Espera es el estado de arranque por default de una unidad Bluetooth. En el mismo, la unidad se encuentra en modo de bajo consumo de potencia.

Si un dispositivo emite un mensaje *búsqueda* o *averiguación*, recibiendo la correspondiente respuesta por parte de un dispositivo remoto, entonces entra en el estado *Conexión* cumpliendo el rol de maestro, en el extremo opuesto, si un dispositivo realiza una exploración en busca de un mensaje *búsqueda* o *averiguación*, el mismo entra en el estado *Conexión* cumpliendo el rol de esclavo tan pronto como encuentre y responda dicho mensaje.

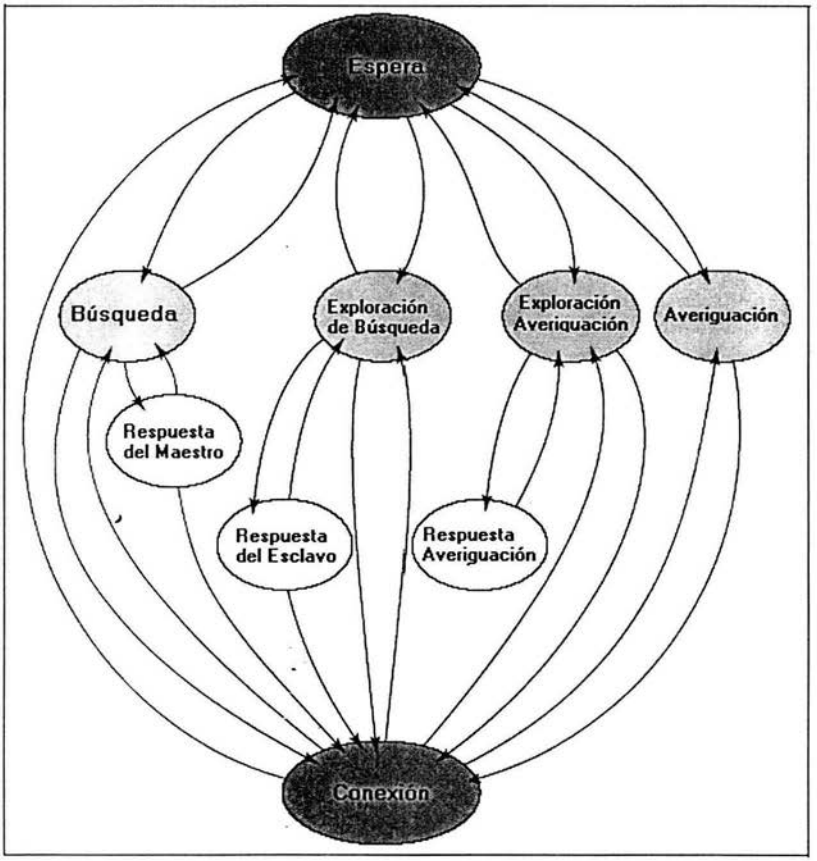

Figura 3-7. Diagrama de estados del controlador del enlace Bluetooth.

3.3.7 Procedimientos de acceso

Para establecer nuevas conexiones se utilizan los procedimientos *Averiguación (Inquiry)* y *Buscando (Paging)*. Normalmente si nada es conocido acerca del dispositivo remoto, se llevan a cabo ambos procedimientos. Si se conoce algún detalle del dispositivo remoto, solo se lleva a cabo el procedimiento de *Buscando*.

Paso 1:

El procedimiento *Averiguación,* permite a un dispositivo descubrir que dispositivos se encuentran en el rango y determina la dirección y relojes de los dispositivos.

a) El procedimiento Averiguación involucra a una unidad (la unidad fuente) a mandar paquetes de averiguación (Estado de *Averiguación*) y después recibe la respuesta de averiguación.

b) La unidad que recibe los paquetes Averiguación (el destino), entrará en Estado Exploración de Averiguación para recibir los paquetes de averiguación.

c) El destino entrará en Estado Respuesta de Averiguación y mandará una respuesta de averiguación a la fuente.

Después del proceso de Averiguación, se puede establecer una conexión usando el procedimiento *Buscando.*

Paso 2:

Con el procedimiento *Buscando*, se puede establecer una conexión, este procedimiento por lo general sigue el procedimiento de Averiguación, solamente se requiere la dirección del dispositivo para establecer una conexión. El conocimiento del reloj (un estimado), acelerará el procedimiento de establecimiento de conexión, una unidad que establezca la conexión; transportará el procedimiento de Descubrimiento y será automáticamente el maestro de la conexión. A continuación se describe el procedimiento:

a) Un dispositivo en estado de Búsqueda (la fuente) busca otro dispositivo (el destino).

b) El destino en estado de Exploración de Búsqueda recibe el mensaje de *Búsqueda.*

c) El destino en estado de Respuesta de Esclavo manda una respuesta a la fuente (Paso 1)

d) La fuente en estado de Respuesta del Maestro manda un paquete FHS al destino (Paso 1)

e) El destino en estado de Respuesta del Esclavo manda su segunda respuesta a la fuente (Paso 2)

f) El destino y la fuente intercambian los parámetros del canal.

La conexión comienza con un paquete **POLL** enviado por el maestro para verificar que el esclavo este switcheando el canal. El esclavo puede responder con cualquier tipo de paquete.

3.3.8 Modos de Conexión

Durante el estado de Conexión las unidades **Bluetooth** pueden entrar en cuatro modos de operación diferentes: *Activo* (Active), *Olfateo* (Sniff), *Sostenido* (Hold), *Estacionado* (Park).

Modo *Activo*: La unidad participa activamente en el canal, el maestro planifica las transmisiones basándose en las demandas del tráfico entrante y saliente de los diferentes esclavos y realiza transmisiones regulares para mantener a los esclavos sincronizados al canal, los esclavos escuchan las ranuras maestro-esclavo en busca de paquetes dirigidos a ellos, para ahorrar energía, estos dispositivos tienen la opción de ingresar en modo de bajo consumo de potencia mientras el maestro no los esté direccionando.

Modo de *Olfateo*: Mientras que en estado *Activo*, un dispositivo esclavo necesita escuchar todas las ranuras de tiempo maestro-esclavo en busca de posibles paquetes que sean para él; en el modo de *Olfateo* se reduce esta actividad. Esto se logra programando un juego específico de ranuras de tiempo durante el cual un dispositivo maestro intentará comunicarse con un esclavo, y viceversa. Para ingresar en este modo, el dispositivo maestro o esclavo debe emitir un comando de olfateo a través del protocolo LMP. Este mensaje debe contener los parámetros que determinan el intervalo *olfateo*, el desplazamiento, y el número de intentos.

Modo *Sostenido*: En este modo, el enlace ACL hacia el esclavo es colocado en modo *Sostenido*, aunque dentro del mismo el esclavo no soporta temporalmente paquetes ACL sobre el canal, aquellos enlaces SCO que posiblemente se encuentren activos siguen utilizándose con normalidad. Debido a que durante el modo *Sostenido* el esclavo sigue perteneciendo a la piconet, el mismo no pierde su dirección **AM_ADDR**. Para ingresar a este modo, el dispositivo maestro o esclavo deben emitir un comando *Sostenido* específico a través del protocolo **LMP**. En dicho proceso ambos dispositivos deben ponerse de acuerdo en el lapso de tiempo durante el cual el esclavo permanecerá dentro del modo *Sostenido*, cada vez que éste ingresa dentro de este modo de bajo consumo de potencia, se inicia un temporizador *holdTO* (Hold Timeout). Una vez expirado el mismo, el esclavo despierta y sincroniza nuevamente con el tráfico del canal en busca de nuevas instrucciones provenientes del maestro. Gracias al empleo de este modo de operación, se puede lograr que un mismo dispositivo libere capacidad para llevar a acabo otros tipos de procesos como búsquedas, averiguaciones, y exploraciones; además, controlando en forma adecuada el ingreso a este modo se puede lograr que el mismo dispositivo puede cumplir roles distintos en dos o más piconets diferentes.

Modo *Estacionado*: Cuando un esclavo no necesita participar activamente en el canal de la piconet, pero desea permanecer sincronizado a la misma, puede entrar es este modo.

El modo *Estacionado* es un modo de bajo consumo de potencia con muy poca actividad en el esclavo, aquí el esclavo renuncia a su dirección **AM_ADDR**, pero recibe dos nuevas direcciones específicas de este modo de operación; la *Dirección de Miembro Estacionado* **PM_ADDR** para distinguir a los esclavos estacionados, y la *Dirección de Solicitud de Acceso* **AR_ADDR** que se utiliza para que el esclavo solicite la salida de este modo por su propia iniciativa.

3.3.9 Scatternet

Varias piconets pueden encontrarse en la misma área. Dado que cada piconet tiene su propio maestro, los saltos en frecuencia son independientes, cada una tiene su propia secuencia de salto de canal y faces determinadas por el maestro. Los paquetes transportados en los canales son procesados por diferentes códigos de acceso determinados por la dirección del dispositivo maestro.

Si varias piconets se encuentran en la misma área, una unidad puede participar en dos o más piconets aplicando multiplexión de tiempo; para participar en el canal apropiado, deberá usar la dirección del maestro asociado y el desplazamiento de reloj para obtener la fase correcta. Una unidad **Bluetooth** puede actuar como esclavo en varias piconets, pero como maestro solamente en una piconet. Un grupo de piconets en las que comparten algún dispositivo, se le conoce como *Scatternet*, como se muestra en la figura 3-8.

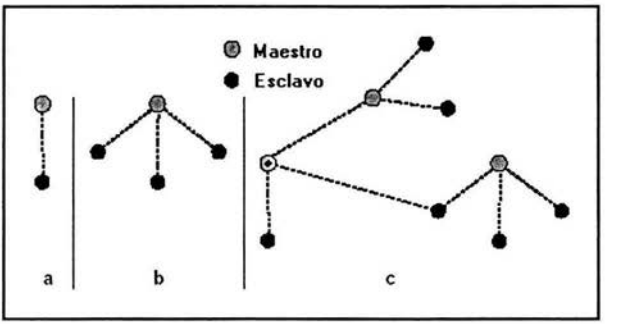

Figura 3-8. (a) Piconet con un solo esclavo, (b) Piconet con varios esclavos, (c) Scatternet.

A veces algún dispositivo quiere intercambiar papeles (maestro-esclavo), este procedimiento consta de dos pasos:

a. Primero un switch TDD a consideración del esclavo y el maestro, seguido de un cambio de piconet de ambos participantes.

b. Después, si se desea, otros esclavos de la vieja piconet, pueden ser transferidos a la nueva piconet.

Cuando una unidad tiene el conocimiento de recepción del paquete **FHS**, esta unidad usa los parámetros de la nueva piconet definidos por el nuevo maestro y el cambio de piconet está completo.

3.3.10 Corrección de errores

Existen tres tipos de esquemas de corrección de errores usados en el protocolo Banda Base: Corrección de Error Directa FEC, con tasa 1/3, FEC con tasa 2/3 y el esquema de Solicitud Automática de Retransmisión ARQ.

Con FEC, hay bits redundantes insertados en los datos que se transmiten que permiten identificar y corregir los errores en el receptor, sin necesidad de retransmisión. Con ARQ, cuando se detecta un error, se envía un mensaje al remitente para solicitar una retransmisión de los datos en los que ha ocurrido el error.

Para FEC se emplea paridad, o técnicas CRC. Para ARQ, una vez que se ha detectado un error en el extremo receptor, se transmite un mensaje corto al remitente, que éste interpreta como una solicitud de reenvío del último bloque de datos. Esto continuará hasta que se obtengan datos libres de errores.

- En el FEC con tasa 1/3, cada bit es representado tres veces por redundancia, como se muestra en la figura 3-9.

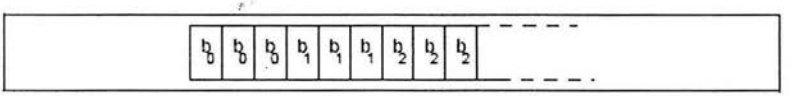

Figura 3-9. Representación de corrección FEC con tasa 1/3.

- En el FEC con tasa 2/3, se emplea un generador polinomial para codificar 10 bits en un código de 15 bits.

- En el esquema ARQ, los campos DM, DH y el campo de datos de los paquetes DV son retransmitidos hasta que se recibe un indicador de que se ha recibido la información correctamente, **Bluetooth** usa un indicador con valor positivo o negativo configurando los valores adecuados del ARQ. Si se excede el tiempo de espera, **Bluetooth** desecha el paquete y manda el siguiente paquete.

3.3.11 Control de Flujo

El protocolo Banda Base recomienda usar colas FIFO en enlaces ACL y SCO. El Administrador de Enlace llena estas colas y el Controlador de Enlace se encarga de vaciarlas automáticamente.

Si las colas FIFO del receptor RX están llenas, el control de flujo es usado para evitar la perdida de paquetes, si los datos no pueden ser recibidos, un indicador *detener* es transmitido por medio del Controlador de Enlace del receptor en la cabecera del paquete de respuesta. Cuando el transmisor recibe la indicación de *detener*, para la cola FIFO. Cuando el receptor está listo, manda un paquete con un indicador *seguir*.

3.3.12 Sincronización

El transreceptor usa un esquema de división de tiempo dúplex TDD, esto es, que puede transmitir y recibir alternativamente de manera síncrona.

Como se había mencionado anteriormente, una piconet es sincronizada por el reloj del sistema del maestro. Para transmitir en el canal de la piconet, se necesitan tres elementos; la Secuencia de Salto (del canal), la fase de la secuencia, y el Código de Acceso al Canal CAC que necesitan llevar los paquetes.

a) **La Secuencia de Salto del Canal**: La Dirección **Bluetooth** del dispositivo BD_ADDR del maestro se utiliza para derivar esta secuencia.

b) **Fase**: El reloj del sistema del maestro determina la fase en la secuencia de saltos.

c) **Código del Acceso al Canal**: Esta es derivada por la Dirección **Bluetooth** BD_ADDR del dispositivo maestro.

Los esclavos adaptan sus relojes nativos con el desplazamiento de tiempo para coincidir con el reloj del maestro, dando un valor estimado de reloj, el desplazamiento del reloj nativo del maestro es cero, el reloj **Bluetooth** debe tener una unidad de tick LSB de **312.5µs**, dando una tasa de reloj de **3.2 kHz**.

Existe una ventana de incertidumbre de **20µs** en el tiempo exacto de recepción para permitir al receptor buscar el código correcto de acceso y sincronizarse con el transmisor, cuando un esclavo regresa del modo *sostenido*, este tiempo de incertidumbre es mayor hasta que se pueda sincronizarse. Un esclavo estacionado, despierta periódicamente para escuchar el canal y re-sincronizar su desplazamiento de reloj.

3.3.13 Seguridad Bluetooth a nivel de enlace

En la capa de enlace, se tiene seguridad por medio de la **autentificación** de parejas y **encriptación** de la información. Para mantener la seguridad a nivel de enlace, se utilizan cuatro parámetros:

- La dirección del dispositivo **Bluetooth** (BD_ADDR).
- La clave de usuario privado de autenticación o de enlace.
- La clave de usuario privado de cifrado.
- Un número aleatorio (RAND).

La clave de autenticación tiene una longitud fija de **128 bits**, mientras que la de cifrado, que normalmente se obtiene a partir del proceso de autenticación, tiene una longitud variable, entre **8** y **128 bits**. El número aleatorio vendrá derivado de un proceso aleatorio o pseudo-aleatorio que tendrá lugar en la unidad **Bluetooth**, este parámetro cambiará frecuentemente.

El primer paso es que un dispositivo haga la autentificación, mandando un desafío basado en la **BD_ADDR** y la clave de enlace compartidas entre ellas, el cual deberá ser contestado por el receptor. Si el proceso de autenticación falla, debe pasar un cierto tiempo antes de intentarlo de nuevo. Este tiempo se incrementa exponencialmente; de tal manera que el tiempo transcurrido después de un intento fallido sea por ejemplo el doble de tiempo de espera antes de ese fallo. Igualmente el tiempo de espera decrecerá exponencialmente cuando no hay fallos en un determinado periodo de tiempo, este procedimiento evita que un intruso repita el proceso de autenticación con un elevado número de claves de enlace diferentes; después de la autentificación se podrá usar la encriptación para la comunicación.

La información de usuario puede ser protegida por encriptación de la carga útil, ya que; el Código de Acceso y la Cabecera del paquete nunca son cifrados. La encriptación se lleva a cabo con un algoritmo de cifrado, que consiste básicamente de tres partes: una parte que realiza la inicialización (generación de la clave de carga útil). Una segunda parte que es el generador de cadenas de claves, y finalmente una tercera parte en la cual se realiza la encriptación o la desencriptación.

Los parámetros de entrada a dicho algoritmo serán la clave de cifrado, la **BD_ADDR** del maestro y el reloj del mismo y un número aleatorio, cada paquete de carga útil es cifrado separadamente, lo cual se consigue si tenemos en cuenta que una de las entradas al algoritmo es el reloj del maestro; el cual cambia una unidad cada intervalo de tiempo (625μs), por lo que la clave de carga útil será diferente para cada paquete excepto para aquellos que ocupen más de un intervalo de tiempo, en cuyo caso el valor del reloj del primer intervalo de tiempo del paquete será el que se utilizará para todo el paquete.

El algoritmo de encriptación genera una cadena de claves que son sumadas a los datos que se desean encriptar. El desencriptado se realizará exactamente de la misma manera usando la misma clave que se usó para la encriptación.

3.4 Protocolo del Administrador de Enlace (LMP, Link Manager Protocol)

El *Administrador de Enlace* (LM, "Link Manager") soporta el establecimiento, la autentificación, la configuración del enlace y otros protocolos. *El Administrador de Enlace* localiza a otros administradores y se comunica con ellos gracias *al Protocolo del Administrador de Enlace* (LMP, "Link Manager Protocol"). Para poder realizar su función de proveedor de servicio, el LM utiliza los servicios incluidos en el *Controlador de Enlace* (LC, "Link Controller").

El Protocolo del Administrador de Enlace consiste principalmente en un número de *Unidades de Datos del Protocolo* "PDUs" (Protocol Data Units), el cual es mandado de un dispositivo a otro, determinado por la AM_ADDR en la cabecera del paquete. Los PDUs del LM son enviados siempre como paquetes sencillos en una sola ranura y la cabecera de la carga útil es de un byte.

Se utilizan paquetes DM1 para transportar los PDUs, excepto cuando se presenta un enlace SCO utilizando paquetes HV1 con contenido menor a 9 bytes; en este caso se utilizan paquetes DV.

En el apéndice C se pueden encontrar los tipos de PDUs, así como sus funciones y operaciones y si son o no obligatorios.

3.5 Interfaz del Controlador Host (HCI, Host Controller Interface)

El **HCI** provee una interfaz de comandos al controlador **Banda Base** y el **Administrador de Enlace**, también provee acceso al estado del hardware y registros de control. En resumen esta interfaz provee un método uniforme de acceso a las capacidades de la **Banda Base Bluetooth**. El HCI existe a través de 3 secciones, el Host - Capa de Transporte - Controlador Host. Cada sección tiene un papel diferente en el sistema HCI.

3.5.1 Entidades funcionales HCI

El HCI está dividido funcionalmente en tres partes, como se muestra en la figura 3-10:

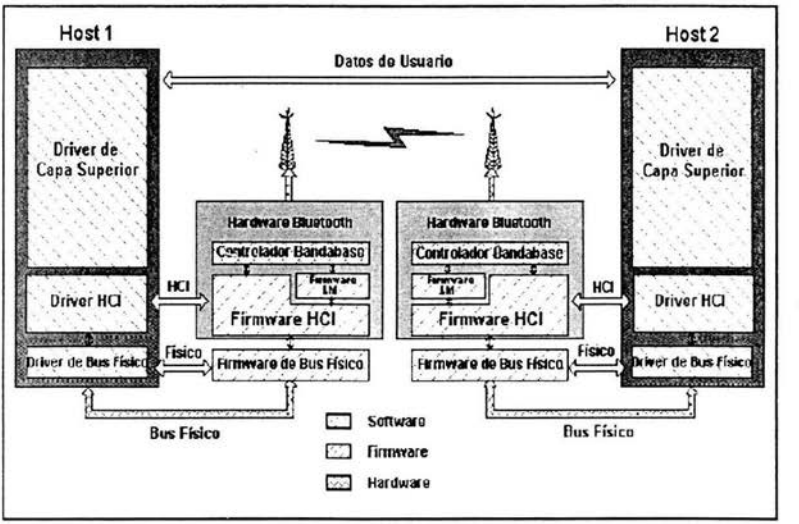

Figura 3-10. Entidades funcionales HCI.

HCI Firmware (localización: Controlador Host)

El Firmware HCI, está localizado en el **Controlador Host** (por ejemplo en el hardware **Bluetooth** del dispositivo actual) implementa los **Comandos HCI** para el Hardware **Bluetooth** accesando a los comandos Banda Base, comandos del Administrador de Enlace, registros de estado del hardware, registros de control, y registros de eventos. El término **Controlador Host** significa el dispositivo **Bluetooth HCI-habilitado**.

Manejador HCI (localización: Host)

El Manejador HCI se encuentra en el Host (por ejemplo en la entidad software). El Host recibirá notificaciones asíncronas de eventos HCI, los eventos HCI se usan para notificar al Host cuando algo ha ocurrido. Cuando el Host descubre que un evento ha ocurrido, analizará gramaticalmente el paquete de evento recibido para determinar que evento ha ocurrido. *El término **Host** significa la Unidad Software **HCI-habilitada**.*

Capa de Transporte del Controlador Host (localización: Capas Intermedias)

El Manejador HCI y el Firmware se comunican vía la **Capa de Transporte del Controlador Host**; varias capas que podrían existir entre el manejador HCI en el sistema Host y el Firmware HCI en el hardware **Bluetooth**. Estas capas intermedias, la Capa de Transporte del Controlador Host, deben proveer la habilidad de transmitir datos sin tomar en cuenta su contenido. Pueden ser usadas diferentes tipos de Capas del Controlador Host, de las cuales se cuentan con tres definidas inicialmente para **Bluetooth**: **USB**, **UART** y **RS232**. El Host debe recibir notificaciones asíncronas de los eventos HCI independientemente de que Capa de Transporte del Controlador Host es usado.

3.5.2 Comandos HCI

El HCI provee un método de comandos uniforme de acceso a las capacidades del hardware **Bluetooth**. Los comandos de Enlace HCI provee al Host con la habilidad de controlar las conexiones de la capa de enlace a otros dispositivos **Bluetooth**. Estos comandos involucran por lo general al LM para cambiar comandos LMP con dispositivos **Bluetooth** remotos. Estos comandos de Políticas proveen al Host con métodos para influenciar cómo el LM maneja la piconet. El Controlador Host y los comandos Banda Base, los comandos Informativos, y los comandos de Estado, permiten al Host accesar a varios registros en el Controlador Host.

Intercambio de Información Especifica-HCI

La Capa de Transporte del Controlador Host permite el intercambio transparente de información específica. Estos mecanismos de transporte proveen la habilidad para el Host de mandar comandos HCI, datos ACL y datos SCO al Controlador Host.

Comandos de Control de Enlace

Los comandos de Control de Enlace permiten al Controlador Host controlar las conexiones a otros dispositivos **Bluetooth**. Cuando se usan estos comandos, el LM controla como son establecidas y mantenidas las piconets y scatternets. Estos comandos instruyen al LM para crear y modificar las conexiones de la capa de enlace con dispositivos **Bluetooth** remotos, realiza solicitudes de otros dispositivos **Bluetooth** en el rango y otros comandos LMP.

Comandos de la Política de Enlace

Estos comandos proveen de métodos al Host para afectar como el LM maneja la piconet. Cuando se usan estos comandos, el LM sigue controlando como se establecen las piconets y scatternets, dependiendo de los parámetros de las políticas.

Comandos del Controlador Host y Bandabase

Estos comandos proveen acceso y control a varias capacidades del hardware **Bluetooth**. Estos parámetros proveen el control de dispositivos **Bluetooth** y de las capacidades del Controlador Host, Administrador de Enlace, y Bandabase. El dispositivo host puede usar estos comandos para modificar el comportamiento del dispositivo local.

Parámetros Informativos

Estos parámetros son establecidos por los fabricantes de hardware **Bluetooth**. Estos parámetros proveen información del dispositivo **Bluetooth** y las capacidades del Controlador Host, el Administrador de Enlace, y Bandabase. El dispositivo host no puede modificar estos parámetros.

Parámetros de Estado

El Controlador Host modifica todos los parámetros de estado. Estos parámetros proveen información acerca del estado actual del Controlador Host, del Administrador de Enlace, y Banda Base.

Comandos de Prueba

Estos comandos se usan para proveer la habilidad de probar la funcionalidad del hardware **Bluetooth**. Estos comandos permiten modificar las condiciones para las pruebas.

3.5.3 Eventos HCI/Códigos de Error/Control de Flujo

Control de Flujo

El Control de Flujo se usa en dirección del Host al Controlador Host, para permitir llenar los buffers de datos del Controlador Host con datos ACL destinados para dispositivos remotos que no estén respondiendo. Por lo que el Host administra los buffers de datos del Controlador Host.

Eventos HCI

Un número de diferentes eventos están definidos para la capa HCI. Estos eventos proveen un método para regresar parámetros y datos asociados a cada evento. Se han implementado más de 32 eventos HCI.

Códigos de Error HCI

Existe un gran número de códigos de errores para la capa HCI. Cuando un comando falla, se regresa un Código de Error para indicar la razón del mismo. Se han definido más de 35 códigos de error HCI.

3.5.4 Capas de transporte del Controlador Host definidas por Bluetooth

Capa de Transporte UART

El objetivo de esta capa, es hacer posible el uso del HCI **Bluetooth** en una interfaz serial entre dos dispositivos en el mismo PCB. La capa de transporte HCI UART asume que la comunicación UART está libre de errores de línea. Los eventos y datos a través fluyen en esta capa, pero la capa no los decodifica.

Capa de Transporte RS232

El objetivo de esta capa es hacer posible el uso del HCI **Bluetooth** en la interfaz física RS232 entre el Host **Bluetooth** y el Controlador Host **Bluetooth**. Los eventos y los datos fluyen a través de esta capa, pero la capa o los decodifica.

Capa de Transporte USB

El objetivo de la Capa de Transporte Bus Serial Universal (USB, Universal Serial Bus) es usar una interfaz de hardware USB para hardware **Bluetooth** (el cual puede estar incorporado de dos maneras: como un dongle USB, o integrado en la motherboard de una computadora portátil). Se usa un código clase, el cual está especificado para todos los dispositivos **Bluetooth** USB. Esto permitirá al propio manejador funcionar, sin importar el fabricante del dispositivo. Esto también permite diferenciar los comandos HCI de los comandos USB a través de los puntos finales de control.

3.6 Protocolo de Adaptación y Control de Enlace Lógico (L2CAP, Logical Link Control and Adaptation Protocol)

El **L2CAP** se encuentra sobre la capa de Protocolo Bandabase y reside en la capa de enlace de datos. El L2CAP provee servicios de datos *orientados a conexión* y *sin conexión* a los protocolos de capas más altas con capacidad de multiplexión de protocolos, segmentación y operación de reensamble y abstracciones de grupo. El L2CAP permite a los protocolos y aplicaciones de niveles más altos transmitir y recibir paquetes de datos L2CAP hasta de **64 kB** de longitud. En la figura 3-11 se muestra la interacción del L2CAP con otras capas.

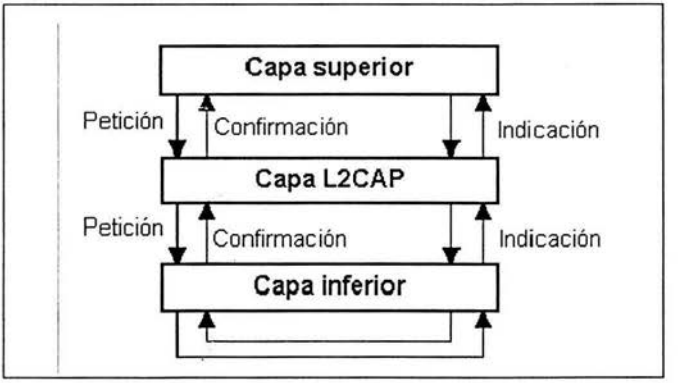

Figura 3-11. Arquitectura L2CAP.

Como lo vimos anteriormente, existen dos tipos de enlaces soportados para Banda Base: enlaces SCO y enlaces ACL. Los enlaces SCO soportan tráfico de voz en tiempo real usando un ancho de banda reservado. Los enlaces ACL soportan el mejor esfuerzo de tráfico. La especificación L2CAP está definida solamente para enlaces ACL y no soporta enlaces SCO.

3.6.1 Requerimientos Funcionales del L2CAP

El L2CAP soporta diferentes requerimientos importantes, los cuales son:

Multiplexaje de Protocolo: El L2CAP soporta multiplexaje de protocolo, debido a que el Protocolo Banda Base no soporta cualquier tipo de campo identificando el protocolo de una capa más alta que la está siendo multiplexada arriba. El L2CAP debe estar habilitado para distinguir entre protocolos de capas más altas como el Protocolo de Descubrimiento de Servicio, RFCOMM, y Control de Telefonía.

Segmentación y re-ensamble: Comparado con otros medios físicos cableados, los paquetes de datos definidos por el Protocolo Banda Base están limitados en tamaño. Exportando una **Unidad de Transmisión Máxima** (MTU) con la carga útil Banda Base

más larga (341 bytes para paquetes DH5), limita la eficiencia del ancho de banda para los protocolos de capas más altas designadas para usar paquetes más largos. Los paquetes largos del L2CAP deben ser segmentados en múltiples paquetes Banda Base más pequeños, antes de transmitirlos. Por lo cual, los múltiples paquetes pequeños Banda Base recibidos deben ser re-ensamblados en un solo paquete L2CAP más largo siguiendo un simple chequeo de integridad; la segmentación y re-ensamble (SAR) son absolutamente necesarios para soportar protocolos que usan paquetes más largos que el soportado por la Banda Base. La figura 3-12 muestra la segmentación L2CAP.

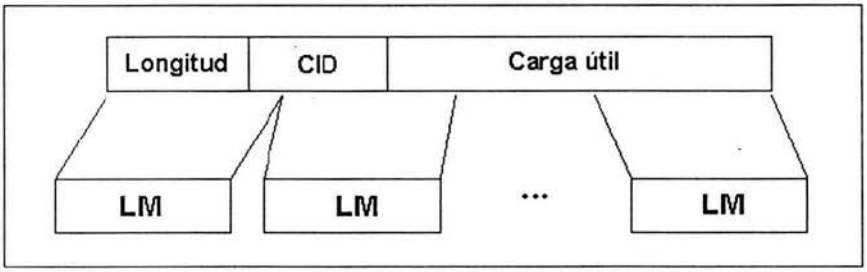

Figura 3-12. Segmentación L2CAP.

Calidad de Servicio: El proceso de establecimiento de conexión L2CAP, permite el intercambio de información con respecto a la calidad de servicio (QoS) entre dos unidades **Bluetooth**. Cada implementación L2CAP debe monitorear los recursos usados por el protocolo para asegurar que el QoS se lleve a cabo.

Grupos: Varios protocolos incluyen el concepto de un grupo de direcciones, el Protocolo Banda Base soporta el concepto de una piconet, un grupo de dispositivos sincronizados para brincar en frecuencia juntos usando el mismo reloj. La abstracción de grupo L2CAP permite implementaciones para un eficiente mapeo de grupo de protocolos en las piconets.

3.6.2 Operación general del L2CAP

La capa L2CAP está basada en el concepto de "*canales*", cada punto final de un canal L2CAP es referido por un *identificador del canal.*

Identificadores del Canal

Los **Identificadores del Canal** (CIDs) son nombres locales representando un canal lógico del punto-final en el dispositivo.

Segmentación y re-ensamble

Una implementación L2CAP expone los MTU salientes y segmenta los paquetes de capas más altas en "chunks" los cuales pueden ser pasados al Administrador de Enlace vía el HCI, siempre y cuando exista uno.

En el lado receptor, una implementación L2CAP recibe los "chunks" del HCI y re-ensambla estos chunks en paquetes L2CAP usando información proveída por el HCI y la cabecera del paquete.

3.7 El Protocolo RFCOMM

El Protocolo **RFCOMM** provee la emulación de puertos seriales sobre el protocolo L2CAP. El protocolo está basado en el estándar **ETSI TS 07.10**. Solo un subconjunto del estándar TS 07.10 es usado, y se usan algunas adaptaciones y extensiones obligatorias en la especificación RFCOMM de **Bluetooth**.

El protocolo RFCOMM soporta hasta 60 conexiones simultáneas entre dos dispositivos; para los propósitos del RFCOMM, un camino de comunicación completo involucra dos aplicaciones corriendo en diferentes dispositivos (los puntos finales de la comunicación) con un segmento de comunicación entre ellos.

El RFCOMM emula los 9 circuitos de una interfase RS-232. En la tabla 3-1 se muestra una lista con el nombre de los pines.

Nombre del Pin
102 Señal común
103 Transmisión de Datos (TD)
104 Recepción de Datos (RD)
105 Solicitud para mandar (RTS)
106 Limpio para mandar (CTS)
107 Datos listos (DSR)
108 Terminal de Datos lista (DTR)
109 Detección de Transporte de Datos (CD)
125 Indicador de Ring (RI)

Tabla 3-1. Nombre de los pines de las señales de control.

Emulación de múltiples Puertos Seriales

Los dispositivos que usan el RFCOMM en sus comunicaciones, pueden emular múltiples puertos seriales, como se muestra en la figura 3-14. RFCOMM soporta la emulación de hasta 60 puertos abiertos; un *Identificador de Conexión de Enlace de Datos* (DLCI, Data Link Connection Identifier) identifica una conexión saliente entre una aplicación cliente-servidor. El DLCI está representado por 6 bits, y se usa un rango de valores del 2 al 61. El DLCI es único en cada sesión RFCOMM.

Si un dispositivo soporta emular múltiples puertos seriales, y las conexiones son en diferentes puntos finales en diferentes dispositivos, entonces la entidad RFCOMM deberá estar disponible para correr varias sesiones TS 07.10 multiplexadas. Cada sesión utiliza su propio ID de canal L2CAP (CID). Esta habilidad de correr múltiples sesiones TS 07.10 es opcional para RFCOMM.

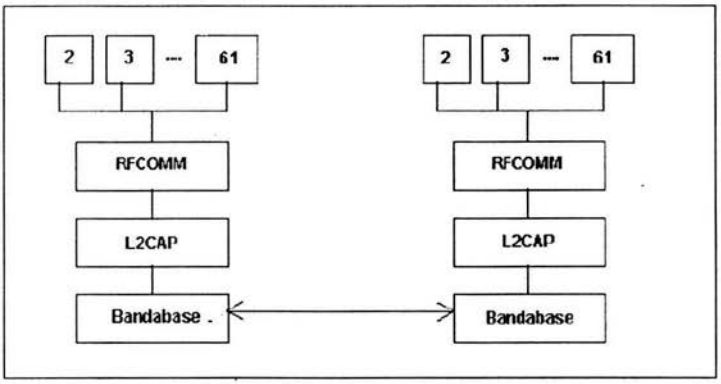

Figura 3-14. Varios puertos seriales emulados con RFCOMM.

3.8 Protocolo de Descubrimiento de Servicio (SDP, Service Discovery Protocol)

El **Protocolo de Descubrimiento de Servicio (SDP)** permite a las aplicaciones descubrir qué servicios están disponibles y determinar las características de esos servicios disponibles.

3.8.1 Configuración del Protocolo SDP

SDP es un simple protocolo con requisitos mínimos. Puede funcionar sobre un transporte de paquetes fiables. SDP usa un modelo demanda/respuesta donde cada transacción consiste en una demanda de la *Unidad de Datos del Protocolo* (PDU, Protocol Data Unit) y una respuesta PDU

En el caso donde SDP se usa con el protocolo de transporte L2CAP, un cliente debe recibir una contestación a cada demanda antes de emitir otra demanda en la misma conexión L2CAP. Esto proporciona una forma simple de control de flujo como se muestra en la figura 3-15.

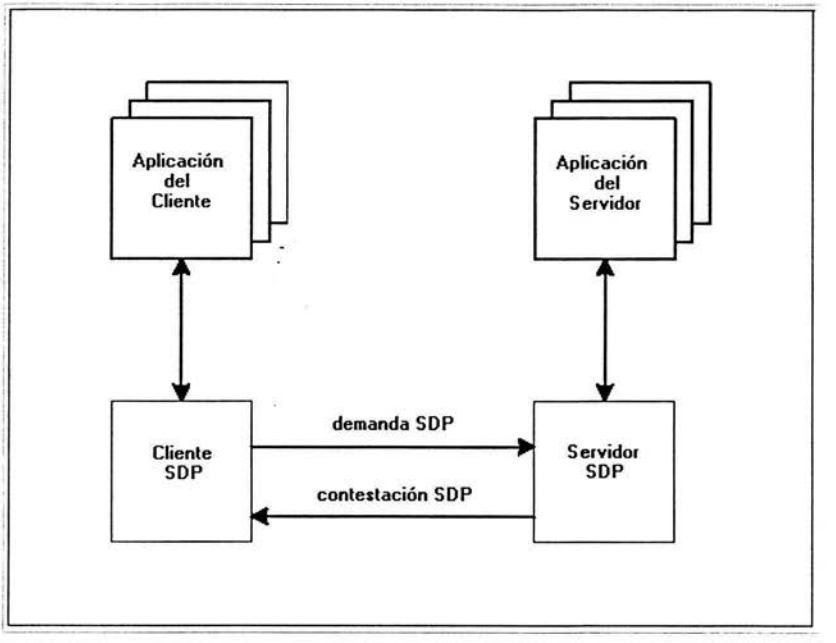

Figura 3-15. Esquema del Protocolo SDP.

Formato PDU

Cada **SDP PDU** consiste en una cabecera PDU seguida de parámetros específicos PDU. La cabecera contiene tres campos:

- El PDU ID: identifica el tipo de PDU.
- El ID de Transición: identifica los PDUs demandantes y se usa con sus correspondientes PDUs de respuesta.
- La longitud de parámetro: especifica la longitud (en bytes) de todos los parámetros contenidos en el PDU.

Contestaciones parciales y Estado de continuación

Algunas demandas de SDP pueden requerir contestaciones que son más grandes que un PDU; en este caso, el servidor de SDP generará una contestación parcial junto con un parámetro de estado de continuación.

Manejo de errores

Cada transacción consiste en una demanda y una contestación PDU; generalmente, cada tipo de demanda PDU tiene un tipo correspondiente de contestación PDU; sin embargo, si el servidor determina que una demanda se estructura inadecuadamente o por cualquier razón el servidor no puede responder con el tipo de PDU apropiado, responderá con un error PDU (SDP_ErrorResponse).

3.8.2 Servicios SDP

En la siguiente sección se describe como se guardan las características individuales de los diferentes dispositivos.

Registro de Servicio

Un servicio es una entidad que puede proveer información, realizar una acción, o controlar un recurso en nombre de otra entidad. Un servicio puede ser implementado como software, o una combinación de software y hardware. Toda la información acerca de un servicio que se mantiene en un servidor SDP está contenida en un simple registro de servicio. El Registro de Servicio consiste en una lista de atributos de servicio.

Atributo de Servicio

Cada atributo de servicio describe una sola característica de un servicio. Un atributo de servicio consiste en dos componentes: un ID de atributo y un valor de atributo.

- Un ID de atributo es un entero de 16 bits que distingue cada atributo de servicio de otros atributos de servicio con un registro de servicio. El ID también identifica la semántica del valor del atributo asociado.

- El valor del atributo es un campo de longitud variable cuyo significado es determinado por el atributo ID asociado con él y por la clase de servicio del registro de servicio en el que el atributo se encuentra. En el Protocolo de Descubrimiento de Servicio, un valor del atributo se representa como un elemento de datos.

Clase de Servicio

Cada servicio es un caso de una clase de servicio, la definición de clase de servicio proporciona las definiciones de todos los atributos contenidas en archivos de servicio que representan casos de esa clase; cada definición del atributo especifica el valor numérico del atributo ID, el uso intencional del valor del atributo y el formato del valor del atributo. Un registro de servicio contiene atributos que son específicos a una clase de servicio; así como, atributos universales que son comunes a todos los servicios.

A cada clase de servicio se asigna un único identificador que está contenido en el atributo ServiceClassIDList, se representa como un **Identificador Universal Único**

(UUID, Universally Unique Identifier). Un UUID es un único identificador universal que garantiza ser único por todo el espacio y todo el tiempo. Un UUID es un valor de **128 bits.**

3.8.3 Descubrimiento de Servicio

El propósito del SDP es permitir a los dispositivos **Bluetooth** descubrir qué otros dispositivos **Bluetooth** pueden ofrecer servicios.

Búsqueda de Servicios

La transacción de **Búsqueda de Servicio** le permite a un cliente recuperar el manejo del registro de servicio basado en los valores de los atributos contenidos dentro de aquéllos registros de servicio.

La capacidad de búsqueda se proporciona para buscar atributos cuyos valores son UUIDs. Los atributos importantes de servicio que se pueden usar para buscar un servicio se representan como UUIDs. El modelo de búsqueda de servicio se usa para localizar el servicio deseado, se usa un modelo de búsqueda de servicio en una lista de UUIDs (atributos de servicio).

Hojeo de Servicios

Este proceso se usa para buscar cualquier servicio ofrecido. En el SDP, el mecanismo para hojear los servicios está basado en un atributo compartido por todas las clases de servicio. Este atributo se llama el atributo de BrowseGroupList. El valor de este atributo contiene una lista de UUIDs, cada UUID representa un grupo del vistazo con el que un servicio puede asociarse con el propósito de hojear; cuando un cliente desea hojear los servicios de un servidor de SDP, crea un modelo de búsqueda de servicio que contiene el UUID que representa el grupo de vistazo raíz.

El Servicio Descubrimiento Protocolo **Bluetooth** se enfoca principalmente en descubrir servicios disponibles a través de los dispositivos **Bluetooth**. SDP no define métodos para acceder a los servicios; una vez que se descubren los servicios con SDP; estos pueden accederse de varias maneras y pueden depender del servicio. Esto podría incluir el uso de otro descubrimiento de servicio y mecanismos de acceso. En los ambientes de **Bluetooth**, pueden descubrirse servicios usando SDP y pueden accederse usando otros protocolos definidos por **Bluetooth**.

3.9 Perfiles

Los perfiles se han desarrollado para describir cómo las aplicaciones de modelos del usuario serán realizadas. Los modelos del usuario describen varios escenarios del usuario donde **Bluetooth** realiza la transmisión de radio. *Un perfil puede describirse como un segmento vertical a través de la pila de protocolos.* Define opciones en cada protocolo obligatorios para cada perfil. También define rangos de parámetros para cada protocolo. *El concepto del perfil se usa para disminuir el riesgo de problemas del interoperabilidad entre los productos de fabricantes diferentes.*

La figura 3-16 muestra la estructura de los perfiles **Bluetooth**:

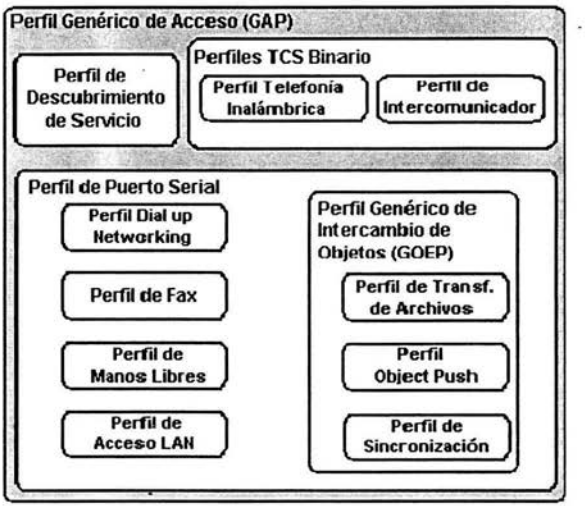

Figura 3-16. Esquema de los perfiles Bluetooth.

Un perfil es dependiente de otro perfil si re-usa partes de ese perfil, haciendo referencia a él implícitamente o explícitamente. La dependencia se ilustra en la figura 3-16: un perfil tiene dependencias en los perfiles en que se contiene directamente e indirectamente, por ejemplo; el perfil de Object Push es dependiente del Perfil Genérico de Intercambio de Objetos, Puerto Serial, y los perfiles de Acceso Genérico.

En la tabla 3-2 se muestran los perfiles definidos en la especificación **Bluetooth 1.1**:

	Código	Versión	Nombre del Perfil
K1	GAP	1.1	Perfil Genérico de Acceso (Generic Access Profile)
K2	SDAP	1.1	Perfil de Aplicación de Descubrimiento de Servicio (Service Discovery Application Profile)
K3	CTP	1.1	Perfil de Telefonía Inalámbrica (Cordless Telephony Profile)
K4	IP	1.1	Perfil de Intercomunicación (Intercom Profile)
K5	SPP	1.1	Perfil de Puerto Serial (Serial Port Profile)
K6	HS	1.1	Perfil de Audífono (Headset Profile)
K7	DNP	1.1	Perfil de Dial-up Networking (Dial-up Networking Profile)
K8	FP	1.1	Perfil de Fax (Fax Profile)
K9	LAP	1.1	Perfil de Acceso LAN (LAN Access Profile)
K10	GOEP	1.1	Perfil de Intercambio de Objetos Genéricos (Generic Object Exchange Profile)
K11	OPP	1.1	Perfil de Empuje de Objeto (Object Push Profile)
K12	FTP	1.1	Perfil de Trasferencia de Archivos (File Transfer Profile)
K13	SP	1.1	Perfil de Sincronización (Synchronization Profile)

Tabla 3-2. Perfiles de la especificación Bluetooth 1.1.

En la tabla 3-3 se muestran los nuevos perfiles que se encuentran en las fases finales de desarrollo:

ESDP	0.95a	Perfil de Descubrimiento de Servicio Extendido (Extended Service Discovery Profile for Universal Plug and Play)
A2DP	0.95b	Perfil de Distribución de Audio Avanzado (Advanced Audio Distribution Profile)
AVRCP	0.95b	Control de Audio Video Remoto (Audio Video Remote Control Profile)
BIP	1.0	Perfil de Imágenes Básicas (Basic Imaging Profile)
BPP	0.95a	Perfil de Impresión Básico (Basic Printing Profile)
CIP	1.0	Perfil Acceso Común ISDN (Common ISDN Access Profile)
GAVDP	0.95b	Perfil de Distribución Genérico Audio Video (Generic Audio Video Distribution Profile)
HFR	0.96	Perfil de Manos Libres (Hands-Free Profile)
HCRP	0.95a	Perfil de Reemplazo de Cable (Hardcopy Cable Replacement Profile)
HID	0.95c	Perfil de Dispositivo de Interfase Humana (Human Interface Device Profile)
PA N	0.95a	Perfil PAN (PAN Profile)
SAP	0.95c	Perfil de Acceso SIM (SIM Access Profile)

Tabla 3-3. Nuevos perfiles para la especificación Bluetooth.

3.9.1 Perfil Genérico de Acceso (GAP).

Este perfil define los procedimientos generales para el descubrimiento y establecimiento de conexión entre dispositivos **Bluetooth**, el *GAP* maneja el descubrimiento y establecimiento entre unidades que no están conectadas, asegura que cualquier par de unidades **Bluetooth**, sin importar su fabricante o aplicación, puedan intercambiar información a través de **Bluetooth** para descubrir qué tipo de aplicaciones soportan las unidades.

3.9.2 Perfil de Puerto Serial.

Este perfil define los requerimientos para dispositivos **Bluetooth**, necesarios para establecer una conexión de cable serial emulada usando RFCOMM entre dos dispositivos similares. Este perfil solamente requiere soporte para paquetes de una ranura. Esto significa que pueden ser usadas tasas de datos de hasta **128 kbps**. El soporte para tasas más altas es opcional. RFCOMM es usado para transportar los datos de usuario, señales de control de *modem* y comandos de configuración, el *perfil de puerto serial* es dependiente del *GAP*.

3.9.3 Perfil de Aplicación de Descubrimiento de Servicio (SDAP).

Este perfil define los protocolos y procedimientos para una aplicación en un dispositivo **Bluetooth** donde se desea descubrir y recuperar información relacionada con servicios localizados en otros dispositivos, el *SDAP* es dependiente del *GAP*.

3.9.4 Perfil Genérico de Intercambio de Objetos (GOEP).

Este perfil define protocolos y procedimientos usados por aplicaciones para ofrecer características de intercambio de objetos. Los usos pueden ser, por ejemplo, sincronización, transferencia de archivos o modelo **Object Push**. Los dispositivos más comunes que usan este modelo son agendas electrónicas, *PDAs*, teléfonos celulares y teléfonos móviles, el *GOEP* es dependiente del *perfil de puerto serial.*

3.9.5 Perfil de Telefonía Inalámbrica.

Este perfil define cómo un teléfono móvil puede ser usado para acceder a un servicio de telefonía de red fija a través de una estación base. Es usado para telefonía inalámbrica de hogares u oficinas pequeñas. El perfil incluye llamadas a través de una estación base, haciendo llamadas de intercomunicación directa entre dos terminales y accediendo adicionalmente a redes externas. Es usado por dispositivos que implementan el llamado "teléfono 3-en-1".

3.9.6 Perfil de Intercomunicador.

Este perfil define usos de teléfonos móviles los cuales establecen enlaces de conversación directa entre dos dispositivos. El enlace directo es establecido usando señalización de telefonía sobre **Bluetooth**. Los teléfonos móviles que usan enlaces directos funcionan como **walkie-talkies.**

3.9.7 Perfil de Manos Libres.

Este perfil define los requerimientos, para dispositivos **Bluetooth**, necesarios para soportar el uso de manos libres. En este caso el dispositivo puede ser usado como unidad de audio inalámbrico de entrada/salida. El perfil soporta comunicación segura y no segura.

3.9.8 Perfil Dial-up Networking.

Este perfil define los protocolos y procedimientos que deben ser usados por dispositivos que implementen el uso del modelo llamado *Puente Internet*, es aplicado cuando un teléfono celular o *módem* es usado como un *módem* inalámbrico.

3.9.9 Perfil de Fax.

Este perfil define los protocolos y procedimientos que deben ser usados por dispositivos que implementen el uso de fax. En el perfil un teléfono celular puede ser usado como un fax inalámbrico.

3.9.10 Perfil de Acceso LAN.

Este perfil define el acceso a una *red de área loca, LAN,* usando el protocolo punto-a-punto, *PPP*, sobre *RFCOMM. PPP* es ampliamente usado para lograr acceder a redes soportando varios protocolos de red. El perfil soporta acceso *LAN* para un dispositivo **Bluetooth** sencillo, acceso LAN para varios dispositivos **Bluetooth** y PC-a-PC (usando interconexión *PPP* con emulación de cable serial).

3.9.11 Perfil Object Push.

Este perfil define protocolos y procedimientos usados en el modelo **object push**. Este perfil usa el *GOEP*. En el modelo **object push** hay procedimientos para introducir en el **inbox**, sacar e intercambiar objetos con otro dispositivo **Bluetooth**.

3.9.12 Perfil de Transferencia de Archivos.

Este perfil define protocolos y procedimientos usados en el modelo de transferencia de archivos. El perfil usa el *GOEP*. En el modelo de transferencia de archivos hay procedimientos para chequear un grupo de objetos de otro dispositivo **Bluetooth**, transferir objetos entre dos dispositivos y manipular objetos de otro dispositivo, los objetos podrían ser archivos o carpetas de un grupo de objetos tal como un sistema de archivos.

3.9.13 Perfil de Sincronización.

Este perfil define protocolos y procedimientos usados en el modelo de sincronización. Este usa el *GOEP*. El modelo soporta intercambios de información, por ejemplo para sincronizar calendarios de diferentes dispositivos.

3.10 Conclusiones del Capítulo.

En este capítulo se estudió la especificación **Bluetooth 1.1**, con lo que podemos resumir:

- La capa de comunicación más baja es el **Radio Bluetooth**; el cual implementa el canal físico real, emplea una secuencia aleatoria de saltos a través de **79** frecuencias de radio diferentes, los paquetes son enviados sobre el canal físico, donde cada uno es enviado en una frecuencia de salto diferente. La *Banda Base* controla la sincronización de las unidades **Bluetooth** y la secuencia de saltos en frecuencia, además es la responsable de la información para el control de enlace a bajo nivel como el reconocimiento, control de flujo y caracterización de carga útil, soporta dos tipos de enlace: *síncrono orientado a la conexión* (SCO), para datos y *asíncrono no orientado a la conexión* (ACL), principalmente para audio.

- El **Protocolo del Administrador de Enlace** (LMP) es el responsable de la autenticación, encriptación, control y configuración del enlace, el *LMP* también se encarga del manejo de los modos y consumo de potencia, además soporta los procedimientos necesarios para establecer un enlace *SCO*.

- El **Interfaz del Controlador Host** (HCI) brinda un método de interfaz uniforme para acceder a los recursos de hardware de **Bluetooth**, éste contiene una interfaz de comandos para el controlador *Banda Base* y el *Administrador de Enlace* para acceder al hardware.

- El **Protocolo Adaptación y Control de Enlace Lógico** (L2CAP) brinda servicios de datos orientados y no orientados a la conexión a capas superiores. *L2CAP* multiplexa los protocolos de capas superiores con el fin de enviar varios protocolos sobre un canal Banda Base, con el fin de manipular paquetes de capas superiores más grandes que el máximo tamaño del paquete *Banda Base*, *L2CAP* los segmenta en varios paquetes Banda Base y los re-ensambla para la capa superior. La conexión L2CAP permite el intercambio de información referente a la calidad de la conexión, además, maneja grupos, de tal manera que varios dispositivos pueden comunicarse entre sí.

- El **Protocolo de Descubrimiento de Servicio** (SDP) define cómo actúa una aplicación de un cliente **Bluetooth** para descubrir servicios disponibles de servidores **Bluetooth**, además de proporcionar un método para determinar las características de dichos servicios.

- El **protocolo RFCOMM** ofrece emulación de puertos seriales sobre el protocolo *L2CAP*. *RFCOMM* emula señales de control y datos *RS-232* sobre la Banda Base **Bluetooth**.

Por último se estudiaron los perfiles **Bluetooth** los cuales como se mencionó; describen varios escenarios del usuario donde **Bluetooth** realiza la transmisión de radio y se usa para disminuir el riesgo de problemas del interoperabilidad entre los productos de diferentes fabricantes.

Con estos elementos se puede tener una visión general de cómo funciona esta tecnología de manera teórica y proseguir cin el diseño de un dispositivo utilizando estos conceptos.

CAPÍTULO 4.
ANÁLISIS DE LA COMUNICACIÓN DE DATOS POR MEDIO DE USB Y UART.

En este capítulo se estudiarán los principios de operación de la comunicación de datos por medio de USB y UART.

Como se había mencionado; el HCI proporciona una interfaz de comandos al controlador *Banda Base* y al *Administrador de Enlace*, además de acceso al hardware y a los registros de control, por lo cual se necesita un bus físico entre el host **Bluetooth** y el hardware **Bluetooth**, como se puede apreciar en la figura 4-1.

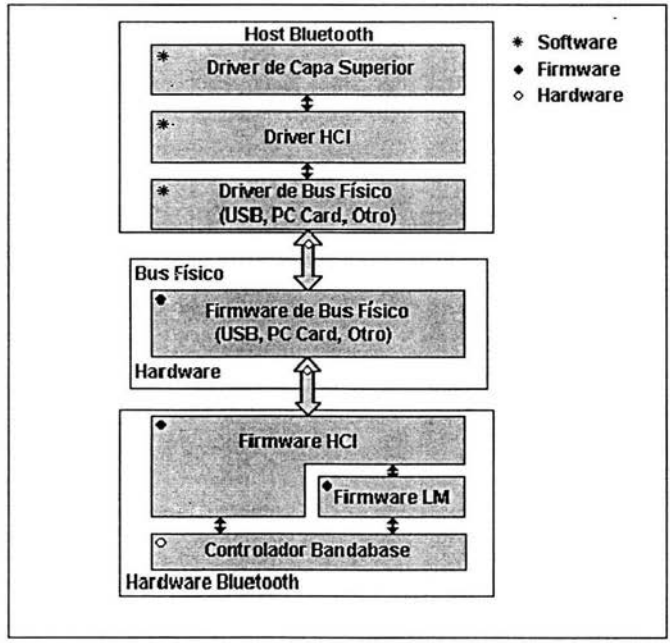

Figura 4-1. Diagrama general de las capas más bajas.

Los dispositivos **Bluetooth** tienen varias interfaces de bus físicas que pueden ser usadas para conectar el hardware. Estos buses pueden tener diferentes arquitecturas y parámetros.

El *controlador de Host* soporta tres arquitecturas de bus físico, *USB, UART y PC Card.* Todas ellas pueden manejar varios canales lógicos sobre el mismo canal físico simple (a través de **puntos finales**), por lo tanto los canales de control, datos y voz no requieren alguna interfaz física adicional.

4.1 Comunicación de datos por medio de USB (Universal Serial Bus)

Como resultado de un intento de dotar al PC de un bus de alta velocidad que ofreciera las características ideales PnP (Plug and Play) de universalidad, facilidad de conexión y desconexión, incluso en caliente ("Hot Swappable"), y sobre todo, que consumiera pocos recursos, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC y Phillips diseñaron el denominado puerto USB, que como su nombre indica, es un bus serie, bidireccional y de bajo costo para periféricos como teclados, ratones, joysticks, impresoras, scanners, dispositivos de almacenamiento, módems, y cámaras de vídeo conferencia.

4.1.1 Estándares

La tecnología USB se ha convertido en un estándar importante. En sus comienzos los interesados en esta tecnología se agruparon en un foro, el **USB-IF** (USB Implementers Forum Inc.) que agrupa a más de 460 compañías, y han publicado diversas revisiones de la norma:

- **USB 0.9**: Primer borrador, publicado en Noviembre de 1995.

- **USB 1.0**: Publicada en 1996 establece dos tipos de conexión: La primera, denominada *velocidad baja* ("Low speed"), ofrece **1.5 Mbps**, y está pensada para periféricos que no requieren un gran ancho de banda, como ratones o joysticks. La segunda, denominada *velocidad completa* ("Full speed"), es de **12 Mbps**, y está destinada a los dispositivos más rápidos.

- **USB 1.1**: Publicada en 1998, añade detalles y precisiones a la norma inicial; es el estándar mínimo que debe cumplir un dispositivo USB.

- **USB 2.0**: Su versión final fue publicada en Abril del 2000; es una extensión de la norma compatible con las anteriores. Permite velocidades de hasta **480 Mbps**, denominada *alta velocidad* ("High speed")

4.1.2 Topología del bus

El bus USB soporta el intercambio simultáneo de datos entre una computadora "host" y un amplio conjunto de periféricos. Todos los periféricos conectados comparten el ancho de banda del bus por medio de un protocolo basado en Testigos ("Tokens"), es decir, mensajes ajustados a un formato, el bus permite conexión y desconexión dinámica, es decir, que los periféricos se conecten, configuren, manipulen y desconecten mientras el sistema host y otros periféricos permanecen en funcionamiento.

La topología del bus USB adopta forma de estrella y se organiza por niveles, en un bus USB existen dos tipos de elementos: *concentradores* y *dispositivos*; a su vez, los dispositivos pueden ser de dos tipos: *concentradores* y *funciones*.

- Los *concentradores* ("Hubs") son el centro de una estrella, y sirven para conectar con el sistema anfitrión, con otro *concentrador* o con una *función*, cada concentrador puede proporcionar **500 mA** de energía de alimentación **(hasta 2.5W)** a cada uno de los dispositivos a él conectados, ya que el cable de conexión tiene hilos de señal (datos) y de alimentación **(5 V. cc ± 0.25 V)**.

- Una *función* es un dispositivo capaz de transmitir o recibir datos o información de control en un bus USB, suele conectarse como un dispositivo independiente enlazado por un cable de menos de 5 metros, a un puerto del *concentrador* o directamente al sistema anfitrión.

Un *concentrador* puede estar conectado a otro *concentrador*; esto significa que pueden conectarse dispositivos en cascada; el sistema soporta un total de **127 dispositivos**. Una característica importante es que el *host* o el *concentrador* proporcionen la energía necesaria a la función por el cable de conexión, lo que evita la necesidad de fuentes de alimentación independientes.

4.1.3 Funcionamiento

El bus serie USB es síncrono, y utiliza el algoritmo de codificación NRZI ("Non Return to Zero Inverted"). En este sistema existen dos voltajes opuestos; una tensión de referencia corresponde a un "1", pero no hay retorno a cero entre bits, de forma que una serie de unos corresponde a un voltaje uniforme; en cambio los ceros se marcan como cambios del nivel de tensión, de modo que una sucesión de ceros produce sucesivos cambios de tensión entre los conductores de señal.

El controlador USB instalado en la computadora, denominado *controlador de host*, o *concentrador raíz* (Root hub), proporcionan un enlace entre el bus de la placa-base y una o más conexiones iniciales con el exterior.

El protocolo de comunicación utilizado es de testigo, guarda cierta similitud con el sistema Token-Ring de IBM. Puesto que todos los periféricos comparten el bus y pueden funcionar de forma simultanea, la información es enviada en paquetes; cada paquete contiene una cabecera que indica el periférico a que va dirigido, existen cuatro tipos de paquetes distintos: Token (8 bytes); Datos (16 Bytes); Handshake (32 bytes) y Especial (64 Bytes). Se utiliza un sistema de detección y corrección de errores tipo CRC ("Cyclical Redundancy Check").

El funcionamiento está centrado en el host, todas las transacciones se originan en él; es el controlador host el que decide todas las acciones, incluyendo el número asignado a cada dispositivo (esta asignación es realizada automáticamente por el controlador "host" cada vez que se inicia el sistema o se añade, o elimina, un nuevo dispositivo en el bus), su ancho de banda, etc. Cuando se detecta un nuevo dispositivo es el host el encargado de cargar los drivers oportunos sin necesidad de intervención por el usuario.

El sistema utiliza cuatro tipos de transacciones que resuelven todas las posibles situaciones de comunicación; cada transacción utiliza un mínimo de tres paquetes, el primero es siempre un **Token**, que avisa al dispositivo que puede iniciar la transmisión.

- **Transferencia de control** ("Control transfer"): Ocurre cuando un dispositivo se conecta por primera vez; en este momento el controlador de host envía un paquete "Token" al periférico notificando el número que le ha asignado.

- **Transferencia de pila de datos** ("Bulk data transfer"): Éste proceso se utiliza para enviar gran cantidad de datos de una sola vez. Es útil para dispositivos que tienen que enviar gran cantidad de datos cada vez, como scanneres o cámaras digitales.

- **Transferencia por interrupción** ("Interrupt data transfer"): Éste proceso se utiliza cuando se solicita enviar información por el bus en una sola dirección (de la función al host).

- **Transferencia de datos isíncrona** ("Isochronous data transfer"): Éste proceso se utiliza cuando es necesario enviar datos en tiempo real. Los datos son enviados con una cadencia precisa ajustada a un reloj, de modo que la transmisión es a velocidad constante.

Las comunicaciones asíncronas ponen más énfasis en garantizar el envío de datos y menos en su temporización; por su parte las comunicaciones isíncronas son justamente lo contrario, ponen más énfasis en la oportunidad de la transmisión que en la velocidad, esta sincronización es importante en situaciones como la reproducción de video, donde no debe existir desfase entre las señales de video y audio.

4.1.4 Cables y conectores

El cable de bus USB es de 4 hilos como se muestra en la tabla 4-1; y comprende líneas de señal (datos) y alimentación, con lo que las funciones pueden utilizar un único cable.

Pin	Nombre	Descripción	Color
1	VBUS	+5 V. CC	Rojo
2	D-	Datos -	Azul
3	D+	Datos +	Amarillo
4	GND	Tierra	Verde

Tabla 4-1. Líneas de señal y alimentación USB.

Existen dos tipos de cable: *blindado* y *sin blindar*, en el primer caso, el par de hilos de señal es trenzado, los de tierra y alimentación son rectos, y la cubierta de protección (blindaje) solo puede conectarse a tierra en el host. En el cable sin blindar todos los hilos son rectos, las conexiones a 15 Mbps y superiores exigen cable blindado, se utilizan diámetros estándar para los hilos de alimentación del bus, para cada sección se autoriza una longitud máxima del segmento, como se muestra en la tabla 4-2.

AWG	mm ∅	Longitud máxima
28	0.321	0.81 m
26	0.405	1.31 m
24	0.511	2.08 m
22	0.644	3.33 m
20	0.812	5.00 m

Tabla 4-2. Diámetros y longitudes de los cables USB.

Se usan dos tipos de conectores, **A** y **B**. Ambos son polarizados (solo pueden insertarse en una posición) y utilizan sistemas de presión para sujetarse:

- Los de tipo **A** utilizan la hembra en el sistema host, y suelen usarse en dispositivos en los que la conexión es permanente (por ejemplo, ratones y teclados). La figura 4-2 muestra un conector USB **A**:

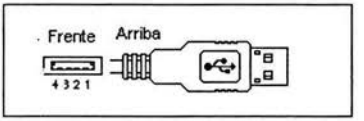

Figura 4-2. Conector USB A.

- Los de tipo **B** utilizan la hembra en el dispositivo USB (*función*), y se utilizan en sistemas móviles (por ejemplo, cámaras digitales o bocinas). La figura 4-3 muestra un conector USB **B**:

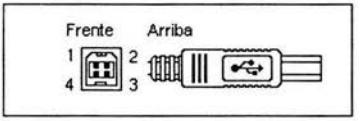

Figura 4-3. Conector USB B.

En general podemos afirmar que los conectores tipo **A** están en el lado del host (PC) o de los *concentradores*, mientras los de tipo **B** están del lado de los periféricos.

4.2 Comunicación de datos por medio del UART

El corazón del sistema de comunicaciones serie es la **UART**, acrónimo de **Universal Asynchronous Receiver-Transmitter**. Es un chip cuya misión principal es convertir los datos recibidos del bus de la PC en formato paralelo, a un formato serie que será utilizado en la transmisión hacia el exterior. También realiza el proceso contrario: transformar los datos serie recibidos del exterior en un formato paralelo entendible por el bus.

La **UART** es un dispositivo programable en el que pueden establecerse las condiciones que se utilizarán para la transmisión (velocidad, paridad, longitud y bits de parada).

Las transmisiones en "serie" transmiten los bits uno por uno; por ejemplo, para transmitir un byte, transmite los 8 bits uno a la vez; esto tiene la ventaja de que la comunicación necesita solamente un hilo para transmitir los 8 bits (mientras la comunicación paralela necesitaría 8 hilos); la desventaja es que el tiempo de transmisión es 8 veces mayor que si se tuvieran los 8 hilos.

Antes de cada byte de dato, el puerto serial manda un bit de inicio, que es un simple bit con valor de "0", después de un byte de datos, manda un bit de alto y en algunos casos también manda un bit de paridad, la comunicación serial es bi-direccional, y puede ser half-dúplex o full-dúplex.

Con el fin de alcanzar una mayor velocidad la mayoria de los UARTs tienen búffers de **16 a 64 kilobytes**, este búffer permite al chip almacenar datos de entrada del bus del sistema mientras procesa datos salientes, la mayoría de los puertos seriales tienen una tasa de transferencia de **115 kbps**, aunque existen otros puertos seriales que pueden alcanzar ahata **460 kbps**.

4.2.1 La conexión serial.

La conexión serial puede ser de 9 pines o 25 pines:

DB9:

1	DCD (Data Carrier Detect)
2	RX
3	TX
4	DTR (Data Terminal Ready)
5	GND
6	DSR (Data Sheet Ready)
7	RTS (Request To Send)
8	CTS (Clear To Send)
9	RI (Ring Indicator)

DB25:

1	No se usa
2	TX
3	RX
4	RTS
5	CTS
6	DSR
7	GND
8	RLSD (received Line Signal Detector)
9	No se usa
10	No se usa
11	No se usa
12	No se usa
13	No se usa
14	No se usa
15	No se usa
16	No se usa
17	No se usa
18	No se usa
19	No se usa
20	DTR
21	No se usa
22	RI
23	No se usa
24	No se usa
25	No se usa

El voltaje enviado por los pines pueden ser los estados: *On* y *Off*. On "1" significa que el pin está transmitiendo una señal entre –3 y –25 volts. Off "0" significa que está transmitiendo entre +3 y +25 volts.

4.3 Conclusiones del Capítulo

Como se analizó en este capítulo, se puede utilizar la comunicación por medio de **USB** o **UART** entre el host **Bluetooth** y el hardware **Bluetooth**, dependiendo de las capacidades del hardware.

Se puede encontrar ventaja en la comunicación por USB; la velocidad de transmisión es mayor, se puede alimentar a los dispositivos; ya que, cuenta con una línea de alimentación de **5V**, por lo cual se podría eliminar el uso de baterías o fuentes externas al dispositivo.

Otra ventaja de la comunicación por USB es que para interfaz UART se necesita un transceptor RS-232 que adapte los diferentes niveles lógicos que utilizan el puerto serial de la PC y la interfaz UART del chip; ya que la PC trabaja con niveles de voltaje de **–12V** para representar un **1** lógico y **+12V** para representar un **0** lógico, mientras que la interfaz UART trabaja con niveles de voltaje de **0V** para representar un **0** lógico y **+3.3V** para representar un **1** lógico.

CAPÍTULO 5.
DISEÑO E IMPLEMENTACIÓN DE DISPOSITIVOS USANDO TECNOLOGÍA BLUETOOTH.

Después de estudiar y revisar la tecnología **Bluetooth** se realizó un estudio de las alternativas para implementar un dispositivo que utilice esta tecnología con el fin de comprobar de forma práctica su funcionamiento.

Para hacer un análisis en cuanto a ventajas, desventajas, costos y accesibilidad, se tomó en cuenta la lista de productos que se presentan en la Tabla 2-5, así como los productos de desarrollo disponibles en el mercado.

Para poder implementar un dispositivo con esta tecnología se necesita un host, el cual puede ser cualquier sistema programable (PCs, teléfonos celulares, mouse, impresoras, teclados, etc.) capaz de ejecutar las líneas de código correspondientes a la pila de protocolos para el host, este punto es el más relevante a la hora de decidir que tipo de dispositivo implementar; ya que, esto implica que se necesita de un sistema programable y de un software para programarlo.

En el caso de optar por un periférico como es un mouse, un teclado, una impresora o un dispositivo de manos libres, se necesitaría un microcontrolador adicional al módulo **Bluetooth** que mande y ejecute las instrucciones; el software para la pila de protocolos **Bluetooth** adecuado, además de que se necesitaría del equipo necesario para integrarlo en un espacio muy pequeño.

Por estas razones se determinó diseñar un dispositivo donde el host sea una PC; como lo es un adaptador **Bluetooth** o "dongle **Bluetooth**" como se conoce en el mercado, ya que es posible tener acceso a la pila de protocolos **Bluetooth** por medio de algún software. Otro factor importante es que no se requiere de un equipo sofisticado para su implementación y el mismo software nos puede proporcionar datos de mediciones a través de algunas capas del protocolo.

5.1 Diseño de un adaptador (dongle) Bluetooth.

Una vez que se determinó el tipo de dispositivo a diseñar y con los conocimientos adquiridos acerca de esta tecnología, se procedió a investigar los elementos de hardware necesarios para implementar este dispositivo, básicamente se necesitan tres elementos: *el módulo **Bluetooth**; la parte de regulación de voltaje para la alimentación del mismo y la parte de la interfaz con el host.*

El elemento principal es el módulo **Bluetooth**, porque de él depende la regulación de voltaje y la interfaz con el host.

Después de hacer una investigación de los sitios donde se ofrecen soluciones de desarrollo de **Bluetooth**, se encontró el sitio de "**Bluetooth designer**" (Inglaterra) http://www.btdesigner.com/, donde se pueden adquirir productos a menudeo y los precios son relativamente accesibles.

En este sitio se pueden encontrar módulos clase 1 y clase 2 de varios fabricantes como: AIR Logic, TAIYO YUDEN, SIEMENS entre otros. Además se pueden encontrar también kits de desarrollo como-"Casira", "Bluelab" e "Interface Express".

Por el tipo de dispositivo a implementar; se decidió que fuera de **clase 2** (máximo **4dBm**, hasta **10mts**) y con interfaz USB por lo concluido en el capítulo 4.

Se eligió el módulo de la marca **TAIYO YUDEN** modelo **"EYMF2CSMM"** (RF + Banda Base Clase 2 USB), ya que a diferencia de otros módulos tiene la antena integrada la cual reduce la complejidad en el diseño y es compatible totalmente con el estándar **Bluetooth versión 1.1**, además de que físicamente es apto para montarse en un circuito impreso con una punta de cautín adecuada.

5.1.1 El módulo Bluetooth EYMF2CSMM

El módulo **EYMF2CSMM**, es un módulo transmisor de radiofrecuencia (clase 2), que satisface la especificación **Bluetooth 1.1**, con aplicaciones para PC portátiles y PDAs.

Tiene interfaz USB (compatible con la versión 1.1 de USB), soporta enlaces Punto-a-Multipunto (7 esclavos) y soporta encriptación; modos de bajo consumo de energía: *Sostenido, Olfateo* y *Estacionado*; y soporta enlaces ACL.

La Figura 5-1 muestra la imagen de dicho módulo y en la Figura 5-2 se muestra un diagrama de bloques con las partes que lo conforman.

Figura 5-1. Módulo TAIYO YUDEN "EYMF2CSMM".

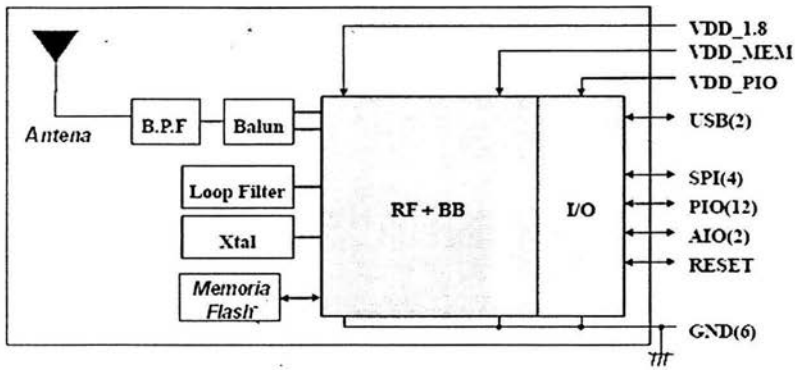

Figura 5-2. Diagrama de bloques del módulo.

La Tabla 5-1 muestra los rangos de operación recomendados para este módulo.

	Símbolo	Rango			
		Mín.	Típico	Máx.	Unidad
Fuente de voltaje 1	VDD_MEM	3.1	3.3	3.45	V
Fuente de voltaje 2	VDD_PIO	3.1	3.3	3.45	V
Fuente de voltaje 3	VDD_1.8	1.75	1.8	1.9	V
Rango de temperatura de operación	Topr	0	25	70	Grados C

Tabla 5-1. Rangos de operación recomendados.

La Tabla 5-2 muestra las características de radiofrecuencia de este módulo con los siguiente parámetros: **Ta=25 grados C, VDD_MEM=VDD_PIO=3.3V, VDD_1.8=1.8V**.

Parámetro	Símbolo	Mínimo	Típico	Máximo	Unidad
Rango de frecuencia	FREQ	2400		2483.5	MHz
Poder de transmisión	PO	-6	0	+4	dBm
Sensibilidad de recepción	SEN		-80	-70	dBm

Tabla 5-2. Especificaciones de radiofrecuencia.

El módulo cuenta con firmware versión 16.4 y soporta todos los comandos y eventos HCI de la especificación **Bluetooth 1.1**. En la Figura 5-3 se muestra la pila de protocolos del módulo.

Figura 5-3. Pila de protocolos del módulo.

En la Figura 5-4 se muestra el plano de tierra del módulo vista en la parte superior.

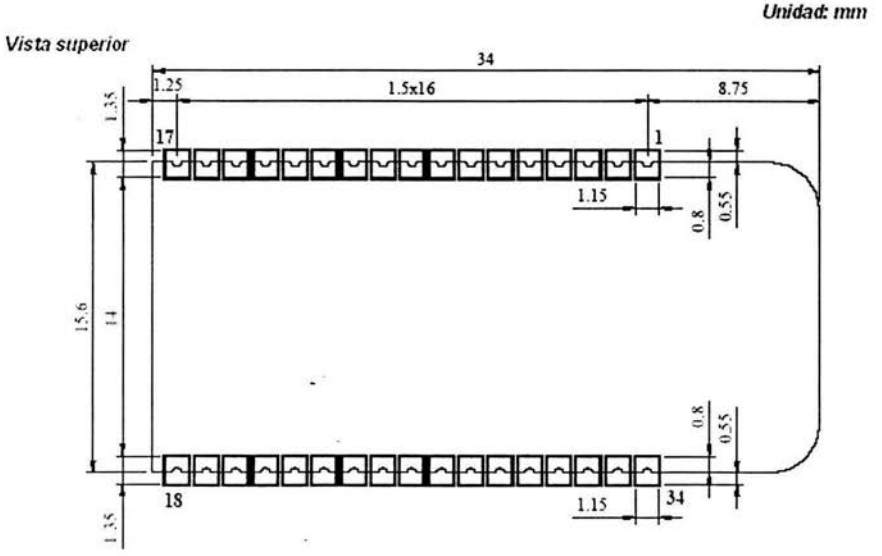

Figura 5-4. Plano de tierra del módulo.

En la Tabla 5-3 se muestra la descripción de los pines del módulo.

Número	Nombre	I/O	Descripción	Bloque
1	GND	-	Tierra	Power
2	AIO(0)	I/O	AIO son líneas programables según las necesidades del cliente	GPIO
3	AIO(1)	I/O		
4	RESET	I	RESET activo en alto con una resistencia de 51kΩ.	RESET
5	SPI_MISO	O	No conectar	SPI
6	SPI_CBS	I	No conectar	SPI
7	SPI_CNK	I	No conectar	SPI
8	SPI_MOSI	I	No conectar	SPI
9	GND	-	Tierra	Power
10	UART_RTS	O	Demanda de envío UART (No conectar)	
11	UART_CTS	I	Limpio para mandar UART (No conectar)	
12	UART_TX	O	UART TX (No conectar)	
13	UART_RX	I	UART RX (No conectar)	
14	VDD_1.8	I	Alimentación de DC de 1.8V	Power
15	VDD_MEM	I	Alimentación de DC de 3.3V	Power
16	VDD_PIO	I	Alimentación de DC de 3.3V	Power
17	GND	-	Tierra	Power
18	GND	-	Tierra	Power
19	USB_DP	I/O	USB D+	USB
20	USB:DN	I/O	USB D-	USB
21	PIO(11)	I/O	PIO son líneas programables según las necesidades del cliente	GPIO
22	PIO(10)	I/O		
23	PIO(9)	I/O		
24	PIO(8)	I/O		
25	PIO(7)	I/O		
26	PIO(6)	I/O		
27	GND	-	Tierra	Power
28	PIO(5)	I/O	PIO son líneas programables según las necesidades del cliente	GPIO
29	PIO(4)	I/O		
30	PIO(3)	I/O		
31	PIO(2)	I/O		
32	PIO(1)	I/O		
33	PIO(0)	I/O		
34	GND	-	Tierra	Power

Tabla 5-3. Descripción de pines del módulo.

Debido a que este módulo es para soldar en superficie, y que no se encontró alguna base para montar en una placa perforada, se diseño una base en una PBC (Printed Board Circuit), para poder ser montado en una protoboard por medio de cables conectados a cada uno de los pines, con el fin de poder cambiar la configuración.

En la Figura 5-5 se muestra el plano de tierra de la base para el módulo; el diseño fue hecho con el software de diseño **Corel Draw**, siguiendo el plano de tierra del módulo, para después poder ser impreso en una tableta para elaborar la PBC por medio de serigrafía.

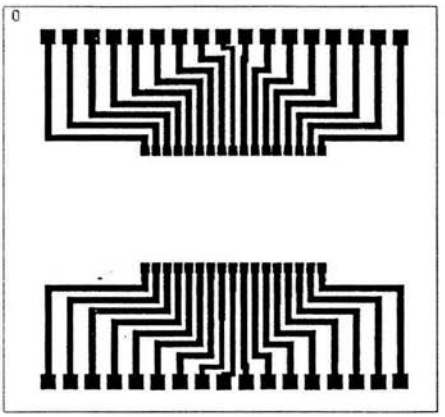

Figura 5-5. Plano de tierra de la base para el módulo.

En la figura 5-6 se muestra el módulo montado en su base, después de haber sido revelada y perforada para poder soldar los cables que van conectados a la protoboard.

Figura 5-6. PBC de la base del módulo.

5.2 Implementación del adaptador Bluetooth.

Después de haber seleccionado el módulo **Bluetooth** y de haber elaborado una base para poder utilizarlo en una protoboard; se procedió a elegir la parte de regulación de voltaje para la alimentación del mismo; y la parte de la interfaz con el host.

5.2.1 Regulación del voltaje para la alimentación del módulo.

La alimentación del voltaje para el módulo se puede elaborar de tres maneras: con una fuente externa de voltaje; por medio del puerto USB que nos proporciona 5V, o por medio de una batería.

Se determinó utilizar una batería de **9V** (el valor exacto es **9.3V**), por dos razones: con la finalidad de darle dependencia de una fuente externa, ya que tendría que estar sujeta a la alimentación de corriente alterna; y de poder disponer de una mayor demanda de corriente sin el riesgo de dañar el puerto USB.

Debido a que se requieren voltajes muy específicos (Tabla 5-1), se necesita algún regulador fijo o variable de voltaje, se encontraron varios reguladores de voltaje para **3.3V** y **1.8V**; sin embargo ninguno se puede conseguir en el mercado local; por esta razón se optó por utilizar divisores de voltaje, utilizando la fórmula:

$$\mathbf{VO = VI * [\ R2 / (R1 + R2)\ ]}$$

Donde:

VO = Voltaje de salida,
VI= Voltaje de entrada (9.3V),
R1= Resistencia 1,
R2= Resistencia 2.

En las Figuras 5-7 y 5-8 se muestran los divisores de voltaje para 3.3V y 1.8V respectivamente simulados en Electronics Workbench.

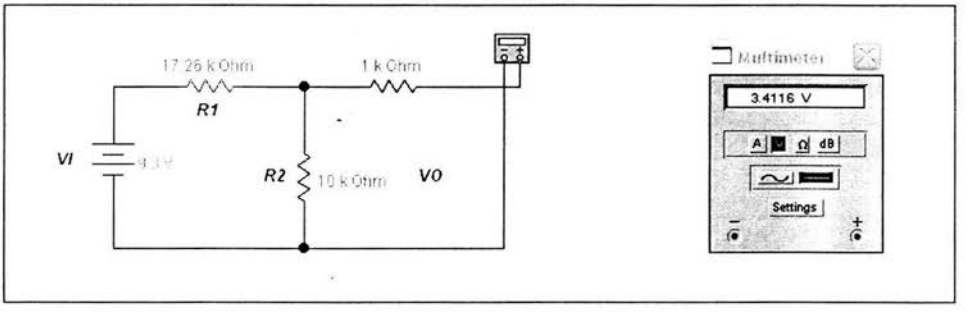

Figura 5-7. Divisor de voltaje para 3.3V.

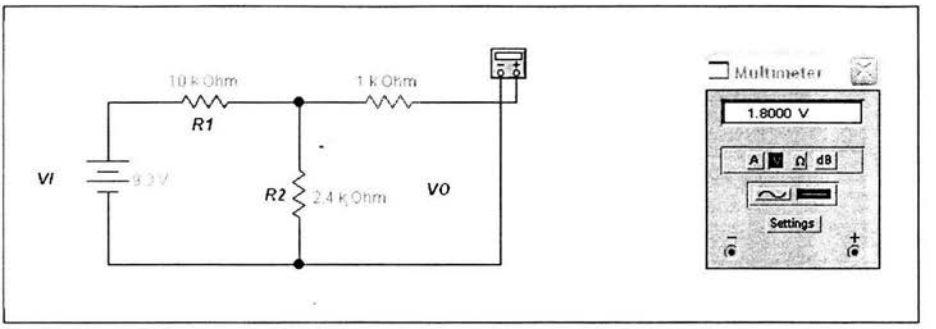

Figura 5-8. Divisor de voltaje para 1.8V.

5.2.2 Adaptador Bluetooth USB.

Para la interfaz USB se utilizó un cable **A/B** macho/macho. Del lado del conector **B** se tomaron las líneas **D+** y **D-** para el módulo, quedando en el otro extremo del cable el conector **A** que se conecta a la computadora.

La configuración final del modulo quedó como lo muestra la Figura 5-9.

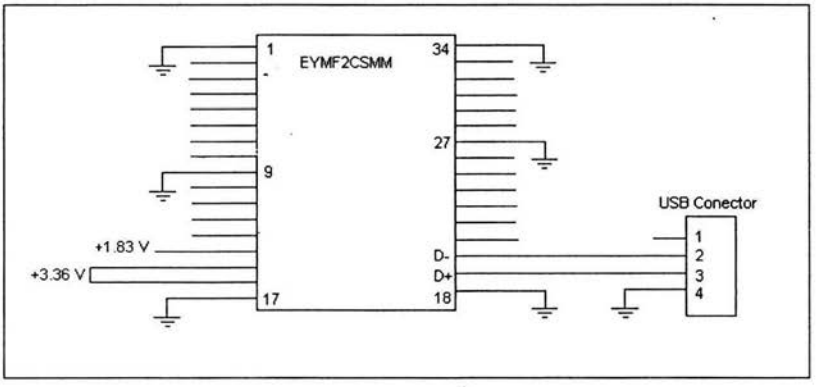

Figura 5-9. Diagrama de la configuración del módulo.

En la Figura 5-10 se muestra la imagen del adaptador **Bluetooth** con la batería de 9V y su cable USB.

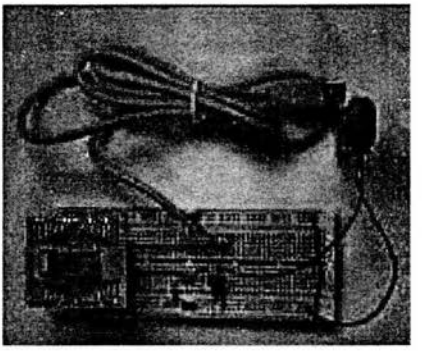

Figura 5-10. Imagen del adaptador Bluetooth USB.

5.3 Software para la pila de protocolos Bluetooth.

El software para el host **Bluetooth** corresponde a las capas de la pila de protocolos y utilidades **Bluetooth** implementadas en software e instaladas en el host.

En esta plataforma es necesario tener una interfaz UART o USB para comunicar el host y el módulo **Bluetooth**.

Existen muchos softwares para el host, implementados en diversos lenguajes de programación y sobre distintas plataformas, siendo también muchas las empresas interesadas en su desarrollo. Independientemente de la plataforma o el lenguaje de programación se basan en la especificación del sistema **Bluetooth**.

La Figura 5-11 muestra un modelo de implementación de la pila de protocolos **Bluetooth**, discriminando las capas que están implementadas en el host y las que están en el hardware **Bluetooth** (módulos, tarjetas, sistemas de desarrollo **Bluetooth**).

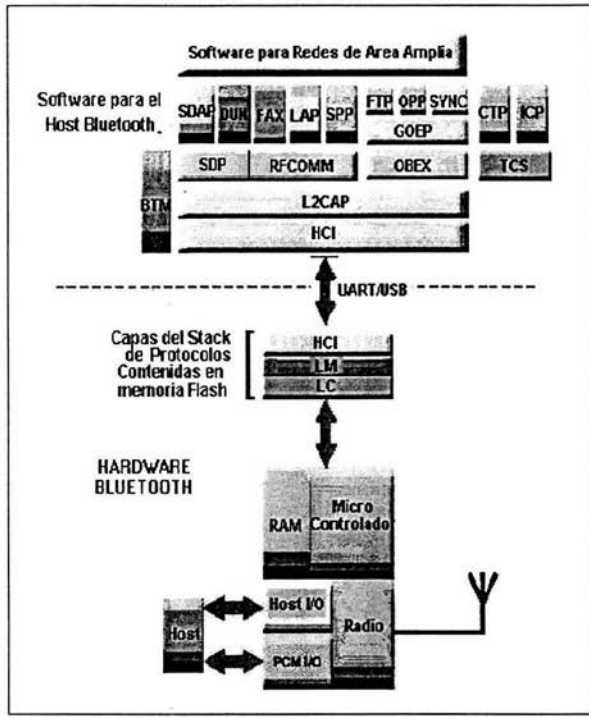

Figura 5-11. Modelo de implementación de la pila de protocolos Bluetooth.

La Figura 5-12 describe esquemáticamente la implementación de la pila de protocolos **Bluetooth** al nivel de software, firmware y hardware, especificando su ubicación ya sea en el host o en el hardware **Bluetooth**.

El host y el hardware **Bluetooth** se comunican a través del HCI. El firmware HCI implementa los Comandos HCI para el hardware **Bluetooth** teniendo acceso a los comandos de banda base, administrador de enlace, registros de estado del hardware, registros de control y registros de eventos.

Pueden existir varias capas entre el driver HCI ubicado en el host y el firmware HCI ubicado en el hardware **Bluetooth**. Estas capas intermedias, se encargan del control y transporte de datos a través de un medio físico (un bus físico ya sea USB, PC Card, RS232, u otro), permitiendo el intercambio de datos y comandos entre estos dos. El host recibe notificaciones asíncronas de los eventos HCI independientemente de la capa de control de transporte del host que se esté usando. Los eventos HCI son usados para notificar al host cuando ocurre algo.

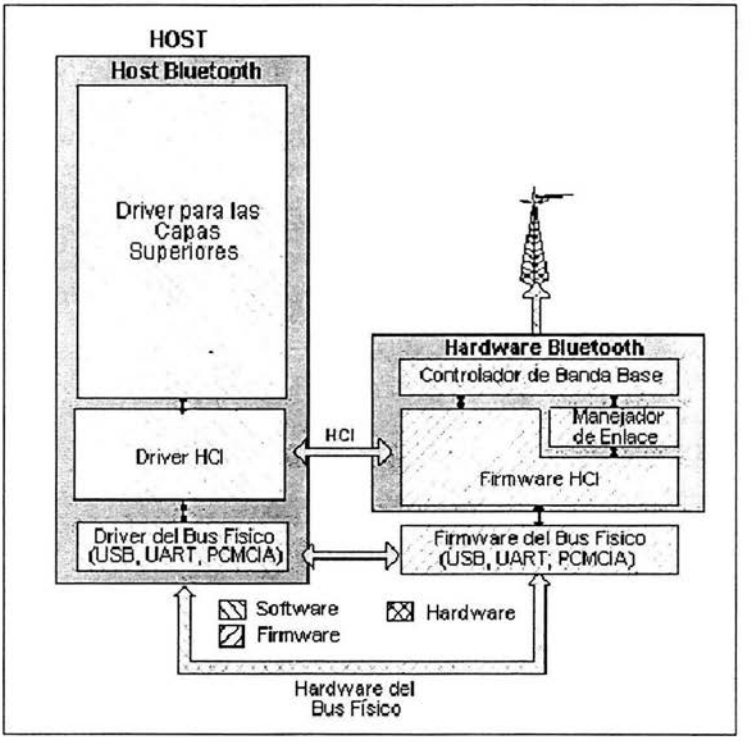

Figura 5-12. Modelo de implementación del protocolo Bluetooth.

Existen varias compañías que desarrollan software para la pila de protocolos **Bluetooth** para el host; principalmente los fabricantes de módulos **Bluetooth**, así como fabricantes de productos con esta tecnología, TDK, Ericsson, Nokia, Mezoe entre otras, éste tipo de software generalmente tiene un costo por licencia, o viene incluido con productos o kits de desarrollo **Bluetooth**. Sin embargo existen también empresas y universidades que desarrollan este tipo de software y de libre distribución con licencia pública **GPL** (GNU Public License).

Debido a que el software para plataforma Windows no es de libre distribución y el costo oscila entre los **850 USD** hasta los **2500 USD**, se optó por trabajar en la plataforma Linux donde se pueden encontrar varios softwares de libre distribución como son:

- **OpenBT.** Axis Communications Inc. desarrolló, en primera instancia, un driver **Bluetooth** para Linux llamado OpenBT, el cual puede ser usado tanto en ambientes eLinux (Linux Embebido) como en Linux para PC, éste fue la primera pila de protocolos disponible, siendo ahora un proyecto de código fuente abierto, desarrollado en un kernel de Linux 2.0.33, de tal manera que no debe presentar problemas de funcionamiento en versiones posteriores del kernel.

- **Affix.** Esta pila de protocolo para Linux fue desarrollado por el Centro de Investigación de Nokia en Helsinki, Finlandia. El software **Bluetooth** de **Affix** trabaja sobre plataformas Intel, ARM (iPaq, ARM9 y otras) y PowerPC (iMac).

- **Bluez.** Este es la pila de protocolos **Bluetooth** oficial de Linux, el cual es parte del kernel 2.4 y posteriores. Bluez brinda a sus usuarios la capacidad de comunicación con dispositivos **Bluetooth**, entre los que se tienen adaptadores USB, teléfonos móviles y puntos de acceso entre otros, además, de conexión inalámbrica entre dos o más computadoras. Bluez consta del core HCI, drivers HCI para UART, USB y emulador de HCI, módulo del protocolo L2CAP y utilidades de configuración y prueba.

Después de revisar la documentación de estos tres paquetes y de probar su instalación, se decidió optar por Bluez; ya que además de contar con la documentación necesaria para su implementación, no causa conflictos con el kernel y cuenta con todos los paquetes necesarios para su instalación; además de que es confiable, porque como se había mencionado es el software oficial de Linux.

5.3.1 Bluez

Bluez provee soporte para las capas y protocolos de la especificación **Bluetooth**. Sus principales características son:

- Arquitectura modular, eficiente y flexible.
- Soporta varios dispositivos **Bluetooth**.
- Procesamiento de datos para multienlaces.

Bluez consta de (Figura 5-13):

- Especificación del HCI.
- Drivers HCI para UART, USB y emulador de HCI.
- Módulos para el protocolo L2CAP.
- Utilidades de configuración y prueba.

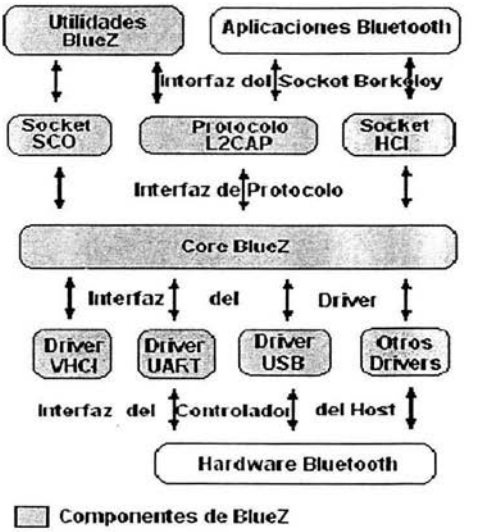

Figura 5-13. Arquitectura Bluez.

El código fuente de Bluez se puede obtener de la página oficial de Bluez en Internet. Requiere para su funcionamiento un Kernel de Linux 2.4.4 o posterior, si se desea utilizar las últimas versiones, es necesario deshabilitar del kernel el soporte nativo de **Bluetooth**.

Bluez se puede usar con dispositivos **Bluetooth** con interfaz Serial o USB, además brinda un dispositivo HCI virtual (*vhci*) el cual permite depurar y probar las aplicaciones sin necesidad de tener conectados dispositivos **Bluetooth** reales.

Bluez se compone de los siguientes paquetes:

- bluez-libs-2.7.tar.gz
- bluez-utils-2.7.tar.gz
- bluez-pin-0.23.tar.gz
- bluez-hcidump-1.8.tar.gz
- bluez-hciemu-1.0.tar.gz
- bluez-bluefw-1.0.tar.gz

Los archivos **README** o **INSTALL** incluidos dentro de cada paquete contienen toda la información sobre los requerimientos e instrucciones para su instalación.

Para que Bluez trabaje correctamente se deben escribir las siguientes líneas en el archivo **/etc/modules.conf** después de instalar los paquetes:

alias net-pf-31 bluez
alias bt-proto-0 l2cap
alias bt-proto-2 sco
alias bt-proto-3 rfcomm
alias bt-proto-4 bnep
alias tty-ldisc-15 hci_uart
alias char-major-10-250 hci_vhci

Después de esto se debe correr el comando "**depmod –a**" para habilitar la carga automática de los módulos Bluez. Estos módulos también se pueden cargar manualmente mediante el comando "**modprobe**" acompañado de uno de los siguientes módulos:

- **bluez** Core **Bluetooth**.
- **hci_usb Driver** *HCI USB*.
- **hci_uart Driver** *HCI UART*.
- **hci_vhci Driver** para el dispositivo virtual *HCI*.
- **l2cap** Módulo *L2CAP*.
- **sco** Módulo SCO.(**Synchronous Connection Oriented**).
- **rfcomm** Módulo *RFCOMM*.

Para inicializar un dispositivo USB, primero se debe de contar con el soporte apropiado para USB en el sistema operativo. Después se debe cargar el siguiente módulo:

modprobe hci_usb

El dispositivo es inicializado cuando se inserta en el puerto USB. Para activar el dispositivo **Bluetooth** se ocupa el comando *hciconfig*:

hciconfig hci0 up

Algunos comandos con los que cuenta Bluez (para poder ver los detalles de estas herramientas se puede utilizar el comando *man*) son:

- **hciconfig:** Herramienta de configuración HCI de dispositivos.
- **L2ping:** Herramienta para hacer un ping en la capa L2CAP.
- **L2test:** Herramienta de prueba de la capa L2CAP.
- **hcitool:** Herramienta de monitoreo y escritura con la interfaz HCI.

5.4 Pruebas de aplicación y desempeño de la pila de protocolos Bluetooth.

Para las pruebas se ocupó el adaptador **Bluetooth** con el módulo **EYMF2CSMM**, una computadora *Sony Vaio* con arquitectura *Athlon* y sistema operativo **SUSE LINUX 9.0** con un **kernel 2.4.21-99-athlon** y la última versión de **Bluez**. También se ocupó un teléfono celular *Sony Ericsson* modelo **T610**, el cual cuenta con tecnología **Bluetooth** y los siguientes servicios: *transferencia de archivos, manos libres, audífono, inserción de objeto, puerto serial, sincronización y marcación.*

En la Figura 5-14 se muestra un esquema de los dispositivos utilizados en las pruebas de la comunicación **Bluetooth**.

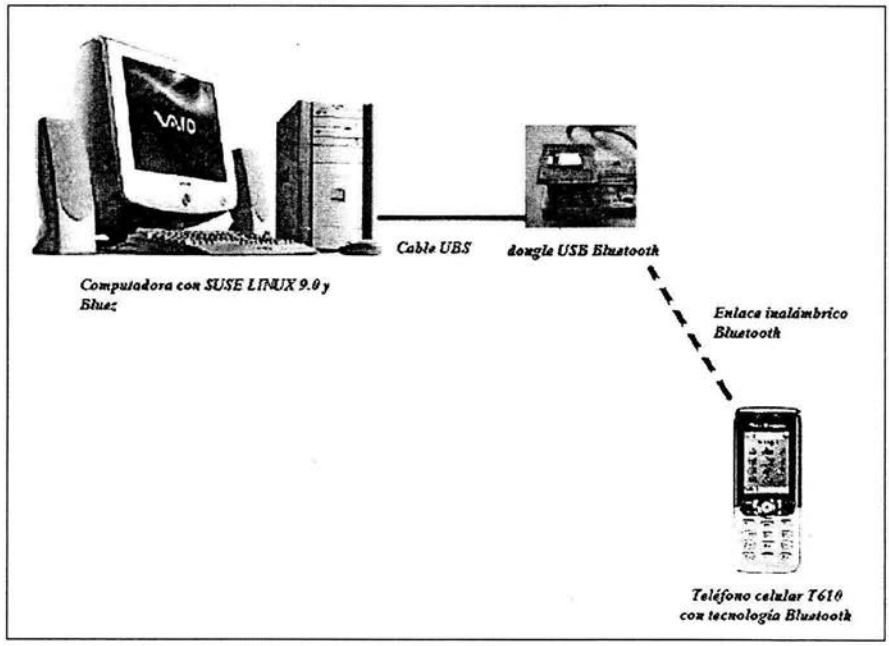

Figura 5-14. Dispositivos de las pruebas de comunicación Bluetooth.

El objetivo de estas pruebas es poder observar por medio de las herramientas de pila de protocolo de Bluez, las características de la pila de protocolos del módulo y del teléfono, y hacer pruebas de comunicación entre ellos.

En la Figura 5-15 muestra un diagrama con las pruebas que se realizaron.

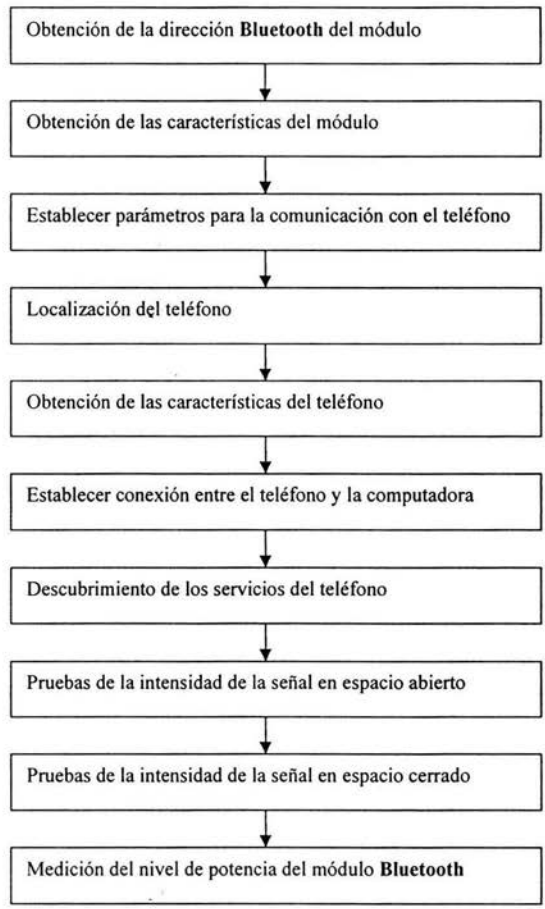

Figura 5-15. Diagrama de las pruebas para el módulo Bluetooth y el teléfono T610.

5.5 Conclusiones del capítulo.

Después de estudiar la especificación **Bluetooth** y los elementos de implementación se decidió diseñar e implementar un adaptador **Bluetooth** para PC, con la finalidad de que el host fuese la PC, ya que de otro modo se tendría que utilizar un microcontrolador, el cual requiere de software para la pila de protocolos **Bluetooth**, el cual no es fácilmente accesible.

Se eligió el módulo **EYMF2CSMM** de TAIYO YUDEN con interfaz USB, ya que es un módulo accesible en el mercado y tiene la antena integrada a diferencia de otros.

Se elaboró una base para poder montarse en una protoboard y poder hacer pruebas debido a que no se cuanta con la configuración exacta de los pines en la hoja de datos del fabricante.

Se utilizó una batería de **9V** para su alimentación con dos divisores de voltaje para poder obtener los dos voltajes requeridos para su alimentación de corriente.

Se eligió utilizar el software "Bluez" para la pila de protocolos **Bluetooth**, ya que es el software oficial de Linux y es de libre distribución, a diferencia de los softwares de plataforma Windows que tienen un costo por licencia, y su precio es poco accesible para este tipo de proyectos, ó vienen acompañados de Kits de desarrollo, los cuales tampoco tienen un precio accesible.

Con esto se tienen los elementos necesarios para poder realizar las pruebas y diseñar aplicaciones prácticas para poder verificar el funcionamiento de la tecnología **Bluetooth**.

Aunque en este trabajo de investigación se desarrollo un dispositivo **Bluetooth** para comunicar una PC, se puede pensar en mucho más aplicaciones que todavía no han sido implementadas.

Por ejemplo, una alarma que funcione con un módulo de Clase 1 para tener un alcance de 100 mts implementado en un automóvil y que envíe una señal de alerta a un teléfono móvil o a una PDA mientras el propietario se encuentra en la oficina o en un Centro Comercial, sustituyendo la ruidosa alarma que se utiliza en la actualidad e incluso se podría mandar alguna instrucción al automóvil.

En la actualidad existen vehículos a los cuales se puede integrar la tecnología **Bluetooth,** por ejemplo los vehículos BMW series 3, 5, 7, X3 y X5, para los cuales existe un dispositivo que se integra en el sistema electrónico del automóvil y que, una vez instalado, permite prescindir del controlador de un manos libres como el kit CK3000 de Parrot, y traspasar todas sus funciones a los mandos del volante del BMW, así que se puede pensar en una variante de este dispositivo para que tenga mayor alcance y sea compatible con los teléfonos móviles o PDAs que cuenten con esta tecnología.

En la Figura 5-16 se muestra un ejemplo donde suponiendo que alguien abre ilegalmente alguna puerta del vehículo en el estacionamiento de un Centro Comercial, inmediatamente se envía una alerta al teléfono móvil del propietario para dar aviso de esta intrusión y tome alguna acción como hacer una llamada a un número de emergencia, inhibir el sistema de arranque o cerrar los seguros permanentemente una vez que la puerta se vuelva a cerrar con la intención de dejar atrapado al intruso.

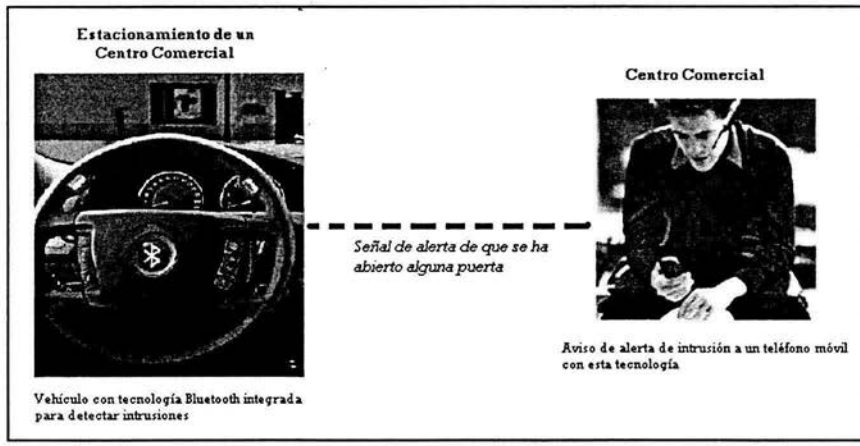

Figura 5-16. Ejemplo de una alarma Bluetooth integrada en un automóvil.

También se podría utilizar para sustituir los cables en los instrumentos musicales, asociando al instrumento con uno o más amplificadores, lo cual sería muy conveniente en un concierto, ya que sustituiría la gran cantidad de cables que se tienen en el suelo pegados con cinta adhesiva por la necesidad de rotación de instrumentos de las bandas.

La Figura 5-17 muestra un ejemplo de lo que podría ser una guitarra eléctrica asociada a un amplificador por medio de la tecnología **Bluetooth** en vez de estar conectadas por cable para lo que se necesitaría un enlace síncrono debido a que la señal enviada sería analógica y se tendría que transmitir como audio.

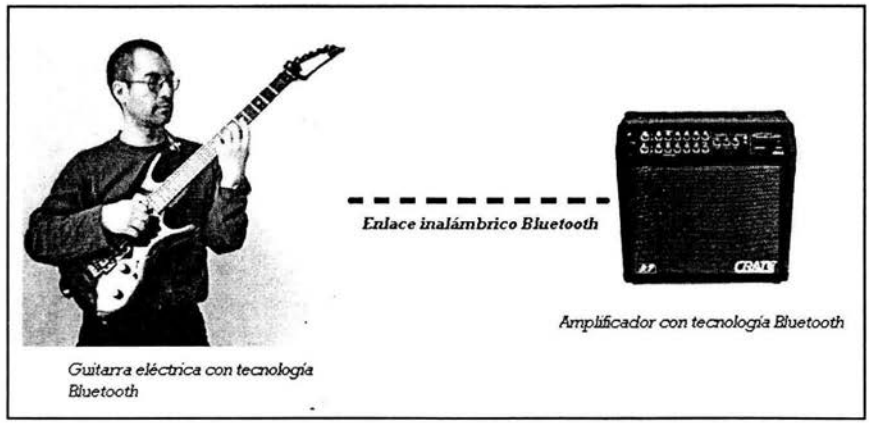

Figura 5-17. Guitarra eléctrica con tecnología Bluetooth para transmitir su audio.

Otra aplicación que se podría encontrar en la tecnología **Bluetooth** es en el hogar, ya que gran cantidad de electrodomésticos y todo tipo de objetos electrónicos son susceptibles de ser monitorizados y controlados dentro de este entorno, sistemas de iluminación, persianas, climatización, detección de movimientos y alarmas. Este tipo de implementación en el hogar se denomina "Domótica", pero actualmente se enfoca en el uso de Internet como se muestra en la Figura 5-18 para el control a distancia de los dispositivos que componen la instalación electrónica de una vivienda inteligente, pero en el momento de llegar a casa es cuando la tecnología **Bluetooth** puede tomar el control.

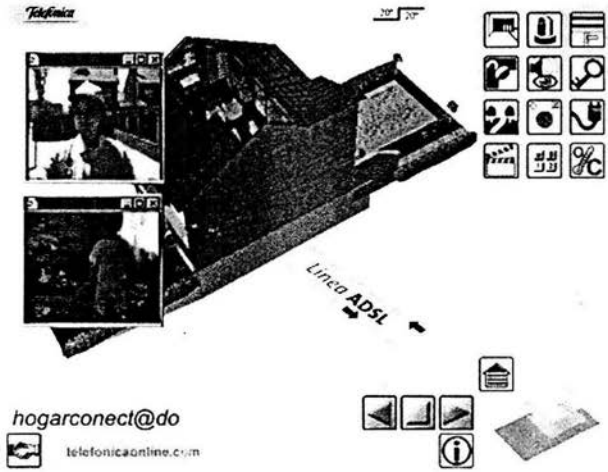

Figura 5-18. Monitoreo y control de dispositivos electrónicos por Internet.

Al llegar al hogar se podría utilizar una PDA para controlar dispositivos electrónicos implementados con tecnología **Bluetooth**, para tener control por ejemplo de una televisión, un stereo, una lavadora, un microondas o un refrigerador. O tal vez monitorear una cámara en el cuarto de un bebe mientras el usuario realiza otra actividad en otro cuarto; también se podría programar con el sistema de iluminación para que se enciendan y se apaguen ciertas luces o ciertos aparatos electrodomésticos cuando el usuario está fuera del hogar algunos días, para simular que hay alguien dentro del hogar.

CAPÍTULO 6.
RESULTADOS Y CONCLUSIONES.

En este capítulo se presentan los resultados (pantallas del shell de Linux) de las pruebas aplicadas al adaptador **Bluetooth** y al teléfono **T610**. Los procedimientos completos y sus resultados se presentan en el apéndice D.

6.1 Resultados de las pruebas de aplicación y desempeño.

Después de instalar el software Bluez adecuadamente se cargaron los módulos necesarios (sección D.1).

- Con el comando *hciconfig* se puede ver el estado de nuestro adaptador USB:

```
linux:~ # hciconfig
hci0:   Type: USB
        BD Address: 00:00:00:00:00:00 ACL MTU: 0:0  SCO MTU: 0:0
        DOWN
        RX bytes:0 acl:0 sco:0 events:0 errors:0
        TX bytes:0 acl:0 sco:0 commands:0 errors:0
```

Este resultado muestra que no hay algún dispositivo **Bluetooth** activo en el sistema, por lo que hay que dar de alta al dispositivo con el comando *hciconfig hci0 up*, donde *hci0* es el identificador de nuestro dispositivo **Bluetooth**:

```
linux:~ # hciconfig hci0 up
linux:~ # hciconfig
hci0:   Type: USB
        BD Address: 00:50:F2:7E:BC:D3 ACL MTU: 192:8  SCO MTU: 64:8
        UP RUNNING PSCAN ISCAN
        RX bytes:69 acl:0 sco:0 events:8 errors:0
        TX bytes:27 acl:0 sco:0 commands:7 errors:0
```

Con este comando podemos ver la dirección **Bluetooth** del dispositivo que consta de 12 hexadecimales (equivalente a 48 bits) como se vio en la sección 3.3.4.

- Con el comando *hciconfig -a* se pueden observar más detalles del módulo (sección D.2), o se puede utilizar el comando *hciconfig hci0 features* para poder observar más características:

```
linux:~ # hciconfig hci0 features
hci0:  Type: USB
       BD Address: 00:50:F2:7E:BC:D3 ACL MTU: 192:8  SCO MTU: 64:8
       Features: 0xff 0xff 0x0f 0x00 0x00 0x00 0x00 0x00
            <3-slot packets> <5-slot packets> <encryption> <slot offset>
            <timing accuracy> <role switch> <hold mode> <sniff mode>
            <park state> <RSSI> <channel quality> <SCO link>
            <HV2 packets> <HV3 packets> <u-law log> <A-law log>
            <CVSD> <paging scheme> <power control> <transparent SCO>
```

Aquí podemos observar que soporta paquetes multiranuras de 3 y 5 ranuras (sección 3.3.5); soporta encriptación (sección 3.2.9); cambio de rol (apéndice C); modos de conexión (sección 3.3.8): *sostenido* (hold), *olfateo* (sniff) y *estacionado* (park); medición del RSSI (sección 3.2.2); enlaces SCO (sección 3.3.2), esquema *Buscando* (sección 3.3.7), control de potencia (sección 3.2.4).

- Con el comando *hciconfig hci0 name* podemos cambiar el nombre amigable (apéndice C) del dispositivo y comprobar el cambio con el comando *hciconfig –a* (sección D.3):

```
linux:~ # hciconfig hci0 name JONATAN
```

- Ahora podemos agregar autentificación y encriptación, ya que no están activos, con el fin de poder comunicarnos con el T610, el cual nos requiere autentificación:

```
linux:~ # hciconfig hci0 auth encrypt
linux:~ # hciconfig
hci0:  Type: USB
       BD Address: 00:50:F2:7E:BC:D3 ACL MTU: 192:8  SCO MTU: 64:8
       UP RUNNING PSCAN ISCAN AUTH ENCRYPT
       RX bytes:680 acl:0 sco:0 events:21 errors:0
       TX bytes:314 acl:0 sco:0 commands:17 errors:0
```

- Para poder autentificarlo con el teléfono se debe establecer una clave (sección D.4), y después se ejecuta el comando que asociará esta clave:

```
linux:~ # chmod 710 /etc/Bluetooth/givepin
```

- Después se utiliza el comando *hcitool scan* para localizar todos los dispositivos **Bluetooth** localizados en el área:

```
linux:~ # hcitool scan
Scanning ...
       00:0A:D9:77:6F:8B      T610JONA
```

Con este comando encontramos el teléfono T610 con la dirección **Bluetooth**: 00:0A:D9:77:6F:8B y con el nombre establecido manualmente: T610JONA.

- Ahora se puede ver la información de este dispositivo con el comando *hcitool info* y la dirección **Bluetooth** del teléfono:

```
linux:~ # hcitool info 00:0A:D9:77:6F:8B
Requesting information ...
    BD Address:  00:0A:D9:77:6F:8B
    Device Name: T610JONA
    LMP Version: 1.1 (0x1) LMP Subversion: 0x400
    Manufacturer: Ericsson Mobile Comunications (0)
    Features: 0x04 0xca 0x31 0x00 0x00 0x00 0x00 0x00
        <encryption> <RSSI> <SCO link> <u-law log>
        <A-law log> <CVSD>
```

- Con el comando *hcitool cc* se crea una conexión, y con el comando *hcitool con*, se puede ver que conexiones hay:

```
linux:~ # hcitool cc 00:0A:D9:77:6F:8B
linux:~ # hcitool con
Connections:
    < ACL 00:0A:D9:77:6F:8B handle 41 state 1 lm MASTER AUTH ENCRYPT
```

Aqui se puede ver que está establecida una conexión ACL en el cual el adaptador **Bluetooth** juega el papel de maestro; y el enlace soporta autentificación y encriptación.

- Con el comando *sdptool browse* podemos explorar los servicios que ofrece el teléfono por medio del protocolo de descubrimiento de servicios visto en la sección 3.8. A continuación se muestra la pantalla arrojada con el servicio de DUN en el canal 1, la pantalla completa se muestra en la sección D.5:

```
linux:~ # sdptool browse 00:0A:D9:77:6F:8B
Browsing 00:0A:D9:77:6F:8B ...
Service Name: Dial-up Networking
Service RecHandle: 0x10000
Service Class ID List:
  "Dialup Networking" (0x1103)
  "Generic Networking" (0x1201)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 1
Profile Descriptor List:
  "Dialup Networking" (0x1103)
    Version: 0x0100
```

Como se puede ver en los resultados anteriores, los dos dispositivos cuentan con Indicador de Intensidad de Señal Recibida (RSSI), así podemos hacer pruebas con el comando *hcitool rssi*, para ver la relación de la intensidad de la señal con respecto a la distancia.

A continuación se muestran los resultados de dos pruebas realizadas; una en espacio abierto, y la otra dentro de una casa atravesando cuartos y puertas. Se tomaron 10 mediciones para cada distancia, variando la posición horizontal y vertical del teléfono, a continuación se muestra la primera medición para cada distancia (los resultados completos aparecen en la sección D.6).

Para espacio abierto sin obstáculos:

A 0 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 9
```

A 1 mt:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 2
```

A 2 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
```

A 3mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
```

A 4 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -7
```

A 5mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
```

A 6mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -3
```

A 7 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
```

A 8 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
```

A 9 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
```

A 10mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
```

Después de los 10 mts. se siguió alejando el teléfono, con el cual se obtuvo un valor constante de –10, hasta llegar a los 14 mts. donde se perdió la conexión.

Para espacio cerrado con obstáculos:

Para 1 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
```

A 2 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -1
```

A 3 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
```

A 4 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
```

A 5mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
```

A 6 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
```

A 7 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
```

En los 8 mts se perdió la conexión.

Después de obtener estos datos, se sacó la media de cada distancia para poder graficar los resultados.

En la Tabla 6-1 se muestran las medias de los resultados para espacio abierto.

Distancia (mts)	Potencia (dBm)	Potencia (mW)
0	24.6	288.403
1	0.3	1.072
2	0.3	1.072
3	-.06	0.871
4	-5.2	0.302
5	-5.6	0.275
6	-3.5	0.447
7	-6.8	0.209
8	-9.2	0.12
9	-8.5	0.141
10	-10	0.1
11	-10	0.1
12	-10	0.1
13	-10	0.1
14	Sin conexión	Sin conexión

Tabla 6-1. Resultados del RSSI en espacio abierto.

En la Tabla 6-2 se muestran las medias de los resultados para espacio cerrado.

Distancia (mts)	Potencia (dBm)	Potencia (mW)
0	24.6	288.403
1	0	1
2	-0.2	0.955
3	-1.9	0.646
4	-7.7	0.17
5	-9.5	0.112
6	-9.7	0.107
7	-10	0.1
8	Sin conexión	Sin conexión
9	Sin conexión	Sin conexión
10	Sin conexión	Sin conexión

Tabla 6-1. Resultados del RSSI en espacio abierto.

En las Figuras 6-1 y 6-2, se muestran las gráficas de los resultados obtenidos.

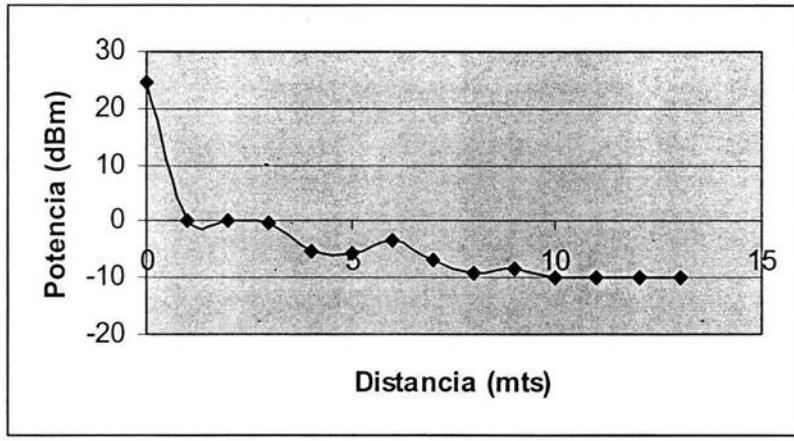

Figura 6-1. Gráfica de los resultados del RSSI en espacio abierto.

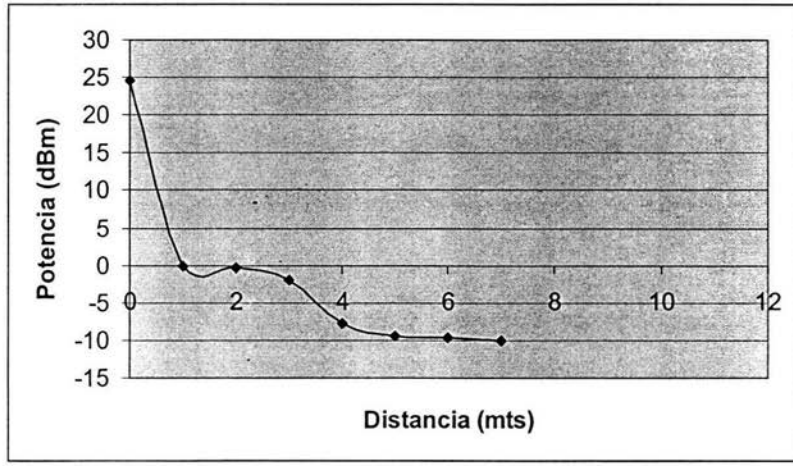

Figura 6-2. Gráfica de los resultados del RSSI en espacio cerrado.

Por último con el comando *hcitool tpl* podemos verificar el nivel de potencia de transmisión:

```
linux:~ # hcitool tpl 00:0A:D9:77:6F:8B
Current transmit power level: 0
```

El cual no varia, se mantiene en 0 dBm sin importar la distancia (sección 3.2.2).

6.2 Conclusiones.

En este proyecto se implementó el hardware y el software de un adaptador **USB Bluetooth** para poder hacer pruebas y comprobar el funcionamiento práctico de esta tecnología. Estas pruebas se realizaron utilizando el dispositivo implementado y un teléfono celular con **Bluetooth** (**T610** de Sony Ericsson); los cuales permitieron verificar los diferentes conceptos de esta tecnología vistos en el capítulo 3.

El proceso inicial del proyecto fue conocer y entender las diferentes tecnologías inalámbricas que existen en la actualidad para ver sus principales ventajas y desventajas, así como sus diferencias, para poder determinar a que aplicaciones se enfoca cada una.

El siguiente paso fue estudiar la especificación **Bluetooth 1.1**, para poder comprender de manera teórica el funcionamiento de esta tecnología; y aunque no contiene instrucciones para la implementación de dispositivos, es la base para poder entender el funcionamiento práctico y los elementos de la implementación.

Después se estudió las interfaces que serían necesarias para la comunicación con el host (en este proyecto, el host fue una computadora de escritorio), donde se encontraría el software con la pila de protocolos **Bluetooth**.

Con estos elementos se realizó la investigación de los componentes necesarios para poder implementar un dispositivo con esta tecnología; durante este proceso se analizaron varias posibilidades de dispositivos así como sus aplicaciones, tomando en cuenta el costo y la disponibilidad de los elementos, los cuales fueron los principales limitantes del proyecto, debido a que el costo actual es muy elevado por ser una tecnología reciente (solamente el módulo tiene un costo aproximado de **60 USD**), además de que los elementos necesarios no están disponibles en México; por ejemplo el módulo **Bluetooth** se tuvo que adquirir en Inglaterra y los reguladores de voltaje no se pudieron encontrar.

Tampoco se encontró una base para montar el módulo por lo que se tuvo que diseñar un circuito impreso que permitiera conectarlo a una protoboard.

Después de contar con la parte de hardware del dispositivo, se procedió a investigar el software necesario para poder hacerlo funcionar y realizar las pruebas. Se determinó trabajar en la plataforma Linux, debido a que el software con la pila de protocolos **Bluetooth** para Windows tiene un costo elevado, a diferencia del software para Linux que es de libre distribución.

Se tuvieron que hacer varias pruebas con los diferentes softwares disponibles debido a que como son de libre distribución, muchos de estos tienen algunos errores o deficiencias, y se implementan de diferente manera según el hardware y la distribución de Linux utilizada.

Después de realizar varias pruebas de instalación y funcionamiento con distintos softwares se decidió trabajar con la pila de protocolos oficial de Linux: "**Bluez**", ya que funciona adecuadamente con el software y hardware que se utilizó y cuenta con las herramientas necesarias para realizar las pruebas.

Después de realizar las pruebas, se ha comprobado la eficiencia de esta tecnología al conectar un dispositivo cercano (máximo 10 mts. en espacio abierto como lo observamos en las gráficas obtenidas de forma práctica), con buena confiabilidad en el enlace y bajo consumo de potencia, por lo cual se puede concluir que una vez que los precios de los módulos disminuyan, podemos encontrar en la tecnología **Bluetooth** la mejor opción para la conexión inalámbrica de dispositivos personales tales como PDAs, teléfonos celulares, computadoras, impresoras, sistemas GPS, cámaras digitales e incluso se puede pensar en el uso de esta tecnología en electrodomésticos. La utilidad de **Bluetooth** sólo está limitada por la imaginación de los ingenieros y de los usuarios.

Todas las herramientas desarrolladas en este trabajo de investigación se conjugaron a favor de esta tecnología, cumpliendo satisfactoriamente los objetivos propuestos e introduciendo una tecnología de punta que apenas se está dando a conocer en México.

Apéndice A.
Glorasio de términos y acrónimos.

ACL (Asynchronous Connectionless): Enlace Asíncrono sin Conexión.

ACK (Acknowledgement): Acuse de recibido.

BD_ADDR: Dirección del dispositivo Bluetooth.

CRC (Cyclic Redundancy Check): Verificación de Redundancia Cíclica.

CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance): Acceso Múltiple por Detección de Portadora con Evasión de Colisiones.

CTS (Clear to Send): Limpio para enviar.

dBm: Decibel referido a un miliwatt; dBm = 10 log (salida potencia mW / 1 mW)

Driver: Controlador (software) que permite gestionar los periféricos que están conectados a una computadora.

FCC (Federal Communications Comisión): Comisión Federal de Comunicaciones. Su principal función es la de mantener el control sobre el amplio sector de las telecomunicaciones en los Estados Unidos.

FIFO (First In First Out): El primero en entrar es el primero en salir.

Firmware: Parte del software de una computadora que no puede modificarse por encontrarse en la ROM (Read Only Memory) o memoria de sólo lectura. Es una mezcla entre el hardware y el software, es decir tiene parte física y una parte de programación. Un ejemplo típico de Firmware lo constituye la BIOS.

GFSK (Gaussian Frequency Shift Keying): Desplazamiento de Frecuencia Gaussiano.

GNU: El Proyecto GNU fue iniciado en 1984 con el propósito de desarrollar un sistema operativo compatible con Unix que fuera software libre (GNU es un acrónimo recursivo para "GNU No es Unix".

GPL (General Public License): Licencia Pública General, desarrollada por la FSF o Free Software Foundation. El programa puede ser instalado sin limitación en una o varias computadoras. En las distribuciones de estos programas debe estar incluido el código fuente.

GPS (Global Positioning System): Sistema de Posicionamiento Global.

Hardware: Componentes electrónicos y electro-mecánicos de una computadora o cualquier otro sistema. Este término es usado para distinguir estos componentes físicos de los datos y programas.

IrDA (Infrared Data Association): Asociación de Datos por Infrarrojo.

Host: Sistema microprocesado programable (PCs, teléfonos celulares, mouse, impresoras, teclados, sensores inalámbricos, etc.) capaz de ejecutar líneas de código.

HyperLAN: Estándar de comunicaciones inalámbricas europeo desarrollado por el Instituto de Telecomunicaciones europeo (ETSI).

Kernel: núcleo, es la parte fundamental de un programa, por lo general de un sistema operativo, que reside en memoria todo el tiempo y que provee los servicios básicos. Es la parte del sistema operativo que está más cerca de la máquina y puede activar el hardware directamente o unirse a otra capa de software que maneja el hardware.

LAN (Local Area Network): Red de Área Local.

OSI (Open Systems Interconnection): Sistema de Interconexión Abierta con modelo de protocolo de siete niveles desarrollado por la Asociación Internacional de Estándares.

PAN (Personal Area Network): Red de Área Personal.

PCB (Printed Circuit Board): Tarjeta de circuito impreso.

PDA (Personal Digital Assistance): Asistente Digital Personal.

QoS: Calidad de Servicio.

RADIUS: Servidor del Servicio de Usuario de Acceso Telefónico de Autenticación Remota.

RF: Radio Frecuencia.

RTS (Ready to Send): Listo para enviar.

SCO (Synchronous Connestion-Oriented): Enlace Síncrono Orientado a la Conexión.

SIG: Special Interest Group.

TOKEN RING: Es una red de topología de anillo que se sirve del pase de *testigos* (Tokens). La frase también se aplica a una topología de pase de *testigos* específica definida por la IBM Corporation.

Transreceptor: Transmisor-receptor.

UART (Universal Asynchronous Receiver Transmitter): El transmisor receptor universal asíncrono, es un dispositivo que multiplexa datos paralelos en seriales para ser transmitidos y convierte en paralelos los datos seriales recibidos.

USB (Universal Serial Bus): Bus Universal Serial.

WAN (Wide Area Nterwork): Red de Área Global.

WEP (Wired Equivalent Privacy): Privacidad equivalente a cable.

Apéndice B.
Logotipos de los estándares.

B.1 Logotipo de la Asociación de datos por infrarrojos IrDA:

B.2. Logotipo de la alianza Wi-Fi:

B3. Logotipo de Bluetooth:

Apéndice C.
Lista de PDUs del LMP.

A continuación se muestran los tipos de PDUs, así como sus funciones y operaciones y si son o no obligatorios: (M) debe ser soportado, (O) opcional.

Respuesta General: (M) LMP_accepted, LMP_not_accepted

Estos PDUs son usados como mensajes de respuesta a otros PDUs en diferentes procesos, conteniendo el código del mensaje que está siendo respondido.

Autentificación: (M) LMP_au_rand, LMP_sres

El procedimiento de autentificación está basado en un esquema de desafío-respuesta. El verificacdor manda un PDU LMP_au_rand al solicitante que contiene un número aleatorio (el desafío). El solicitante calcula una respuesta, la cual está en función del desafío, la dirección BD_ADDR del solicitante y una llave secreta. Para calcular la respuesta es necesario que dos dispositivos compartan una clave secreta. Tanto el maestro como el esclavo pueden ser verificadores.

Parejas: (M) LMP_in_rand, LMP_au_rand, LMP_sres, LMP_comb_key, LMP_unit_key

Cuando dos dispositivos no cuentan con una clave de enlace común, se crea una clave de inicialización (Kinit) basada en un PIN y un número aleatorio. Cuando ambos dispositivos han calculado la Kinit la clave de enlace es creada, y finalmente se puede hacer una autentificación mutua. El procedimiento para hacer parejas comienza con un dispositivo mandando un LMP_in_rand; el cual se le conoce como "iniciador" o "iniciador LM". El otro dispositivo se conoce como "respondiendo LM" o "contestador".

Cambiar Clave de Enlace: (M) LMP_comb_key

Si la clave de enlace está derivada de una combinación de claves y el enlace actual es la clave de enlace semi-permanete, la clave de enlace puede ser cambiada. Si la clave de enlace es la clave de una unidad, las unidades deben llevar a cabo el procedimiento de parejas para poder cambiar esta clave. El contenido de LMP_comb_key está protegida por una operación XOR con la clave de enlace actual.

Cambio de la Clave de Enlace Actual: (M) LMP_temp_rand, LMP_temp_key, LMP_use_semi_permanent_key

La clave de enlace actual puede ser una clave de enlace semi-permanente o una clave de enlace temporal. Esta puede ser cambiada temporalmente, pero el cambio es válido solamente para la sesión. El cambio a una clave de enlace temporal, es necesario cuando se está manejando en la piconet bróadcast encriptado.

Encriptación: (O) LMP_encryption_mode_req,LMP_encryption_key_size_req, LMP_start_encryption_req, LMP_stop_encryption_req

Si se ha llevado a cabo por lo menos una autentificación, se puede utilizar encriptación. Si el maestro quiere que todos los esclavos de la piconet utilicen los mismos parámetros de encriptación, debe emitir una clave temporal (Kmaster) y hacer esta clave la clave de enlace actual para todos los esclavos en la piconet antes de empezar la encriptación. Esto es necesario si los paquetes de un bradcast se necesitan encriptar.

Requisición del desplazamiento de reloj: (M) LMP_clkoffset_req, LMP_clkoffset_res

Cuando un esclavo recibe el paquete FHS, la diferencia es calculada entre su propio reloj y el reloj del maestro incluyendo la carga útil del paquete FHS. El desplazamiento también es actualizado cada vez que un paquete es recibido del maestro. El maestro puede solicitar este desplazamiento de reloj en cualquier momento de la conexión. Guardando este desplazamiento de reloj, el maestro sabe en que canal de RF el esclavo despierta de *Exploración de Averiguación* después de que ha dejado la piconet.

Requisición del desplazamiento de la ranura: (O) LMP_slot_offset

Con el LMP_slot_offset se puede transmitir la información de la diferencia de los márgenes de las ranuras en diferentes piconets. Este PDU transporta los parámetros del desplazamiento de las ranuras y la BD_ADDR. Antes de hacer un switcheo maestro y esclavo, este PDU debe ser transmitido por el dispositivo que será el maestro. Este PDU puede ser útil también comunicaciones inter-piconet.

Solicitud de Información de Exactitud de Tiempo: (O) LMP_timing_accuracy_req, LMP_timing_accuracy_res

LMP soporta la petición de la exactitud de tiempo. Esta información puede ser usada para minimizar la búsqueda de la ventana cuando se hace una búsqueda de ranuras de modo *olfateo* o paquetes guías del modo *estacionado*..

Versión LMP: (M) LMP_version_req, LMP_version_res

El dispositivo al que se le solicita este parámetro manda de respuesta tres parámetros: VersNr, Compld y Sub-VersNr. VersNr especifica la versión de la especificación LMP **Bluetooth** que el dispositivo soporta. Compld es usado para detectar posibles problemas con las capas más bajas de **Bluetooth**, y es propia de cada fabricante. Cada fabricante es responsable de la administración y mantenimiento del SubVersNr. Es recomendable que cada fabricante tenga un único SubVersNr por cada implementación RF/BB/LM.

Características Soportadas: (M) LMP_features_req, LMP_features_res

El Radio Bluettoth y el controlador de enlace pueden soportar solo algún subconjunto de tipos de paquetes descritos en la especificación Banda Base y la especificación de Radio. Por lo tanto un dispositivo no mandará otros paquetes que no sean ID, FHS, NULL, POLL, DM1 o DH1 antes de asegurarse de las características soportadas por el otro dispositivo.

Switcheo Maestro-Esclavo: (O) LMP_switch_req, LMP_slot_offset

En el momento que un dispositivo en búsqueda toma el papel de maestro en la piconet, se puede necesitar este switcheo. Supongamos que un dispositivo A es esclavo y un dispositivo B es maestro. El dispositivo que inicializa el switcheo finaliza la transmisión L2CAP actual y manda un LMP_switch_req. Si el switcheo es aceptado, el otro dispositivo finaliza el mensaje L2CAP actual y responde con LMP_accepted. Después se lleva a cabo el procedimiento de switcheo, donde el dispositivo A se convierte en el maestro y el dispositivo B en esclavo.

Requisición de Nombre: (M) LMP_name_req, LMP_name_res

El nombre es un nombre amigable asociado con el dispositivo **Bluetooth**, y consiste en un código de máximo de 248 bytes de acuerdo el estándar UFT-8. El nombre está fragmentado en uno o más paquetes DM1.

Separación: (M) LMP_detach

La conexión entre dos dispositivos **Bluetooth** puede finalizarse en cualquier momento por el esclavo o el maestro. Se incluye un parámetro de razón en el mensaje, para informar a la otra parte porqué se finaliza la conexión.

Modo Estacionado: (O) LMP_hold, LMP_hold_req

Una conexión de enlace ACL entre dos dispositivos **Bluetooth** puede ponerse en modo *estacionado* por un tiempo *estacionado* determinado. Durante este tiempo el maestro no transmitirá paquetes ACL. El modo *estacionado* se ocupa por lo general cuando no se necesita enviar datos en un período relativamente largo. En este modo, el transreceptor puede apagarse para ahorrar energía. Otro uso de este modo es en el caso de que un dispositivo quiera descubrir o ser descubierto por otros dispositivos **Bluetooth**, o si quiere unirse a otras piconets.

Modo Olfateo: (O) LMP_sniff_req, LMP_unsniff_req

Para entrar en este modo, el maestro y el esclavo negocian un intervalo Tsniff y un desplazamiento Olfateo Dsniff, los cuales especifican el tiempo de las ranuras Olfateo. El desplazamiento determina el tiempo de la primera ranura Olfateo; después las ranuras Olfateo seguirán después de un intervalo Tsniff. Cuando el enlace está en modo Olfateo, el maestro solo puede empezar la transmisión en una ranura Olfateo. Dos parámetros controlan la actuvidad de escuchar del esclavo. El parámetro Olfateo attempt, determina por cuantas ranuras el esclavo debe escuchar, empezando en la ranura Olfateo, aunque no reciba un paquete con su propia dirección AM. El parámetro Olfateo timeout determina por cuantas ranuras adicionales el esclavo debe escuchar si continua recibiendo solo paquetes con su propia dirección AM.

Modo Estacionado: (O) LMP_park_req , LMP_unpark_PM_ADDR_req , LMP_unpark_BD_ADDR_req , LMP_set_broadcast_scan_window , LMP_modify_beacon

Si el esclavo no necesita participar en el canal, pero necesita estar sincronizado FH, puede ser puesto en modo *estacionado*. En este modo, el dispositivo renuncia a su dirección

AM_ADDR, pero sigue re-sincronizando con el canal despertando en instantes guías, separados por intervalos guías. En los instantes guías, el esclavo en modo *estacionado* puede ser activado nuevamente por el maestro, el maestro puede cambiar los parámetros del modo *estacionado*, transmitir información broadcast o dejar a los esclavos en este modo solicitar el acceso al canal. Cuando un esclavo está en modo *estacionado*, se le asigna una única dirección PM_ADDR, la cual puede ser usada por el maestro para despertar un esclavo.

Control de Potencia: (O) LMP_incr_power_req , LMP_decr_power_req , LMP_max_power , LMP_min_power

Si los valores RSSI diferencian mucho del valor preferido de un dispositivo **Bluetooth**, este puede solicitar un incremento o decremento de la potencia del TX del otro dispositivo. Del lado del maestro, la potencia del TX es completamente independiente para cada esclavo, por lo cual una petición de cambio de potencia, solo afecta al esclavo en cuestión. Si el dispositivo no soporta control de potencia, este es indicado en la lista de Características Soportadas.

Cambio de la Calidad-Manejada del Canal: (O) LMP_auto_rate , LMP_preferred_rate

El envío de un tipo de paquete de información depende de la calidad del canal RF. La medición de la Calidad en el receptor de un dispositivo puede ser usada para determinar esto. Si un dispositivo A quiere tener el control de un dispositivo remoto B, manda un LMP_auto_rate una vez. El dispositivo B puede mandar de respuesta un LMP_preferred_rate si quiere cambiar el tipo de paquetes que transmite el dispositivo A. Este PDU contiene el parámetro del código preferido (con o sin 2/3 FEC) y la medida preferida de los paquetes (medido en ranuras).

Calidad de Servicio: (M) LMP_quality_of_service, LMP_quality_of_service_req

El LM provee capacidad de Calidad de Servicio. Esto se refiere a que el maestro y el esclavo pueden negociar el número de repeticiones para los paquetes del broadcast (NBC).

Enlaces SCO: (O) LMP_SCO_link_req, LMP_remove_SCO_link_req

Cuando se ha establecido una conexión entre dos dispositivos **Bluetooth**, la conexión consiste en un enlace ACL. Se pueden establecer uno o más enlaces SCO. El enlace SCO reserva ranuras separadas por un intervalo, Tsco. La primera ranura reservada para el enlace SCO está definido por Tsco y el retraso, Dsco.

Control de paquetes Multi-ranuras: (M) LMP_max_slot, LMP_max_slot_req

El número de ranuras usadas por un dispositivo pueden ser limitadas. Un dispositivo permite al dispositivo remoto usar un máximo número de ranuras usando el PDU LMP_max_slot. También un dispositivo puede solicitar usar un número máximo de ranuras con el PDU LMP_max_slot_req. Después de establecer una nueva conexión, el valor de default es una ranura, después de establecer la conexión estos PDUs pueden utilizarse en cualquier momento.

Esquema Buscando: (O) LMP_page_mode_req, LMP_page_scan_mode_req

En adición al esquema *buscando* obligatorio, el sistema **Bluetooth** define un esquema opcional de *búsqueda*. LMP provee la negociación de un nuevo esquema *buscando*, usado la siguiente vez que una unidad es buscada.

Supervisión de Enlace: (M) LMP_supervision_timeout

Cada enlace **Bluetooth** tiene un temporizador para supervisa el enlace. Este temporizador se usa para detectar enlaces perdidos debido a dispositivos en movimiento o fuera de rango, dispositivos apagados, o algunos otros casos. Se usa un procedimiento LMP para establecer el valor del tiempo de espera para la supervisión.

Establecimiento de la Conexión: (M) LMP_host_connection_req , LMP_setup_complete

Cuando un dispositivo en búsqueda, desea crear una conexión involucrando capas LM, este manda un LMP_host_connection_req. Cuando el otro lado recibe el mensaje, el host es informado acerca de una conexión entrante. El dispositivo remoto puede o no aceptar la solicitud. Si es aceptado, los procedimientos de seguridad LMP (parejas, autentificación y encriptación) pueden ser invocados. Cuando un dispositivo ya no va a iniciar más procedimientos de seguridad manda un LMP_setup_complete. Cuando ambos dispositivos han mandado sus LMP_setup_complete puede ser transmitido el primer paquete en un canal lógico diferente del LMP.

Modo de Prueba: (M) LMP_test_activate , LMP_test_control

El LMP tiene PDUs que soportan diferentes modos de prueba, los cuales son usados para la certificación y pruebas de conformidad del Radio Y Banda Base **Bluetooth**.

Manejo de Errores: (M) LMP_not_accepted

Si el Administrador de enlace recibe un PDU con código irreconocible, este responde con LMP_not_accepted con la razón de código del PDU LMP desconocido. El parámetro de respuesta es este código desconocido.

Apéndice D.
Procedimientos de Bluez.

En este apartado se presentan los procedimientos completos y las pantallas del shell de Linux que arrojan algunos comandos ocupados en las pruebas.

D.1. Se necesitan cargar los módulos *bluez, hci_usb, l2cap* con el comando *modprobe*, después de cargarlos, deben aparecer en la lista de módulos con el comando *lsmod:*

```
linux:~ # lsmod
Module                  Size  Used by    Not tainted
l2cap                  17392   0  (unused)
usbserial              19836   0  (autoclean) (unused)
parport_pc             28648   1  (autoclean)
isa-pnp                32712   0  (unused)
lp                      6304   0  (autoclean)
parport                25608   1  (autoclean) [parport_pc lp]
ipv6                  227264  -1  (autoclean)
key                    70456   0  (autoclean) [ipv6]
thermal                 6180   0  (unused)
processor               8280   0  [thermal]
fan                     1472   0  (unused)
button                  2380   0  (unused)
battery                 5600   0  (unused)
ac                      1696   0  (unused)
hci_usb                 7992   0  (unused)
bluez                  31332   1  [l2cap hci_usb]
st                     29648   0  (autoclean) (unused)
sr_mod                 14616   0  (autoclean)
sg                     35232   0  (autoclean)
keybdev                 2156   0  (unused)
mousedev                4340   0  (unused)
joydev                  5440   0  (unused)
evdev                   3840   0  (unused)
input                   3488   0  [keybdev mousedev joydev evdev]
usb-ohci               19848   0  (unused)
ehci-hcd               18028   0  (unused)
usbcore                64364   1  [usbserial hci_usb usb-ohci ehci-hcd]
raw1394                18288   0  (unused)
ohci1394               25808   0  (unused)
ieee1394              188260   0  [raw1394 ohci1394]
af_packet              13168   1  (autoclean)
8139too                15084   1
mii                     2640   0  [8139too]
ide-scsi               11056   0
scsi_mod              100788   4  [st sr_mod sg ide-scsi]
ide-cd                 32416   0
cdrom                  29216   0  [sr_mod ide-cd]
nls_cp437               4348   1  (autoclean)
vfat                   11052   1  (autoclean)
fat                    32792   0  (autoclean) [vfat]
nls_iso8859-1           2844   3  (autoclean)
```

```
ntfs            80300  2 (autoclean)
reiserfs       217908  1
```

D.2. El comando *hciconfig -a* arroja la siguiente pantalla:

```
linux:~ # hciconfig -a
hci0:   Type: USB
        BD Address: 00:50:F2:7E:BC:D3 ACL MTU: 192:8  SCO MTU: 64:8
        UP RUNNING PSCAN ISCAN
        RX bytes:69 acl:0 sco:0 events:8 errors:0
        TX bytes:27 acl:0 sco:0 commands:7 errors:0
        Features: 0xff 0xff 0x0f 0x00 0x00 0x00 0x00 0x00
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy:
        Link mode: SLAVE ACCEPT
        Name: 'BlueZ (0)'
        Class: 0x000000
        Service Classes: Unspecified
        Device Class: Miscellaneous,
        HCI Ver: 1.1 (0x1) HCI Rev: 0x1f9 LMP Ver: 1.1 (0x1) LMP Subver: 0x1f9
        Manufacturer: Cambridge Silicon Radio (10)
```

Con este comando podemos ver los tipos de paquetes (vistos en la sección 3.3.5) que soporta el módulo.

D.3. Cambio del nombre amigable con el comando *hciconfig hci0 name:*

```
linux:~ # hciconfig hci0 name JONATAN
linux:~ # hciconfig -a
hci0:   Type: USB
        BD Address: 00:50:F2:7E:BC:D3 ACL MTU: 192:8  SCO MTU: 64:8
        UP RUNNING PSCAN ISCAN
        RX bytes:391 acl:0 sco:0 events:16 errors:0
        TX bytes:297 acl:0 sco:0 commands:12 errors:0
        Features: 0xff 0xff 0x0f 0x00 0x00 0x00 0x00 0x00
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy:
        Link mode: SLAVE ACCEPT
        Name: 'JONATAN'
        Class: 0x000000
        Service Classes: Unspecified
        Device Class: Miscellaneous,
        HCI Ver: 1.1 (0x1) HCI Rev: 0x1f9 LMP Ver: 1.1 (0x1) LMP Subver: 0x1f9
        Manufacturer: Cambridge Silicon Radio (10)
```

D.4. Como el teléfono solo soporta claves numéricas, mientras la clave de default de Bluez es "Bluez", se debe de modificar el archivo: **/etc/Bluetooth/givenpin**, con la clave deseada de la siguiente manera:

```
##############givepin##########
#!/bin/sh
echo "PIN:123456"
#################
```

D.5 Los servicios completos ofrecidos por el teléfono aparecen en la siguiente pantalla:

```
linux:~ # sdptool browse 00:0A:D9:77:6F:8B
Browsing 00:0A:D9:77:6F:8B ...
Service Name: Dial-up Networking
Service RecHandle: 0x10000
Service Class ID List:
  "Dialup Networking" (0x1103)
  "Generic Networking" (0x1201)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 1
Profile Descriptor List:
  "Dialup Networking" (0x1103)
    Version: 0x0100

Service Name: Voice gateway
Service RecHandle: 0x10002
Service Class ID List:
  "Headset Audio Gateway" (0x1112)
  "Generic Audio" (0x1203)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 3
Profile Descriptor List:
  "Headset" (0x1108)
    Version: 0x0100

Service Name: Serial Port 1
Service RecHandle: 0x10003
Service Class ID List:
  "Serial Port" (0x1101)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 4

Service Name: Serial Port 2
Service RecHandle: 0x10004
Service Class ID List:
  "Serial Port" (0x1101)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 5

Service Name: OBEX Object Push
Service RecHandle: 0x10005
Service Class ID List:
  "OBEX Object Push" (0x1105)
Protocol Descriptor List:
  "L2CAP" (0x0100)
```

```
  "RFCOMM" (0x0003)
    Channel: 10
  "OBEX" (0x0008)
Profile Descriptor List:
  "OBEX Object Push" (0x1105)
    Version: 0x0100

Service Name: IrMC Synchronization
Service RecHandle: 0x10006
Service Class ID List:
  "IrMCSync" (0x1104)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 11
  "OBEX" (0x0008)
Profile Descriptor List:
  "IrMCSync" (0x1104)
    Version: 0x0100

Service Name: HF Voice gateway
Service RecHandle: 0x10007
Service Class ID List:
  "" (0x111f)
  "Generic Audio" (0x1203)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 6
Profile Descriptor List:
  "" (0x111e)
    Version: 0x0100

Service Name: OBEX Basic Imaging
Service RecHandle: 0x1000b
Service Class ID List:
  "" (0x111b)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 15
  "OBEX" (0x0008)
Profile Descriptor List:
  "" (0x111a)
    Version: 0x0100

Service Name: OBEX File Transfer
Service RecHandle: 0x1000f
Service Class ID List:
  "OBEX File Transfer" (0x1106)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 7
  "OBEX" (0x0008)
Profile Descriptor List:
```

"OBEX File Transfer" (0x1106)
Version: 0x0100

Aquí se pueden observar todos los servicios que ofrece el teléfono, con la descripción de protocolos y el canal que ocupa. El teléfono T610 ofrece los siguientes servicios: Dial-up Networking en le canal 1, Voice gateway en el canal 3, Serial Port 1 en el canal 4, Serial Port 2 en el canal 5, OBEX Object Push en el canal 10, IrMC Synchonization en el canal 11, HF Voice gateway en el canal 6, OBEX Basic Imaging en el canal 15 y OBEX File Transfer en el canal 7.

D.6. Los resultados completos de las mediciones del RSSI son:

Para espacio abierto sin obstáculos:

A 0 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 13
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 27
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 28
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 27
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 28
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 27
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 27
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 30
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 30
```

a 1 mt:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 2
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
```

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 1
```

a 2 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -1
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
```

a 3mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ #
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -1
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -1
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -1
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -1
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -2
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
```

a 4 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -7
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -8
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -7
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -3
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -3
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -5
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -5
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -3
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -2
```

a 5mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -8
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -6
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -4
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -1
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -1
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -3
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -6
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -7
```

a 6mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -3
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -3
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -4
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -5
```

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -6
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -3
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -1
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -3
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -4
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -3
```

a 7 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -8
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -5
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -5
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -4
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -6
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -7
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -6
```

a 8 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
```

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -8
```

a 9 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -8
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -7
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -6
```

a 10mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
```

Para espacio cerrado con obstáculos:

Para 1 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
```

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
```

a 2 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -1
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -1
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
```

a 3 mts:

```
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: 0
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -3
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -3
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -4
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -3
```

linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -2

linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -1
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -1
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -2

a 4 mts:

linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -8
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -8
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -7
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -5
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -5
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -6

a 5mts:

linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10

a 6 mts:

linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -9
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10

a 7 mts:

linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10
linux:~ # hcitool rssi 00:0A:D9:77:6F:8B
RSSI return value: -10

REFERENCIAS

Referencias del capítulo 2.

- Universidad de Ovideo, Departamento de Ingeniería eléctrica, electrónica, de computación y de sistemas. *"Tema 10: REDES INALÁMBRICAS"*. Disponible en Internet: **www.dieecs.uniovi.es/**

- 3COM. *"Cuaderno tecnológico 3com. Soluciones inalámbricas"*. Disponible en Internet: **www.3Com/parters/wireless_sales_guide**

- The Infrared Data Association. Sitio en Internet: **www.irda.org/**

- 3COM. *"Deploying 802.11 Wireless LANs"*. Disponible en Internet: **www.3Com.com/wireless**

- Microsoft Corporation. *"Tecnologías para redes LAN inalámbricas y Windows XP"*. Disponible en Internet: **www.microsoft.com/windowsxp**

- Zona Bluetooth. Sitio en Internet: **www.zonabluetooth.com/que_es_bluetooth.htm**

- Special Interest Group (SIG). Sitio en Internet: **www.bluetooth.org**

- Hewlett-Packard Company. *"Bluetooth Technology Overview"*. Disponible en Internet: **www1.hp.com/products/wireless/ wpan/files/**

- Domotica.Net. *"IrDA versus Bluetooth"*. Disponible en Internet: **http://www.domotica.net/IrDA_versus_Bluetooth.htm**

Referencias del capítulo 3.

- Special Interest Group (SIG). "*Specification of the Bluetooth System. Volúmen 1: Core*". Disponible en Internet: **www.bluetooth.org**

- Special Interest Group (SIG). *"Specification of the Bluetooth System. Volúmen 2: Profiles"*. Disponible en Internet: **www.bluetooth.org**

- Palowireless. *"Bluetooth Tutorial"*. Disponible en Internet: **www.palowireless.com/infotooth/tutorial.asp**

- Dowhuszko, Ramos Silveyra, Vera. "*Solución integral para el desarrollo de aplicaciones sobre tecnología Bluetooth*". Disponible en Internet: **http://lcd.efn.unc.edu.ar/lab/adowhuszko/web/**

- Rodríguez, Oscar. "Implementación de una red inalámbrica Bluetooth". Disponible en Internet: **http://eiee.univalle.edu.co/~telecomunicaciones/**

Referencias del capítulo 4

- Tecnología del PC. "*2.5.3. Puertos USB*". Disponible en Internet: **zator.com/Hardware/H2_5_3.htm**
- Universal Serial Bus (USB) (technical): Disponible en Internet: **www.hardwarebook.net/connector/bus/usb_tech.html**
- Tecnología del PC. "*2.5.1.1 La UART*". Disponible en Internet: **www.zator.com/Hardware/H2_5_1_1.htm**
- How Stuff works. "*How serial ports works*". Disponible en Internet: **http://computer.howstuffworks.com/serial-port1.htm**
- Serial and UART Tutorial. Disponible en Internet: **www.freebsd.org/doc/en_US.ISO8859-1/articles/serial-uart/**

Referencias del capítulo 5

- Bluetooth designer. Sitio en Internet: **www.btdesigner.com**
- TAIYO YUDEN. "*Hoja de Especificaciones del módulo **EYMF2CSMM***" Disponible en Internet: **www.yuden.co.jp/bluetooth/module.html**
- Dowhuszko, Ramos Silveyra, Vera. "Solución integral para el desarrollo de aplicaciones sobre tecnología Bluetooth". Disponible en Internet: **http://lcd.efn.unc.edu.ar/lab/adowhuszko/web/**
- MEZOE. Sitio en Internet: **www.mezoe.com**
- ERICSSON MOBILE COMMUNICATIONS. Sitio en Internet: **www.ericsson.com/bluetooth/**
- AXIS. Sitio en Internet: **http://sourceforge.net/projects/openbt/**
- AFFIX. Sitio en Internet: **http://affix.sourceforge.net**
- BlueZ. "*How to*". Disponible en Internet: **http://bluez.sourceforge.net**
- Sony Ericsson. "*Specification*". Disponible en Internet: **www.sonyericsson.com**